MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÃO
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

sid.inpe.br/mtc-m21b/2015/05.19.01.59-TDI

ESTUDO DOS EFEITOS DA SINCRONIZAÇÃO SOBRE O TRANSITÓRIO E A ESTABILIDADE DE SISTEMAS DE CONTROLE POR REDE

Eloy Martins de Oliveira Junior

Tese de Doutorado do Curso de Pós-Graduação em Engenharia e Tecnologia Espaciais/Mecânica Espacial e Controle, orientada pelo Dr. Marcelo Lopes de Oliveira e Souza, aprovada em 27 de maio de 2015.

URL do documento original:
<http://urlib.net/8JMKD3MGP3W34P/3JGJC7E>

INPE
São José dos Campos
2015

PUBLICADO POR:

Instituto Nacional de Pesquisas Espaciais - INPE
Gabinete do Diretor (GB)
Serviço de Informação e Documentação (SID)
Caixa Postal 515 - CEP 12.245-970
São José dos Campos - SP - Brasil
Tel.:(012) 3208-6923/6921
Fax: (012) 3208-6919
E-mail: pubtc@sid.inpe.br

MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÃO
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

sid.inpe.br/mtc-m21b/2015/05.19.01.59-TDI

ESTUDO DOS EFEITOS DA SINCRONIZAÇÃO SOBRE O TRANSITÓRIO E A ESTABILIDADE DE SISTEMAS DE CONTROLE POR REDE

Eloy Martins de Oliveira Junior

Tese de Doutorado do Curso de Pós-Graduação em Engenharia e Tecnologia Espaciais/Mecânica Espacial e Controle, orientada pelo Dr. Marcelo Lopes de Oliveira e Souza, aprovada em 27 de maio de 2015.

URL do documento original:
<http://urlib.net/8JMKD3MGP3W34P/3JGJC7E>

INPE
São José dos Campos
2015

Dados Internacionais de Catalogação na Publicação (CIP)

Oliveira Junior, Eloy Martins de.

Ol4e Estudo dos efeitos da sincronização sobre o transitório e a estabilidade de sistemas de controle por rede / Eloy Martins de Oliveira Junior. – São José dos Campos : INPE, 2015.

xl + 427 p. ; (sid.inpe.br/mtc-m21b/2015/05.19.01.59-TDI)

Tese (Doutorado em Engenharia e Tecnologia Espaciais/Mecânica Espacial e Controle) – Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2015.

Orientador : Dr. Marcelo Lopes de Oliveira e Souza.

1. Sincronização de relógios. 2. Sistemas de controle por rede. 3. IEEE 1588. 4. Welch-lynch. 5. Sistemas de tempo real. I.Título.

CDU 681.503

Aprovado (a) pela Banca Examinadora em cumprimento ao requisito exigido para obtenção do Título de *Doutor(a)* em

Engenharia e Tecnologia Espaciais/Mecânica Espacial e Controle

Dr. Mario Cesar Ricci

Presidente / INPE / SJCampos - SP

Dr. Marcelo Lopes de Oliveira e Souza

Orientador(a) / INPE / SJCampos - SP

Dr. Paulo Giácomo Milani

Membro da Banca / INPE / SJCampos - SP

Dr. Luiz Carlos Gadelha de Souza

Membro da Banca / UNB / Brasília - DF

Dr. Fernando José de Oliveira Moreira

Convidado(a) / EMBRAER / SJCampos - SP

Dr. Rolf Henry Vargas Valdivia

Convidado(a) / Friuli / São José dos Campos - SP

Este trabalho foi aprovado por:

( ) maioria simples

(X) unanimidade

Título: ***"Estudo dos efeitos da sincronização sobre o transitório e a estabilidade de sistemas de controle por rede".***

Aluno (a): ***Eloy Martins de Oliveira Junior***

São José dos Campos, 27 de Maio de 2015

“Mesmo que já tenha feito uma longa caminhada, sempre haverá mais um caminho a percorrer”.

Santo Agostinho

"Acreditar em sincronicidades não é nada difícil. Elas acontecem o tempo todo. Difícil é saber o que significam".

Ronaud Pereira

A Deus, Nossa Senhora e minha família.

AGRADECIMENTOS

Chegou o momento de expressar sinceros agradecimentos a familiares e amigos, tanto aos velhos e queridos, quanto aos que se revelaram ao longo destes anos como uma verdadeira fonte de conhecimento e sabedoria. Corro o risco de não dar conta de todos eles, como é merecido, porque é difícil lembrar-se de todos os instantes, de todas as energias e impulsos emanados, de toda a solidariedade e conhecimento com que fui brindado. Não foi uma simples caminhada, foi, antes de tudo, um esplêndido voo em busca de uma percepção mais precisa dos movimentos do universo, do tempo, do espaço, da sincronia matemática que há em todas as coisas. Pareceu-me um voo sem fim, com intercorrências pessoais de toda ordem que atropelavam o raciocínio e a vontade. Percalços que, ao invés de tolherem meus movimentos, libertaram-me de certos conceitos; que ao invés de me deterem deram-me força para um impulso mais pragmático. O desafio foi grande, mas as motivações foram maiores ainda. A transformação de instantes sofridos e angustiosos, ao bater numa muralha intransponível, quando não encontrava a solução para determinados problemas, teve a participação importante de generosos apoios para transformar esses momentos em esperança e, na sequência, vitória.

Esta tese é o resultado mais visível desse processo doloroso de construção em meio a um furacão de demonstrações de confiança, fé, carinho e amizade. Aos que fizeram parte direta ou indiretamente ou, ainda, pelo fato de simplesmente existirem em minha vida nesse período, meu mais profundo agradecimento.

A minha família: minha mãe, Rosy; minhas irmãs, Erika e Fabiana; minha avó querida, Rosa; minha adorada sobrinha Isabella, minha madrinha Jane, meus tios Luiz e Gorete que depositaram toda a sua confiança em mim, agradeço de joelhos a presença de vocês na minha vida.

Ao meu amor Rita Helena que, com tanto amor e carinho, me conduz aos caminhos da harmonia e compreensão, com muito amor e carinho o meu agradecimento.

Ao professor Marcelo L. O. Souza, na qualidade de amigo e orientador, os tantos produtivos e inesquecíveis diálogos, a confiança e o socorro nos momentos de dificuldade. Sou inteiramente grato pela orientação que ultrapassou os caminhos da tese. Agradeço, sobretudo, o privilégio de trabalhar em um tema absolutamente antigo, atual e inovador com sua indispensável colaboração na discussão deste trabalho, dosando críticas com comentários de incentivo. Aos professores Dr. Mario Ricci, Dr. Paulo Milani, Dr. Luiz C. Gadelha, Dr. Fernando Moreira e Dr. Rolf Valdivia, meus agradecimentos pela generosidade em todas as fases do projeto, bem como pelo cuidado e estímulo na discussão do projeto, bem como por seus questionamentos e contribuições na etapa da qualificação. Sou profundamente grato à Dra. Nadjara Santos que sempre me apoiou incontestavelmente, meu amoroso agradecimento; e aos amigos: Jairo Amaral, Jairo Siqueira, Alain, Wagner, Gitsuzo, Willer, Chicão, Lorena, Pierre, Fabio, Adolfo, Rafão, Ximena, Valdirene, Catito, Queila, pela indescritível solidariedade e amizade que se traduziram sempre em singulares e cúmplices conversas, numa busca entusiasmada das respostas às perguntas que me escapavam. Aos colegas de trabalho: Guilherme, Salvador, João, Thiago, Marisa, Philipe, Deivid, Sidney, Everaldo, que me apoiaram na conclusão deste trabalho. Aos meus amigos-irmãos: Luiz Gustavo, Lucas, Tulio, Guima, Binho, Mairon em que, mesmo distante, o apoio e amizade sempre estiveram presentes. Agradeço também ao INPE, ao departamento de Mecânica Espacial e Controle, à CAPES e ao CNPQ por todo o suporte durante o curso de doutorado. Há muito mais a quem agradecer. A todos aqueles que, embora, não nomeados, me fortaleceram com seus inestimáveis apoios em distintos momentos e por suas presenças, o meu reconhecido muito obrigado!

Mais uma vez e sempre, agradeço a Deus, porque é a luz, fortaleza, proteção e sabedoria que dá sentido a minha vida.

"*Tudo tem o seu tempo determinado, e há tempo para todo o propósito debaixo do céu*". Eclesiastes 3:1

RESUMO

Com o crescimento da complexidade e integração dos sistemas, a sincronização torna-se cada vez mais importante, pois a de-sincronização em tempo ou em fase pode causar falhas graves ou até catastróficas. Particularmente, a sincronização de tempo, ou de fase, é um aspecto muito importante para se atingir alto desempenho, confiabilidade e determinismo em sistemas de controle por rede. A implementação distribuída, a integração de sistemas, a complexidade e o aumento do requisito de confiabilidade tornam diversos problemas que antes não eram importantes, agora relevantes; demandando o desenvolvimento de novas tecnologias, técnicas e algoritmos. Dentre estes problemas está à sincronização de tempo entre diversas entidades de uma implementação distribuída e como estas soluções influenciam o desempenho de sistemas de controle por rede. Esta tese estuda os efeitos da sincronização de relógios sobre o transitório e a estabilidade de sistemas de controle por rede, principalmente no âmbito de sistemas com ciclo de vida extenso tal como o espacial. O trabalho propõe-se um conjunto de algoritmos reconfiguráveis e suavizáveis de sincronização com métrica mista (Precisão, Exatidão e Controle) corrigindo os vieses iniciais, os vieses instantâneos causados pela deriva dos relógios locais e também a deriva dos relógios lógicos, utilizando técnicas de amortização das descontinuidades de tempo durante o ajuste dos relógios. Ainda propõe a aplicação da técnica "*cross-fading"* para amortizar a descontinuidade de tempo dos algoritmos de sincronização de relógios, analisando o uso de três tipos de funções amortizadoras (degrau, rampa e exponencial). Também analisa-se e valida-se o conjunto de algoritmos (ReS, PReS, AReS, PAReS) de sincronização de relógios sobre um sistema de controle por redes comparando os resultados com os algoritmos FTM, PTP e SR, abordando o problema através de teoria, análise, controle e simulação. Além disso, propõe um modelo em espaço de estados de sincronização de relógios que faz a ponte entre o mundo do controle e o mundo da sincronização de relógios. O conjunto de algoritmos ReS melhorou o desempenho tanto da sincronização quanto do sistema de controle. As métricas mistas de precisão, exatidão e controle sobre um algoritmo de reconfiguração e suavizável demonstraram ser uma abordagem a ser seguida no projeto de algoritmos de sincronização de relógios para sistemas de controle por redes. A verificação e validação sobre dos algoritmos propostos no NCS foram feitas através de simulações desenvolvidas no ambiente Matlab/Simulink com ajuda da ferramenta TrueTime.

STUDY OF THE EFFECTS OF THE SINCRONIZATION ON THE TRANSIENT AND STABILITY OF NETWORKED CONTROL SYSTEMS

ABSTRACT

With the growing complexity and integration of systems, synchronization becomes increasingly important, as the time or phase de-synchronization can cause serious or even catastrophic failures. Time or phase synchronization is a very important aspect to achieve high performance, reliability and determinism in networked control systems. Therefore, this work studies the effects of synchronization on the transient and stability of networked control systems. The distributed approach the systems integration, the complexity and the increase in reliability turns relevant several problems as the time synchronization problem of distributed systems and their effects on the performance of networked control systems, requiring the development of new techniques and algorithms. This thesis studies the effects of the synchronization on the transient and stability of networked control systems, mainly in the framework of systems with extended life cycle such as space systems. The work proposes a set of reconfigurable and smoothable clock synchronization algorithms with multiple metrics (Precision, Accuracy and Control) correcting the initial offset, offsets caused by the drift of local clocks and also the drift of the logical clocks, using amortization techniques to time discontinuities problem during the clock adjustments. The cross-fading technique is proposed to amortize the time discontinuity of clock synchronization algorithms, using at least three types amortized functions (step, ramp and exponential). Further, this thesis analyzes and validates the set of clock synchronization algorithms (ReS, PReS, AReS, PAReS) on a networked control system comparing the results with the FTM, PTP and SR clock synchronization algorithms, by means of theory, analysis, control and simulation. In addition, it is proposed a first clock synchronization model in state space to bridge the gap between the control world and clock synchronization world. The set of ReS algorithms improves the performance of both synchronization and control system. Multiple metrics as precision, accuracy and control on a reconfigurable and smoothable algorithm proved itself to be a good approach to be used in the design of clock synchronization algorithms for network control systems. The verification and validation of the proposed algorithms on the NCS were made through simulations developed in Matlab/Simulink environment with the help of TrueTime toolbox.

LISTA DE FIGURAS

Pág.

LISTA DE TABELAS

Pág.

LISTA DE SIGLAS E ABREVIATURAS

INPE	Instituto Nacional de Pesquisas Espaciais
SID	Serviço de Informação e Documentação
TDI	Teses e Dissertações Internas
SPG	Serviço de Pós-Graduação
PTP	*Precision Time Protocol*
FTM	*Fault-Tolerant Mid-Point*
NCS	Sistema de Controle por Redes
VC	Relógio virtual
PC	Relógio físico
SR	*Algoritmo State-Rate Correction*
UTC	Tempo Universal Coordenado
TAI	Tempo Universal Atômico
GPS	Sistema de Posicionamento Global por Satélites
GNSS	Sistema Global de Navegação por Satélites
AReS	Algoritmo de Sincronização de Relógios Reconfigurável, Suavizável por Métrica de Exatidão
PReS	Algoritmo de Sincronização de Relógios Reconfigurável, Suavizável por métrica de Precisão
PAReS	Algoritmo de Sincronização de Relógios Reconfigurável, Suavizável por métrica de Precisão e Exatidão
ReS	Algoritmo de Sincronização de Relógios Reconfigurável, Suavizável
STM	Modo de Inicialização
NOMO	Modo Nominal de Correção de Viés Instantâneo
NOMD	Modo Nominal de Correção de Deriva
CFR	Função *Cross-Fading* Rampa
CFD	Função *Cross-Fading* Degrau
CFE	Função *Cross-Fading* Exponencial
PWM	Modulação por Largura de Pulso

TDMA	*Time Division Multiple Access*
CSMA/CD	*Carrier Sense Multiple Access / Colision Detection* - Ethernet
FTA	*Fault-Tolerant Average*
MMCF	Fator de Conversão *Microtick - Macrotick*
GMT	Tempo Médio de Greenwich
CC	Corrente Contínua
GLONASS	*Globalnaya Navigatsionnaya Sputnikovaya Sistema*
Compass	*Chinese Satellite Navigation*
CPU	Unidade processamento computacional
BIPM	Escritório Internacional de Pesos e Medidas na França

LISTA DE SÍMBOLOS

μT	Microtick
mT	Macrotick
MiT(z)	Função microtick no domínio Z
MaT(z)	Função macrotick no domínio Z
μT(k)	Microtick no instante k
mT(k)	Macrotick no instante k
μT(t)	Microtick no instante t
mT(t)	Macrotick no instante t
$\mu T_i(k)$	Microtick do relógio i no instante k
$mT_i(k)$	Macrotick do relógio i no instante k
ΔμT(k)	Diferença entre os microticks de dois relógios no instante k
∆mT(k)	Diferença entre os macroticks de dois relógios no instante k
c(t)	Equação do relógio no domínio contínuo
c(k)	Equação do relógio no domínio discreto
k	Instante da amostragem
T	Período de amostragem
G	Ganho
M	Chaveamento
R	Período de re-sincronização
corr(t)	Correção dos valores no domínio contínuo calculado por um algoritmo.
corr(k)	Correção dos valores no domínio discreto calculado por um algoritmo.
$corr_i(k)$	Correção dos valores no domínio discreto no instante k calculado por um algoritmo do relógio i.
$corr_r(k)$	Correção dos valores no domínio discreto no instante k calculado por um algoritmo do relógio de referência.
f(t)	Função "cross-fading" no domínio contínuo
f(k)	Função "cross-fading" no domínio discreto
ηG	Métrica que considera o valor zero
Π	Precisão

f	Falha bizantina
n	Número de relógios ou nós no sistema de sincronização
Γ	Máximo viés instantâneo
VC(t)	Relógio virtual
PC(t)	Relógio físico
ADJ	Variável local que armazena o valor da correção de tempo
PTP	Algoritmo *Precision Time Protocol*
FTM	Algoritmo *Fault Tolerant Mid-Point*
SR	Algoritmo *State-Rate Correction*
g_0	Sinal de oscilação local
n_u	Divisor de frequências do microtick
n_m	Divisor de frequências do macrotick
g_l	Sinal de tempo de operações de baixo nível
g_u	Sinal de tempo do microtick
g_l	Sinal de tempo do macrotick
g^i	Resolução do microtick
ρ	Taxa de deriva
ρ(k)	Taxa de deriva no instante k
ρ_i	Taxa de deriva do relógio i
$\rho_i(k)$	Taxa de deriva do relógio i no instante k
P(k)	Taxa de deriva do macrotick no instante k
R_{int}	Intervalo de re-sincronização
Φ	Valor de tempo da Função de convergência imediatamente após a re-sincronização
δ	Máximo valor do atraso de rede
ε	Valor do atraso de rede
$C(t_1)$	Valor de tempo do relógio C no tempo t_1
$C(t_2)$	Valor de tempo do relógio C no tempo t_2
∆C	Matriz de diferenças de tempo
A	A é formado pelos valores da linha do nó da matriz ∆C
corr(A)	função de ponto-médio tolerante a falhas bizantinas

ξ	Flutuação de tempo
u(n,f)	Custo em relação à precisão da sincronização
$MMCF_i(k)$	Fator de conversão microtick - macrotick do relógio i no instante k
$MMCF_z(k)$	Fator de conversão microtick - macrotick do relógio de referência escolhido no instante k
$MMCF_r(k)$	Fator de conversão microtick - macrotick do relógio de referência no instante k
UTC	Tempo Universal Coordenado
TAI	Tempo Atômico Internacional
GMT	Tempo Médio de Greenwich
b	Valor de viés inicial
Ш	Trem de impulsos
PWM	*Pulse Width Modulation*
A	Matriz de estado (dimensão nxn);
B	Matriz de controle (dimensão nxp)
C	Matriz de saída (dimensão qxn)
D	Matriz de transmissão direta (dimensão qxp)
x(.)	Vetor de estados (n)
y(.)	Vetor de saída (dimensão q)
u(.)	Vetor de controle (dimensão p)
α	Inverso da taxa de deriva do macrotick no instante k
β	Período de amostragem dividido pelo MMCF do relógio, no modelo por estados
Ts	Instante de re-sincronização atual
β	fator de re-sincronização de deriva, nos algoritmos
tr_1	estampa de tempo do relógio de referência no instante de envio do broadcast;
ts_2	estampa de tempo do relógio escravo no instante que recebe o valor de tempo do relógio de referência
ts_3	estampa de tempo do relógio escravo no instante de envio da requisição de atraso de tempo

tr_4 estampa de tempo do relógio de referência no instante de recebimento da requisição de atraso de tempo

SUMÁRIO

Pág.

1 INTRODUÇÃO

1.1. Contexto

Atualmente, sistemas como satélites artificiais, aviões, estações espaciais, controles de tráfego aéreo, terrestre, ferroviário, fazendas de energia eólica, usinas nucleares, controles industriais, e outros estão cada vez mais complexos e/ou altamente integrados. Sistemas como estes trabalham cada vez mais integrando a computação e a comunicação com o controle de sistemas em diferentes níveis de operação e operando em tempo real. Nestes casos, a tradicional arquitetura ponto a ponto (isto é, onde existe um cabo que conecta o computador central com cada sensor e/ou atuador) não é mais satisfatória. Assim, o tradicional sistema não é mais adequado para satisfazer os novos requisitos e especificações tais como redução do número de fios e seus inconvenientes. Assim, estes sistemas são desenvolvidos em uma arquitetura distribuída, pois além de outras facilidades, permite a composição de subsistemas. O uso de subsistemas tem várias vantagens tais como a modularidade; no entanto as características de um subsistema não devem ser alteradas quando este for parte de um sistema maior.

De acordo com Kopetz (1992) o desempenho do sistema distribuído em tempo real deve ser previsível e determinístico. Previsível no sentido de que todo o sistema deve ser completamente especificado no domínio lógico (valores e transições) e determinístico no sentido de que todo o sistema deve ser especificado também no domínio temporal (instantes, intervalos e tempo).

Dentre um vasto repertório e possibilidades, uma das arquiteturas distribuídas que vem sendo muito utilizada em sistemas de controle é o controle por rede (NCS - *Networked Control System*). Segundo Zhang, Branicky E Phillips (2001) e Valdivia (2009) são considerados um sistema de controle por rede quando a malha de controle é fechada através de uma rede de comunicação, como mostra a Figura 1.1.

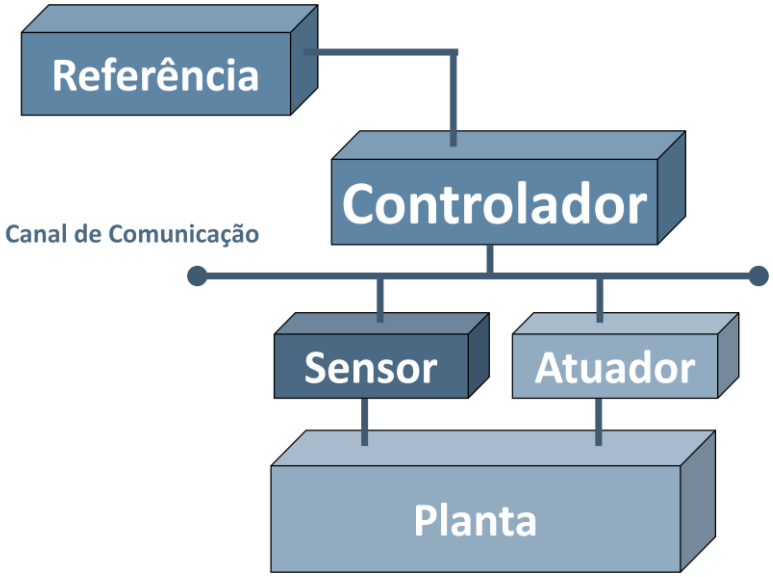


Figura 1.1 - Sistema de Controle por Rede - NCS.

O modelo NCS apresentado na Figura 1.1, percebe-se o uso de um canal de comunicação. Este canal de comunicação, também conhecido como barramento de comunicação ou rede de comunicação, é a parte do sistema responsável por estabelecer a comunicação entre os nós do sistema de controle, tais como sensores, atuadores e controladores.

De acordo com Martí, Lozoya E Fuertes (2008), a rede de comunicação é uma parte fundamental para suportar todas as interações existentes dos nós do sistema de controle. A proposta da rede de comunicação é entregar as mensagens de comunicação com confiabilidade, segurança, eficiência e temporalmente correta. Existem vários protocolos de comunicação disponíveis no mercado que realizam estas tarefas de maneiras, métodos e formas diferentes.

De acordo com Kopetz (1992), nos sistemas distribuídos síncronos (dependentes do tempo), uma ligeira variação do tempo ou a ocorrência de falhas em diferentes nós pode levar o sistema a estados indesejados, o que pode causar transitórios e instabilidades e até falências não previstas pelo

sistema. Por isso, a sincronização de tempo e/ou fase em diversos níveis torna-se um aspecto importante do projeto de sistemas. Ou seja, em sistemas de controle por rede é necessário alcançar os requisitos temporais do sistema com alto desempenho, confiabilidade e segurança.

Problemas relacionados ao tempo, tal como a medição, datação e sincronização são tão antigos quanto à humanidade, como mostra cronologia da Figura 1.2.

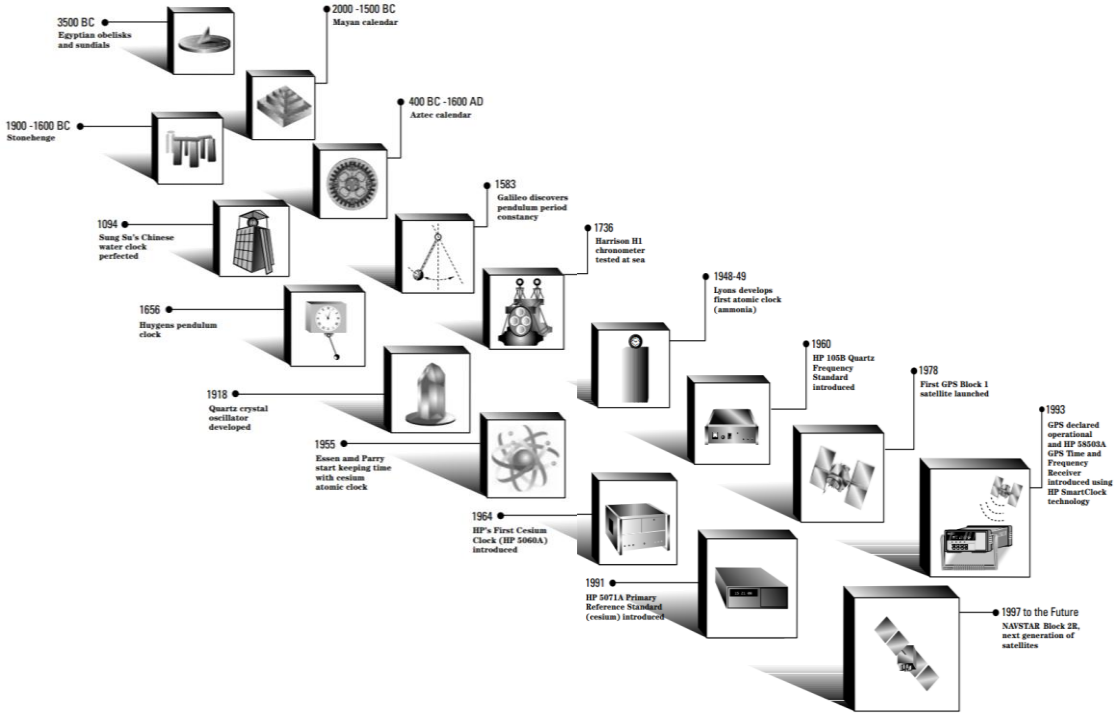


Figura 1.2 - Cronologia da Ciência Relacionada à Medição do Tempo até 1997.

Fonte: David, Neil e Clifford (1997)

Este trabalho tem o objetivo de estudar problemas relacionados à sincronização de relógios. Apesar de não considerar os problemas relacionados à medição e datação, os mesmos vão aparecer direta e indiretamente no problema de sincronização, como será observado mais à frente.

O problema da sincronização por si só é um problema que contempla vários níveis e complexidades. Para facilitar o entendimento do problema de sincronização é necessário trabalhar em camadas de abstração. Com isso, é possível abordar diversos problemas ignorando detalhes desnecessários e focando em características essenciais. A Figura 1.3 mostra o modelo de três camadas de abstração, utilizado neste trabalho.

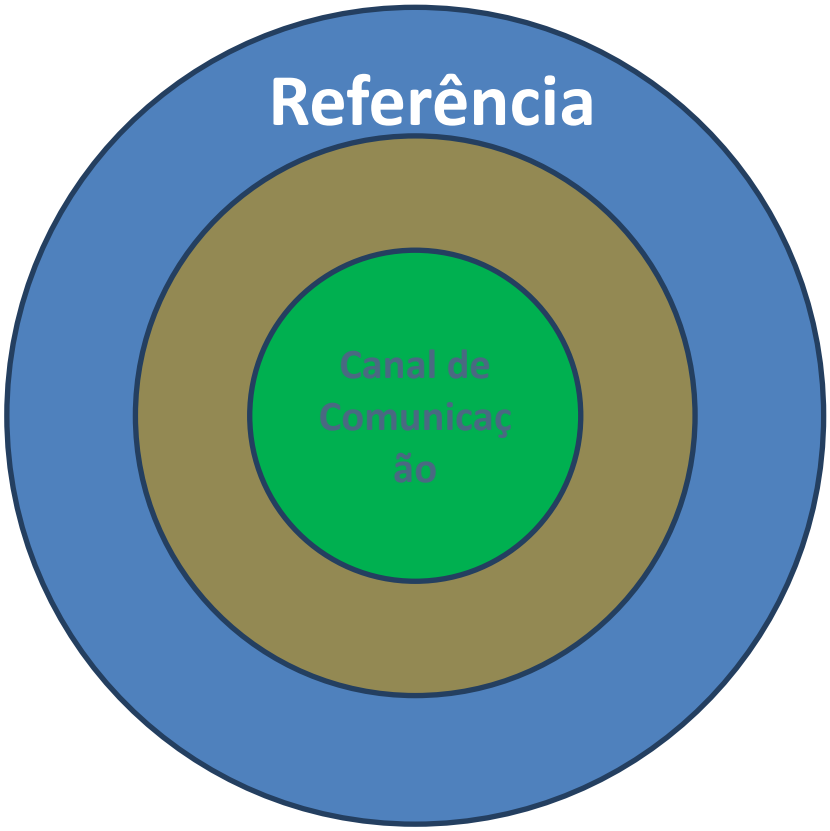


Figura 1.3 - Camadas de Abstração da Temporização.

Como mostra a Figura 1.3, as três camadas de abstração da temporização são:

a) **Física**: é a camada que representa os materiais e suas propriedades como por exemplo o cristal de quartzo, necessários para a geração da oscilação;

b) **Hardware**: é a camada que representa os mecanismos de sensoriamento e transmissão da oscilação gerada pela camada física, sendo a interface entre a parte física e de software;

c) **Software**: é a camada de abstração de alto nível que reflete os mecanismos de sensoriamento e transmissão da oscilação gerada, que pode ser controlada via software.

Ao longo dos anos, muitas soluções e algoritmos de sincronização vêem sucessivamente sendo desenvolvidos para as diferentes camadas de abstração apresentadas. Mas basicamente, sincronizar em tempo e/ou fase é fazer com que todos os nós da malha, a ser sincronizada, estejam em comum acordo com um relógio mestre real ou virtual. Em termos de controle, todos os nós devem fazer o rastreio da referência (real ou virtual).

Diversos problemas relacionados à sincronização de tempo e/ou fase nas três camadas de abstração e suas influências no transitório e estabilidade de sistemas de controle por rede ainda não foram explicados e/ou solucionados. Entre esses problemas, está a falta de uma melhor teoria, abordagem e/ou modelagem para possibilitar o uso da teoria de controle para análise, projeto e síntese de soluções/algoritmos de sincronização de relógios para sistemas de controle por rede. Sendo o espaço (âmbito de interesse deste trabalho) um ambiente de extrema agressividade devido à radiação, variações bruscas de temperatura e outros fenômenos físicos; além de que, sistemas espaciais normalmente possuem um ciclo de vida grande (meses e anos) muito maior que sistemas terrestres, a necessidade de soluções/algoritmos mais robustas e/ou ótimas para sincronização de relógios para sistemas de controle por redes, torna-se um importante aspecto para a concepção destes problemas que por consequência torna a teoria de controle uma abordagem interessante a ser explorada na concepção de algoritmos de sincronização de relógios.

1.2. Motivação e Justificativa

De acordo com Meyr e Ascheid (1990), existe uma impressão dos projetistas de que a tarefa de sincronizar tempo ou fase é uma tarefa trivial para os sistemas. No entanto, isto é uma concepção errada. A tarefa de sincronizar tempo ou fase, mesmo no passado, nunca foi uma tarefa trivial. Além disso, a tarefa de sincronização vem se tornando cada vez mais complexa e crítica dado que, nos últimos anos, de acordo com Hiesh e Hung (1996) e Oklobdzija, Stojanovic, *et al.* (2003), os requisitos de velocidade de relógio, alto desempenho, alta integração e baixo consumo de energia vêm crescendo rapidamente. Além disso, a implementação distribuída, a integração de sistemas, a complexidade e o aumento da confiabilidade fazem com que diversos problemas que antes não eram relevantes se tornem importantes e demandem o desenvolvimento de novas tecnologias. Entre estes problemas está à sincronização de tempo ou fase por métricas mistas e como estas soluções influenciam o desempenho de sistemas de controle por rede.

Diversas soluções de sincronização de tempo e fase vêm sendo desenvolvidas ao longo dos anos, para as diferentes camadas de abstração.

Na camada física podemos citar as diferentes soluções e alternativas para os materiais que geram a oscilação, como os osciladores de quartzo e atômicos, visando minimizar a deriva de tempo.

Na camada de hardware, podemos citar as diferentes técnicas de sincronização de fase em relação a uma referência. Entre estas técnicas, pode-se citar a técnica PLL (*Phase Locked Loop*) como uma das técnicas mais utilizadas na literatura. Existem vários artigos e livros que abordam este assunto. Entre eles podem-se citar alguns:

a) (LINDSEY, 1972);

b) (MEYR e ASCHEID, 1990);

c) (HIESH e HUNG, 1996);

d) (KIHARA; ESKELINEN; ONO, 2003).

Na camada de software, existem vários algoritmos, cada um com suas particularidades e suposições. Entre eles podem-se citar alguns:

a) Algoritmo Welch-Lynch (WELCH e LYNCH, 1988);

b) Algoritmo de Lamport (LAMPORT, 1978);

c) Algoritmo FTA (*Fault Tolerant Average*) (KOPETZ e OCHSENREITER, 1987);

d) Algoritmo de Cristian (CRISTIAN, 1989);

e) Padrão IEEE 1588 (EIDSON, 2006);

f) Algoritmo Ótimo de Sincronização de Relógios com restrição de energia (ATTIYA, HAY e WELCH, 2006);

g) Algoritmo *State-Correction* e *Rate-Correction* (HANZLIK; ADEMAJ e KOPETZ, 2006);

h) Algoritmo *Self-Stabilizing* (MALEKPOUR, 2006).

Entretanto, as várias soluções citadas acima seguem basicamente o mesmo esquema, diferenciadas entre si por diferentes premissas, condições iniciais, lógicas, equações de correção e se a métrica é baseada em um relógio mestre real ou virtual. A Figura 1.4 mostra um diagrama em blocos com o esquema mais básico que estas soluções seguem.

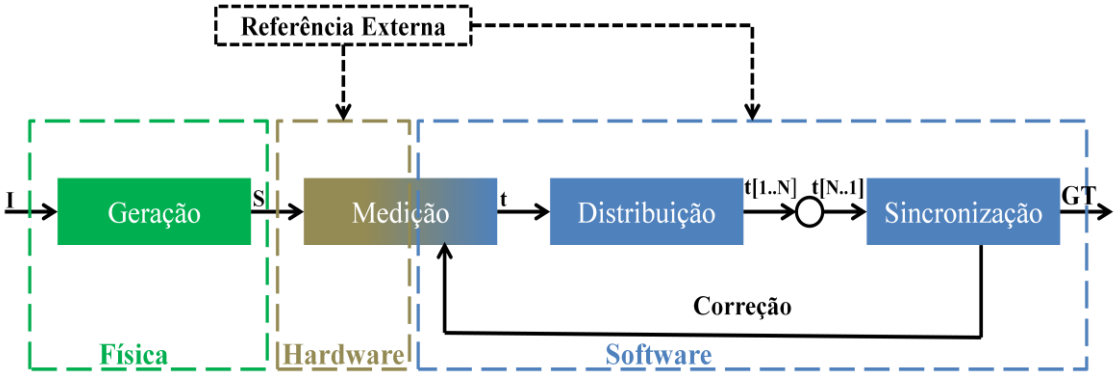


Figura 1.4 - Diagrama em Blocos de um Esquema Básico de Sincronização.

Da Figura 1.4 temos que:

a) I = entrada necessária para gerar a oscilação;

b) S = sinal gerado pela oscilação;

c) t = medida gerada pelo sensoriamento da oscilação denominado *tick* ;

d) t[1..n] = transmissão do *tick* de 1 para N blocos de sincronização;

e) t[N..1] = recepção de N *ticks* em 1 bloco de sincronização;

f) Correção = valor que corrige a medida t e estabiliza o sistema em um relógio virtual;

g) GT = Relógio virtual.

h) Referência Externa: Pode ser real ou virtual;

A parte de geração é relacionada à camada de abstração física. A parte de medição, relacionada à camada de abstração de hardware, é a interface entre o mundo físico e o mundo de software (lógico). No entanto, esta camada de hardware também pode receber uma referência de um relógio real externo, e por isso existe uma seta advinda de uma referência externa. A parte relacionada à camada de abstração de software mostra um esquema básico que a maioria dos algoritmos de sincronização de relógios segue: distribuir,

analisar e corrigir. Os algoritmos se diferenciam nas premissas, formas, análise e correção dos dados. Em alguns algoritmos, uma referência externa é necessária, pois o algoritmo considera como premissa que irá seguir uma referência externa (relógio mestre real). Por isto, existe uma seta da referência externa ligada a esta camada.

No entanto, os algoritmos de sincronização não cobrem parcialmente e/ou totalmente, alguns pontos, tais como: 1) pontos comuns de falha; 2) condições iniciais de operação dos relógios diferentes; 3) métricas mistas (sincronizar a precisão e exatidão simultaneamente), 4) restrições de energia; 5) operação em diferentes modos; entre outras impostas por diversos atributos diferentes.

Por isso, este trabalho se propôs a estudar os efeitos da sincronização sobre o transitório e a estabilidade de sistemas de controle por rede, propondo um conjunto de algoritmos de sincronização de tempo para serem aplicados a sistemas de controle por redes.

1.3. Objetivo

O objetivo desta tese é estudar os efeitos da sincronização de relógios sobre o transitório e a estabilidade de sistemas de controle por rede. Para isto serão adotados os seguintes passos:

a) Rever a literatura sobre algoritmos de sincronização de relógios e os efeitos da sincronização sobre o transitório e a estabilidade de sistemas de controle por rede;

b) Propor e desenvolver um novo algoritmo de sincronização de tempo, dentro da camada de software, dentro da precisão e condições supostas;

c) Validar o algoritmo de sincronização de tempo, dentro da camada de software, através de modelagem e simulação;

d) Verificar seus efeitos sobre o transitório e a estabilidade de sistemas de controle por rede através de modelagem e simulação.

1.4. Originalidade, Generalidade e Utilidade

É esperado que todo trabalho de doutorado tenha essas três características e que essas qualidades tenham que ser cumpridas com rigor. Referente a este trabalho, pode-se dizer que:

A **originalidade** vem da intenção de propor novos algoritmos de sincronização de relógios por métrica mista (Controle, Precisão e Exatidão), ou seja, aplicar os métodos de análise da teoria de controle para garantir o correto funcionamento do algoritmo de sincronização com baixo impacto sobre o sistema de controle por rede (métrica de controle) e também sem degradar as métricas de precisão e exatidão.

A **generalidade** vem do fato de que o algoritmo de sincronização de relógios proposto e métodos de análise usados são de propósitos geral que podem ser usados em diferentes algoritmos/arquiteturas de sincronização e sistemas de controle por rede, com o mínimo prejuízo de sua eficácia.

A **utilidade** vem do fato de que a sincronização de relógios em sistemas de controle por rede é uma tarefa necessária e está se tornando cada vez mais crítica devido à complexidade, integração dos sistemas e cada vez mais da necessidade de determinismo. Além disso, o ambiente espacial, devido a sua agressividade, tem muita influência sobre a sincronização de relógios.

1.5. Organização deste Trabalho

No capítulo 1 é apresentado o contexto, a motivação e os objetivos deste trabalho.

No capítulo 2 é apresentado a revisão bibliográfica e conceitos básicos necessários para o entendimento do problema. Na seção 2.1 e 2.2 são

apresentados uma revisão dos conceitos de controle e os protocolos de comunicação de interesse. Na seção 2.3 são apresentados conceitos da teoria de relógios, incluindo desde conceitos básicos como monotonicidade, métricas, imperfeições, arquiteturas até algoritmos de sincronização de relógios e padrões de tempo. Na seção 2.4, é feito uma revisão sobre a influência que a sincronização de relógios pode ter sobre sistemas de controle por redes. A seção 2.5 aborda as ferramentas de simulação utilizadas.

No capítulo 3 é descrita a formulação do problema e abordagens para a solução. A seção 3.1 aborda o problema relacionado aos algoritmos de sincronização de relógios que este trabalho se propôs a estudar. A seção 3.2 descreve-se o sistema em estudo. A seção 3.3 detalha as abordagens para solução.

No capítulo 4 são descritos os modelos matemáticos de um relógio. Os modelos propostos neste capítulo, é um modelo por equação de diferenças, um por transformada Z, um por espaço de estados e outros por PWM. Sendo que durante toda a tese, os modelos utilizados para modelagem, simulação e análise são os modelos de equação de diferenças e transformada Z.

No capítulo 5, na seção 5.1 é feito o estudo da descontinuidade de tempo e seus efeitos. A seção 5.2 descreve o modelo teórico, as técnicas de amortização e métricas do projeto que podem ser no desenvolvimento da amortização para minimizar o problema da descontinuidade de tempo em algoritmos de sincronização de relógios.

No capítulo 6 é desenvolvido um conjunto de algoritmos de sincronização de relógios reconfiguráveis e suavizáveis com métrica mista (Precisão, Exatidão e Controle)

O capítulo 7 contempla os casos de simulação do capitulo 5. Na seção 7.1 é feito um estudo dos efeitos da descontinuidade de tempo causado por algoritmos de sincronização de relógios sobre um sistema de controle por

redes. Na seção 7.2 são apresentadas as simulações da técnica de amortização por rampa, desenvolvida na seção 5.2, aplicada em algoritmos de sincronização. Na seção 7.3 são apresentadas as simulações da técnica de amortização por exponencial, desenvolvida na seção 5.2, aplicada em algoritmos de sincronização.

O capítulo 8 contempla os casos de simulação do capitulo 6. A seção 8.1 apresenta o projeto da simulação, demonstrando os conceitos necessários para entendimento das mesmas. A seção 8.2 traz uma tabela com o resumo de todos os casos de estudo. Da seção 8.3 até a seção 8.43 são apresentadas as simulações e resultados dos respectivos casos de estudo. A seção 8.44 detalha os indicadores de desempenho utilizados para analisar os resultados; e por fim, a seção 8.45 apresenta uma síntese de todas as simulações realizadas.

O capítulo 9 faz a análise de estabilidade dos modelos matemáticos do microtick e macrotick e também, individualmente, dos modos de operação STM e NOMO com e sem função amortizadora, utilizando transformada Z, análise de pólos e zeros e lugar das raízes

O capítulo 10 apresenta as conclusões e sugestões de trabalhos futuros.

2 CONCEITOS BÁSICOS E REVISÃO BIBLIOGRÁFICA

Neste capítulo são apresentados os conceitos básicos e a revisão da literatura.

2.1. Sistemas de Controle por Rede (NCS)

Segundo Moreira (2006), computação em tempo real não é equivalente à computação rápida. O objetivo da computação rápida é minimizar o tempo de resposta médio de um dado conjunto de tarefas. O objetivo da computação de tempo real é atender aos requisitos individuais de tempo de cada tarefa. A propriedade mais importante de sistemas de tempo real deve ser a previsibilidade: sua funcionalidade e seu comportamento temporal devem ser tão previsíveis em tempo de projeto quanto forem necessários para satisfazer as especificações do sistema em tempo de execução.

Computação rápida por si só não garante previsibilidade. Segundo Moreira (2006), sistemas de tempo real podem ser compostos por três estágios: aquisição de dados através de sensores, processamento de dados, no controlador e saída para atuadores. Uma arquitetura de um sistema de tempo real deve ser projetada para suportar esses componentes com a fidelidade adequada, de maneira a se obter um controle satisfatório se comparado a um controle contínuo dentro de um envelope de operação previamente estabelecido.

Uma das arquiteturas mais utilizadas em sistemas de controle, principalmente no âmbito de controle "*safety critical*", é o controle por rede (*Networked Control System – NCS*), que integra a comunicação, a computação e o controle. Segundo Valdivia (2009), um sistema de controle por rede existe quando a malha de controle é fechada através de uma rede de comunicação (NCS), como mostra a Figura 1.1.

A introdução de uma arquitetura de rede com um barramento comum oferece certas características que podem melhorar a eficiência, a flexibilidade e a

confiabilidade, reduzir os tempos de instalação, configuração, manutenção e, sobretudo, o custo. Além disso, sistemas de controle por barramento de comunicação, facilitam o uso da teoria de tolerância a falhas, pois permitem que seja feito com mais facilidade o uso de arquiteturas reconfiguráveis, redundâncias, entre outras. Alguns protocolos de comunicação (normalmente os do tipo TDMA) facilitam alcançar o determinismo, necessário para sistemas de tempo real.

No entanto, este tipo de arquitetura introduz um atraso de tempo (*delay*) entre os componentes, sendo este atraso dependente do protocolo de comunicação utilizado e pode ser causado por diversos atributos, entre os principais estão:

a) Compartilhamento do meio de comunicação;

b) Tempo computacional para a codificação física do sinal;

c) O próprio processo de transmissão (o que hoje em dia, dada às altas taxas de transferência de dados alcançadas por algumas redes fazem que o tempo de transporte seja desprezível; e

d) A sincronização dos relógios (no caso de protocolos do tipo TDMA).

Em alguns protocolos de comunicação é necessário um algoritmo de sincronização de tempo para ordenar e estabelecer uma correta base de tempo entre todos os nós da rede e com isso estabelecer o determinismo e a confiabilidade na comunicação.

2.2. Protocolos de Comunicação

Existe uma ampla variedade de protocolos de comunicação empregados em sistemas de controle por rede. A diversidade de aplicações faz que um protocolo seja mais adequado para uma aplicação que para outra. Portanto é comum ver na bibliografia estudos de comparação entre protocolos como em Gwaltney e Briscoe (2006).

O estudo comparativo feito por Gwaltney e Briscoe (2006), chamou muito a atenção por três motivos: 1) pela quantidade de protocolos que são estudados (11 no total) incluindo o TTP e o Ethernet; 2) porque o projeto onde serão utilizados os protocolos estudados, pertence à área aeroespacial, mais especificamente a um projeto da NASA.

Este trabalho utiliza-se de dois tipos de protocolo:

a) O TDMA (*Time Division Multiple Access*) que tem entre sua classe o protocolo TTP (*Time Triggered Protocol*), o MIL-STD-1553, o TTEthernet;

b) O CSMA/CD (*Carrier Sense Multiple Access Collision Detection*) que tem entre sua classe a Ethernet.

O motivo a escolher estes protocolos são os seus potenciais usos na área aeroespacial.

2.2.1. TDMA

2.2.1.1. Time Triggered Protocol - TTP

Em Lustosa (2009) é feita uma excelente revisão bibliográfica sobre arquitetura TTA, onde é dito que a Arquitetura Disparada por Tempo (*Time-Triggered Architecture - TTA*) engloba um modelo completo de projeto e funcionamento de sistemas distribuídos com características de tempo-real. A proposta apresentada em Kopetz e Bauer (2003), é resultado de um extenso trabalho desenvolvido no contexto do projeto MARS (*Maintainable Real-time System*) que é um sistema distribuído tolerante a falhas para controle de processos de tempo real. O protocolo TTP, utiliza-se da abordagem TDMA. Esta abordagem é o princípio fundamental do modelo de comunicação gatilhado por tempo (*time-triggered* - TT) que estabelece que as ações mais relevantes do sistema sejam disparadas em instantes de tempo previamente definidos à medida que o tempo avança, conforme ilustra a Figura 2.1.

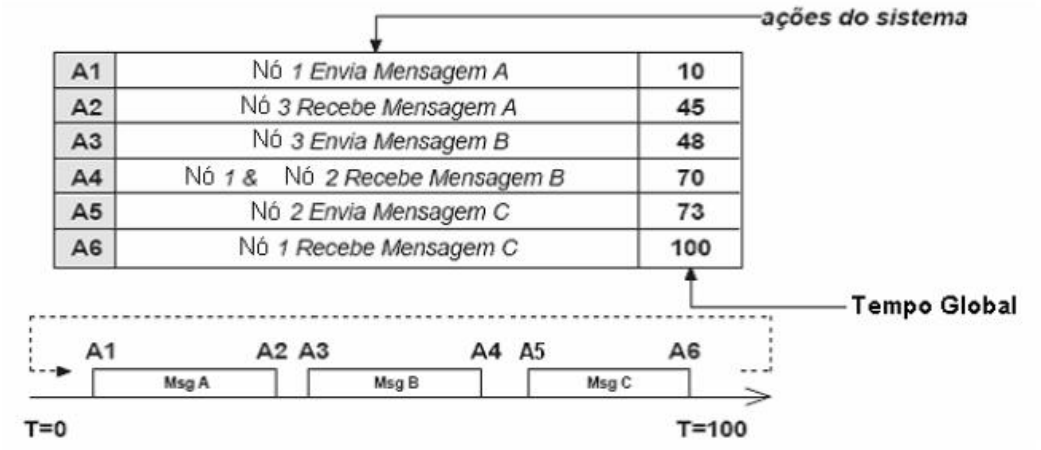


Figura 2.1 – Modelo de Comunicação Time-Triggered.

Fonte: Lustosa (2009).

Em se tratando de sistemas de controle por rede (NCS), onde as unidades de processamento estão fisicamente separadas no espaço, fica clara a necessidade de uma base de tempo global sincronizada. Em razão disso, cada controlador de rede da TTA possui um mecanismo de sincronização de relógios (*clocks*). De acordo com a versão do protocolo utilizado, o mecanismo de sincronização pode apresentar as seguintes arquiteturas: 1) centralizada, como na versão TTP/A, versão mais simples e de baixo custo; 2) essencialmente distribuída, como na versão TTP/B; e 3) distribuídas e tolerantes às falhas como na versão TTP/C.

O tempo de acesso ao barramento é particionado em intervalos de tempo (*slots*) dedicado, segundo um mecanismo TDMA. Todos os *slots* têm o mesmo tamanho. O que precisa ser definido na fase de projeto é a alocação dos *slots* aos nós, em função do tempo de transmissão do maior quadro e da velocidade do controlador de rede na execução das funções primitivas da plataforma, de acordo com Lustosa e Souza (2008).

A sequência na qual cada nó possui um *slot* de tempo exclusivo para transmitir é chamada de rodada TDMA (*TDMA round*). Após o término de uma rodada, outra rodada com uma seqüência diferente de mensagens é iniciada. O

conjunto de todas as rodadas previstas recebe o nome de ciclo TDMA (*TDMA cycle*), ou ciclo do aglomerado de computadores (*cluster cycle*), que consiste numa linha base de tempo finita dentro da qual as ações do sistema são agendadas. Quando o ciclo chega ao seu fim, a sincronização dos relógios ocorre e o ciclo é reiniciado.

O TTP/C tem sido empregado numa ampla variedade de transportes tripulados, incluindo o sistema de controle de pressão da cabine do Airbus A380, controladores digitais de motores com completa autoridade (*FADECs*), em sinalização de trens, veículos espaciais e outros. Tem sido também usado no sistema de *drive-by wire* de automóveis conceituais, além de estar presente em partes do ônibus espacial da NASA.

As hipóteses de falha do TTP/C garantem que o sistema de comunicação pode tolerar qualquer falha simples em qualquer componente da arquitetura e pode tolerar, também, múltiplas falhas dependendo da aplicação. O TTP/C utiliza um algoritmo por média tolerante a falhas bizantinas, como o algoritmo Welch e Lynch (1988) e Kopetz e Ochsenreiter (1987) para sincronização de relógios.

2.2.2. CSMA/CD

2.2.2.1. Ethernet

O protocolo de comunicação Ethernet é amplamente utilizado em arquiteturas de comunicação para rede de computadores comerciais, governamentais e instituições de educação. Está sendo utilizado, com adaptações, também em aplicações militares e aeroespaciais, e atualmente é usado na Estação Espacial Internacional. O padrão IEEE 802.3-2002 define a Ethernet em sua versão atual, incluindo as especificações para o Gigabit Ethernet.

Como sugere a designação, a Ethernet promove a transmissão de dados com taxa de 10 MB/s, 100 MB/s e agora 1 Gb/s sobre cabo de par trançado. A Ethernet opera em modo *half-duplex* (todos os nós compartilham o mesmo

cabo) ou em modo *full-duplex* (os nós podem se comunicar sobre um cabo dedicado com outro nó). Em operação *half-duplex*, o CSMA/CD governa o compartilhamento dos canais. Em operação *full-duplex* o *Flow-Control* governa o compartilhamento de canais. Cada nó da rede transmite dados somente quando o canal está vazio. Se dois nós iniciam a transmissão ao mesmo tempo, a colisão é detectada e a transmissão é cancelada. Cada nó espera um tempo aleatório para a linha de transmissão ficar disponível, e inicia a transmissão novamente. Se houver colisão a operação é repetida. Claramente, isto pode resultar em uma ineficiente comunicação, especialmente quando o tráfego de dados é intenso, pois a Ethernet é uma rede de “melhor-esforço” (*Best-effort*). Isto é, os pacotes são tratados sem distinção alguma, sem prioridade, sem garantia de tempo de entrega e até mesmo sem garantia de entrega ao destino, independente da sua aplicação, porém, sendo feito o máximo de esforço para a entrega destes. Assim todos os pacotes são aceitos e não havendo congestionamento nem colisões, eles são encaminhados ao destino caso contrário podem ser descartados. O modo full-duplex é possível quando os nós são conectados a um comutador (switch) que permite uma conexão dedicada entre a porta do switch e a do nó. O switch fica assim, responsável pelo roteamento das mensagens para o destinatário sem colisão.

Os sistemas com Ethernet de hoje têm limites quando se combinam redes de Ethernet clássicas com dispositivos e serviços. A escalabilidade destes sistemas também é limitada e a solução de rede é desenvolvida para uma área de aplicação específica. Com a intenção de melhorar o determinismo das redes ethernet, foi desenvolvido um padrão de sincronização denominado IEEE 1588. Este padrão utiliza uma arquitetura de sincronização centralizada, isto é, o sistema possui um relógio mestre real que é responsável por sincronizar os relógios de todos os nós da rede. Com isto, é possível aumentar a confiabilidade e o determinismo da comunicação.

Em 2009 a empresa TTTECH lançou o protocolo de comunicação *TTEthernet* que combina o determinismo, tolerância a falhas e propriedades de tempo real

da tecnologia *time-triggered* com a flexibilidade, dinâmica e o legado de “melhor-esforço” do protocolo Ethernet, servindo de comunicação para diversos tipos de aplicações diferentes, (TTTECH COMPUTERTECHNIK AG, 2010).

2.3. Teoria de Relógios

A noção de tempo é fundamental para a nossa existência e muito familiar em nossas vidas. Nós podemos refletir sobre eventos passados, presentes e possíveis eventos futuros, e assim argumentamos sobre eventos no domínio do tempo. Em muitos modelos de fenômenos naturais, o tempo é a variável independente que determina a seqüência de estados do sistema (KOPETZ, 1997).

O tempo possui várias propriedades, sendo uma das principais a monotonocidade. Fazendo uma breve reflexão e análise é possível, baseado nos conhecimentos sobre o tempo, observar que fisicamente o tempo possui somente uma ordem, que no caso sempre avança. Em um sistema distribuído, conceitos como este são de suma importância, pois os relógios podem ser facilmente ajustados, intencionalmente ou não, para representar um tempo no passado ou no futuro, variando e, em alguns casos, até modificando sua propriedade monotônica ao decorrer do ciclo de vida do sistema. Além disso, um sistema distribuído pode apresentar várias bases de tempo diferentes com propriedades monotônicas diferentes.

Com o advento da ciência moderna, mecanismos para medir a progressão do tempo utilizando relógios físicos têm sido inventados. Um relógio é um dispositivo para medir a progressão do tempo. Com o uso de relógios em sistemas, houve a necessidade de se criarem métodos para geração, medição, distribuição e sincronização do tempo. Na seqüência, são apresentados os principais conceitos utilizados no decorrer deste trabalho.

2.3.1. Monotonicidade

Uma das principais propriedades do tempo é a monotonicidade. Para melhor descrever esta propriedade, este trabalho utiliza-se da teoria de funções monotônicas. Para um sistema ser considerado monotônico é necessário que o mesmo preserve a sua ordem inicial definida, ou seja, o sistema possui uma única direção (crescente ou decrescente).

Assim, sejam A e B dois conjuntos ordenados com uma função f definida em A com os valores em B ($A \rightarrow B$), então pode-se definir a função f como:

a) Monótona crescente: quando para todo x_1 e x_2 em B, com $x_1 < x_2$, tem-se $f(x_1) \leq f(x_2)$.

b) Monótona estritamente crescente: quando para todo x_1 e x_2 em B, com $x_1 < x_2$, tem-se $f(x_1) < f(x_2)$.

c) Monótona decrescente: quando para todo x_1 e x_2 em B, com $x_1 < x_2$, tem-se $f(x_1) \geq f(x_2)$.

d) Estritamente decrescente: quando para todo x_1 e x_2 em B, com $x_1 < x_2$, tem-se $f(x_1) > f(x_2)$.

Assim, trazendo este conceito ao mundo dos relógios, temos que dado dois relógios A e B, t1 e t2 dois instantes de medida e f o valor da medida do tempo do relógio B em relação a um relógio A.

Os métodos de sincronização do tempo, que é o principal interesse deste trabalho, são usados para manter as medições de tempo dentro de uma precisão ou exatidão pré-definida, como também para manter a ordem monotônica de todo o sistema.

2.3.2. Acontecer Antes

Diferentes aplicações podem ser sensíveis a problemas relativos à sincronização de tempo. Em uma aplicação distribuída de tempo real, o sistema de computação distribuída executa suas diferentes funções simultaneamente. Estas funções são, normalmente, executadas em diferentes nós. Para garantir um comportamento consistente e estável, do sistema distribuído, é necessário que todos os nós do processo obedeçam à mesma ordem temporal (devem possuir a mesma ordem monotônica), isto é, deve ser obedecido o conceito de “Acontecer Antes (*Happening Before*)” proposto por Lamport (1978). Para que a ordem temporal e o conceito “Acontecer Antes” sejam obedecidos é necessário o uso de relógios físicos e/ou virtuais.

Em Lamport (1978) define-se que relógios lógicos são relógios virtuais que são gerenciados por software. Além disso é descrito um método simples para sincronizar os relógios lógicos de cada nó dentro de um limite de precisão e condições de operação, mostrando que algoritmos de sincronização de relógios podem ser usados para garantir a ordem temporal de um sistema distribuído.

2.3.3. Relógios

2.3.3.1. Relógio Físico

Os relógios físicos são caracterizados pelo material utilizado para gerar a oscilação. Existem muitos tipos de relógios físicos, podendo-se citar entres os principais, os relógios atômicos e os relógios de quartzo. O relógio atômico é um medidor de tempo que funciona baseado em uma propriedade do átomo que quando estimulado por ondas eletromagnéticas faz com que a sua energia oscile de forma regular. Aproximadamente a cada 9.192.631.770 oscilações de energia do átomo de césio-133 o relógio entende que se passou um segundo. O relógio de quartzo baseia-se na vibração do quartzo quando estimulado por uma tensão elétrica alternada.

Um relógio físico é um dispositivo para medir o tempo. Kopetz (1997) define o relógio físico como um contador e um mecanismo físico de oscilação que, periodicamente, gera eventos que vão adicionar contagens ao contador, onde esse evento periódico é denominado *ticks* (Ti) do relógio. A duração entre dois *ticks* consecutivos é denominada de resolução (em inglês, *granularity*) do relógio. A resolução de um relógio pode ser calculada pelo inverso da frequência de oscilação do oscilador do relógio. Em um relógio digital suas medidas de tempo possuem um erro de quantização.

2.3.3.2. Relógio Virtual

Um relógio virtual também é um dispositivo para medir o tempo, no entanto controlado via software. De fato, o relógio virtual é uma abstração do conceito de relógios físicos. Ou seja, todo relógio virtual possui um relógio físico. Este conceito ficará mais evidente em definições subseqüentes.

2.3.3.3. Relógio de Referência

É o relógio utilizado como referência para sincronizar o conjunto. Se for um relógio físico, normalmente é o relógio proveniente de padrões de tempo universais, tais com UTC ou TAI ou de relógios atômicos (normalmente provenientes de satélites de alguma constelação GNSS[1]). Se for um relógio virtual, pode ser proveniente também de algum padrão de tempo, relógios físicos ou relógios virtuais.

2.3.3.4. Tempo Global (Global Time)

Tempo global, ou em inglês *Global Time*, é o nome utilizado para indicar um relógio de referência virtual. Algoritmos de sincronização de relógios por métrica de precisão, utilizam deste artifício de se sincronizar por uma referência virtual, que no caso é denominado tempo global. Em algoritmos de

[1] GNSS - Sistemas de Navegação Global via Satélites, tais como GPS (Sistema de Posicionamento Global) que possui em cada um de seus satélites, um relógio atômico sincronizado com a constelação.

sincronização de relógios, este tempo é atingido localmente, ou seja, baseado em equações matemáticas, os relógios individualmente tenderão a um valor, este valor é o tempo global. A ideia do tempo global é fazer com que todos os nós do sistema distribuído gerem seus eventos de tempo, conhecidos como macroticks, em períodos iguais ou o mais próximo possível dentro de uma determinada precisão. Com o uso do conceito de tempo global é possível estabelecer uma ordem temporal de eventos entre todos os nós do sistema distribuído. Protocolos de comunicação que usam o TDMA como acesso ao meio normalmente fazem uso do conceito de tempo global para ordenar as transmissões dos nós do sistema distribuído.

2.3.4. Microtick e Macrotick

Uma boa representação do tempo é de extrema importância para entender e estabelecer a sincronização entre os relógios. De acordo com Kopetz, Hexel, *et al.* (1997), em sistemas aglomerados, onde cada nó possui um oscilador local com uma freqüência individual, a representação do tempo deve ser independente das características individuais de cada oscilador. Isto é, a representação do tempo precisa ser relativa a uma freqüência nominal comum a todo o conjunto. Além disso, a representação deve satisfazer os seguintes critérios:

a) Deve ser entendível pelos humanos;

b) Deve ser independente de detalhes de implementações e velocidades da comunicação;

c) Deve ser controlável e/ou observável pelo computador;

Padrões internacionais de tempo, tal como o TAI (*International Atomic Time*) já definem o padrão do segundo, seguindo tais critérios. Contudo, a grande dificuldade do entendimento é saber separar os diferentes conceitos/escalas de tempo que cada nó individualmente possui e como estas representações

refletem os padrões internacionais. Relógios, por padrão, devem produzir suas medidas de tempo seguindo tais critérios. Portanto, considerando estes critérios satisfeitos, é importante diferenciar corretamente as diversas escalas de tempo que um relógio local pode produzir.

De acordo com Kopetz, Hexel, *et al.* (1997), os osciladores, locais, produzem diversos sinais que servem a vários propósitos, tais como:

a) Gerar os sinais temporais para todas as unidades computacionais locais. Este sinal é denominado ***tick***;

b) Gerar um sinal referencial local para as medidas de tempo e a geração da codificação e amostragem dos sinais para o controle do fluxo de comunicação. Este sinal é denominado ***microtick***;

c) Gerar um sinal temporal para referências externas (tempo global, ou relógio de referência). Este sinal é denominado ***macrotick***.

A Figura 2.2 ilustra um diagrama em blocos com os diversos sinais que um relógio local possui.

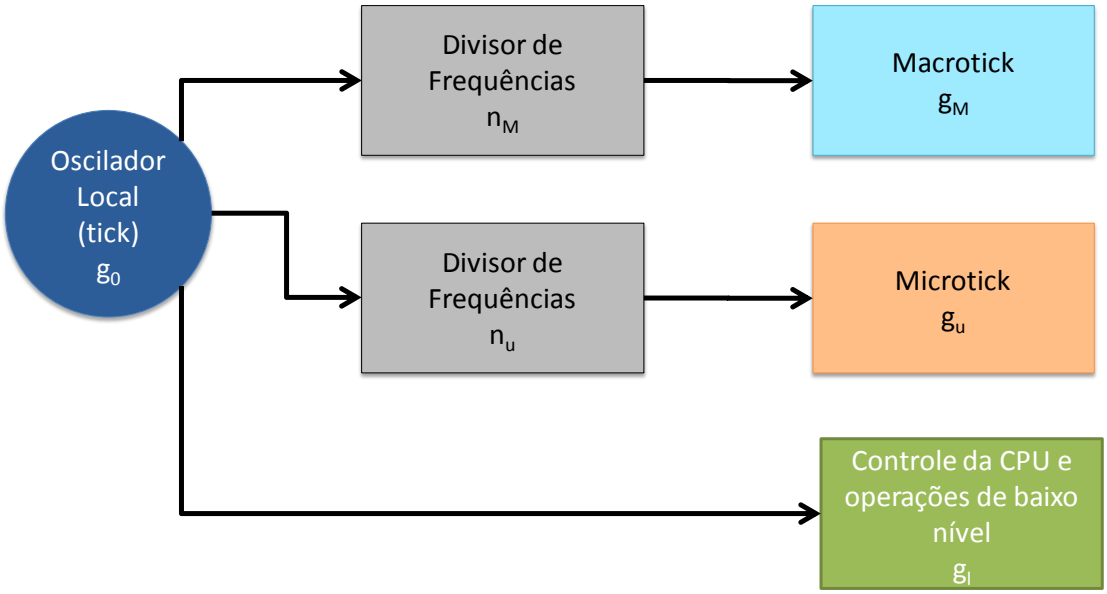


Figura 2.2 - Sinais de tempo do oscilador local.

Fonte adaptada: Kopetz, Hexel, *et al.* (1997)

Na Figura 2.2 é possível ver a presença do oscilador local, que gera os *ticks*, para o controle da CPU; o microtick e macrotick que são provenientes de um divisor de freqüências ligados ao oscilador local.

Assim, este trabalho, define que cada nó local possui:

a) Um relógio físico local como um contador e um mecanismo físico de oscilação que, periodicamente, gera eventos que vão adicionar contagens ao contador. Este evento periódico é definido como *ticks* (T) do relógio;

b) Um tempo local como microtick (μT) que é definido como um evento periódico gerado pelo divisor de freqüências, com freqüência diferente do relógio físico local;

c) Um tempo virtual como macrotick (mT) que é definido como um evento periódico gerado pelo divisor de freqüências;

d) A duração entre dois microticks consecutivos como a resolução (*granularity*) do relógio local.

e) A duração entre dois macroticks consecutivos como a resolução (*granularity*) do tempo global.

A Figura 2.3 ilustra os sinais do microtick (uT) e macroticks (mT).

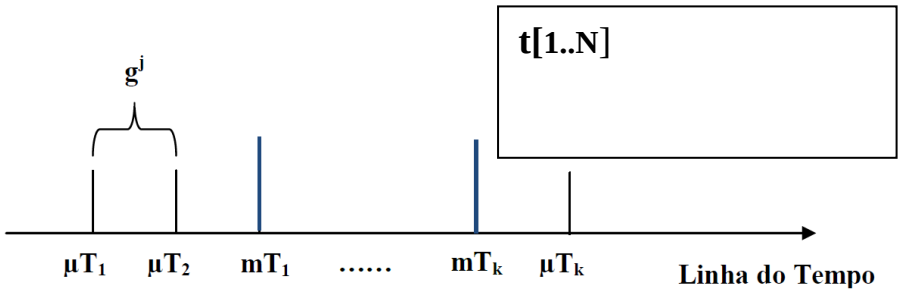


Figura 2.3 - Microtick e Macrotick.

2.3.5. Imperfeições de Relógios

Relógios físicos ou virtuais possuem imperfeições. Essas imperfeições podem ser causadas por vários motivos, dentre eles estão as variações das condições ambientais; tais como variação de temperatura, variação de tensão ou envelhecimento do cristal (no caso de um relógio de cristal de quartzo). A Figura 2.4 abaixo, apresenta as principais imperfeições na operação de um relógio.

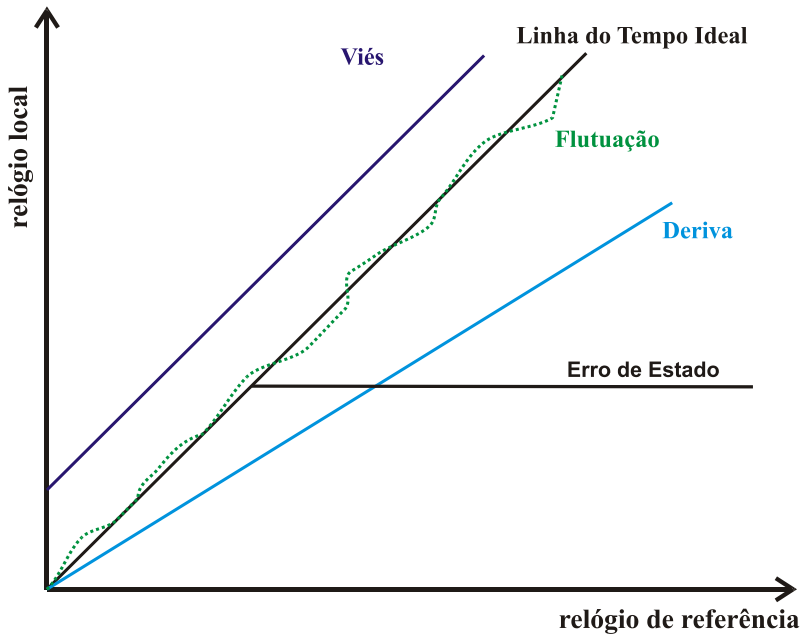


Figura 2.4 - Imperfeições do relógio.

Estas imperfeições do relógio são conhecidas como a deriva (*drift*), viés inicial/instantâneo (*offset*), flutuação (*jitter*) e erro de estado (*state error*):

a) A **deriva** é quando um relógio local tem uma freqüência de oscilação maior ou menor que um relógio de referência, isto é, a deriva é a variação (ou diferença) entre dois relógios;

b) Existem dois tipos de vieses, o **viés inicial** que é quando as condições iniciais entre o tempo dos relógios do conjunto são

diferentes; e o **viés instantâneo** que é a diferença entre o tempo dos relógios locais do conjunto a cada instante;

c) A **flutuação** é a incerteza na medida do relógio, podendo ser causada por diversos fatores;

d) O **erro de estado** é quando um relógio interrompe seu funcionamento, isto é, o relógio fixa em um valor.

Quando um relógio local possui uma freqüência de oscilação mais rápida ou mais lenta do que outro relógio, é dito que o relógio local possui uma taxa de variação em relação ao outro relógio; conseqüentemente os *ticks* dos relógios se diferem. Para se alcançar a sincronização do tempo de dois ou mais relógios diferentes, é necessário o uso de técnicas de sincronização de relógios. Estas técnicas normalmente visam estabelecer uma referência, podendo este ser real ou virtual, e assim fazer com que os relógios do conjunto façam o rastreamento desta referência.

2.3.6. Precisão e Exatidão

A sincronização de relógios visa reduzir ou anular o efeito da deriva que o relógio proporciona. Para isso, as métricas precisão e exatidão são muito importantes para o entendimento do processo de sincronização, pois os algoritmos de sincronização de relógios são baseados nas mesmas.

A diferença, em um instante de interesse, entre dois *microticks* de dois relógios quaisquer (relógio i e relógio j) é definido como viés instantâneo, descrito pela equação (2.1), em que o subscrito indica o relógio e k o instante.

$$\Delta uT(k) = uT_i(k) - uT_j(k) \tag{2.1}$$

De acordo com Kopetz (1997), a precisão (em inglês *precision*) é o máximo viés instantâneo de um conjunto de relógios, durante um período de interesse.

Em outras palavras, a precisão descreve a máxima diferença dos *microticks* em torno de um tempo global. Assim, pode-se definir a equação da precisão como:

$$\Pi = \max\left[\Delta uT(k)\right] \tag{2.2}$$

Algoritmos de sincronização de relógios baseados nesta métrica, visam minimizar ao longo do tempo a diferença dos microticks em torno de um tempo global.

De acordo com Kopetz (1997) a exatidão (em inglês *accuracy*) é o máximo viés instantâneo entre o relógio local e um relógio de referência externo (virtual ou real), durante um período de interesse. Em outras palavras, a exatidão descreve o quão perto os tempos dos relógios locais estão de uma referência. Algoritmos de sincronização baseados nesta métrica, visam minimizar ao longo do tempo as diferenças locais dos microticks em relação a uma referência. Em outras palavras, os relógios locais rastreiam o relógio de referência. Esta métrica é muito utilizada em sistemas que possuem um relógio externo como referência, tal como o GPS ou um padrão de tempo internacional.

A Figura 2.5 ilustra as diferenças entre precisão e exatidão, como medidas que atingem um alvo.

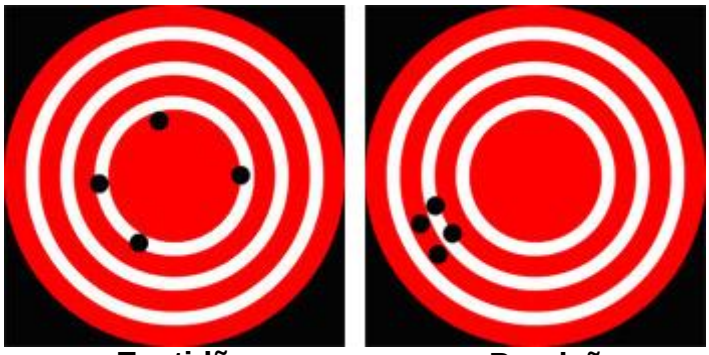


Figura 2.5 - Representação da exatidão e precisão ao atingir um alvo.

Fonte: Carpi e Egger (2008).

A Figura 2.5 ilustra que a exatidão é máxima diferença em relação a uma referência real (que no caso é o centro do alvo) e a precisão é a máxima diferença a uma referência virtual (que no caso é a média de todos os pontos pretos). Na métrica de precisão, os pontos do conjunto estão próximos entre si, podendo se afastar da referência real. Assim, da Figura 2.5 pode-se concluir que um sistema pode ter as seguintes configurações:

a) Exato e não preciso - Pontos perto da referência real, mas distantes entre si;

b) Exato e preciso - Pontos perto da referência real e próximos entre si;

c) Preciso e não exato - Pontos próximos entre si e distantes da referência real.

2.3.7. Arquiteturas de Sincronização

Existem muitos algoritmos de sincronização de relógios na literatura. Estes algoritmos de sincronização seguem alguns padrões de arquitetura, tais como distribuídos e centralizados. A maioria dos algoritmos de sincronização de relógios segue um destes padrões, sendo que a grande diferença das duas arquiteturas é que: 1) a arquitetura distribuída estabelece um relógio mestre virtual, também chamado de relógio global virtual; 2) a arquitetura centralizada é dependente de um relógio mestre real ou virtual, também chamado de relógio global real ou virtual.

2.3.7.1. Arquitetura Distribuída

De acordo com Kopetz (1997) um algoritmo que tolera erros bizantinos é aquele que respeita a condição da equação (2.3).

$$n \geq 3f + 1 \tag{2.3}$$

Em que *f* é o número de falhas que o algoritmo pode tolerar e *n* é o número de relógios ou nós no sistema. Isto é um dos principais atributos que caracteriza a

arquitetura distribuída neste trabalho. Em Kopetz (1997) mostra-se que a re-sincronização de uma arquitetura distribuída opera em três fases diferentes. A Figura 2.6 mostra um diagrama em blocos da re-sincronização da arquitetura distribuída.

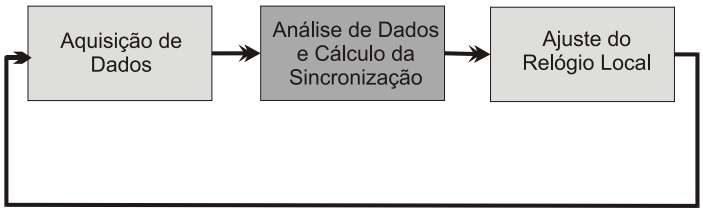


Figura 2.6 - Diagrama em blocos de fases de uma arquitetura distribuída.

No primeiro bloco da Figura 2.6, o nó recebe os dados de tempo dos relógios presentes na rede de comunicação. O segundo bloco é responsável pela análise, para evitar erros bizantinos, e pelo cálculo da função de convergência. O terceiro bloco é responsável pelo ajuste do relógio lógico do nó local e com isso a seqüência se reinicia.

Todos os nós presentes na rede de comunicação utilizam-se destas três fases para manter sua sincronização de relógios.

A Figura 2.7 mostra uma arquitetura de sincronização distribuída, aonde é possível ver que não existe a presença de um relógio mestre.

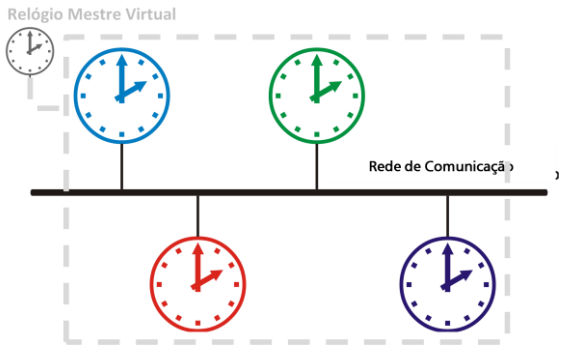


Figura 2.7 - Arquitetura de Sincronização Distribuída.

A Figura 2.7 mostra quatro relógios interligados por um barramento de rede. Cada relógio é um nó da rede. A arquitetura distribuída não utiliza um relógio mestre real como referência, esta arquitetura faz uso do tempo global para se sincronizar.

Protocolos do tipo TDMA, tal como o TTP, normalmente aplicam essa arquitetura de sincronização em suas operações de sincronização de relógios. Alguns algoritmos de sincronização, tal como FTM, utilizam-se desta arquitetura para sincronização.

2.3.7.2. Arquitetura Centralizada

A arquitetura centralizada utiliza-se de um relógio mestre que periodicamente envia os dados de tempo aos relógios escravos. Os relógios escravos recebem o valor de tempo do relógio e comparam com o tempo de chegada mais o atraso de rede. Com isso, é possível medir a diferença entre o relógio mestre e o relógio escravo e assim corrigir os valores do relógio escravo periodicamente. A Figura 2.8 mostra a arquitetura de sincronização centralizada, aonde é possível ver a presença de um relógio mestre.

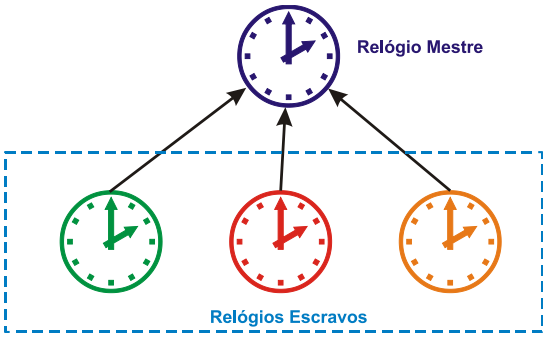


Figura 2.8 - Arquitetura de Sincronização Centralizada.

O relógio mestre real normalmente é um relógio de excelente qualidade, tal como um relógio atômico, ou um padrão de tempo internacional. Esta arquitetura é muito utilizada em sistemas que se utilizam do GNSS, como relógio mestre, para sincronização de relógios. Em alguns casos também é possível que o relógio mestre seja um relógio virtual, mas sendo importante

ressaltar que a principal característica desta arquitetura é a presença do relógio mestre real ou virtual.

Esta arquitetura centralizada além de normalmente não ser tolerante a erros bizantinos tem a desvantagem de possuir um ponto comum de falha, isto é, se o relógio mestre falhar, todos os relógios escravos perdem a referência comprometendo a sincronização do sistema. Essa desvantagem é minimizada com o uso de relógios de excelente qualidade, com elevado preço no mercado, ou o uso de um GNSS como referência, o que pode elevar o custo de um sistema.

Para eliminar o ponto comum de falha, é possível utilizar a estratégia multi-mestre. Se o relógio mestre principal falhar, um dos relógios da rede, previamente selecionado, assume o papel de mestre e continua a re-sincronização do sistema.

Alguns algoritmos de sincronização, tal como o PTP, utilizam-se desta arquitetura para sincronização.

2.3.8. Sincronização Interna

Em sistemas distribuídos de controle em tempo real que utilizam o TDMA para comunicação, ou algum protocolo que necessite de sincronismo, é crucial se ter uma boa sincronização interna de relógios. A proposta da sincronização interna de relógios é estabelecer uma base de tempo global, baseado na métrica de precisão, com tolerância a erros bizantinos.

Cada nó do sistema distribuído possui seu relógio local que gera seus microticks de acordo com a freqüência do oscilador local. O macrotick de cada nó é formado pelo conjunto de microticks.

Como os relógios físicos de cada nó possuem taxas de derivas diferentes é necessário o uso de algoritmos de sincronização que ajustem os relógios do nó periodicamente para que uma precisão pré-definida seja alcançada e estabelecida. Esse período é denominado de período de re-sincronização.

Os macroticks de cada nó devem ser periodicamente re-sincronizados para estabelecer o tempo global dentro de uma determinada precisão Π. O período de re-sincronização é normalmente chamado de intervalo de re-sincronização. Na Figura 2.9 é ilustrado o período de re-sincronização, e o local aonde é aplicada a função de convergência que atualiza os dados de relógio para manter os valores de tempo dentro de uma determinada precisão.

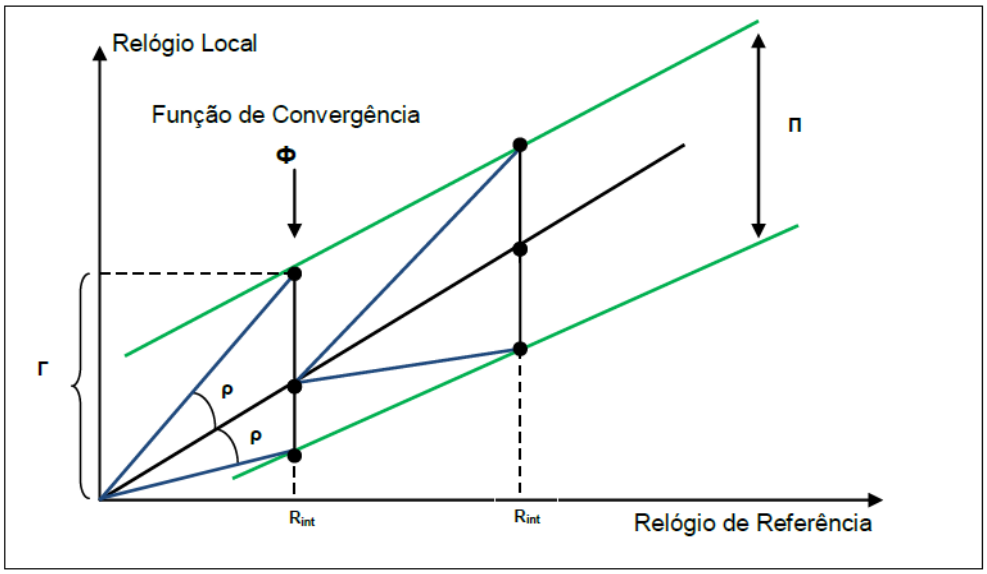


Figura 2.9 - Intervalo de re-sincronização e função de convergência.

O valor Γ indica o máximo viés durante o intervalo de re-sincronização, ρ é a taxa de deriva e Φ a função de convergência. O máximo viés depende do período de re-sincronização. A função de convergência Φ denota o valor do viés de tempo imediatamente após a re-sincronização.

Da Figura 2.9 tem-se o máximo viés, dado pela equação:

$$\Gamma = 2\rho R_{int} \tag{2.4}$$

Da equação (2.4), da Figura 2.9 e supondo um algoritmo tolerante a erros bizantinos, têm-se as seguintes condições de sincronização de relógios:

d) Condição de Sincronização:

$$\Gamma + \Phi \leq \Pi \tag{2.5}$$

e) Teorema de Tolerância a Erros Bizantinos, onde f é número de erros possíveis e n o número de relógios do conjunto, dado pela equação (2.3).

Os algoritmos de sincronização por métrica de precisão devem garantir que o viés instantâneo dos relógios do conjunto não divirja mais do que o intervalo de precisão definido, além de serem tolerantes a erros bizantinos.

O máximo viés é calculado pela equação (2.5) e a função de convergência é dependente do algoritmo de sincronização utilizado.

2.3.9. Sincronização Externa

A proposta da sincronização externa de relógios é sincronizar o tempo global com algum padrão de tempo externo, ou seja, pela métrica exatidão. O tempo global, por causa da influência de alguns parâmetros como a taxa de deriva dos relógios, a latência da rede, entre outros pode derivar em relação a um padrão de tempo de referência e/ou relógio externo. Existem vários meios para se ter uma referência de tempo, tal como o protocolo NTP, o GPS, entre outros. O GPS, atualmente, é um dos meios mais utilizados, principalmente em sistemas de controle, como referência externa para sincronização dos relógios. A Figura 2.10 ilustra a sincronização externa com GPS.

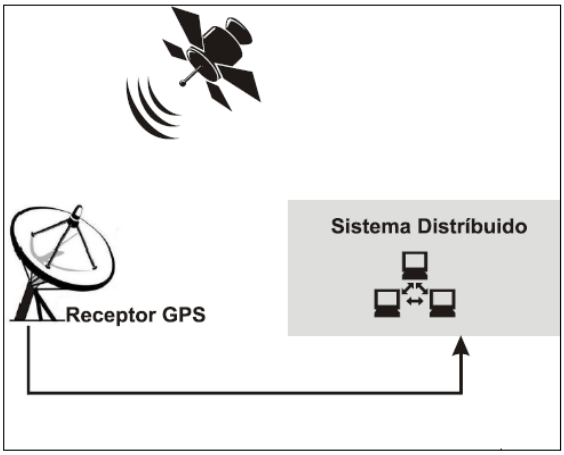


Figura 2.10 - Sincronização externa com GPS.

O GPS é muito utilizado no mundo para navegação e sistemas que necessitam de boa exatidão. Periodicamente, é realizado um broadcast na rede enviando os dados de tempo a todos os nós onde, no caso, o GPS é responsável pelo fornecimento de dados do tempo. Um algoritmo de sincronização de relógios então é utilizado para o cálculo da correção.

2.3.10. Algoritmos de Sincronização

Com o avanço do conceito de relógio lógico, em meados da década de 1980, começaram a surgir algoritmos para a sincronização dos relógios lógicos. Os relógios lógicos apesar de não serem tão precisos quanto uma solução por hardware possuem a vantagem da flexibilidade e de menor custo (RAMANATHAN; SHIN e BUTLER, 1990).

Para manter os relógios dos nós do sistema em sincronia, existem algoritmos de sincronização que são baseados em lógicas matemáticas e que, basicamente, definem as operações que o sistema deve realizar para manter seus relógios sincronizados ou com o tempo mais próximo possível dos limites pré-especificados. Existem também outras soluções além de uma solução por software. Existem soluções por hardware e soluções híbridas, que envolvem hardware e software.

Vários algoritmos de sincronização vêm sendo desenvolvidos ao longo dos anos. Em Lamport (1985) é proposto um algoritmo para solucionar o problema de sincronização de relógios lógicos em um sistema distribuído com tolerância a falhas bizantinas, que ficou conhecido como Algoritmo de Lamport.

Em Welch e Lynch (1988) é proposto outro algoritmo de sincronização de relógios lógicos em um sistema distribuído com tolerância a falhas bizantinas, que ficou conhecido como Algoritmo de Welch-Lynch ou FTM (Fault Tolerant Mid-Point). Os algoritmos de Lamport e Welch-Lynch, apesar de serem muito parecidos, diferem-se na leitura dos relógios e na função de convergência, de acordo com Dutertre (1998). O Algoritmo de Welch-Lynch possui uma função de convergência baseada na mediana do vetor de valores de tempo. Kopetz e Ochsenreiter (1987) propõem uma modificação no algoritmo de Welch-Lynch, ao invés de somente trabalhar com a mediana, é proposto um algoritmo que tem a função de convergência por média dos valores de tempo. Ainda em Kopetz e Ochsenreiter (1987) é introduzido o fator de custo da tolerância a erros bizantinos em relação à precisão final alcançada pelo algoritmo.

Em Cristian (1989) é proposto um algoritmo de sincronização para redes sujeitas a atrasos de redes aleatórios. Para isto foi utilizada uma abordagem probabilística.

Em Attiya, Hay E Welch (2006) é proposto um algoritmo ótimo de sincronização de relógios com restrição de energia para rede wireless *ad-hoc*. É proposto um algoritmo para alcançar uma sincronização de relógios ótima com restrição de energia.

Em Hanzlik, Ademaj e Kopetz (2006) é proposto um algoritmo que combina a sincronização entre os nós da rede por um algoritmo tolerante a falhas bizantinas, utilizando uma arquitetura distribuída, com o a sincronização do tempo externo utilizando um algoritmo com arquitetura distribuída. Com isso o algoritmo minimiza o problema de deriva entre o relógio mestre virtual com o relógio mestre real.

De acordo com Malekpour (2006), um dos grandes problemas da maioria dos algoritmos de sincronização tolerante a falhas bizantinas, citados anteriormente, é que os algoritmos preveem as falhas inesperadas dentro de um número máximo de falhas previstas (vide equação 3.1) em um determinado instante. Outro grande problema dos algoritmos anteriores é que os algoritmos preveem que todos os relógios da malha estejam sincronizados entre si inicialmente. Dado isso, Malekpour (2006) propõe um novo algoritmo denominado *self-stabilizing*. O algoritmo se propõe a se auto-estabilizar de qualquer estado para outro, tolerar rajadas de falhas transitórias e convergir linearmente para o estado desejado em um período determinado.

Miner (1993) apresenta uma verificação formal sobre o funcionamento e a convergência de alguns algoritmos tolerantes a falhas bizantinas em sistemas. No entanto, muitos algoritmos padecem da necessidade de provas formais de que funcionam e convergem dentro das condições previstas.

2.3.10.1. Algoritmo FTM (*Fault-Tolerant Mid-Point*)

O algoritmo de Welch-Lynch, também conhecido como FTM (*Fault-Tolerant Mid-Point*), é um algoritmo com arquitetura distribuída e que apresenta tolerância a falhas bizantinas para a sincronização de relógios de sistemas distribuídos (WELCH e LYNCH, 1988).

Muitas informações sobre o algoritmo de Welch-Lynch pode ser encontrado na literatura, como em Dutertre (1998), onde é feito um relatório com as provas de que o algoritmo mantém os relógios dentro de uma dada precisão. Welch e Lynch (1988) é o artigo original do algoritmo, com muitas informações relevantes sobre o mesmo. Neste trabalho, o algoritmo será explicado abaixo, da melhor forma possível.

O algoritmo de Welch-Lynch faz uma série de suposições, tais como:

a) Todos os nós estão conectados por uma rede de comunicação confiável;

b) Relógios da rede estão inicialmente sincronizados entre si, ou seja, não possuem viés inicial;

c) A comunicação por rede é confiável e os atrasos de transmissão da rede são limitados por duas constantes, em que δ é o máximo valor do atraso de rede e ε o valor do atraso de rede:

$$0 \leq \varepsilon < \delta \tag{2.6}$$

d) O nó não possui controle sobre o relógio físico, somente sobre o relógio lógico; assim, o tempo local para o nó é dado por um relógio virtual (VC) obtido adicionando a correção ao relógio físico (PC). A correção é periodicamente computada e armazenada na variável local ADJ. O relógio virtual do nó então é definido por:

$$VC(t) = PC(t) + Adj \tag{2.7}$$

e) Para ser tolerante a falhas bizantinas, a condição da equação (2.3) deve ser obedecida;

f) Cada nó tem seu relógio físico, o qual pode estar com uma deriva menor (ou maior) em relação ao tempo real a uma taxa limitada por uma pequena constante ρ tal que 0< ρ << *1*.

g) Se o relógio C não falhar durante o intervalo do tempo real $[t_1,t_2]$, então a condição abaixo deve ser obedecida:

$$(1-\rho)(t_2 - t_1) \leq C(t_2) - C(t_1) \leq (1+\rho)(t_2 - t_1) \tag{2.8}$$

Onde $C(t_1)$ e $C(t_2)$ são os valores do relógio C no tempo t_1 e no tempo t_2 respectivamente. O tempo decorrido do relógio $C(t_2) – C(t_1)$ está dentro de $\rho(t_2 – t_1)$ do atraso de tempo real t_2-t_1. Durante o mesmo intervalo, os relógios físicos de dois nós podem derivar separadamente no máximo $2\rho(t_2-t_1)$. Até mesmo para valores pequenos de ρ, o erro pode ser significante para grandes valores de t_2.

Para assegurar que todos os nós tenham uma consistente visão do tempo, é necessária a re-sincronização de relógios regularmente (periodicamente). Para isso o algoritmo segue uma seqüência lógica. Cada nó aplica essa seqüência com o objetivo de chegar a um termo de correção. Com esse termo de correção, os desvios dos relógios causados pela deriva são corrigidos para que todos os relógios do sistema estejam dentro de uma determinada precisão.

A Figura 2.11 mostra o fluxograma do algoritmo de Welch-Lynch.

Na Figura 2.11, onde se tem o número 1 indicando, tem-se um *loop* de condição. Está condição significa que, se o tempo do relógio local do nó é igual ao tempo de re-sincronização, então a sincronização tem o seu início. R_{int} é o período de re-sincronização pré-determinado e *k* é o instante.

No número 2, onde se tem a leitura de dados da rede, significa que os relógios locais trocam informações de seus relógios entre si, isto é, todos os relógios enviam um “broadcast” com a estampa de tempo de seu relógio. A Figura 2.12, ajuda a entender como o “broadcast” é realizado. Neste exemplo utiliza-se um sistema com 4 nós.

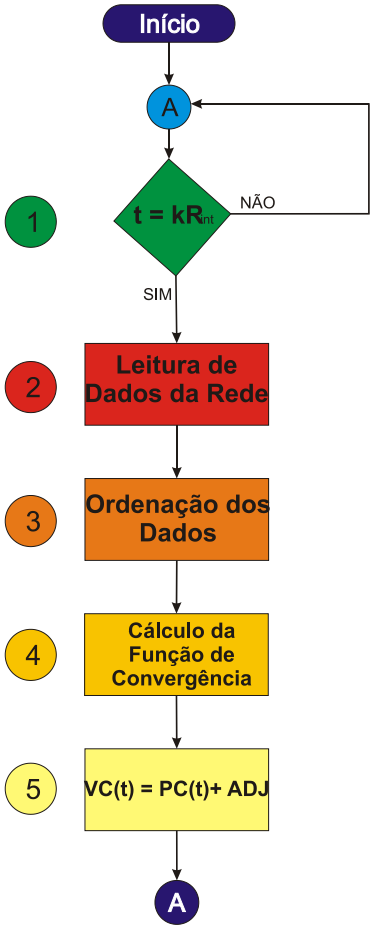


Figura 2.11 - Fluxograma do Algoritmo de Welch-Lynch.

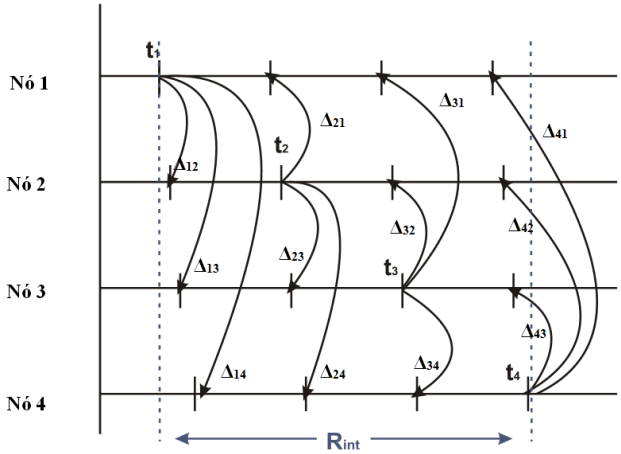


Figura 2.12 - Broadcast do Algoritmo de Welch-Lynch.

Na Figura 2.12 o nó 1 envia sua estampa de tempo, t_1, ao nó 2, 3 e 4. O nó 2, envia sua estampa de tempo, t_2, ao nó 1, 3 e 4. O nó 3 envia sua estampa de tempo, t_3, ao nó 1, 2 e 3. O nó 4 envia sua estampa de tempo, t_4, ao nó 1, 2 e 3. Quando os valores da estampa de tempo chegam ao nó, é feita a diferença

de tempo entre o valor recebido e o valor da estampa de tempo do nó que recebeu. Com isso tem-se uma matriz de diferenças de tempo dada na equação (2.9).

$$\Delta C = \begin{bmatrix} \Delta_{11} & \Delta_{12} & \Delta_{13} & \Delta_{14} \\ \Delta_{21} & \Delta_{22} & \Delta_{23} & \Delta_{24} \\ \Delta_{31} & \Delta_{32} & \Delta_{33} & \Delta_{34} \\ \Delta_{41} & \Delta_{42} & \Delta_{43} & \Delta_{44} \end{bmatrix} \tag{2.9}$$

Na matriz de diferenças de tempo, da equação (2.9), o *Δ* indica a diferença das estampas de tempo dos relógios do nó. Onde o índice que indica a linha indica o nó do relógio que enviou sua estampa de tempo, e o índice que indica a coluna indica o nó do relógio que recebe a estampa de tempo. Quando os índices de linha e coluna são iguais, indica que o relógio enviou e recebeu seu próprio valor, e com isso a diferença de tempo é nula. Então, é possível reescrever a matriz da equação (2.9):

$$\Delta C = \begin{bmatrix} 0 & \Delta_{12} & \Delta_{13} & \Delta_{14} \\ \Delta_{21} & 0 & \Delta_{23} & \Delta_{24} \\ \Delta_{31} & \Delta_{32} & 0 & \Delta_{34} \\ \Delta_{41} & \Delta_{42} & \Delta_{43} & 0 \end{bmatrix} \tag{2.10}$$

A equação (2.10) mostra a matriz de desvios de tempo dos nós do sistema onde cada nó consegue enxergar somente a sua linha.

Após a formação da matriz de desvios de tempo, o próximo passo do algoritmo é ordenar os dados.

Na Figura 2.11, o número 2 indica a ordenação de dados. Cada nó enxerga somente a sua linha da matriz (2.10), formando um vetor de valores A. Esse vetor é ordenado em ordem crescente, seguindo a regra da equação:

$$A[1] \leq A[2] \leq \cdots \leq A[n] \tag{2.11}$$

Onde *n* indica o número de relógios ou nós do sistema, e o vetor *A* é formado pelos valores da linha do nó da matriz (2.10).

Após ordenar os dados de forma crescente, o algoritmo de Welch-Lynch faz o cálculo da função de convergência, representado pelo número 3 na Figura 2.11.

Com os dados do vetor *A* ordenados de forma crescente, agora é necessário encontrar o ponto-médio do vetor. Descarta-se o maior e o menor valor de A. Esse descarte faz com que a função seja tolerante a falhas bizantinas. Depois se faz a média aritmética do maior e menor valor dos elementos restantes no vetor. A equação (2.12), mostra a função de ponto-médio tolerante a falhas bizantinas, onde assumimos que *f* é o número de falhas que o algoritmo pode tolerar e *n* é o número de relógios ou nós no sistema:

$$cfn(A) = \frac{A[f+1] + A[n-f]}{2} \tag{2.12}$$

A equação (2.12) encontra o ponto médio do vetor, com isso agora é possível encontrar a função de ajuste, dado pela equação (2.13):

$$Adj = kR_{int} + \delta - corr(A) \tag{2.13}$$

Onde *k* é o instante, R_{int} é o período de re-sincronização, δ é o termo de compensação do atraso de rede e *corr(A)* é a função de ponto-médio tolerante a falhas bizantinas.

A função de ajuste foi calculada em (2.13). Representado o número 5 da Figura 2.11, o relógio virtual agora corrige o seu valor através da equação (2.14).

$$VC(t) = PC(t) + Adj \tag{2.14}$$

Onde, *VC(t)* indica o relógio virtual, *PC(t)* é o relógio físico e *Adj* é o ajuste calculado pelo algoritmo.

Entre duas sucessivas re-sincronizações, os relógios virtuais podem derivar entre si, mas com o ajuste em todos os nós do sistema faz com que todos os relógios fiquem com um desvio limitado. Esse desvio limitado é chamado de precisão. De acordo com Kopetz (1997), a flutuação do tempo, a flutuação no atraso de rede, a flutuação e o atraso causado pelo agendador (*scheduler*) do computador, o sistema operacional, interrupções, entre outros, afeta e deteriora a precisão da sincronização de relógios.

A Tabela 2.2 apresenta valores de flutuação nas mensagens de sincronização que o sistema pode apresentar em diferentes níveis de análise.

Tabela 2.1 - Valores da flutuação em mensagens de sincronização.

Mensagem de Sincronização	Valores aproximados da Flutuação
Em nível de software	500 µs a 5 ms
No núcleo do sistema operacional	10 µs a 100 µs
No hardware de controle de comunicação	Menor que 10 µs

Fonte: Traduzida de Kopetz (1997).

De acordo com Schneider (1986) a precisão do algoritmo de Welch-Lynch é dada pela equação (2.15):

$$\Pi(\Gamma, \varepsilon) = \frac{\Gamma}{2} + \xi \qquad (2.15)$$

Onde Γ é o máximo offset possível entre dois relógios e ξ é a flutuação de tempo, que pode ser causada por diversos fatores.

Para que o sistema distribuído alcance uma ótima precisão em seu tempo global, é importante que a flutuação nas mensagens de sincronização seja pequena.

2.3.10.2. Algoritmo FTA (*Fault-Tolerant Average*)

O algoritmo FTA (*Fault-Tolerant Average*) é o algoritmo de Welch-Lynch modificado por *Kopetz e Ochsenreiter*. O algoritmo FTA possui uma arquitetura

distribuída e apresenta tolerância a falhas bizantinas para a sincronização de relógios de sistemas distribuídos. Mais informações sobre o algoritmo podem ser encontradas na literatura, como em Kopetz e Ochsenreiter (1987).

O Algoritmo FTA é praticamente igual ao algoritmo de Welch-Lynch, modificando somente a forma de calcular a função de convergência. Portanto o algoritmo segue toda a seqüência e condições apresentadas anteriormente para o algoritmo Welch-Lynch.

O algoritmo ordena os dados de forma crescente, assim como o algoritmo de Welch-Lynch, e faz o cálculo da função de convergência. Os dados da matriz 2.40 são guardados em um vetor *A*. Com os dados do vetor A ordenados de forma crescente, agora é necessário encontrar o valor-médio do vetor. Com isso descarta-se o maior e o menor valor de A, esse descarte faz com que a função seja tolerante a falhas bizantinas. Agora, diferentemente do algoritmo de Welch-Lynch, o algoritmo FTA faz a média aritmética dos elementos restantes no vetor. A equação (2.16), mostra a função de valor-médio tolerante a falhas bizantinas, onde assumimos que *f* é o número de falhas que o algoritmo pode tolerar e *n* é o número de relógios ou nós no sistema:

$$corr(A) = \frac{1}{n-2f} \sum_{i=f+1}^{n-f} \Delta_{i,j} \qquad (2.16)$$

A equação (2.16) encontra o valor médio do vetor e com isso, agora é possível encontrar a função de ajuste, dado pela equação (2.17):

$$Adj = kR_{int} + \delta - corr(A) \qquad (2.17)$$

Onde *k* é o intervalo, R_{int} é o período de re-sincronização, δ é o termo de compensação do atraso de rede e *corr(A)* é a função de valor-médio tolerante a falhas bizantinas.

Todos os outros passos deste algoritmo FTA são iguais aos do algoritmo de Welch-Lynch.

A exatidão e precisão de ambos os algoritmos FTA e Welch-Lynch dependem de:

h) A taxa de deriva dos relógios físicos locais e a duração do intervalo de re-sincronização;

i) Erro ou flutuação na leitura de um relógio em relação a outro;

j) Falhas no sistema causadas por perda de mensagens na rede ou falhas maliciosas (conhecidos como erros bizantinos);

A precisão do algoritmo FTA de acordo com Kopetz e Ochsenreiter (1987) e Schneider (1986) obedece à equação (2.18):

$$\Pi(\Gamma, n, f, \varepsilon) = \frac{f\Gamma}{n - 2f} + \varepsilon \tag{2.18}$$

Onde Γ é o máximo offset possível entre dois relógios, ε é a flutuação, que pode ser causada por vários fatores e f é o número de falhas bizantinas que o sistema distribuído pode suportar.

De acordo com Kopetz e Ochsenreiter (1987) para o sistema ser tolerante a falhas bizantinas existe um custo em relação à precisão da sincronização. Esse custo é dado pela equação (2.19):

$$u(n, f) = \frac{n - 2f}{n - 3f} \tag{2.19}$$

Com a equação (2.19) é possível montar uma tabela de custo da tolerância a falhas bizantinas em relação à precisão da sincronização.

Tabela 2.2 - Dados de custo de tolerância às falhas bizantinas em relação à precisão.

Falhas f	Número de nós n									
	4	5	6	7	8	9	10	15	20	30
1	2	1,5	1,33	1,25	1,2	1,16	1,14	1,08	1,06	1,03
2				3	2	1,66	1,5	1,22	1,14	1,08
3							4	1,5	1,27	1,14
4								2,33	1,5	1,22

Fonte: Traduzida de Kopetz (1997).

É possível reescrever a equação (2.18) incluindo o custo da tolerância a falhas bizantinas da equação (2.19), tem-se então a equação (2.20) dada por Kopetz (1997).

$$\Pi(\Gamma, n, f, \xi) = \frac{n-2f}{n-3f}\left(\frac{f\Gamma}{n-2f} + \xi\right) = \left(\frac{f\Gamma}{n-2f} + \xi\right) u(n, f) \qquad (2.20)$$

É possível também aplicar ao algoritmo de Welch-Lynch o conceito de custo da falha bizantina.

Existem muitas outras funções de convergência propostas para sincronização interna de relógios. Em Schneider (1988) existem análises de várias funções de convergência diferentes.

2.3.10.3. Algoritmo PTP (Precision Time Protocol)

O algoritmo PTP é definido no padrão IEEE 1588-2002. Este padrão além de definir o algoritmo, define regras para que se alcance a sincronização consistente nos relógios de um sistema distribuído sobre uma rede de comunicação.

O algoritmo PTP (*Precision Time Protocol*) segue uma arquitetura centraliza com um algoritmo por métrica de exatidão. O algoritmo utiliza a abordagem mestre/escravo, *i.e.*, os relógios escravos recebem periodicamente o tempo do relógio mestre real, comparam seu valor e ajustam seus tempos. Os relógios

mestres, normalmente, são de boa qualidade para estabelecer uma base de tempo exata e consistente.

A Figura 2.13 mostra o esquema do algoritmo PTP.

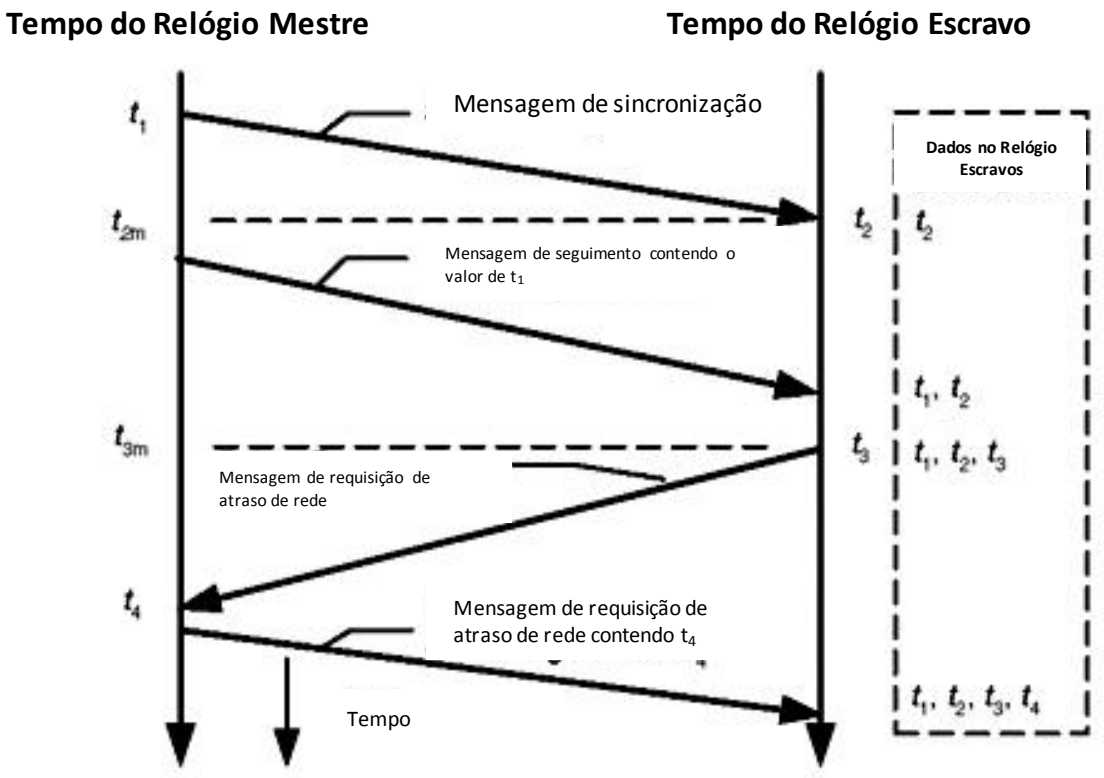


Figura 2.13 - Fluxo do Algoritmo PTP.

Fonte Adaptada de Eidson (2006).

De acordo com Eidson (2006), a seqüência básica do PTP é:

a) O relógio mestre envia periodicamente a mensagem de sincronismo a todos os relógios escravos;

b) Os relógios escravos recebem a mensagem de sincronismo, faz a estampa de tempo t_2 e grava (t_2 é a estampa de tempo do relógio escravo quando a mensagem de sincronismo é recebida);

c) O relógio mestre envia aos relógios escravos a mensagem '*follow up*' contendo o valor de t_1 (t_1 é a estampa de tempo do relógio mestre no instante que foi enviada a mensagem de sincronismo);

d) O relógio escravo grava t_1 e envia a mensagem pedindo o atraso de rede ao relógio mestre e estampa e segura o valor de t_3 (t_3 é a estampa do relógio escravo no instante que a mensagem de atraso foi enviada);

e) O relógio mestre recebe a mensagem de atraso, estampa t_4 (t_4 é a estampa de tempo do relógio mestre no instante que a mensagem de atraso foi recebida);

f) Os relógios escravos recebem a resposta da mensagem de atraso e gravam a estampa t_4. Então, os relógios escravos utilizam estes 4 tempos (t_1, t_2, t_3 e t_4) para calcular o viés instantâneo e o atraso de rede entre o relógio escravo e mestre.

Assumindo um atraso de rede simétrico entre o relógio mestre e escravo, o viés instantâneo entre o mestre e escravo é baseado nas seguintes equações:

$$offset = \frac{(t_2 - t_1) - (t_4 - t_3)}{2} \tag{2.21}$$

$$delay = \frac{(t_2 - t_1) + (t_4 - t_3)}{2} \tag{2.22}$$

A abordagem, além de não ser tolerante a erros bizantinos, tem a desvantagem de ter um ponto comum de falha, i.e., se o relógio mestre falhar então todos os relógios escravos perdem a referência. Além disso, é desejável um atraso de rede simétrico para que as equações possam ser corretamente aplicadas. As desvantagens são minimizadas com:

a) O uso de relógios mestres de boa qualidade;

b) Uso de técnicas que minimizam o ponto comum de falha;

c) Assumir em casos não críticos uma rede simétrica.

Tudo isto aumenta o custo, mas também aumenta a confiabilidade desta abordagem.

2.3.10.4. Algoritmo SR (*State-Rate Correction*)

O algoritmo SR (HANZLIK, ADEMAJ e KOPETZ, 2006) segue a mesma abordagem que o algoritmo FTM, com a diferença que o algoritmo SR propõe mais uma correção. O algoritmo propõe corrigir o MMCF (fator de conversão microtick-macrotick) do sistema. Para tanto, escolhe-se um relógio real de referência e o mesmo transmite seus dados aos relógios do conjunto para que depois sejam feitos os ajustes.

A equação abaixo define a equação de ajuste do MMCF:

$$MMCF_i(k) = MMCF_r(k) - \frac{(corr_i(k) - corr_r(k))}{mT_i(k)} \quad (2.23)$$

Em que $MMCF_i$ e $MMCF_r$ são os fatores de conversão no instante k dos relógios i e do relógio escolhido como referência r para este modo respectivamente; $corr_i$ e $corr_r$ são os valores de correção do modo nominal de viés instantâneo no instante k e mT_i é o valor do macrotick no instante k.

2.3.11. Características e Indicadores de Desempenho de Um Algoritmo de Sincronização de Relógios

Características de um algoritmo de sincronização são o conjunto de parâmetros que distinguem as qualidades desse algoritmo sobre seus semelhantes.

As principais características de um algoritmo de sincronização são:

a) **Arquitetura**: Distribuída ou Centralizada;

b) **Camada**: onde se definem os diferentes níveis que devem ser sincronizados (físico, hardware e/ou software);

c) **Métrica**: estabelece o tipo de sincronização que o algoritmo de relógios deve proceder (baseado em exatidão e/ou precisão). Os algoritmos tradicionais, como o Welch-Lynch (WL), o *State-Rate Correction* (SR) e o PTP, trabalham com duas diferentes e conflitantes métricas, são elas a precisão (*precision*) e a exatidão (*Accuracy*). A explanação da diferença entre as duas métricas está descrita na seção 2.3.6. A Figura 2.14 ilustra, de forma vetorial, as diferenças métricas entre 3 tradicionais algoritmos.

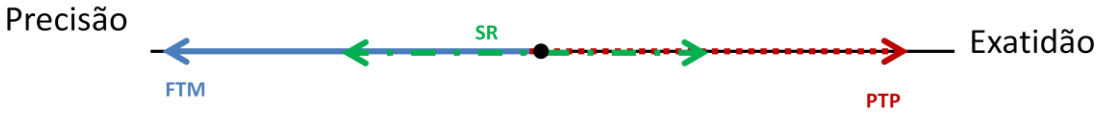


Figura 2.14 – Métricas dos Algoritmos FTM, SR e PTP.

Pode-se observar da Figura 2.14 que os algoritmos de sincronização WL, SR e PTP têm como métrica somente a sincronização de relógios, ou seja, abordam a precisão e a exatidão. O WL tem como métrica a precisão, o PTP a exatidão e o SR possui métrica mista, ou seja, procura abordar as duas ao mesmo tempo;

d) **Tolerante a Falhas Bizantinas**: se o algoritmo considera ou não os mecanismos necessários para evitar falhas bizantinas durante o processo de sincronização;

e) **Atraso de Comunicação**: estabelece como o atraso de comunicação é tratado (probabilístico e/ou pior-caso);

f) **Imperfeições**: para se definir qual a técnica e modelos é necessário definir as imperfeições (parâmetros) para o qual o sistema será todo

definido. As principais imperfeições que um algoritmo leva em consideração são: deriva, *offset* inicial, descontinuidade de tempo, *jitter*. A Figura 2.15 estabelece a relação *algoritmo* ↔ *imperfeições* aos quais os algoritmos Welch-Lynch (WL), *State-Rate Correction* (SR) e o PTP são definidos. Ou seja, cada vetor da Figura 2.15 indica um algoritmo e geometricamente indica as principais métricas abordadas por cada algoritmo.

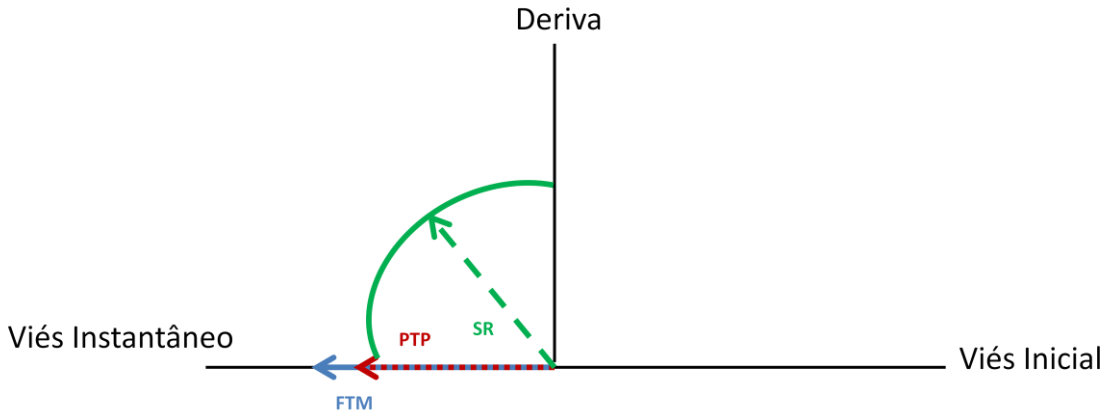


Figura 2.15 – Imperfeições dos Algoritmos FTM, SR e PTP.

É possível outras considerações nos algoritmos de sincronização de relógios, contudo, estas são as características mínimas que um algoritmo de sincronização deve levar em consideração.

Além disso, é importante estabelecer os indicadores de desempenho de um algoritmo, para que seja possível uma análise sobre o comportamento e desempenho do mesmo sobre o projeto. Dentre vários indicadores, os principais são:

a) Exatidão;

b) Precisão;

c) Tempo necessário para estabelecer a sincronização;

d) Descontinuidade de tempo;

e) Atraso de rede.

2.3.12. Padrões de Tempo e GNSS

É possível medir a diferença de tempo entre dois eventos e com isso, é possível ter a noção da progressão do tempo. Padrões de tempo servem para uniformizar os métodos utilizados para medir o tempo e fazer com que as medidas de tempo tenham a maior exatidão possível. Padrões de tempo são muito utilizados, principalmente, quando o sistema necessita de sincronização externa.

Muitos padrões de tempo já foram inventados. Os mais conhecidos são o UTC (Tempo Universal Coordenado) e o TAI (Tempo Atômico Internacional).

O UTC é um padrão de tempo mantido pela *Bureau International de L'Heure*. O padrão de tempo é derivado das observações astronômicas da rotação terrestre em relação ao Sol. Em 1972, o UTC substituiu o GMT (Tempo Médio de Greenwich). Por causa da rotação da Terra ser irregular, o padrão GMT se modificava com o tempo. Com isso, definiu-se que o padrão UTC deve seguir, com a maior exatidão possível, conforme o padrão de tempo TAI. O UTC não é um padrão de tempo contínuo, é necessário em alguns momentos à inserção de um segundo extra para se manter a sincronia do UTC com o fenômeno astronômico.

O TAI é um padrão de tempo que define a duração do segundo. A cada 9.192.631.770 oscilações de energia do átomo de césio-133 o relógio entende que se passou um segundo. Foi calculada pelo BIPM, Escritório Internacional

de Pesos e Medidas, na França. A intenção do TAI é definir a duração do segundo em conformidade com os efeitos astronômicos da rotação terrestre.

Dos GNSSs (*Global Navigation Satellite Systems*), o mais conhecido é o GPS (*Global Positioning System*). Este tem se tornado o principal meio para fornecer medidas de tempo no mundo, principalmente para aplicações de navegação. O GPS é formado por um conjunto de 24 satélites em orbita da Terra, cada um com um relógio atômico interno sincronizado. Com isso é possível em qualquer ponto da Terra, em qualquer instante de tempo, ter medidas de posição e tempo com uma boa exatidão.

Figura 2.16 - Exemplo de Constelação GNSS.

O GPS, apesar de ser o sistema GNSS mais conhecido, pertencente aos EUA e até a alguns anos atrás ser o único, atualmente não é mais único. Atualmente em operação também temos o GLONASS (*Globalnaya Navigatsionnaya Sputnikovaya Sistema*) operado pela Rússia; e também estão em desenvolvimento o Galileo, pela União Europeia, e o Compass (*Chinese Satellite Navigation*), pela China.

Excelentes artigos que tratam mais detalhadamente sobre esses assuntos tal como em (DAVID; NEIL e CLIFFORD, 1997).

2.4. Ferramenta de Simulação

Não existem muitas ferramentas especificas disponíveis para simulação de algoritmos de sincronização de relógios. Mas podemos citar a ferramenta de

simulação TrueTime que pode ser utilizada, desenvolvida por Henriksson, Cervin e Arzén (2002). Neste trabalho a ferramenta TrueTime é candidata a uso, pois se mostrou uma ferramenta com a possibilidade de analisar, programar e simular algoritmos de sincronização em conjunto com o sistema de controle em tempo real.

2.4.1. TrueTime

O TrueTime é um simulador baseado no Matlab/Simulink para sistemas de controle em tempo real. Com o TrueTime é possível simular o comportamento temporal de núcleos (*kernels*) de tempo real multitarefa contendo tarefas de controle e estudar os efeitos da CPU e agendamento da rede sob a performance do controle.

O *kernel* de simulação de tempo real é “event-driven” e pode manipular interrupções externas assim como detalhes de alto grau no caso de switches. Políticas arbitrárias de agendamento podem ser definidas, e tarefas de controle podem ser implementadas usando funções na linguagem C, arquivos M do Matlab ou diagramas em blocos do Simulink.

A Figura 2.17 apresenta as bibliotecas do TrueTime na forma de blocos do Simulink.

Seis modelos simples de rede podem ser simulados: CSMA/CD (e.g. Ethernet), CSMA/AMP (e.g. CAN), Round Robin (e.g. Token Bus); FDMA, TDMA (e.g. TTP), e Switched Ethernet (e.g. AFDX). Em sua versão 2.0 Beta 4, já são suportados mais dois protocolos de comunicação para simulação: FlexRay e PROFINET. O retardo de propagação é ignorado, porque este é muito pequeno em uma rede de área local. Só a simulação ao nível de pacote é suportada. Supõe-se que protocolos de níveis maiores dividem longas mensagens em pacotes menores.

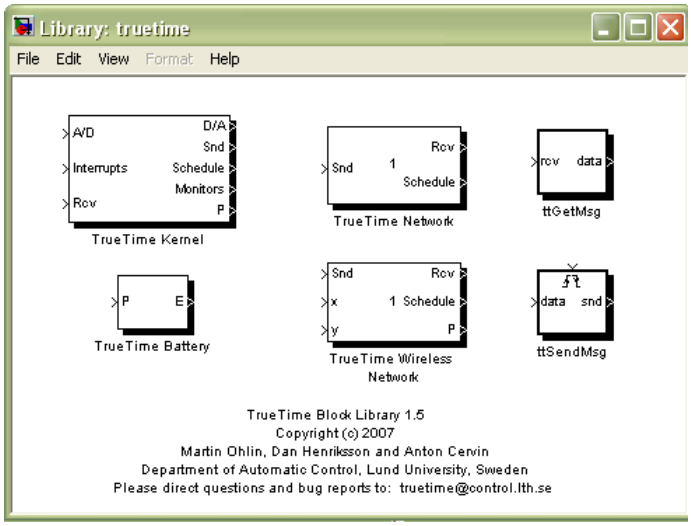


Figura 2.17 - Biblioteca do TrueTime 1.5.

O TrueTime não possui os algoritmos de sincronização implementados e prontos a serem simulados, mas o TrueTime possui funções para que se possa trabalhar com os relógios virtuais de cada processador virtual. Cada processador virtual utiliza o relógio virtual do Simulink como seu relógio de referência.

3 FORMULAÇÃO DO PROBLEMA

3.1. Algoritmo de Sincronização de Relógios

Atualmente, existem vários algoritmos para sincronização de relógios com diferentes propósitos. No entanto, a maioria não leva em conta as características do controle, i.e., o algoritmo não foi desenvolvido para operar simultaneamente com o sistema de controle.

Este trabalho estuda os efeitos da sincronização de relógios sobre o transitório e a estabilidade de sistemas de controle por rede. O trabalho propõe um algoritmo de sincronização de métrica mista (Precisão, Exatidão e Controle) corrigindo os vieses instantâneos causados pela deriva dos relógios locais reduzindo as descontinuidades de tempo durante o ajuste dos relógios, assim como, analisar e validar o algoritmo de sincronização de relógios sobre o NCS. Para isso, este trabalho utiliza teoria, controle e simulação como abordagens.

Devido à complexidade e extensão do problema, este é dividido em três partes:

a) **Descontinuidade de Tempo:** aqui se desenvolve uma técnica de amortização que reduz a descontinuidade de tempo causada por algoritmos de sincronização de relógios, com pouco prejuízo sobre a precisão e/ou exatidão;

b) **Algoritmo de sincronização por métrica mista**: aqui se desenvolve um algoritmo que engloba as métricas exatidão, precisão e controle. Em outras palavras, o algoritmo deve ser capaz de sincronizar os relógios com o tempo global e o com um relógio de referência, sem degradar muito o sistema de controle.

c) **Análise e validação do algoritmo de sincronização de relógios sobre o NCS:** aqui se aplica o algoritmo sobre um sistema de controle por redes com rede TDMA e CSMA/CD; e compara-se o

desempenho com três algoritmos da literatura: a) FTM; b) PTP; c) SR (em inglês, *State e Rate Correction*).

3.2. Sistema em Estudo

Esta seção descreve o sistema em estudo utilizado nas simulações, que é um par de motores de corrente continua (CC) interligados via rede de comunicação em forma de NCS. Neste caso, optou-se por dois protocolos de comunicação, um que utilize o TDMA como política de comunicação e outro que utilize o CSMA/CD como política de comunicação. Em Gwaltney e Briscoe (2006) o protocolo TTP (TDMA) é um dos recomendados para aplicações espaciais, já o CSMA/CD é a política usada pelo protocolo ethernet que vem sendo um dos protocolos mais usados na indústria e também alguns sistemas espaciais.

3.2.1. NCS de Segunda Ordem

Para o estudo dos casos foi utilizado um modelo de segunda ordem. O modelo é um controle de motor de corrente contínua ligado por uma rede de comunicação, do tipo TDMA ou CSMA/CD.

3.2.1.1. Motor CC

Para as simulações foi utilizado o ambiente *TrueTime/Matlab/Simulink*. Neste trabalho foram simulados dois modelos de motores CC interligados por uma rede de comunicação, sendo um com sincronismo por métrica de precisão, utilizando o algoritmo FTM, e outro por métrica de exatidão, utilizando o algoritmo PTP.

A Figura 3.1 mostra o modelo com o algoritmo FTM, onde o sensor, o atuador, o controlador e mais um bloco adicional chamado relógio mestre são conectados por uma rede de comunicação.

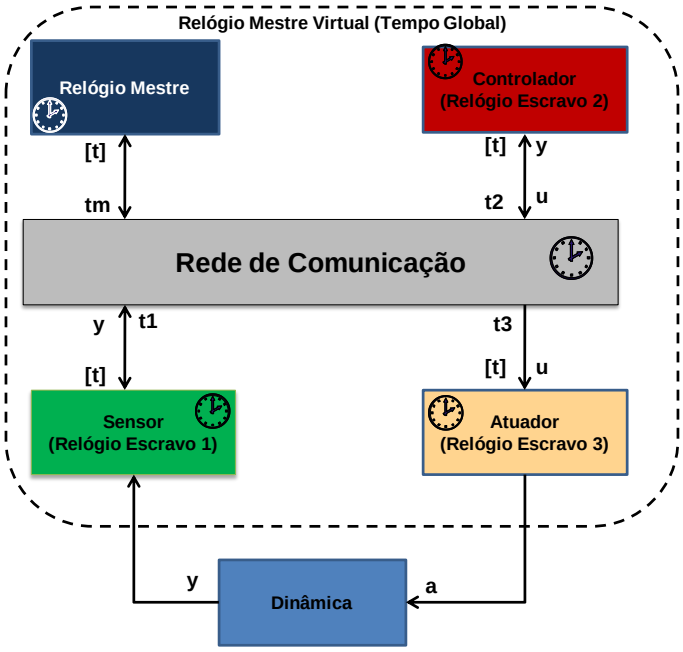


Figura 3.1 - Diagrama em blocos do modelo NCS e motores CC com algoritmo FTM.

Em que [t] é o vetor de tempo e t_m, t_1, t_2 e t_3 são as estampas de tempo locais.

O atuador/dinâmica é modelado como uma função de segunda ordem marginalmente estável, contínua no tempo:

$$G(s) = \frac{1000}{s^2 + s} \tag{3.1}$$

O controlador é um PID (Proporcional, Integral, Derivativo).

Todos os nós possuem seus relógios locais lógicos provenientes do computador virtual do *TrueTime Kernel*, utilizando um barramento de comunicação para troca dos dados.

Os tempos dos relógios locais podem ser ajustados, de acordo com o algoritmo utilizado. Para garantir que todos os nós tenham uma visão consistente do tempo, todos os nós necessitam de uma re-sincronização periódica. Para o algoritmo FTM, todos os nós trocam dados de tempo entre si. Assim, a equação (2.12) é utilizada em cada relógio para calculo da correção necessária. Com isso, todos os relógios tenderão a um valor, denominado tempo global.

A Figura 3.2 ilustra o mesmo modelo anterior, com a diferença que agora o algoritmo utilizado é o PTP.

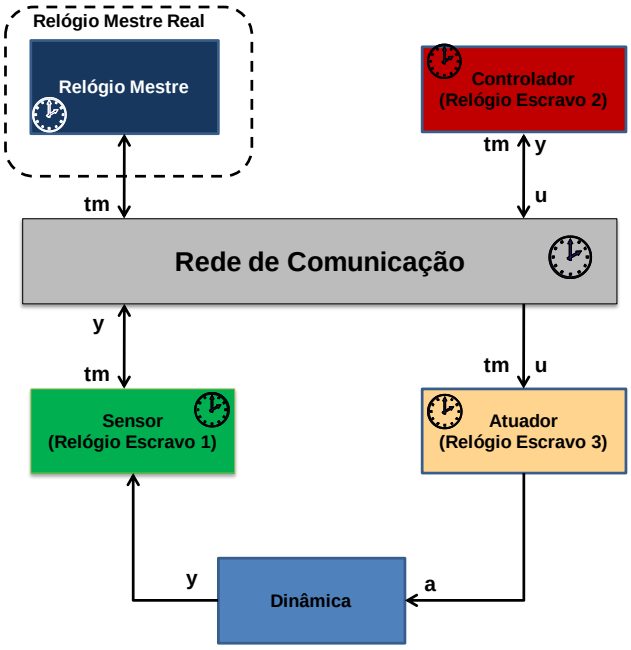


Figura 3.2 - Diagrama em blocos do modelo NCS e motores CC com algoritmo PTP.

A diferença deste modelo está no algoritmo de sincronização de relógios utilizado, em que para a sincronização e cálculo da correção, os relógios escravos recebem do relógio mestre os dados de tempo periodicamente e com isso utilizam as equações (2.21) e (2.22) para o cálculo da correção.

O objetivo destes modelos é manter a sincronização entre os nós da rede com o mínimo prejuízo para o sistema de controle do motor CC.

3.3. Abordagens para Solução

3.3.1. Metodologias

Algoritmos de sincronização de relógios podem ser estudados mediante as seguintes abordagens:

a) Teoria e Análise, especialmente de Controle;

b) Modelagem e Simulação, especialmente Computacional/Heurística;

c) Experimental.

Cada uma destas abordagens tem suas vantagens e desvantagens.

A abordagem por Teoria e Análise, tem uma de suas principais virtudes na generalidade. Porém, estabelecer equacionamentos gerais pode não ser uma tarefa fácil. Particularmente, a abordagem por Teoria de Controle tem a vantagem de usar todas as ferramentas disponíveis do mundo do controle, tais como transformadas de Laplace, Z, Fourier, e propor soluções de controle utilizando métodos clássicos. A desvantagem é que esta solução necessita de um modelo matemático bem definido.

A abordagem por Modelagem e Simulação permite a variação de parâmetros ou configurações e obter resultados rápidos sobre o impacto dessas variações. Entretanto, todas as simulações feitas vão depender do modelo implementado na simulação. Particularmente, a abordagem Computacional/Heurística visa desenvolver várias regras formais a fim de se chegar a heurísticas e regras para o problema em questão. Atualmente, vem sendo muito utilizada na teoria de relógios, pois este método é expresso em termos de pré-condições e pós-condições que são verdadeiras antes e depois do sistema executar uma tarefa; no entanto, o seu uso encapsula a solução de tal forma que, muitas vezes, há perda da generalidade da solução.

Finalmente, na abordagem por experimentos, como se trabalha diretamente com equipamentos, o nível de realismo é máximo. Entretanto, para algumas aplicações, o custo da construção de alguns protótipos pode ser inconcebível.

Neste trabalho serão utilizadas as abordagens de Teoria e Análise, especialmente de Controle; e Modelagem e Simulação, especialmente Computacional.

4 MODELOS MATÉMATICOS DE RELÓGIOS

4.1. Metodologia

Neste trabalho, optou-se por seguir a metodologia de Teoria, Análise, Modelagem e Simulação.

Existe na literatura um vasto conjunto de modelos de relógios, utilizando diferentes ferramentas matemáticas, computacionais e metodologias. Com o avanço da ciência da computação, modelos heurístico-computacionais baseados em métodos formais vêm se destacando no mundo da sincronização de relógios. No entanto estes modelos possuem algumas limitações quanto à análise matemática e à perda de generalidade.

Esta tese vai estabelecer, seguindo a metodologia de teoria, análise (especialmente controle) e modelagem e simulação, os modelos matemáticos de um relógio. O modelo de um relógio pode ser produzido de diferentes maneiras e diferentes níveis de abstração. Assim sendo, esta tese utiliza o conceito de Microtick e Macrotick, proposto por Kopetz (1997), ilustrado pela Figura 2.2.

Define-se que:

a) O Microtick é o relógio físico e a menor instância de tempo que o sistema enxerga;

b) O Macrotick é o relógio lógico baseado no microtick e a instância de mais alto nível. Nesta camada é que serão estabelecidos todos os algoritmos descritos por esta tese;

A seguir, são apresentados os modelos matemáticos de relógios utilizados no decorrer deste trabalho.

4.2. Modelo por Equações de Diferenças

Devido à propriedade monotônica do tempo e desconsiderando os efeitos de não linearidade dos osciladores locais, é possível modelar a equação de um relógio utilizando uma reta crescente.

Considerando uma deriva constante e fazendo o uso da equação reduzida de uma reta, tem-se:

$$\mu T(t) = \rho t + b \tag{4.1}$$

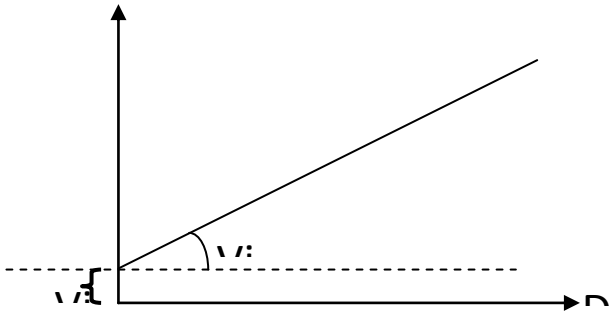


Figura 4.1 – Representação de uma reta no plano.

Fazendo a discretização da equação (4.1), tem-se:

$$t = kT \tag{4.2}$$

em que, k é o instante e T é o período de amostragem.

Tem-se assim, usando a taxa de deriva ρ_i constante há todos os instantes e o correspondente b_i, a equação discretizada:

$$mt_i(k) = \rho_i(k)T + b_i \tag{4.3}$$

Em que o i subscrito indica o relógio i do conjunto. Assim, fazendo (k+1) na equação (4.3), tem-se:

$$\mu T_i(k+1) = \rho_i(k+1)T + b_i \tag{4.4}$$

Expandindo:

$$\mu T_i(k+1) = \rho_i kT + \rho_i T + b_i \tag{4.5}$$

Substituindo (4.3) em (4.5), tem-se:

$$\mu T_i(k+1) = \mu T_i(k) + \rho_i T \tag{4.6}$$

A equação (4.6) é a equação de diferenças do microtick de um relógio local no domínio dos números reais.

No entanto, Kopetz *et. al* (2006), apresenta uma unidade tempo com uma instancia de abstração em um nível mais alto. Esta abstração do tempo, conforme explicado na seção 2.3.4, apresenta diversas vantagens. Esta abstração do tempo é denominada de macrotick. Seu modelo, baseado no microtick é dado por:

$$mT_i(k) = \frac{\mu T_i(k)}{MMCF_i} \tag{4.7}$$

Em que MMCF é o fator de conversão entre o microtick e o macrotick (*Microtick-Macrotick Conversor Factor*, em inglês). Este fator de conversão, no caso, funciona como um divisor de frequências. O que se observa na equação (4.7) é que o Macrotick tem sua evolução em uma frequência própria, diferente do microtick, mas dependente do mesmo. O Apêndice A, demonstra as a razão do MMCF e as condições necessárias para o MMCF.

Fazendo (k+1) na equação (4.7), tem-se:

$$mT_i(k+1) = \frac{\mu T_i(k+1)}{MMCF_i} \tag{4.8}$$

Substituindo (4.6) em (4.8), tem-se:

$$mT_i(k+1) = \frac{\mu T_i(k) + \rho_i T}{MMCF_i} \tag{4.9}$$

Fazendo (4.7) em (4.9), tem-se:

$$mT_i(k+1) = mT_i(k) + \frac{\rho_i T}{MMCF_i} \tag{4.10}$$

Da equação (4.6), fazendo uma simples operação aritmética chega-se em:

$$\rho_i T = \mu T_i(k+1) - \mu T_i(k) \tag{4.11}$$

Substituindo (4.11) em (4.10), chega-se a equação de diferenças do Macrotick no domínio dos reais:

$$mT_i(k+1) = mT_i(k) + \frac{\mu T_i(k+1) - \mu T_i(k)}{MMCF_i} \tag{4.12}$$

A equação (4.12) é a equação de diferenças do macrotick de um relógio local no domínio dos reais. No entanto, a segunda parte da equação (4.12) é incrementada em período diferente de T. Portanto, considerando a deriva constante, é necessária a seguinte modificação:

$$mT_i(k+1) = mT_i(k) + \frac{[\mu T_i(k+1) - \mu T_i(k - MMCF_i)] \cdot \text{Ш}}{MMCF_i} \tag{4.13}$$

Onde Ш é definido como um trem de impulsos (chave), tendo o seu modelo matemático definido como:

$$\text{Ш} = \begin{cases} \sum_{k=1}^{N} \delta(t - kT \cdot MMCF_z) \\ 0, \qquad se\ k \leq 0 \end{cases} \tag{4.14}$$

O Apêndice A, demonstra a equação geral do macrotick para o caso de uma deriva não constante.

4.3. Equação de Diferenças do Macrotick via PWM

O modelo do macrotick definido em (4.13) possui um trem de impulsos que amostra a segunda parte da equação. O termo $MMCF_i$ é o fator de conversão do Microtick para o Macrotick, em outras palavras é o divisor de frequências. No entanto, existe outra maneira de se modelar esta divisão de frequências através do uso da Modulação por Largura de Pulso (PWM, em inglês *Pulse Width Modulation*). A Figura 4.2 ilustra um sinal PWM.

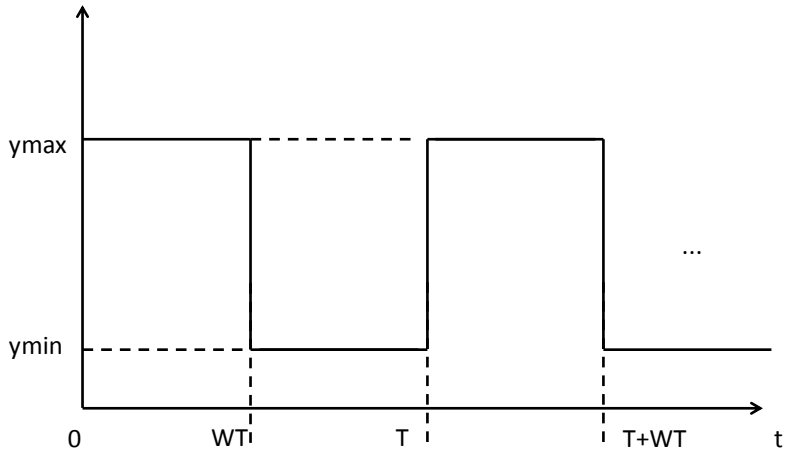


Figura 4.2 – Sinal PWM.

O PWM apresentado na Figura 4.2, apresenta um modelo matemático não linear, pois a sua largura de pulso varia de acordo com o número de microticks. No entanto, sendo W a largura da forma de onda e y_{max} e y_{min} respectivamente os valores máximo e mínimo de y; e supondo que a largura dos pulsos (W) do PWM corresponde a certo número de microticks e fazendo ymax = $1/MMCF_i$ e ymin =0, pode-se então, reescrever a equação (4.13) do macrotick utilizando PWM da seguinte maneira:

$$mT_i(k+1) = mT_i(k) + \mu T_i(k) \cdot \mathrm{y} \tag{4.15}$$

A equação (4.15) descreve a equação de diferenças do macrotick utilizando PWM. Este modelo foi apresentado para fins de futuras implementações e não será utilizado no decorrer do trabalho.

4.4. Modelo no Domínio Z

As equações (4.6) e (4.13), representam as equações de diferenças de um relógio com deriva constante. Dada estas equações, é possível encontrar o modelo na transformada Z.

Da equação (4.6) tem-se:

$$\mathrm{MiT_i(z)} = \mathrm{Z}[\ \mu T_i(k)] \tag{4.16}$$

Utilizando a propriedade do deslocamento tempo e linearidade da transformada Z chega-se na seguinte equação do microtick no domínio Z:

$$MiT_i(z) = \frac{\rho_i T}{z-1} \tag{4.17}$$

Seguindo o mesmo procedimento para a equação (4.13), chega-se a seguinte equação do macrotick no domínio Z:

$$MaT_i(z) = MiT_i(z)\frac{\left(1 - z^{-(1+MMCF_i)}\right) * \text{Ш}}{z^{-1}(z-1)MMCF_i} \tag{4.18}$$

Para fins de análise o modelo da equação (4.18) se mostra muito melhor, já que existe uma gama de ferramentas matemáticas que auxiliam nesta análise.

4.5. Modelo no Espaço de Estados

As equações de diferenças acima demonstradas, por analogia, podem ser modeladas como um sistema de controle discreto no espaço de estados.

O espaço de estados de um sistema linear de tempo discreto é dado por:

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k) + Du(k) \end{aligned} \tag{4.19}$$

Em que:

a) x(.) é o vetor de estados (*n*);

b) y(.) é o vetor de saída (dimensão *q*);

c) u(.) é o vetor de controle (dimensão *p*);

d) A é a matriz de estado (dimensão *nxn*);

e) B é a matriz de controle (dimensão *nxp*);

f) C é a matriz de saída (dimensão *qxn*);

g) D é a matriz de transmissão direta (dimensão *qxp*);

O diagrama em blocos da equação (4.19) é dado por:

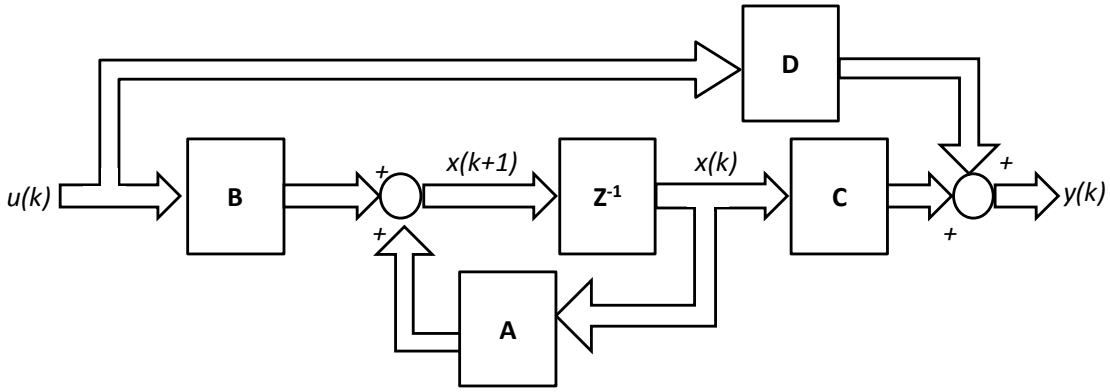


Figura 4.3 – Diagrama em blocos de um espaço de estados de um sistema linear de tempo discreto.

Para modelar a equação de diferenças (4.6) do microtick no espaço de estados, as seguintes suposições são feitas:

a) A função correção possui um valor constante entre os instantes [*RkT,R(k+1)T]* (intervalo de transição);

b) A deriva (ρ(k)) é constante em todos os instantes [*0,(k+n)T*];

Assim, com estas suposições o modelo em espaço de estados do microtick um relógio fica:

$$\begin{bmatrix} \mu T(k+1) \\ \rho(k+1) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mu T(k) \\ \rho(k) \end{bmatrix} + \begin{bmatrix} corr \\ 0 \end{bmatrix} u(k)$$

$$y(k) = \begin{bmatrix} 1 & T \end{bmatrix} \begin{bmatrix} \mu T(k) \\ \rho(k) \end{bmatrix} \tag{4.20}$$

Reescrevendo a equação (4.20) para *n* microticks de relógios, tem-se:

$$\begin{bmatrix} \mu T_1(k+1) \\ \rho_1(k+1) \\ \mu T_2(k+1) \\ \rho_2(k+1) \\ \vdots \\ \mu T_n(k+1) \\ \rho_n(k+1) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & & 0 & 0 \\ 0 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 1 & & 0 & 0 \\ 0 & 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & & \vdots & & \ddots & \vdots & \\ 0 & 0 & 0 & 0 & & 0 & 1 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} \mu T_1(k) \\ \rho_1(k) \\ \mu T_2(k) \\ \rho_2(k) \\ \vdots \\ \mu T_n(k) \\ \rho_n(k) \end{bmatrix} + \begin{bmatrix} corr_1 & \cdots & \cdots & 0 \\ 0 & \cdots & \cdots & 0 \\ 0 & corr_2 & \cdots & 0 \\ & \vdots & & \\ & \vdots & & \\ & \vdots & & \\ 0 & 0 & \cdots & corr_p \end{bmatrix} \begin{bmatrix} f_1(k) \\ f_2(k) \\ \vdots \\ f_p(k) \end{bmatrix}$$

$$\begin{bmatrix} y_1(k) \\ y_2(k) \\ \vdots \\ y_q(k) \end{bmatrix} = \begin{bmatrix} 1 & T & 0 & \cdots & 0 & 0 \\ 0 & \cdots & 1 & T & 0 & 0 \\ \vdots & \cdots & \vdots & \ddots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 1 & T \end{bmatrix} \begin{bmatrix} \mu T_1(k) \\ \rho_1(k) \\ \mu T_2(k) \\ \rho_2(k) \\ \vdots \\ \mu T_n(k) \\ \rho_n(k) \end{bmatrix} \tag{4.21}$$

Contudo, este modelo refere-se somente ao modelo do microtick. É possível ampliar o modelo adicionando-se os estados para modelar corretamente o macrotick.

Adicionando à equação (4.20) os estados do macrotick mais o fator de conversão microtick-macrotick (MMCF) tem-se:

$$\begin{bmatrix} \mu T(k+1) \\ \rho(k+1) \\ mT(k+1) \\ P(k+1) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & \alpha & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mu T(k) \\ \rho(k) \\ mT(k) \\ P(k) \end{bmatrix} + \begin{bmatrix} c1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & c2 & 0 \\ 0 & 0 & c3 \end{bmatrix} \begin{bmatrix} u1(k) \\ u2(k) \\ u3(k) \end{bmatrix} \tag{4.22}$$

$$y(k) = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & \beta & 1 & 0 \end{bmatrix} \begin{bmatrix} \mu T(k) \\ \rho(k) \\ mT(k) \\ P(k) \end{bmatrix} + \begin{bmatrix} corr1 & 0 & 0 \\ 0 & corr2 & 0 \end{bmatrix} \begin{bmatrix} u1(k) \\ u2(k) \\ u3(k) \end{bmatrix}$$

Em que:

a) T: Período de amostragem;

b) α=1/P(k)

c) β=T/MMCF

d) P(0) = MMCF

e) c1, c2, c3 = são os ganhos da matriz B;

f) corr1, corr2 = são os ganhos da matriz D;

O modelo da equação (4.22) é um modelo para um relógio com 1 microtick e 1 macrotick. Pode-se perfeitamente generalizar, assim como na equação (4.21), para um modelo de n relógios. O modelo generalizado, portanto, fica:

$$\begin{bmatrix} \mu T_1(k+1) \\ \rho_1(k+1) \\ mT_1(k+1) \\ P_1(k+1) \\ \vdots \\ \mu T_n(k+1) \\ \rho_n(k+1) \\ mT_n(k+1) \\ P_n(k+1) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & \alpha_1 & 0 & 0 & & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \cdots & 0 & 0 & 0 & 0 \\ \vdots & & \vdots & & & & \ddots & \vdots & \\ 0 & 0 & 0 & 0 & & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & \alpha_n & 0 & 0 \\ 0 & 0 & 0 & 0 & & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mu T_1(k) \\ \rho_1(k) \\ mT_1(k) \\ P_1(k) \\ \vdots \\ \mu T_n(k) \\ \rho_n(k) \\ mT_n(k) \\ P_n(k) \end{bmatrix} + \begin{bmatrix} corr_1 & \cdots & \cdots & 0 \\ 0 & \cdots & \cdots & 0 \\ 0 & corr_2 & \cdots & 0 \\ & & \vdots & \\ & & \vdots & \\ & & \vdots & \\ 0 & 0 & \cdots & corr_p \end{bmatrix} \begin{bmatrix} u_1(k) \\ u_2(k) \\ \vdots \\ u_p(k) \end{bmatrix} \tag{4.23}$$

$$\begin{bmatrix} y_1(k) \\ y_2(k) \\ \vdots \\ y_q(k) \end{bmatrix} = \begin{bmatrix} 1 & T & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & \beta_1 & 1 & & 0 & 0 & 0 & 0 \\ 0 & \ldots & \vdots & \ldots & 0 & 0 & 0 & 0 \\ \vdots & \ldots & & \ddots & 0 & 0 & 0 & 0 \\ 0 & 0 & \vdots & 0 & 1 & T & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 & \beta_n & 1 & 0 \end{bmatrix} \begin{bmatrix} \mu T_1(k) \\ \rho_1(k) \\ mT_1(k) \\ P_1(k) \\ \vdots \\ \mu T_n(k) \\ \rho_n(k) \\ mT_n(k) \\ P_n(k) \end{bmatrix} + \begin{bmatrix} corr_1 & \ldots & \ldots & 0 \\ 0 & \ldots & \ldots & 0 \\ 0 & corr_2 & \ldots & 0 \\ & \vdots & & \\ & \vdots & & \\ & \vdots & & \\ 0 & 0 & \ldots & corr_p \end{bmatrix} \begin{bmatrix} u_1(k) \\ u_2(k) \\ \vdots \\ u_p(k) \end{bmatrix}$$

Com este modelo, é possível aplicar toda a teoria de controle discreto moderno, abrindo assim várias possibilidades para análise e projeto para o amortecimento e sincronização de relógios.

5 ESTUDO DA DESCONTINUIDADE DE TEMPO E DESENVOLVIMENTO DE UMA TÉCNICA DE AMORTIZAÇÃO

5.1. Estudo da Descontinuidade de Tempo

Em sistemas distribuídos, tal como um sistema NCS, algoritmos de sincronização de relógios são desenvolvidos para especificar meios, de forma consistente, de se obter, ou minimizar a de-sincronização dos relógios. Assim, é possível estabelecer uma base de tempo sincronizada entre os nós da rede. Contudo, freqüentemente os algoritmos de sincronização de relógios, no processo de ajustes, forçam os relógios a voltar ou avançar no tempo a cada instante de re-sincronização, desrespeitando assim sua propriedade monotônica. Este fenômeno é conhecido como descontinuidade de tempo.

A Figura 5.1 e Figura 5.2 ilustram os dois tipos de descontinuidade de tempo, o avanço e o atraso no tempo, respectivamente.

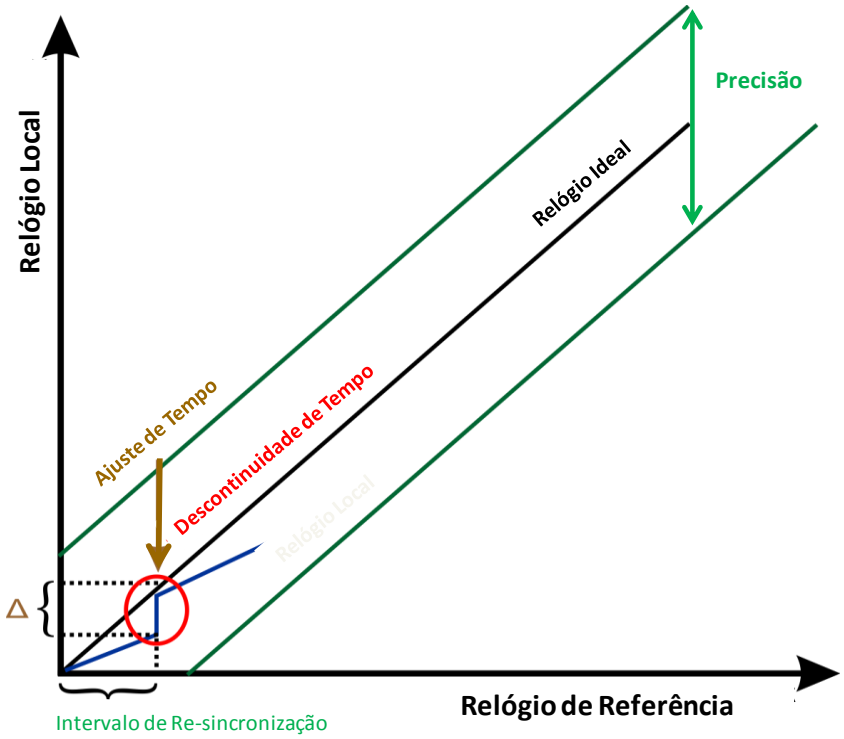


Figura 5.1 - Descontinuidade de Tempo - Avanço no Tempo.

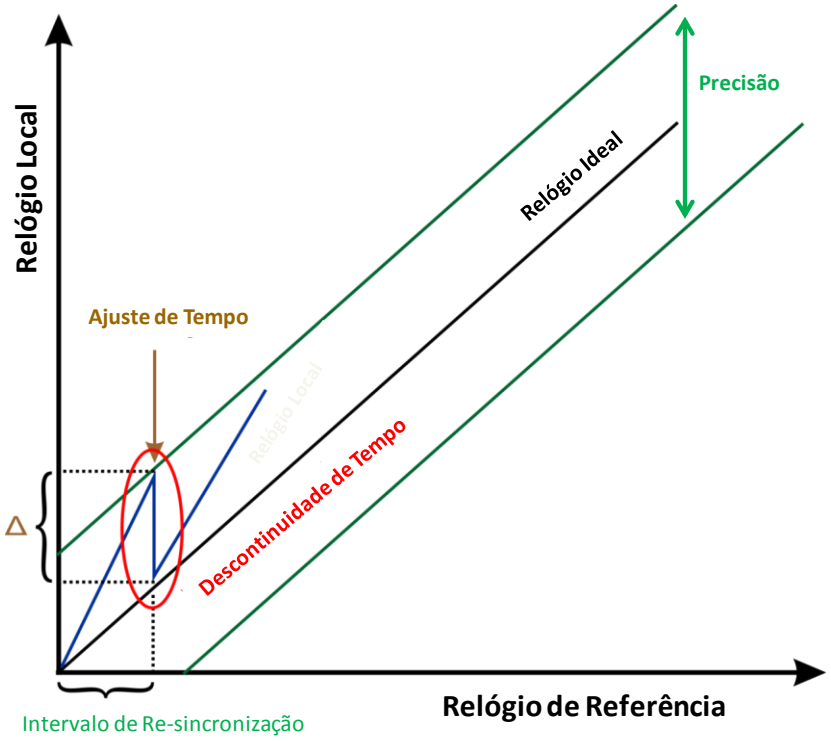


Figura 5.2 - Descontinuidade de Tempo - Avanço no Tempo.

De acordo com Ryu, Park e Hong (1999), a descontinuidade de tempo pode causar confusões em tempo de execução e fazer com que as tarefas de tempo real tenham uma falha de julgamento. Logo, pode levar a falhas ou estados indesejáveis. Em Ryu, Park e Hong (1999) são classificados dois tipos de restrição:

a) **Desaparecimento**: Esta restrição é devida ao avanço do tempo. Este avanço pode ter efeitos em nível de computação e/ou de comunicação, causando indiretamente problemas ao controle. No nível da computação, uma tarefa pode achar, erroneamente, que houve perda de seu *deadline* e assim perder a chance de ser despachada causando uma abertura temporária da malha de controle, como ilustra a Figura 5.3. No nível da comunicação, uma tarefa pode avançar no tempo, prejudicando a comunicação que por conseqüência: 1) aumenta a incerteza (*jitter*) e o atraso de tempo

(*delay*); e 2) diminui o determinismo (*predictability*) e o desempenho do sistema (*performance*), como ilustra a Figura 5.4.

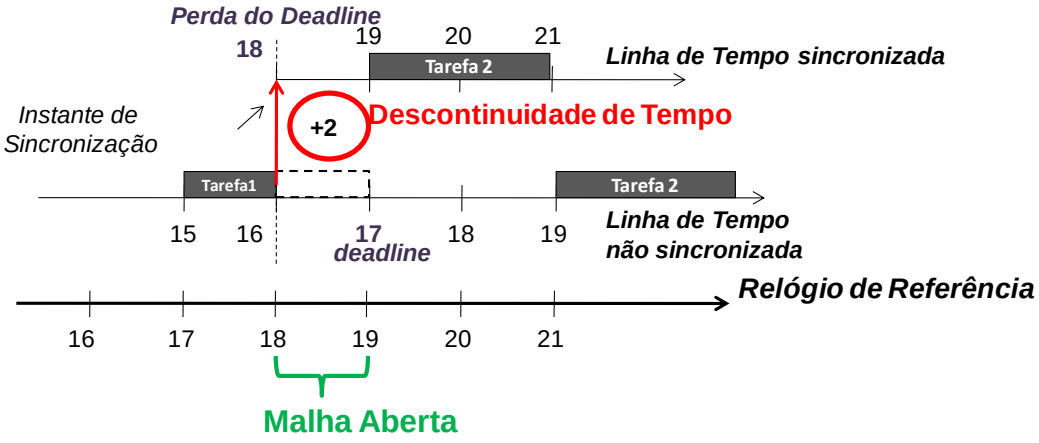


Figura 5.3 - Efeitos do avanço no tempo no nível da computação.

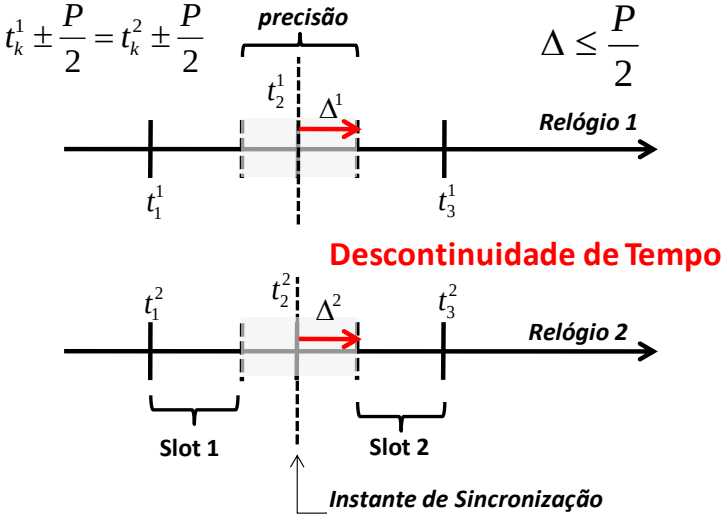


Figura 5.4 - Efeitos do avanço no tempo no nível da comunicação.

b) **Reaparecimento**: Esta restrição é devida ao atraso do tempo. Este atraso pode ter efeitos em nível de computação e/ou de comunicação, causando indiretamente problemas ao controle. No nível da computação, uma tarefa é despachada e pode erroneamente ser

despachada novamente. No entanto, sistemas de tempo real do tipo *hard-real time* uma vez a tarefa enviada, a tarefa não é reenviada, causando uma abertura temporária da malha de controle, como ilustra a Figura 5.5. No nível da comunicação, uma tarefa pode atrasar no tempo, prejudicando a comunicação que por conseqüência: 1) aumenta a incerteza (*jitter*) e o atraso de tempo (*delay*); e 2) diminui o determinismo (*predictability*) e o desempenho do sistema (*performance*), como ilustra a Figura 5.6.

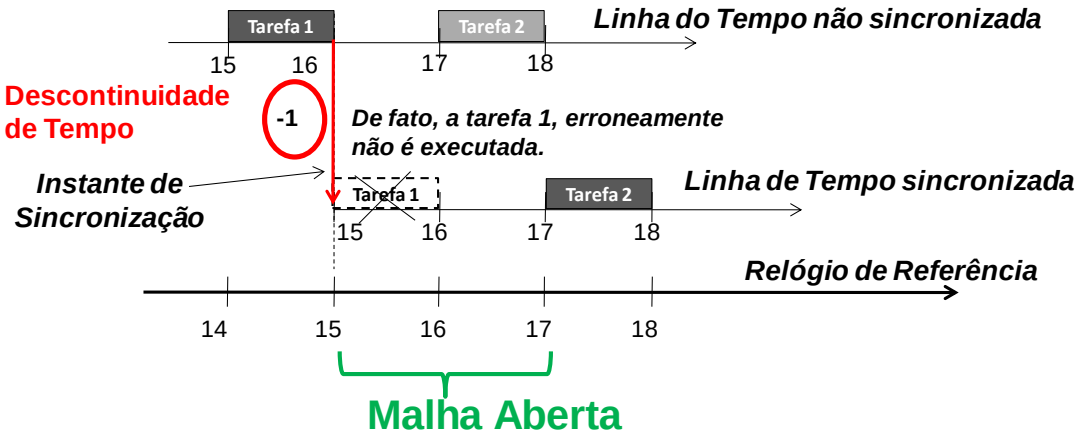


Figura 5.5 - Efeitos do atraso no tempo no nível da computação.

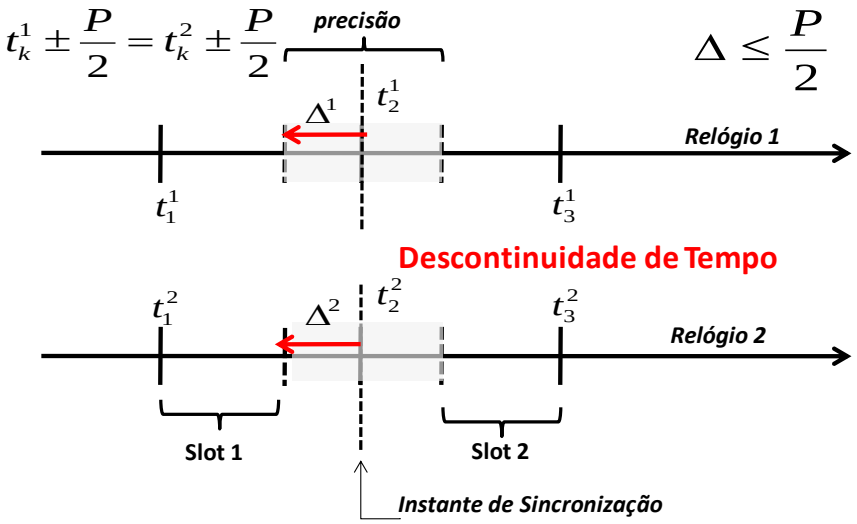


Figura 5.6 - Efeitos do atraso no tempo no nível da comunicação.

5.2. Técnica de Amortização - "*Cross-Fading*"

A técnica do "*cross-fading*" é uma técnica que visa fazer a transição suave entre diferentes sinais. De acordo com Amaral (2013), o método de "*cross-fading*" consiste em usar ambos os sinais de controle durante uma fase de transição, ou seja, o método consiste em gerar uma função que multiplica as funções durante um tempo determinado fazendo a transição suave de sinais.

A Figura 5.7 mostra um exemplo de transição entre sinais de controle, utilizando o "*cross-fading*".

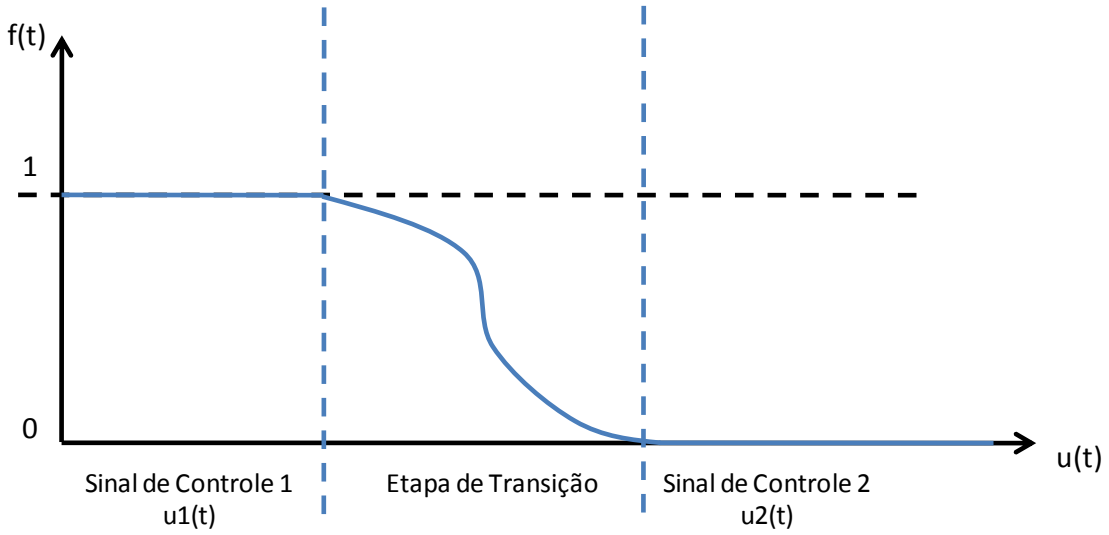


Figura 5.7 –Gráfico da função de multiplexação a(t) genérica ao longo do tempo.

Fonte Adaptada de Amaral (2013)

Da Figura 5.7, pode-se considerar que o sinal resultante final é u(t), dado pela equação:

$$u(t) = u_1(t)f(t) + u_2(t)[1 - f(t)] \quad (5.1)$$

A função f(t) varia seu valor de 0 a 1 de forma suave e pré-estabelecida. A forma da função, o tempo de transição e outras características dependem da planta, dos requisitos de transição e do ambiente disponível.

No entanto, este exemplo utilizado em Amaral (2013) considera a presença de dois sinais diferentes (u_1 e u_2) com uma transição entre eles. Mas, supondo que:

$$u_1(t) = u_2(t) + corr(t) \tag{5.2}$$

Onde corr(t) é uma função que faz uma correção no sinal u_2(t), e substituindo a equação (5.2) em (5.1), tem-se, portanto, como resultado de saída:

$$u(t) = u_2(t) + corr(t)f(t) \tag{5.3}$$

Da equação (5.3), pode-se concluir que a técnica do "*cross-fading*" além de ser aplicada à transição de sinais com diferentes fontes, pode ser aplicada também a sinais individuais que sofrem correções ao longo do tempo.

Algoritmos de sincronização de relógios possuem a característica de um sinal que sofre correções periódicas (que geram descontinuidades no sinal) ao longo do tempo. Com isto, o "cross-fading" pode ser utilizado como técnica de amortização de descontinuidades de tempo causadas pelas correções de algoritmos de sincronização. Portanto, isto dá base para separar em pelo menos duas etapas o desenvolvimento e projeto de algoritmos de sincronização de relógios: 1) Desenvolver o algoritmo que faz a correção temporal (*corr(t)*); 2) Desenvolver a técnica que amortiza as descontinuidades de tempo causadas pela correção (*f(t)*).

5.2.1. Modelo Teórico

Para modelar a propagação do sinal de um relógio, será utilizado um modelo linear de primeira ordem, considerando a presença de duas imperfeições: 1) deriva; 2) viés inicial. Este modelo é dado pela equação de uma reta, dada por:

$$c(t) = D \cdot t + b \tag{5.4}$$

Em que:

a) c(t) - é a medida do tempo no instante t;

b) D - é a deriva;

c) t - é a medida do tempo de referência;

d) b - viés inicial;

Este modelo é contínuo, no entanto relógios têm a característica de possuírem sinais discretos. Considerando uma deriva constante e fazendo a discretização com t=kT e t=(k+1)T, onde k é a amostra e T o período de amostragem e fazendo algumas manipulações algébricas chega-se a equação de diferenças do relógio dada por:

$$c(k+1) = c(k) + D(k)T \tag{5.5}$$

Para k=0,1,2,3.....N;

Adicionando à função de correção de um algoritmo de sincronização de relógios, a equação (5.5) fica:

$$c(k+1) = c(k) + D(k)T + corr(k) \tag{5.6}$$

Onde corr(k) é um valor de correção calculado pelo algoritmo de sincronização de relógios.

A equação (5.6) representa a equação de diferenças de primeira ordem de um relógio. Assim, aplicando a equação (5.3) na equação (5.6), tem-se:

$$c(k+1) = c(k) + D(k) \cdot T + corr(k) \cdot f(k) \tag{5.7}$$

A equação (5.7) é a equação do relógio, mais a correção do algoritmo de sincronização sendo multiplicado por uma função "cross-fading".

Para todas as análises e simulações, primeiramente, será considerado um algoritmo de sincronização genérico por média. Isto é, os relógios se sincronizam baseado na média do conjunto. A equação da correção fica:

$$corr(k+1) = \begin{cases} corr(k), & Fora\ do\ Instante\ de\ Resincronização \\ \sum_{i=1}^{N} \frac{c_i(k)}{N}, & No\ instante\ de\ Resincronização \end{cases} \quad (5.8)$$

Em que:

N - número de relógios no conjunto;

i - Relógio Local;

5.2.2. Função Rampa

Neste primeiro caso, para aplicar um amortecimento da correção ao longo do tempo, foi modelada uma função rampa.

A função no domínio discreto fica na forma:

$$f(k+1) = f(k) - \frac{G}{(R-M)}$$

$$Condição\ Inicial:\ \ f(0) = G \quad (5.9)$$

$$para\ k = 0,1,2,\dots n.$$

Em que T é o período de amostragem, G é um valor constante (denominado ganho), R é o intervalo de re-sincronização do relógio local e M é o período de tempo até o chaveamento. A Figura 5.8 ilustra o caso geral.

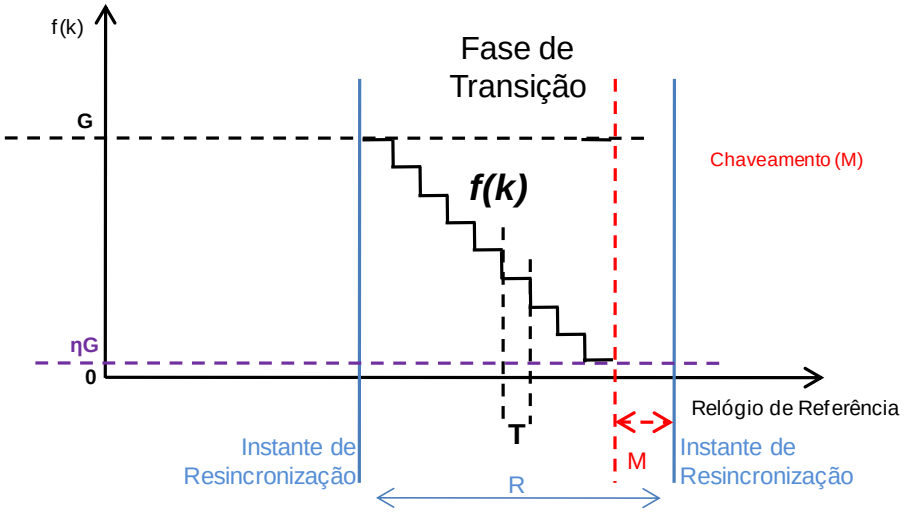


Figura 5.8 –Gráfico da função "*cross-fading*" rampa.

5.2.2.1. Métricas de Projeto

Para o projeto do *"cross-fading"* rampa o projetista precisa levar em conta as seguintes métricas:

a) Ganho (G) - O ganho é o estado inicial da função rampa. O ganho define qual é a parcela de contribuição que a correção faz inicialmente e, a partir dai, há um decaimento linear desta contribuição;

b) Intervalo de re-sincronização (R) - O intervalo de tempo entre os instantes de re-sincronização. Este parâmetro é proveniente do algoritmo de sincronização de relógios. É um número inteiro, medido em *ticks* do relógio;

c) Chaveamento (M) - Devido à função rampa ser uma função monótona estritamente crescente e/ou estritamente decrescente, o chaveamento da função é uma condição necessária. O parâmetro é um número inteiro medido em *ticks* do relógio.

d) Zero (ηG) - Dadas as possibilidades de erros provenientes de um sistema discreto, tal como os de quantização, este parâmetro serve

para indicar ao sistema o estado considerado zero. O η é dado em porcentagem. O parâmetro fica a critério do projetista e do sistema.

As simulações e resultados são mostradas mais à frente no capítulo 7.

5.2.3. Função Exponencial

Neste segundo caso, para aplicar um amortecimento da correção ao longo do tempo, foi modelada uma função exponencial.

A função no domínio discreto fica na forma:

$$f(k+1) = f(k) \cdot e^{-(R+M)T}$$

$$Condição\ Inicial\colon\ \ f(0) = G \qquad (5.10)$$

$$para\ k = 0,1,2,\dots n.$$

Em que T é o período de amostragem, G é um valor constante (denominado ganho), R é o intervalo de re-sincronização do relógio local e M é o instante que a função chega a 5% de G.

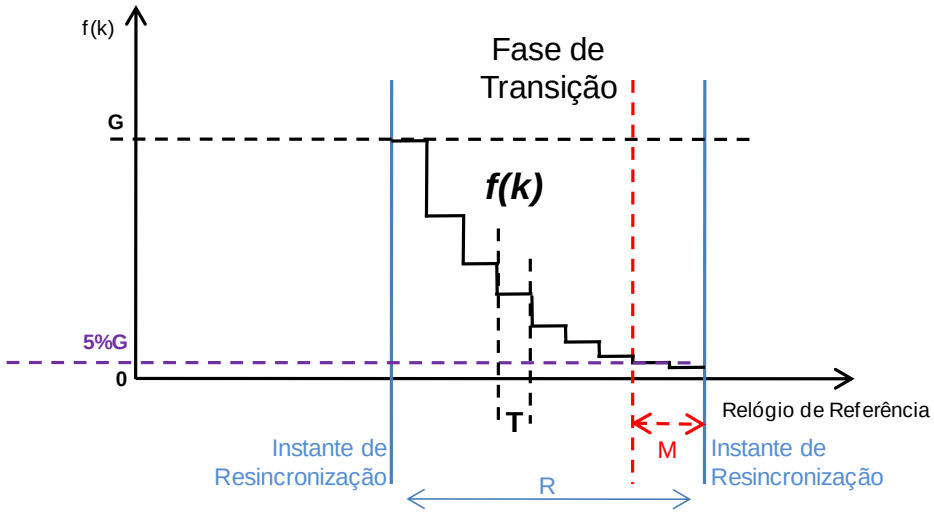


Figura 5.9 –Gráfico da função "*cross-fading*" exponencial.

5.2.3.1. Métricas de Projeto

Para o projeto do *"cross-fading"* por decaimento exponencial, o projetista precisa levar em conta as seguintes métricas:

a) Ganho (G) - O ganho é o estado inicial da função exponencial. O ganho define qual é a parcela de contribuição que a correção faz inicialmente e a partir dai há um decaimento exponencial desta contribuição;

b) Chaveamento (M) - Diferentemente da função rampa, a função exponencial não necessita de chaveamento, pois seu valor tende a zero. O parâmetro é um número inteiro medido em *ticks* do relógio.

As simulações e resultados são mostradas mais a frente no capítulo 7.

6 ALGORITMO DE SINCRONIZAÇÃO DE RELÓGIOS VIA MODELAGEM DISCRETA E TRANSFORMADA Z

O problema de sincronizar relógios é um problema pervasivo a vários sistemas, existindo um repertório imenso de problemas a serem explorados. O desafio de projetar um sistema de sincronização de relógios não é somente o projeto das leis de controle, mas também a partir das leis de controle, definir formalmente e logicamente os passos necessários para que o método possa ser utilizado para qualquer sistema, desde que o mesmo obedeça às suposições pré-estabelecidas. Esta tese visa desenvolver um algoritmo para a camada de software utilizando os conceitos de microtick e macrotick estabelecidos na seção 2.3.4, além de levar em conta a influência sobre o controle para o projeto de algoritmos de sincronização de relógios.

A seção 2.3.11 mostra as características tradicionais que especificam um algoritmo de sincronização. São elas: a) Métricas; b) Imperfeições; c) Atraso de Rede; d) Arquitetura; e) Tolerância a Falhas.

Na sequência será apresentado o algoritmo de sincronização proposto caracterizando individualmente seus modos de sincronização de acordo com as características tradicionais.

6.1. Abordagem e Métricas

A abordagem via modelagem discreta e transformada Z proporciona aos algoritmos de sincronização de relógios um repertório matemático e de análise maior e mais geral, do que a tradicional abordagem heurística, para especificar meios, de forma consistente, a fim de se obter, ou minimizar, a tarefa de-sincronização dos relógios. No entanto, para se especificar o meio (controle) para se atingir um fim (sincronização), primeiramente é necessário estabelecer os objetivos, ou seja, as métricas para as quais o sistema é projetado.

Com o crescimento da integração e complexidade dos sistemas distribuídos, uma nova métrica é necessária, a do **controle**. Ou seja, os algoritmos de sincronização, antes, desenvolvidos para trabalharem para si e somente si, agora devem possuir uma nova métrica para levar em consideração o mundo real. Além de o algoritmo trabalhar para estabelecer a sincronização de relógios, o algoritmo trabalha para o controle, ou seja, suas ações devem ter uma mínima influência (e determinística) sobre o controle. A Figura 6.1, ilustra a nova métrica, proposta por esta tese, a ser inserida em projetos de algoritmos de sincronização de relógios.

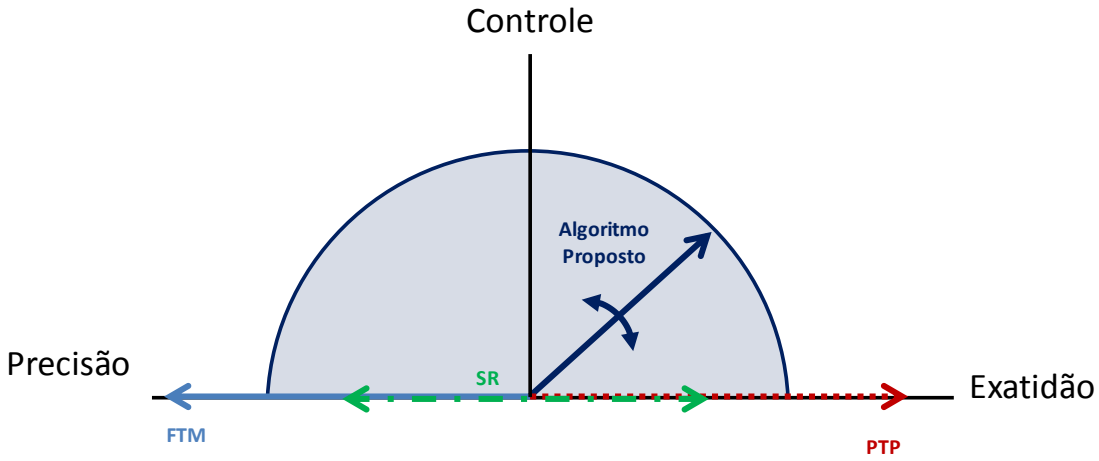


Figura 6.1 – Nova métrica para projeto de algoritmos de sincronização.

Da Figura 6.1, é importante destacar que a introdução da nova métrica, a de Controle, não exclui as antigas (Precisão e Exatidão), ou seja, a nova concepção também engloba as anteriores. Assim, os algoritmos de sincronização desta tese são elaborados sobre o preceito desta métrica proposta. As vantagens e desvantagens desta abordagem e métricas serão esclarecidas mais à frente.

6.2. Modelo de Imperfeições

Para se definir o meio (controle), além da definição das métricas, é necessário definir os parâmetros (imperfeições) para os quais o sistema será todo definido. A Figura 2.15 ilustra as imperfeições para as quais os algoritmos Welch-Lynch

(FTM), *State-Rate Correction* (SR) e o *Precision Time Protocol* (PTP) são definidos. No entanto, esta concepção não leva em consideração dois fatores que têm uma grande influência (mesmo que indireta) sobre o sistema de controle: são elas o viés inicial e a descontinuidade de tempo. Os Capítulos 5 e 7 mostram a influência que a descontinuidade de tempo tem sobre um sistema de controle.

A Figura 6.2, ilustra o novo modelo de imperfeições, proposta por esta tese, para projetos de algoritmos de sincronização de relógios.

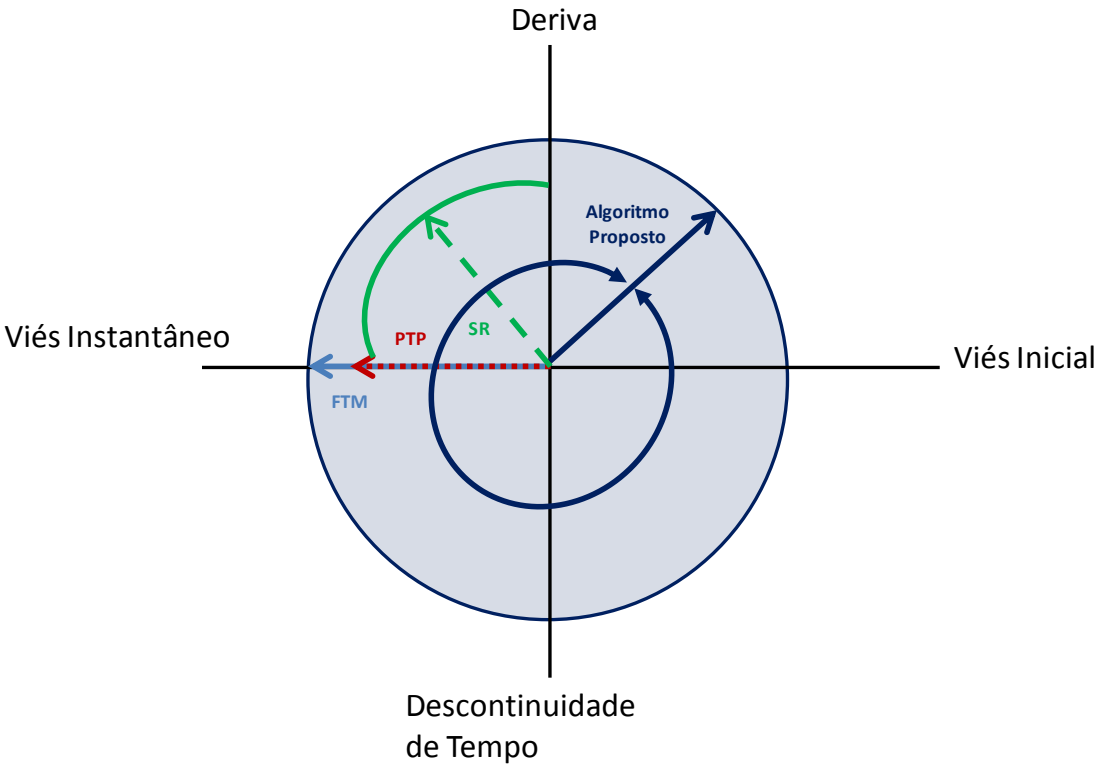


Figura 6.2 – Modelo de imperfeições para projeto de algoritmos de sincronização.

Da Figura 6.2, é bom destacar que este modelo de imperfeições é um modelo de alto nível, ou seja, é um modelo que estabelece a relação algoritmo ↔ imperfeições. A consideração e a adição da descontinuidade de tempo como uma imperfeição abre uma nova perspectiva para projetos de algoritmos de relógios, ou seja, é possível *by-design* desenvolver uma solução que leve em

conta a descontinuidade de tempo. Este modelo não exclui as antigas relações e introduz novas relações possíveis. Os algoritmos de sincronização desta tese são elaborados sobre este modelo de imperfeições proposto, onde suas vantagens e desvantagens serão esclarecidas mais à frente. É importante ressaltar, que a influência sobre a dinâmica do relógio é apresentada na Figura 2.4.

6.3. Algoritmos de Sincronização ReS

Para um sistema distribuído ser determinístico é necessário que todos os nós do sistema tenham a mesma base temporal. No entanto, cada nó tem suas imperfeições que fazem com que haja necessidade de re-sincronização periódica, conforme explica a seção 2.3.5.

Os algoritmos de sincronização propostos são denominados por esta tese como algoritmos ReS (do inglês, *Reconfigurable Smoothable*) que é um conjunto de algoritmos de sincronização de tempo que se propõem a ser reconfiguráveis e suavizáveis, com métrica mista (exatidão, precisão e controle).

Os algoritmos propostos são:

a) Algoritmo ReS: indica o conjunto de algoritmos de sincronização de tempo reconfiguráveis e suavizáveis, em que a métrica principal não é especificada e, portanto, deve ser especificada antes do projeto do algoritmo.

b) Algoritmo AReS: indica que é um algoritmo de sincronização de tempo reconfigurável e suavizável, em que a métrica principal de sincronização é a exatidão (do inglês, *Accurate*).

c) Algoritmo PReS: indica que é um algoritmo de sincronização de tempo reconfigurável e suavizável, em que a métrica principal de sincronização é a precisão (do inglês, *Precise*).

d) Algoritmo PAReS: indica que é um algoritmo de sincronização de tempo reconfigurável e suavizável, em que as métricas principais de sincronização são a exatidão e precisão.

Assim, o conjunto de algoritmos é especificado em três modos de operação, conforme o diagrama de modos da Figura 6.3.

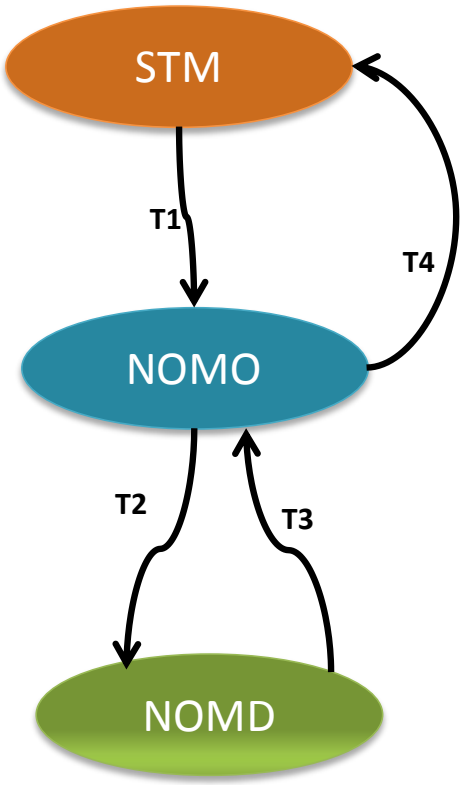


Figura 6.3 – Diagrama de Modos do Algoritmo de Sincronização Proposto.

Sendo os modos de operação:

a) **Modo STM:** No modo de inicialização (em inglês, *Startup Mode*) todos os algoritmos devem seguir como métrica principal a exatidão e a imperfeição a ser corrigida é a sincronização inicial, pois o objetivo neste modo é que todos os relógios estejam sincronizados entre si da forma mais rápida possível.

b) **Modo NOMO:** No modo nominal de viés instantâneo (em inglês, *Nominal Offset Mode*) as métricas e imperfeições dos algoritmos não são especificadas. As métricas vão depender do algoritmo e dos objetivos de cada sistema em questão. No entanto, o objetivo deste modo é corrigir de forma eficaz os vieses instantâneos do macrotick gerados pela deriva dos microtick dos relógios.

c) **Modo NOMD:** No modo nominal de deriva (em inglês, *Nominal Drift Mode*) a métrica e imperfeição dos algoritmos são respectivamente exatidão e deriva. Pois, o objetivo deste modo é ajustar os MMCF´s de forma que os macroticks do conjunto de relógios tenham seus eventos sincronizados. Ou seja, o modo visa corrigir de forma eficaz a curvatura do macrotick corrigindo os fatores de geração de macrotick em relação ao microtick de cada relógio.

Estabelecidos os modos, na sequência, serão definidas as características de cada modo juntamente com os algoritmos propostos de cada modo, dentro da metodologia escolhida. Ao final, o conjunto de algoritmos ReS será especificado dentro desta arquitetura.

6.3.1. Modo de Inicialização - STM

Muitos algoritmos de sincronização na literatura têm em sua concepção a suposição de que todos os relógios inicialmente estão sincronizados. No entanto, em vários sistemas, observa-se que isto não é uma verdade absoluta, principalmente os reconfiguráveis que estão se readaptando em vários instantes durante o ciclo de vida. O modo de inicialização tem como métrica principal a exatidão como mostra a Figura 6.4, pois visa acertar todos os relógios de maneira mais rápida possível a um relógio de referência. Ou seja, o algoritmo utilizado no modo STM tem como objetivo se sincronizar em relação a uma referência real.

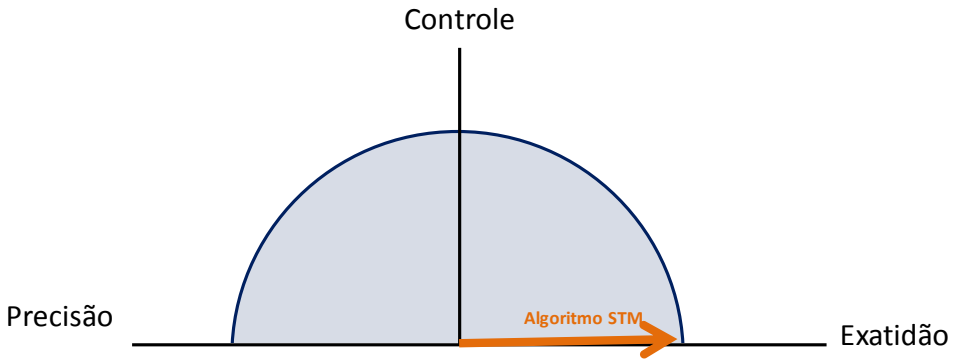


Figura 6.4 – Métrica para Algoritmo STM.

Visando este problema, aqui se propõe o uso do controlador *deadbeat* para a sincronização inicial.

Um sistema digital com o controlador *deadbeat* deve satisfazer às seguintes considerações:

- A resposta do controlador *deadbeat* é realizada somente nos instantes de amostragem e somente sobre uma entrada em particular para a qual o controlador foi projetado.

- Na realização do projeto a função de transferência do controlador *deadbeat* deve ser fisicamente realizável bem como estável.

Sobre a primeira consideração, o sistema em estudo que é a sincronização de relógios tem características bem conhecidas, ou seja, conhece-se muito o comportamento da planta, bem como a entrada. A planta de um relógio é modelada como um sistema de primeira ordem (rampa), como pode ser visto no capítulo 4. Quanto à entrada, sabe-se a priori a forma da função que também é uma rampa. No entanto, se for uma sincronização com uma referência externa, é possível conhecer e/ou estimar as características necessárias do sinal proveniente do relógio externo. Se a sincronização for uma referência virtual, será necessário o uso da teoria de tempo global, apresentado na seção 2.3.3.

Em relação à segunda consideração, sendo o controlador aplicado ao mundo virtual, podemos a priori considerar que todos os nossos controladores serão fisicamente realizáveis. Atenta-se que esta consideração em sistemas físicos não pode ser facilmente descartada.

A vantagem do controlador *deadbeat* é que, *by-design,* o controlador define o tempo necessário para o erro convergir à zero. A desvantagem é que o valor do controle pode ser muito alto.

6.3.1.1. Função de Convergência - Controlador Deadbeat

A função de transferência do microtick descritas nas equações (4.6) e (4.17) descrevem a evolução de um relógio. Como relógios foram modelados como retas, pode-se utilizar estas equações parar projetar o controlador *deadbeat* que servirá de propósito geral para o modo STM. O diagrama em blocos abaixo ilustra o sistema de controle.

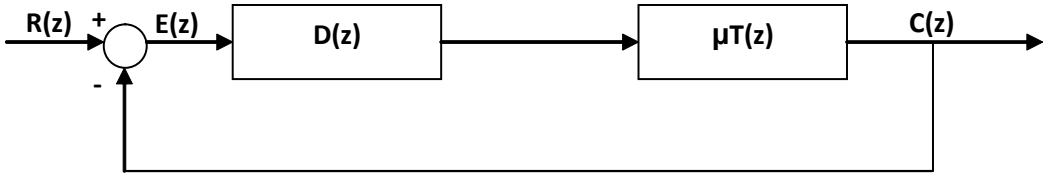


Figura 6.5 – Modelo em Malha Fechada para Controlador *Deadbeat*.

Onde D(z) é controlador *deadbeat*, μT(z) é a função de transferência do microtick, R(z) é o sinal de um relógio de referência.

A função de transferência em malha fechada do sistema é:

$$\frac{C(z)}{R(z)} = M(z) = \frac{D(z)\mu T(z)}{1 + D(z)\mu T(z)} \tag{6.1}$$

Da equação (6.1), isolando o termo do controlador, tem-se:

$$D(z) = \frac{1}{\mu T(z)} \frac{M(z)}{1 - M(z)} \tag{6.2}$$

O sinal de entrada do sistema é:

$$R(z) = \frac{A(z)}{(1 - z^{-1})^N} \tag{6.3}$$

Em que:

- A(z) é um polinômio em z-1 sem nenhum zero em z=1;
- N é um número inteiro positivo.

Assim, sendo R(z) uma rampa, tem-se que

$$A(z) = Tz^{-1} \tag{6.4}$$
$$N = 2$$

De acordo com Kuo (1977), o critério de projeto do controlador *deadbeat* é que o erro da resposta em estado estacionário a uma entrada especifica deve ser zero. Nota-se que a função de entrada pode variar de acordo com entrada. Para o projeto deste controlador, foi escolhida uma função rampa.

Assim, utilizando o teorema do valor final, tem-se:

$$\lim_{k \to \infty} e(kT) = \lim_{z \to 1} (1 - z^{-1}) E(z) = 0 \tag{6.5}$$

Do diagrama em blocos e fazendo algumas manipulações algébricas, tem-se que:

$$E(z) = R(z)[1 - M(z)] \tag{6.6}$$

Com isso, utilizando o teorema do valor final, tem-se:

$$\lim_{k\to\infty} e(kT) = \lim_{z\to 1}(1-z^{-1})\frac{A(z)[1-M(z)]}{(1-z^{-1})^N} = 0 \tag{6.7}$$

Sendo A(z) um polinômio com nenhum zero em z=1, a condição necessária para o erro zero em estado estacionário é que [1-M(z)] deve conter a parcela $(1\text{-}z^{-1})^N$. Desta forma, o critério que satisfaz o controlador *deadbeat* é:

$$1 - M(z) = (1-z^{-1})^N F(z) \tag{6.8}$$

em que F(z) é um polinômio em que z^{-1} é a variável independente, com pólos somente em z=0.

Sendo as funções de transferência de μT(z) e M(z) fisicamente realizáveis, elas podem ser expressas por uma série de potências do tipo:

$$\begin{aligned} \mu T(z) &= \mu t_n z^{-n} + \mu t_{n+1} z^{-n-1} + \cdots \\ M(z) &= m_k z^{-k} + m_{k+1} z^{-k-1} + \cdots \end{aligned} \tag{6.9}$$

Em que n≥0 e k≥0. Substituindo as duas equações na equação de D(z), tem-se:

$$D(z) = \frac{m_k z^{-k} + m_{k+1} z^{-k-1} + \cdots}{(\mu t_n z^{-n} + \mu t_{n+1} z^{-n-1} + \cdots)(1 - m_k z^{-k} - m_{k+1} z^{-k-1} + \cdots)} \tag{6.10}$$

$$D(z) = d_p z^{-p} + d_{p+1} z^{-p-1} + \cdots$$

Em que p=k-n.

Para D(z) ser fisicamente realizável, p não deve ser negativo. Então, k≥n, ou a menor potência da expansão em série de M(z) em z-1 deve ser pelo menos maior do que a menor potência em z-1 da expansão em séries de μT(z). Este é o mínimo requisito de M(z) estabelecido para uma entrada especificada. Assim, F(z) deve ser escolhido de acordo com o critério do *deadbeat* para satisfazer este requisito.

Portanto, para o sistema em estudo, tem-se que μT(z) não possui zeros e considerando que a entrada do sistema é uma rampa unitária, tem-se que a função de transferência em malha fechada tem a seguinte forma:

$$M(z) = 1 - (1 - z^{-1})^2 F(z) \tag{6.11}$$

Assim, deve-se escolher um F(z) de tal forma que a expansão em série de M(z) em z^{-1} deva iniciar com pelo menos o termo z^{-1}. Assim, escolhendo F(z)=1, tem-se:

$$M(z) = 1 - (1 - z^{-1})^2 = 2z^{-1} - z^{-2} \tag{6.12}$$

Substituindo na equação de D(z), tem-se, portanto uma função de transferência do controlador *deadbeat* para uma entrada rampa:

$$D(z) = \frac{1}{\mu T(z)} \frac{M(z)}{1 - M(z)} = \frac{(z-1)}{(\rho T - \mu t(0))} \frac{(2z^{-1} - z^{-2})}{(1 - 2z^{-1} + z^{-2})} \tag{6.13}$$

Reorganizando, tem-se:

$$D(z) = \frac{2 - 3z^{-1} + z^{-2}}{\big(\rho T - \mu t(0)\big) - 2\big(\rho T - \mu t(0)\big)z^{-1} + (\rho T - \mu t(0))z^{-2}} \tag{6.14}$$

Com a função de transferência do controlador, agora é possível achar a equação de diferenças do controlador. Para isso, são feitas algumas manipulações algébricas.

A equação D(z) pode ser reescrita da seguinte maneira:

$$D(z) = \frac{1}{\big(\rho T - \mu t(0)\big)} \frac{(2z^2 - 3z + 1)z^{-2}}{(z^2 - 2z + 1)z^{-2}} \tag{6.15}$$

Tem-se:

$$D(z) = \frac{1}{\left(\rho T - \mu t(0)\right)} \frac{(2 - 3z^{-1} + z^{-2})}{(1 - 2z^{-1} + z^{-2})} \tag{6.16}$$

Sabendo que a função de transferência D(z) é dada por:

$$D(Z) = \frac{U(z)}{E(z)} \tag{6.17}$$

Em que, U(z) é o sinal de controle e E(z) é o sinal do erro. Assim, temos:

$$U(z) = D(z)E(z) \tag{6.18}$$

Agora, substituindo D(z) na equação acima, tem-se:

$$U(z) = \frac{1}{\left(\rho T - \mu t(0)\right)} \frac{(2 - 3z^{-1} + z^{-2})}{(1 - 2z^{-1} + z^{-2})} E(z) \tag{6.19}$$

Reorganizando, tem-se:

$$\begin{gathered} U(z)(1 - 2z^{-1} + z^{-2})\left(\rho T - \mu t(0)\right) = (2 - 3z^{-1} + z^{-2})E(z) \\ (U(z) - U(z)2z^{-1} + U(z)z^{-2})\left(\rho T - \mu t(0)\right) = 2E(z) - 3E(z)z^{-1} + E(z)z^{-2} \end{gathered} \tag{6.20}$$

Fazendo a transformada Z inversa, utilizando a propriedade do deslocamento no tempo da transformada Z, tem-se:

$$u(k) - 2u(k-1) + u(k-2) = \frac{1}{\left(\rho T - \mu t(0)\right)}[2e(k) - 3e(k-1) + e(k-2)] \tag{6.21}$$

Portanto, a equação de diferenças (função de convergência) da lei de controle *deadbeat* para uma entrada rampa fica da seguinte maneira:

$$u(k) = 2u(k-1) - u(k-2) + \frac{1}{\left(\rho T - \mu t(0)\right)}[2e(k) - 3e(k-1) + e(k-2)] \tag{6.22}$$

6.3.1.2. Algoritmo de Inicialização STM - Arquitetura Centralizada

A Figura 6.6 mostra o fluxograma do algoritmo de inicialização para o modo STM com uma arquitetura centralizada, também chamada de topologia *Broadcast*. A arquitetura centralizada (ou topologia *broadcast*), prevê a existência de um relógio de referência e, a partir deste relógio, o valor de tempo é enviado a todos os relógios escravos.

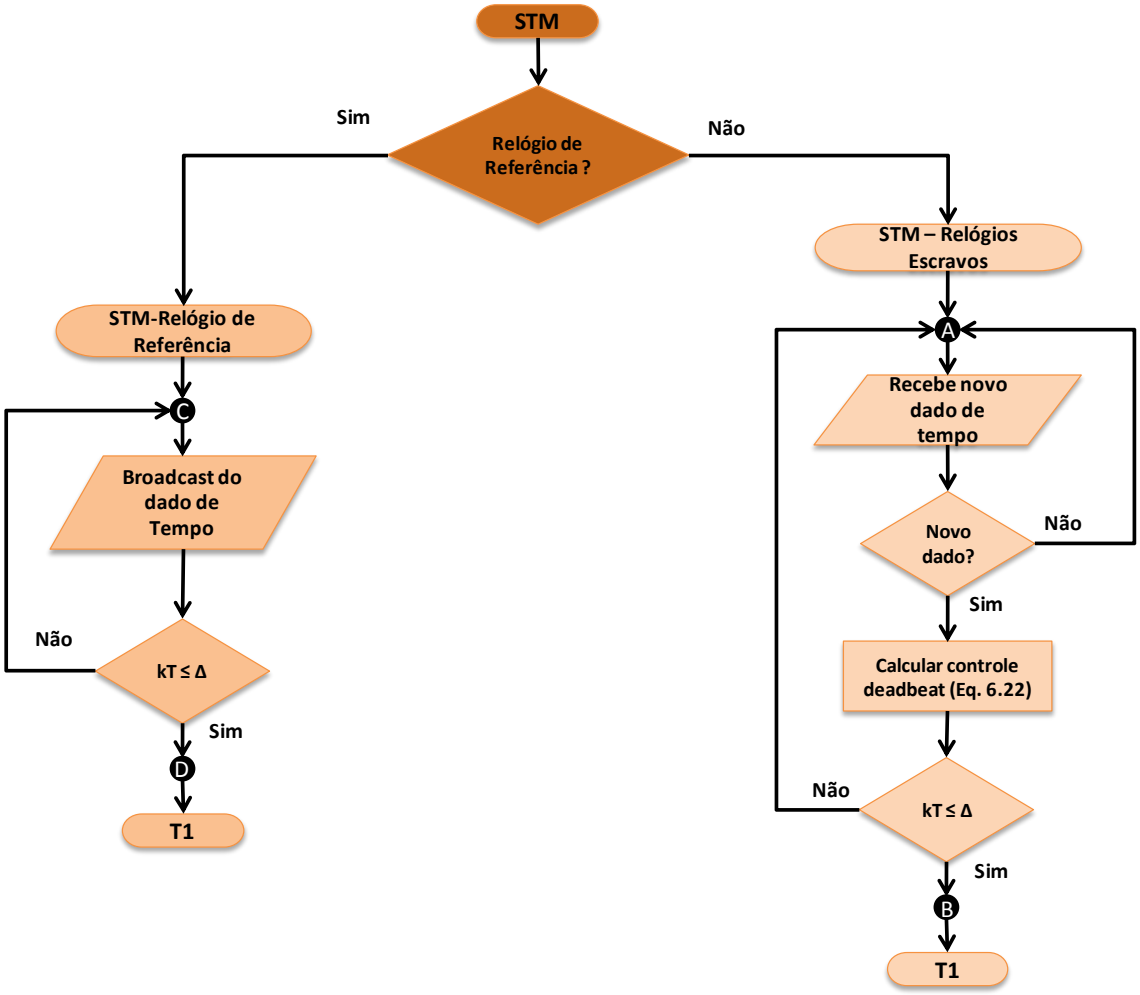


Figura 6.6 – Algoritmo STM de arquitetura centralizada.

Esta arquitetura pode ser utilizada para sincronizações onde é necessário o uso da arquitetura centralizada. A transição (T1) é realizada quando os relógios atingirem o valor de transição estabelecido (Δ).

6.3.1.3. Algoritmo de Inicialização STM - Arquitetura Distribuída

A Figura 6.7 mostra o fluxograma do algoritmo de inicialização para o modo STM com uma arquitetura distribuída, também chamado de topologia Anel Virtual.

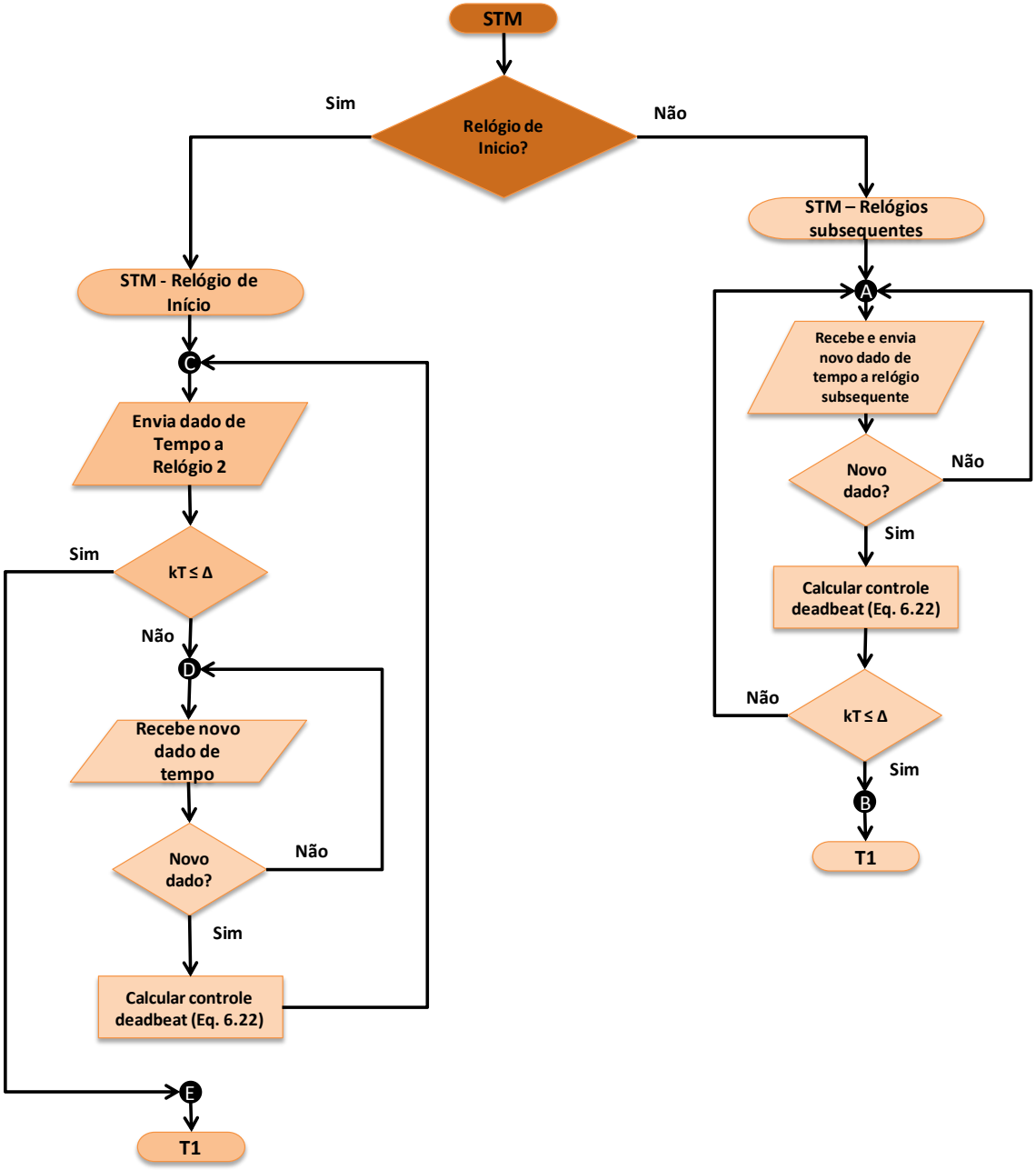


Figura 6.7 – Algoritmo STM de topologia anel virtual.

A arquitetura distribuída (ou topologia anel virtual), não prevê a existência de um relógio de referência real. Por isso, escolhe-se a priori um relógio que fará a inicialização inicial e, a partir daí, cada relógio ao receber um dado envia o seu dado de tempo ao relógio subsequente, tal como um anel. No entanto, este anel é virtual, já que não é necessária uma formação de rede em anel real. Esta arquitetura é utilizada para sincronizações em que uma arquitetura distribuída é primordial. A transição (T1) é realizada quando cada relógio atingir o valor de transição estabelecido (Δ).

6.3.1.4. Atraso de Rede

A rede de comunicação tem grande influência sobre os algoritmos de sincronização de relógios. Não somente sobre a medida do tempo, mas também sobre a métrica de transição (T1).

Em uma rede TDMA é possível estimar o atraso de pior caso (*worst case delay*) e uma rede CSMA/CD o atraso é estimado por uma distribuição de probabilidade.

No entanto, no modo de inicialização a rede ainda não está totalmente preparada e, portanto, dificultando a estimação do atraso de rede tanto em redes TDMA quanto em redes CSMA/CD, pois ainda existe a possível variabilidade do atraso devido a ruídos.

Para este trabalho, utiliza-se a abordagem de atraso de pior caso, ou seja, supõe-se que o pior caso de atraso de rede é conhecido.

Desta suposição, portanto, chega-se a um novo valor para a transição (T1):

$$\Delta = \alpha \cdot T + \delta \tag{6.23}$$

Em que na equação (6.23), tem-se:

- α = número inteiro que indica o número de períodos de amostragem necessários para o controlador *deadbeat* zerar o erro;

- T = período de amostragem;
- δ = atraso de rede de pior caso.

O valor de atraso de rede deve ser compensado nas medidas de tempo ou diretamente na formula de correção do algoritmo.

6.3.2. Modo Nominal de Viés - NOMO - Centralizado

Para a arquitetura centralizada, o modo nominal de viés instantâneo tem como principal objetivo corrigir a diferença de tempo (viés instantâneo) entre os relógios escravos e um relógio de referência, devido à deriva, ao longo do ciclo de vida do sistema. Para alcançar este objetivo, esta tese propõe um algoritmo de sincronização de relógios que faz uma re-sincronização periódica mantendo os valores tempo dentro de uma margem definida.

6.3.2.1. Métricas

O modo nominal de viés instantâneo, diferente do modo STM, não tem uma métrica pré-estabelecida, podendo ter diferentes métricas de acordo com o objetivo, técnica e necessidades individuais de cada sistema. A Figura 6.8 ilustra a região de métricas do algoritmo proposto. Isto estabelece que para esta arquitetura e este modo, a região de importância para o projeto do algoritmo é a região entre a exatidão e o controle, pois visa acertar os relógios em relação a uma referência real com o mínimo de efeito sobre o controle.

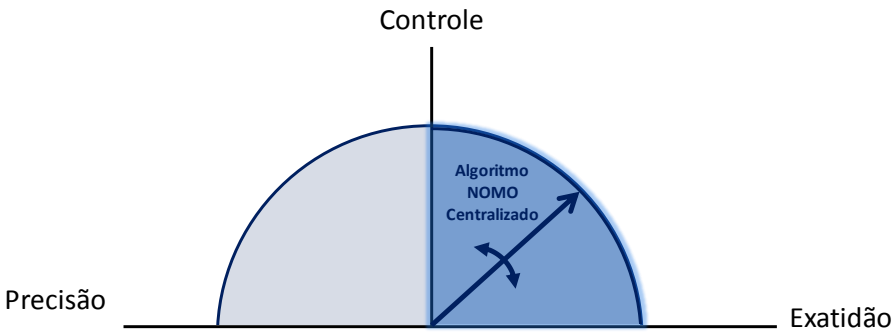


Figura 6.8 – Métrica para Algoritmo NOMO centralizado.

6.3.2.2. Imperfeições

Conhecendo as métricas, é necessário definir as imperfeições para o qual o algoritmo é definido. A Figura 6.9, ilustra a região de imperfeições deste modo.

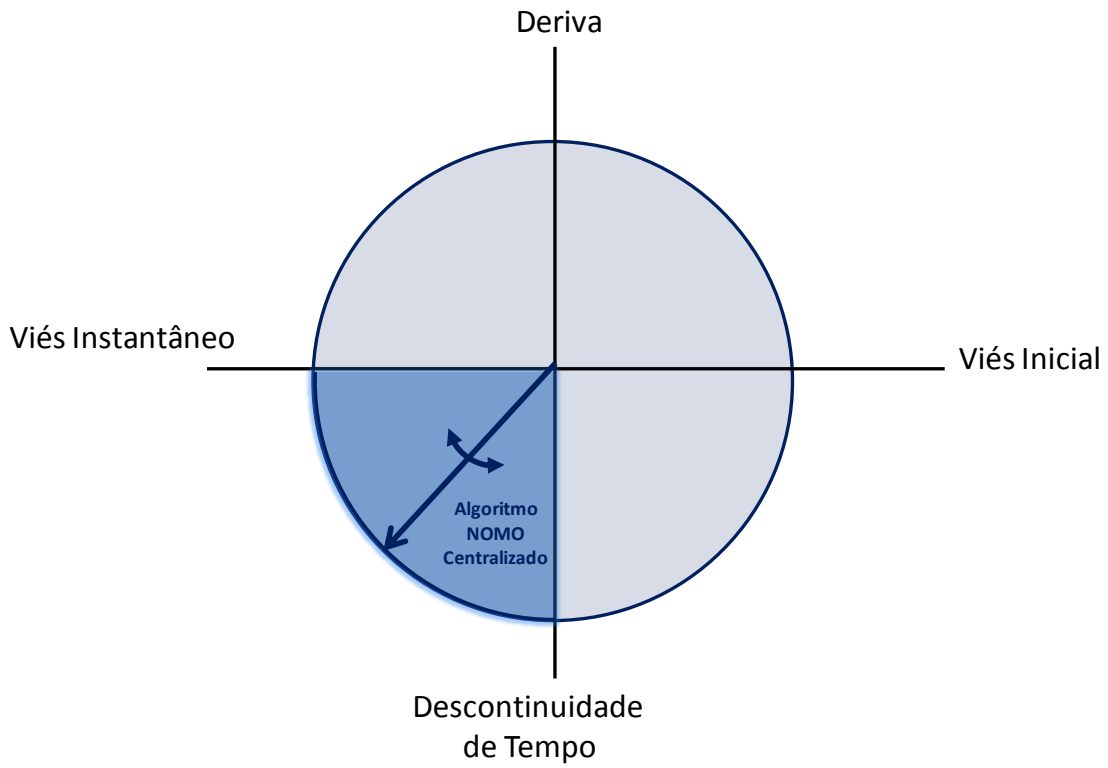


Figura 6.9 – Imperfeições para Algoritmo NOMO centralizado.

Observa-se na Figura 6.9 que o vetor do algoritmo está direcionado entre o viés instantâneo e a descontinuidade de tempo. Isto quer dizer que o algoritmo proposto tem o objetivo não só de sincronizar os relógios, mas também amortizar as descontinuidades de tempo geradas pela própria correção dos algoritmos.

6.3.2.3. Equação de Convergência

A equação de convergência é a função matemática que estabelece o valor de correção para o relógio. Para este modo, será utilizada a mesma técnica e

procedimento, baseado em média, estabelecido pelo algoritmo PTP na seção 2.3.10.3. A equação é dada por:

$$corr = \frac{(t_{s2} - t_{r1}) - (t_{r4} - t_{s3})}{2} \tag{6.24}$$

Em que:

a) t_{r1}: estampa de tempo do relógio de referência no instante de envio do *broadcast*;

b) t_{s2}: estampa de tempo do relógio escravo no instante que recebe o valor de tempo do relógio de referência;

c) t_{s3}: estampa de tempo do relógio escravo no instante de envio da requisição de atraso de tempo;

d) t_{r4}: estampa de tempo do relógio de referência no instante de recebimento da requisição de atraso de tempo;

Com estas estampas de tempo é possível calcular a correção dos relógios escravos para se corrigirem. No entanto, esta técnica não leva em conta a descontinuidade de tempo. Para isso, esta tese utiliza a técnica *cross-fading*, detalhada no capítulo 5. Com isso, a equação (6.24) toma a seguinte forma:

$$corr(k) = \frac{(t_{s2} - t_{r1}) - (t_{r4} - t_{s3})}{2} \cdot u(k) \tag{6.25}$$

Em que u é a função *cross-fading* que pode ser projetada com diferentes funções, como detalhado no capítulo 5.

6.3.2.4. Atraso de Rede

Para este modo, o atraso de rede utiliza-se da mesma abordagem de atraso detalhado pelo algoritmo PTP na seção 2.3.10.3. Pressupondo que o atraso de rede de comunicação é simétrico, o relógio escravo troca valores de tempo

com o relógio de referência e a partir disso é possível calcular utilizando a equação (2.22). Esta técnica é conhecida como *Round-Trip-Delay*.

6.3.2.5. Tolerância a Falhas

Este algoritmo não possui técnicas de tolerância a falhas. Para tanto, é necessário incluir técnicas para minimizar o ponto comum de falha (se o mestre falha, todos falham) tal como algoritmos para seleção de novo relógio mestre como o BMC e redundâncias.

De fato, este algoritmo utiliza a abordagem conhecida no mundo de falhas como "evitar a falha" (*fault-avoidance*) e redundâncias, pois o relógio mestre é normalmente um relógio de ótima qualidade com baixa probabilidade de falha. Como exemplo, pode-se citar o GPS e relógios atômicos.

6.3.2.6. Algoritmo

A Figura 6.10 mostra o fluxograma do algoritmo proposto para o modo STM com uma arquitetura centralizada.

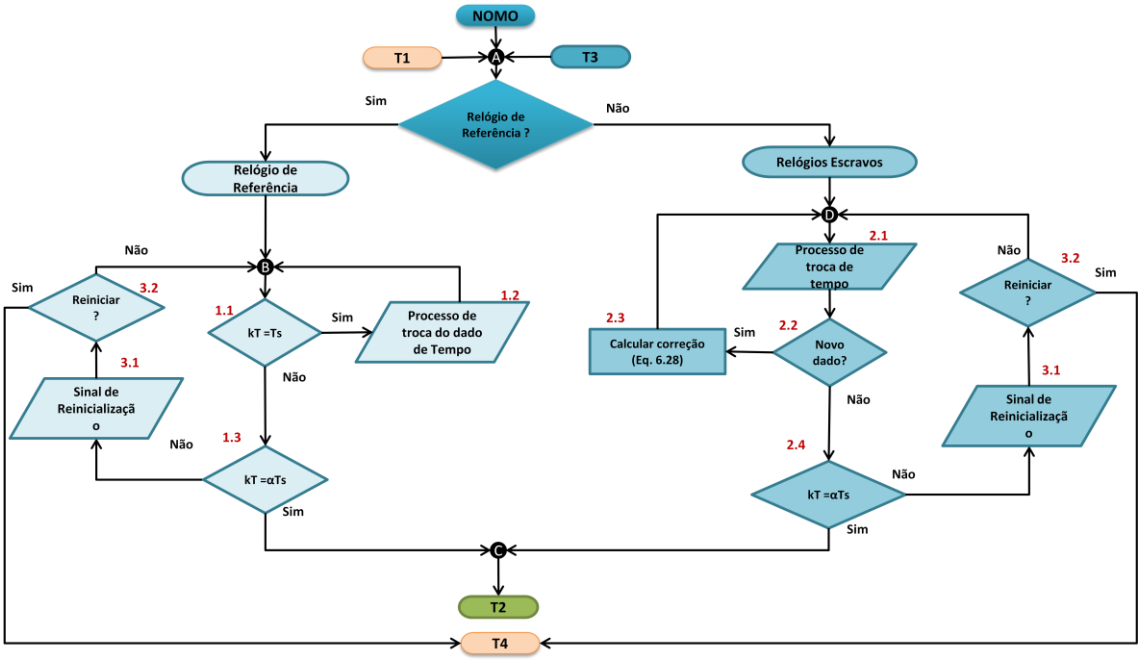


Figura 6.10 – Fluxograma do Algoritmo do modo NOMO centralizado.

Em que, da Figura 6.10, tem-se:

a) k: Instante atual;

b) T: Período de amostragem;

c) Ts: Instante de re-sincronização atual;

d) α: fator de re-sincronização de deriva.

Além disso, observam-se dois caminhos, um para o relógio de referência e outro para os relógios escravos.

Os blocos correspondem à:

a) 1.1: é um bloco lógico que verifica se o instante atual corresponde ao instante de re-sincronização;

b) 1.2: Corresponde a todo o processo de re-sincronização de um relógio de referência descrito pelo algoritmo PTP;

c) 1.3: é um bloco lógico que verifica se o instante atual corresponde ao instante de re-sincronização para a deriva;

d) 2.1: Corresponde a todo o processo de re-sincronização dos relógios escravos descrito pelo algoritmo PTP;

e) 2.2: Bloco lógico que verifica se é um novo dado recebido do relógio de referência;

f) 2.3: Bloco que calcula e ajusta o tempo do relógio local;

g) 2.4: é um bloco lógico que verifica se o instante atual corresponde ao instante de re-sincronização para a deriva;

h) 3.1: Entrada do sinal de reinicialização do sistema;

i) 3.2: Bloco lógico que verifica se houve alguma sinalização de reinicialização.

6.3.3. Modo Nominal de Viés - NOMO - Distribuído

Para a arquitetura distribuída, o modo nominal de viés instantâneo tem como principal objetivo corrigir a diferença de tempo (viés instantâneo) entre os relógios de um conjunto, devido à deriva, ao longo do ciclo de vida do sistema. Neste caso, o relógio de referência é virtual, ou seja, os relógios fazem operações matemáticas que fazem os valores de tempo convergirem a uma referência virtual. Para alcançar este objetivo, esta tese propõe um algoritmo de sincronização de relógios que faz uma re-sincronização periódica mantendo os valores tempo dentro de uma margem definida.

6.3.3.1. Métricas

A Figura 6.11 ilustra a região de métricas do algoritmo distribuído proposto. Isto estabelece que, para esta arquitetura e este modo, a região de importância para o projeto do algoritmo é a região entre a precisão e o controle, pois visa acertar os relógios em relação a uma referência virtual com o mínimo de efeito sobre o controle.

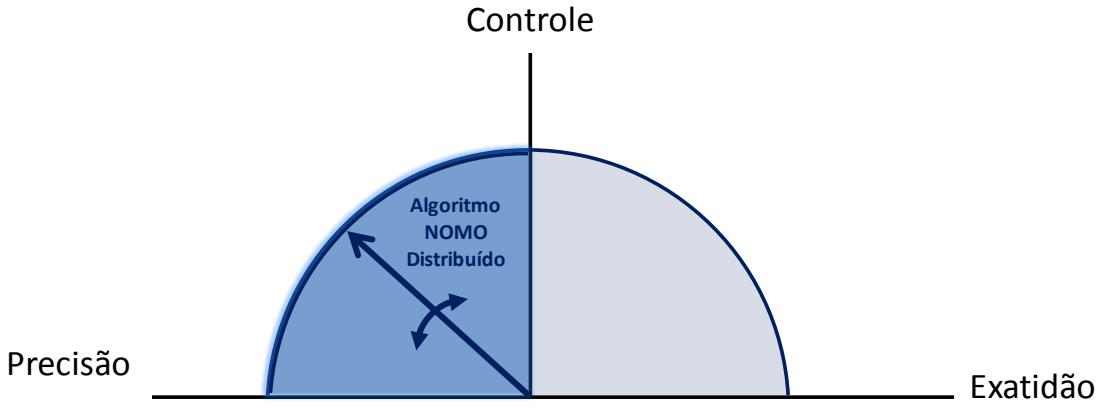


Figura 6.11 – Métrica para Algoritmo NOMO distribuído.

6.3.3.2. Imperfeições

A Figura 6.12, ilustra a região de imperfeições deste modo.

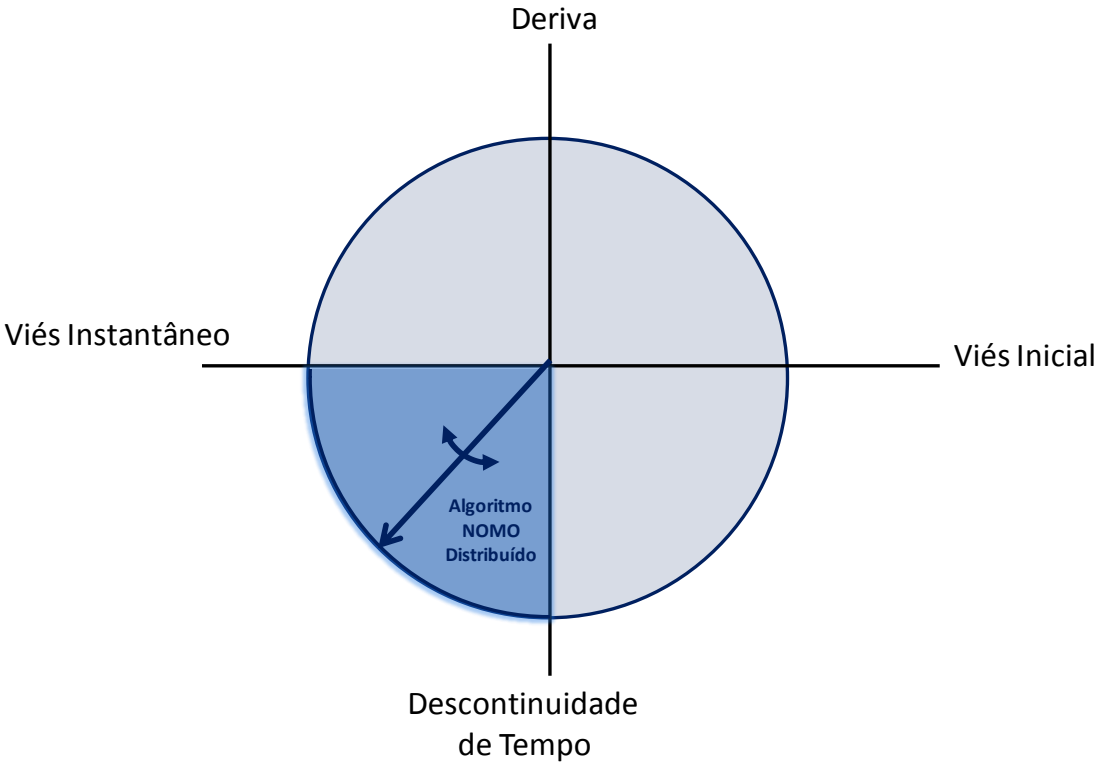


Figura 6.12 – Imperfeições para Algoritmo NOMO distribuído.

Observa-se na Figura 6.12 que o vetor do algoritmo está direcionado entre o viés instantâneo e a descontinuidade de tempo. O algoritmo distribuído proposto tem o objetivo não só de sincronizar os relógios entre si, mas também amortizar as descontinuidades de tempo geradas pela própria correção dos algoritmos.

6.3.3.3. Equações de Convergência

A equação de convergência é a função matemática que estabelece o valor de correção para os relógios. Para este modo, será utilizado a mesma técnica e procedimento, baseada em média, estabelecida pelo algoritmo FTM na seção

2.3.10.1, em que a função de convergência é dada pela equação (2.12), repetida abaixo:

$$cnf(A) = \frac{A[f+1] + A[n-f]}{2} \tag{6.26}$$

Em que A é o vetor de tempo ordenado de forma crescente, f é o número de falhas que o algoritmo pode tolerar e n é o número de relógios do conjunto. Depois, faz-se a média aritmética do maior e menor valor dos elementos restantes no vetor. No entanto, esta técnica não leva em conta a descontinuidade de tempo. Para isso, esta tese utiliza a técnica *cross-fading*, detalhada no capítulo 5. Com isso, a equação (6.26) toma a seguinte forma:

$$corr(k) = \frac{A_{f+1}(k) + A_{n-f}(k)}{2} \cdot u(k) \tag{6.27}$$

Em que, $A_{f+1}(k)$ e $A_{n-f}(k)$ é o valor do vetor de tempo ordenado de forma crescente no instante k; e u é a função *cross-fading* que pode ser projetada com diferentes funções, como detalhado no capítulo 5.

6.3.3.4. Tolerância a Falhas

Este algoritmo é tolerante a falhas bizantinas. O algoritmo de média foi projetado para evitar que as falhas de relógios propaguem na média, ou seja, se um relógio falha, na função de convergência o seu valor é descartado. Esta técnica foi concebida utilizando o teorema dos generais bizantinos. A função de *cross-fading* não influencia a tolerância a falhas, já que o propósito da mesma é amortizar a descontinuidade de tempo.

De fato, este algoritmo utiliza a abordagem conhecida no mundo de falhas como "tolerância a falhas" (*fault-tolerant*). Esta abordagem evita um ponto de falha única e torna o algoritmo tolerante as falhas bizantinas, ou seja, o sistema continua operando, dentro de limites estabelecidos, mesmo na presença de falhas bizantinas.

6.3.3.5. Algoritmo

A Figura 6.13 mostra o fluxograma do algoritmo proposto para o modo NOMO com uma arquitetura distribuída.

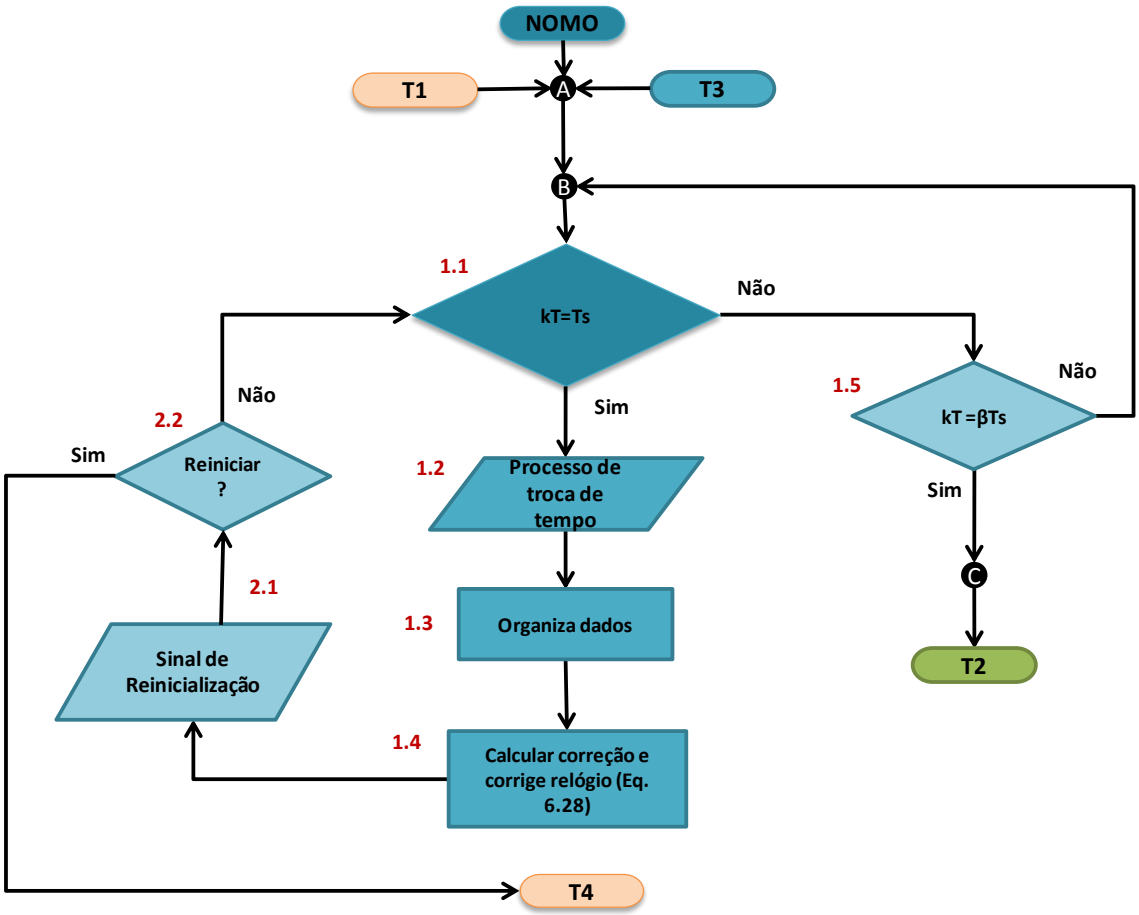


Figura 6.13 – Fluxograma do Algoritmo do modo NOMO distribuído.

Em que, da Figura 6.13, tem-se:

a) k: Instante atual;

b) T: Período de amostragem;

c) Ts: Instante de re-sincronização atual;

d) β: fator de re-sincronização de deriva.

Os blocos correspondem à:

a) 1.1: É um bloco lógico que verifica se o instante atual corresponde ao instante de re-sincronização;

b) 1.2: Corresponde a todo o processo de re-sincronização de um relógio de referência descrito pelo algoritmo FTM;

c) 1.3: É um bloco lógico que organiza de forma crescente os dados de tempo em um vetor de dados;

d) 1.4: Bloco que calcula a correção e ajusta o tempo do relógio local;

e) 1.5: Bloco lógico que verifica se o instante atual corresponde ao instante de re-sincronização para a deriva.

f) 2.1: Entrada do sinal de reinicialização do sistema;

g) 2.2: Bloco lógico que verifica se houve alguma sinalização de reinicialização.

6.3.3.6. Atraso de Rede

Para este modo, o atraso de rede utiliza-se da mesma abordagem de atraso detalhado pelo algoritmo FTM na seção 2.3.10.1. Ou seja, pressupõe-se que o atraso de rede de comunicação é conhecido ou possível de ser estimável. Com isso, o valor de maior atraso de rede é utilizado por todos os relógios no ajuste da correção de relógio. Esta técnica é conhecida como *Worst-Case-Delay*.

6.3.4. Modo Nominal de Deriva - NOMD

O modo nominal de deriva tem como principal objetivo corrigir a diferença entre os fatores de conversão do microtick e macrotick (MMCF) entre os relógios de um conjunto. Neste caso, um relógio do conjunto é escolhido como referência e todos os outros relógios fazem operações matemáticas que fazem os valores de MMCF se adequarem em relação ao valor do MMCF do relógio de

referência escolhido. Para alcançar este objetivo, esta tese propõe um algoritmo de sincronização de MMCF que faz uma re-sincronização periódica.

6.3.4.1. Métricas

A Figura 6.14 ilustra a região de métricas do algoritmo proposto. Isto estabelece que para este modo o objetivo seja a correção da deriva, pois visa acertar os MMCF dos relógios em relação ao MMCF de um relógio de referência.

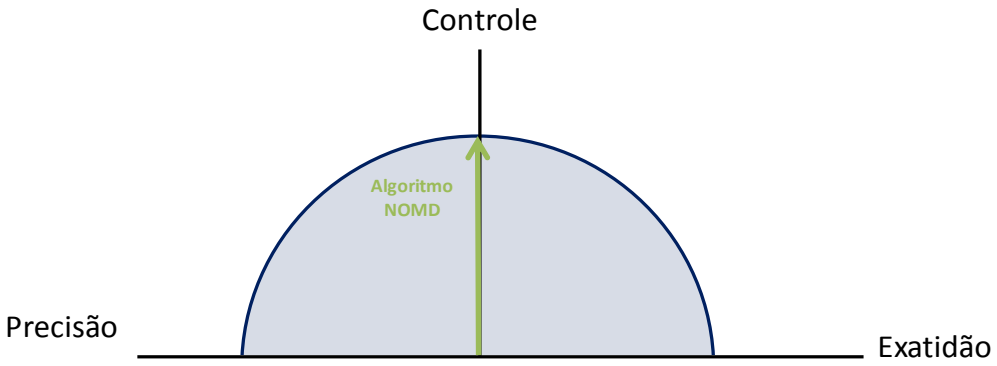


Figura 6.14 – Métrica para Algoritmo NOMD.

6.3.4.2. Imperfeições

A Figura 6.15, ilustra a região de imperfeições deste modo. Observa-se na Figura 6.15 que o vetor do algoritmo esta direcionado à deriva. O algoritmo proposto tem o objetivo único de minimizar/eliminar as derivas, ou neste caso, acertar os MMCF´s.

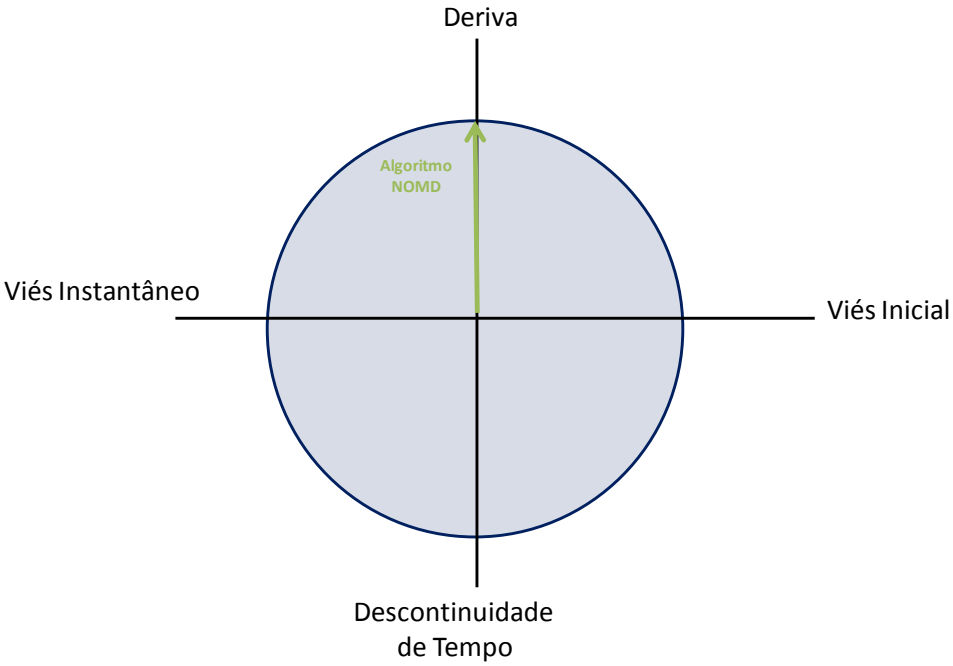


Figura 6.15 – Imperfeições para Algoritmo NOMD.

6.3.4.3. Equação de Convergência

A equação de convergência é a função matemática que estabelece o valor de correção para os relógios. Para este modo, será utilizado a mesma técnica e procedimento estabelecido pelo algoritmo SR na seção 2.3.10.4.

O procedimento de correção do MMCF não é um simples ajuste referência-controle. Na verdade, a correção do MMCF dos relógios deve levar em conta a deriva entre o relógio de referência e o relógio a ser ajustado, assim soma-se ao valor de MMCF para que os relógios tenham o evento de macrotick no mesmo instante (ou região). A equação é dada por:

$$MMCF_i(k) = MMCF_r(k) - \frac{(corr_i(k) - corr_r(k))}{maT_i(k)} \qquad (6.28)$$

Em que $MMCF_i$ e $MMCF_r$ são os fatores de conversão no instante k dos relógios i e do relógio escolhido como referência para este modo

respectivamente; $corr_i$ e $corr_r$ são os valores de correção do modo nominal de viés instantâneo no instante k e maT_i é o valor do macrotick no instante k.

6.3.4.4. Tolerância a Falhas

Este algoritmo não possui técnicas de tolerância a falhas. Para tanto, é necessário incluir técnicas para minimizar o ponto comum de falha (se o mestre falha, todos falham) tal como algoritmos para seleção de novo relógio mestre como o BMC e redundâncias.

6.3.4.5. Algoritmo

A Figura 6.16 mostra o fluxograma do algoritmo proposto para o modo NOMD.

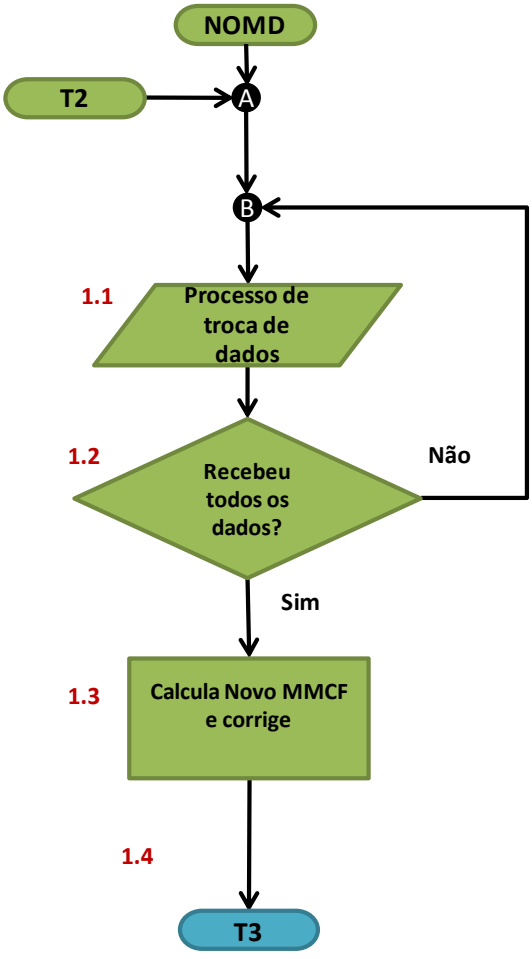


Figura 6.16 – Fluxograma do Algoritmo do modo NOMD.

Em que, da Figura 6.16, os blocos correspondem à:

a) 1.1: Corresponde a todo o processo de troca de dados entre os relógios;

b) 1.2: É um bloco lógico que verifica se todos os dados necessários foram recebidos;

c) 1.3: Calcula o novo MMCF e atualiza e corrige o novo valor;

d) 1.4: Bloco faz a transição para o modo NOMO;

6.3.4.6. Atraso de Rede

Para este modo, o atraso de rede não é relevante, pois todo o atraso de rede já foi considerado dentro das equações de correção do modo NOMO. Além disso, o valor de MMCF do relógio de referência é estático e sua atualização só afeta a próxima amostragem.

6.4. Especificação do Conjunto de Algoritmos ReS

6.4.1. Algoritmo AReS

O algoritmo AReS especifica um algoritmo com abordagem centralizada utilizando dois modos de correção: 1) STM e 2) NOMO (centralizado). O primeiro modo, STM, visa corrigir o viés inicial com o controlador *deadbeat* como especificado na seção 6.3.1. O segundo modo, NOMO, visa corrigir os vieses instantâneos utilizando as técnicas e equações de convergência especificada pela seção 6.3.2. As técnicas de suavização (métrica de controle), utilizam-se das técnicas definidas pelo Capítulo 5.

6.4.2. Algoritmo PReS

O algoritmo PReS especifica um algoritmo com abordagem distribuída utilizando dois modos de correção: 1) STM e 2) NOMO (distribuído). O primeiro modo, STM, visa corrigir o viés inicial com o controlador *deadbeat* como

especificado na seção 6.3.1. O segundo modo, NOMO, visa corrigir os vieses instantâneos utilizando as técnicas e equações de convergência especificadas pela seção 6.3.3. As técnicas de suavização (métrica de controle), utilizam-se das técnicas definidas pelo Capítulo 5.

6.4.3. Algoritmo PAReS

O algoritmo PAReS especifica um algoritmo com abordagem distribuída utilizando três modos de correção: 1) STM, 2) NOMO e 3) NOMD. O primeiro modo, STM, visa corrigir o viés inicial com o controlador *deadbeat* como especificado na seção 6.3.1. O segundo modo, NOMO, visa corrigir os vieses instantâneos utilizando as técnicas e equações de convergência especificadas pela seção 6.3.3. Por fim, o modo NOMD corrige os fatores de conversão microtick-macrotick (MMCF) utilizando as técnicas e equações de convergência especificadas na seção 6.3.4. As técnicas de suavização (métrica de controle), utilizam-se das técnicas definidas pelo Capítulo 5.

6.5. Síntese do Algoritmo

Os algoritmos ReS apresentados, possuem 3 modos de sincronização, onde cada modo possui seu próprio algoritmo com duas arquiteturas diferentes, uma centralizada e outra distribuída.

De fato, além das especificações do algoritmo, os fluxogramas estabelecem um procedimento de sincronização. Utilizando os procedimentos descritos pelos fluxogramas e alterando algumas características é possível de fato utilizar diferentes algoritmos dentro deste procedimento. Este procedimento permite que os algoritmos de sincronização trabalhem com diferentes métricas e imperfeições, estabelecendo assim um conjunto de algoritmos reconfiguráveis e suavizáveis de métrica mista (precisão, exatidão e controle) para uso em sistemas de controle por rede e/ou sistemas complexos e altamente integrados.

Todos os procedimentos foram estabelecidos para sincronização de relógios lógicos (macrotick). No entanto, é possível aplicar o mesmo procedimento sobre relógios físicos (microtick) com o mínimo de alterações. A Figura 6.17 ilustra a visão geral de todo o procedimento especificado por este Capítulo.

As simulações que verificam e validam o procedimento são apresentadas no Capítulo 8.

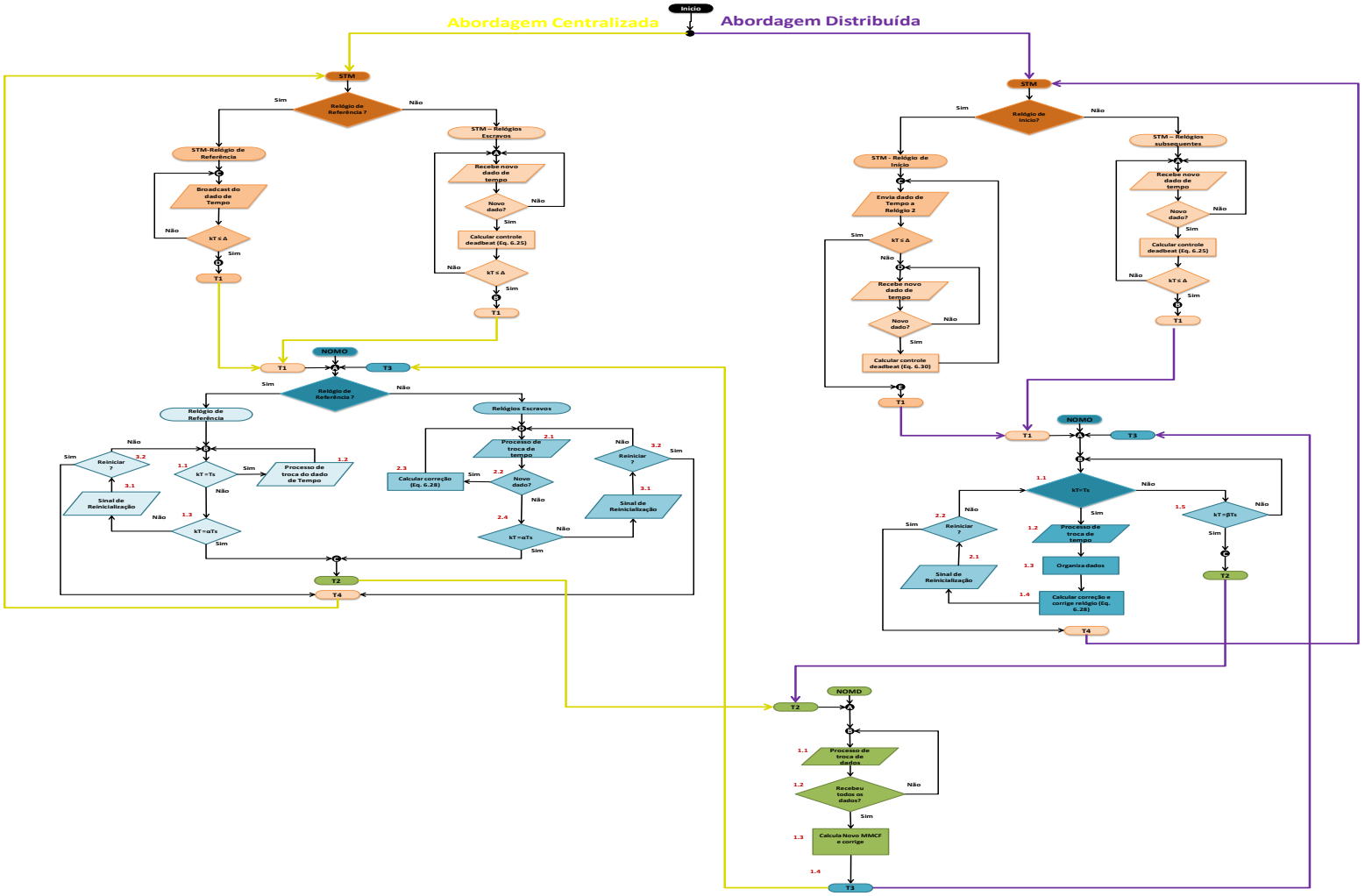


Figura 6.17 – Visão Geral do Algoritmo de Sincronização de Relógios.

7 SIMULAÇÕES DAS TÉCNICAS DE AMORTIZAÇÃO DE ALGORITMOS DE SINCRONIZAÇÃO DE RELÓGIOS

7.1. Efeito da Descontinuidade de Tempo sobre um Sistema de Controle por Redes

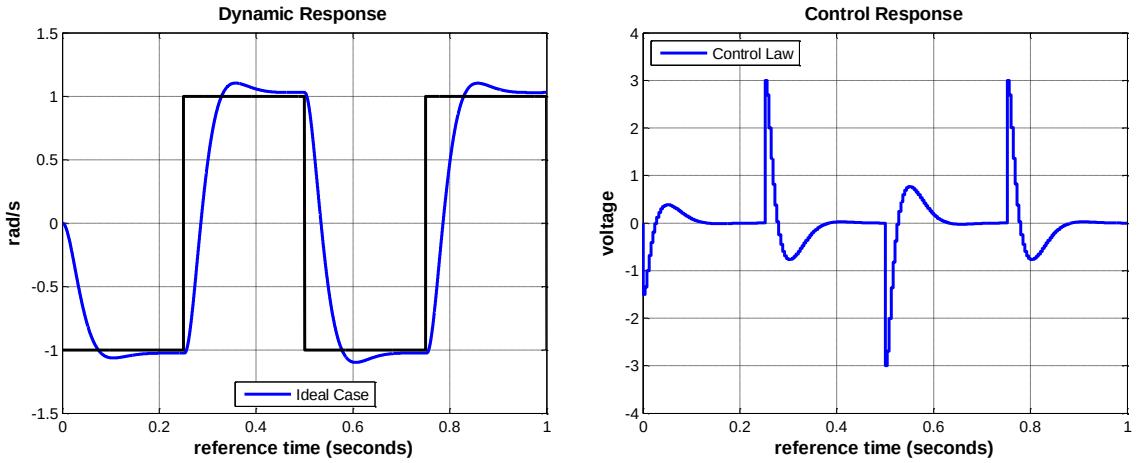
Para mostrar o efeito que a descontinuidade de tempo pode ter sobre a resposta do sistema de controle por redes, esta tese simula o modelo descrito no capitulo 3 com uma rede de comunicação TDMA, utilizando dois algoritmos de sincronização de relógios (FTM e PTP), variando as derivas no sensor. Os casos estudados estão descritos na Tabela 7.1.

Tabela 7.1 - Casos de Estudos - Descontinuidade de Tempo

Caso	Algoritmo de Sincronização	Deriva no Sensor
0 (Ideal)	NA	0
1	PTP	0.001%
2	PTP	1%
3	PTP	10%
4	FTM	0.001%
5	FTM	1%
6	FTM	10%

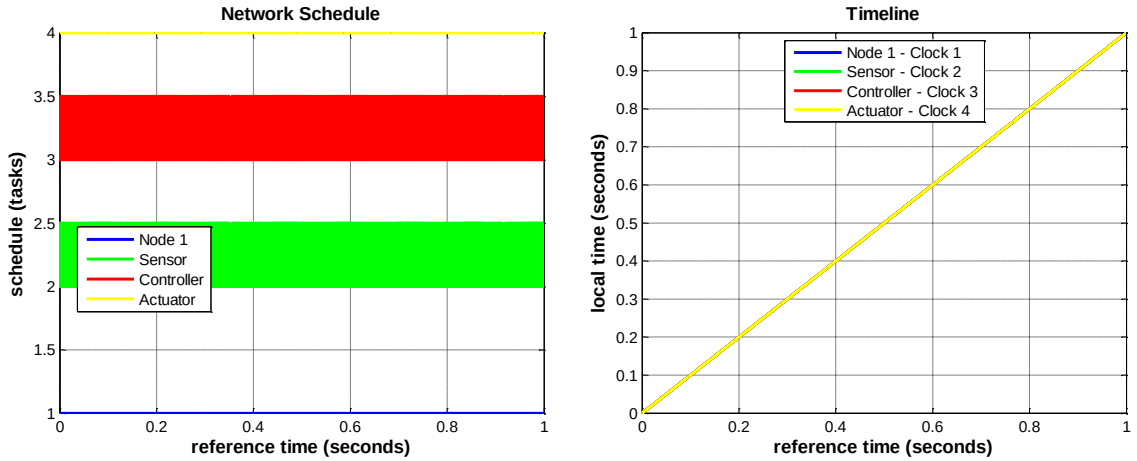
7.1.1. Caso 0 - Ideal

Este caso mostra um Sistema NCS sem nenhuma imperfeição e nenhum algoritmo de sincronização de relógios. A Figura 7.1 e a Figura 7.2 mostram, respectivamente, a resposta ao degrau, lei de controle, agendador da rede e linhas de tempo do caso ideal.



(a) (b)

Figura 7.1 – Caso Ideal: (a) Resposta ao Degrau. (b) Controle.

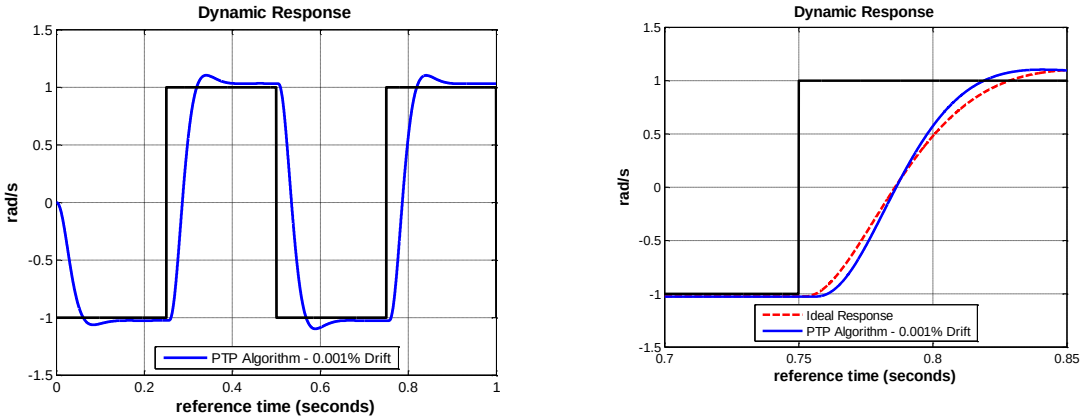


(a) (b)

Figura 7.2 – Caso Ideal: (a) Agendador da Rede. (b) Linhas de Tempo.

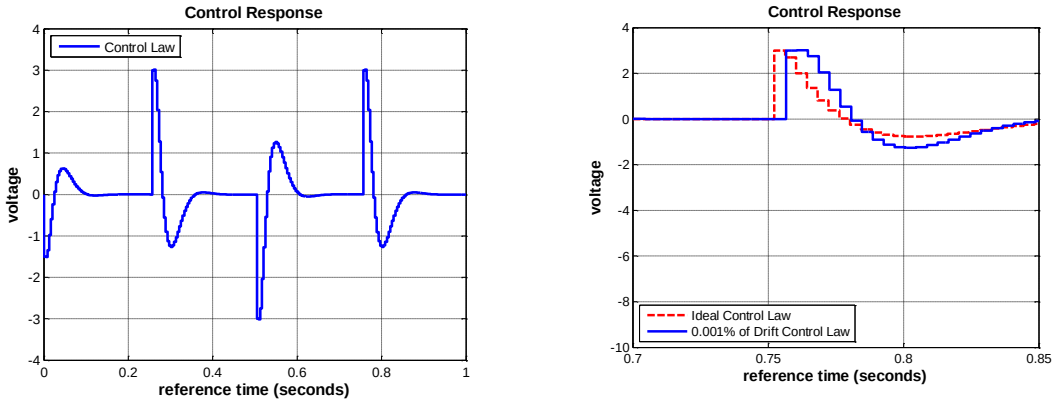
7.1.2. Caso 1 - PTP

Da Figura 7.3 à Figura 7.6 mostram-se os resultados simulados de um NCS com 0.001% de deriva no relógio do sensor (chamado de caso 1), ou seja, o relógio do sensor tem 0.001% de velocidade maior que os outros nós da rede. Utilizando como algoritmo de sincronização de relógios o algoritmo PTP.



(a) (b)

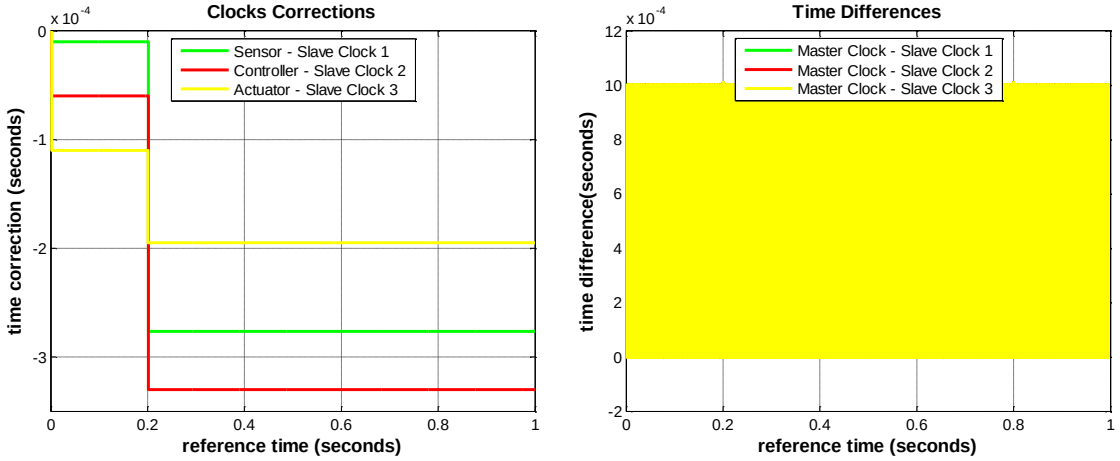
Figura 7.3 – Caso 1: 0.001% de Deriva: (a) Resposta Dinâmica. (b) Aproximação.



(a) (b)

Figura 7.4 – Caso 1: 0.001% de Deriva: (a) Lei de Controle. (b) Aproximação.

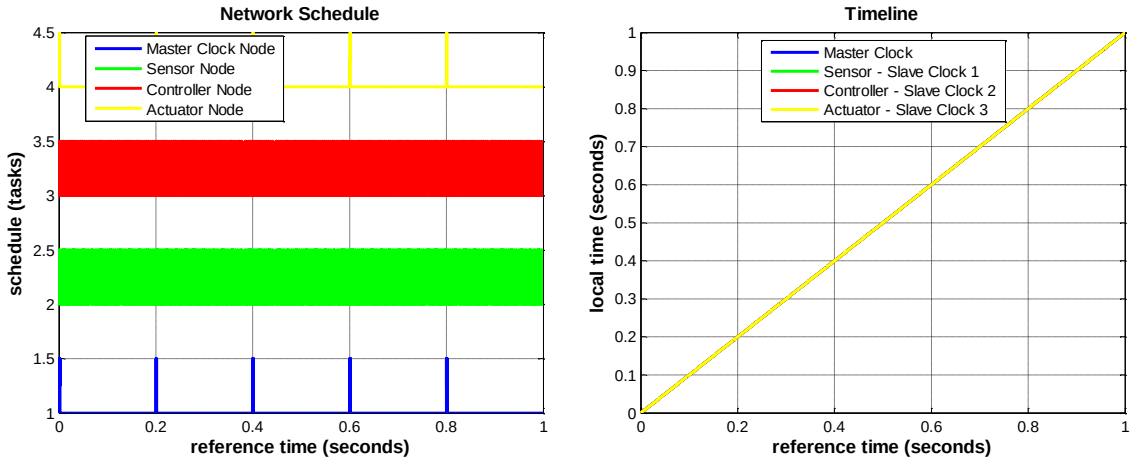
A Figura 7.5 (a), mostra as correções de tempo aplicadas devidas à correção calculada pelo algoritmo PTP, a cada 0.2 segundos; e a Figura 7.5 (b) mostra as diferenças de tempo entre o relógio mestre e relógios escravos: Sensor (Relógio Escravo 1 – Verde), Controlador (Relógio Escravo 2 – Vermelho) e Atuador (Relógio Escravo 3 – Amarelo). Todos os relógios foram sincronizados dentro de uma exatidão de 10 milissegundos, como mostra a Figura 7.5 (b).



(a) (b)

Figura 7.5 – Caso 1: (a) Correções do Relógio. (b) Diferenças de Tempo.

A Figura 7.6 mostra o agendador da rede (a) e as linhas de tempo(b).



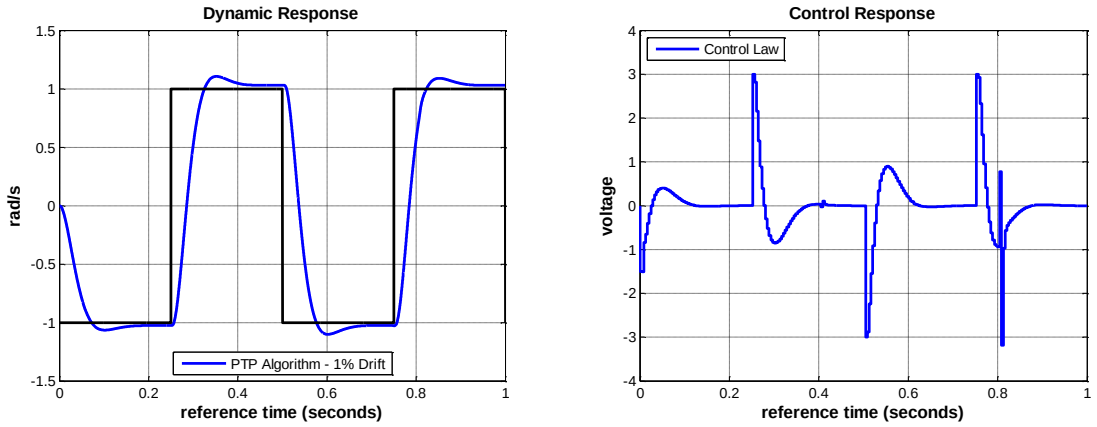
(a) (b)

Figura 7.6 – Caso 1: (a) Agendador da Rede. (b) Linhas de Tempo.

Neste caso, com 0.001% de deriva no sensor, diferentemente do caso 0 é possível observar uma pequena diferença na resposta e na lei de controle, como mostra a Figura 7.3 e a Figura 7.4. Isto é devidas à correção dos relógios. Este efeito é causado devido à descontinuidade de tempo, no entanto, não causa uma degradação suficiente que cause uma abertura de malha na resposta dinâmica do NCS.

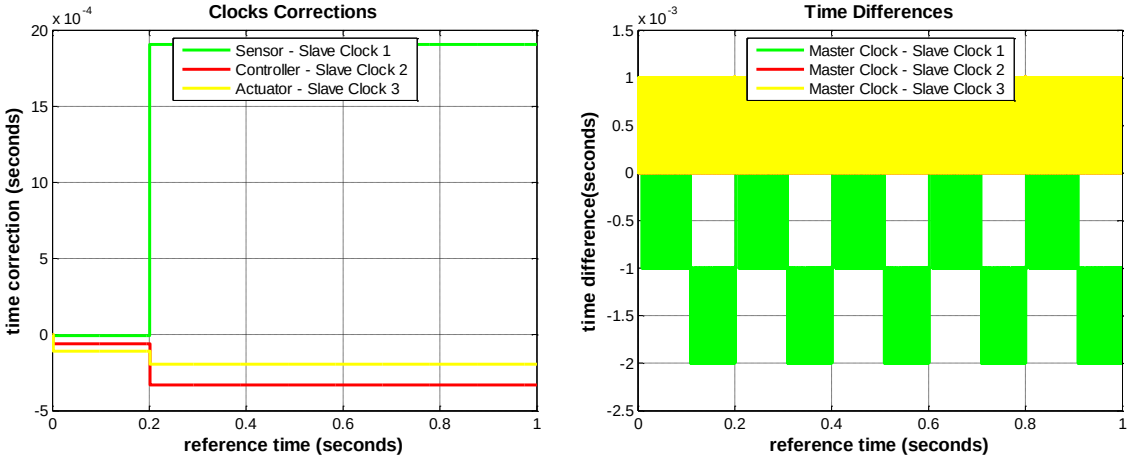
7.1.3. Caso 2 - PTP

Da Figura 7.7 à Figura 7.9 mostram-se os resultados simulados de um NCS com 1% de deriva no relógio do sensor (chamado de caso 2), ou seja, o relógio do sensor tem 1% de velocidade maior que os outros nós da rede, caracterizando um relógio com falha. Foi utilizado como algoritmo de sincronização de relógios o algoritmo PTP.



(a) (b)

Figura 7.7 – Caso 2: 1% de Deriva: (a) Resposta Dinâmica. (b) Lei de Controle.



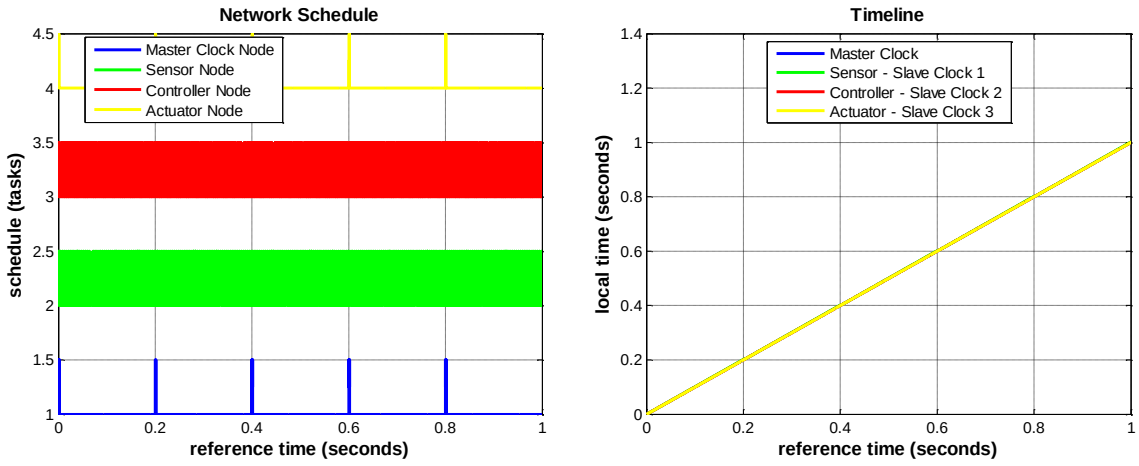
(a) (b)

Figura 7.8 – Caso 2: (a) Correções do Relógio.(b) Diferenças do Tempo.

É possível observar, em comparação com os outros casos, na Figura 7.7 (b) que a lei de controle já possui uma degradação visível. Além disso, ainda na Figura 7.7 (b), observam-se pequenas perturbações no controle. Isto ocorre

devido à sincronização de relógios. Sendo a sincronização aplicada a cada 0.2 seg., o controle é afetado devido à introdução de descontinuidades de tempo que geram atraso nas tarefas de controle e rechaveamento de contexto temporal. No entanto, durante o período estacionário o efeito de perturbação é muito menor que durante um período de transição. A Figura 7.8 (a), mostra as correções de tempo aplicadas devidas à correção calculada pelo algoritmo PTP, a cada 0.2 segundos; e a Figura 7.8 (b) mostra as diferenças de tempo (sobrepostas) entre o relógio mestre e relógios escravos: Sensor (Relógio Escravo 1 – Verde), Controlador (Relógio Escravo 2 – Vermelho) e Atuador (Relógio Escravo 3 – Amarelo). Todos os relógios foram sincronizados entre uma exatidão de 1 a -2 milissegundos, como mostra a Figura 7.8 (b), aumentando a incerteza em relação ao caso 0.

A Figura 7.9 mostra em (a) o agendador da rede; em (b) as linhas de tempo.



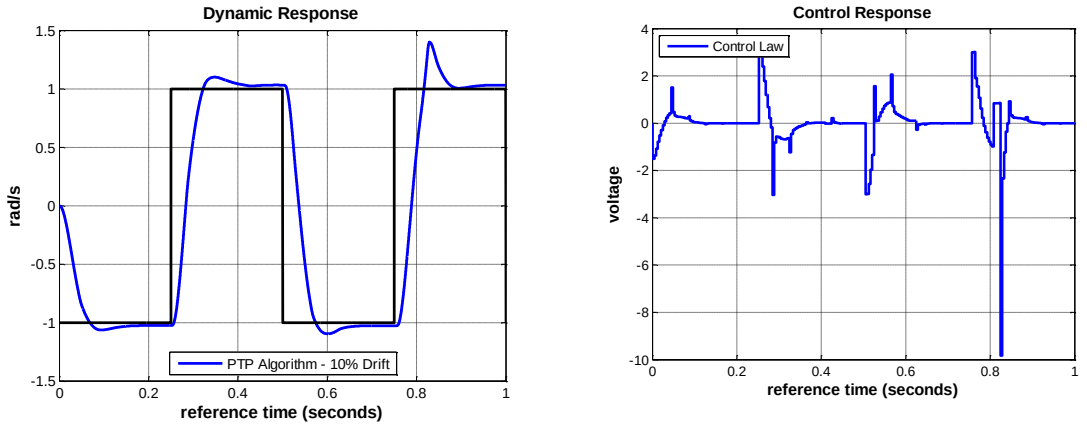
(a) (b)

Figura 7.9 – Caso 2: (a) Agendador da Rede. (b) Linhas de Tempo.

Neste caso, com 1% de deriva no sensor, é possível observar uma diferença na resposta dinâmica (Figura 7.7 (a)) e na lei de controle (Figura 7.7 (b)). No entanto, a resposta dinâmica não apresenta uma aparente degradação.

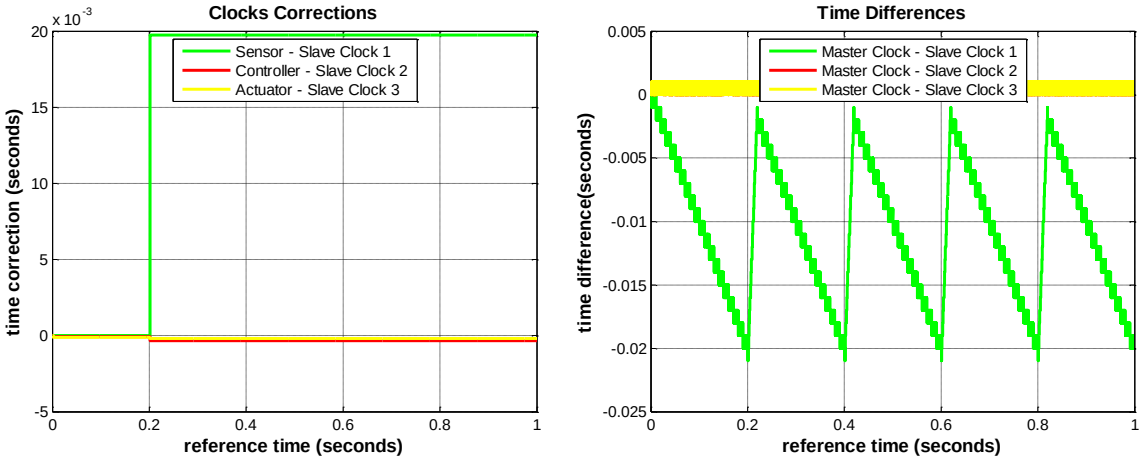
7.1.4. Caso 3 - PTP

Da Figura 7.10 à Figura 7.12 mostram-se os resultados simulados de um NCS com 10% de deriva no relógio do sensor (chamado de caso 3), ou seja, o relógio do sensor tem 10% de velocidade maior que os outros nós da rede, caracterizando um relógio com falha. Foi utilizado, como algoritmo de sincronização de relógios, o algoritmo PTP.



(a) (b)

Figura 7.10 – Caso 3: 10% de Deriva: (a) Resposta Dinâmica. (b) Lei de Controle.



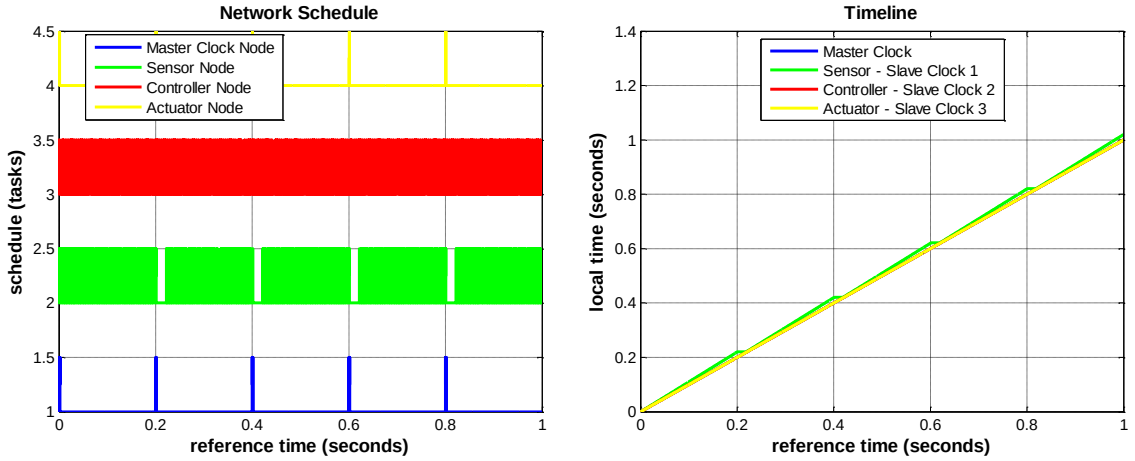
(a) (b)

Figura 7.11 – Caso 3: (a) Correções do Relógio.(b) Diferenças do Tempo.

A Figura 7.11 (a), mostra as correções de tempo aplicadas devidas à correção calculada pelo algoritmo PTP, a cada 0.2 segundos; e a Figura 7.11 (b) mostra

as diferenças de tempo entre o relógio mestre e relógios escravos: Sensor (Relógio Escravo 1 – Verde), Controlador (Relógio Escravo 2 – Vermelho) e Atuador (Relógio Escravo 3 – Amarelo). Todos os relógios foram sincronizados entre uma exatidão de 1 a -20 milissegundos, como mostra a Figura 7.11 (b), aumentando a inexatidão em relação aos casos anteriores.

A Figura 7.12 mostra em (a) o agendador da rede; em (b) as linhas de tempo.



(a) (b)

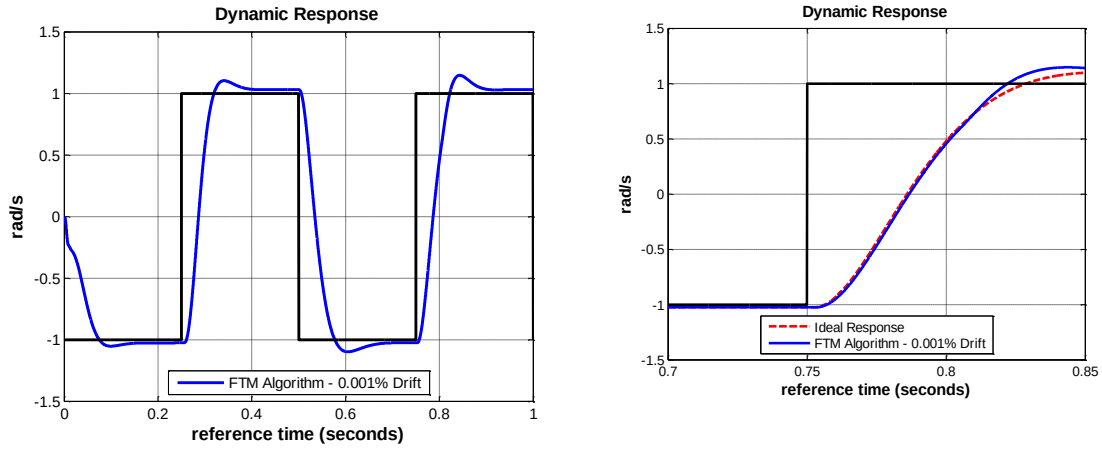
Figura 7.12 – Caso 3: (a) Agendador da Rede. (b) Linhas de Tempo.

Com 10% de deriva no sensor, é possível observar na Figura 7.10 uma grande diferença na resposta dinâmica e na lei de controle. Apesar de o algoritmo de sincronização PTP cumprir o seu papel de sincronizar os relógios (mesmo sobre uma exatidão alta), a resposta dinâmica e a lei de controle são degradadas, especialmente durante as transições do sistema. Este efeito é causado devido à descontinuidade de tempo. O controle é afetado devido à introdução de descontinuidades de tempo que geram atraso nas tarefas de controle e rechaveamento de contexto temporal. No entanto, durante o período estacionário o efeito de perturbação é muito menor que durante um período de transição, pois o atraso e rechaveamento causam perdas de tarefas devido a deadlines e outros que é muito mais prejudicial ao sistema durante as transições. Este caso é importante, pois mostra que os problemas de descontinuidade são um problema consistente em sistemas de controle

distribuídos que necessitam de sincronização de relógios por métrica de exatidão, principalmente sistemas com um ciclo de vida grande, como no caso de sistemas aeroespaciais.

7.1.5. Caso 4 - FTM

Da Figura 7.13 à Figura 7.16 mostram-se os resultados simulados de um NCS com 0.0011% de deriva no relógio do sensor (chamado de caso 4), ou seja, o relógio do sensor tem 0.001% de velocidade maior que os outros nós da rede, caracterizando um relógio. Foi utilizado como algoritmo de sincronização de relógios o algoritmo FTM.



(a) (b)

Figura 7.13 – Caso 4: 0.001% de Deriva: (a) Resposta Dinâmica. (b) Aproximação.

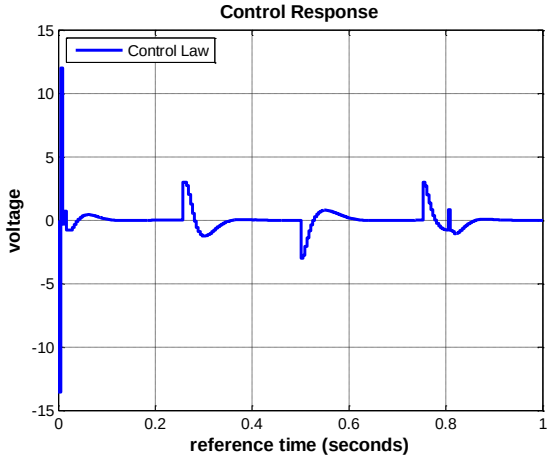
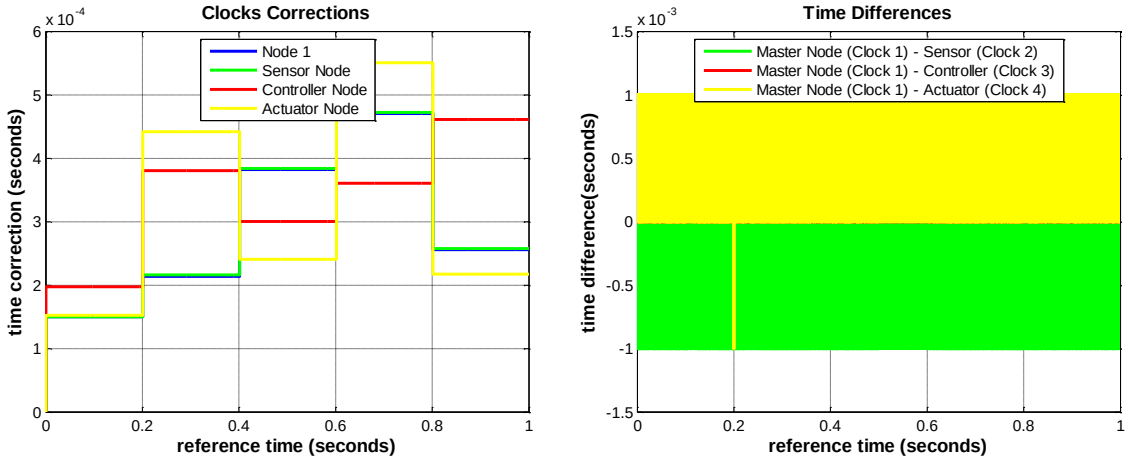


Figura 7.14 – Caso 4: Lei de Controle.

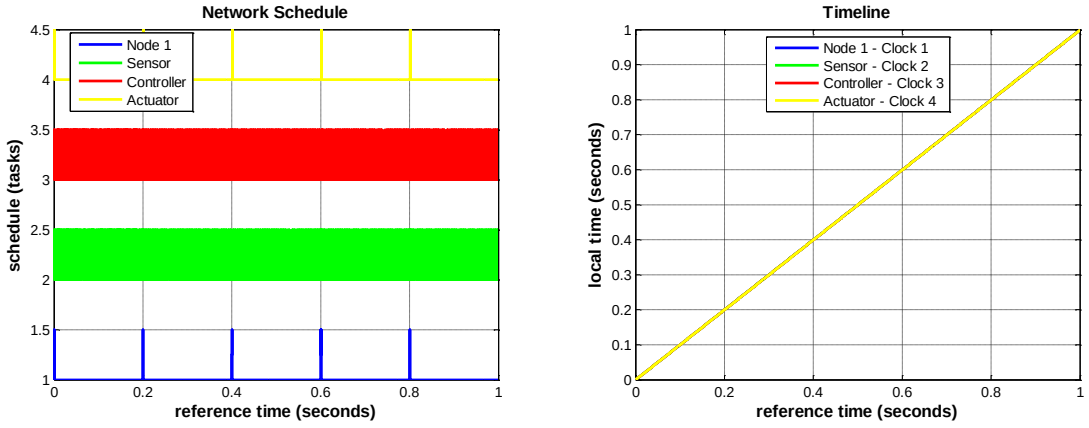
A Figura 7.14 mostra a lei de controle aplicada e, a Figura 7.15 (a), mostra as correções de tempo aplicadas devidas à correção calculada pelo algoritmo FTM, a cada 0.2 segundos; e a Figura 7.15 (b) mostra as diferenças de tempo entre os relógios do conjunto Nó 1 (Relógio 1), Sensor (Relógio 2 – Verde), Controlador (Relógio 3 – Vermelho) e Atuador (Relógio 4 – Amarelo). Todos os relógios foram sincronizados entre uma precisão de -1 a 1 milissegundo, como mostra a Figura 7.15 (b).



(a) (b)

Figura 7.15 – Caso 4: (a) Correções do Relógio. (b) Diferenças de Tempo.

A Figura 7.16 mostra em (a) o agendador da rede; em (b) as linhas de tempo.



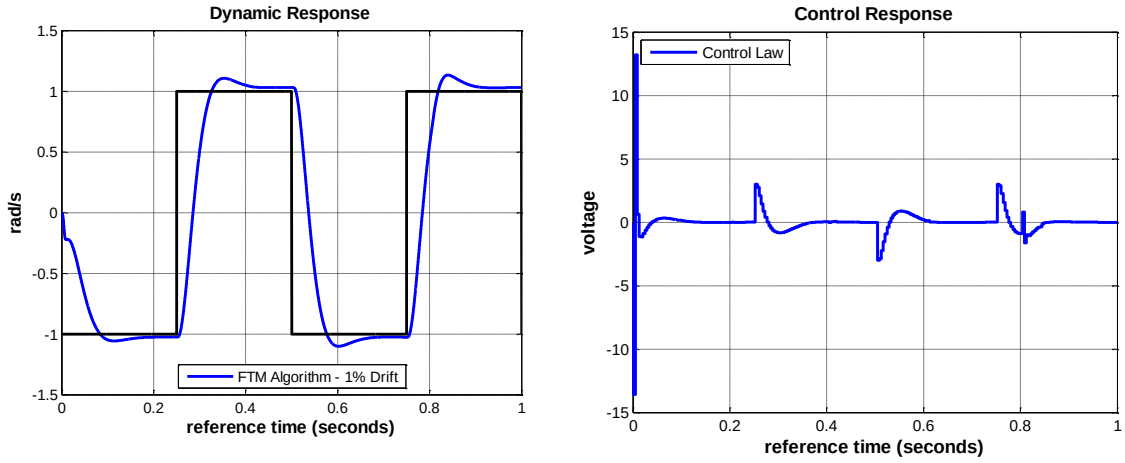
(a) (b)

Figura 7.16 – Caso 4: (a) Agendador da Rede. (b) Linhas de Tempo.

Neste caso, com 0.001% de deriva do sensor, é observada na Figura 7.13 uma pequena diferença da lei de controle em relação ao caso ideal. Isto sugere que este caso, apesar da baixa deriva, é afetado pela descontinuidade de tempo (atraso no tempo).

7.1.6. Caso 5 - FTM

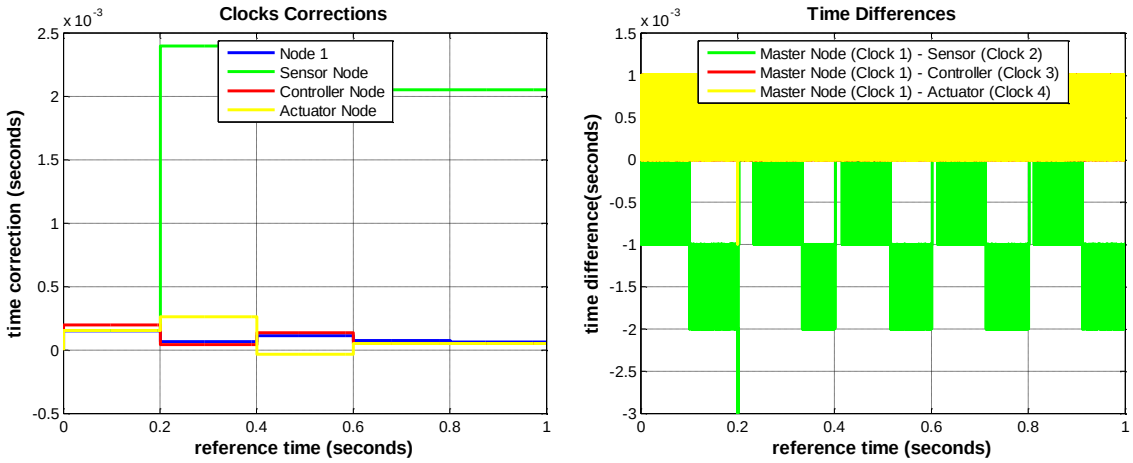
Da Figura 7.17 à Figura 7.19 mostram-se os resultados simulados de um NCS com 1% de deriva no relógio do sensor (chamado de caso 5), ou seja, o relógio do sensor tem 1% de velocidade maior que os outros nós da rede, caracterizando um relógio com falha. Foi utilizado, como algoritmo de sincronização de relógios o algoritmo FTM.



(a) (b)

Figura 7.17 – Caso 5: 1% de Deriva: (a) Resposta Dinâmica. (b) Lei de Controle.

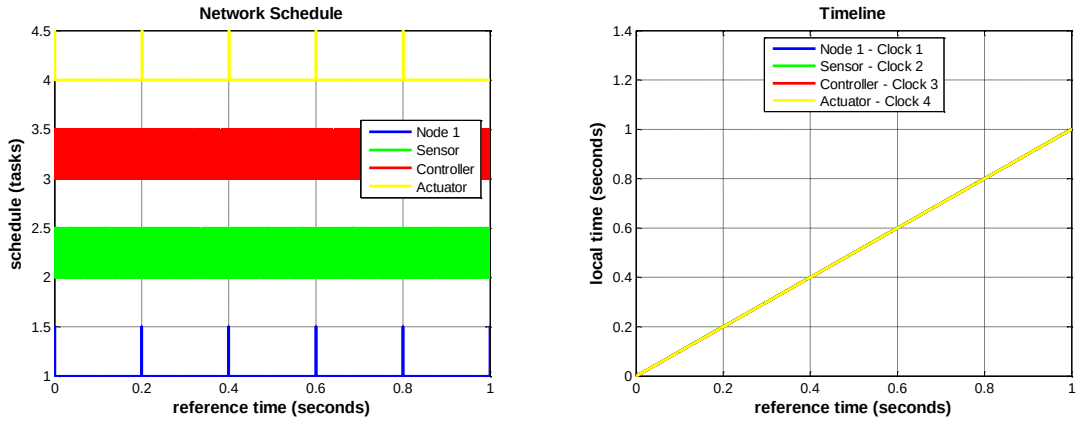
O controle é afetado devido à introdução de descontinuidades de tempo que geram atraso nas tarefas de controle e rechaveamento de contexto temporal. No entanto, durante o período estacionário efeito de perturbação é muito menor que durante um período de transição e inicialização, pois o atraso e rechaveamento causam perdas e atrasos de tarefas que degradam o controle. Neste caso especificamente, a da degradação inicial é causada devido à descontinuidade de tempo gerada no inicio para corrigir os relógios. Isto causa rechaveamento de contexto temporal atrapalhando o agendamento das tarefas.



(a) (b)

Figura 7.18 – Caso 5: (a) Correções do Relógio. (b) Diferenças de Tempo.

É possível observar, em comparação com os outros casos, na Figura 7.17 (b) que a lei de controle também possui uma degradação visível. A Figura 7.18 (a), mostra as correções de tempo aplicadas devidas à correção calculada pelo algoritmo FTM, a cada 0.2 segundos; e a Figura 7.18 (b) mostra as diferenças de tempo entre os relógios do conjunto: Nó 1 (Relógio 1 – Azul), Sensor (Relógio 2 – Verde), Controlador (Relógio 3 – Vermelho) e Atuador (Relógio 4 – Amarelo). Todos os relógios foram sincronizados entre uma precisão de 1 a -3 milissegundos, como mostra a Figura 7.18 (b), aumentando a imprecisão em relação ao caso 4. A Figura 7.19 mostra em (a) o agendador da rede; em (b) as linhas de tempo.



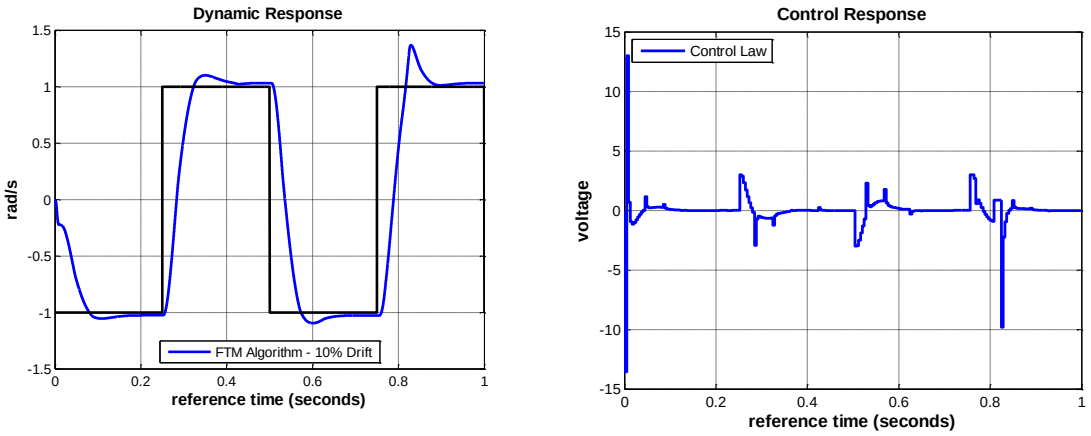
(a) (b)

Figura 7.19 – Caso 5: (a) Agendador da Rede. (b) Linhas de Tempo.

Neste caso, com 1% de deriva do sensor, é observada uma pequena diferença da lei de controle e da resposta dinâmica em relação aos outros casos. Os relógios se sincronizam, mas o sistema é afetado pela descontinuidade de tempo (atraso no tempo).

7.1.7. Caso 6 - FTM

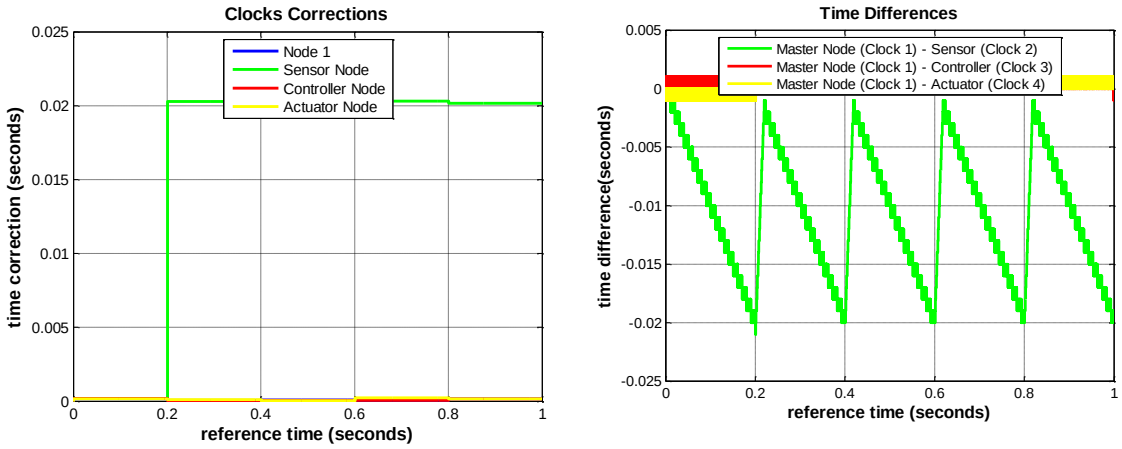
Da Figura 7.20 à Figura 7.22 mostram-se os resultados simulados de um NCS com 10% de deriva no relógio do sensor (chamado de caso 6), ou seja, o relógio do sensor tem 10% de velocidade maior que os outros nós da rede, caracterizando um relógio com falha. Foi utilizado como algoritmo de sincronização de relógios o algoritmo FTM.



(a) (b)

Figura 7.20 – Caso 6: 10% de Deriva: (a) Resposta Dinâmica. (b) Lei de Controle.

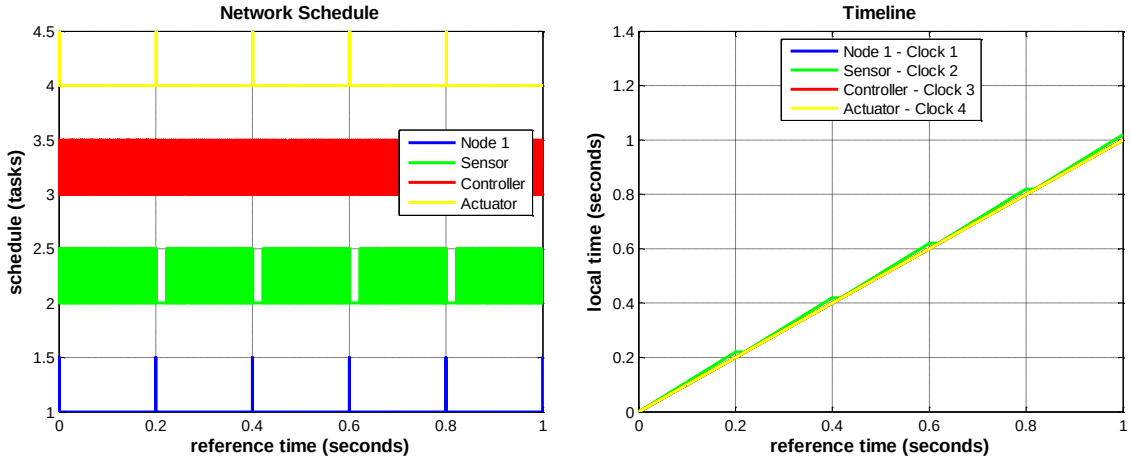
A Figura 7.21 (a), mostra as correções de tempo aplicadas devidas à correção calculada pelo algoritmo PTP, a cada 0.2 segundos; e a Figura 7.21 (b) mostra as diferenças de tempo entre os relógios do conjunto: Nó 1 (Relógio 1 – Azul), Sensor (Relógio 2 – Verde), Controlador (Relógio 3 – Vermelho) e Atuador (Relógio 4 – Amarelo). Todos os relógios foram sincronizados entre uma precisão de 1 a -20 milissegundos, como mostra a Figura 7.21 (b), aumentando a imprecisão em relação aos casos anteriores.



(a) (b)

Figura 7.21 – Caso 6: (a) Correções do Relógio. (b) Diferenças de Tempo.

A Figura 7.22 mostra em (a) o agendador da rede; em (b) as linhas de tempo.



(a) (b)

Figura 7.22 – Caso 6: (a) Agendador da Rede. (b) Linhas de Tempo.

Com 10% de deriva no sensor, é possível observar na Figura 7.20 uma grande diferença na resposta dinâmica e na lei de controle. Apesar de o algoritmo de sincronização FTM cumprir o seu papel de sincronizar os relógios (mesmo sobre uma precisão alta), a resposta dinâmica e a lei de controle são degradadas, especialmente durante as transições e inicializações do sistema. . O controle é afetado devido à introdução de descontinuidades de tempo que geram atraso nas tarefas de controle e rechaveamento de contexto temporal. No entanto, durante o período estacionário efeito de perturbação é muito menor

que durante um período de transição e inicialização, pois o atraso e rechaveamento causam perdas e atrasos de tarefas que degradam o controle. Este caso, também é importante, pois também mostra que os problemas de descontinuidade posam como um consistente problema de sistemas de controle distribuídos que necessitam de sincronização de relógios por métrica de precisão, principalmente sistemas com um ciclo de vida grande como no caso de sistemas aeroespaciais.

7.2. Técnica de Amortização - Função Rampa

Diversos casos foram simulados para verificar e validar a técnica do "*cross-fading*" utilizando uma função rampa. A Tabela 7.2 mostra os estudos de casos realizados que simulam um algoritmo genérico de média descrito pela equação de convergência (5.8), uma função "*cross-fading*" do tipo rampa descrito pela equação (5.9), um algoritmo FTM descrito pela equação de convergência (2.12).

Tabela 7.2 - Casos de Estudos - Amortização por Função Rampa.

Caso	Algoritmo de Sincronização	Imperfeição	T (Amostragem)	R (período de Re-sincronização)	M (Chaveamento)	G (Ganho)
1	Média (2 relógios)	Deriva = 0.1%	0.01	20	20	1
2	Média (2 relógios)	Deriva = 0.1%	0.01	20	10	0.25
3	Média (2 relógios)	Deriva = 0.1%	0.01	20	0	0.1
4	Média (2 relógios)	Viés Inicial = 0.1 (s)	0.01	20	20	1
5	Média (2 relógios)	Viés Inicial = 0.1 (s)	0.01	20	10	0.25
6	Média (2 relógios)	Viés Inicial = 0.1 (s)	0.01	20	0	0.1
7	FTM (4 relógios)	Deriva = 0.1%	0.01	20	20	1
8	FTM (4 relógios)	Deriva = 0.1%	0.01	20	10	0.25

Efeitos de atraso de rede, não são considerados. Os resultados apresentados na seqüência mostram as vantagens e desvantagens de se utilizar uma função rampa para suavizar as correções dos algoritmos de sincronização de relógios.

7.2.1. Caso 1 - Função Rampa - Média - 2 relógios

Este caso simula um algoritmo genérico de média descrito pela equação (5.8), com ganho (G) unitário, período de re-sincronização (R) de 20 *ticks*, chaveamento (M) de 20 *ticks* e com o relógio 1 e 3 com 0.1% de deriva. Com o chaveamento em 20, significa que o "*cross-fading*" é um degrau, pois o chaveamento é feito logo após o primeiro instante e portanto todo o ganho é aplicado em um único tick.

A Figura 7.23 apresenta os estados propagados dos relógios com e sem utilizar a função "*cross-fading*". A Figura 7.24 apresenta a precisão dos relógios, isto é diferenças entre relógio 1 e relógio 2; e diferença entre relógio 3 e 4. Por ser um algoritmo que sua métrica é a precisão, nos casos com o algoritmo média não são levados em conta à exatidão (diferença dos relógios para o relógio de referência real). A Figura 7.25 mostra o gráfico das correções calculadas e aplicadas aos estados do relógio, com e sem "cross-fading". A Figura 7.26 mostra as funções "cross-fading" dos relógios 1 e 2.

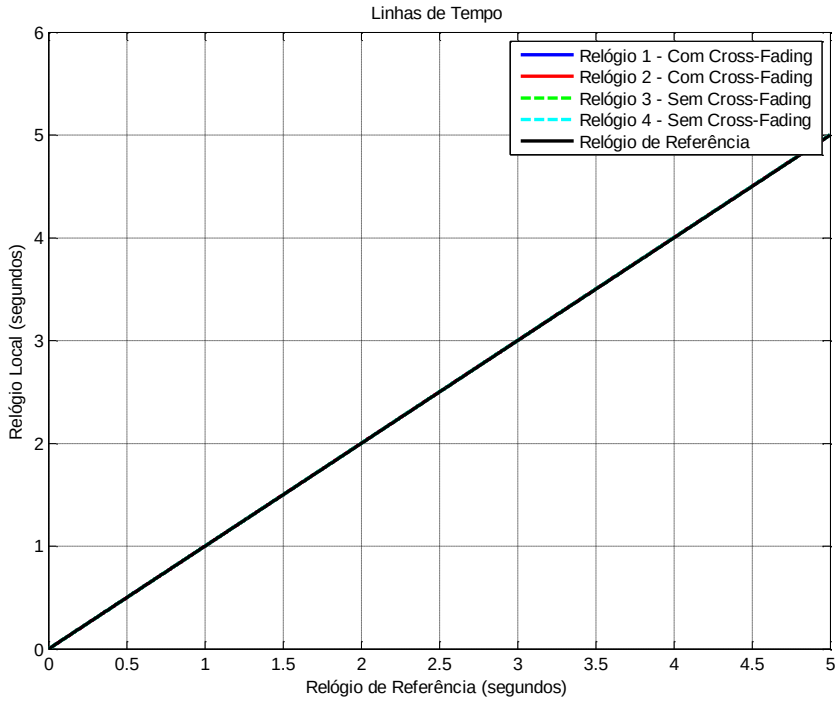


Figura 7.23 – Caso 1 - Cross-Fading Rampa: Linhas de Tempo.

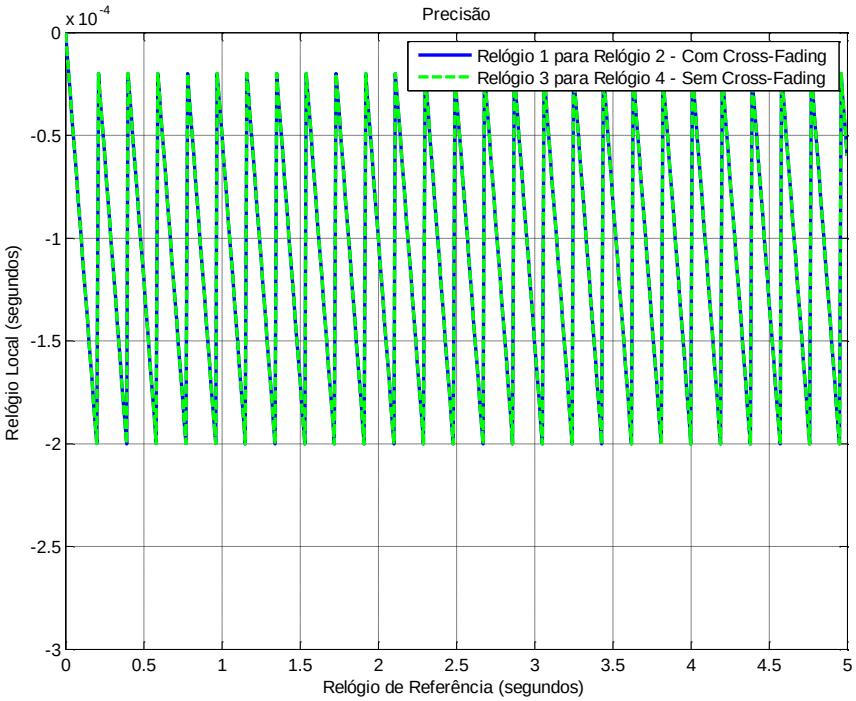


Figura 7.24 – Caso 1 - Cross-Fading Rampa: Precisão dos relógios.

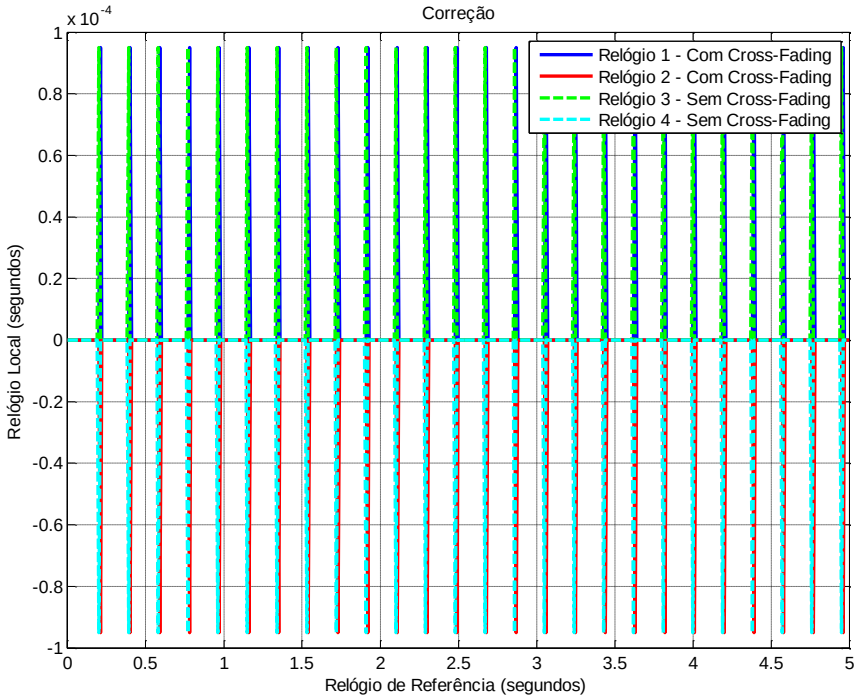


Figura 7.25 – Caso 1 - Cross-Fading Rampa: Correções aplicadas.

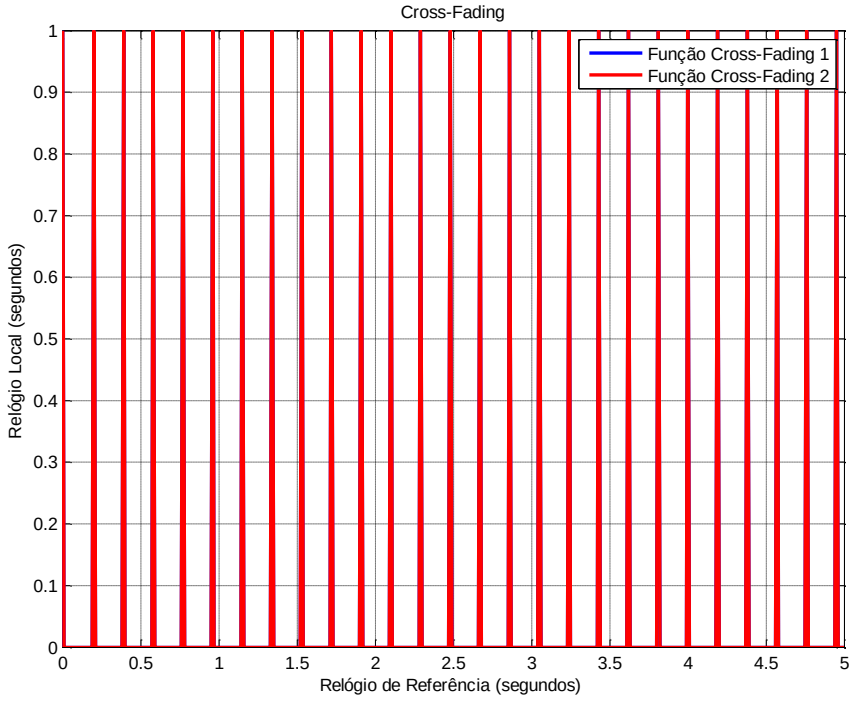


Figura 7.26 – Caso 1: Função *"cross-fading"*.

É possível observar que os resultados com e sem "cross-fading" obtiveram resultados iguais. Os resultados foram iguais pois ambos os algoritmos aplicam um degrau no instante de re-sincronização, atualizando os relógios. O objetivo

de amortizar a correção ao longo do tempo, neste caso, não foi alcançado. No entanto, este caso demonstra a generalidade da técnica ao reproduzir o mesmo resultado que o algoritmo sem *"cross-fading"*.

7.2.2. Caso 2 - Função Rampa - Média - 2 relógios

Este caso simula um algoritmo genérico de média descrito pela equação (5.8), com ganho (G) 0.25, período de re-sincronização (R) de 20 *ticks*, chaveamento (M) de 10 *ticks* e com o relógio 1 e 3 com 0.1% de deriva. Com o chaveamento em 10, significa que o "*cross-fading*" é uma rampa que será chaveada para zero após 10 ticks.

A Figura 7.27 apresenta os estados propagados dos relógios com e sem utilizar a função "*cross-fading*". A Figura 7.28 apresenta a precisão dos relógios, isto é diferenças entre relógio 1 e relógio 2; e diferença entre relógio 3 e 4. A Figura 7.29 mostra o gráfico das correções calculadas e aplicadas aos estados do relógio, com e sem "*cross-fading*". A Figura 7.30 mostra as funções "*cross-fading*" dos relógios 1 e 2.

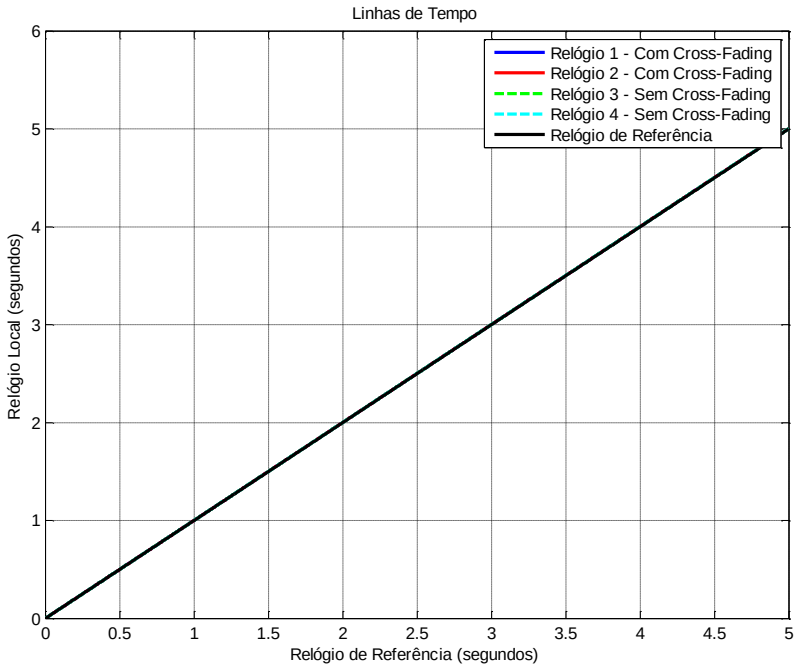


Figura 7.27 – Caso 2 - *Cross-Fading* Rampa: Linhas de Tempo.

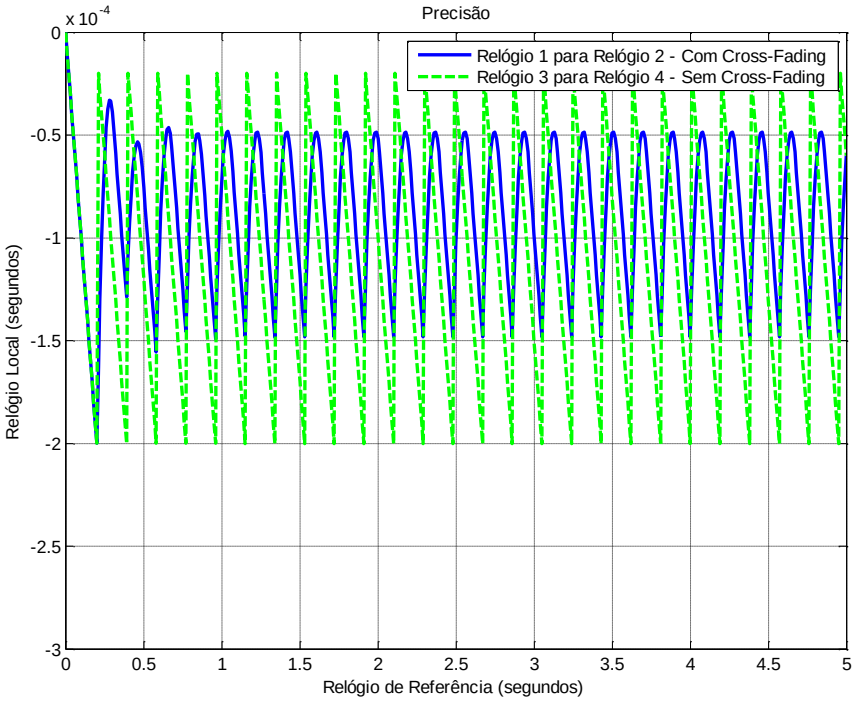


Figura 7.28 – Caso 2 - *Cross-Fading* Rampa: Precisão dos relógios.

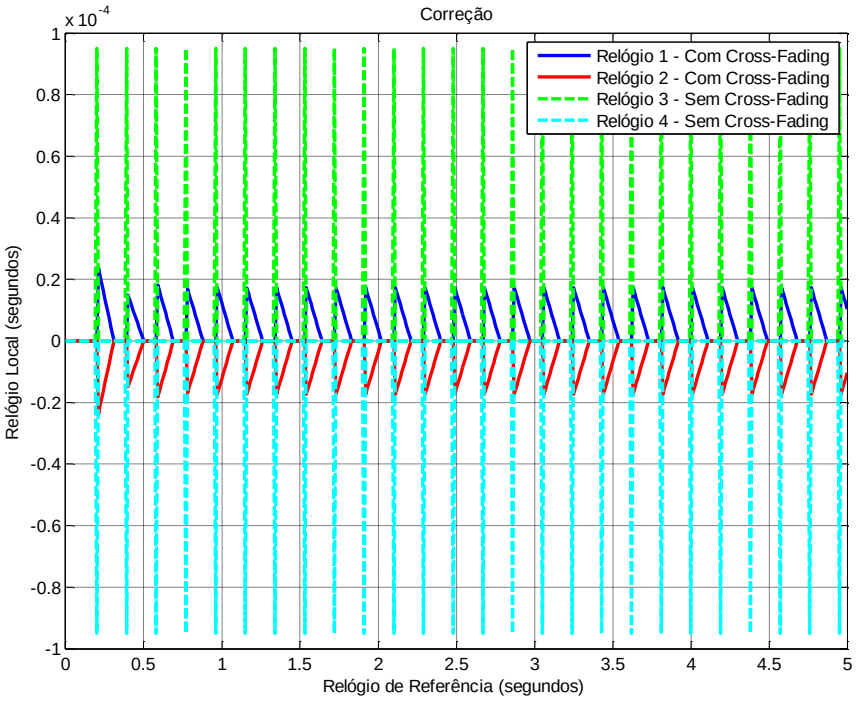


Figura 7.29 – Caso 2 - *Cross-Fading* Rampa: Correções aplicadas.

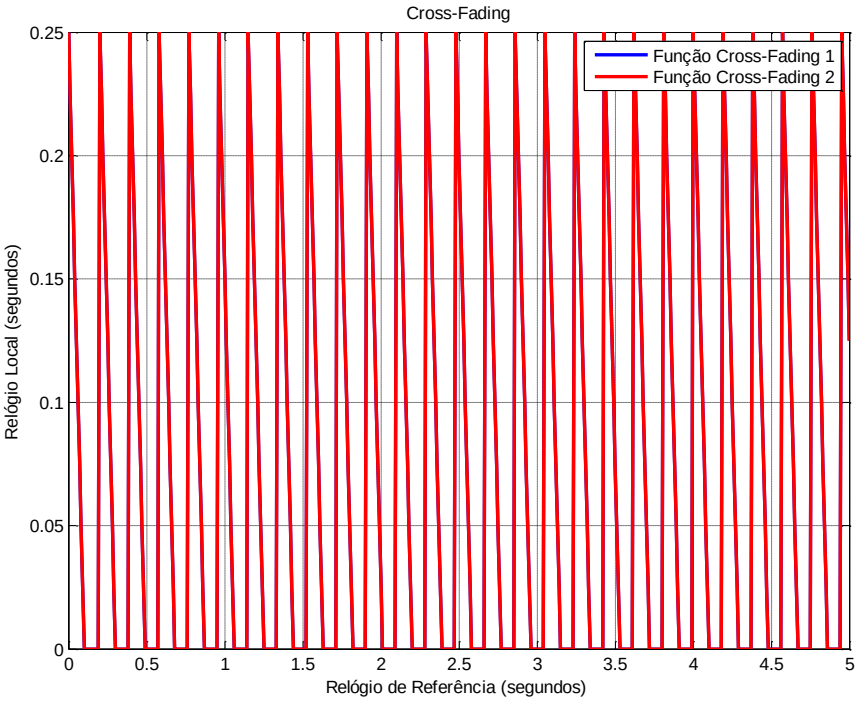


Figura 7.30 – Caso 2: Função *"cross-fading"*.

Neste caso, é possível observar que os relógios com "*cross-fading*" tiveram uma melhora em sua precisão em comparação com os relógios sem "cross-fading". Mas é possível observar na Figura 7.28 que a média do conjunto (precisão dos relógios) foi um pouco alterada em relação ao caso sem "cross-fading". O objetivo de amortizar a correção ao longo do tempo foi alcançado, além de melhorar a precisão dos relógios.

7.2.3. Caso 3 - Função Rampa - Média - 2 relógios

Este caso simula um algoritmo genérico de média descrito pela equação (5.8), com ganho (G) 0.1, período de re-sincronização (R) de 20 *ticks*, chaveamento (M) de 0 *ticks* e com o relógio 1 e 3 com 0.1% de deriva. Com o chaveamento em 0, significa que o "*cross-fading*" é uma rampa e seu chaveamento será feito no instante da próxima re-sincronização. A Figura 7.31 apresenta os estados propagados dos relógios com e sem utilizar a função "*cross-fading*". A Figura 7.32 apresenta a precisão dos relógios, isto é diferenças entre relógio 1 e relógio 2; e diferença entre relógio 3 e 4. A Figura 7.33 mostra o gráfico das

correções calculadas e aplicadas aos estados do relógio, com e sem "*cross-fading*". A Figura 7.34 mostra as funções "*cross-fading*" dos relógios 1 e 2.

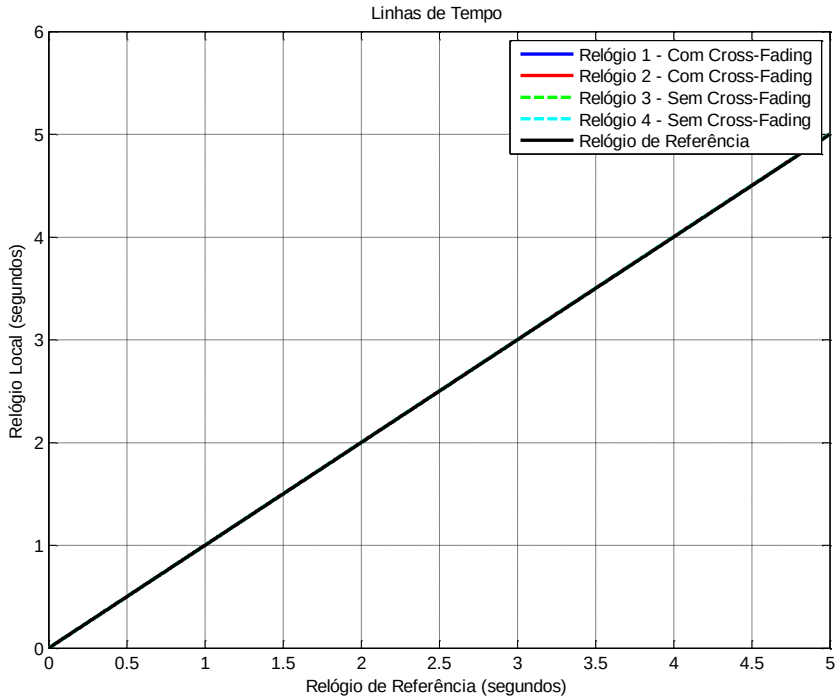


Figura 7.31 – Caso 3 - *Cross-Fading* Rampa: Linhas de Tempo.

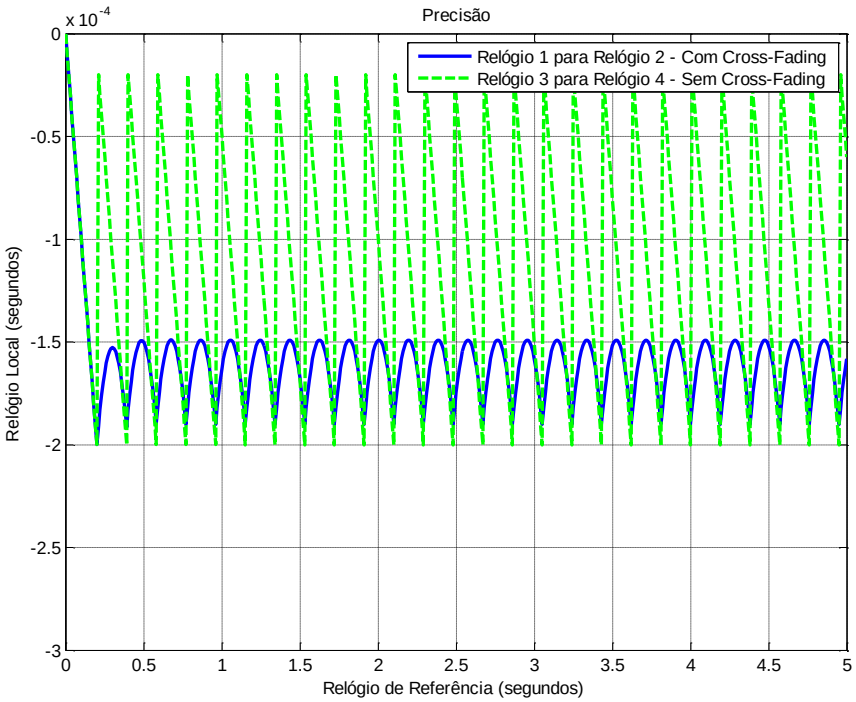


Figura 7.32 – Caso 3 - *Cross-Fading* Rampa: Precisão dos relógios.

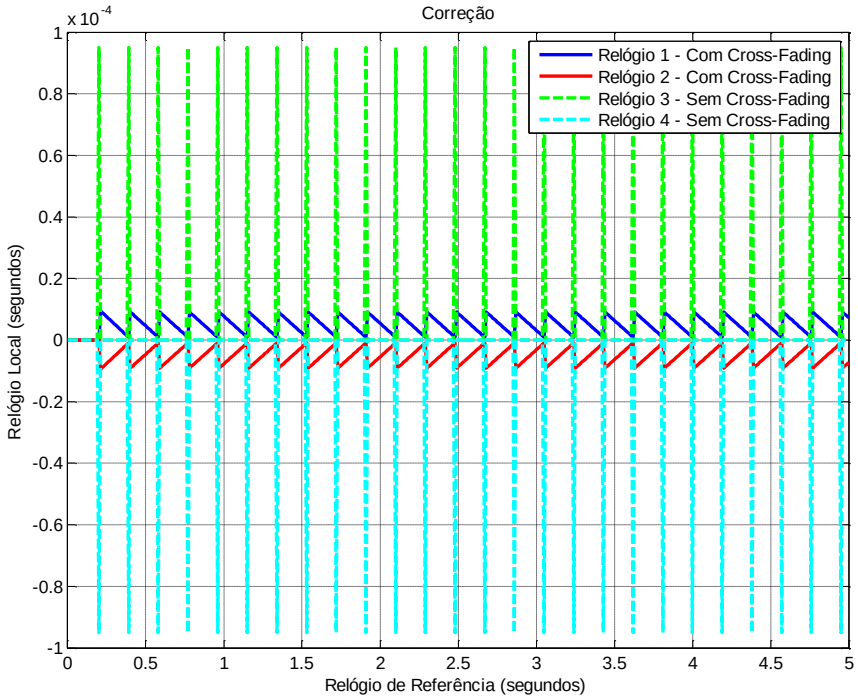


Figura 7.33 – Caso 3 - *Cross-Fading* Rampa: Correções aplicadas.

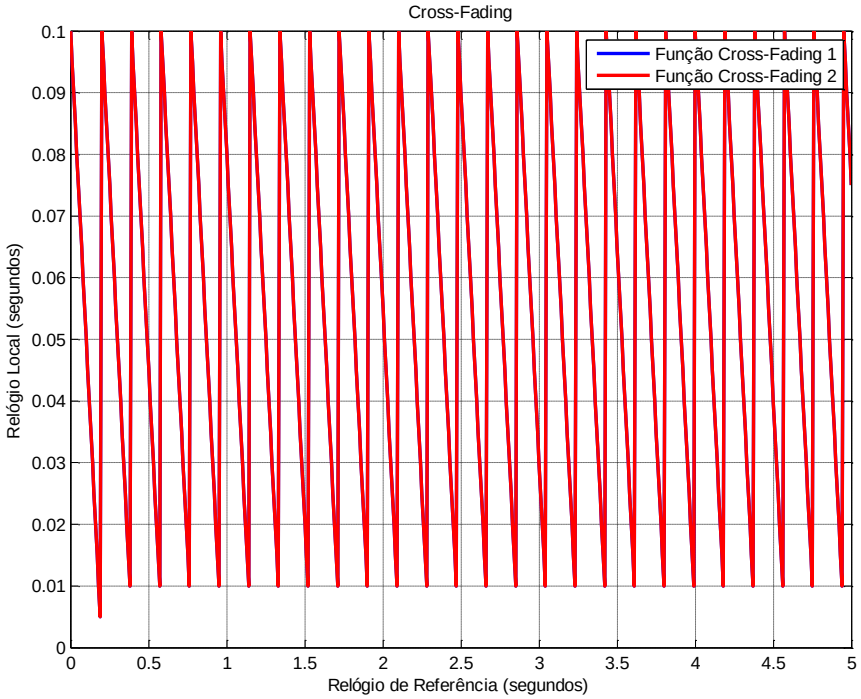


Figura 7.34 – Caso 3: Função *"cross-fading"*.

Neste caso, é possível observar que os relógios com "*cross-fading*" também tiveram uma melhora em sua precisão em comparação com os relógios sem

"*cross-fading*". Mas é possível observar na Figura 7.32 que a média do conjunto foi alterada em relação ao caso sem "cross-fading", o que pode impactar na exatidão. O objetivo de amortizar a correção ao longo do tempo foi alcançado, além de melhorar a precisão dos relógios.

7.2.4. Caso 4 - Função Rampa - Média - 2 relógios

Este caso simula um algoritmo genérico de média descrito pela equação (5.8), com ganho (G) 1, período de re-sincronização (R) de 20 *ticks*, chaveamento (M) de 20 *ticks* e com viés inicial de 0.1(segundos) no relógio 1 e 3. Com o chaveamento em 20, significa que o "*cross-fading*" é um degrau.

A Figura 7.35 apresenta os estados propagados dos relógios com e sem utilizar a função "*cross-fading*". A Figura 7.36 apresenta a precisão dos relógios, isto é diferenças entre relógio 1 e relógio 2; e diferença entre relógio 3 e 4. A Figura 7.37 mostra o gráfico das correções calculadas e aplicadas aos estados do relógio, com e sem "*cross-fading*". A Figura 7.38 mostra as funções "*cross-fading*" dos relógios 1 e 2.

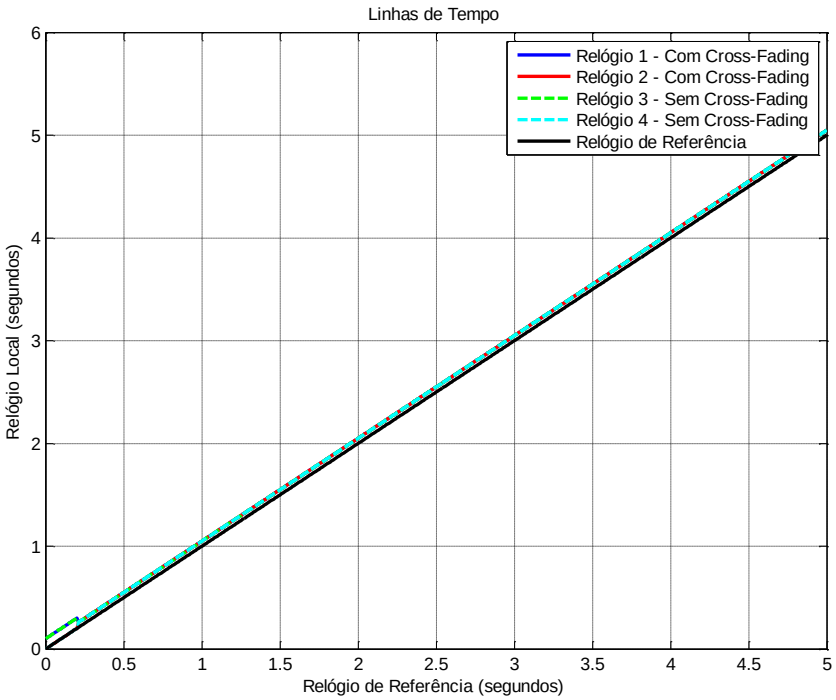


Figura 7.35 – Caso 4 - *Cross-Fading* Rampa: Linhas de Tempo.

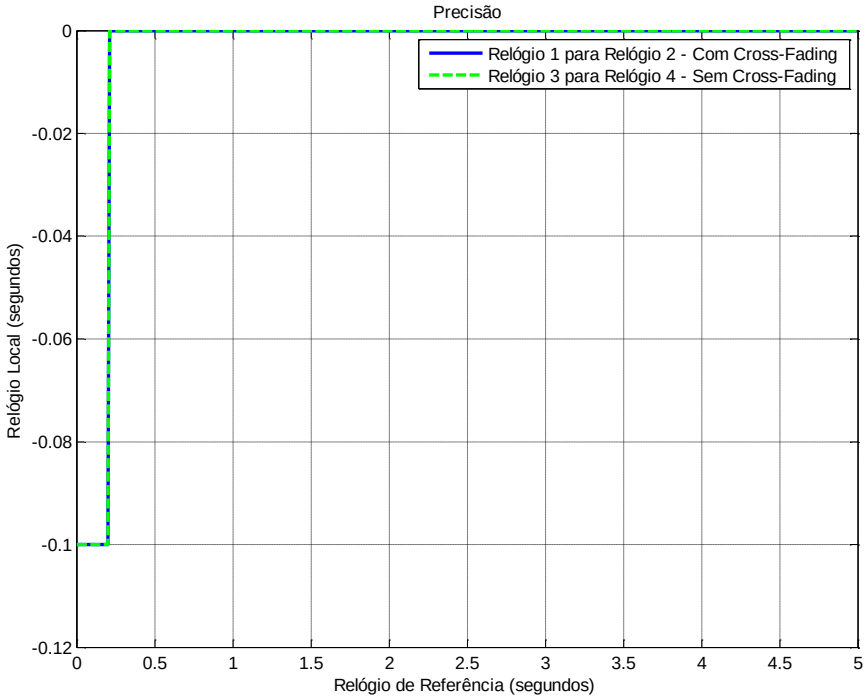


Figura 7.36 – Caso 4 - *Cross-Fading* Rampa: Precisão dos relógios.

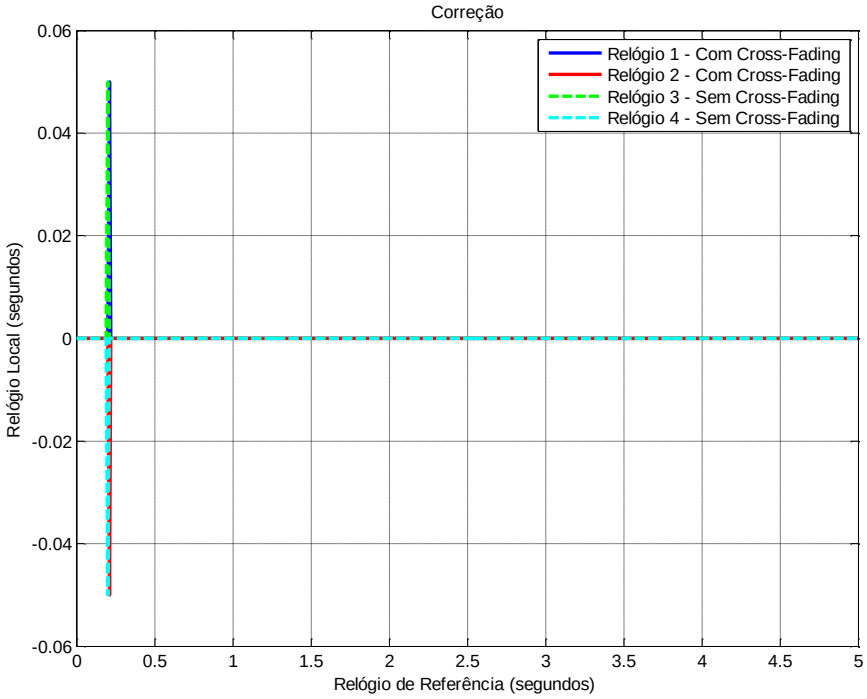


Figura 7.37 – Caso 4 - *Cross-Fading* Rampa: Correções aplicadas.

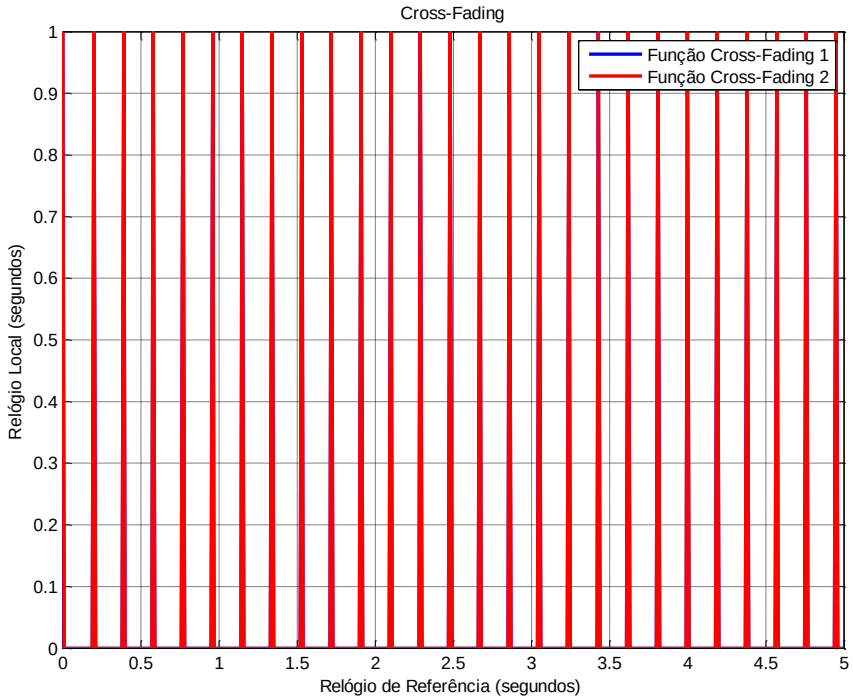


Figura 7.38 – Caso 4 - Função *"cross-fading"*.

É possível observar que os resultados com e sem "*cross-fading*" obtiveram resultados iguais. Os resultados foram iguais pois ambos os algoritmos aplicam um degrau no instante de re-sincronização, atualizando os relógios. O objetivo de amortizar a correção, não foi alcançado. No entanto, este caso demonstra a generalidade da técnica ao reproduzir o mesmo resultado que o algoritmo sem "cross-fading".

7.2.5. Caso 5 - Função Rampa - Média - 2 relógios

Este caso simula um algoritmo genérico de média descrito pela equação (5.8), com ganho (G) 0.25, período de re-sincronização (R) de 20 *ticks*, chaveamento (M) de 10 *ticks* e com viés inicial de 0.1(segundos) no relógio 1 e 3. Com o chaveamento em 10, significa que o "*cross-fading*" é uma rampa que será chaveada para zero após 10 ticks. A Figura 7.39 apresenta os estados propagados dos relógios com e sem utilizar a função "*cross-fading*". A Figura 7.40 apresenta a precisão dos relógios, isto é diferenças entre relógio 1 e relógio 2; e diferença entre relógio 3 e 4. A Figura 7.41 mostra o gráfico das

correções calculadas e aplicadas aos estados do relógio, com e sem "*cross-fading*". A Figura 7.42 mostra as funções "*cross-fading*" dos relógios 1 e 2.

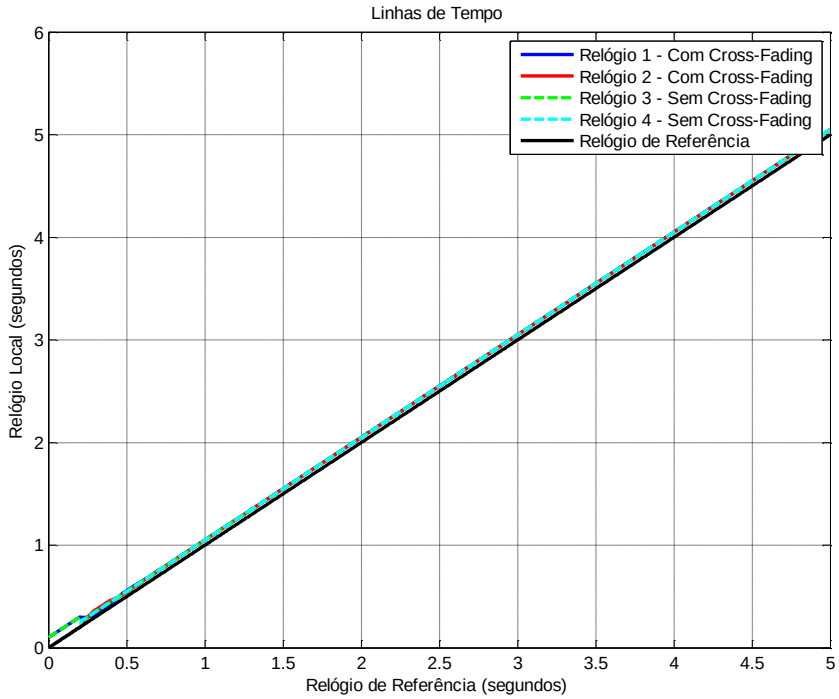


Figura 7.39 – Caso 5 - *Cross-Fading* Rampa: Linhas de Tempo.

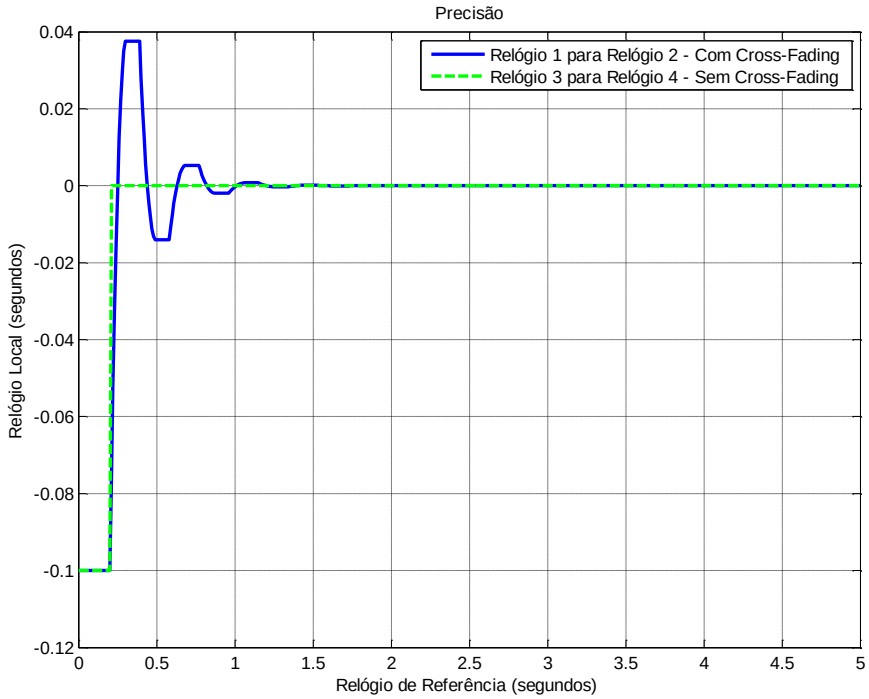


Figura 7.40 – Caso 5 - *Cross-Fading* Rampa: Precisão dos relógios.

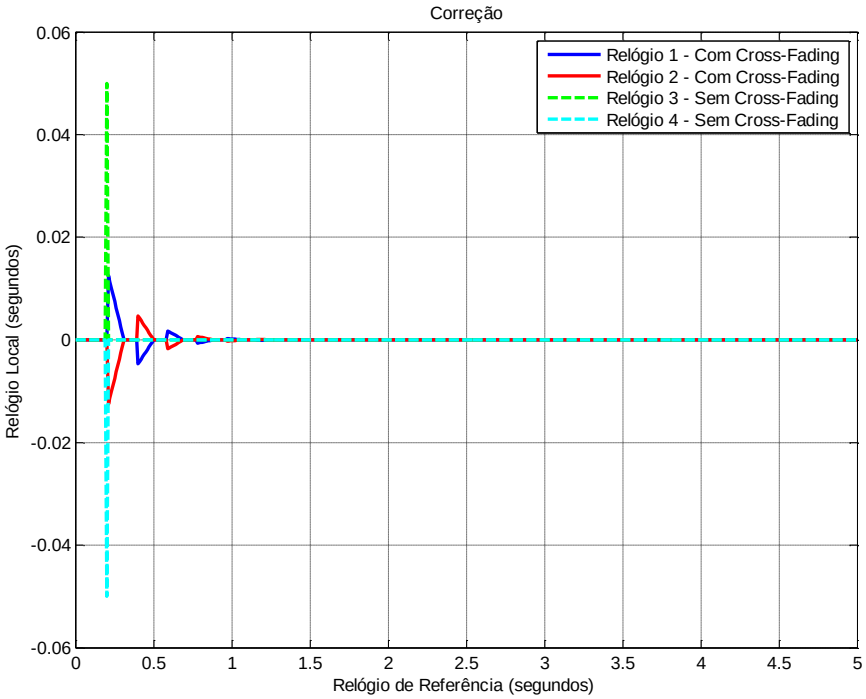


Figura 7.41 – Caso 5 - *Cross-Fading* Rampa: Correções aplicadas.

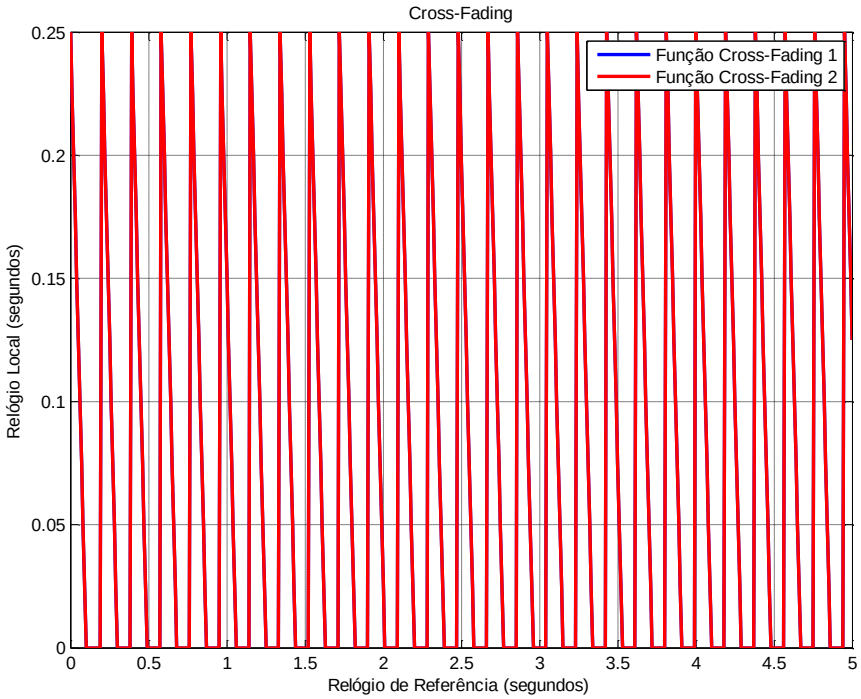


Figura 7.42 – Caso 5 - Função *"cross-fading"*.

Neste caso, é possível observar que os relógios com "*cross-fading*" amorteceram o impacto inicial devido ao viés inicial. Na Figura 7.40 é possível observar que o "cross-fading" causa um "*overshoot*". O objetivo de amortizar a

correção ao longo do tempo foi alcançado, mas não foi satisfatório por causa do *overshoot* causado inicialmente. O resultado com "*cross-fading*" em comparação com o resultado sem "*cross-fading*" foi pior para a sincronização de relógios.

7.2.6. Caso 6 - Função Rampa - Média - 2 relógios

Este caso simula um algoritmo genérico de média descrito pela equação (5.8), com ganho (G) 0.1, período de re-sincronização (R) de 20 *ticks*, chaveamento (M) de 0 *ticks* e com viés inicial de 0.1(segundos) no relógio 1 e 3. Com o chaveamento em 0, significa que o "*cross-fading*" é uma rampa que será chaveada para zero somente no próximo intervalo de re-sincronização. A Figura 7.43 apresenta os estados propagados dos relógios com e sem utilizar a função "*cross-fading*". A Figura 7.44 apresenta a precisão dos relógios, isto é diferenças entre relógio 1 e relógio 2; e diferença entre relógio 3 e 4. A Figura 7.45 mostra o gráfico das correções calculadas e aplicadas aos estados do relógio, com e sem "*cross-fading*". A Figura 7.46 mostra as funções "*cross-fading*" dos relógios 1 e 2.

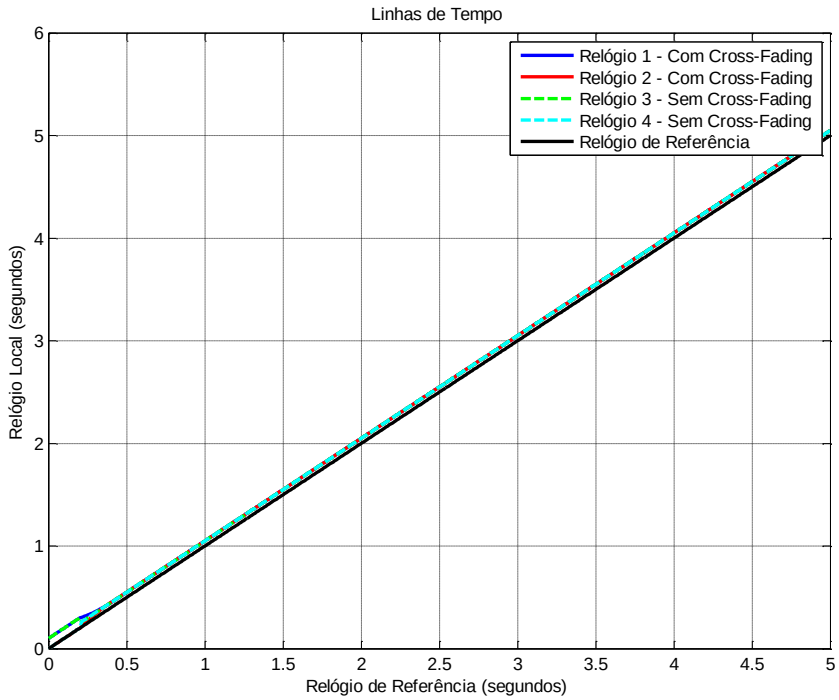


Figura 7.43 – Caso 6 - *Cross-Fading* Rampa: Linhas de Tempo.

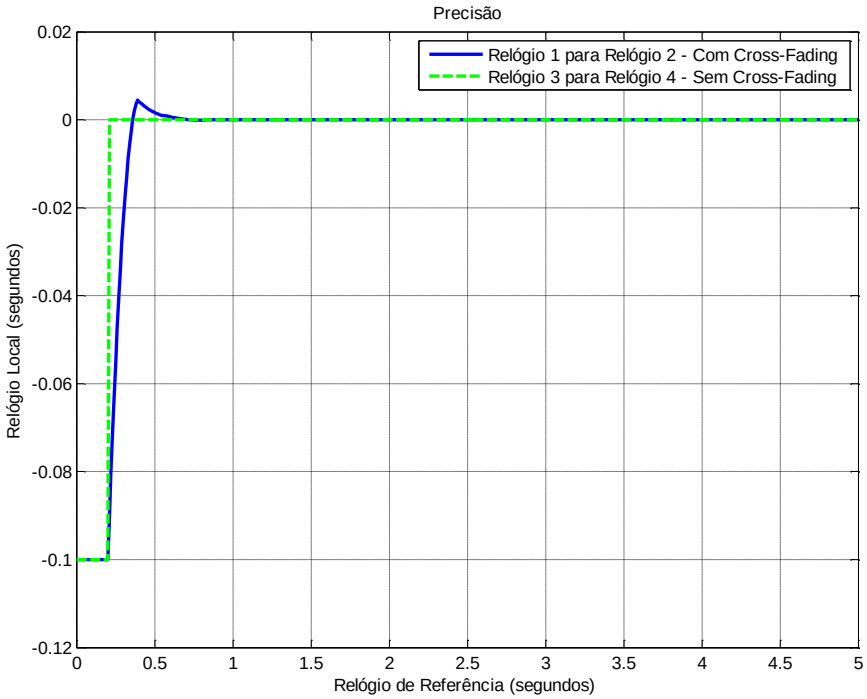


Figura 7.44 – Caso 6 - *Cross-Fading* Rampa: Precisão dos relógios.

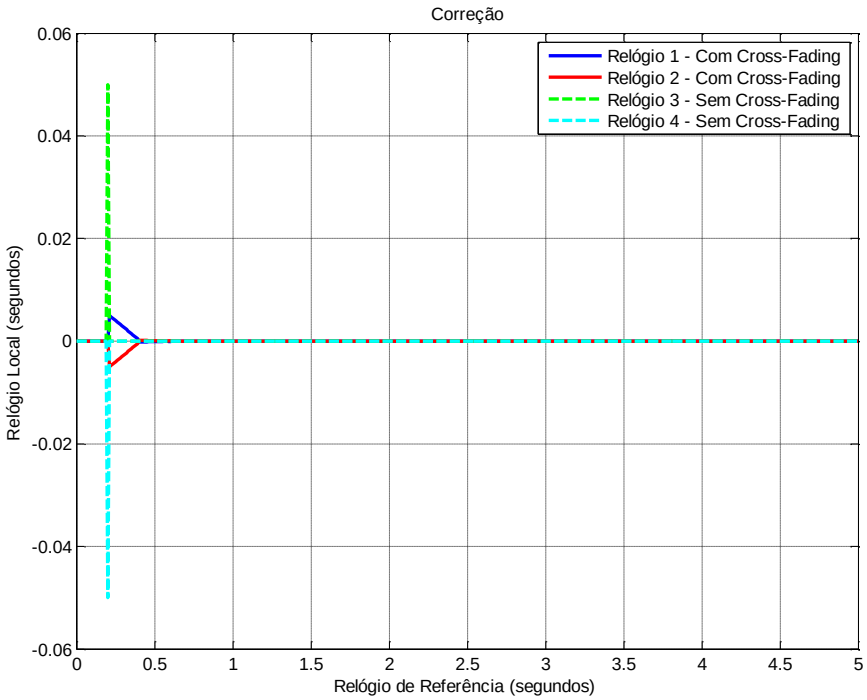


Figura 7.45 – Caso 6 - *Cross-Fading* Rampa: Correções aplicadas.

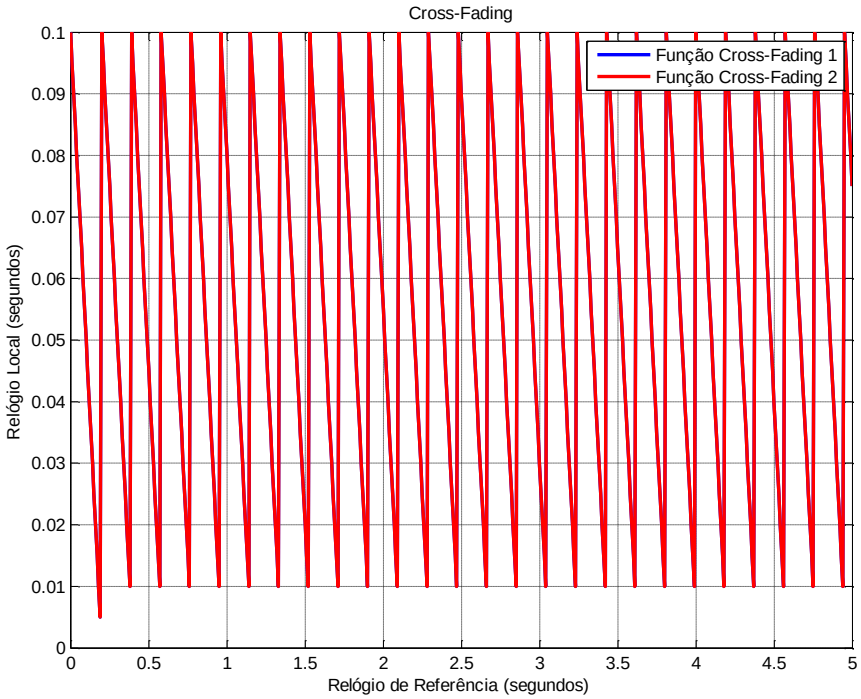


Figura 7.46 – Caso 6 - Função *"cross-fading"*.

Neste caso, é possível observar que os relógios com "*cross-fading*" amorteceram o impacto inicial devido ao viés inicial. Na Figura 7.44 é possível observar que o "cross-fading" causa um pequeno "*overshoot*". O objetivo de amortizar a correção ao longo do tempo foi alcançado, mas houve um pequeno "*overshoot"* inicialmente. O resultado com "*cross-fading*" em comparação com o resultado sem "*cross-fading*" foi satisfatório para a sincronização de relógios.

Em uma rápida análise dos casos 5 e 6, pode-se concluir que ajustes nos parâmetros do "*cross-fading*" podem melhorar e/ou igualar os resultados em comparação o caso sem "*cross-fading*".

7.2.7. Caso 7 - Função Rampa - FTM - 4 relógios

Este caso simula o algoritmo FTM, por métrica de precisão, com ganho (G) 1, período de re-sincronização (R) de 20 *ticks*, chaveamento (M) de 20 *ticks* e com deriva de 0.1% no relógio 1. O chaveamento em 20, significa que o "*cross-fading*" é um degrau.

A Figura 7.47 apresenta os estados propagados dos relógios com (lado esquerdo) e sem (lado direito) aplicar a função "*cross-fading*". A Figura 7.48 apresenta a precisão dos relógios (com e sem "*cross-fading*"), isto é a diferença entre os relógios do conjunto. A Figura 7.49 mostra o gráfico das correções calculadas e aplicadas aos estados dos relógios, com (lado esquerdo) e sem (lado direito) "*cross-fading*". A Figura 7.50 mostra as funções "*cross-fading*" dos relógios.

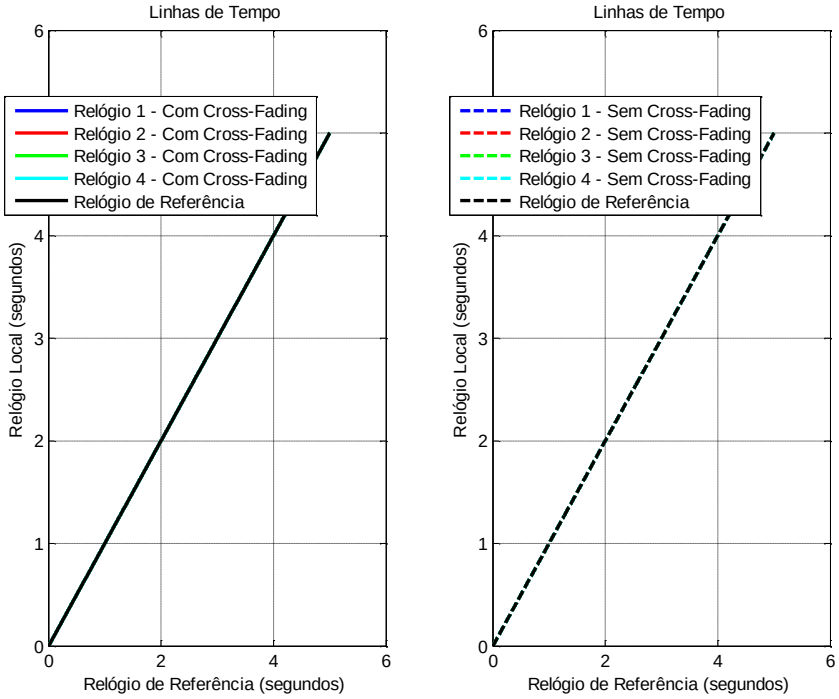


Figura 7.47 – Caso 7 - *Cross-Fading* Rampa: Linhas de Tempo.

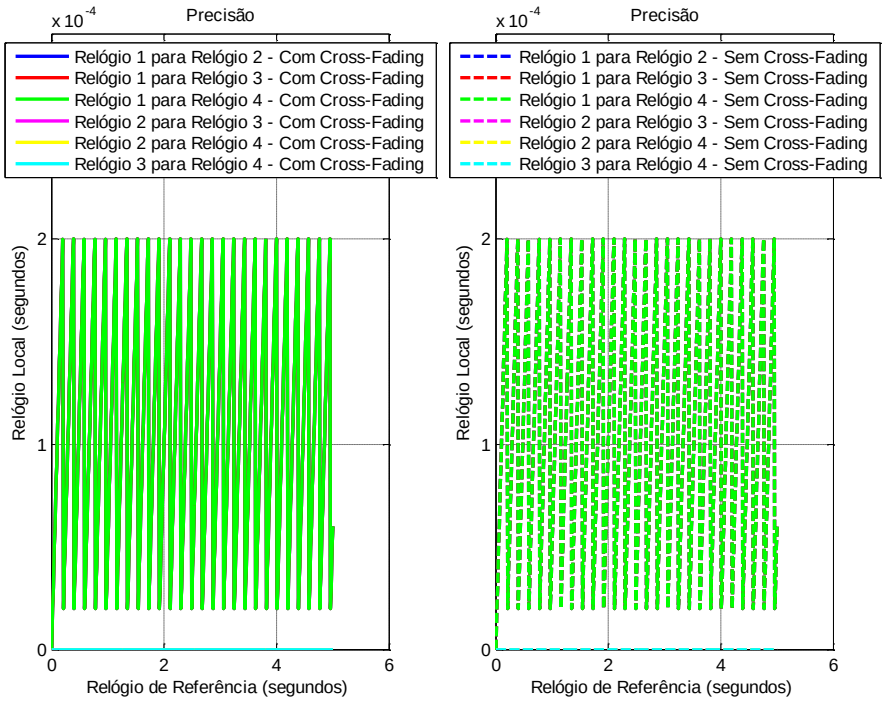


Figura 7.48 – Caso 7 - *Cross-Fading* Rampa: Precisão dos relógios.

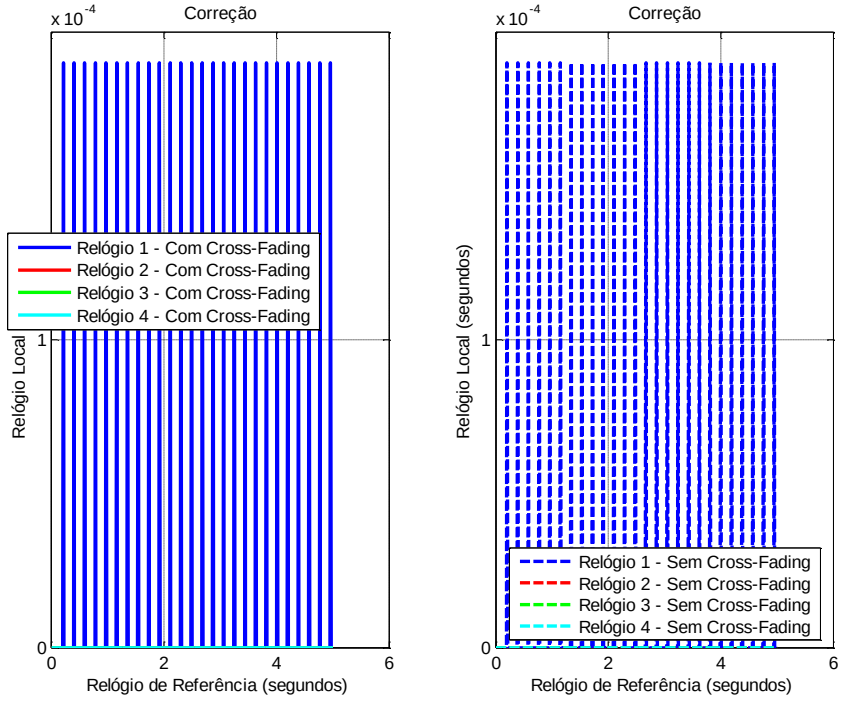


Figura 7.49 – Caso 7 - *Cross-Fading* Rampa: Correções aplicadas.

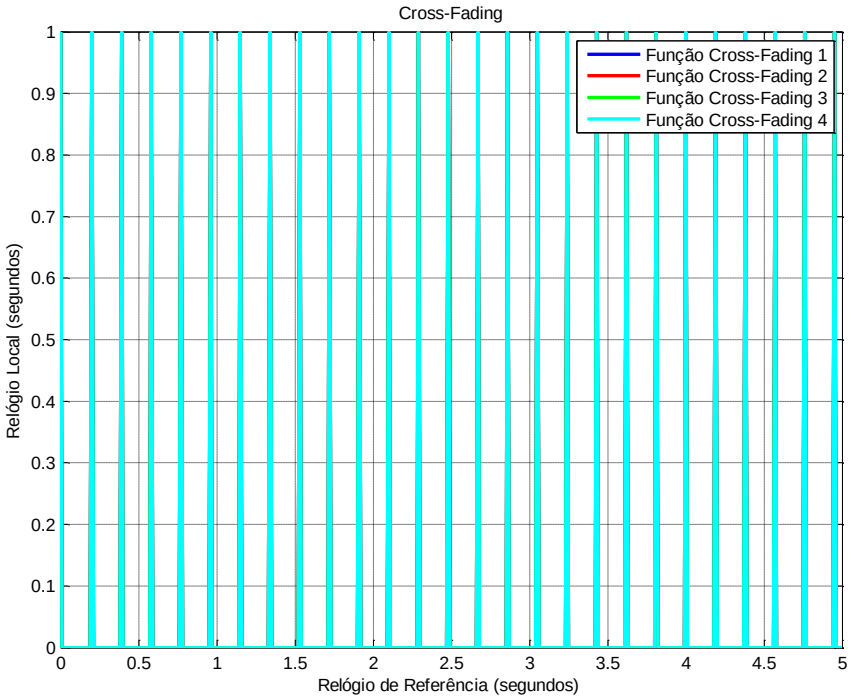


Figura 7.50 – Caso 7 - Função *"cross-fading"*.

É possível observar que os resultados com e sem "cross-fading" obtiveram resultados iguais. Os resultados foram iguais pois ambos os algoritmos aplicam um degrau no instante de re-sincronização, atualizando os relógios. O objetivo de amortizar a correção ao longo do tempo, neste caso, não foi alcançado. No entanto, este caso demonstra a generalidade da técnica ao reproduzir o mesmo resultado que o algoritmo sem "cross-fading".

7.2.8. Caso 8 - Função Rampa - FTM - 4 relógios

Este caso simula o algoritmo FTM, por métrica de precisão, com ganho (G) 0.25, período de re-sincronização (R) de 20 *ticks*, chaveamento (M) de 10 *ticks* e com deriva de 0.1% no relógio 1. O chaveamento em 10, significa que o "*cross-fading*" é uma rampa que será chaveada para zero após 10 ticks.

A Figura 7.51 apresenta os estados propagados dos relógios com (lado esquerdo) e sem (lado direito) aplicar a função "*cross-fading*". A Figura 7.52 apresenta a precisão dos relógios (com e sem "*cross-fading*"), isto é a diferença entre os relógios do conjunto. A Figura 7.53 mostra o gráfico das

correções calculadas e aplicadas aos estados dos relógios, com (lado esquerdo) e sem (lado direito) "*cross-fading*". A Figura 7.54 mostra as funções "*cross-fading*" dos relógios.

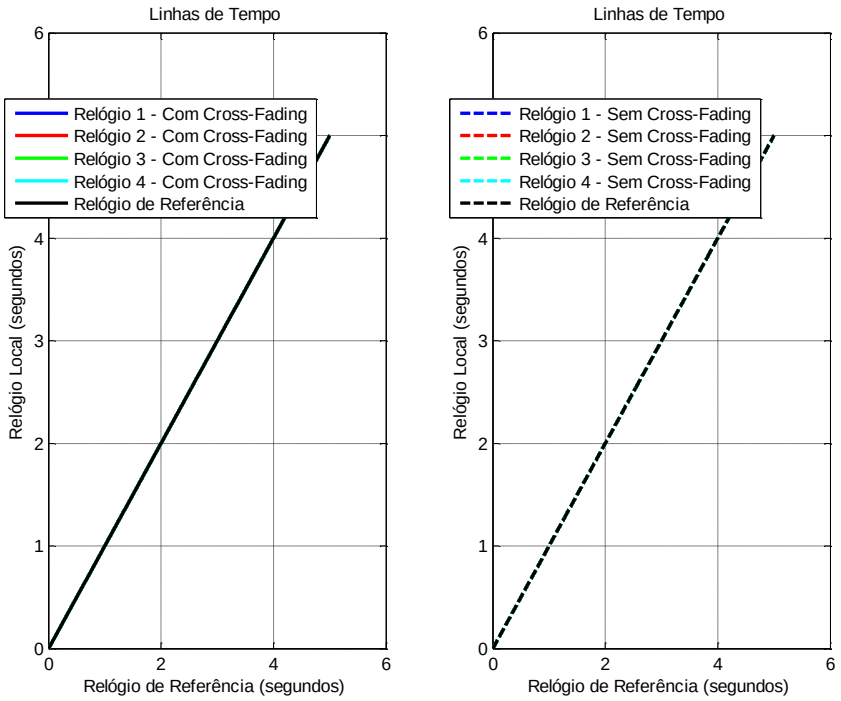


Figura 7.51 – Caso 8 - *Cross-Fading* Rampa: Linhas de Tempo.

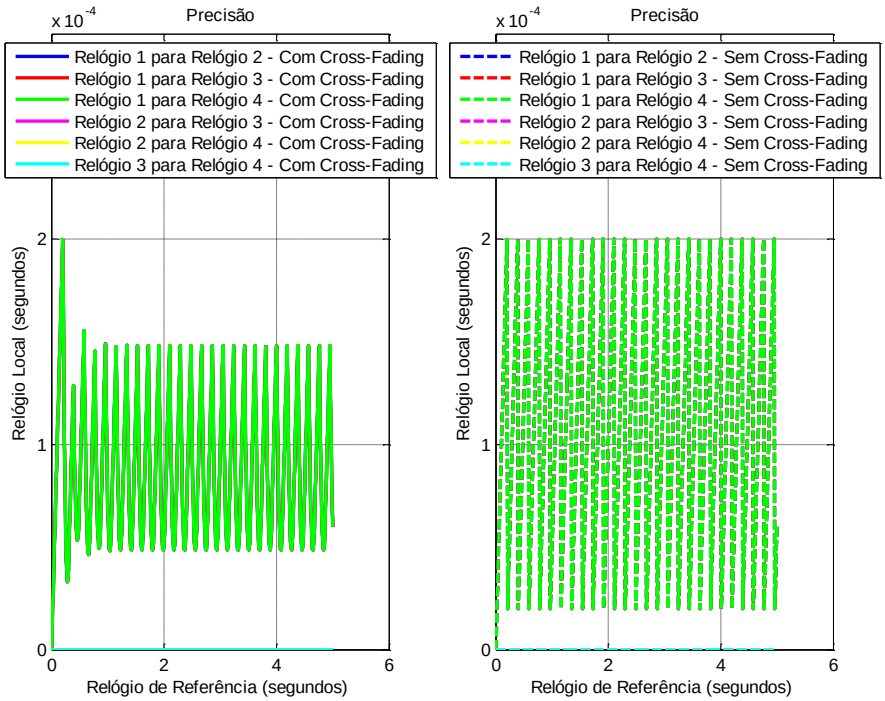


Figura 7.52 – Caso 8 - *Cross-Fading* Rampa: Precisão dos relógios.

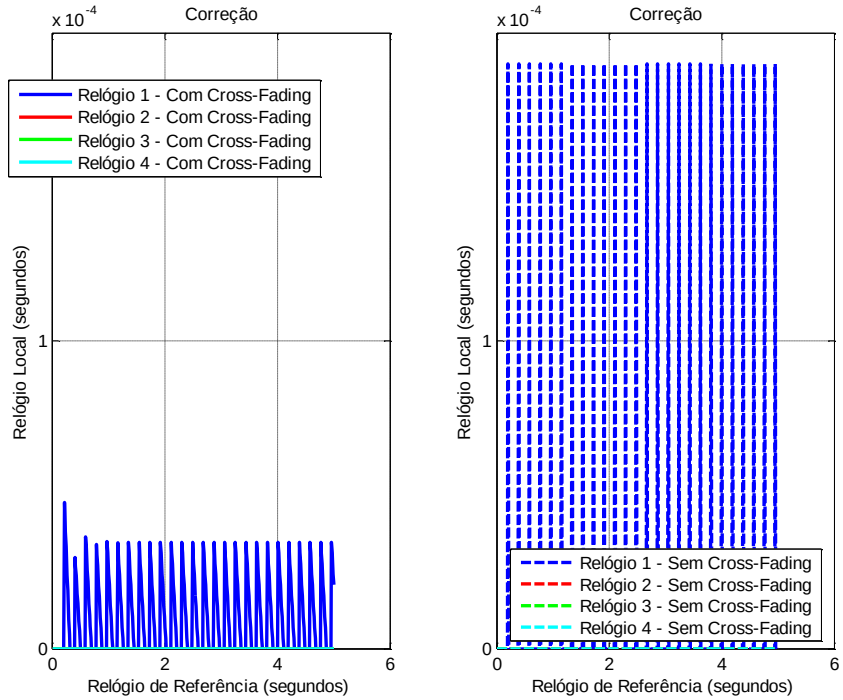


Figura 7.53 – Caso 8 - *Cross-Fading* Rampa: Correções aplicadas.

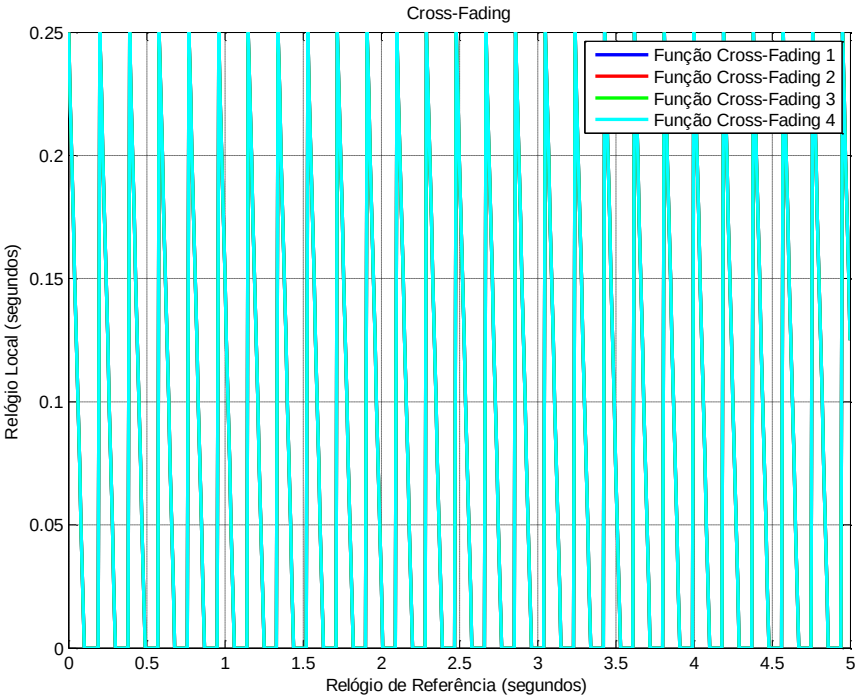


Figura 7.54 – Caso 8 - Função *"cross-fading"*.

Neste caso, é possível observar que os relógios com "*cross-fading*" tiveram uma melhora em sua precisão em comparação com os relógios sem "cross-

fading". O objetivo de amortizar a correção ao longo do tempo foi alcançado, além de melhorar a precisão dos relógios.

7.3. Técnica de amortização - Função Exponencial

Diversos casos foram simulados para verificar e validar a técnica do "cross-fading" utilizando uma função exponencial. A Tabela 7.3 mostra os estudos de casos realizados simulam um algoritmo genérico de média descrito pela equação de convergência (5.8), uma função "*cross-fading"* do tipo exponencial descrito pela equação (5.10), um algoritmo FTM descrito pela equação de convergência (2.12). Efeitos de atraso de rede, não são considerados.

Os resultados apresentados na seqüência, mostram as vantagens e desvantagens de se utilizar uma função exponencial para suavizar as correções dos algoritmos de sincronização de relógios.

Tabela 7.3 - Casos de Estudos - Amortização por Função Exponencial.

Caso	Algoritmo de Sincronização	Imperfeição	T	R	M	G
1	Média (2 relógios)	Deriva = 0.1%	0.01	20	20	1
2	Média (2 relógios)	Deriva = 0.1%	0.01	20	10	0.25
3	Média (2 relógios)	Deriva = 0.1%	0.01	20	0	0.1
4	Média (2 relógios)	Viés Inicial = 0.1 (s)	0.01	20	20	1
5	Média (2 relógios)	Viés Inicial = 0.1 (s)	0.01	20	10	0.25
6	Média (2 relógios)	Viés Inicial = 0.1 (s)	0.01	20	0	0.1
7	FTM (4 relógios)	Deriva = 0.1%	0.01	20	20	1
8	FTM (4 relógios)	Deriva = 0.1%	0.01	20	10	0.25

7.3.1. Caso 1 - Função Exponencial - Média - 2 relógios

Este caso simula um algoritmo genérico de média descrito pela equação (5.8), com ganho (G) unitário, período de re-sincronização (R) de 20 *ticks*, chaveamento (M) de 20 *ticks* e com o relógio 1 e 3 com 0.1% de deriva. Com o

chaveamento em 20, significa que o valor da função "*cross-fading*" chega a 5% do ganho no instante M.

A Figura 7.55 apresenta os estados propagados dos relógios com e sem utilizar a função "*cross-fading*". A Figura 7.56 apresenta a precisão dos relógios, isto é diferenças entre relógio 1 e relógio 2; e diferença entre relógio 3 e 4. Por ser um algoritmo que sua métrica é a precisão, nos casos com o algoritmo média não são levados em conta à exatidão (diferença dos relógios para o relógio de referência). A Figura 7.57 mostra o gráfico das correções calculadas e aplicadas aos estados do relógio, com e sem "cross-fading". A Figura 7.58 mostra as funções "cross-fading" dos relógios 1 e 2.

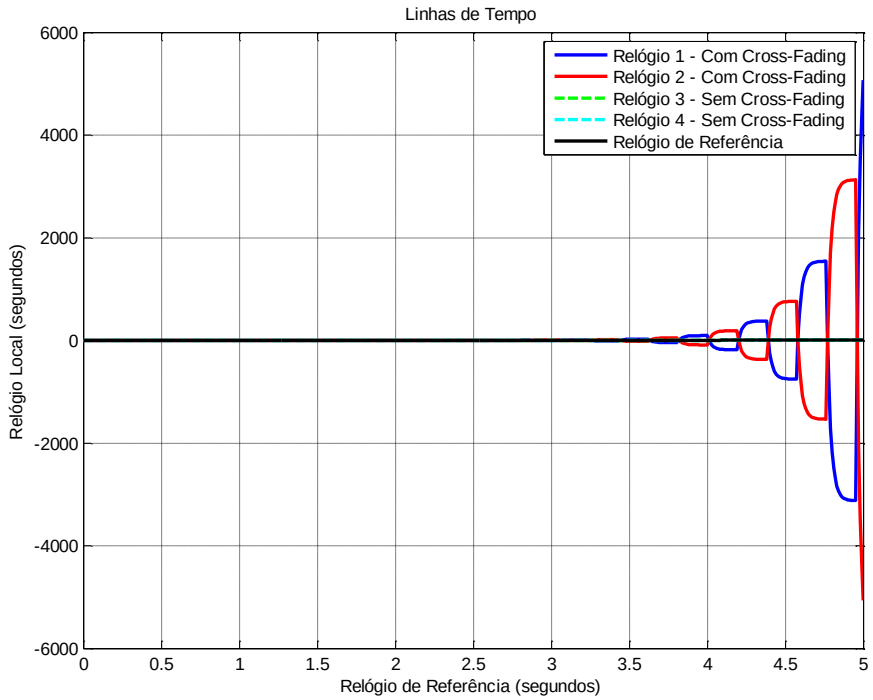


Figura 7.55 – Caso 1 - Cross-Fading Exponencial: Linhas de Tempo.

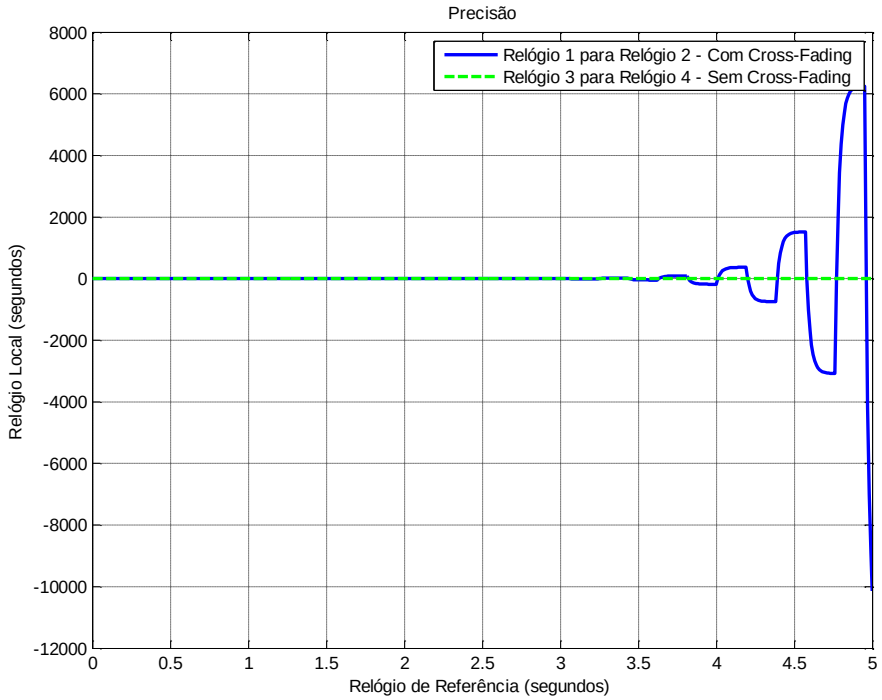


Figura 7.56 – Caso 1 - Cross-Fading Exponencial: Precisão dos relógios.

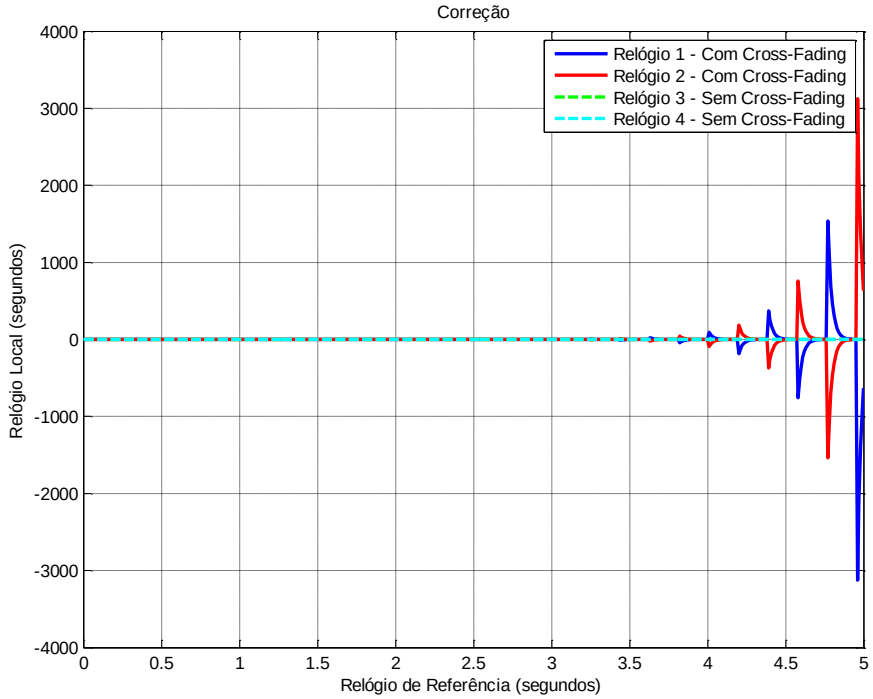


Figura 7.57 – Caso 1 - Cross-Fading Exponencial: Correções aplicadas.

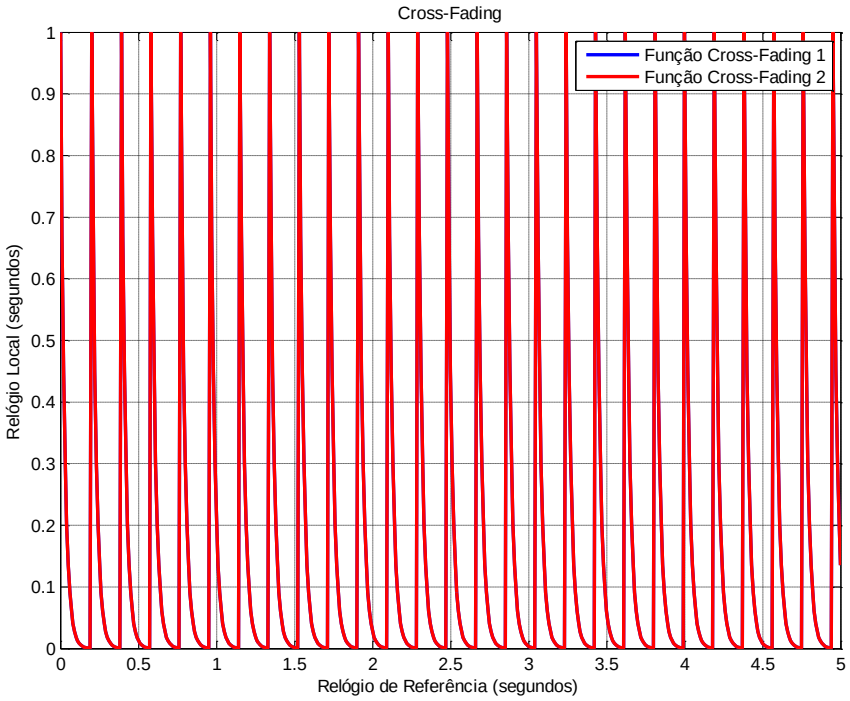


Figura 7.58 – Caso 1: Função *"cross-fading"* exponencial.

É possível observar que os resultados com "*cross-fading*" foram piores. Isto ocorre pois sem o chaveamento intermediário, como no caso da rampa, um ganho muito alto pode causar instabilidades no sistema, como o observado. Isto ocorre pois a exponencial introduz atrasos ao sistema. O objetivo de amortizar a correção ao longo do tempo, neste caso, não foi alcançado.

7.3.2. Caso 2 - Função Exponencial - Média - 2 relógios

Este caso simula um algoritmo genérico de média descrito pela equação (5.8), com ganho (G) 0.25, período de re-sincronização (R) de 20 *ticks*, chaveamento (M) de 10 *ticks* e com o relógio 1 e 3 com 0.1% de deriva. Com o chaveamento em 10, significa que o "*cross-fading*" é uma exponencial que atingira 5% de G após 10 ticks.

A Figura 7.59 apresenta os estados propagados dos relógios com e sem utilizar a função "*cross-fading*". A Figura 7.60 apresenta a precisão dos relógios, isto é diferenças entre relógio 1 e relógio 2; e diferença entre relógio 3 e 4. A Figura

7.61 mostra o gráfico das correções calculadas e aplicadas aos estados do relógio, com e sem "*cross-fading*". A Figura 7.62 mostra as funções "*cross-fading*" dos relógios 1 e 2.

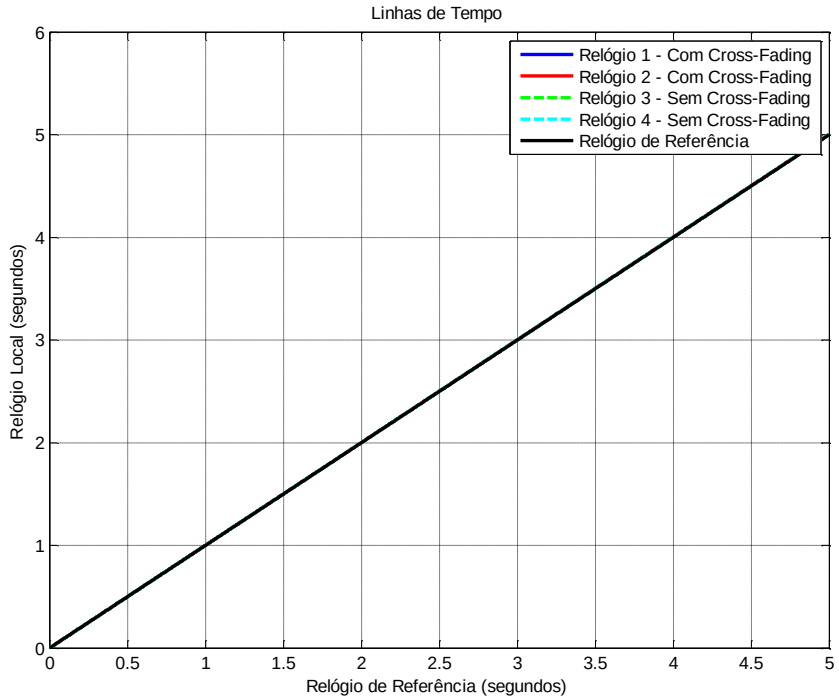


Figura 7.59 – Caso 2 - *Cross-Fading* Exponencial: Linhas de Tempo.

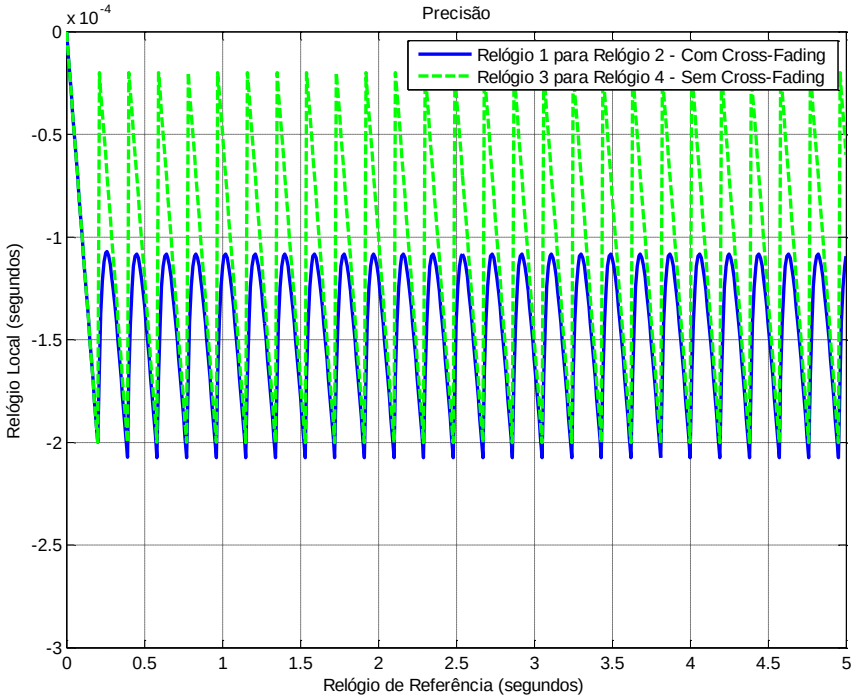


Figura 7.60 – Caso 2 - *Cross-Fading* Exponencial: Precisão dos relógios.

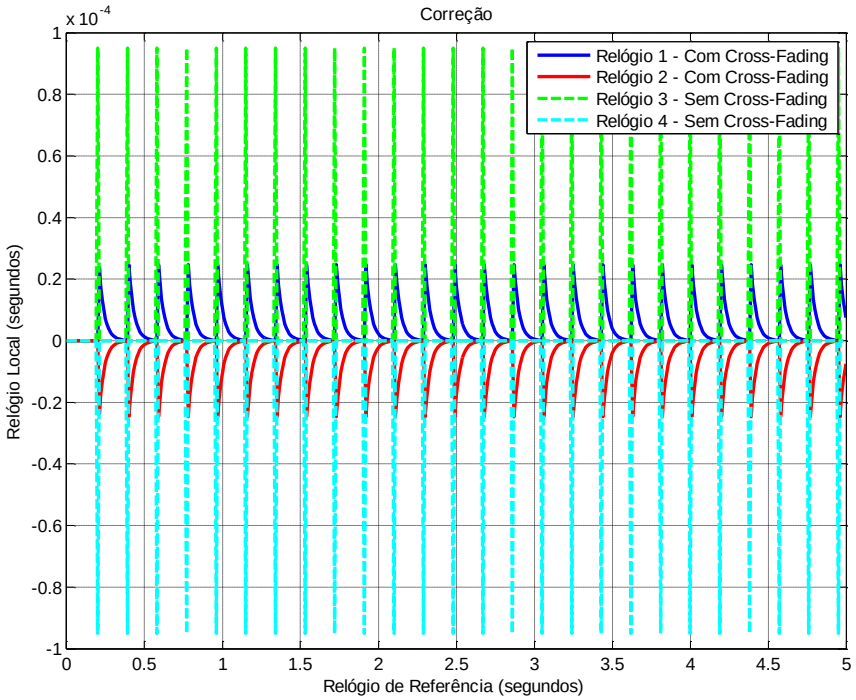


Figura 7.61 – Caso 2 - *Cross-Fading* Exponencial: Correções aplicadas.

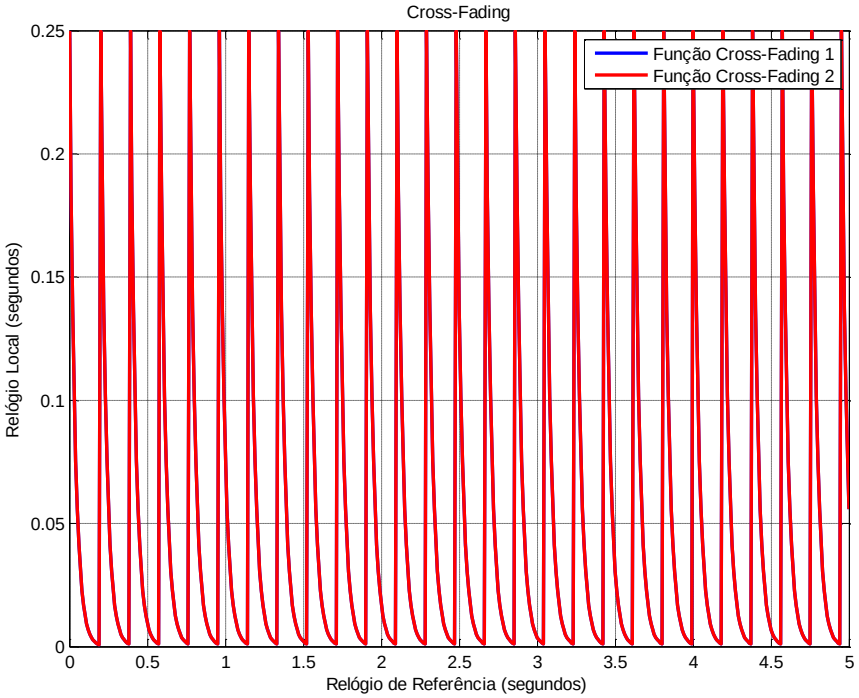


Figura 7.62 – Caso 2: Função *"cross-fading"* exponencial.

Neste caso, é possível observar que os relógios com "*cross-fading*" obtiveram uma melhora em sua precisão em comparação com os relógios sem "cross-fading". Mas, a média do conjunto foi um pouco alterada, o que pode prejudicar

a exatidão. O objetivo de amortizar a correção ao longo do tempo foi alcançado, além de melhorar a precisão dos relógios.

7.3.3. Caso 3 - Função Exponencial - Média - 2 relógios

Este caso simula um algoritmo genérico de média descrito pela equação (5.8), com ganho (G) 0.1, período de re-sincronização (R) de 20 *ticks*, chaveamento (M) de 0 *ticks* e com o relógio 1 e 3 com 0.1% de deriva. Com o chaveamento em 0, significa que o "*cross-fading*" que é uma exponencial, atinge 5% de G no instante da próxima re-sincronização.

A Figura 7.63 apresenta os estados propagados dos relógios com e sem utilizar a função "*cross-fading*". A Figura 7.64 apresenta a precisão dos relógios, isto é diferenças entre relógio 1 e relógio 2; e diferença entre relógio 3 e 4. A Figura 7.65 mostra o gráfico das correções calculadas e aplicadas aos estados do relógio, com e sem "*cross-fading*". A Figura 7.66 mostra as funções "*cross-fading*" dos relógios 1 e 2.

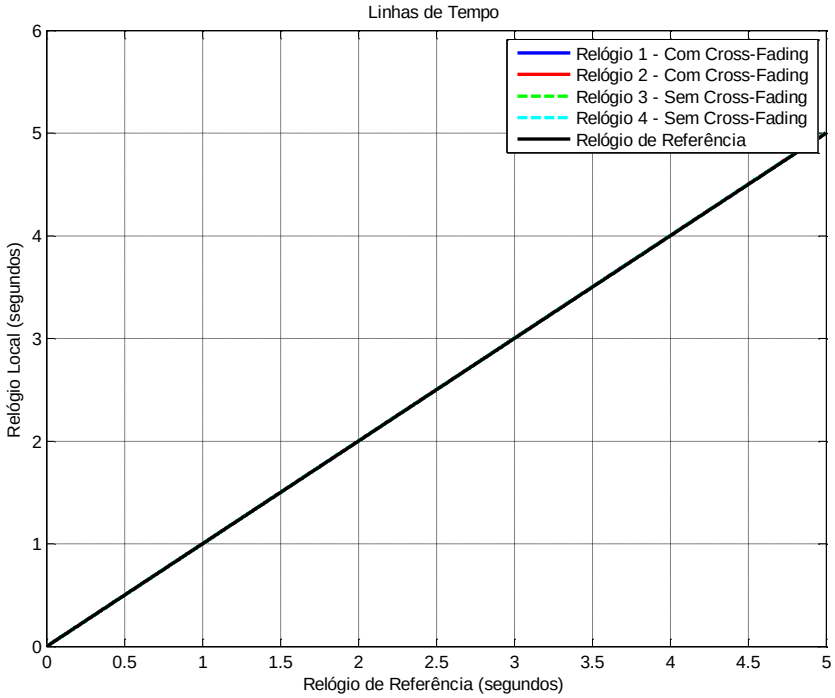


Figura 7.63 – Caso 3 - *Cross-Fading* Exponencial: Linhas de Tempo.

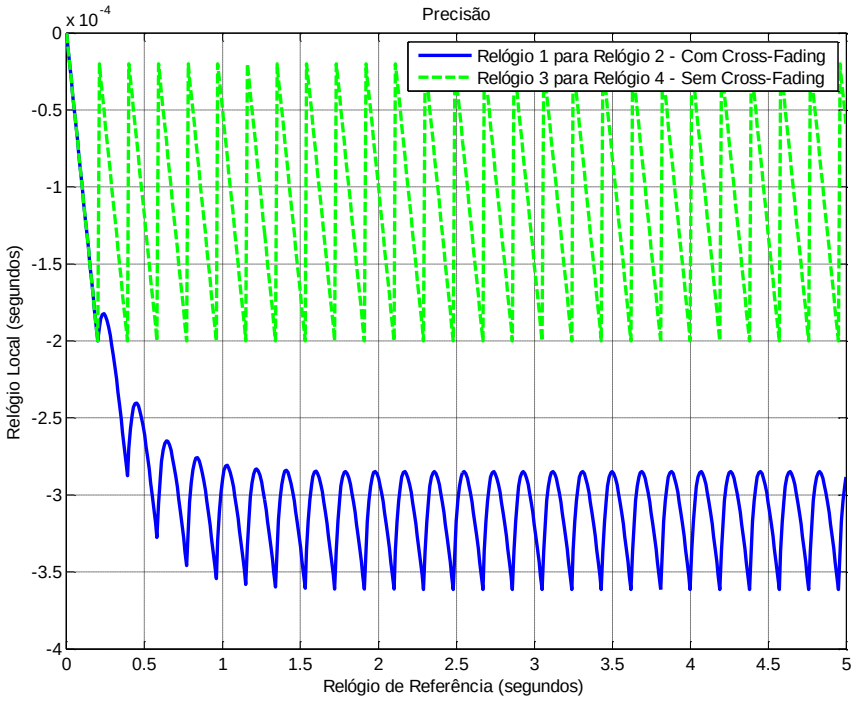


Figura 7.64 – Caso 3 - *Cross-Fading* Exponencial: Precisão dos relógios.

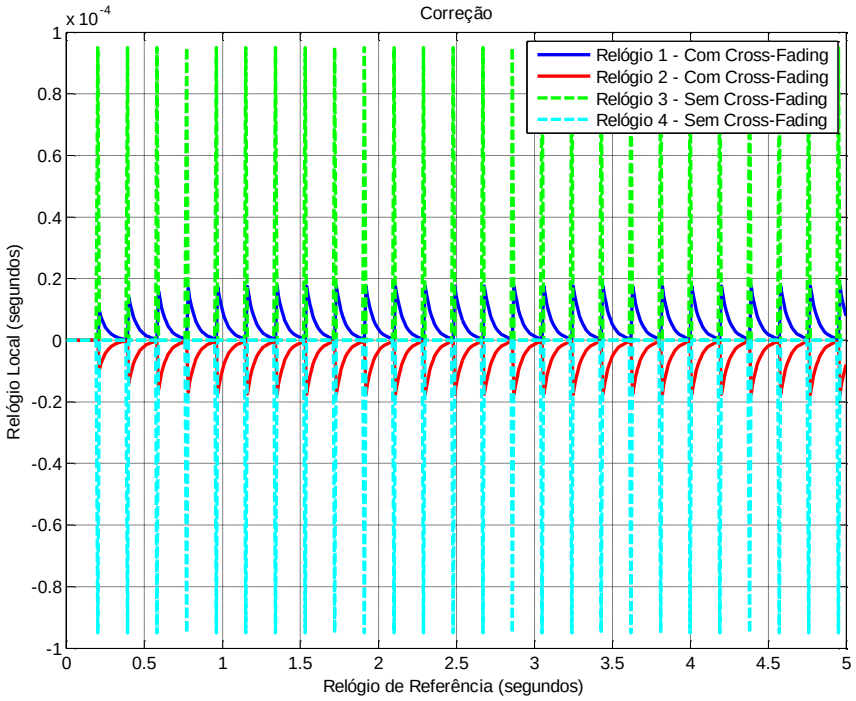


Figura 7.65 – Caso 3 - *Cross-Fading* Exponencial: Correções aplicadas.

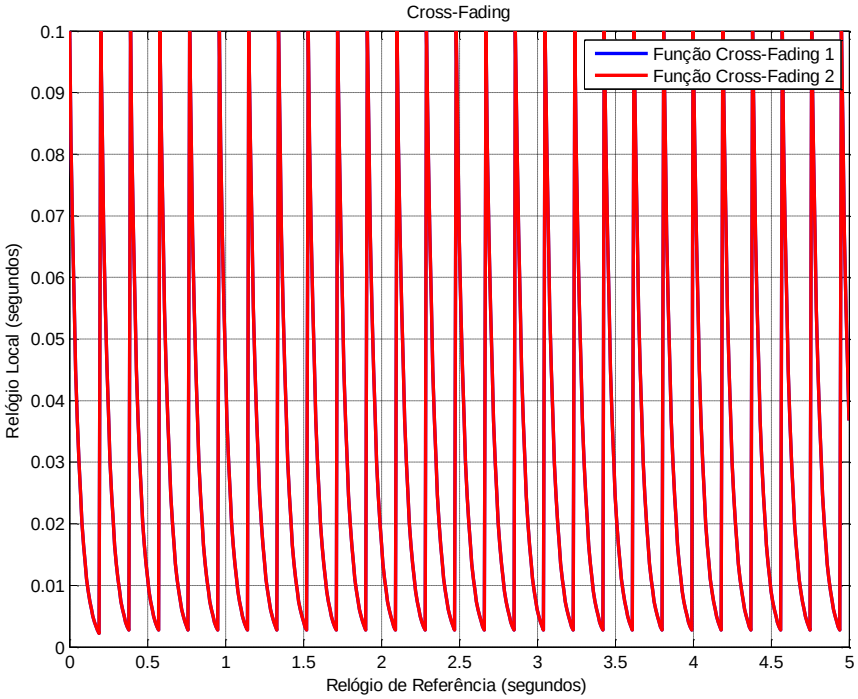


Figura 7.66 – Caso 3: Função *"cross-fading"* exponencial.

Neste caso, é possível observar que os relógios com "*cross-fading*" também tiveram uma melhora em sua precisão em comparação com os relógios sem "*cross-fading*". Mas, é possível observar que a média do conjunto foi bem alterada, o que pode impactar na exatidão. O objetivo de amortizar a correção ao longo do tempo foi alcançado, além de melhorar a precisão dos relógios.

7.3.4. Caso 4 - Função Exponencial - Média - 2 relógios

Este caso simula um algoritmo genérico de média descrito pela equação (5.8), com ganho (G) 1, período de re-sincronização (R) de 20 *ticks*, chaveamento (M) de 20 *ticks* e com viés inicial de 0.1(segundos) no relógio 1 e 3. Com o chaveamento em 20, significa que o "*cross-fading*" que é uma exponencial, atinge 5% do ganho rapidamente.

A Figura 7.67 apresenta os estados propagados dos relógios com e sem utilizar a função "*cross-fading*". A Figura 7.68 apresenta a precisão dos relógios, isto é diferenças entre relógio 1 e relógio 2; e diferença entre relógio 3 e 4. A Figura 7.69 mostra o gráfico das correções calculadas e aplicadas aos estados do

relógio, com e sem "*cross-fading*". A Figura 7.70 mostra as funções "*cross-fading*" dos relógios 1 e 2.

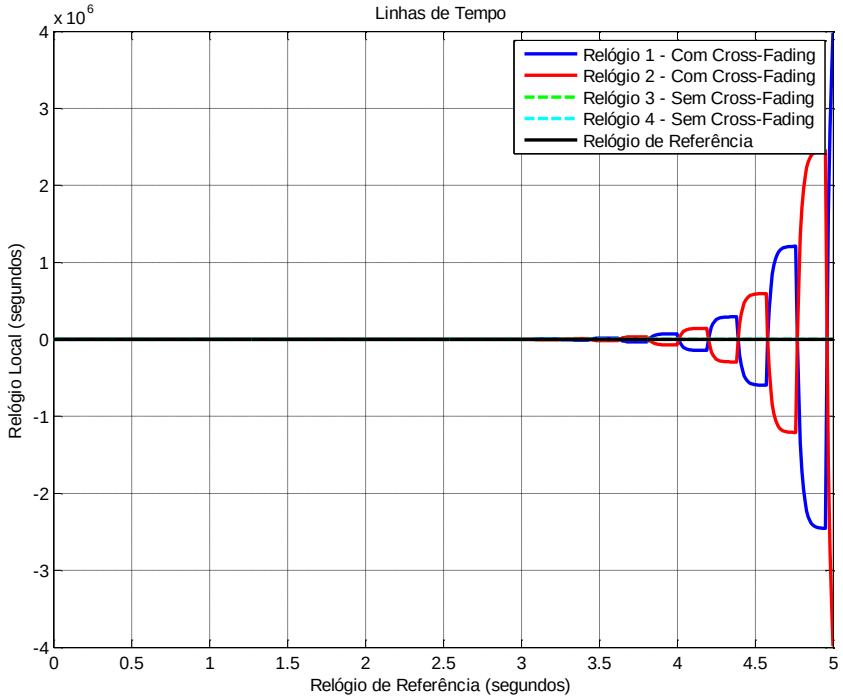


Figura 7.67 – Caso 4 - *Cross-Fading* Exponencial: Linhas de Tempo.

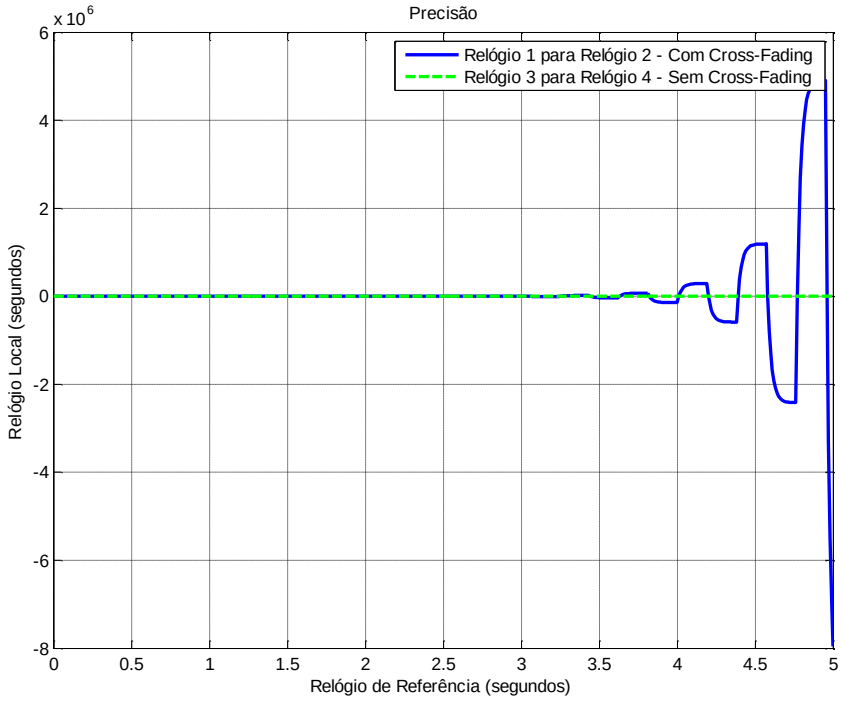


Figura 7.68 – Caso 4 - *Cross-Fading* Exponencial: Precisão dos relógios.

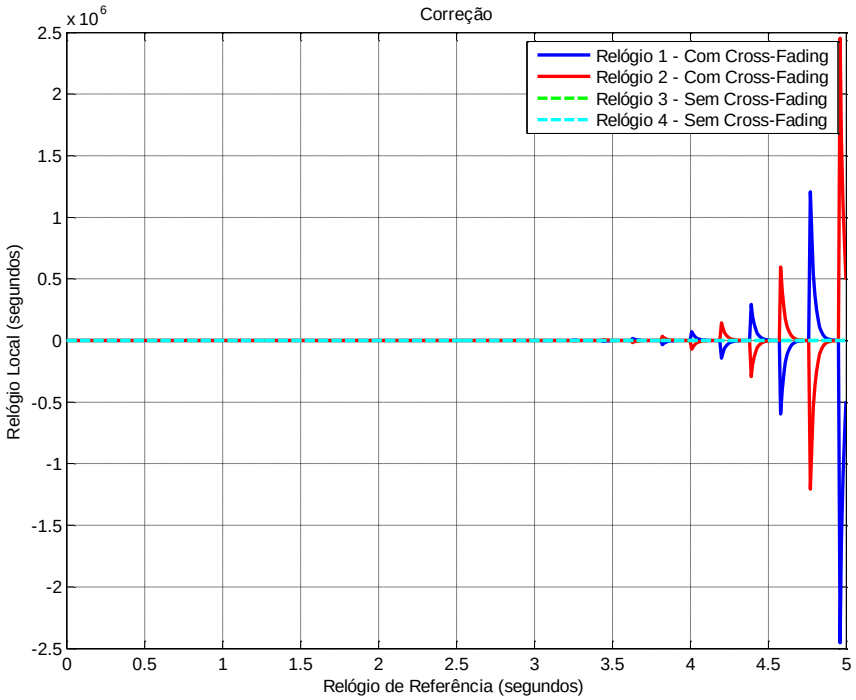


Figura 7.69 – Caso 4 - *Cross-Fading* Exponencial: Correções aplicadas.

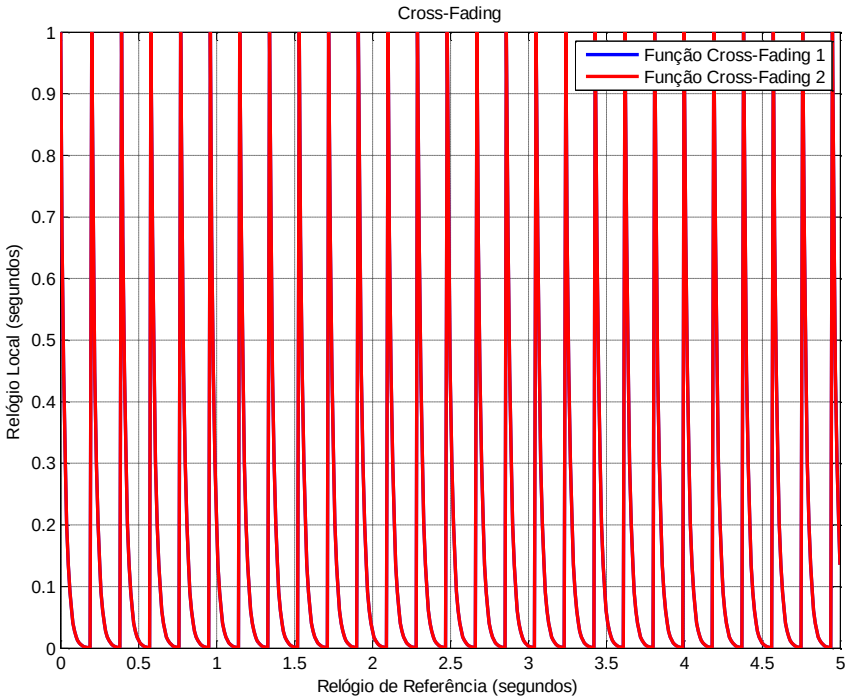


Figura 7.70 – Caso 4 - Função *"cross-fading"* exponencial.

É possível observar que os resultados com "*cross-fading*" foram piores. Isto ocorre pois sem o chaveamento intermediário, como no caso da rampa, um ganho muito alto pode causar instabilidades no sistema, como o observado.

Isto ocorre pois a exponencial introduz atrasos ao sistema. O objetivo de amortizar a correção ao longo do tempo, neste caso, não foi alcançado.

7.3.5. Caso 5 - Função Exponencial - Média - 2 relógios

Este caso simula um algoritmo genérico de média descrito pela equação (5.8), com ganho (G) 0.25, período de re-sincronização (R) de 20 *ticks*, chaveamento (M) de 10 *ticks* e com viés inicial de 0.1(segundos) no relógio 1 e 3. Com o chaveamento em 10, significa que o "*cross-fading*" que é uma exponencial, atinge 5% do ganho após 10 ticks.

A Figura 7.71 apresenta os estados propagados dos relógios com e sem utilizar a função "*cross-fading*". A Figura 7.72 apresenta a precisão dos relógios, isto é diferenças entre relógio 1 e relógio 2; e diferença entre relógio 3 e 4. A Figura 7.73 mostra o gráfico das correções calculadas e aplicadas aos estados do relógio, com e sem "*cross-fading*". A Figura 7.74 mostra as funções "*cross-fading*" dos relógios 1 e 2.

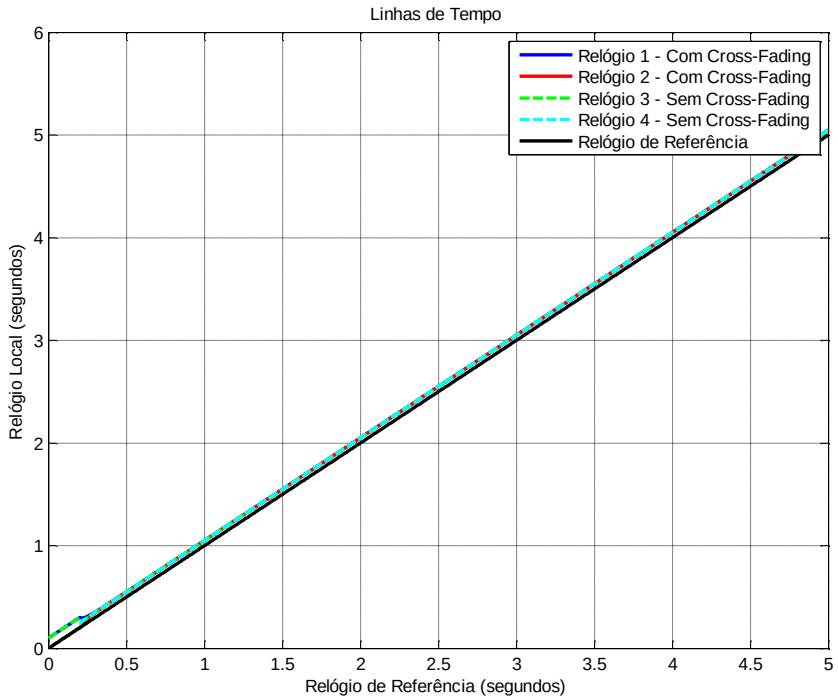


Figura 7.71 – Caso 5 - *Cross-Fading* Exponencial: Linhas de Tempo.

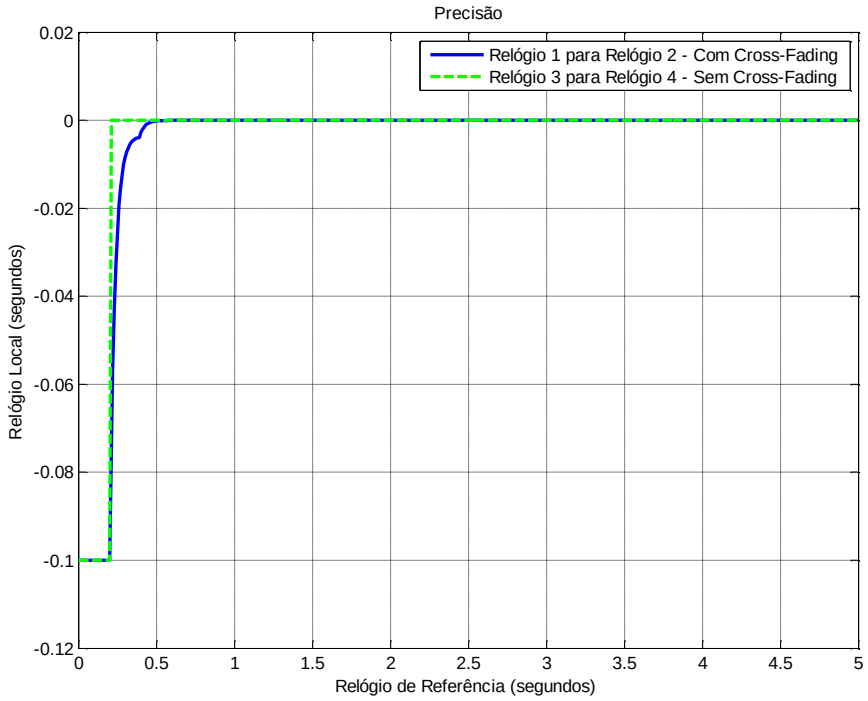


Figura 7.72 – Caso 5 - *Cross-Fading* Exponencial: Precisão dos relógios.

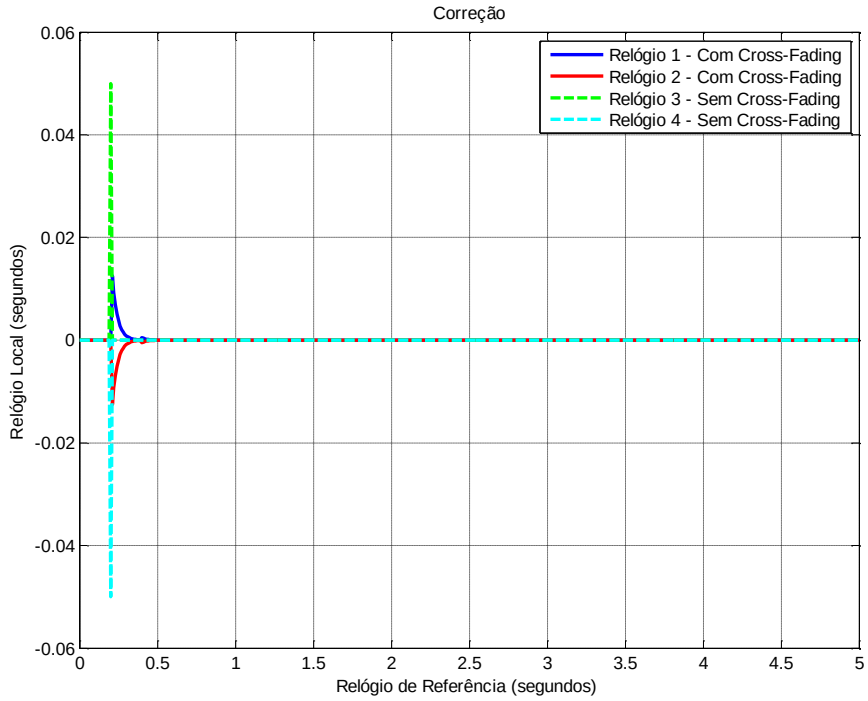


Figura 7.73 – Caso 5 - *Cross-Fading* Exponencial: Correções aplicadas.

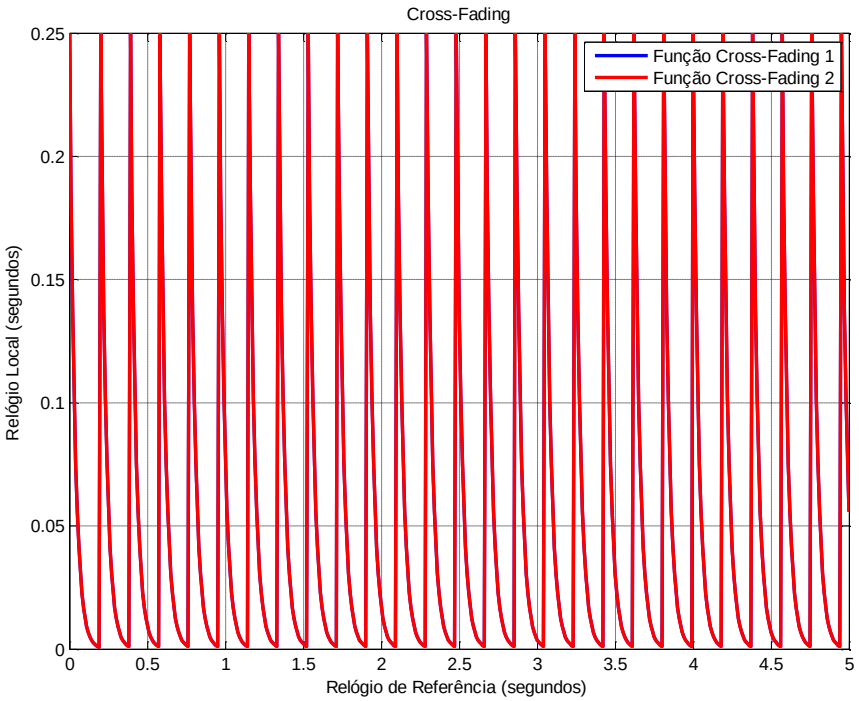


Figura 7.74 – Caso 5 - Função *"cross-fading" e*xponencial.

Neste caso, é possível observar que os relógios com "*cross-fading*" amorteceram o impacto inicial devido ao viés inicial. Na Figura 7.72 é possível observar que o "cross-fading" não causa um "*overshoot*". O objetivo de amortizar a correção ao longo do tempo foi alcançado com um resultado muito bom, pois o "overshoot" é eliminado, o algoritmo têm uma convergência rápida, no entanto, com um tempo maior devido ao atraso inserido pela função exponencial. O resultado com "*cross-fading*" em comparação com o resultado sem "*cross-fading*" foi melhor.

7.3.6. Caso 6 - Função Exponencial - Média - 2 relógios

Este caso simula um algoritmo genérico de média descrito pela equação (5.8), com ganho (G) 0.1, período de re-sincronização (R) de 20 *ticks*, chaveamento (M) de 0 *ticks* e com viés inicial de 0.1(segundos) no relógio 1 e 3. Com o chaveamento em 0, significa que o "*cross-fading*" que é uma exponencial atinge 5% do ganho no próximo intervalo de re-sincronização.

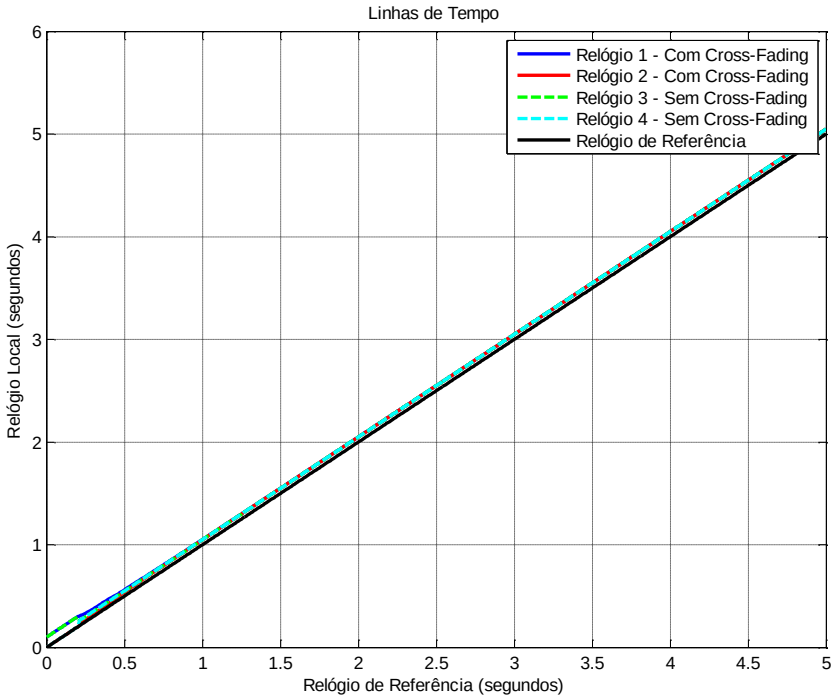


Figura 7.75 – Caso 6 - *Cross-Fading* Exponencial: Linhas de Tempo.

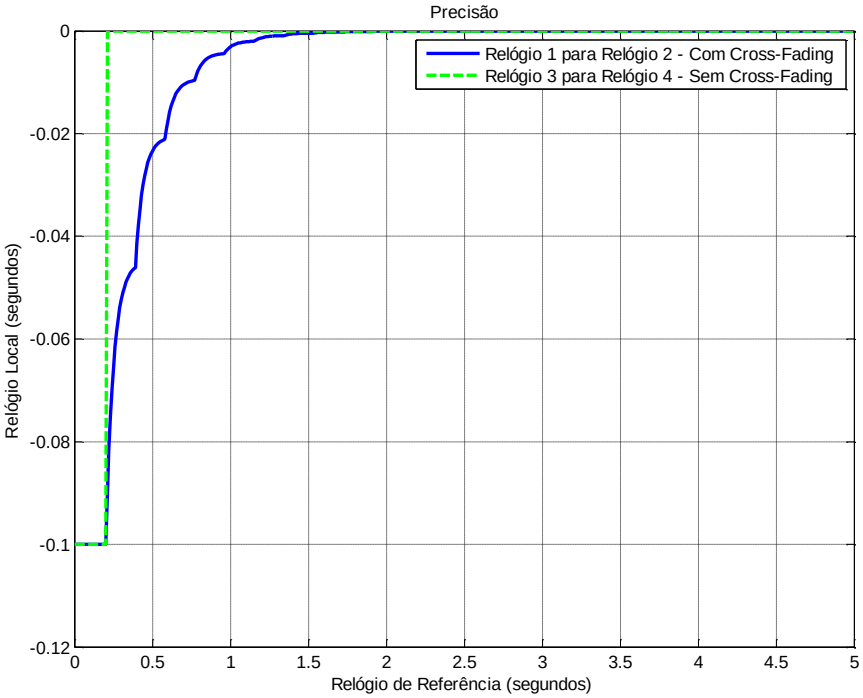


Figura 7.76 – Caso 6 - *Cross-Fading* Exponencial: Precisão dos relógios.

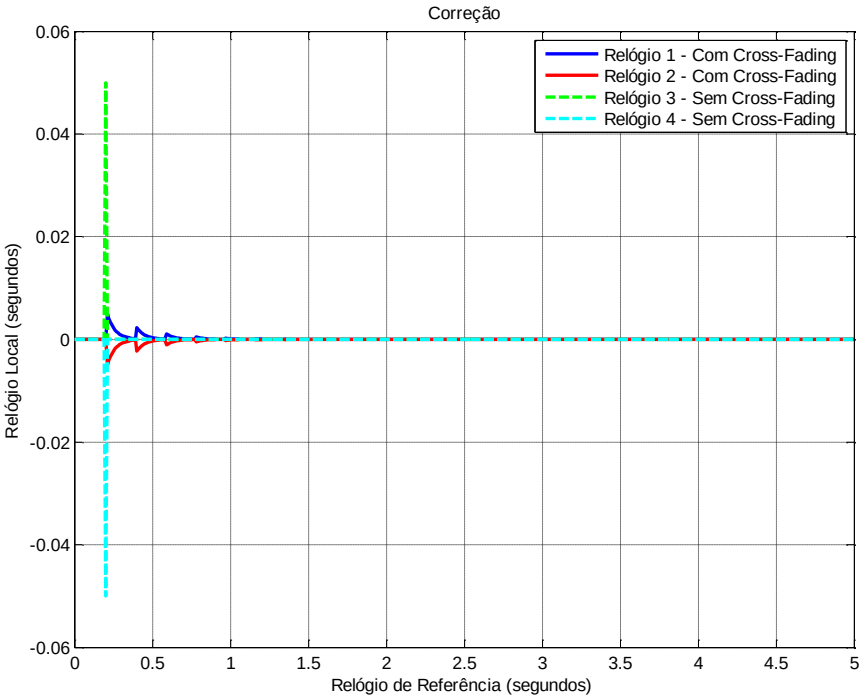


Figura 7.77 – Caso 6 - *Cross-Fading* Exponencial: Correções aplicadas.

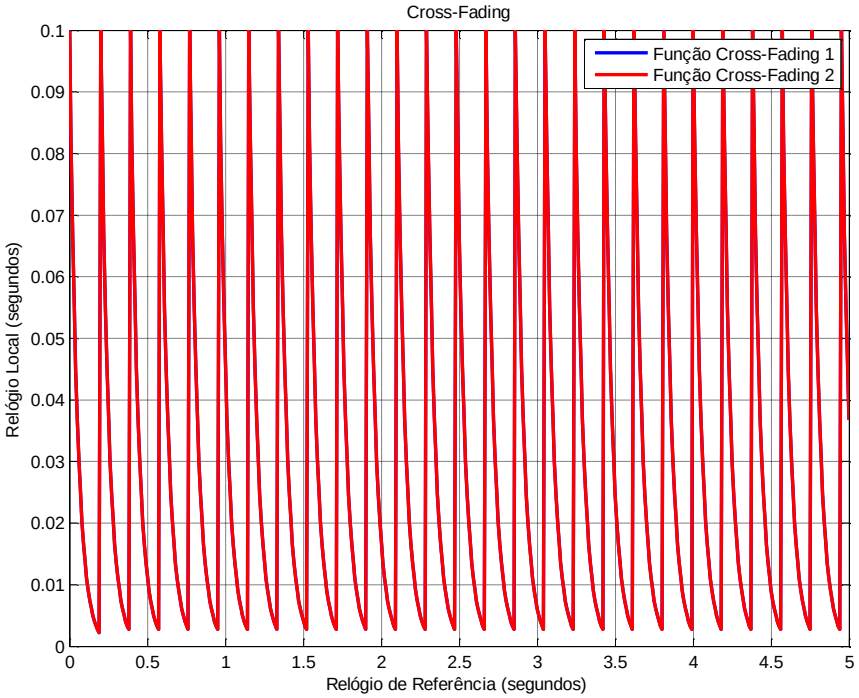


Figura 7.78 – Caso 6 - Função *"cross-fading"* exponencial.

A Figura 7.75 apresenta os estados propagados dos relógios com e sem utilizar a função "*cross-fading*". A Figura 7.76 apresenta a precisão dos relógios, isto é diferenças entre relógio 1 e relógio 2; e diferença entre relógio 3 e 4. A Figura

7.77 mostra o gráfico das correções calculadas e aplicadas aos estados do relógio, com e sem "*cross-fading*". A Figura 7.78 mostra as funções "*cross-fading*" dos relógios 1 e 2.

Neste caso, é possível observar que os relógios com "*cross-fading*" amorteceram o impacto inicial devido ao viés inicial. Na Figura 7.76 é possível observar que o "cross-fading" não causou "*overshoot*". O objetivo de amortizar a correção ao longo do tempo foi alcançado com um resultado muito bom, no entanto, o atraso devido a exponencial foi maior. O resultado com "*cross-fading*" em comparação com o resultado sem "*cross-fading*" foi melhor.

Em uma rápida análise dos casos 5 e 6, pode-se concluir que ajustes nos parâmetros do "*cross-fading*" podem melhorar e/ou aproximar os resultados em comparação ao caso sem "*cross-fading*".

7.3.7. Caso 7 - Função Exponencial - FTM - 4 relógios

Este caso simula o algoritmo FTM, por métrica de precisão, com ganho (G) 1, período de re-sincronização (R) de 20 *ticks*, chaveamento (M) de 20 *ticks* e com deriva de 0.1% no relógio 1. O chaveamento em 20, significa que o "*cross-fading*" que é uma exponencial, atinge 5% do ganho rapidamente.

A Figura 7.79 apresenta os estados propagados dos relógios com (lado esquerdo) e sem (lado direito) aplicar a função "*cross-fading*". A Figura 7.80 apresenta a precisão dos relógios (com e sem "*cross-fading*"), isto é a diferença entre os relógios do conjunto. A Figura 7.81 mostra o gráfico das correções calculadas e aplicadas aos estados dos relógios, com (lado esquerdo) e sem (lado direito) "*cross-fading*". A Figura 7.82 mostra as funções "*cross-fading*" dos relógios.

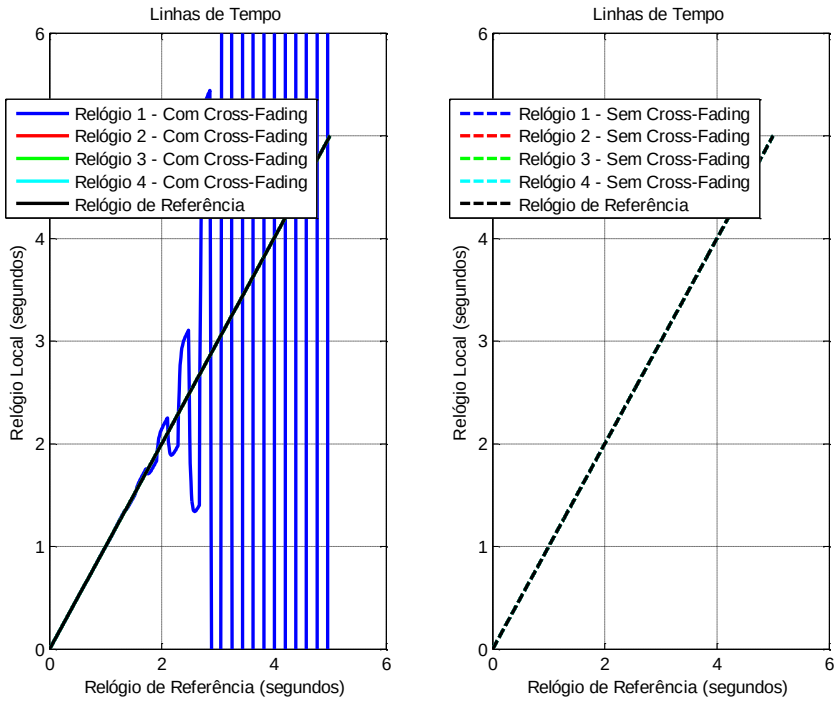


Figura 7.79 – Caso 7 - *Cross-Fading* Exponencial: Linhas de Tempo.

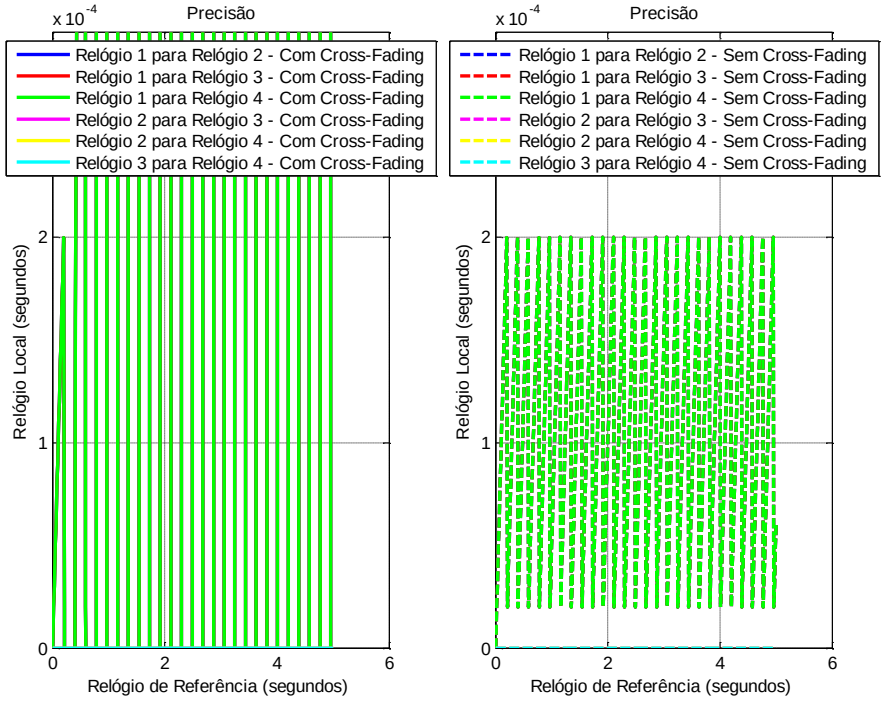


Figura 7.80 – Caso 7 - *Cross-Fading* Exponencial: Precisão dos relógios.

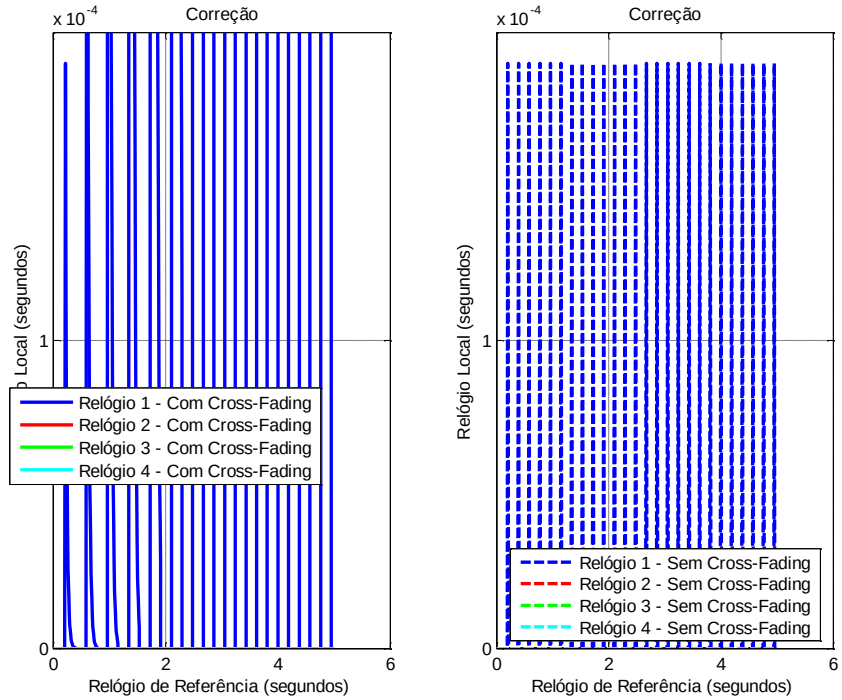


Figura 7.81 – Caso 7 - *Cross-Fading* Exponencial: Correções aplicadas.

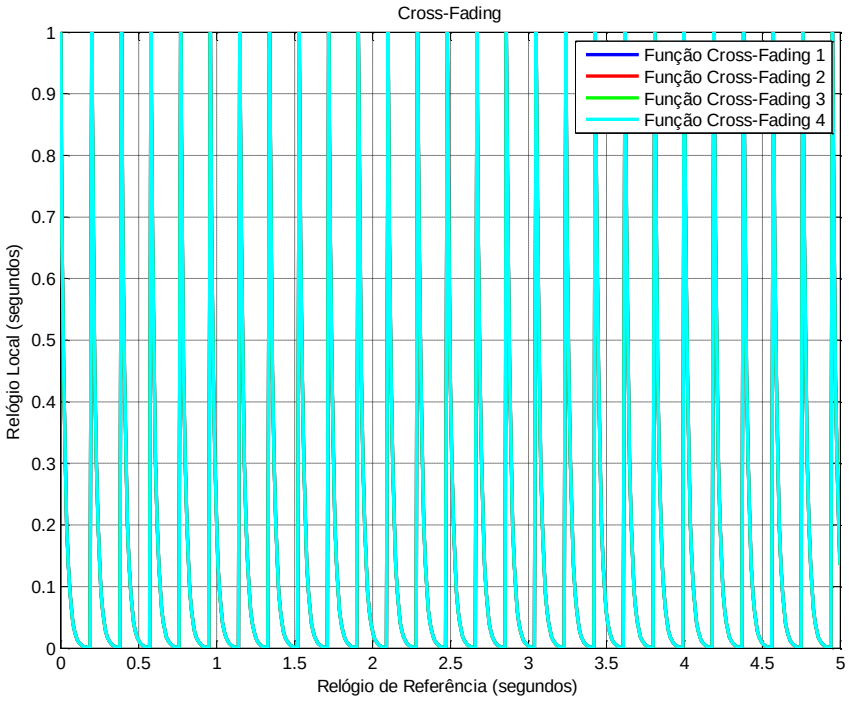


Figura 7.82 – Caso 7 - Função *"cross-fading" e*xponencial.

É possível observar que os resultados com "cross-fading" foram piores em relação ao caso sem "cross-fading", chegando a causar instabilidade sobre o relógio 1. Isto ocorre pois sem o chaveamento intermediário, como no caso da

rampa, um ganho muito alto pode causar instabilidades no sistema pois a exponencial introduz atraso ao sistema e com isso diminuindo a margem de fase. No entanto, somente o relógio 1 ficou instável. Devido à característica de tolerância a falhas bizantinas do algoritmo FTM, a instabilidade do relógio 1 não se propagou aos outros relógios. No entanto, mesmo assim a precisão dos relógios foi afetada. O objetivo de amortizar a correção ao longo do tempo, neste caso, não foi alcançado no relógio 1, com falha.

7.3.8. Caso 8 - Função Exponencial - FTM - 4 relógios

Este caso simula o algoritmo FTM, por métrica de precisão, com ganho (G) 0.25, período de re-sincronização (R) de 20 *ticks*, chaveamento (M) de 10 *ticks* e com deriva de 0.1% no relógio 1. O chaveamento em 10, significa que o "*cross-fading*" que é uma exponencial, alcança 5% do ganho após 10 ticks.

A Figura 7.83 apresenta os estados propagados dos relógios com (lado esquerdo) e sem (lado direito) aplicar a função "*cross-fading*". A Figura 7.84 apresenta a precisão dos relógios (com e sem "*cross-fading*"), isto é a diferença entre os relógios do conjunto. A Figura 7.85 mostra o gráfico das correções calculadas e aplicadas aos estados dos relógios, com (lado esquerdo) e sem (lado direito) "*cross-fading*". A Figura 7.86 mostra as funções "*cross-fading*" dos relógios.

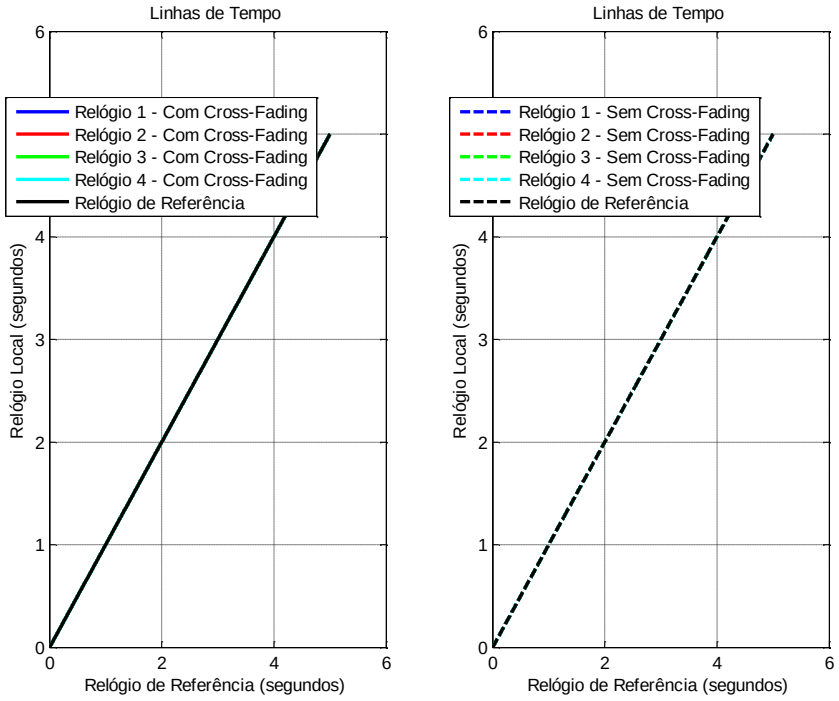


Figura 7.83 – Caso 8 - *Cross-Fading* Exponencial: Linhas de Tempo.

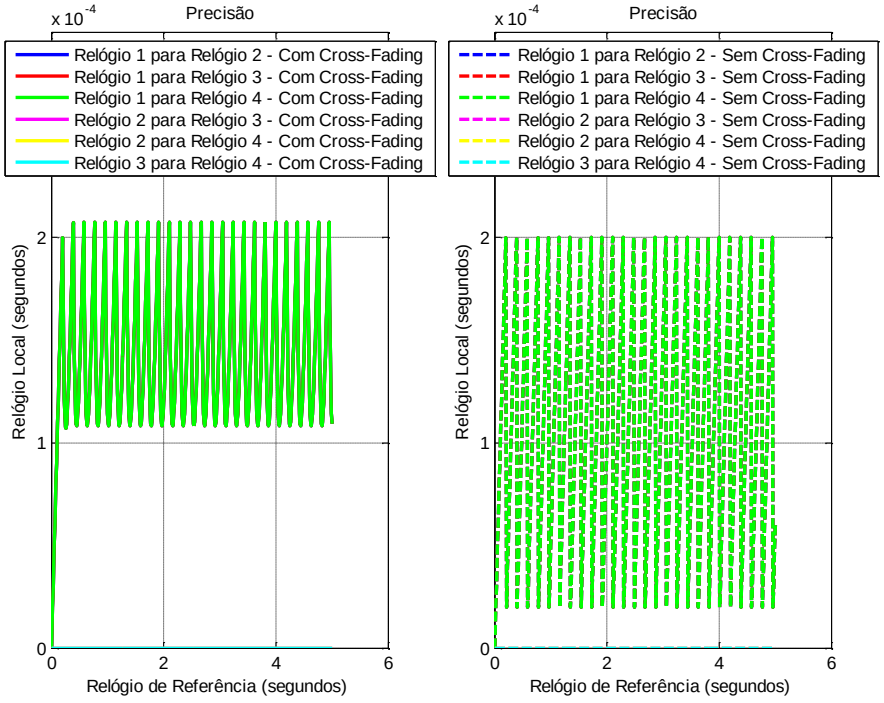


Figura 7.84 – Caso 8 - *Cross-Fading* Exponencial: Precisão dos relógios.

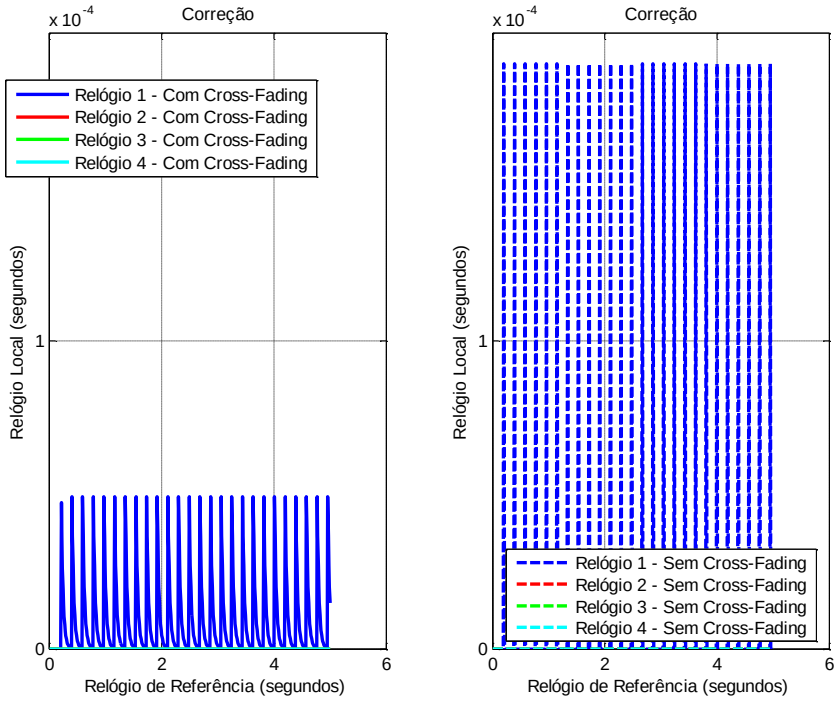


Figura 7.85 – Caso 8 - *Cross-Fading* Exponencial: Correções aplicadas.

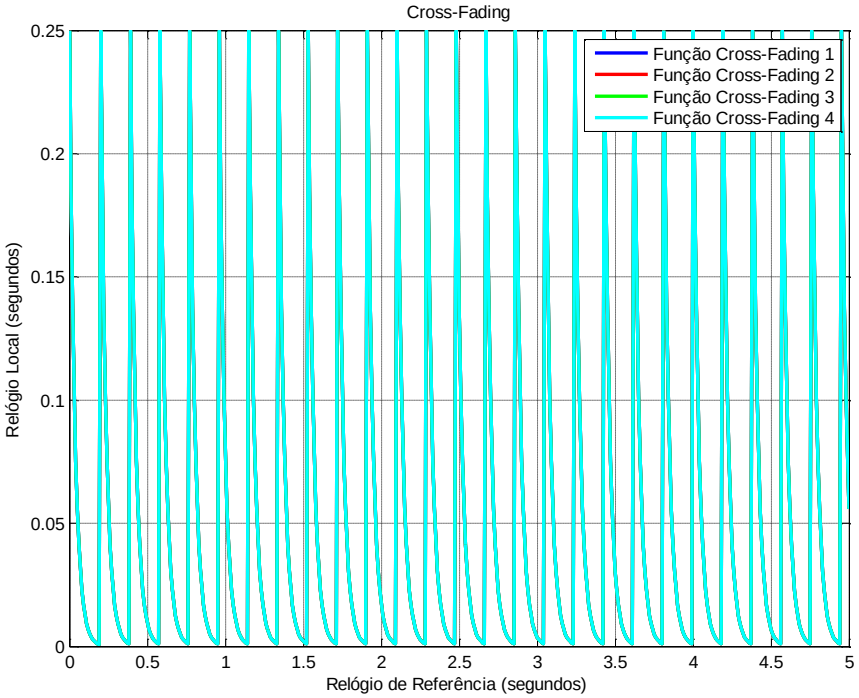


Figura 7.86 – Caso 8 - Função *"cross-fading"* exponencial.

Neste caso, é possível observar que os relógios com "*cross-fading*" obtiveram uma melhora em sua precisão em comparação com os relógios sem "cross-fading". Mas, a média do conjunto foi um pouco alterada, o que pode

prejudicar a exatidão. O objetivo de amortizar a correção ao longo do tempo foi alcançado, além de melhorar a precisão do conjunto. O resultado, neste caso foi excelente.

7.4. Análise dos Resultados da Amortização

Diversos casos foram simulados para as duas funções de amortização, rampa e exponencial. A Tabela 7.4 mostra uma comparação dos resultados de cada caso.

Tabela 7.4 - Comparação dos Casos de Estudos de Amortização.

Parâmetros Função *Cross-Fading*				**Imperfeição**	**Melhor Função**	**Melhor Algoritmo**	**Desempenho**
T	**R**	**M**	**G**				
0.01	20	20	1	Deriva	Rampa	FTM	Caso rampa: Igual ao caso sem amortização Caso Exponencial: Instabilidades no relógio falhado
0.01	20	10	0.25	Deriva	Rampa e Exponencial	FTM	Caso rampa: Melhor com amortização Caso Exponencial: Melhor com amortização; não necessita chaveamento
0.01	20	0	0.1	Deriva	Rampa e Exponencial	Média	Caso rampa: Melhor com amortização Caso Exponencial: Melhor com amortização; não necessita chaveamento
0.01	20	20	1	Viés Inicial	Rampa	Média	Caso rampa: Igual ao caso sem amortização Caso Exponencial: Instabilidades no relógio falhado
0.01	20	10	0.25	Viés Inicial	Exponencial	Média	Caso Exponencial: Melhor com amortização; menor "overshoot"; não necessita chaveamento
0.01	20	0	0.1	Viés Inicial	Exponencial	Média	Caso Exponencial: Melhor com amortização; menor "*overshoot*"; não necessita chaveamento

É possível observar que as amortizações tiveram resultados excelentes, tanto exponencial quanto a rampa, sobre os algoritmos de sincronização de relógios que utilizam amortizações em degrau e/ou que não utilizam amortizações.

Observa-se que se os parâmetros da rampa forem ajustados de tal maneira que a mesma se torne um degrau unitário, este caso se iguala aos algoritmos de sincronização de relógios que não se utilizam de técnicas de amortização. Isto demonstra a generalidade da técnica. Mais além, com ajustes de parâmetros é possível chegar a resultados melhores da sincronização de relógios (precisão e exatidão) com menor impacto sobre o sistema de controle.

Observa-se também que a exponencial introduz atraso ao sistema o que consome a margem de fase podendo levar o sistema (ou relógio) a instabilidades dependendo dos parâmetros ajustados.

Melhores resultados podem ser alcançados utilizando da teoria de controle para chegar a parâmetros que tenham um desempenho ótimo ou robusto.

8 VERIFICAÇÃO E VALIDAÇÃO DO ALGORITMO DE SINCRONIZAÇÃO DE RELÓGIOS

8.1. Projeto da Simulação

Para verificar e validar o algoritmo de sincronização proposto no capítulo 6 foram realizadas várias simulações de sincronização de relógios sobre sistemas de controle por redes. No entanto, para desenvolver a simulação foi necessário o desenvolvimento de todo um conjunto de regras e mapeamento de tempo das várias entidades e domínios de conhecimento envolvidos na simulação utilizando o Matlab/Simulink em conjunto com o TrueTime.

A simulação desenvolvida contou com, pelo menos, vários domínios do conhecimento, como ilustra a Figura 8.1:

a) Dinâmica: Toda a parte matemática da dinâmica, sensores e atuadores.

b) Comunicação: é a parte que simula a camada física e de enlace de um canal de comunicação, no caso CSMA/CD e TDMA. Este nível possui seu tempo próprio (microtick) e único.

c) Computação: é a parte que simula a computação em tempo real, ou seja, em que define-se todo o agendamento das tarefas necessárias do sistema (tarefa de controle, tarefa de sensoriamento, atuação, comunicação, sincronização e outras). Cada bloco possui seu microtick e macrotick. Aqui é possível colocar as características de imperfeições de relógios. Aqui, portanto é realizado todo o procedimento de sincronização de relógios dos macroticks baseado em troca de mensagens via tarefas e remapeamento do agendador. Para tanto, todas as características temporais das tarefas foram baseadas no relógio macrotick lógico. Com isto, a cada intervalo de

re-sincronização, o agendador local re-contextualiza as características temporais de cada tarefa.

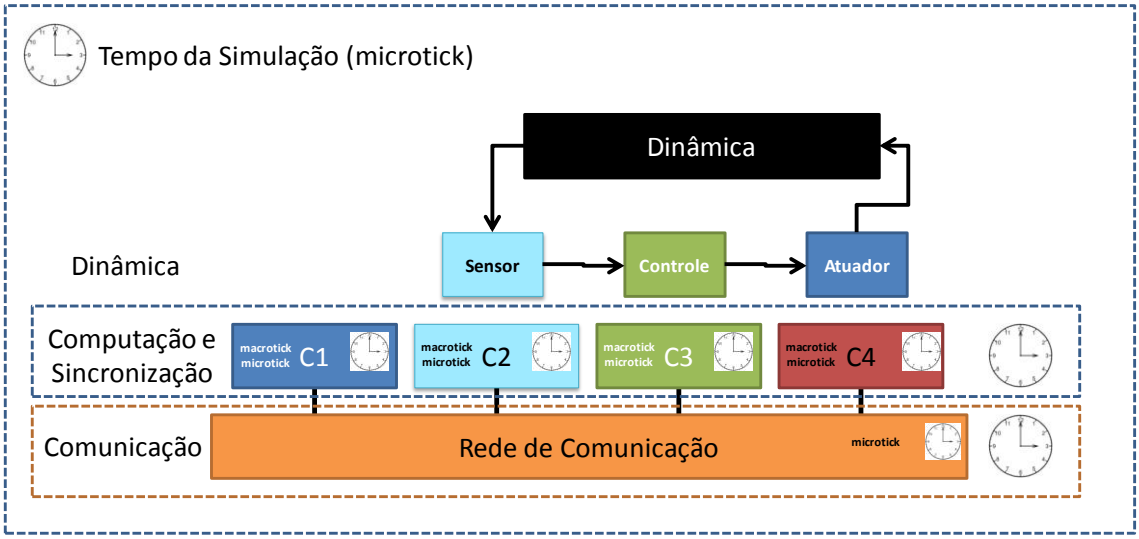


Figura 8.1 – Diagrama em blocos da simulação.

A Figura 8.2 ilustra as influências que as imperfeições temporais (deriva e viés inicial) causam sobre o agendamento computacional. Observa-se que a deriva positiva (relógio acelerado) causa uma diminuição no tempo de processamento da tarefa ao longo do tempo e por consequência no período da tarefa. Já a deriva negativa (relógio mais lento), causa um aumento no tempo de processamento e por consequência no período da tarefa. O viés inicial desloca todo o conjunto para frente ou para trás.

Para realizar a sincronização e visualizar os efeitos sobre a dinâmica, foi necessário desenvolver nesta tese o remapeamento. Ou seja, a cada mudança temporal que o algoritmo de sincronização de relógios causa ao sistema, o efeito sobre o agendamento computacional deve ser refletido. Com isso, todas as vezes que há uma correção temporal de viés instantâneo, existe uma correção temporal do período da tarefa. A Figura 8.3 ilustra este procedimento.

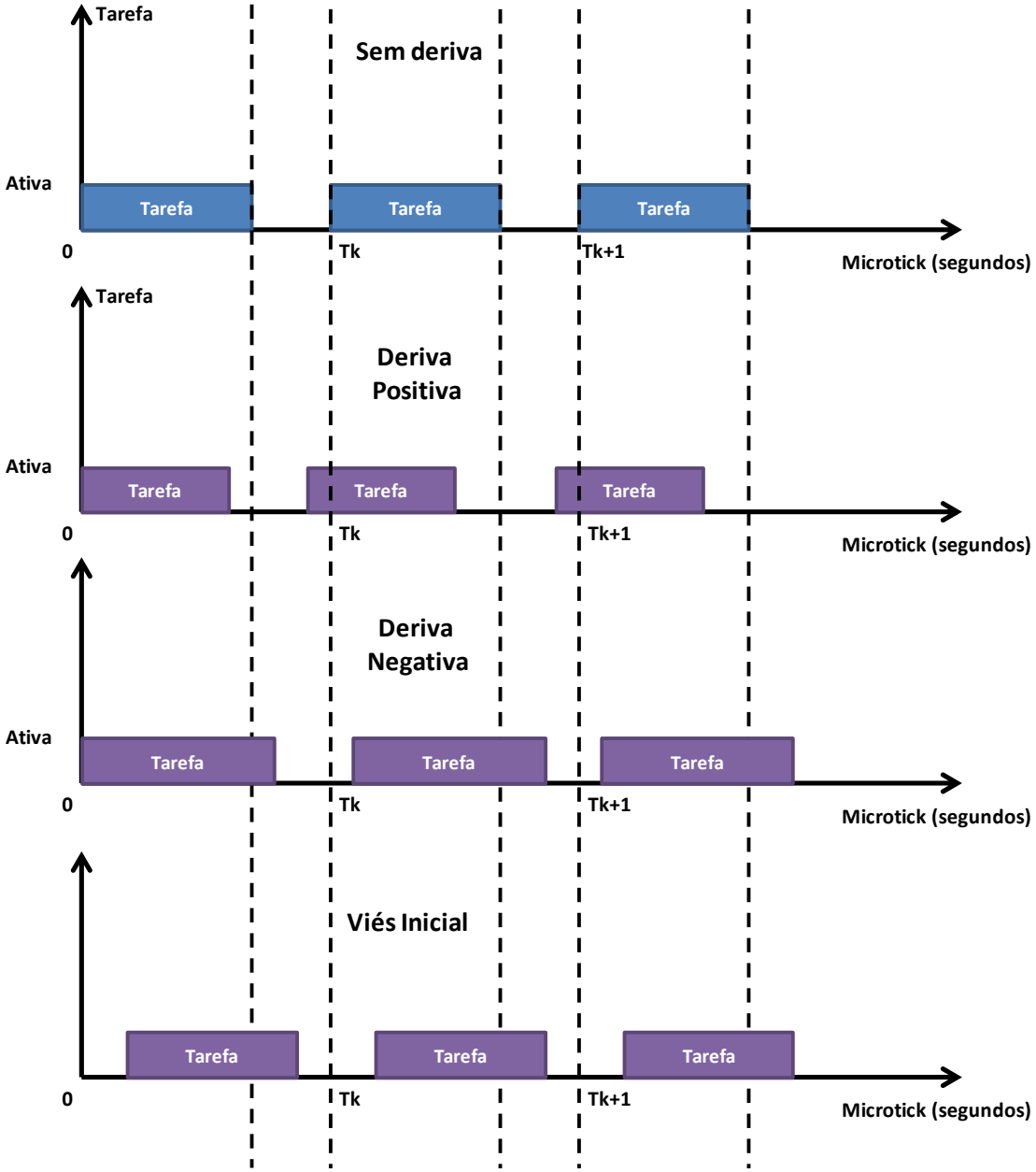


Figura 8.2 – Influência do tempo no agendamento computacional.

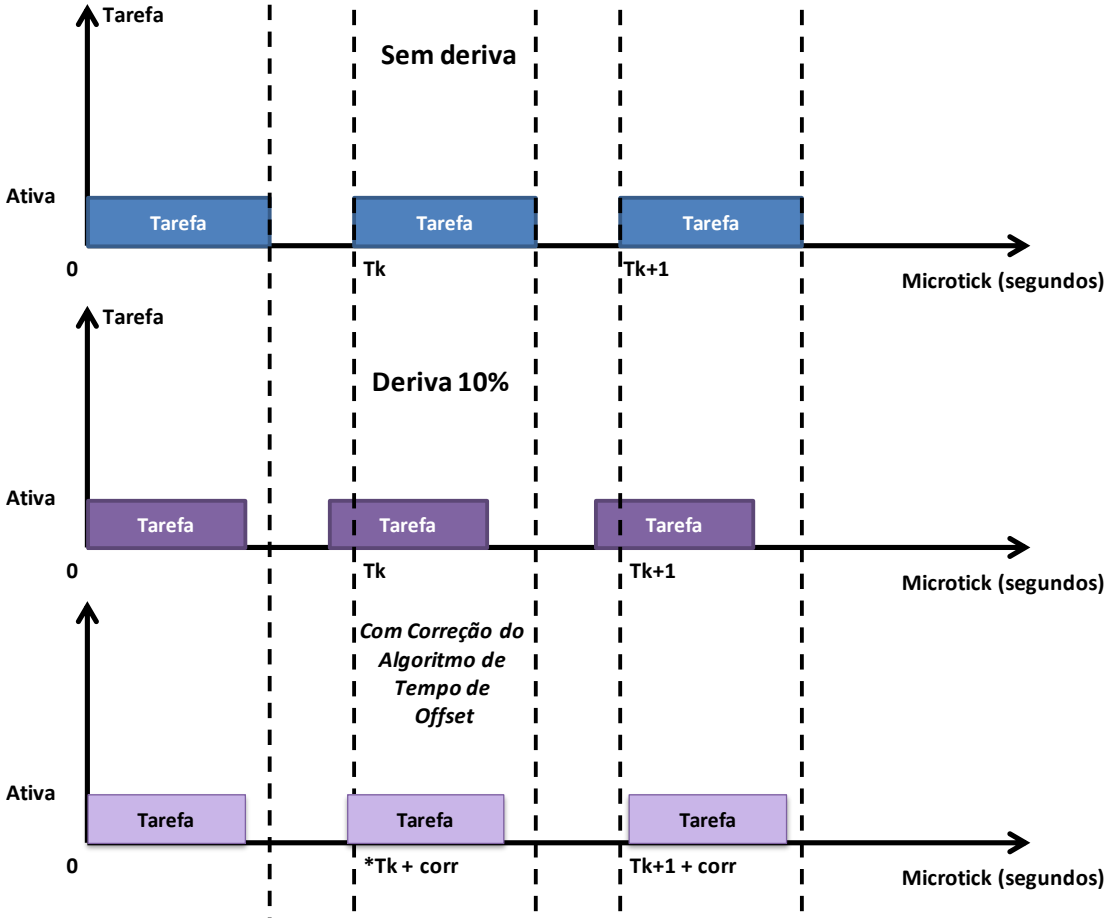


Figura 8.3 – Influência da correção do tempo no agendamento computacional.

Na sequência, serão apresentados os casos simulados, ilustrando os efeitos sobre a sincronização e a dinâmica utilizando os algoritmos PTP, FTM e SR e também com os algoritmos ReS propostos por este trabalho.

8.2. Resumo dos Casos

Os casos a serem estudados estão descritos na Tabela 8.1. Vários casos podem ser desenvolvidos ou estudados. Os casos de estudo ilustram casos de estudo feitos com cada algoritmo e utilizando os métodos propostos pela capítulo 6.

O modo STM, quando aplicável, foi simulado utilizando o algoritmo com *deadbeat* broadcast. O modo NOMO, quando aplicável, foi simulado utilizando as funções e lógicas descritas pela capítulo 6. O modo NOMD, quando aplicável, foi simulado utilizando a equação de convergência descrita pela capítulo 6.

Os casos escolhidos foram divididos em 4 grupos, diferenciados pelas cores (azul, salmão, verde e roxo), além dos casos 1 e 2. Dentro de cada grupo, existem as simulações com e sem rede de comunicação.

O caso 1 é o caso ideal, ou seja, é o caso em que todos os relógios estão sincronizados entre si. O caso 2 é o caso em que os relógios estão inicialmente dessincronizados. Este caso foi escolhido para demonstrar a eficiência do método de sincronização inicial utilizando o controlador *deadbeat*.

O grupo azul são os casos com o algoritmo PTP e o Algoritmo AReS. O primeiro caso do grupo azul é o algoritmo PTP puro, os outros são as variações do algoritmo AReS, em que são variadas as funções *cross-fading* em Função Degrau (CFD), Rampa (CFR) e Exponencial (CFE). As 5 primeiras simulações do grupo azul são realizadas sem uma malha de controle, com o objetivo de demonstrar a eficiência da sincronização em relação ao caso ideal (caso 1) e o algoritmo PTP (caso 3); os últimos 5 casos do grupo azul são repetidos os 5 primeiros com uma malha de controle em rede CSMA/CD, com o objetivo de demonstrar além da eficiência da sincronização, como a mesma influencia o sistema de controle comparado com os casos 1, 3, 8.

O grupo salmão são os casos com o algoritmo FTM e o Algoritmo PReS. O primeiro caso do grupo salmão é o algoritmo FTM puro, os outros são as variações do algoritmo PReS, em que são variadas as funções *cross-fading* em Função Degrau (CFD), Rampa (CFR) e Exponencial (CFE). As 5 primeiras simulações do grupo salmão são realizadas sem uma malha de controle, com o objetivo de demonstrar a eficiência da sincronização em relação ao caso ideal (caso 1) e o algoritmo FTM (caso 13); os últimos 5 casos do grupo salmão são repetidos os 5 primeiros, deste grupo salmão, com uma malha de controle em rede CSMA/CD, com o objetivo de demonstrar além da eficiência da sincronização, como a mesma influencia o sistema de controle, comparando com o caso 1, 13 e 18.

O grupo verde são os casos com o algoritmo SR e o Algoritmo PAReS. O primeiro caso do grupo verde (caso 23) é o algoritmo SR puro, os outros são as variações do algoritmo PAReS, em que são variadas as funções *cross-fading* em Função Degrau (CFD), Rampa (CFR) e Exponencial (CFE). As 5 primeiras simulações do grupo verde são realizadas sem uma malha de controle, com o objetivo de demonstrar a eficiência da sincronização em relação ao caso ideal (caso 1) e o algoritmo SR (caso 23); os últimos 5 casos do grupo verde são repetidos os 5 primeiros, deste grupo verde, com uma malha de controle em rede CSMA/CD, com o objetivo de demonstrar além da eficiência da sincronização, como a mesma influencia o sistema de controle, comparando com o caso 1, 23 e 28.

O grupo roxo são a repetição dos 3 últimos casos de cada grupo (azul, salmão e verde), no entanto, aplicados sobre uma rede TDMA. Os valores de viés inicial, deriva, MMCF, resincronização e transição (T2) dentro de um repertório imenso, foram escolhidos por propiciar uma melhor visualização dos resultados de forma gráfica e numérica sem distanciar demais de valores de relógios reais.

Tabela 8.1 - Casos de Estudos - Verificação e Validação do Algoritmo Proposto.

Casos	STM				Algoritmo STM	NOMO				Algoritmo NOMO	Algoritmo NOMD	NCS	T1	Resincronização	T2
	Viés Inicial					Deriva									
	C1	C2	C3	C4		C1	C2	C3	C4						
Caso 1	0	0	0	0	-	0	0	0	0	-	-	-	-	-	-
Caso 2	0	0,1	0,01	-0,1	Deadbeat Broadcast	0	0,01	0,0001	-0,0001	-	-	-	-	-	
Caso 3	0	0,1	0,01	-0,1	-	0	0,01	0,0001	-0,0001	PTP	-	-		10uT	
Caso 4	0	0,1	0,01	-0,1	Deadbeat Broadcast	0	0,01	0,0001	-0,0001	PTP	-	-	10mT	10uT	
Caso 5	0	0,1	0,01	-0,1	Deadbeat Broadcast	0	0,01	0,0001	-0,0001	PTP+CFD	-		10mT	10uT	
Caso 6	0	0,1	0,01	-0,1	Deadbeat Broadcast	0	0,01	0,0001	-0,0001	PTP+CFR	-		10mT	10uT	
Caso 7	0	0,1	0,01	-0,1	Deadbeat Broadcast	0	0,01	0,0001	-0,0001	PTP+CFE	-		10mT	10uT	
Caso 8	0	0,1	0,01	-0,1	-	0	0,01	0,0001	-0,0001	PTP	-	CSMA/CD		10uT	
Caso 9	0	0,1	0,01	-0,1	Deadbeat Broadcast	0	0,01	0,0001	-0,0001	PTP	-	CSMA/CD	10mT	10uT	
Caso 10	0	0,1	0,01	-0,1	Deadbeat Broadcast	0	0,01	0,0001	-0,0001	PTP+CFD	-	CSMA/CD	10mT	10uT	
Caso 11	0	0,1	0,01	-0,1	Deadbeat Broadcast	0	0,01	0,0001	-0,0001	PTP+CFR	-	CSMA/CD	10mT	10uT	
Caso 12	0	0,1	0,01	-0,1	Deadbeat Broadcast	0	0,01	0,0001	-0,0001	PTP+CFE	-	CSMA/CD	10mT	10uT	
Caso 13	0	0,1	0	-0,1	-	0	0,01	0,0001	-0,0001	FTM	-	-		10uT	
Caso 14	0	0,1	0,01	-0,1	Deadbeat Broadcast	0	0,01	0,0001	-0,0001	FTM	-	-	10mT	10uT	
Caso 15	0	0,1	0,01	-0,1	Deadbeat Broadcast	0	0,01	0,0001	-0,0001	FTM+CFD	-	-	10mT	10uT	
Caso 16	0	0,1	0,01	-0,1	Deadbeat Broadcast	0	0,01	0,0001	-0,0001	FTM+CFR	-	-	10mT	10uT	
Caso 17	0	0,1	0,01	-0,1	Deadbeat Broadcast	0	0,01	0,0001	-0,0001	FTM+CFE	-	-	10mT	10uT	
Caso 18	0	0,1	0	-0,1	-	0	0,01	0,0001	-0,0001	FTM	-	CSMA/CD		10uT	
Caso 19	0	0,1	0,01	-0,1	Deadbeat Broadcast	0	0,01	0,0001	-0,0001	FTM	-	CSMA/CD	10mT	10uT	
Caso 20	0	0,1	0,01	-0,1	Deadbeat Broadcast	0	0,01	0,0001	-0,0001	FTM+CFD	-	CSMA/CD	10mT	10uT	
Caso 21	0	0,1	0,01	-0,1	Deadbeat Broadcast	0	0,01	0,0001	-0,0001	FTM+CFR	-	CSMA/CD	10mT	10uT	
Caso 22	0	0,1	0,01	-0,1	Deadbeat Broadcast	0	0,01	0,0001	-0,0001	FTM+CFE	-	CSMA/CD	10mT	10uT	
Caso 23	0	0,1	0	-0,1	-	0	0,01	0,0001	-0,0001	FTM	SR	-		10uT	3mT

Caso 24	0	0,1	0,01	-0,1	Deadbeat Broadcast	0	0,01	0,0001	-0,0001	FTM	SR	-	10mT	10uT	3mT
Caso 25	0	0,1	0,01	-0,1	Deadbeat Broadcast	0	0,01	0,0001	-0,0001	FTM+CFD	SR	-	10mT	10uT	3mT
Caso 26	0	0,1	0,01	-0,1	Deadbeat Broadcast	0	0,01	0,0001	-0,0001	FTM+CFR	SR	-	10mT	10uT	3mT
Caso 27	0	0,1	0,01	-0,1	Deadbeat Broadcast	0	0,01	0,0001	-0,0001	FTM+CFE	SR	-	10mT	10uT	3mT
Caso 28	0	0,1	0	-0,1	-	0	0,01	0,0001	-0,0001	FTM	SR	CSMA/CD		10uT	3mT
Caso 29	0	0,1	0,01	-0,1	Deadbeat Broadcast	0	0,01	0,0001	-0,0001	FTM	SR	CSMA/CD	10mT	10uT	3mT
Caso 30	0	0,1	0,01	-0,1	Deadbeat Broadcast	0	0,01	0,0001	-0,0001	FTM+CFD	SR	CSMA/CD	10mT	10uT	3mT
Caso 31	0	0,1	0,01	-0,1	Deadbeat Broadcast	0	0,01	0,0001	-0,0001	FTM+CFR	SR	CSMA/CD	10mT	10uT	3mT
Caso 32	0	0,1	0,01	-0,1	Deadbeat Broadcast	0	0,01	0,0001	-0,0001	FTM+CFE	SR	CSMA/CD	10mT	10uT	3mT
Caso 33	0	0,1	0,01	-0,1	Deadbeat Broadcast	0	0,01	0,0001	-0,0001	PTP+CFD	-	TDMA	10mT	10uT	
Caso 34	0	0,1	0,01	-0,1	Deadbeat Broadcast	0	0,01	0,0001	-0,0001	PTP+CFR	-	TDMA	10mT	10uT	
Caso 35	0	0,1	0,01	-0,1	Deadbeat Broadcast	0	0,01	0,0001	-0,0001	PTP+CFE	-	TDMA	10mT	10uT	
Caso 36	0	0,1	0,01	-0,1	Deadbeat Broadcast	0	0,01	0,0001	-0,0001	FTM+CFD	-	TDMA	10mT	10uT	
Caso 37	0	0,1	0,01	-0,1	Deadbeat Broadcast	0	0,01	0,0001	-0,0001	FTM+CFR	-	TDMA	10mT	10uT	
Caso 38	0	0,1	0,01	-0,1	Deadbeat Broadcast	0	0,01	0,0001	-0,0001	FTM+CFE	-	TDMA	10mT	10uT	
Caso 39	0	0,1	0,01	-0,1	Deadbeat Broadcast	0	0,01	0,0001	-0,0001	FTM+CFD	SR	TDMA	10mT	10uT	3mT
Caso 40	0	0,1	0,01	-0,1	Deadbeat Broadcast	0	0,01	0,0001	-0,0001	FTM+CFR	SR	TDMA	10mT	10uT	3mT
Caso 41	0	0,1	0,01	-0,1	Deadbeat Broadcast	0	0,01	0,0001	-0,0001	FTM+CFE	SR	TDMA	10mT	10uT	3mT

8.3. Caso 1 - Ideal

Este caso é o ideal para a sincronização de relógios, pois, nenhum dos relógios do conjunto possui alguma imperfeição e atraso de rede. A Figura 8.4 mostra o microtick dos relógios e a Figura 8.5 a diferença entre os microticks que é zero. Os valores de tempo de todos os relógios são idênticos.

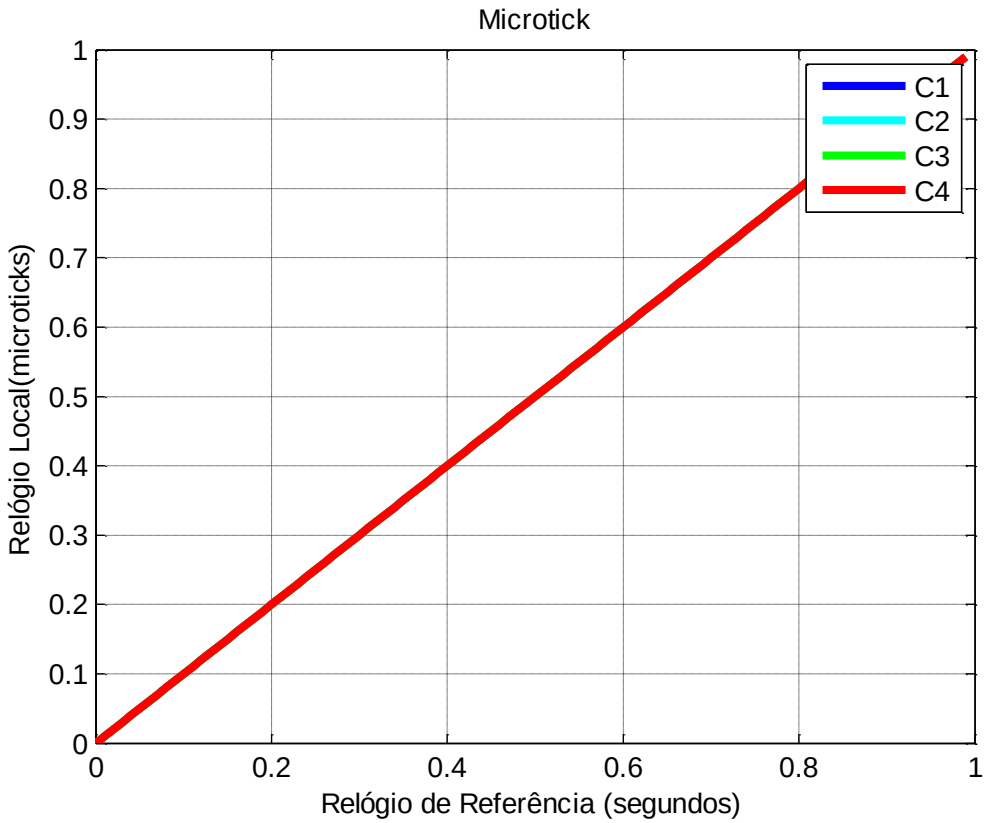


Figura 8.4 – Caso 1 - Microtick.

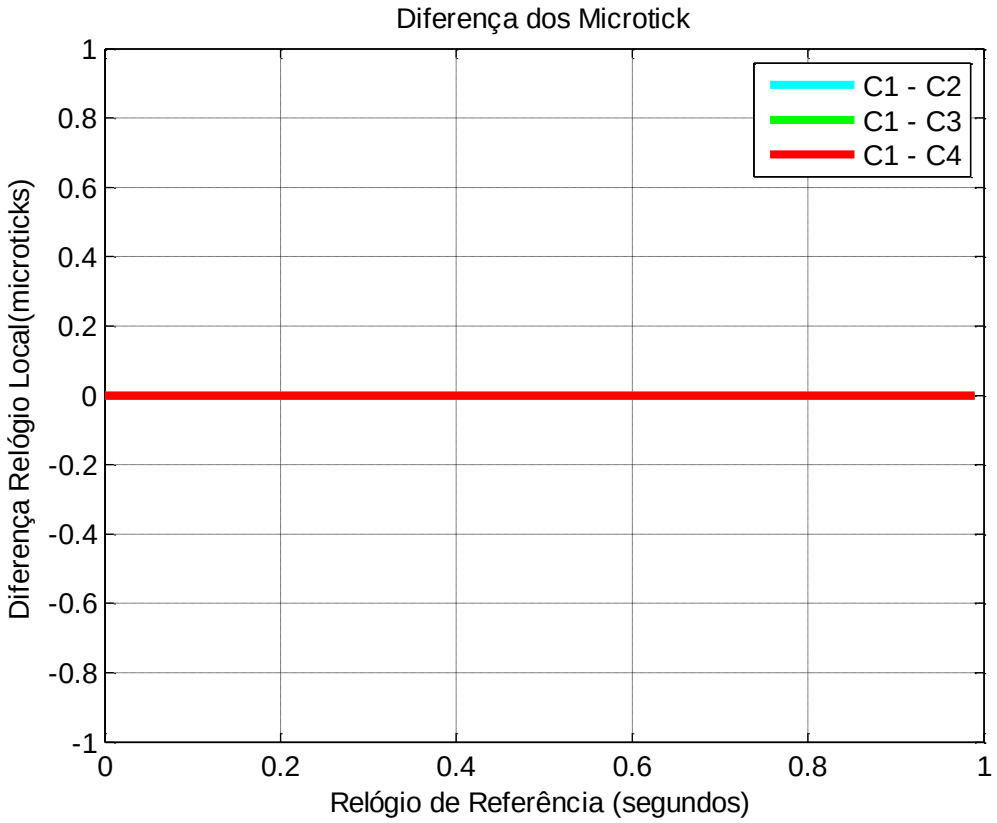


Figura 8.5 – Caso 1 - Diferença entre Microticks.

Já a Figura 8.6 mostra os valores de macrotick e a Figura 8.7 é a diferença entre os macroticks. O fator de conversão microtick-macrotick (MMCF) é 1. E, portanto, o valor do macrotick é muito próximo do microtick e as diferenças entre os macroticks é zero.

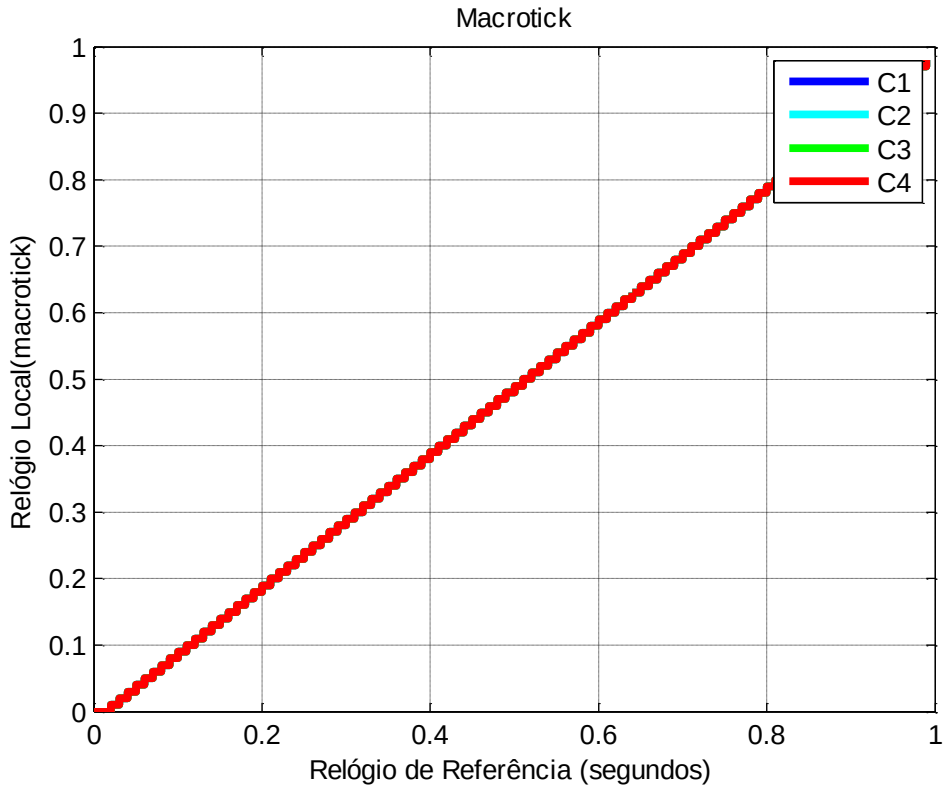


Figura 8.6 – Caso 1 - Macrotick.

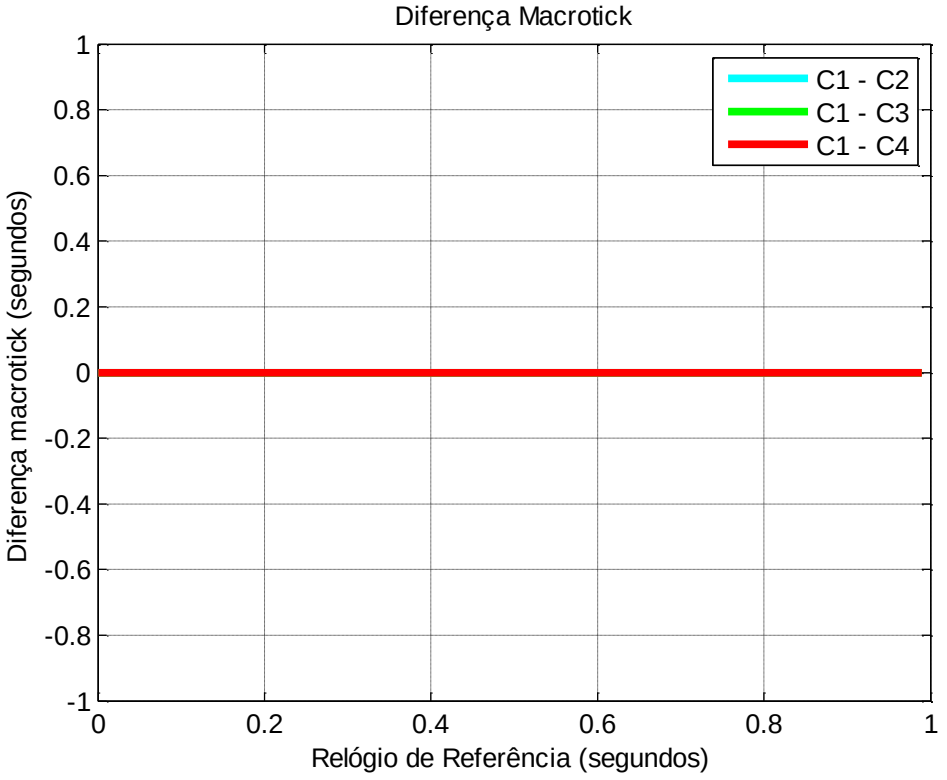


Figura 8.7 – Caso 1 - Diferença entre Macroticks.

Todos os casos seguintes, foram implementados com imperfeições para demonstrar a sincronização e os efeitos sobre a dinâmica.

8.4. Caso 2 - Deadbeat

Este caso possui valores de viés inicial igual a: C1: 0; C2: 0.1 seg. ; C3: 0.01 seg. ; C4: -0.1 seg.; e deriva igual a: C1: 0; C2: 1%; C3: 0.01%; C4: -0.01%.

Assim, a Figura 8.8 e Figura 8.9 apresentam os valores de microticks e diferença entre os microticks respectivamente. Observa-se que inicialmente os valores estão de-sincronizados e a diferença entre os relógios, apesar de pequena, vai aumentando com o tempo. Isto é causado devido ao viés inicial e a deriva. C1 será considerado como relógio de referência.

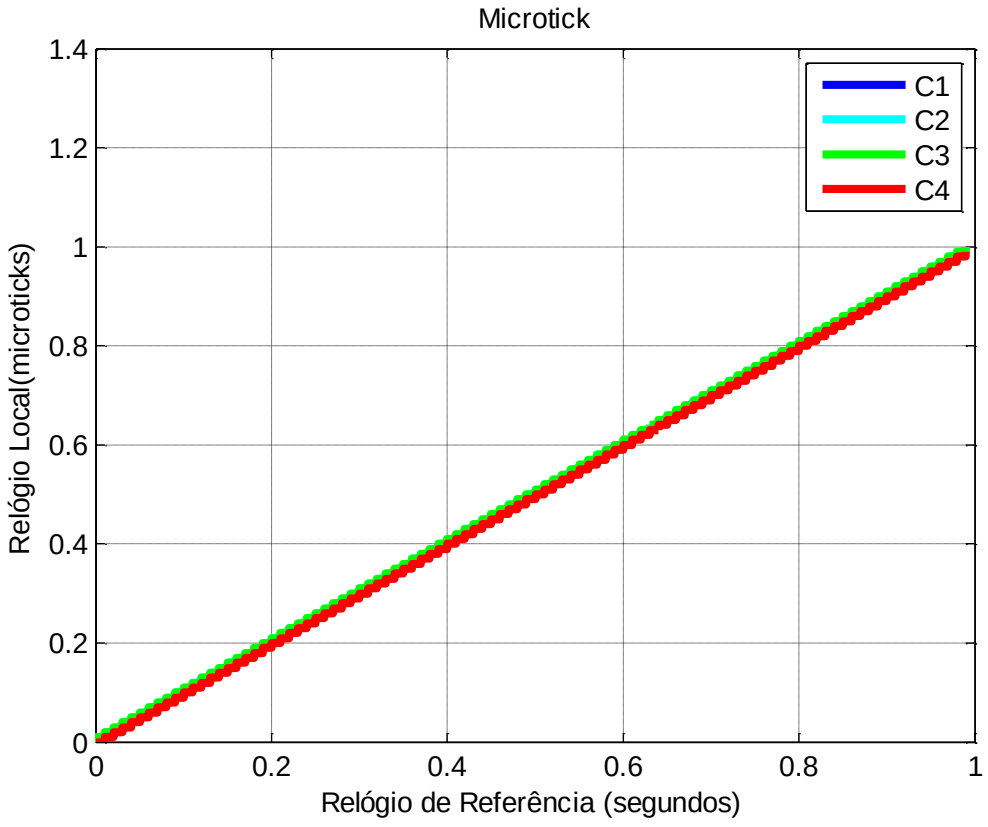


Figura 8.8 – Caso 2 - Microtick.

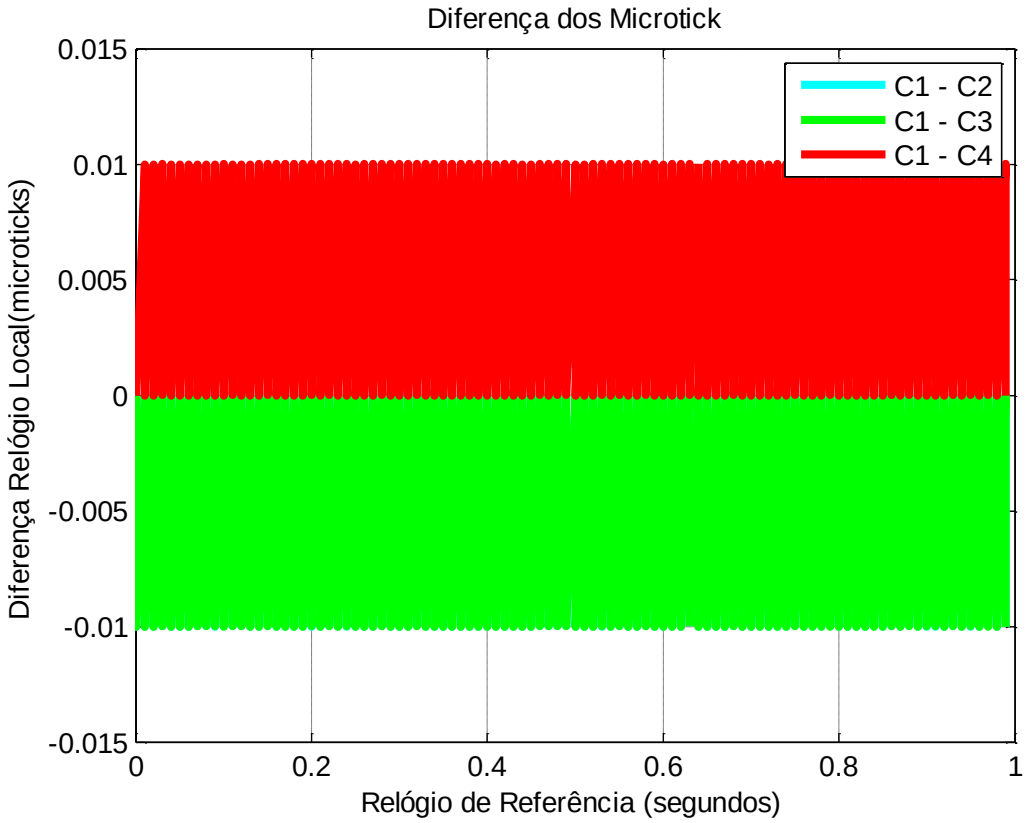


Figura 8.9 – Caso 2 - Diferença entre Microticks.

A Figura 8.8 e Figura 8.9 serão iguais para todos os outros casos de simulação e, portanto, serão apresentadas somente neste caso.

Assim, devido às imperfeições do microtick, os macroticks também se de-sincronizam entre si. Neste caso 2, para corrigir a de-sincronização causada por estas imperfeições aplica-se uma sincronização com o controlador *deadbeat*.

A Figura 8.10 e Figura 8.11 mostram os valores de macrotick e a diferença dos macroticks em relação ao relógio de referência (C1), respectivamente. A Figura 8.12 mostra o valor do controlador *deadbeat* (valor de correção do relógio).

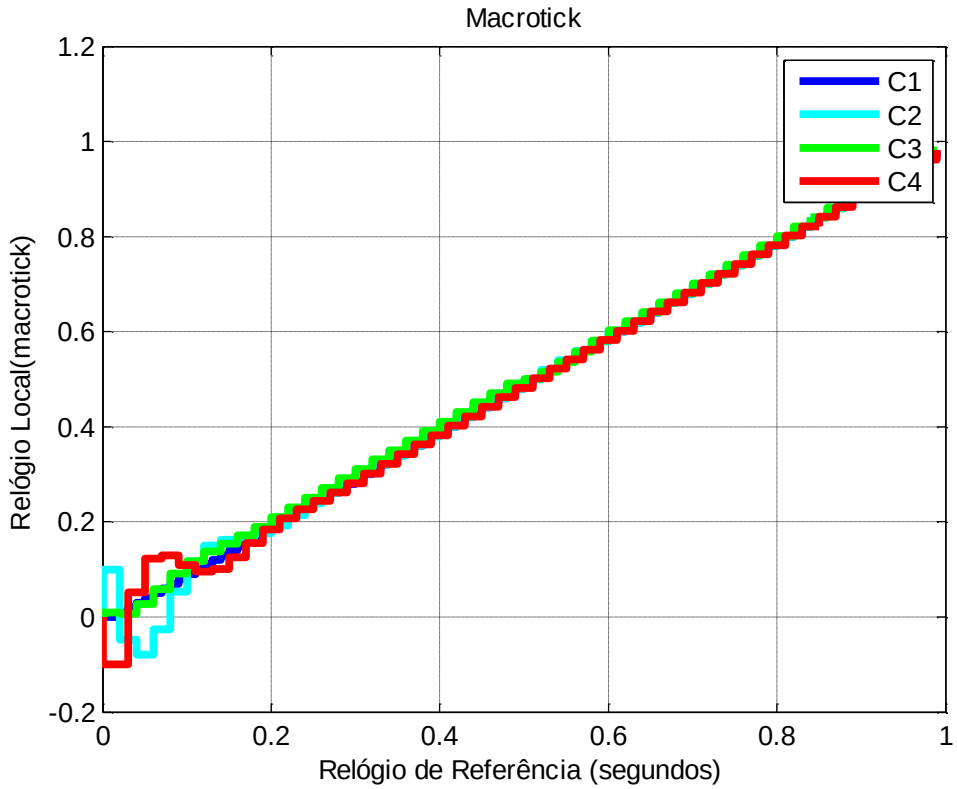


Figura 8.10 – Caso 2 - Macrotick.

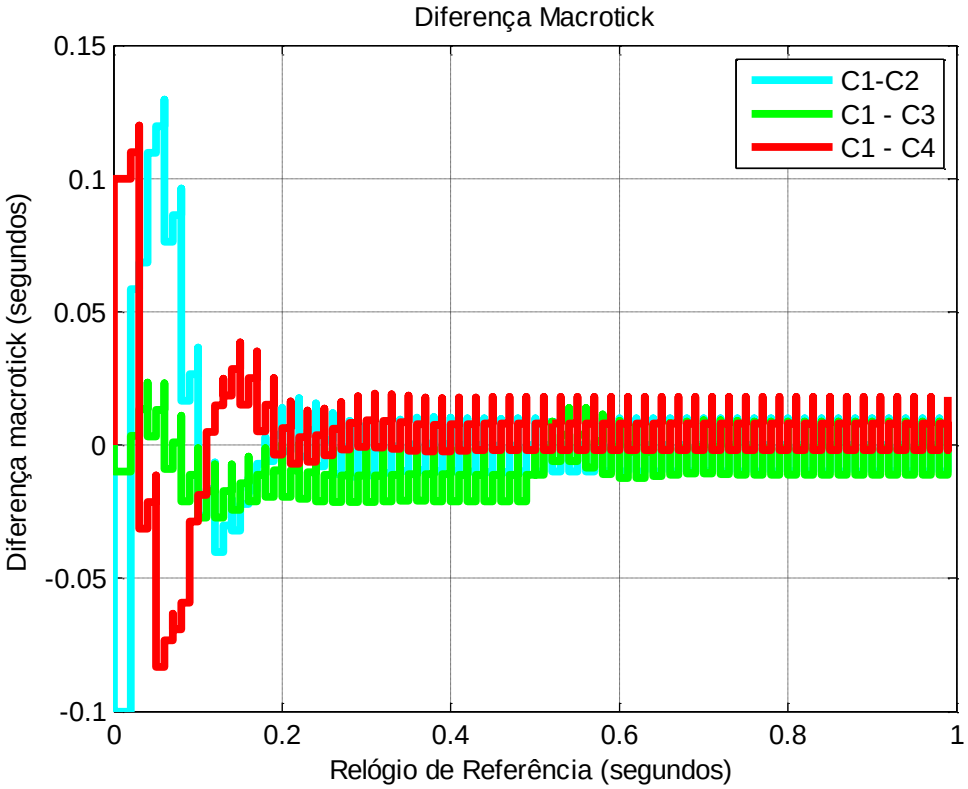


Figura 8.11 – Caso 2 - Diferença entre os Macrotick.

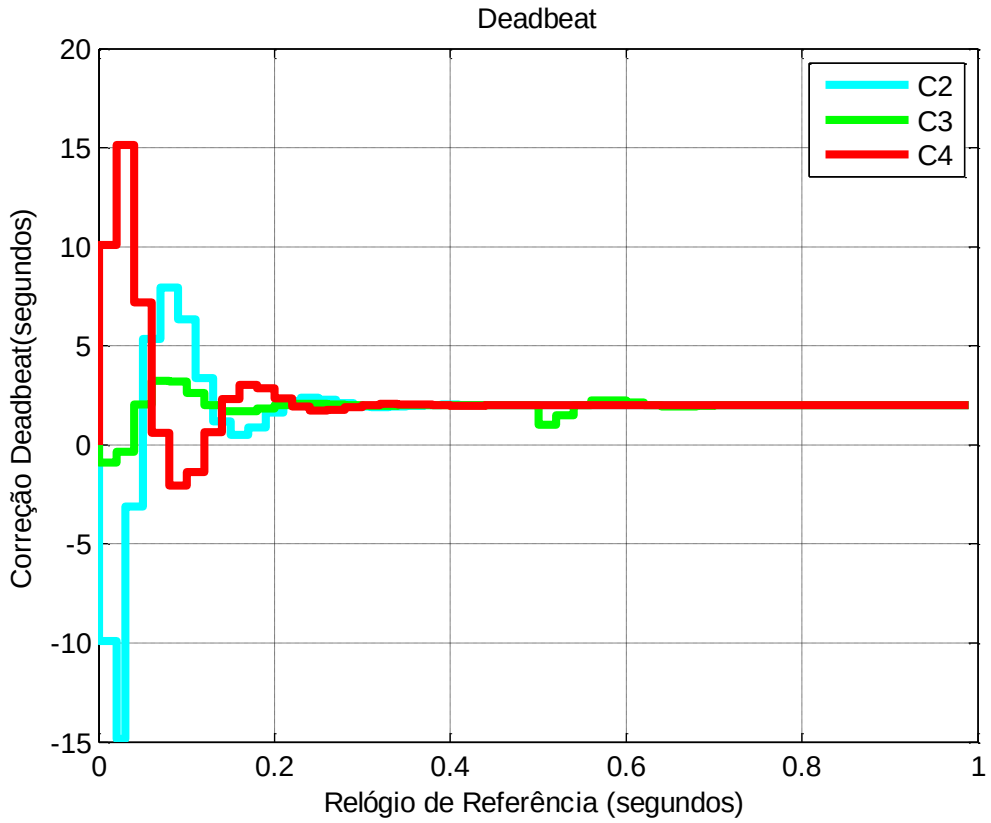


Figura 8.12 – Caso 2 - Controle *Deadbeat*.

Este caso demonstra que o controlador *deadbeat* pode ser aplicado para sincronizar relógios. O controlador se mostrou muito eficiente, sendo as uma das principais vantagens da sincronização ser em malha fechada e ter uma convergência do erro rápido e conhecida. A desvantagem é que o controlador é definido para uma entrada especifica e deve ser aplicado a cada instante do macrotick.

8.5. Caso 3 - Algoritmo PTP

Este caso possui valores de viés inicial igual a: C1: 0; C2: 0.1 seg. ; C3: 0.01 seg. ; C4: -0.1 seg.; e deriva igual a: C1: 0; C2: 1%; C3: 0.01%; C4: -0.01%.

A Figura 8.13 e Figura 8.14 mostram os valores de macrotick e diferença dos macroticks em relação ao relógio de referência (C1), respectivamente.

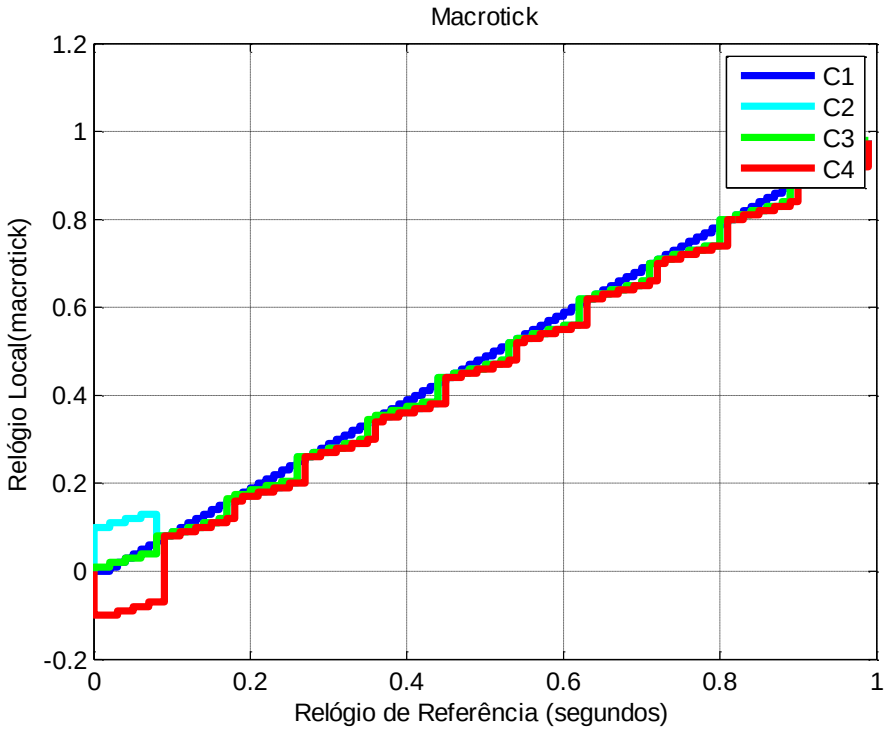


Figura 8.13 – Caso 3 - Macrotick.

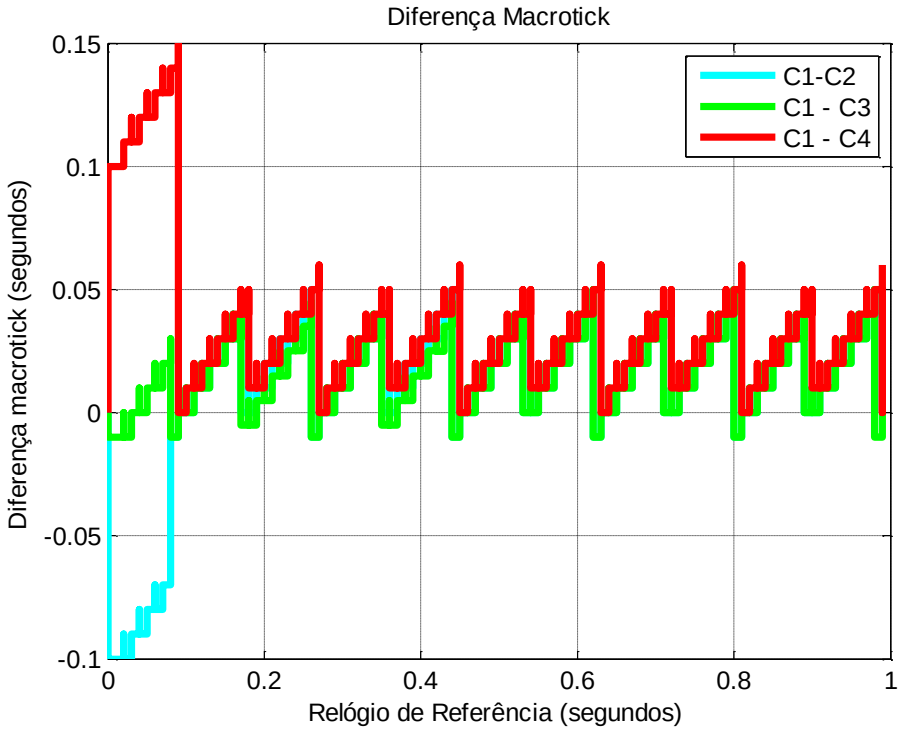


Figura 8.14 – Caso 3 - Diferença entre Macroticks.

A Figura 8.15 mostra o valor de correção do relógio calculado pela função de convergência do algoritmo PTP. O período de re-sincronização é de 10 microticks, ou seja, a cada 10 microticks aplica-se o processo de sincronização do algoritmo PTP.

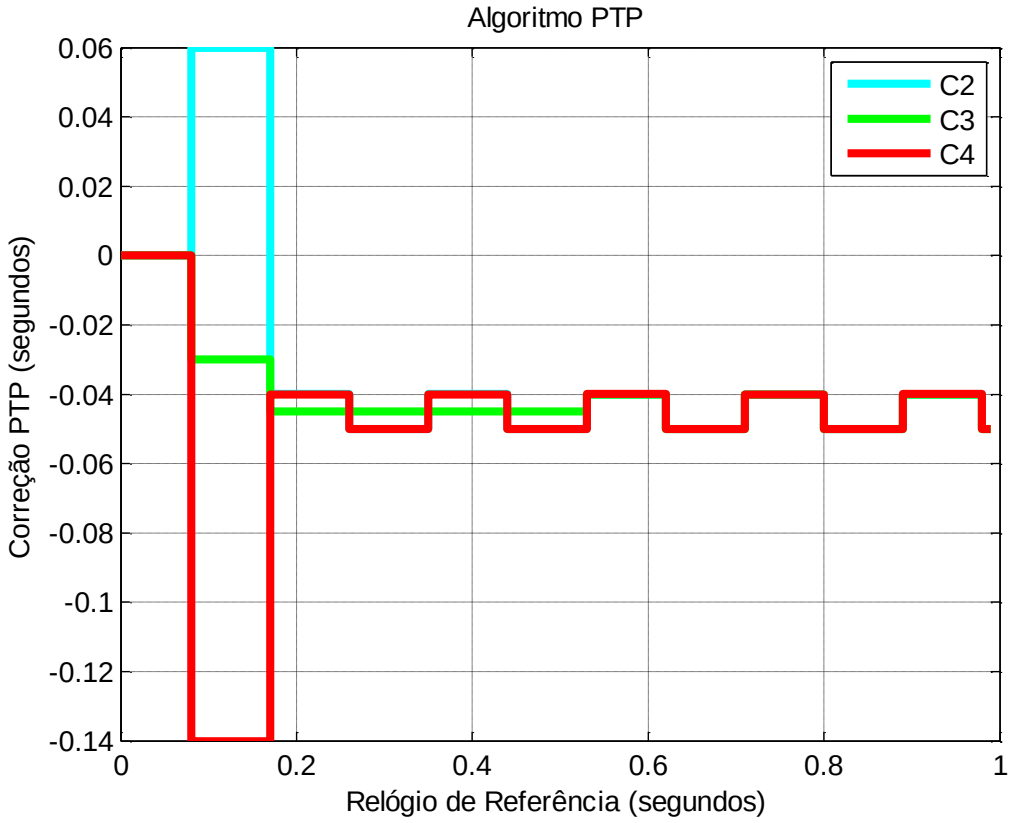


Figura 8.15 – Caso 3 - Correção PTP.

Este caso mostra uma sincronização de relógios com o algoritmo PTP. O algoritmo se mostrou eficiente, sendo as principais vantagens a de a sincronização ser uma equação de convergência simples e para qualquer tipo entrada de tempo. A desvantagem é que o algoritmo não faz o gerenciamento da descontinuidade de tempo e com isso a volta de tempo é muito grande como se observa na Figura 8.15; E também não é preparado para levar em conta o viés inicial, apesar de corrigi-lo.

8.6. Caso 4 - DeadBeat - PTP

Este caso possui valores de viés inicial igual a: C1: 0; C2: 0.1 seg. ; C3: 0.01 seg. ; C4: -0.1 seg.; e deriva igual a: C1: 0; C2: 1%; C3: 0.01%; C4: -0.01%.

Neste caso, aplica-se um algoritmo com os dois modos de correção: 1) STM e o 2) NOMO, descritos no capítulo 6. Com isto, visa-se inicialmente corrigir o viés inicial com o controlador *deadbeat* e depois utiliza-se a técnica PTP. A Figura 8.16 e Figura 8.17 mostram, respectivamente, os valores de macrotick e diferença entre o relógio de referência e os macroticks.

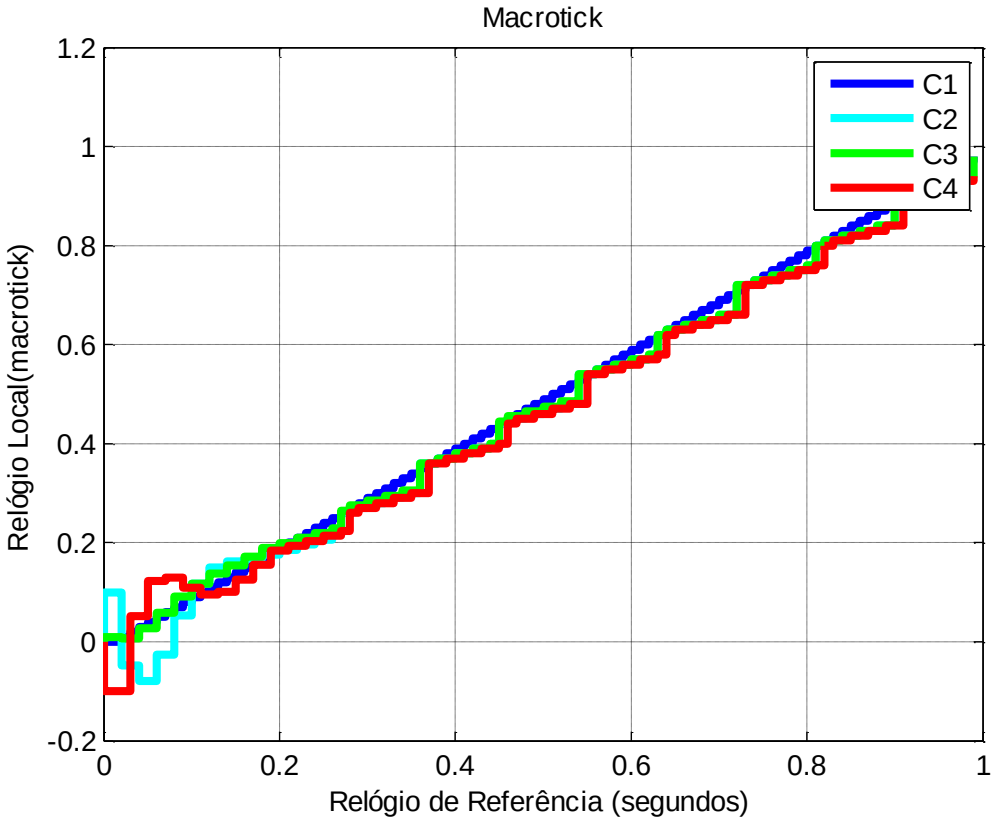


Figura 8.16 – Caso 4 - Macrotick.

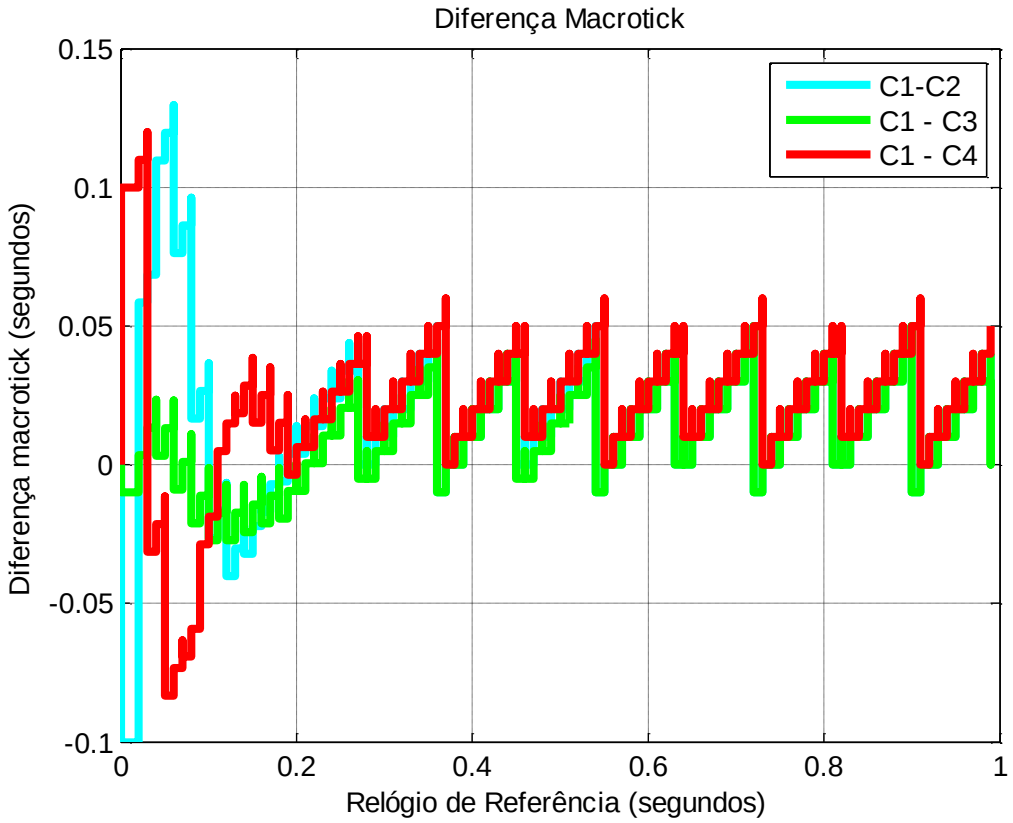


Figura 8.17 – Caso 4 - Diferença entre Macroticks.

A Figura 8.18 mostra o valor de correção do relógio calculado pelo controlador *deadbeat*; a Figura 8.19 a correção calculada pela função de convergência do algoritmo PTP. O tempo de transição dos modos de correção é de 10 macroticks e o período de re-sincronização é de 10 microticks, ou seja, após 10 macroticks, a correção passa a ser feita pelo PTP que a cada 10 microticks aplica-se o processo de sincronização do algoritmo.

Este caso mostra uma sincronização de relógios de métrica mista com o algoritmo proposto. O algoritmo se mostrou eficiente, sendo que a principal vantagem é de a sincronização evitar uma descontinuidade de tempo muito grande sem perda da exatidão, o que foi observado entre a Figura 8.15 e a Figura 8.19. A desvantagem é que o algoritmo precisa de um gerenciamento de transições. No entanto, o tempo da transição é conhecido, devido ao *deadbeat*, o que minimiza esta desvantagem.

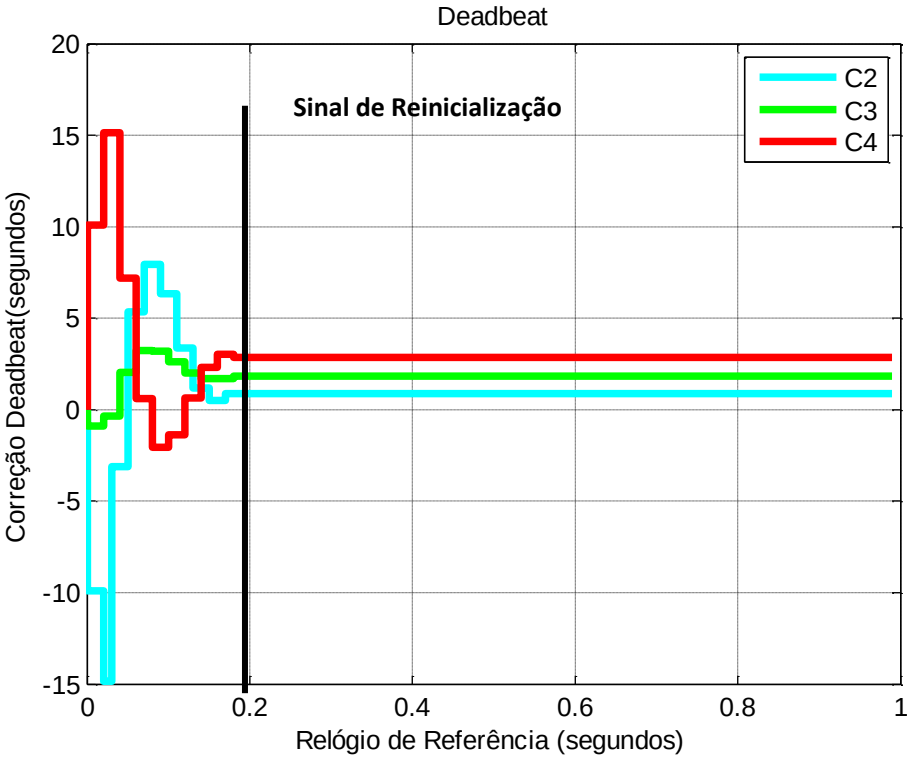


Figura 8.18 – Caso 4 - Correção STM - Controle *Deadbeat*.

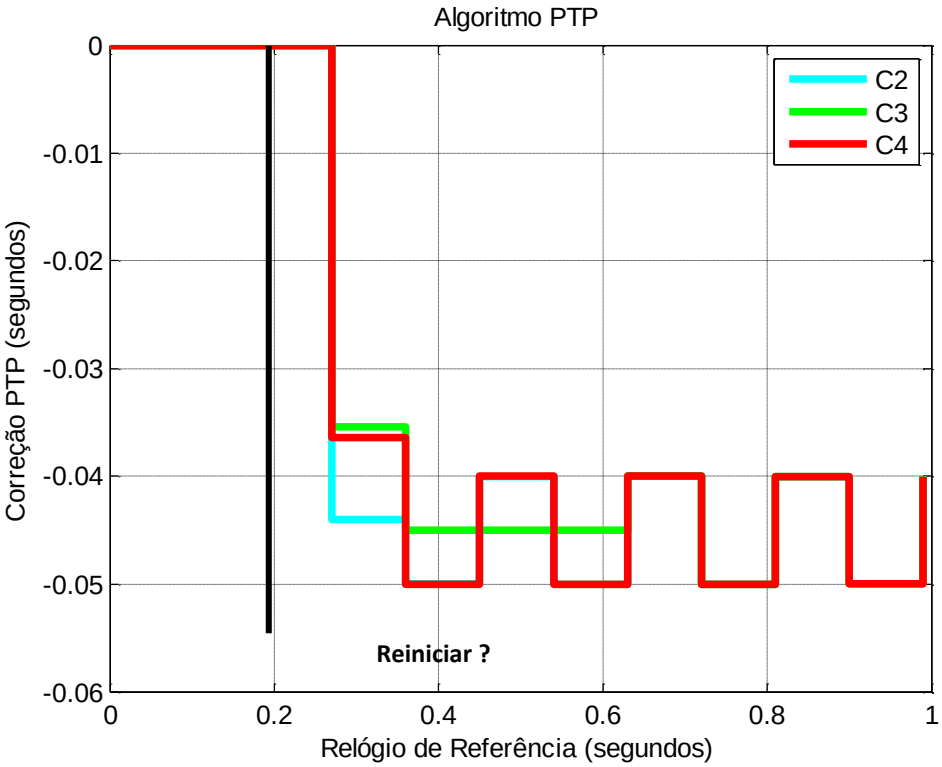


Figura 8.19 – Caso 4 - Correção NOMO - Algoritmo PTP.

8.7. Caso 5 - Algoritmo AReS Degrau

Este caso possui valores de viés inicial igual a: C1: 0; C2: 0.1 seg. ; C3: 0.01 seg. ; C4: -0.1 seg.; e deriva igual a: C1: 0; C2: 1%; C3: 0.01%; C4: -0.01%.

Neste caso, aplica-se um algoritmo com os dois modos de correção: 1) STM e o 2) NOMO, descritos no capítulo 6. Com isto, visa-se inicialmente corrigir o viés inicial com o controlador *deadbeat* e depois utiliza-se a técnica PTP em conjunto com uma função *cross-fading* degrau, definindo o algoritmo AReS degrau. A Figura 8.20 e Figura 8.21 mostram, respectivamente, os valores de macrotick e diferença entre o relógio de referência e os macroticks.

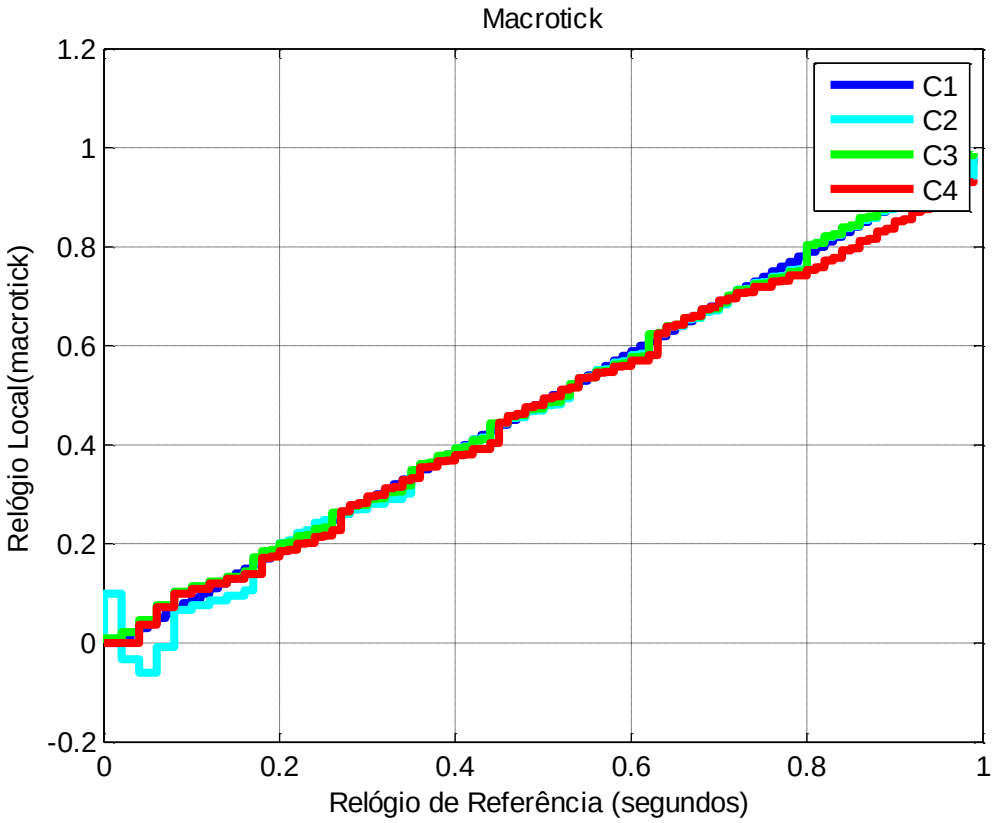


Figura 8.20 – Caso 5 - Macrotick.

A Figura 8.22 mostra o valor de correção do relógio calculado pelo controlador *deadbeat*.

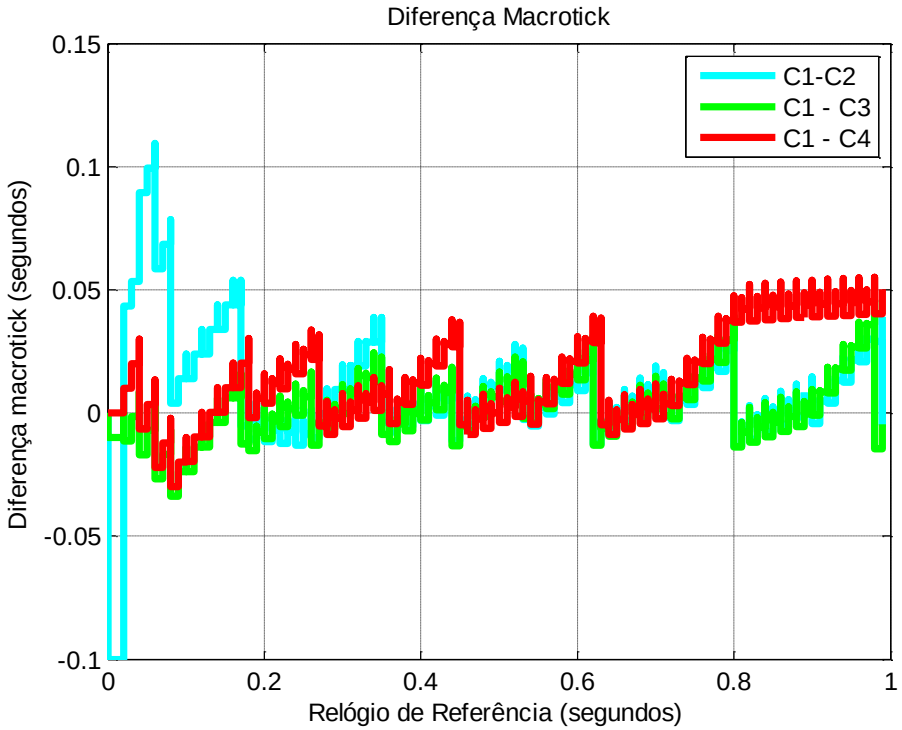


Figura 8.21 – Caso 5 - Diferença entre Macroticks.

A Figura 8.23 mostra a função *cross-fading* degrau. A Figura 8.24 mostra a correção calculada pela função de convergência do algoritmo PTP mais a função *cross-fading* degrau. O tempo de transição entre os modos de correção é de 10 macroticks (existe uma transição entre o STM e NOMO após 10 macroticks) e o período de re-sincronização é de 10 microticks, ou seja, após 10 macroticks, a correção é calculada pela função de convergência do algoritmo PTP. No entanto, com o uso da função degrau, a cada instante (assim como no *deadbeat*) é aplicada uma correção multiplicada pelo valor da função *cross-fading*.

Este caso mostra uma sincronização de relógios de métrica mista com o algoritmo AReS degrau. O algoritmo se mostrou eficiente, sendo que a principal vantagem é da sincronização modular a descontinuidade de tempo via uma função degrau, evitando excursões de tempo muito grandes e, ainda, melhorando a exatidão. Isto ocorre, devido à função *cross-fading* modular a correção do algoritmo ao longo de cada instante, diferente do algoritmo PTP

que aplica toda a correção em um único instante. A desvantagem é que o algoritmo precisa de um gerenciamento de transições e necessita atualizar a cada instante o valor de tempo. No entanto, o tempo da transição é conhecido, devido ao *deadbeat*, o que minimiza esta desvantagem.

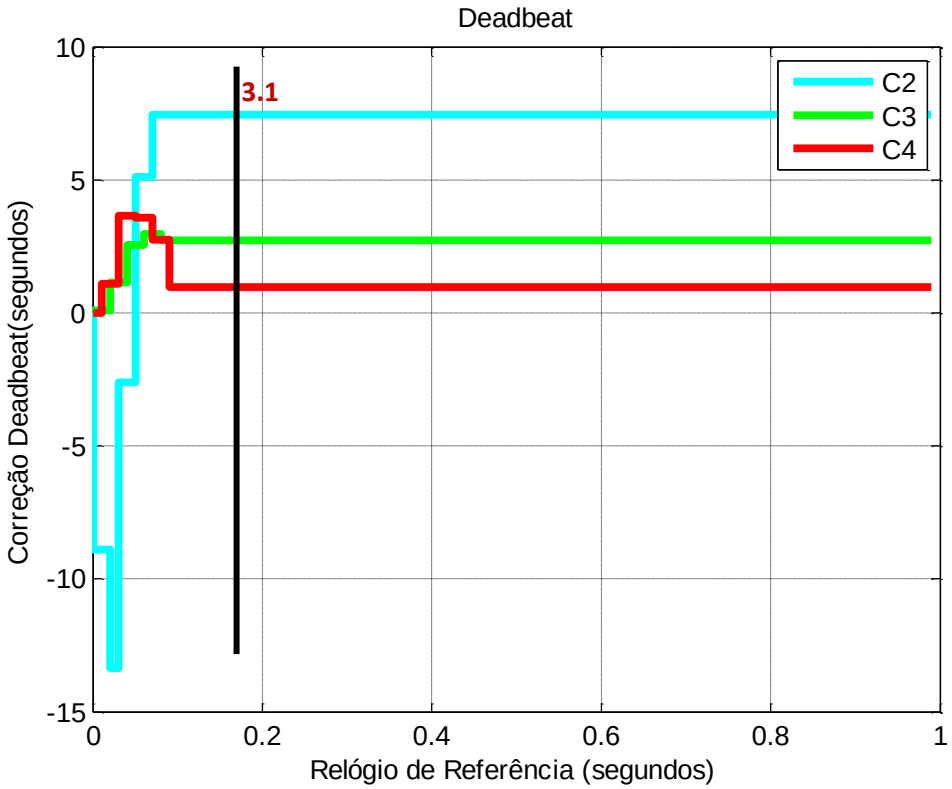


Figura 8.22 – Caso 5 - Correção STM - Algoritmo AReS Degrau.

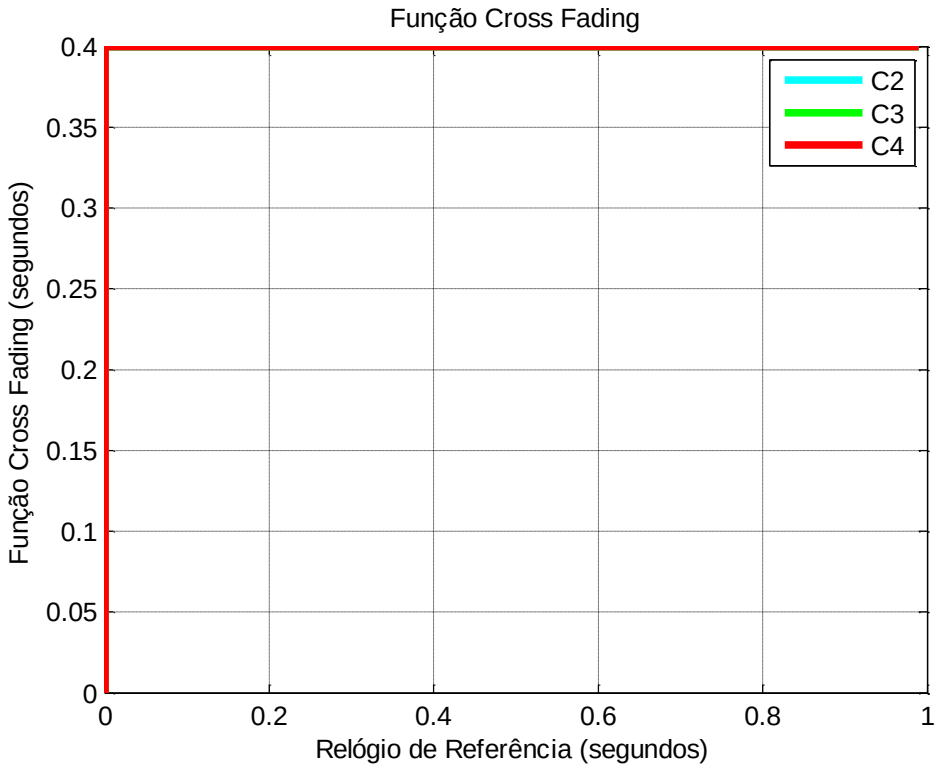


Figura 8.23 – Caso 5 - Função Cross-Fading Degrau.

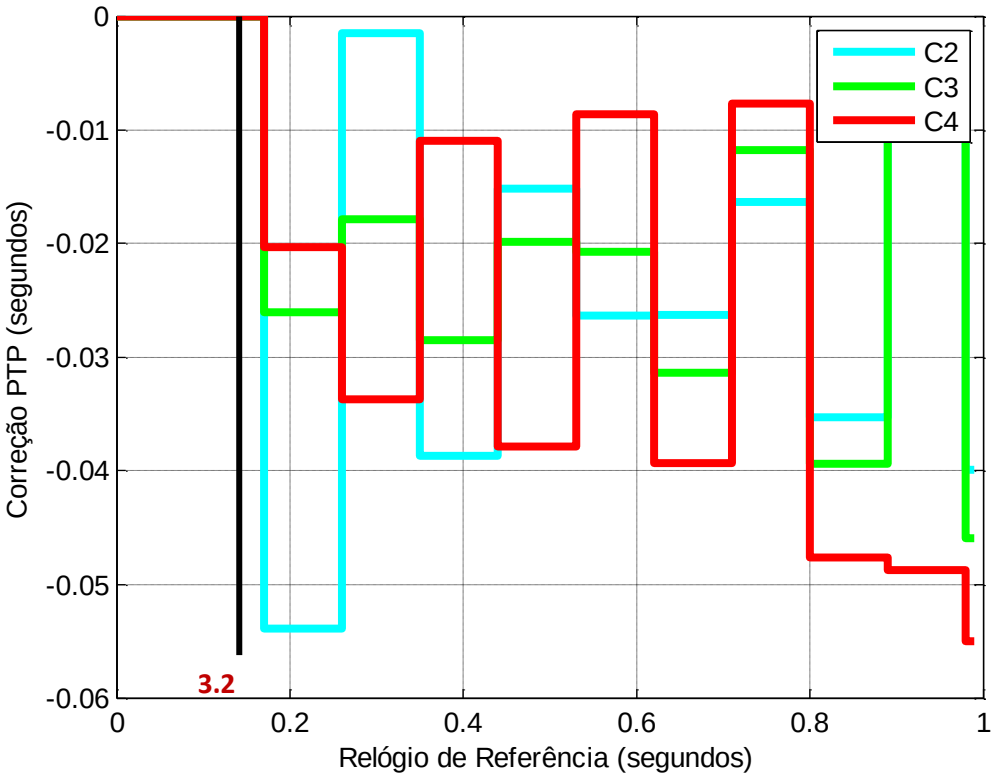


Figura 8.24 – Caso 5 - Correção NOMO - Algoritmo AReS Degrau.

8.8. Caso 6 - Algoritmo AReS Rampa

Este caso possui valores de viés inicial igual a: C1: 0; C2: 0.1 seg. ; C3: 0.01 seg. ; C4: -0.1 seg.; e deriva igual a: C1: 0; C2: 1%; C3: 0.01%; C4: -0.01%.

Neste caso, aplica-se um algoritmo com os dois modos de correção: 1) STM e o 2) NOMO, descritos no capítulo 6. Com isto, visa-se inicialmente corrigir o viés inicial com o controlador *deadbeat* e depois utiliza-se a técnica PTP em conjunto com uma função cross-fading rampa, definindo o algoritmo AReS. A Figura 8.25 e Figura 8.26 mostram, respectivamente, os valores de macrotick e diferença entre o relógio de referência e os macroticks.

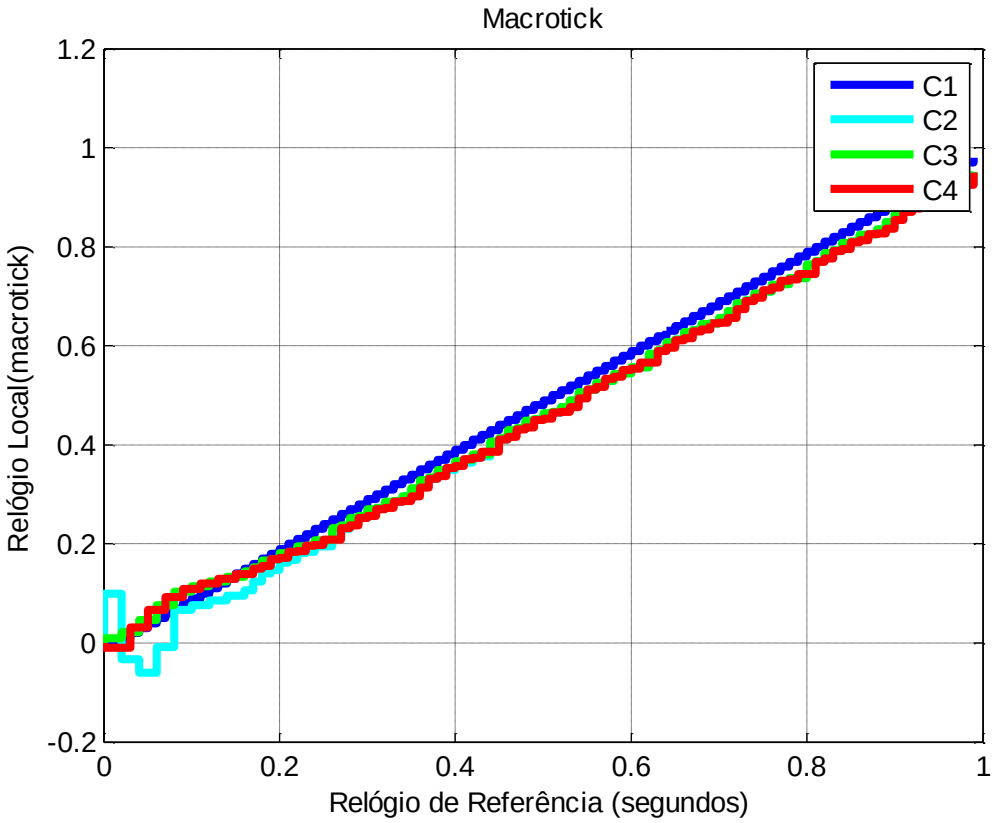


Figura 8.25 – Caso 6 - Macrotick.

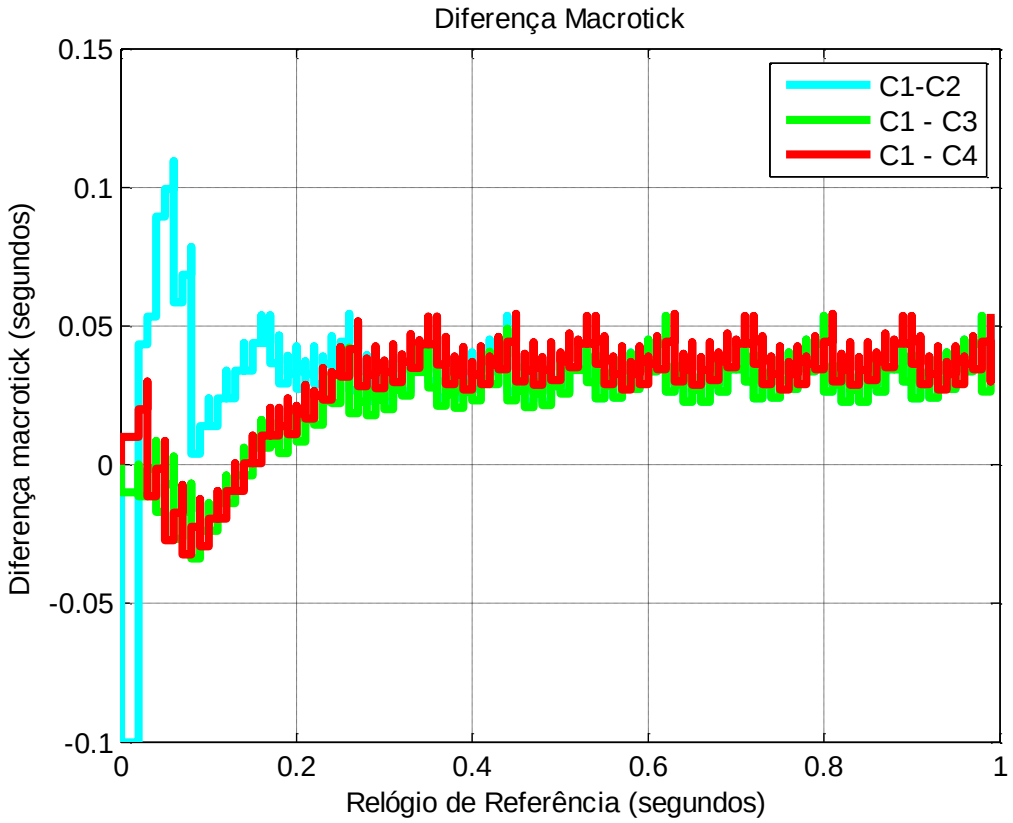


Figura 8.26 – Caso 6 - Diferença entre Macroticks.

A Figura 8.27 mostra o valor de correção do relógio calculado pelo controlador *deadbeat*. A Figura 8.28 mostra a função *cross-fading* rampa. A Figura 8.29 mostra a correção calculada pela função de convergência do algoritmo PTP mais a função *cross-fading* rampa. O tempo de transição entre os modos de correção é de 10 macroticks (existe uma transição entre o STM e NOMO após 10 macroticks) e o período de re-sincronização é de 10 microticks, ou seja, após 10 macroticks, a correção é calculada pela função de convergência do algoritmo PTP. No entanto, com o uso da função rampa, a cada instante é aplicada uma correção multiplicada pelo valor da função *cross-fading*. Esta correção vai decaindo de valor de acordo com a função *cross-fading*. No caso da reta, é necessária uma segunda transição para evitar que o valor da função *cross-fading* rampa fique negativa e inverta a fase do sistema tornando-o instável.

Este caso mostra uma sincronização de relógios de métrica mista com o algoritmo AReS. O algoritmo se mostrou eficiente, sendo que a principal vantagem é da possibilidade de modular a descontinuidade de tempo devido à sincronização e ainda melhorar a exatidão. Isto ocorre, devido à função cross-fading modular a correção do algoritmo ao longo de cada instante, diferente do algoritmo PTP que aplica toda a correção em um único instante. A desvantagem é que o algoritmo precisa de um gerenciamento de transições e necessita atualizar a cada instante o valor de tempo. No entanto, o tempo da transição é conhecido, devido ao *deadbeat*, o que minimiza esta desvantagem.

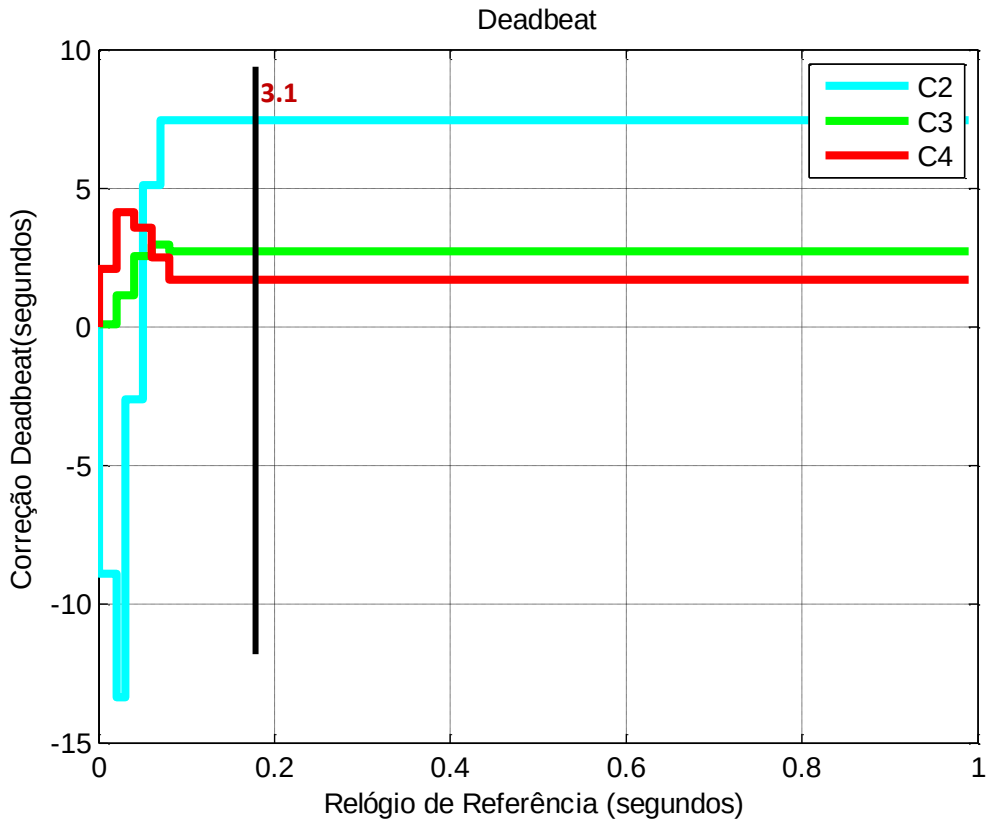


Figura 8.27 – Caso 6 - Correção STM - Algoritmo AReS Rampa.

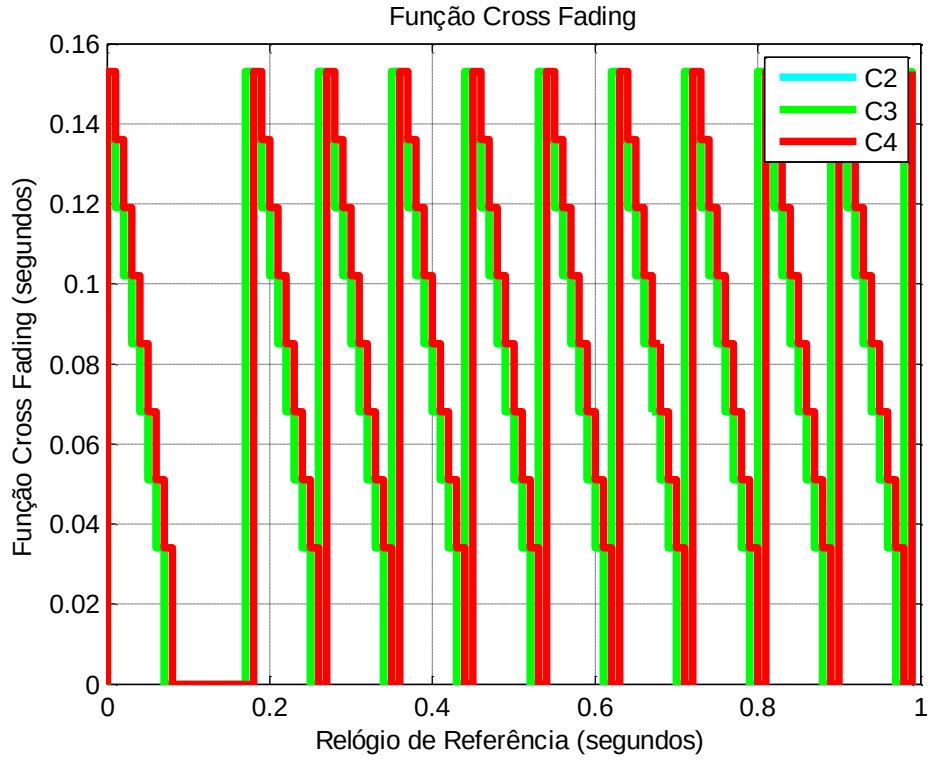


Figura 8.28 – Caso 6 - Função *Cross-Fading* Degrau.

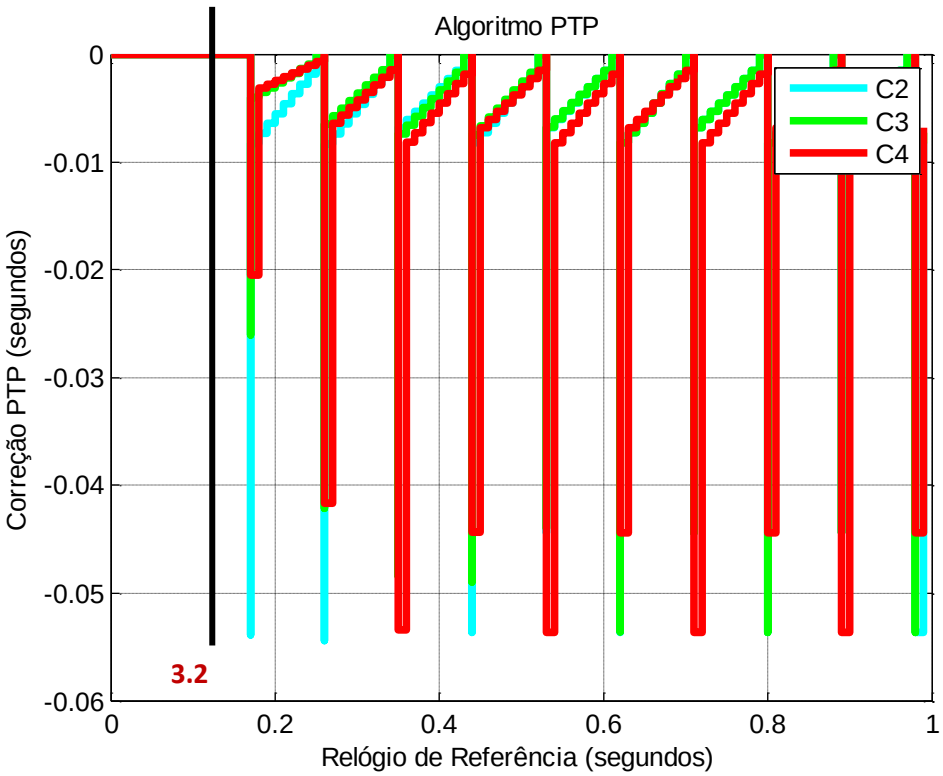


Figura 8.29 – Caso 6 - Correção NOMO - Algoritmo AReS - Rampa.

8.9. Caso 7 - Algoritmo AReS Exponencial

Este caso possui valores de viés inicial igual a: C1: 0; C2: 0.1 seg. ; C3: 0.01 seg. ; C4: -0.1 seg.; e deriva igual a: C1: 0; C2: 1%; C3: 0.01%; C4: -0.01%.

Neste caso, aplica-se um algoritmo com os dois modos de correção: 1) STM e o 2) NOMO, descritos no capítulo 6. Com isto, visa-se inicialmente corrigir o viés inicial com o controlador *deadbeat* e depois utiliza-se a técnica PTP em conjunto com uma função cross-fading exponencial, definindo o algoritmo AReS Exponencial. A Figura 8.30 e Figura 8.31 mostram, respectivamente, os valores de macrotick e diferença entre o relógio de referência e os macroticks.

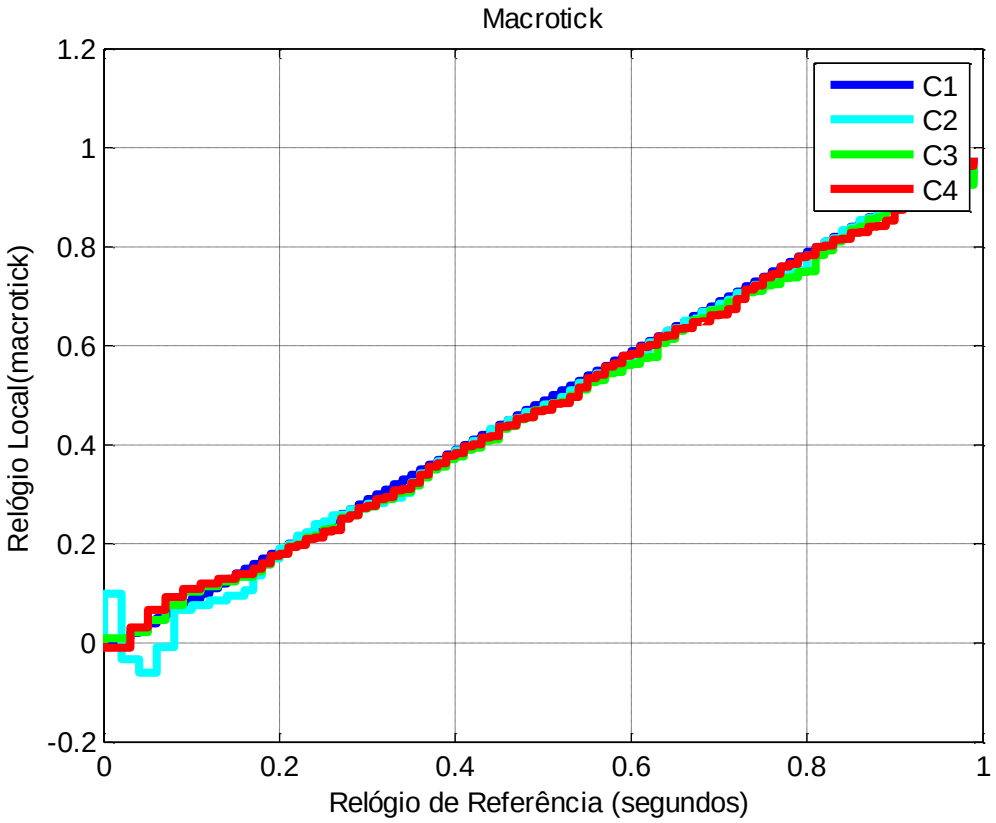


Figura 8.30 – Caso 7 - Macrotick.

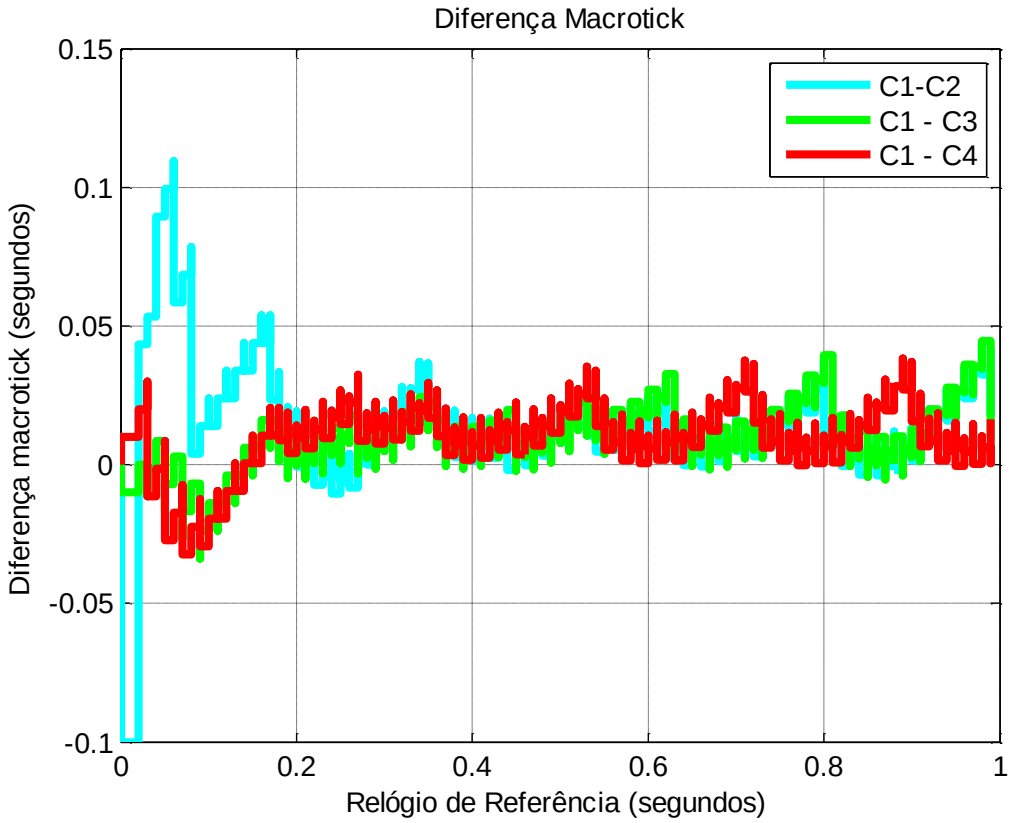


Figura 8.31 – Caso 7 - Diferença entre Macroticks.

A Figura 8.32 mostra o valor de correção do relógio calculado pelo controlador *deadbeat* no modo STM. A Figura 8.33 mostra a função *cross-fading* exponencial. A Figura 8.34 mostra a correção calculada pela função de convergência do algoritmo PTP mais a função *cross-fading* exponencial no modo NOMO. O tempo de transição entre os modos de correção é de 10 macroticks (existe uma transição entre o STM e NOMO após 10 macroticks) e o período de re-sincronização é de 10 microticks, ou seja, após 10 macroticks, a correção é recalculada. No entanto, com o uso da função exponencial, a cada instante é aplicada uma correção multiplicada pelo valor da função exponencial. Esta correção vai decaindo de valor de acordo com a função *cross-fading*. No caso da exponencial, como o valor da função tende à zero, não é necessário uma segunda transição para evitar que o valor da função *cross-fading* fique negativa.

Este caso mostra uma sincronização de relógios de métrica mista com o algoritmo AReS Exponencial. O algoritmo se mostrou eficiente, sendo que a principal vantagem é da possibilidade de modular a descontinuidade de tempo devido à sincronização via uma função exponencial e ainda melhorar a exatidão. Isto ocorre, devido à função cross-fading modular a correção do algoritmo ao longo de cada instante, diferente do algoritmo PTP que aplica toda a correção em um único instante. A desvantagem é que o algoritmo precisa de um gerenciamento de transições e necessita atualizar a cada instante o valor de tempo. No entanto, o tempo da transição é conhecido, devido ao *deadbeat*, o que minimiza esta desvantagem.

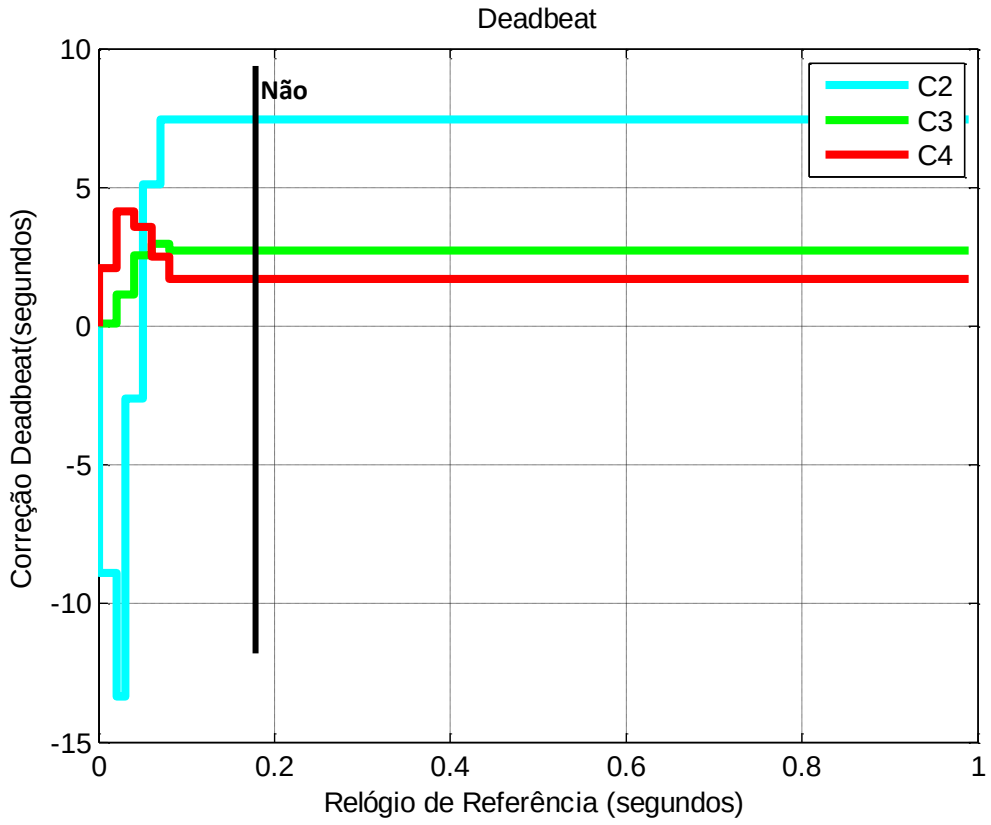


Figura 8.32 – Caso 7 - Correção STM - AReS Exponencial.

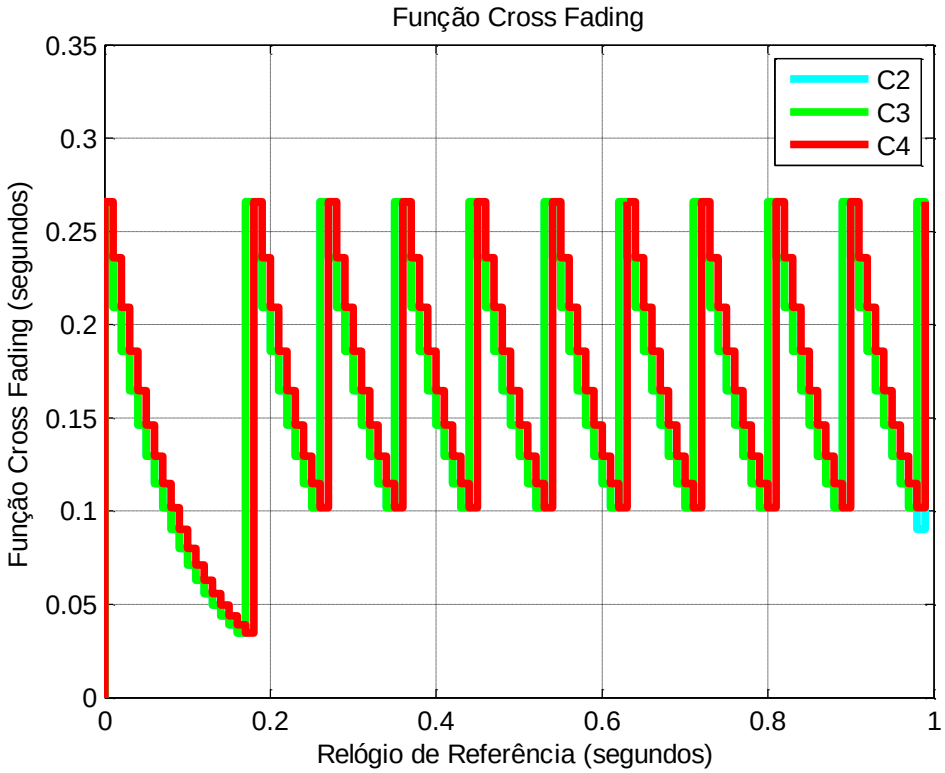


Figura 8.33 – Caso 7 - Função *Cross-Fading* Exponencial.

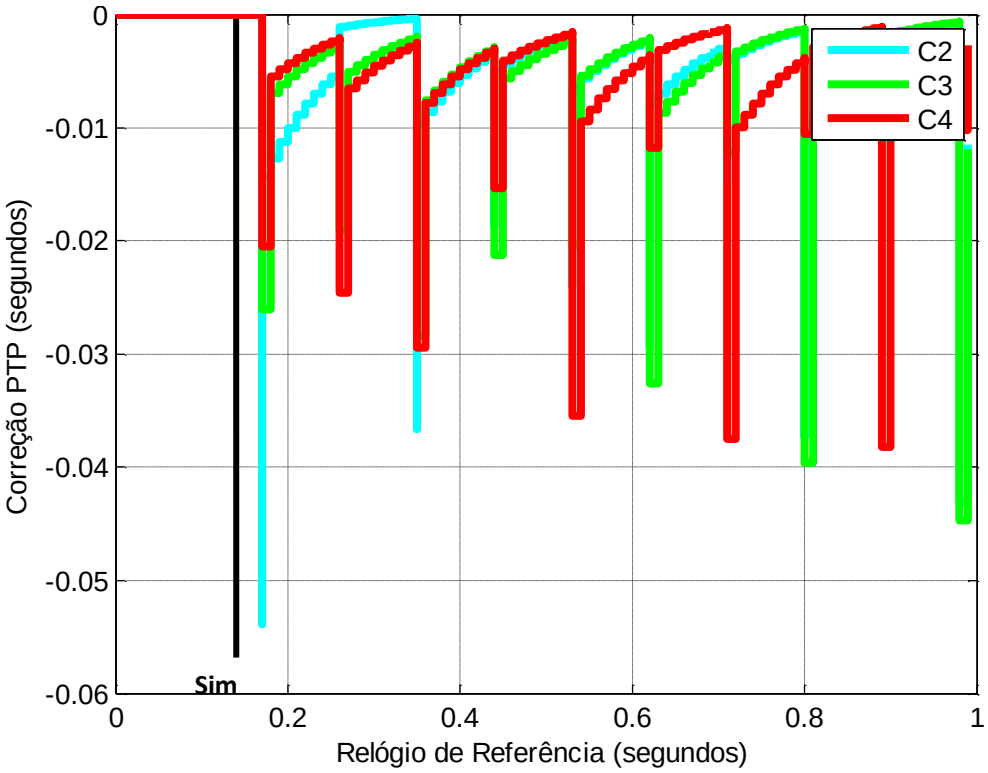


Figura 8.34 – Caso 7 - Correção NOMO - Algoritmo AReS - Exponencial.

8.10. Caso 8 - Algoritmo PTP - CSMA/CD

Este caso possui valores de viés inicial igual a: C1: 0; C2: 0.1 seg. ; C3: 0.01 seg. ; C4: -0.1 seg.; e deriva igual a: C1: 0; C2: 1%; C3: 0.01%; C4: -0.01%.

Neste caso, aplica-se o algoritmo PTP para corrigir o tempo sobre um sistema de controle por redes com uma rede CSMA/CD. A Figura 8.35 e Figura 8.36 mostram os valores de macrotick e diferença dos macroticks em relação ao relógio de referência (C1), respectivamente.

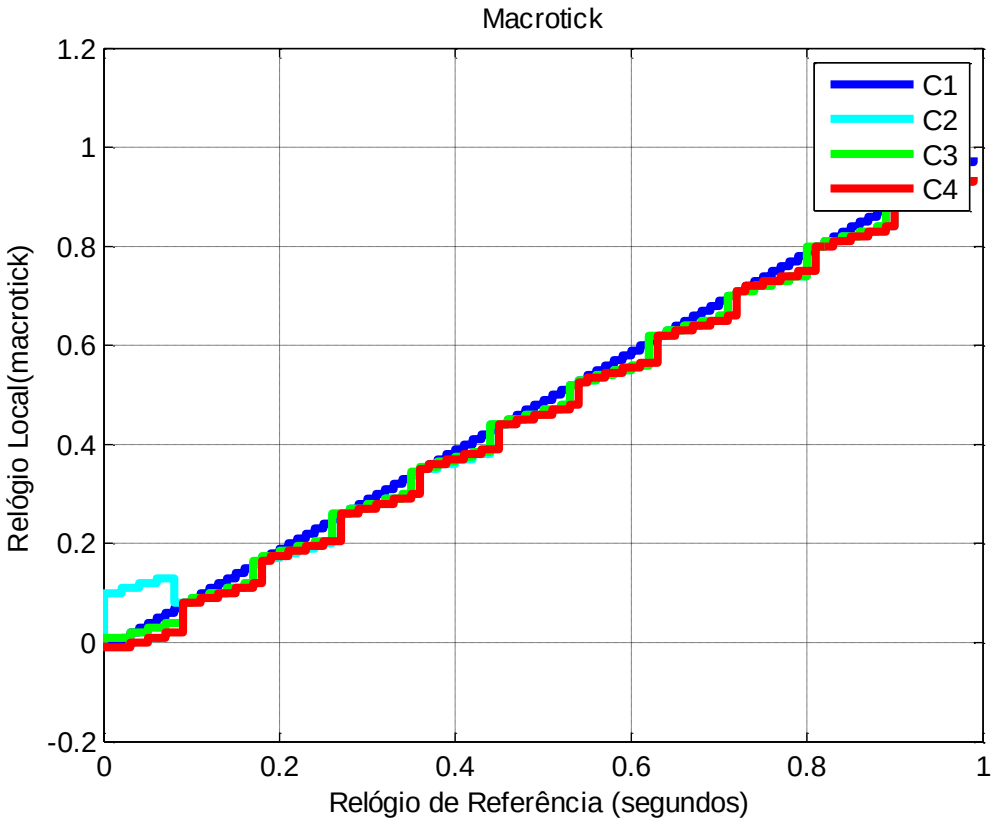


Figura 8.35 – Caso 8 - Macrotick.

A Figura 8.37 mostra o valor de correção do relógio calculado pela função de convergência do algoritmo PTP. O período de re-sincronização é de 10 microticks, ou seja, a cada 10 microticks aplica-se o processo de sincronização do algoritmo PTP. A Figura 8.38 e Figura 8.39 mostram a lei de controle PID e a resposta dinâmica do sistema de controle por rede em uma rede CSMA/CD.

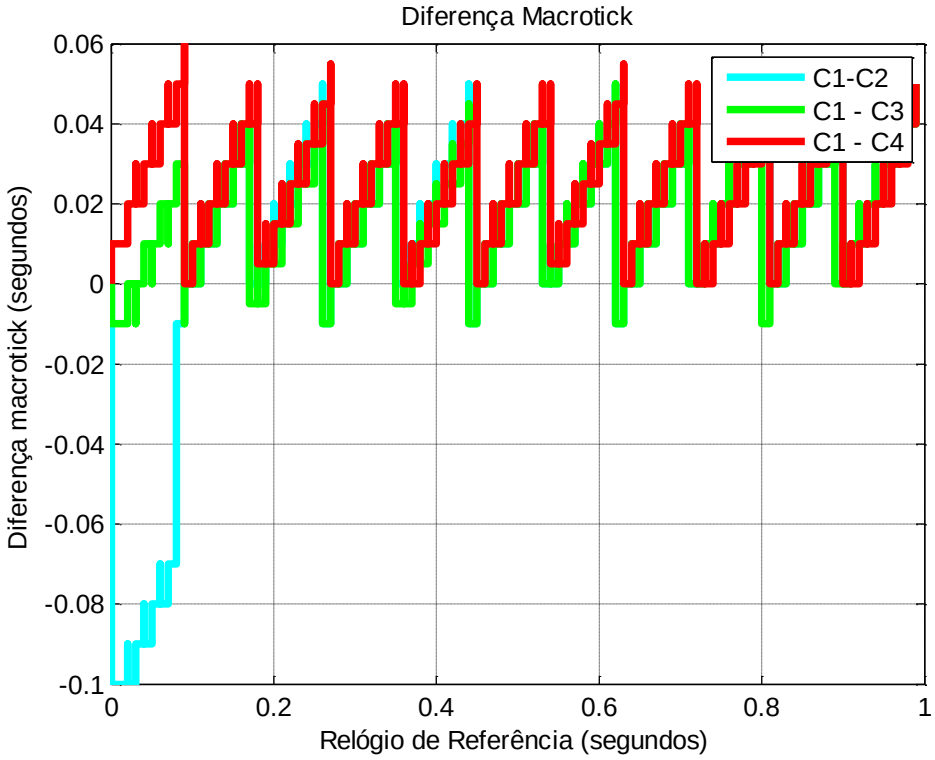


Figura 8.36 – Caso 8 - Diferença entre Macroticks.

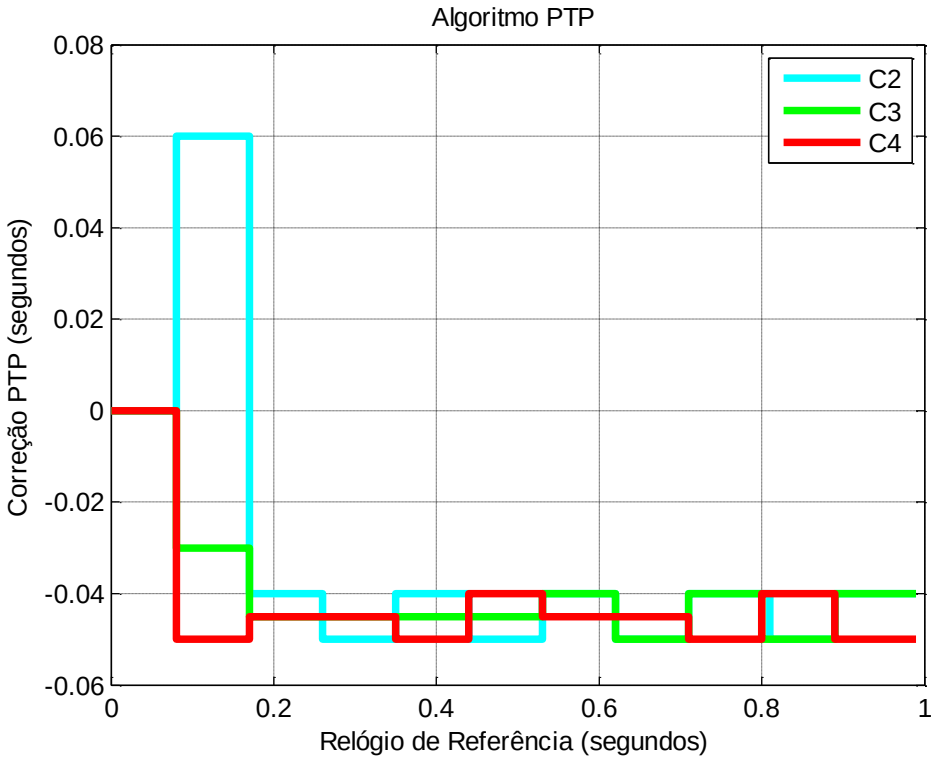


Figura 8.37 – Caso 8 - Correção Algoritmo PTP.

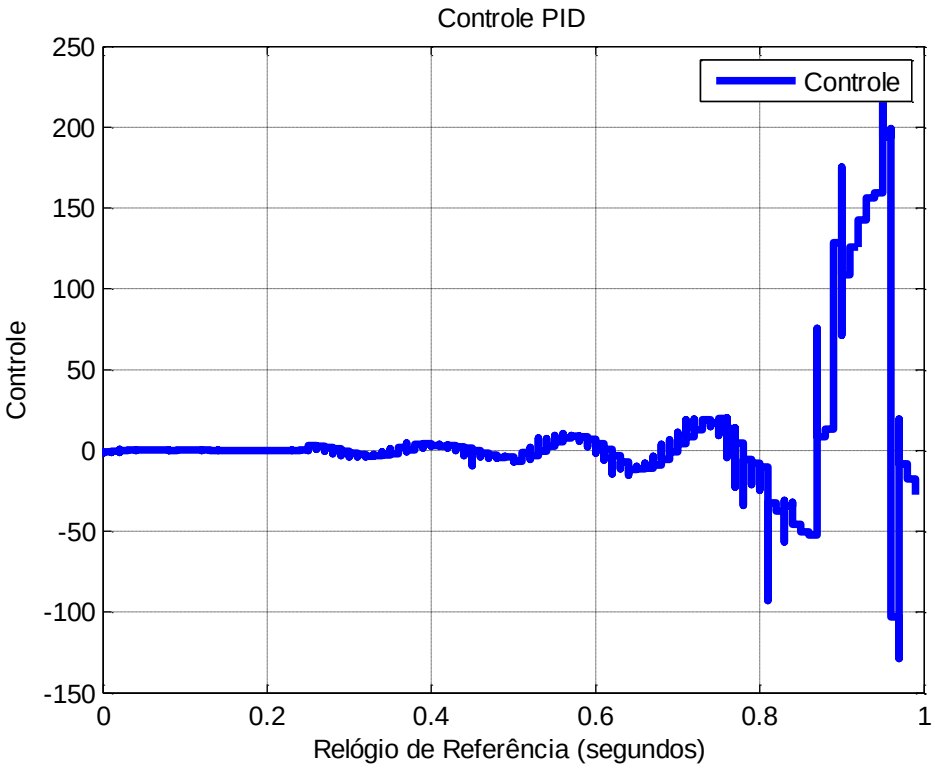


Figura 8.38 – Caso 8 - Controle PID.

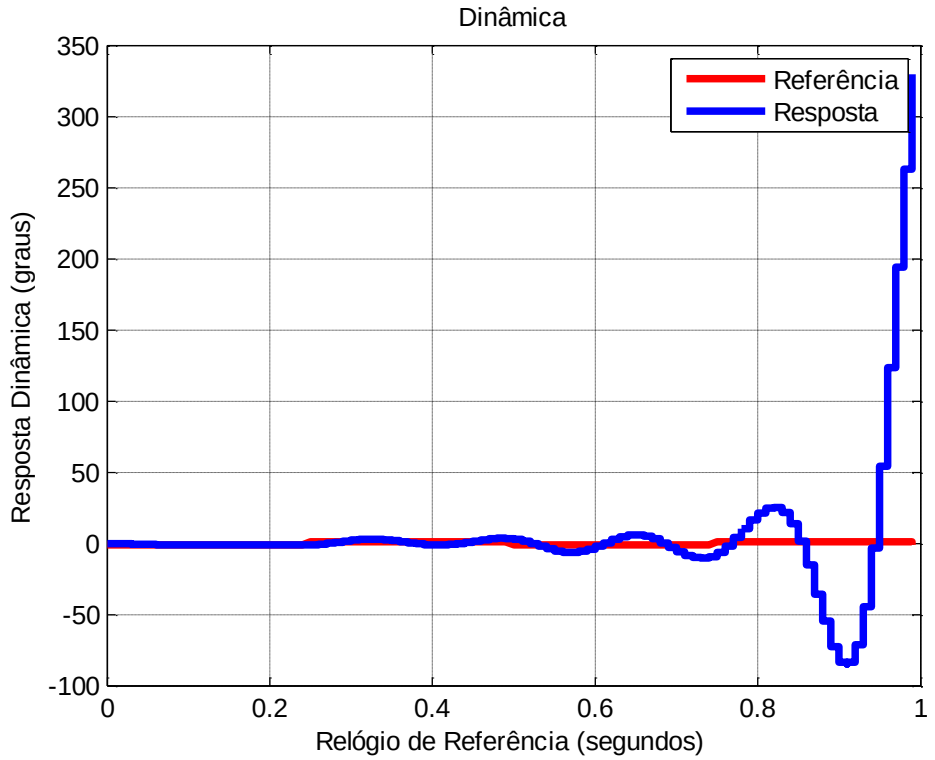


Figura 8.39 – Caso 8 - Resposta Dinâmica.

Este caso mostra uma sincronização de relógios com o algoritmo PTP sobre um sistema de controle por redes em uma rede CSMA/CD. O algoritmo de sincronização se mostrou eficiente para estabelecer a sincronização. No entanto, devido à grande excursão de tempo (descontinuidade de tempo) o algoritmo ao estabelecer a sincronização causa indiretamente à instabilidade do sistema de controle. Isto ocorre, pois o algoritmo PTP não foi desenvolvido para levar em conta o sistema de controle e a de-sincronização inicial.

8.11. Caso 9 - DeadBeat e Algoritmo PTP - CSMA/CD

Este caso possui valores de viés inicial igual a: C1: 0; C2: 0.1 seg. ; C3: 0.01 seg. ; C4: -0.1 seg.; e deriva igual a: C1: 0; C2: 1%; C3: 0.01%; C4: -0.01%. Este caso aplica-se um algoritmo com os dois modos de correção: 1) STM e o 2) NOMO, descritos no capítulo 6. Com isto, visa-se inicialmente corrigir o viés inicial com o *deadbeat* e depois com o algoritmo PTP sobre um sistema de controle por redes sobre uma rede CSMA/CD. A Figura 8.40 e Figura 8.41 mostram, respectivamente, os valores de macrotick e diferença entre o relógio de referência e os macroticks.

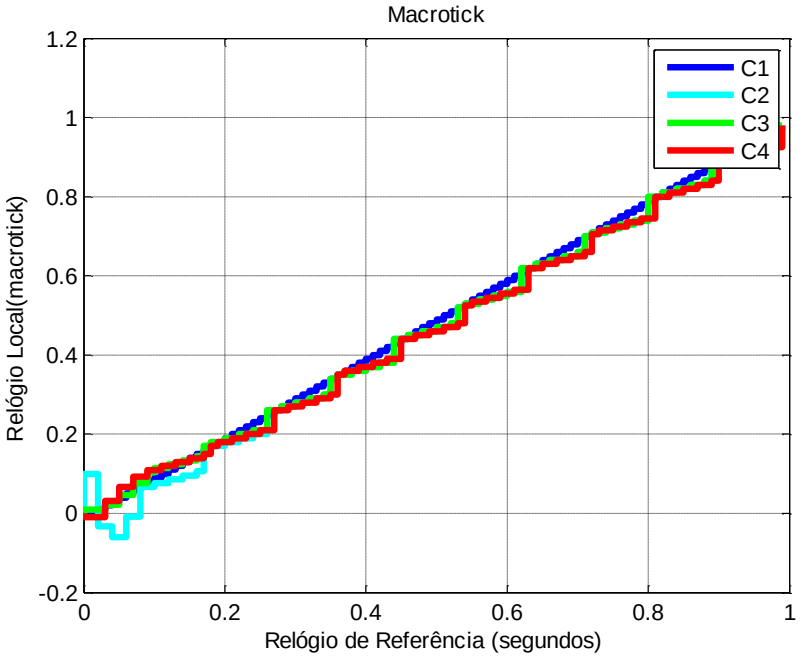


Figura 8.40 – Caso 9 - Macrotick.

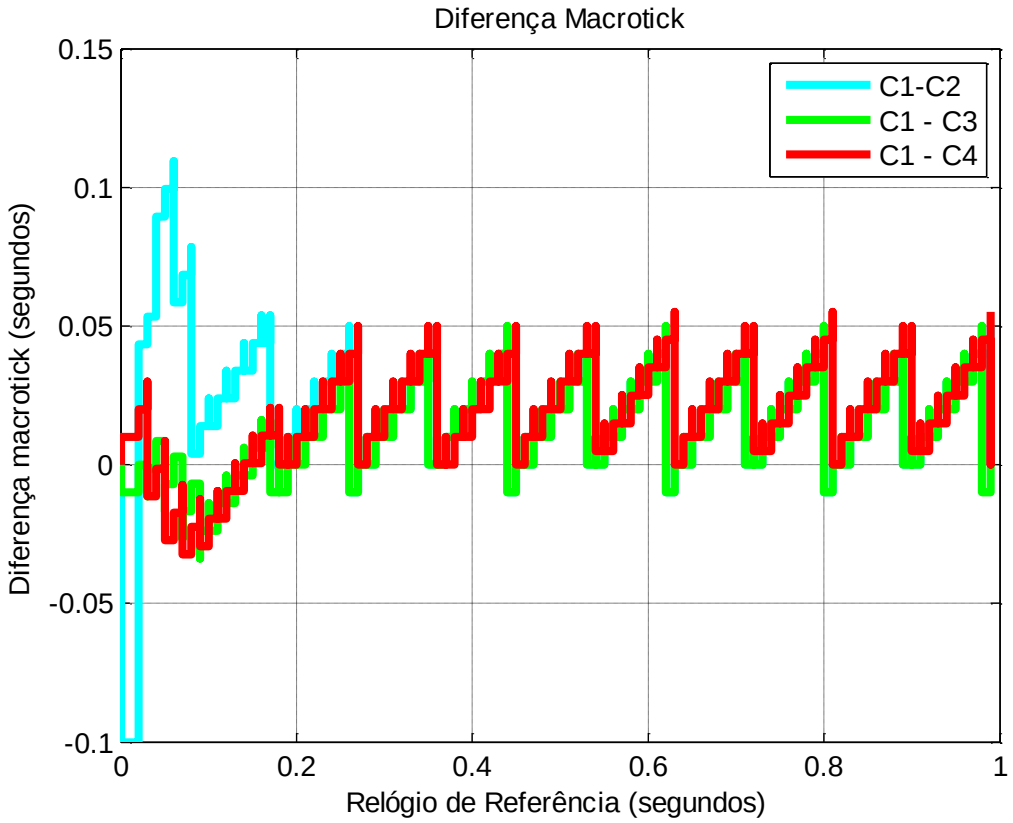


Figura 8.41 – Caso 9 - Diferença entre Macroticks.

A Figura 8.42 mostra o valor de correção do relógio calculado pelo controlador *deadbeat* no modo STM e a Figura 8.43 a correção calculada pela função de convergência do algoritmo PTP no modo NOMO. O tempo de transição dos modos de correção é de 10 macroticks e o período de re-sincronização é de 10 microticks, ou seja, após 10 macroticks, a correção passa a ser feita pelo PTP que a cada 10 microticks aplica-se o processo de sincronização do algoritmo.

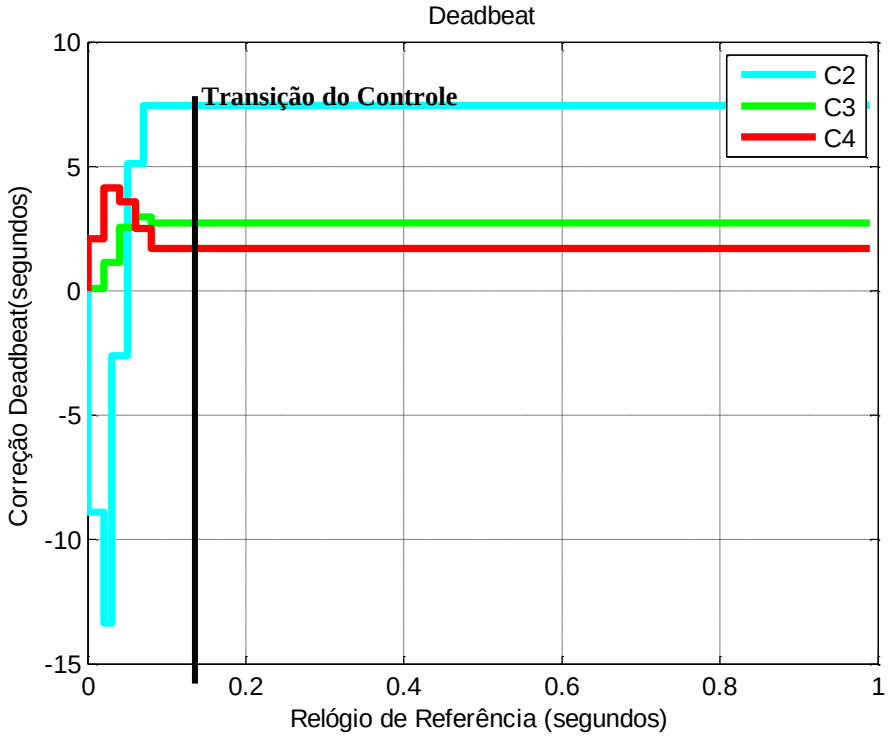


Figura 8.42 – Caso 9 - Modo STM - Controle *Deadbeat*.

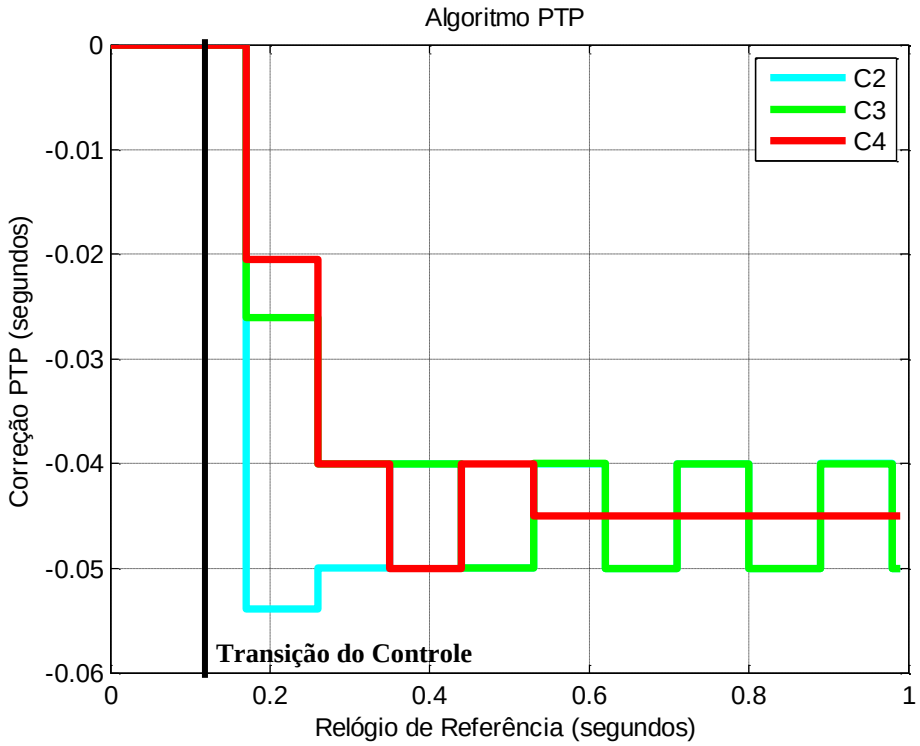


Figura 8.43 – Caso 9 - Correção modo NOMO - Algoritmo PTP.

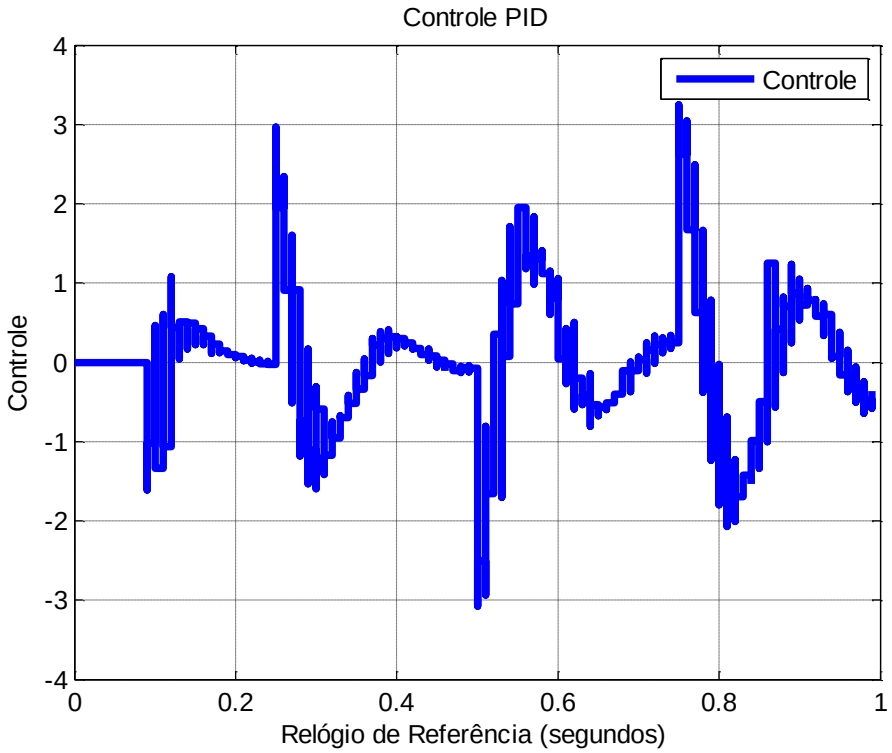


Figura 8.44 – Caso 9 - Controle PID.

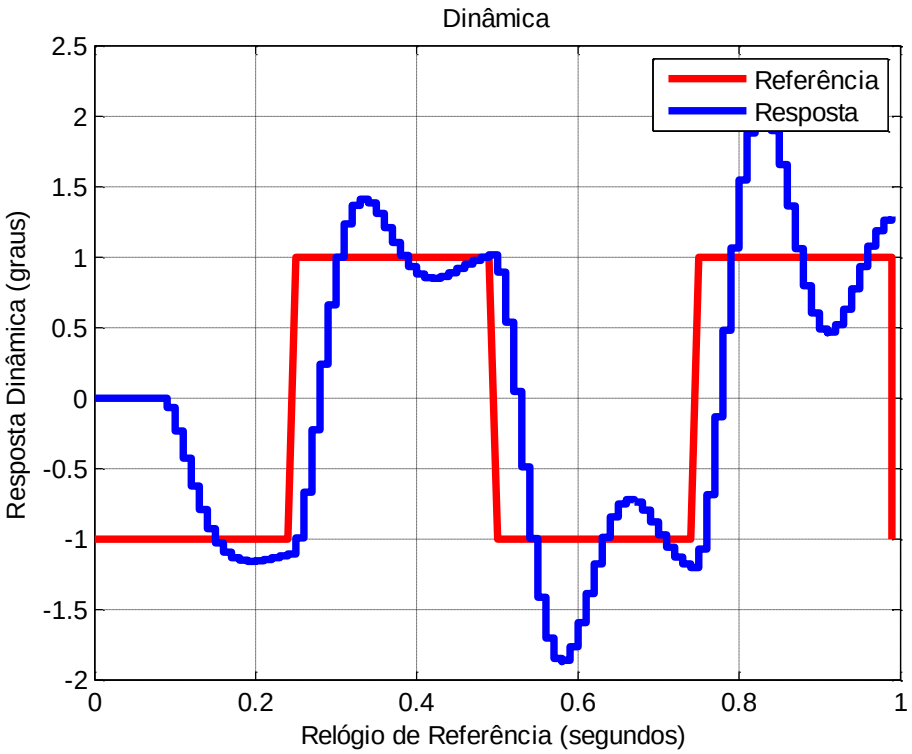


Figura 8.45 – Caso 9 - Resposta Dinâmica.

Este caso mostra uma sincronização de relógios com o controlador *deadbeat* no modo STM e o algoritmo PTP no modo NOMO, sobre um sistema de controle por redes em uma rede CSMA/CD. No modo STM, não é realizado o controle da dinâmica, pois é um modo de inicialização de tempo. No modo NOMO o algoritmo de sincronização se mostrou eficiente para estabelecer a sincronização. No entanto, devido à sincronização de tempo a resposta dinâmica apresentou um sobressinal alto (de 50 a 100%). No sistema de controle. Isto ocorre, pois o algoritmo PTP não foi desenvolvido para levar em conta o sistema de controle. A de-sincronização inicial que antes levou o algoritmo a gerar uma instabilidade no sistema de controle, agora foi corrigida aplicando a reconfiguração dos algoritmos de tempo e utilizando o *deadbeat*. Esta técnica se mostrou eficiente e pode ser melhorada através da otimização dos parâmetros.

8.12. Caso 10 - Algoritmo AReS Degrau - CSMA/CD

Este caso possui valores de viés inicial igual a: C1: 0; C2: 0.1 seg. ; C3: 0.01 seg. ; C4: -0.1 seg.; e deriva igual a: C1: 0; C2: 1%; C3: 0.01%; C4: -0.01%.

Neste caso, aplica-se um algoritmo com os dois modos de correção: 1) STM e o 2) NOMO, descritos no capítulo 6. Com isto, visa-se inicialmente corrigir o viés inicial com o controlador *deadbeat* e depois utiliza-se a técnica PTP em conjunto com uma função *cross-fading* degrau, definindo o algoritmo AReS degrau, sobre um sistema de controle por redes em uma rede CSMA/CD.

A Figura 8.46 e Figura 8.47 mostram, respectivamente, os valores de macrotick e diferença entre o relógio de referência e os macroticks. A Figura 8.48 mostra o valor de correção do relógio calculado pelo controlador *deadbeat* no modo STM. A Figura 8.49 mostra a função *cross-fading* degrau.

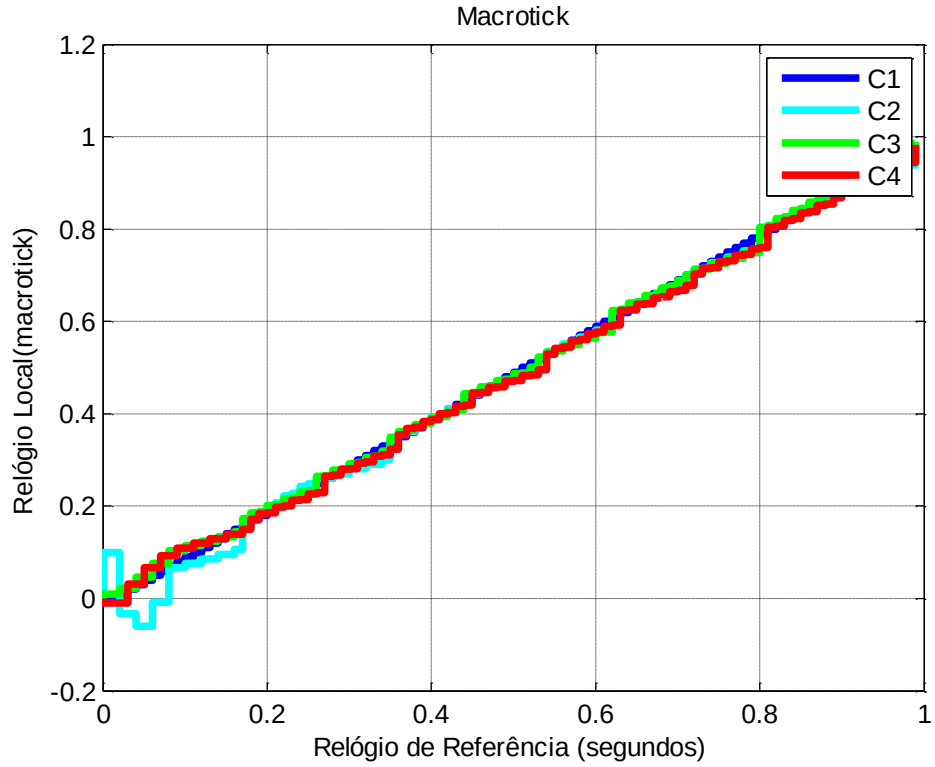


Figura 8.46 – Caso 10 - Macrotick.

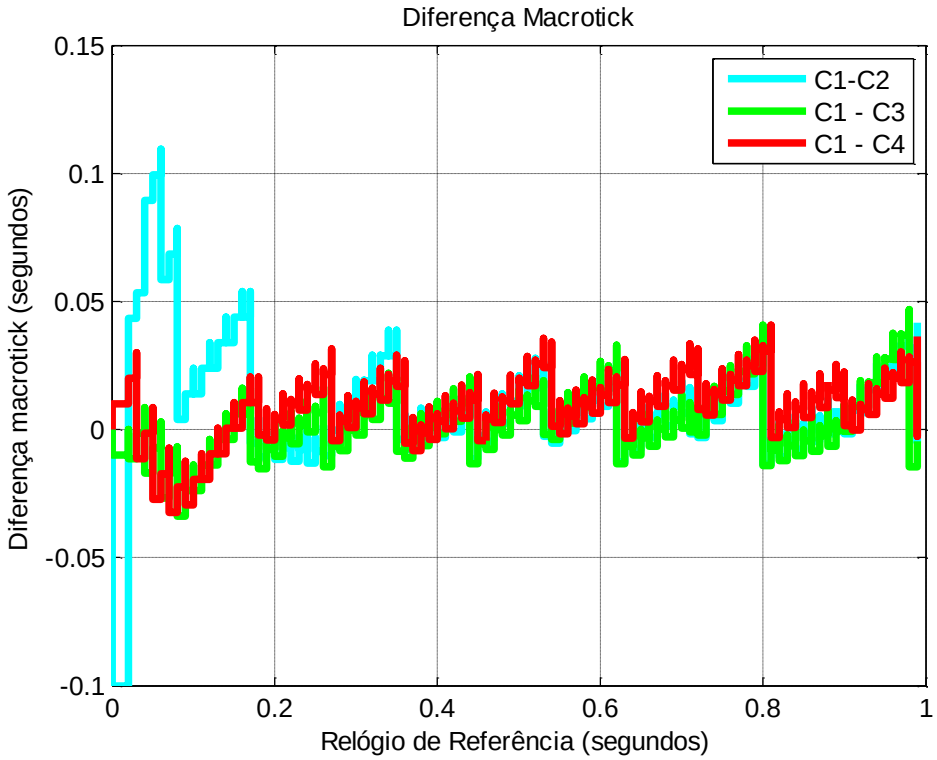


Figura 8.47 – Caso 10 - Diferença entre Macroticks.

A Figura 8.50 mostra a correção calculada pela função de convergência do algoritmo PTP mais a função *cross-fading* degrau. O tempo de transição entre os modos de correção é de 10 macroticks (existe uma transição entre o STM e NOMO após 10 macroticks) e o período de re-sincronização é de 10 microticks, ou seja, após 10 macroticks, a correção é recalculada. No entanto, com o uso da função degrau, a cada instante é aplicada uma correção multiplicada pelo valor da função *cross-fading*.

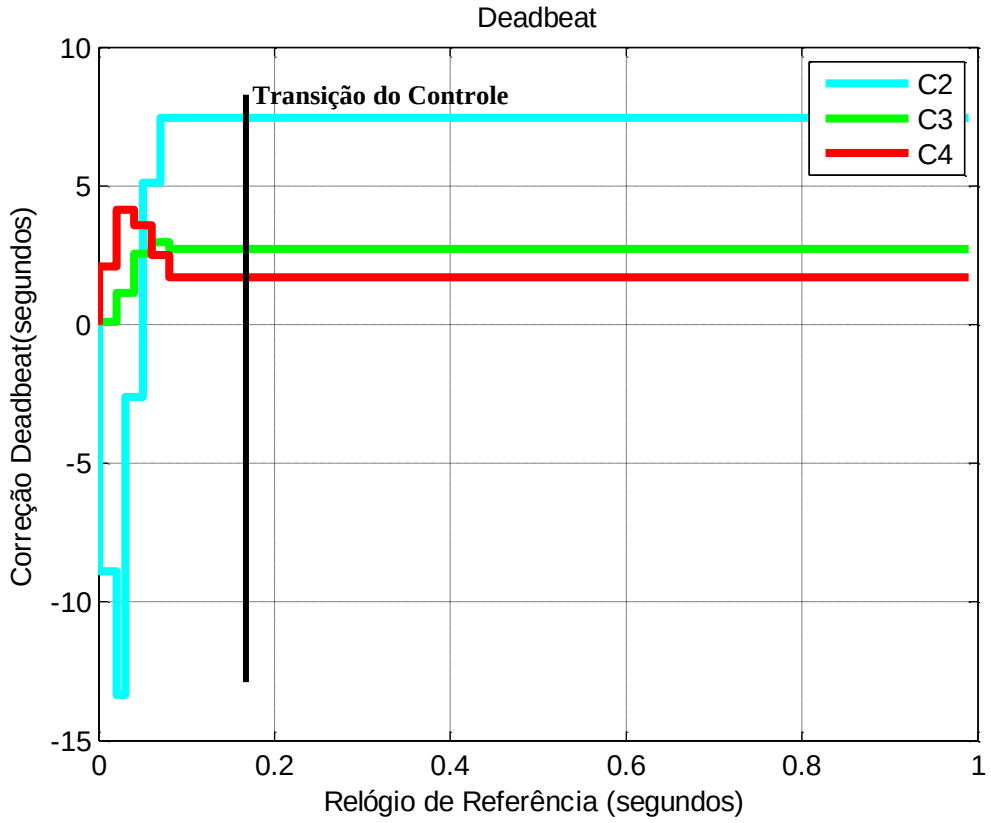


Figura 8.48 – Caso 10 - Modo STM - Algoritmo AReS Degrau.

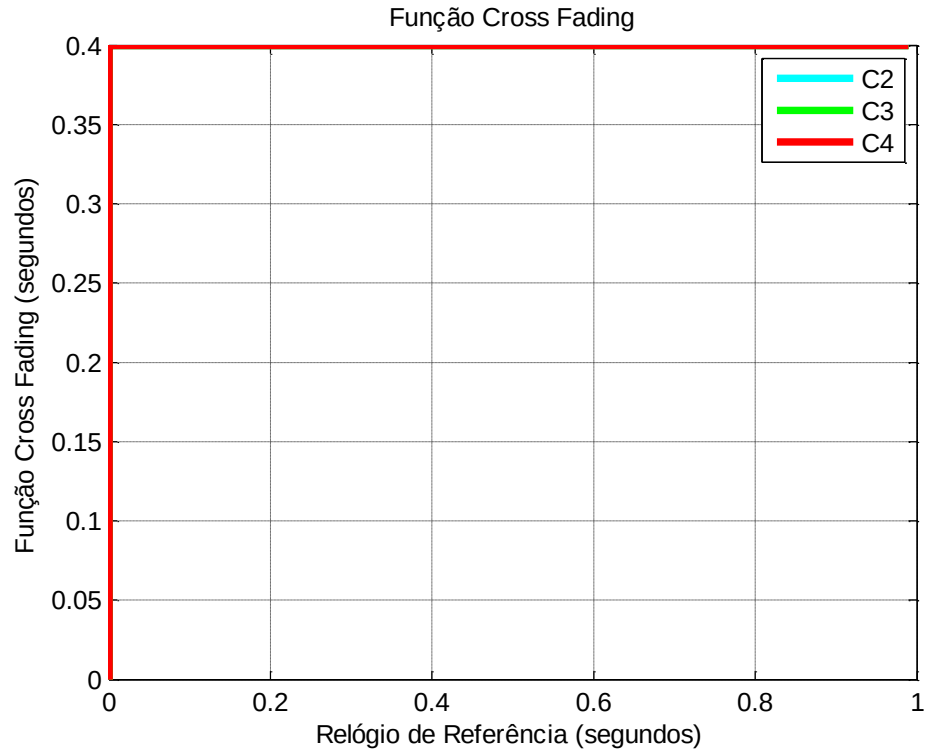


Figura 8.49 – Caso 10 - Função Cross-Fading Degrau.

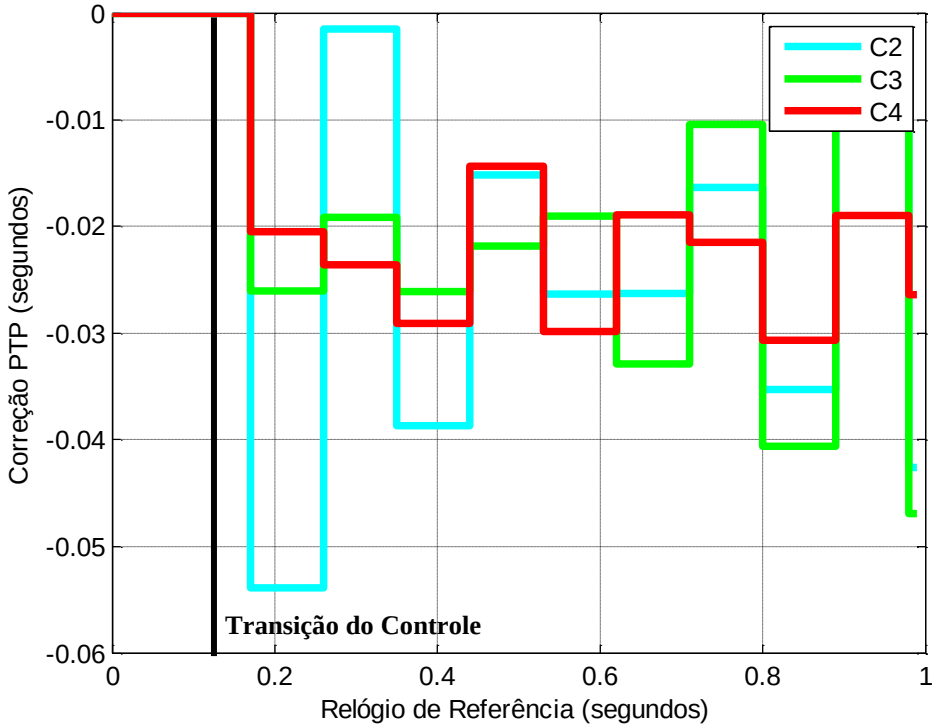


Figura 8.50 – Caso 10 - Correção modo NOMO - Algoritmo AReS Degrau.

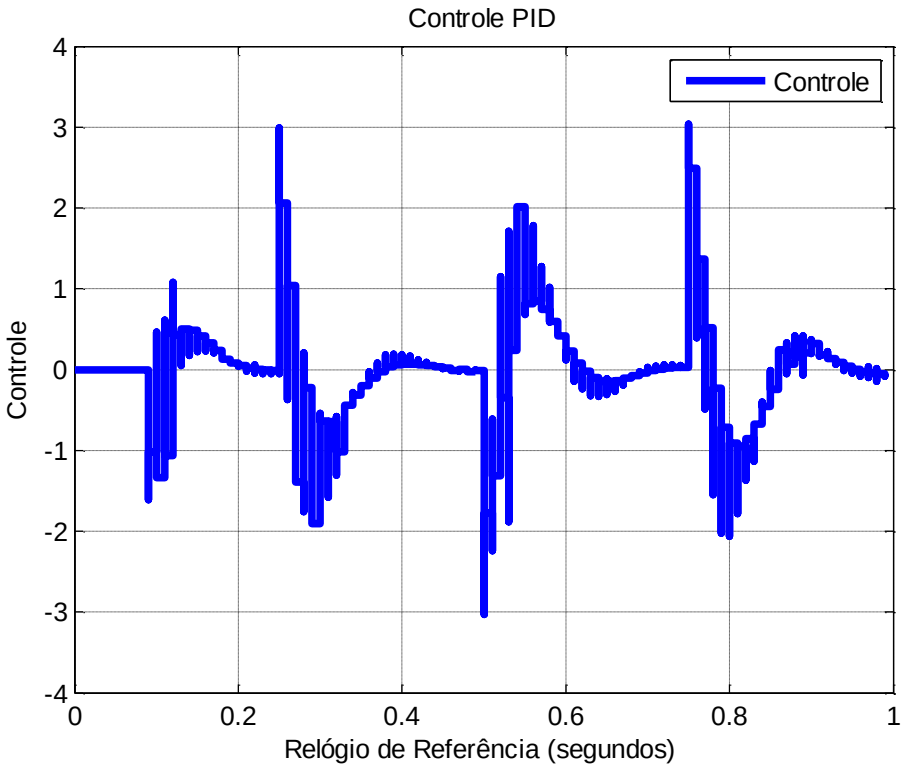


Figura 8.51 – Caso 10 - Controle PID.

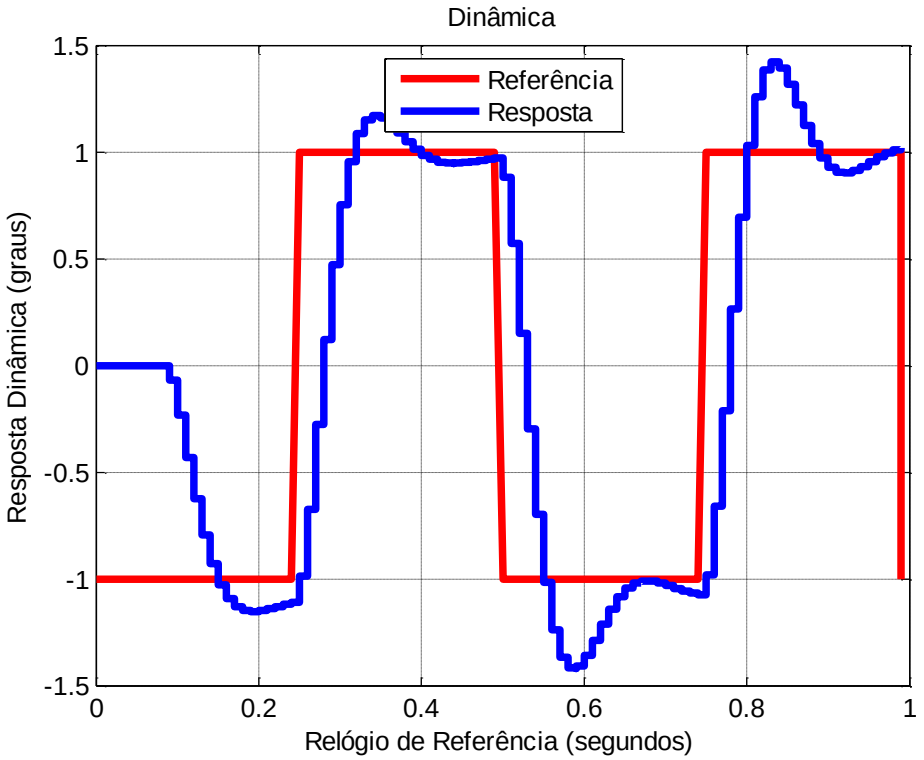


Figura 8.52 – Caso 10 - Resposta Dinâmica.

Este caso mostra uma sincronização de relógios com o algoritmo AReS degrau que utiliza um controlador *deadbeat* no modo STM e o algoritmo PTP com *cross-fading* degrau no modo NOMO, sobre um sistema de controle por redes em uma rede CSMA/CD. No modo STM, não é realizado o controle da dinâmica, pois é um modo de inicialização de tempo. No modo NOMO o algoritmo de sincronização se mostrou eficiente para estabelecer a sincronização. No entanto, devido à sincronização de tempo a resposta dinâmica apresentou um sobressinal melhor que o caso anterior no sistema de controle. Isto ocorre, pois a de-sincronização inicial que antes levou o algoritmo a gerar uma instabilidade no sistema de controle, agora foi corrigida aplicando a reconfiguração dos algoritmos de tempo e utilizando o *deadbeat*; e o método suaviza a descontinuidade de tempo. Esta técnica se mostrou eficiente para sincronizar relógios em sistemas de controle por redes e pode ser melhorada através da otimização dos parâmetros.

8.13. Caso 11 - Algoritmo AReS Rampa - CSMA/CD

Este caso possui valores de viés inicial igual a: C1: 0; C2: 0.1 seg. ; C3: 0.01 seg. ; C4: -0.1 seg.; e deriva igual a: C1: 0; C2: 1%; C3: 0.01%; C4: -0.01%.

Neste caso, aplica-se um algoritmo com os dois modos de correção: 1) STM e o 2) NOMO, descritos no capítulo 6. Com isto, visa-se inicialmente corrigir o viés inicial com o controlador *deadbeat* e depois utiliza-se a técnica PTP em conjunto com uma função *cross-fading* rampa, definindo o algoritmo AReS rampa, sobre um sistema de controle por redes em uma rede CSMA/CD.

A Figura 8.53 e Figura 8.54 mostram, respectivamente, os valores de macrotick e diferença entre o relógio de referência e os macroticks. A Figura 8.55 mostra o valor de correção do relógio calculado pelo controlador *deadbeat* no modo STM. A Figura 8.56 mostra a função *cross-fading* degrau.

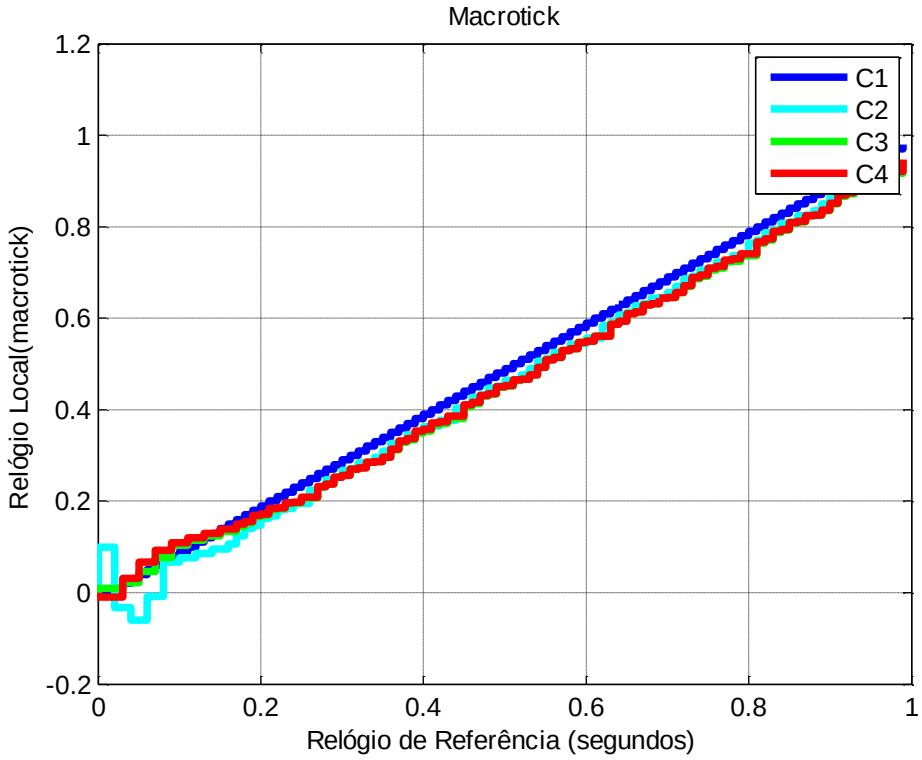


Figura 8.53 – Caso 11 - Macrotick.

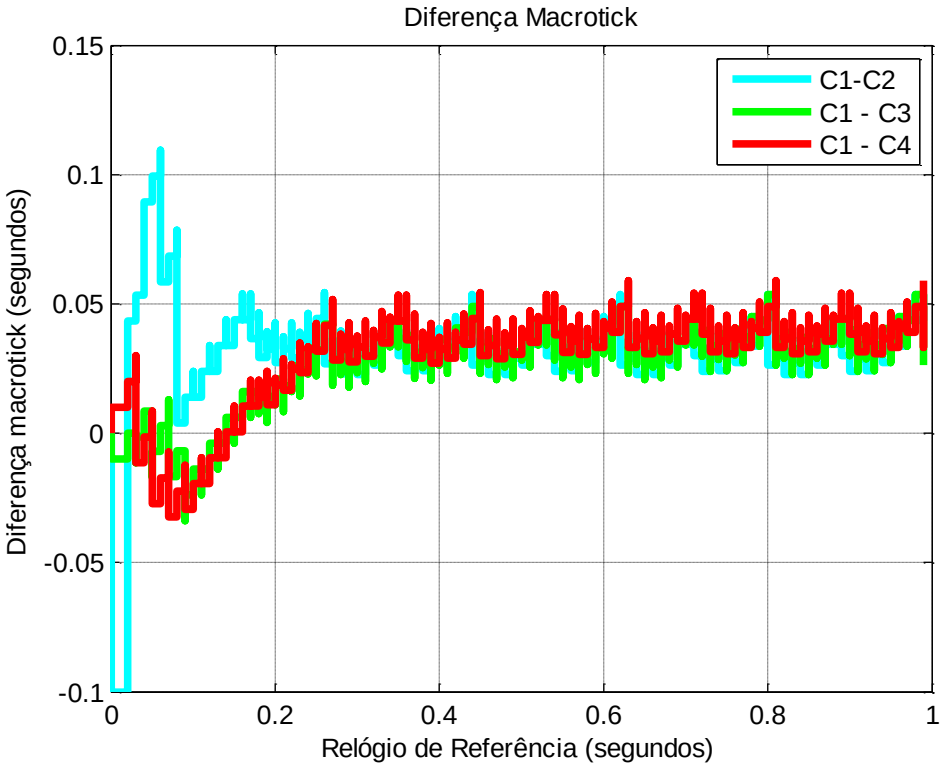


Figura 8.54 – Caso 11 - Diferença entre Macroticks.

A Figura 8.57 mostra a correção calculada pela função de convergência do algoritmo PTP mais a função *cross-fading* rampa. O tempo de transição entre os modos de correção é de 10 macroticks (existe uma transição entre o STM e NOMO após 10 macroticks) e o período de re-sincronização é de 10 microticks, ou seja, após 10 macroticks, a correção é recalculada. No entanto, com o uso da função rampa, a cada instante é aplicada uma correção multiplicada pelo valor da função *cross-fading*. No caso da reta, é necessário uma segunda transição para evitar que o valor da função *cross-fading* rampa fique negativa e inverta a fase do sistema tornando-o instável.

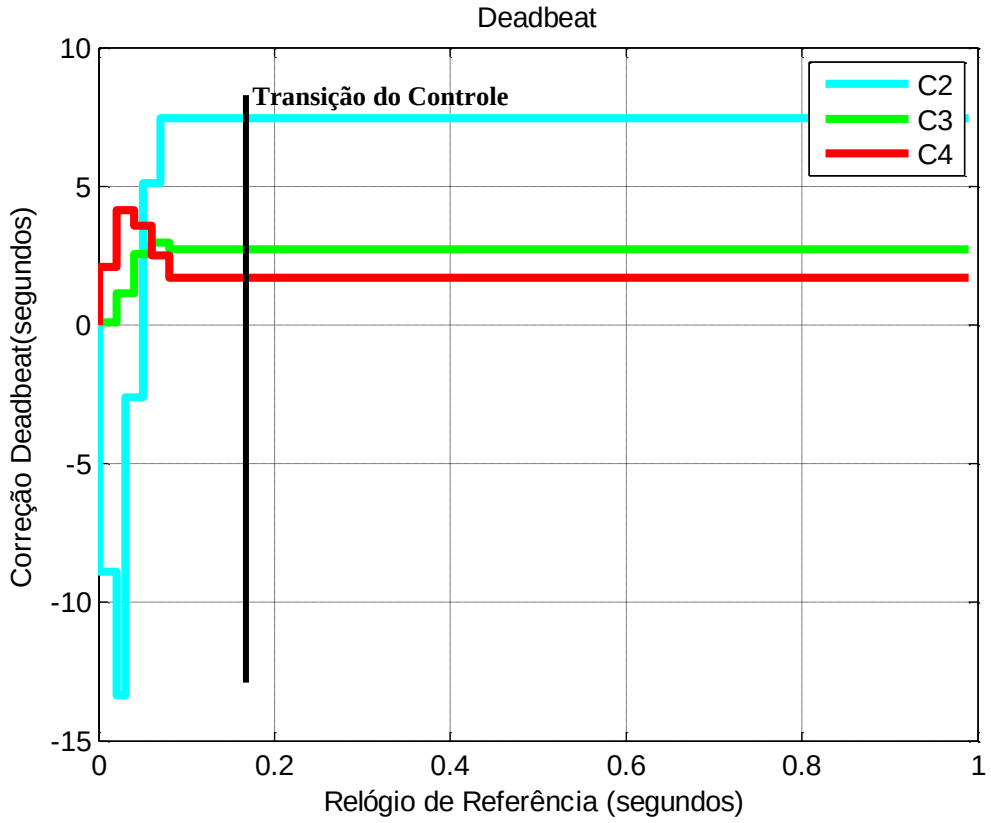


Figura 8.55 – Caso 11 - Modo STM - Algoritmo AReS Rampa.

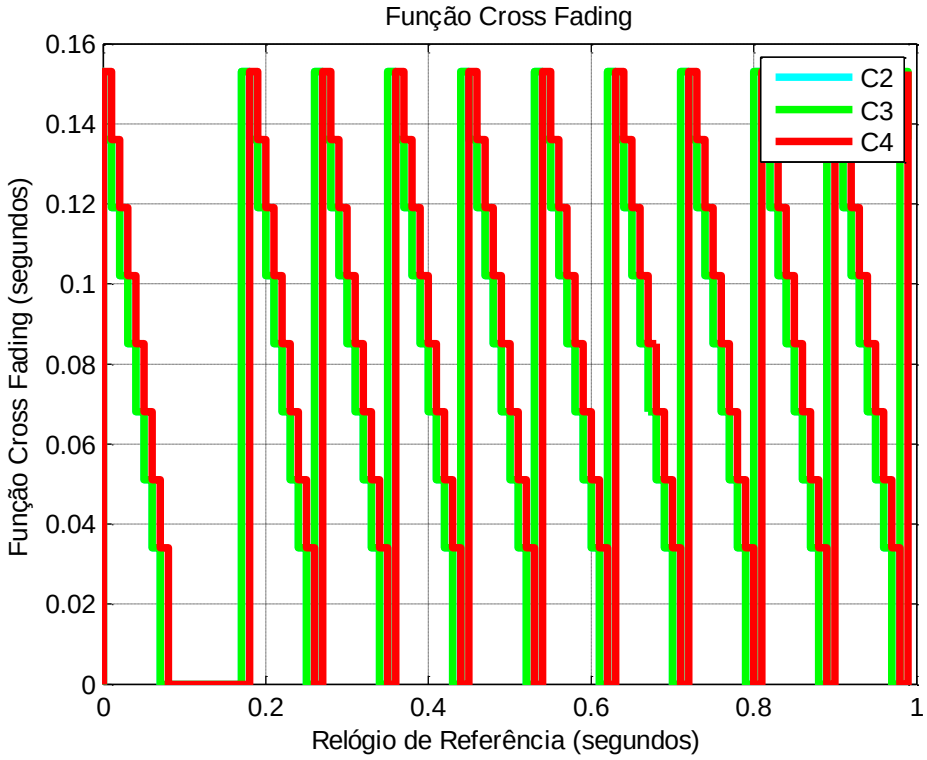


Figura 8.56 – Caso 11 - Função Cross-Fading Rampa.

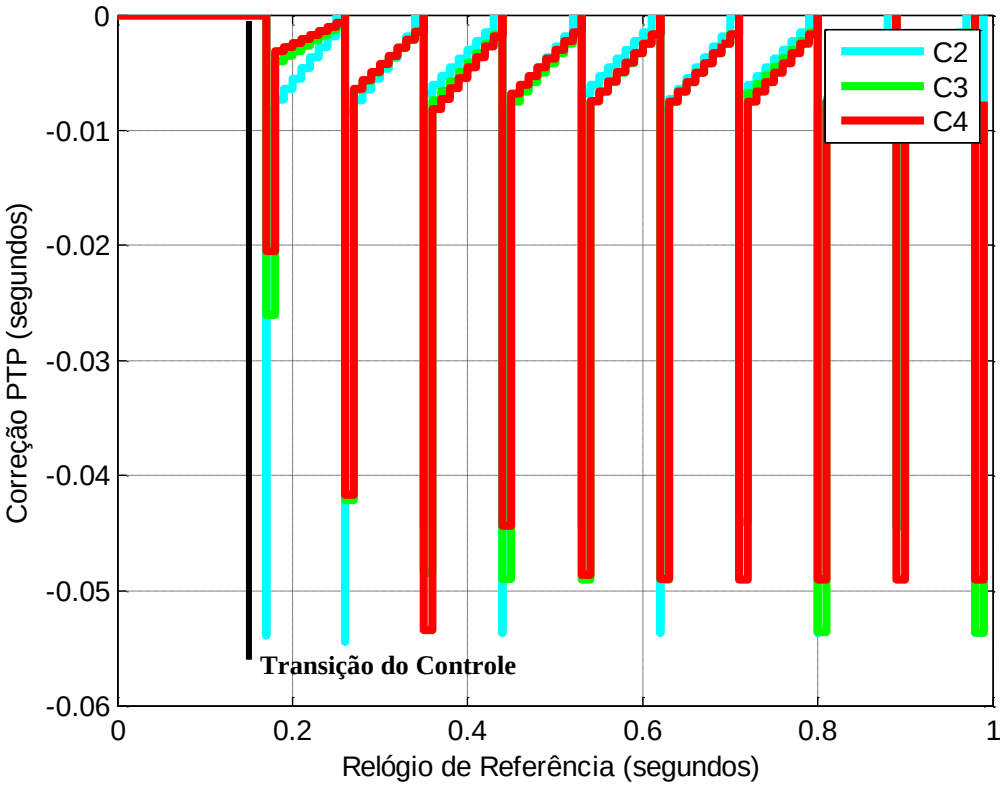


Figura 8.57 – Caso 11 - Correção modo NOMO - Algoritmo AReS Rampa.

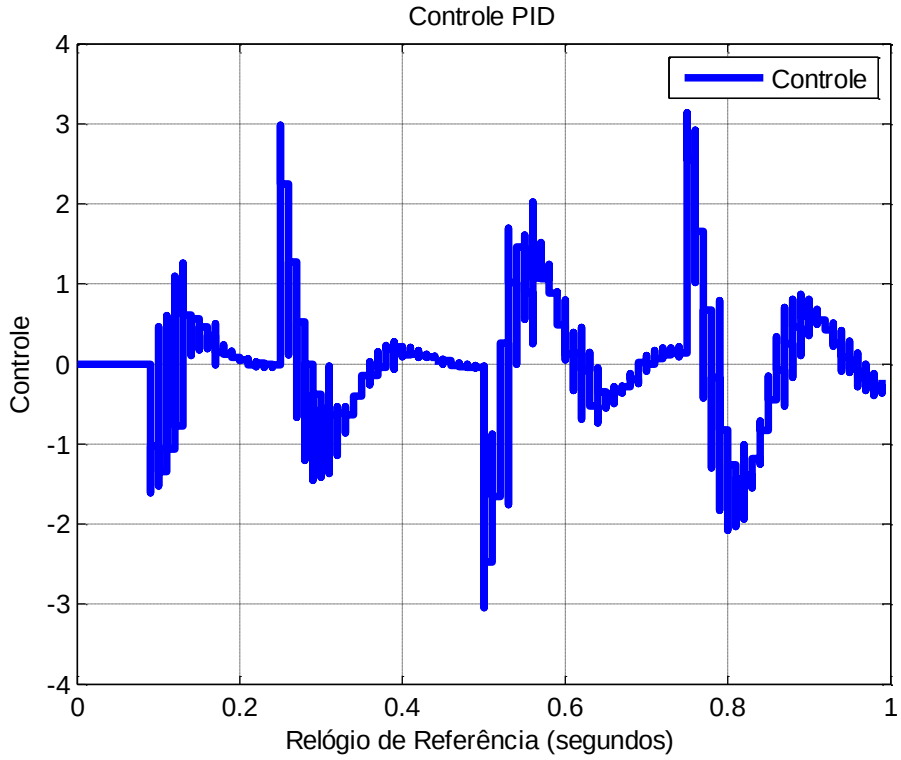


Figura 8.58 – Caso 11 - Controle PID.

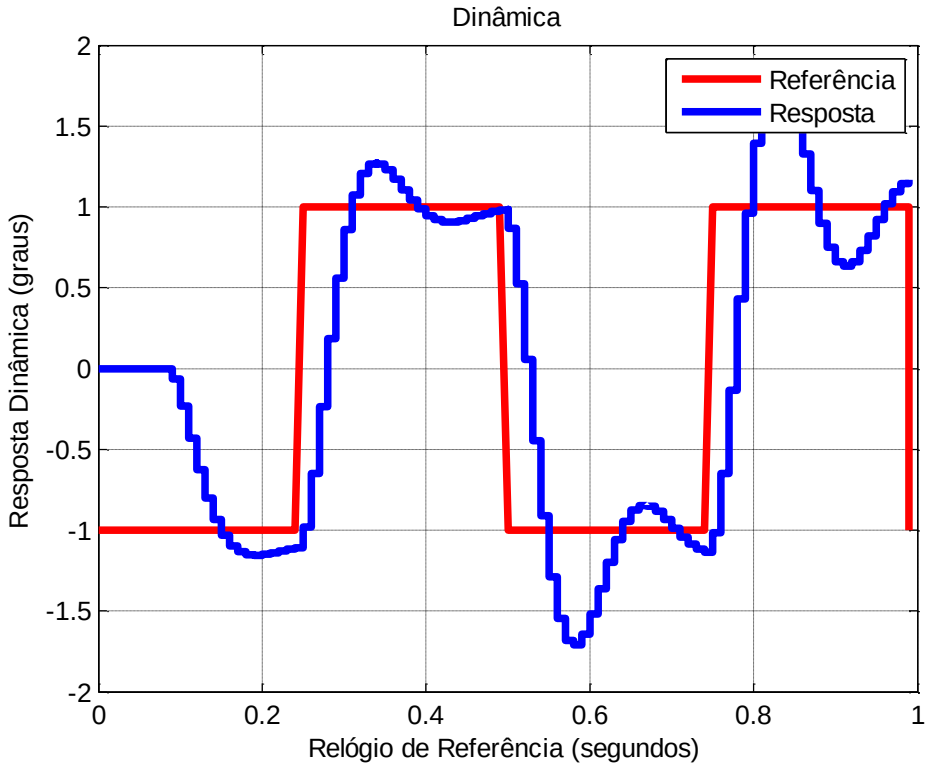


Figura 8.59 – Caso 11 - Resposta Dinâmica.

Este caso mostra uma sincronização de relógios com o algoritmo AReS rampa que utiliza um controlador *deadbeat* no modo STM e o algoritmo PTP com *cross-fading* rampa no modo NOMO, sobre um sistema de controle por redes em uma rede CSMA/CD. No modo STM, não é realizado o controle da dinâmica, pois é um modo de inicialização de tempo. No modo NOMO o algoritmo de sincronização se mostrou eficiente para estabelecer a sincronização. No entanto, devido à sincronização de tempo a resposta dinâmica apresentou um sobressinal. Isto ocorre, pois o método suaviza a descontinuidade de tempo. A de-sincronização inicial que antes levou o algoritmo a gerar uma instabilidade no sistema de controle, agora foi corrigida aplicando a reconfiguração dos algoritmos de tempo e utilizando o *deadbeat*. Esta técnica se mostrou eficiente para sincronizar relógios em sistemas de controle por redes e pode ser melhorada através da otimização dos parâmetros.

8.14. Caso 12 - Algoritmo AReS Exponencial - CSMA/CD

Este caso possui valores de viés inicial igual a: C1: 0; C2: 0.1 seg. ; C3: 0.01 seg. ; C4: -0.1 seg.; e deriva igual a: C1: 0; C2: 1%; C3: 0.01%; C4: -0.01%.

Neste caso, aplica-se um algoritmo com os dois modos de correção: 1) STM e o 2) NOMO, descritos no capítulo 6. Com isto, visa-se inicialmente corrigir o viés inicial com o controlador *deadbeat* e depois utiliza-se a técnica PTP em conjunto com uma função *cross-fading* exponencial, definindo o algoritmo AReS exponencial, sobre um sistema de controle por redes em uma rede CSMA/CD.

A Figura 8.60 e Figura 8.61 mostram, respectivamente, os valores de macrotick e diferença entre o relógio de referência e os macroticks. A Figura 8.62 mostra o valor de correção do relógio calculado pelo controlador *deadbeat* no modo STM. A Figura 8.63 mostra a função *cross-fading* exponencial.

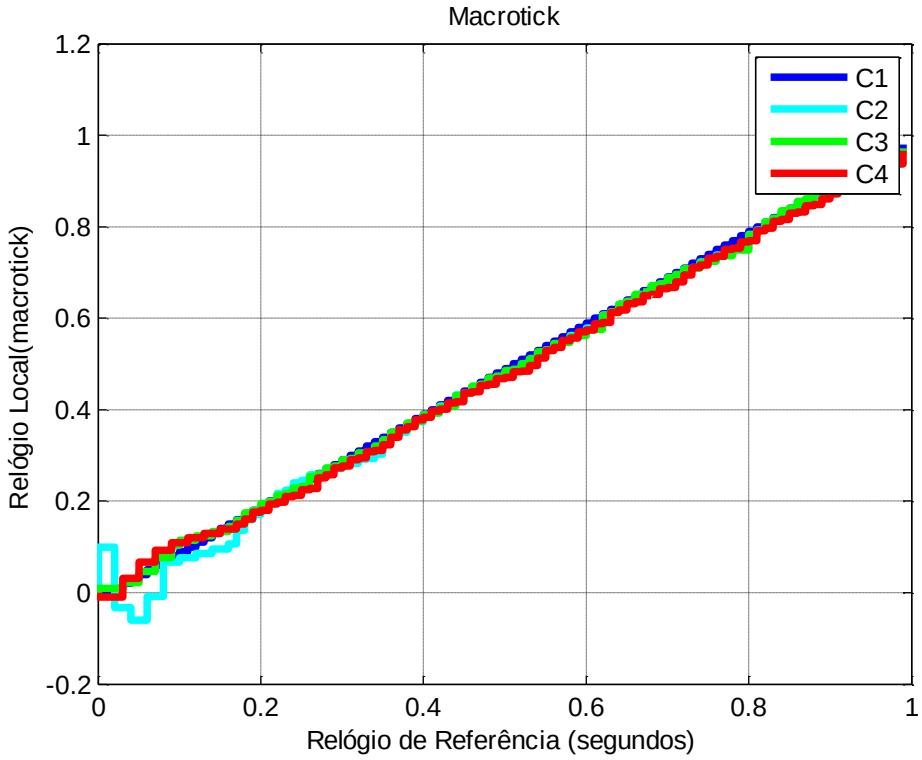


Figura 8.60 – Caso 12 - Macrotick.

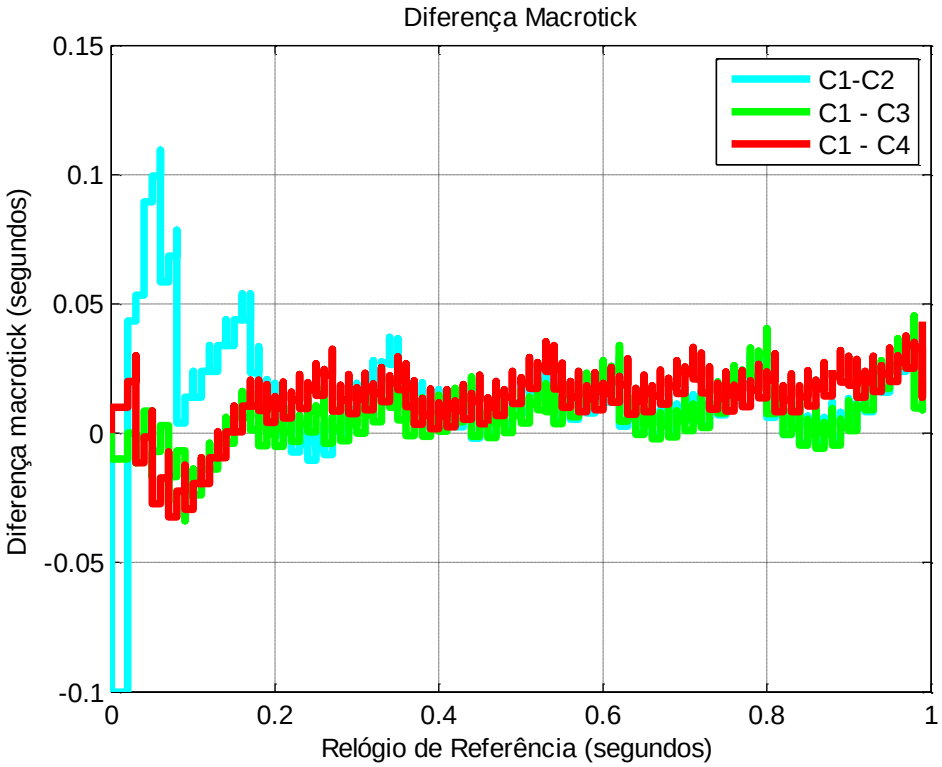


Figura 8.61 – Caso 12 - Diferença entre Macroticks.

A Figura 8.64 mostra a correção calculada pela função de convergência do algoritmo PTP mais a função *cross-fading* exponencial. O tempo de transição entre os modos de correção é de 10 macroticks (existe uma transição entre o STM e NOMO após 10 macroticks) e o período de re-sincronização é de 10 microticks, ou seja, após 10 macroticks, a correção é recalculada. No entanto, com o uso da função exponencial, a cada instante é aplicada uma correção multiplicada pelo valor da função *cross-fading*. No caso da exponencial, não é necessário uma segunda transição para evitar que o valor da função *cross-fading* exponencial fique negativa.

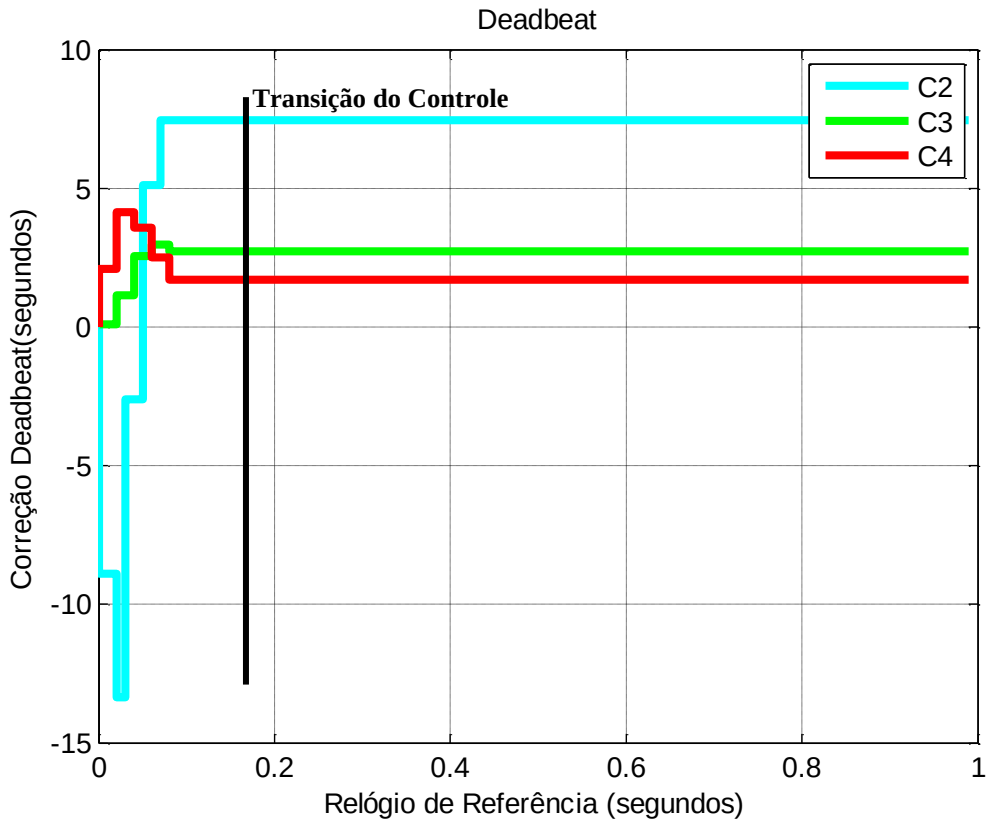


Figura 8.62 – Caso 12 - Modo STM - Algoritmo AReS Exponencial.

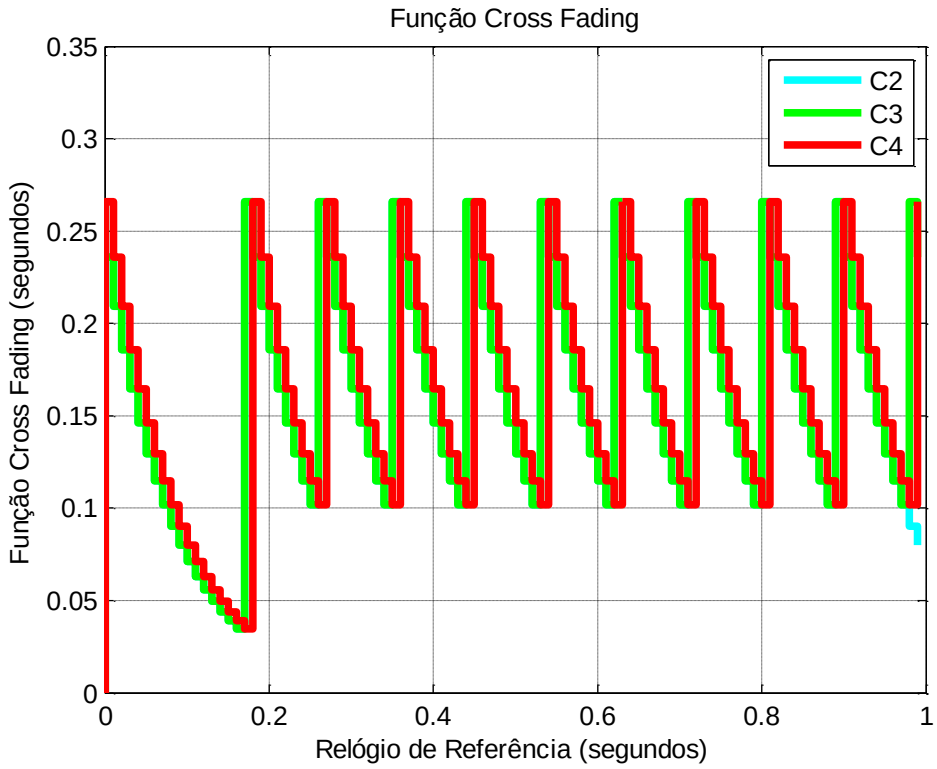


Figura 8.63 – Caso 12 - Função Cross-Fading Rampa.

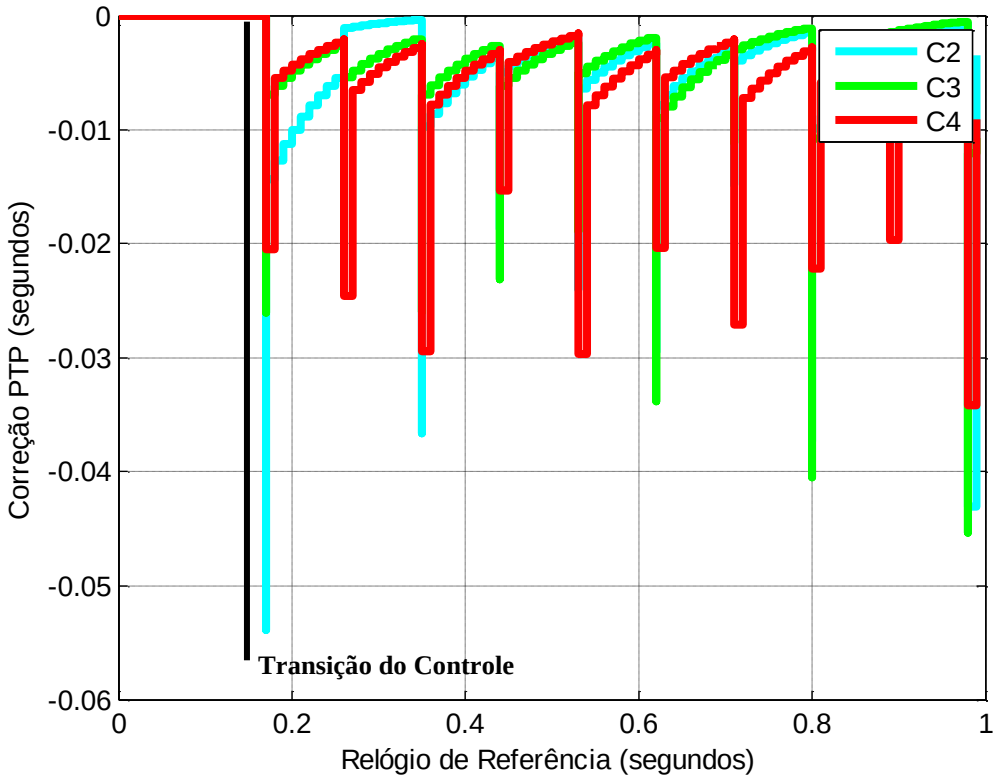


Figura 8.64 – Caso 12 - Correção modo NOMO - Algoritmo AReS Exponencial.

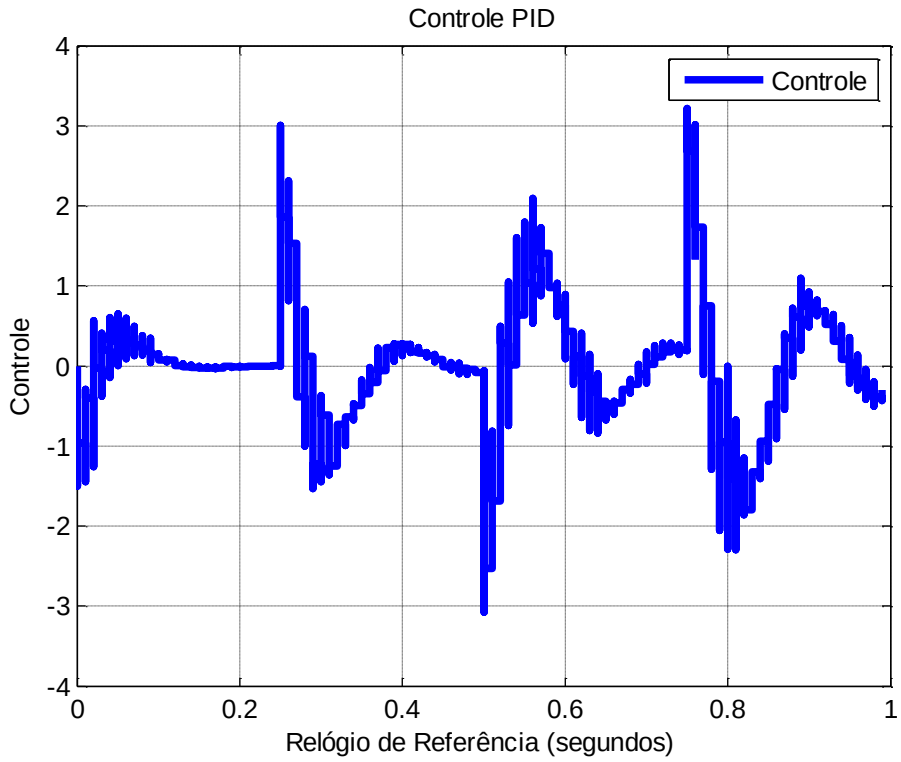


Figura 8.65 – Caso 12 - Controle PID.

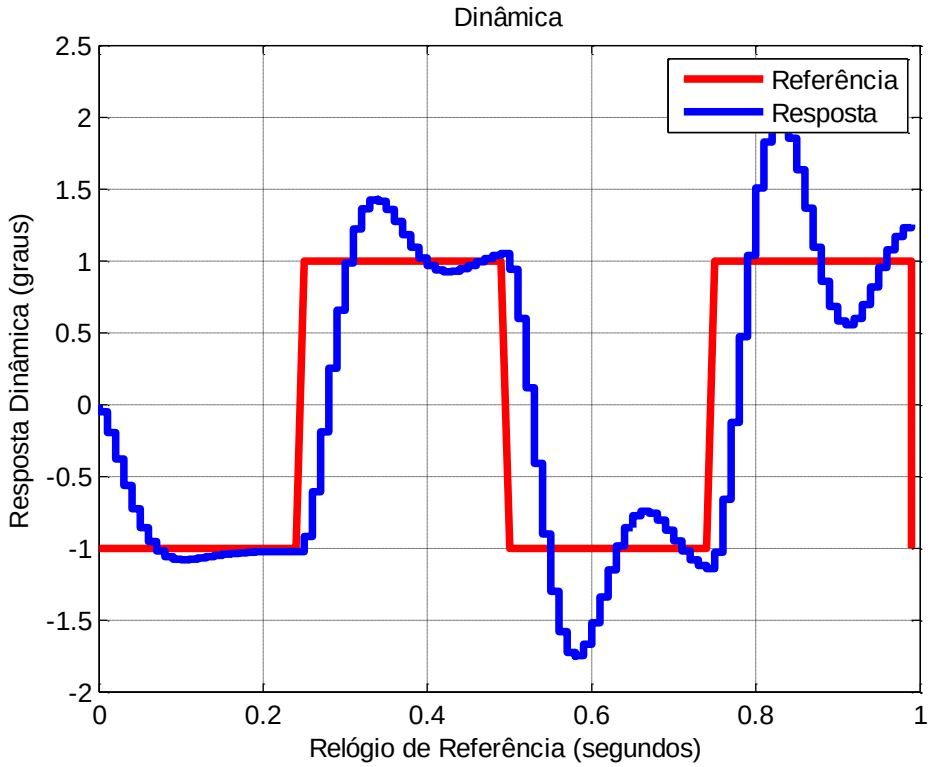


Figura 8.66 – Caso 12 - Resposta Dinâmica.

Este caso mostra uma sincronização de relógios com o algoritmo AReS exponencial que utiliza um controlador *deadbeat* no modo STM e o algoritmo PTP com *cross-fading* exponencial no modo NOMO, sobre um sistema de controle por redes em uma rede CSMA/CD. No modo STM, não é realizado o controle da dinâmica, pois é um modo de inicialização de tempo. No modo NOMO o algoritmo de sincronização se mostrou eficiente para estabelecer a sincronização. No entanto, devido à sincronização de tempo a resposta dinâmica apresentou um sobressinal. Isto ocorre, pois o método suaviza a descontinuidade de tempo. A de-sincronização inicial que antes levou o algoritmo a gerar uma instabilidade no sistema de controle, agora foi corrigida aplicando a reconfiguração dos algoritmos de tempo e utilizando o *deadbeat*. Esta técnica se mostrou eficiente para sincronizar relógios em sistemas de controle por redes e pode ser melhorada através da otimização dos parâmetros.

8.15. Caso 13 - Algoritmo FTM

Este caso possui valores de viés inicial igual a: C1: 0; C2: 0.1 seg. ; C3: 0.01 seg. ; C4: -0.1 seg.; e deriva igual a: C1: 0; C2: 1%; C3: 0.01%; C4: -0.01%.

A Figura 8.67 e Figura 8.68 mostram os valores de macrotick e diferença dos macroticks entre si (precisão), respectivamente. Nota-se que nesta abordagem não existe a visão de um relógio de referência real e sim de um relógio virtual.

Observa-se na Figura 8.67 que o relógio C1 possui grandes correções devido a fator de conversor MMCF ser diferente do restante do conjunto. No entanto, o algoritmo FTM consegue sincronizar o conjunto.

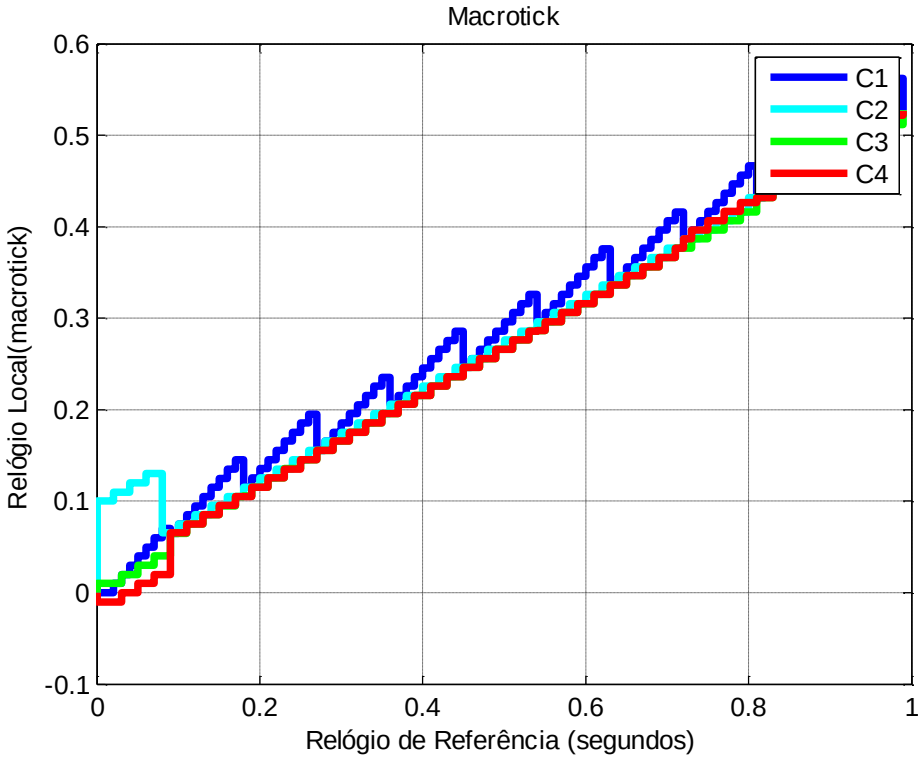


Figura 8.67 – Caso 13 - Macrotick.

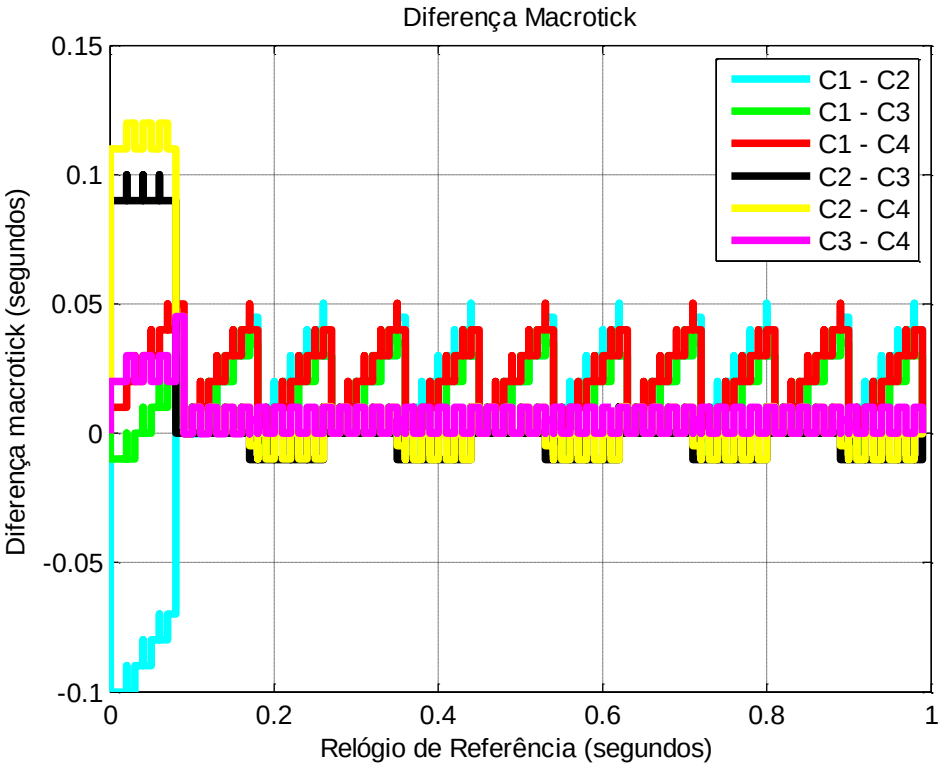


Figura 8.68 – Caso 13 - Diferença entre Macroticks - Precisão.

A Figura 8.69 mostra os valores de correção dos relógios calculados pela função de convergência do algoritmo FTM. O período de re-sincronização é de 10 microticks, ou seja, a cada 10 microticks aplica-se o processo de sincronização do algoritmo PTP.

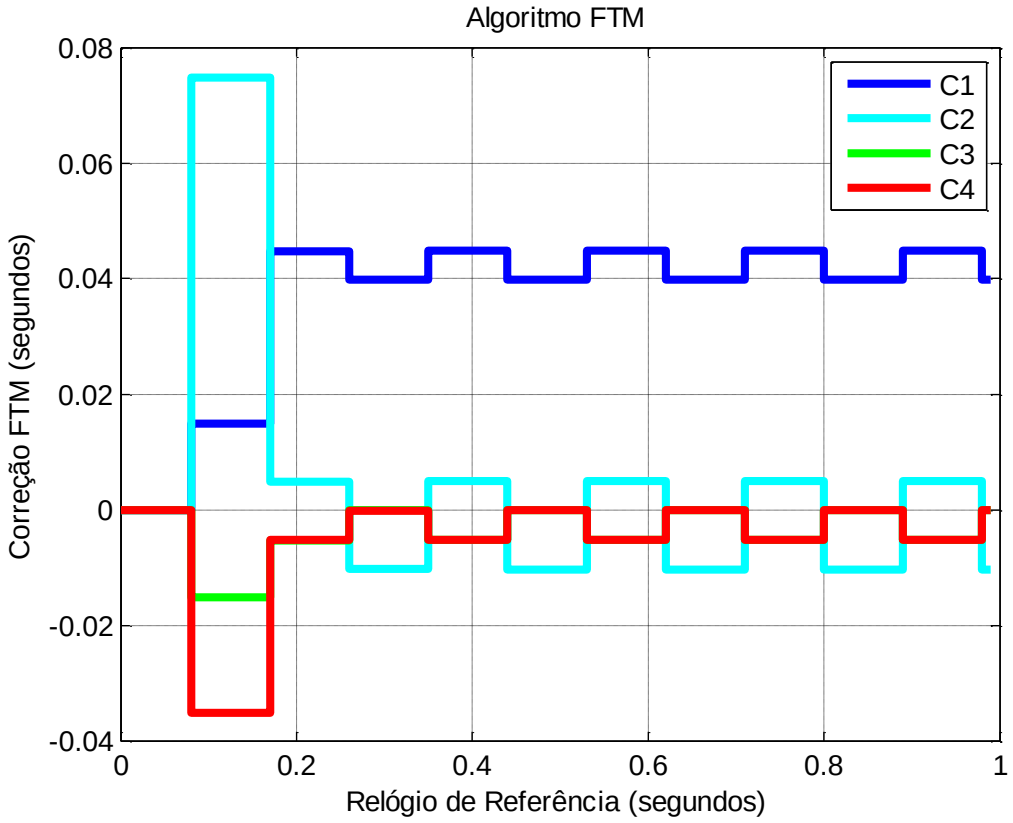


Figura 8.69 – Caso 13 - Correção Algoritmo FTM.

Este caso mostra uma sincronização de relógios com o algoritmo FTM. O algoritmo se mostrou eficiente, sendo as principais vantagens a de a sincronização ser uma equação de convergência simples e estabelecer a sincronização sem a necessidade de um relógio de referência real. A desvantagem é que o algoritmo não possui gerenciamento da descontinuidade de tempo e com isso a descontinuidade de tempo pode ser muito grande como observa-se na Figura 8.69; e também não é preparado para levar em conta o viés inicial, apesar de corrigi-lo.

8.16. Caso 14 - Deadbeat - Algoritmo FTM

Este caso possui valores de viés inicial igual a: C1: 0; C2: 0.1 seg. ; C3: 0.01 seg. ; C4: -0.1 seg.; e deriva igual a: C1: 0; C2: 1%; C3: 0.01%; C4: -0.01%.

Este caso aplica-se um algoritmo com os dois modos de correção: 1) STM e o 2) NOMO, descritos no capítulo 6, definido pelo o algoritmo ReS distribuído. Com isto, visa-se inicialmente corrigir o viés inicial com o controlador *deadbeat* e depois utiliza-se o algoritmo FTM para estabelecer a sincronização dentro de uma precisão. A Figura 8.70 e Figura 8.71 mostram, respectivamente, os valores de macrotick e diferença entre os macroticks.

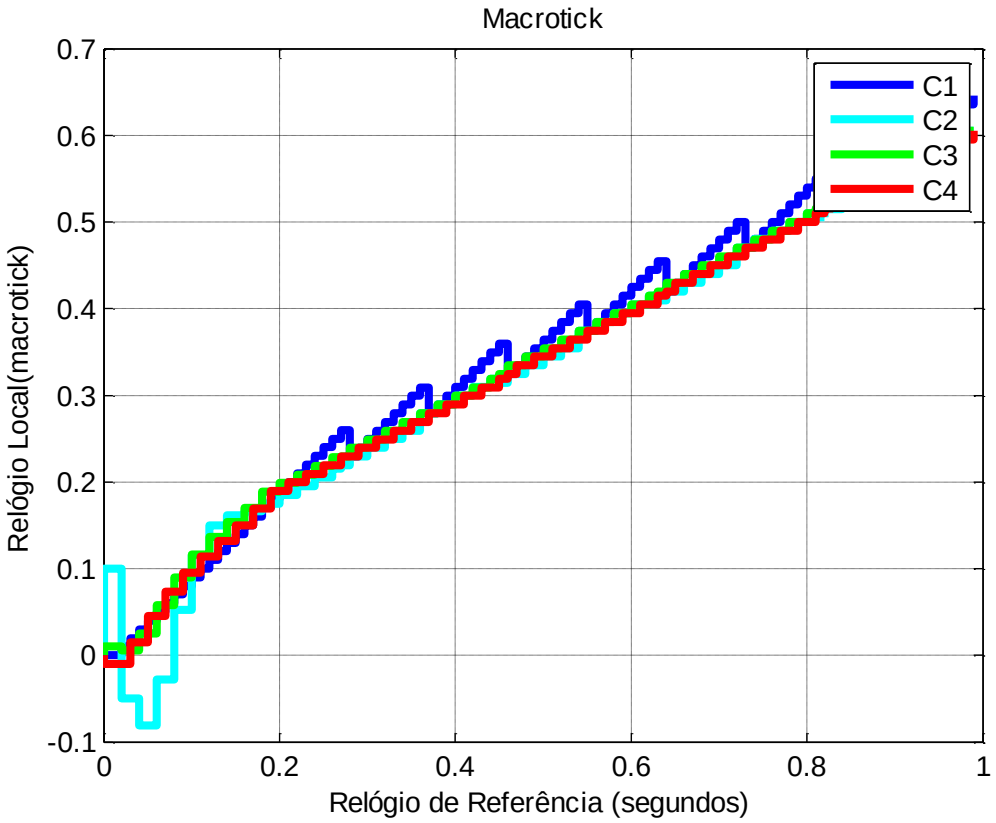


Figura 8.70 – Caso 14 - Macrotick.

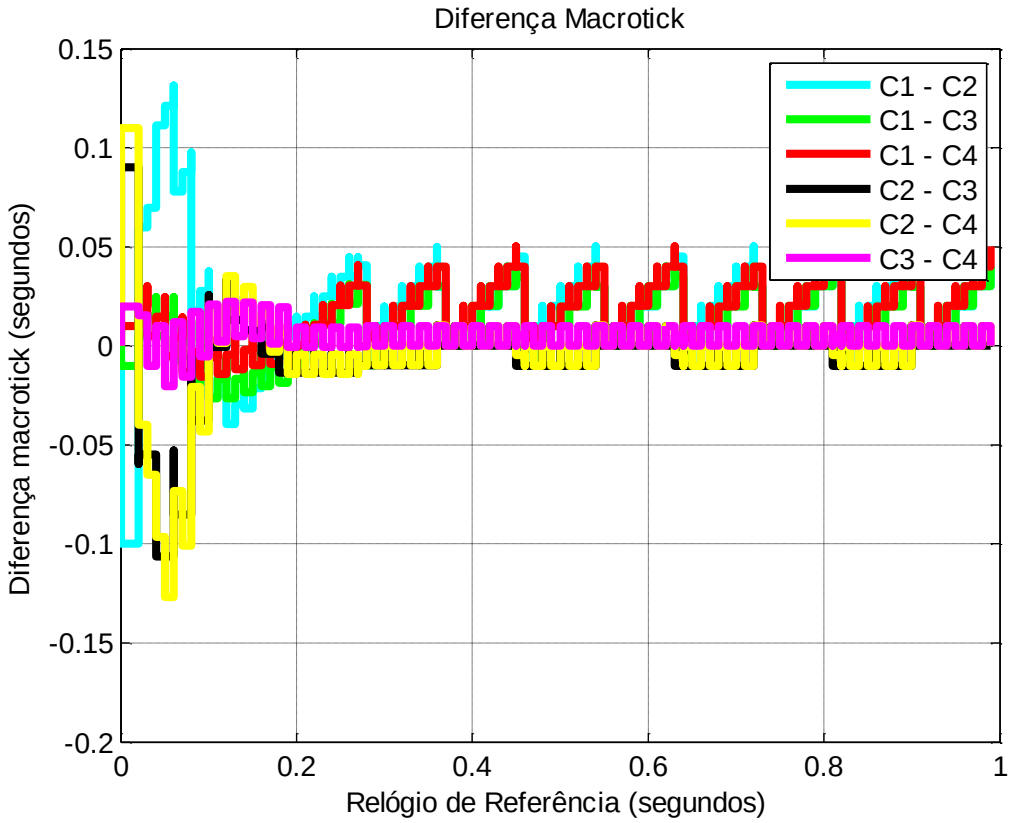


Figura 8.71 – Caso 14 - Diferença entre Macroticks - Precisão.

A Figura 8.72 mostra o valor de correção do relógio calculado pelo controlador *deadbeat* e a Figura 8.73 a correção calculada pelo algoritmo FTM. O tempo de transição dos modos de correção é de 10 macroticks vezes o maior valor de MMCF e o período de re-sincronização é de 10 microticks, ou seja, após 10 macroticks, a correção passa a ser feita pelo FTM que a cada 10 microticks aplica-se o processo de sincronização do algoritmo.

O algoritmo se mostrou eficiente, sendo que a principal vantagem é de a sincronização evitar uma descontinuidade de tempo muito grande sem perda da exatidão, observado entre os valores da Figura 8.69 e da Figura 8.73. A desvantagem é que o algoritmo precisa de um gerenciamento de transições. No entanto, o tempo da transição é conhecido, devido ao *deadbeat*, o que minimiza esta desvantagem.

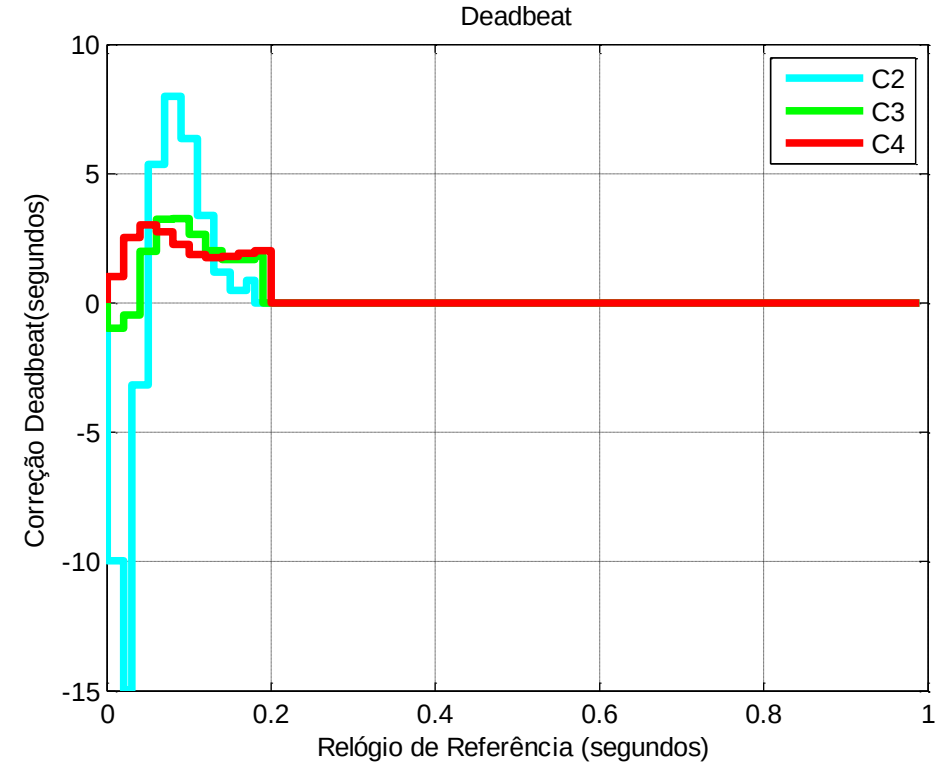


Figura 8.72 – Caso 14 - Correção STM - Controle *Deadbeat*.

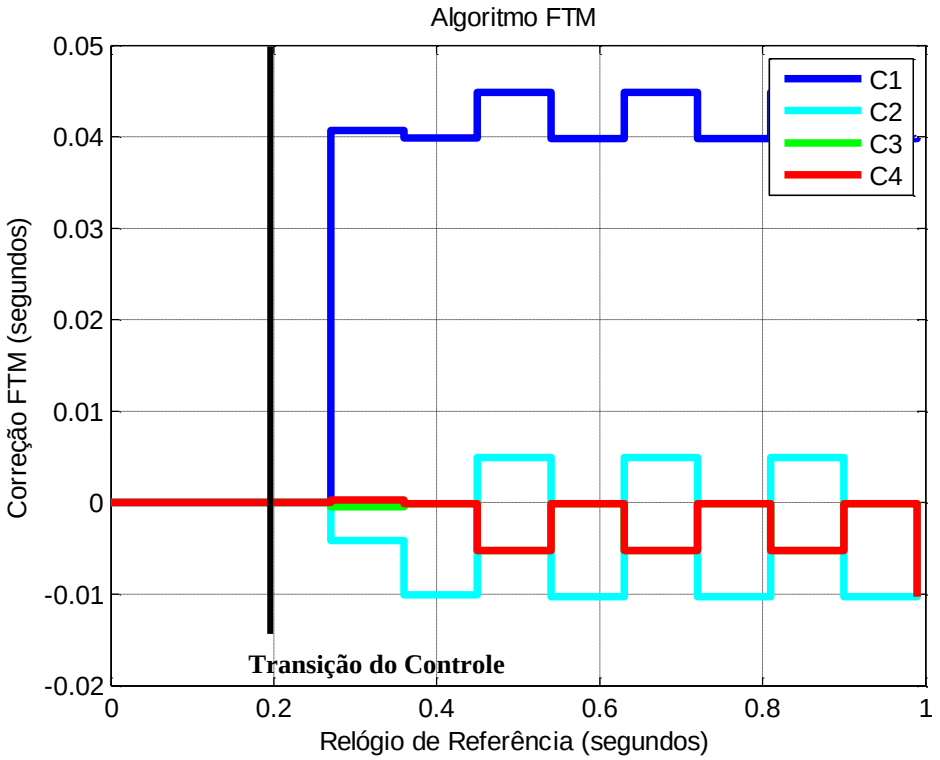


Figura 8.73 – Caso 14 - Correção NOMO - Algoritmo FTM.

8.17. Caso 15 - Algoritmo PReS Degrau

Este caso possui valores de viés inicial igual a: C1: 0; C2: 0.1 seg. ; C3: 0.01 seg. ; C4: -0.1 seg.; e deriva igual a: C1: 0; C2: 1%; C3: 0.01%; C4: -0.01%.

Neste caso, aplica-se um algoritmo com os dois modos de correção: 1) STM e o 2) NOMO, descritos no capítulo 6. Com isto, visa-se inicialmente corrigir o viés inicial com o controlador *deadbeat* e depois utiliza-se a técnica FTM em conjunto com uma função *cross-fading* degrau, definido pelo o algoritmo PReS com uma função degrau. A Figura 8.74 e Figura 8.75 mostram, respectivamente, os valores de macrotick e diferença entre os macroticks.

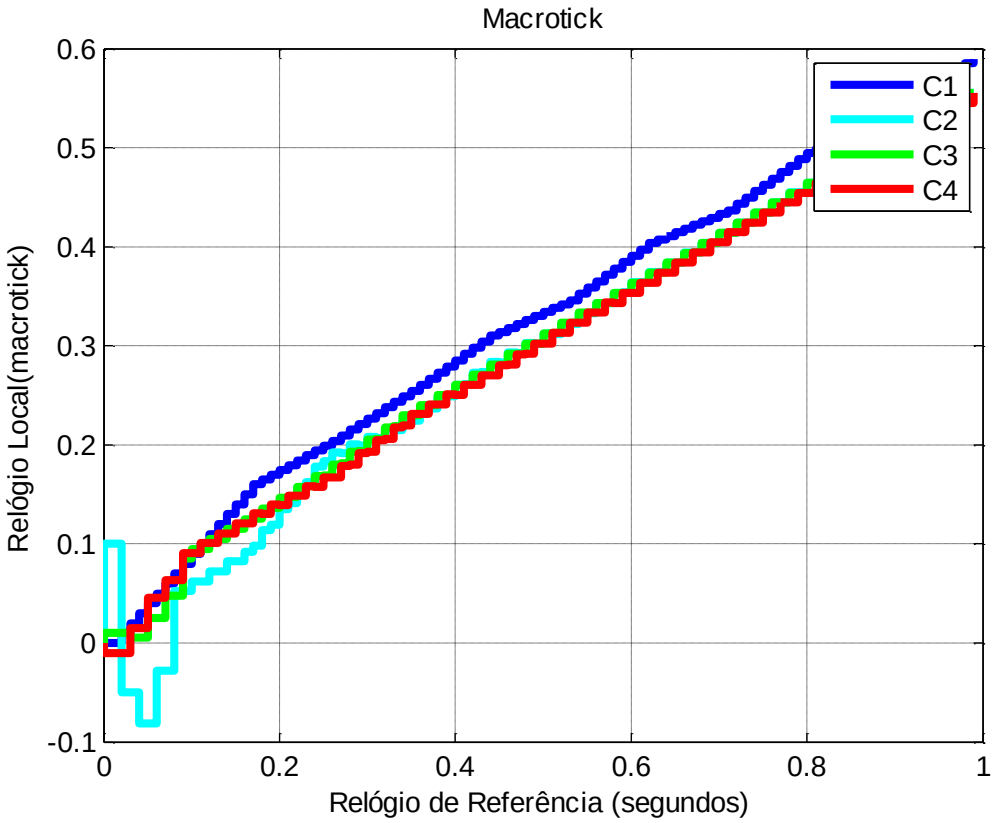


Figura 8.74 – Caso 15 - Macrotick.

A Figura 8.76 mostra o valor de correção do relógio calculado pelo controlador *deadbeat*.

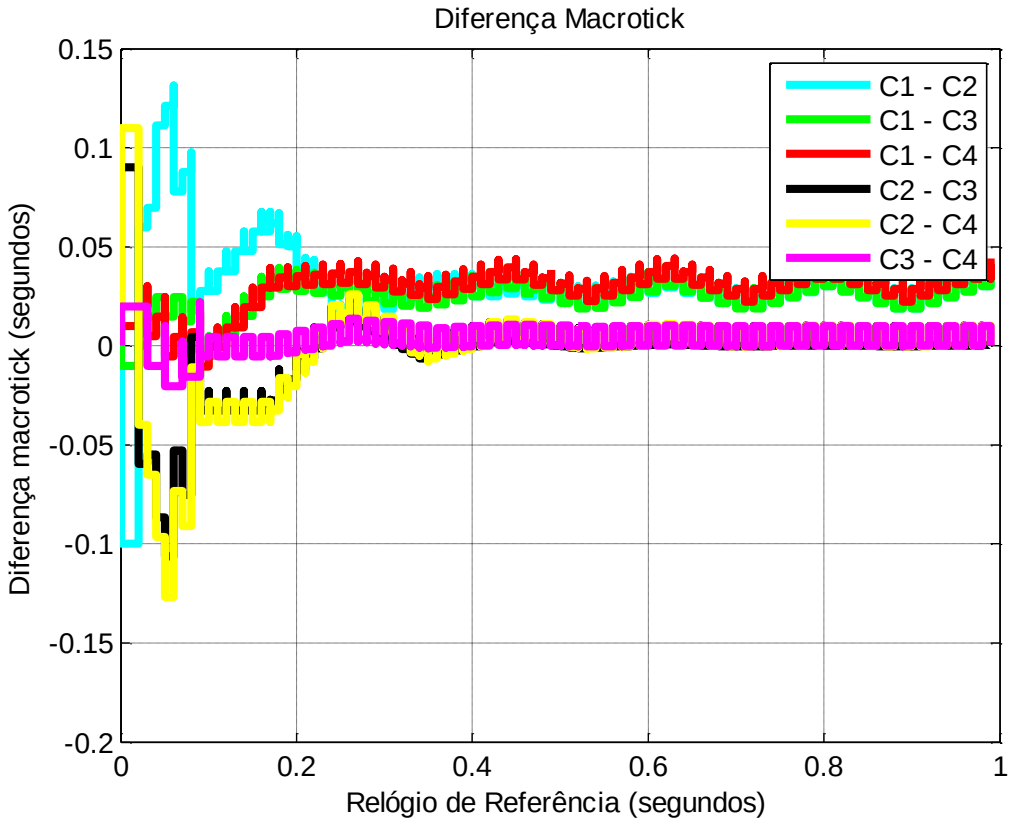


Figura 8.75 – Caso 15 - Diferença entre Macroticks.

A Figura 8.77 mostra a função *cross-fading* degrau. A Figura 8.78 mostra a correção calculada pela função de convergência do algoritmo FTM vezes a função *cross-fading* degrau. O tempo de transição entre os modos de correção é de 10 macroticks vezes o maior MMCF (tempo em que ocorre a transição entre o STM e NOMO) e o período de re-sincronização é de 10 microticks, ou seja, após 10 macroticks, a correção é recalculada pela função de convergência do algoritmo FTM. No entanto, com o uso da função degrau, a cada instante (assim como no *deadbeat*) é aplicada uma correção multiplicada pelo valor da função *cross-fading*. Assim, a cada instante existe uma correção do algoritmo.

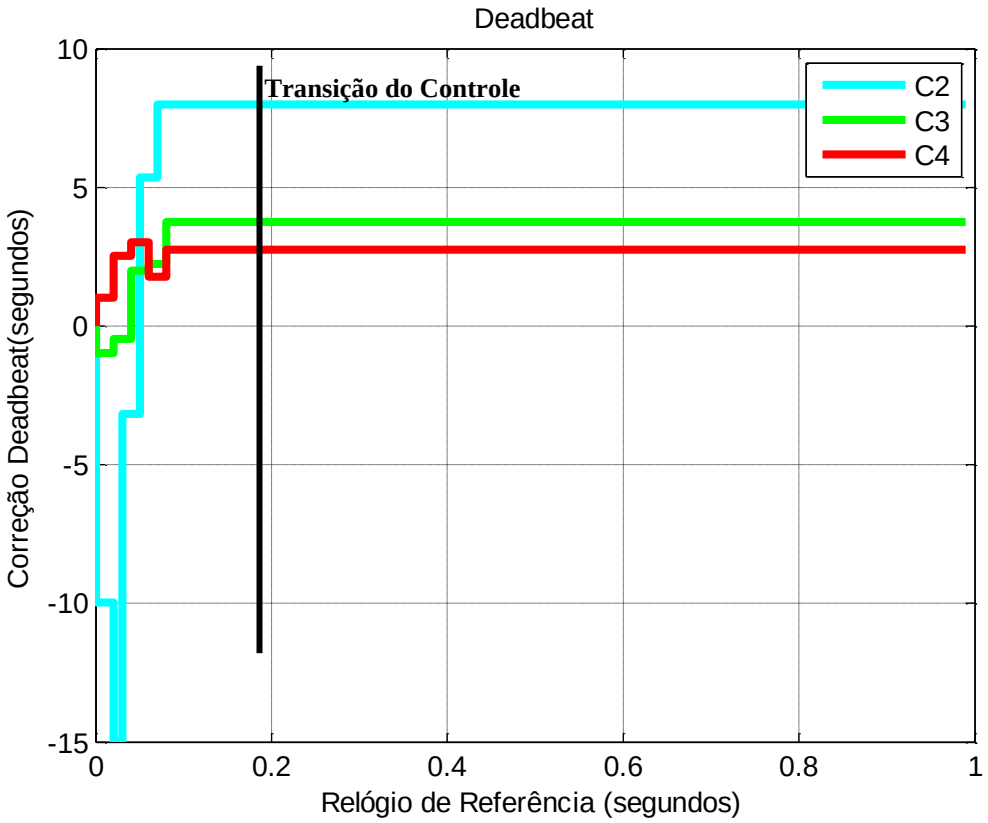


Figura 8.76 – Caso 15 - Correção STM - Algoritmo PReS Degrau.

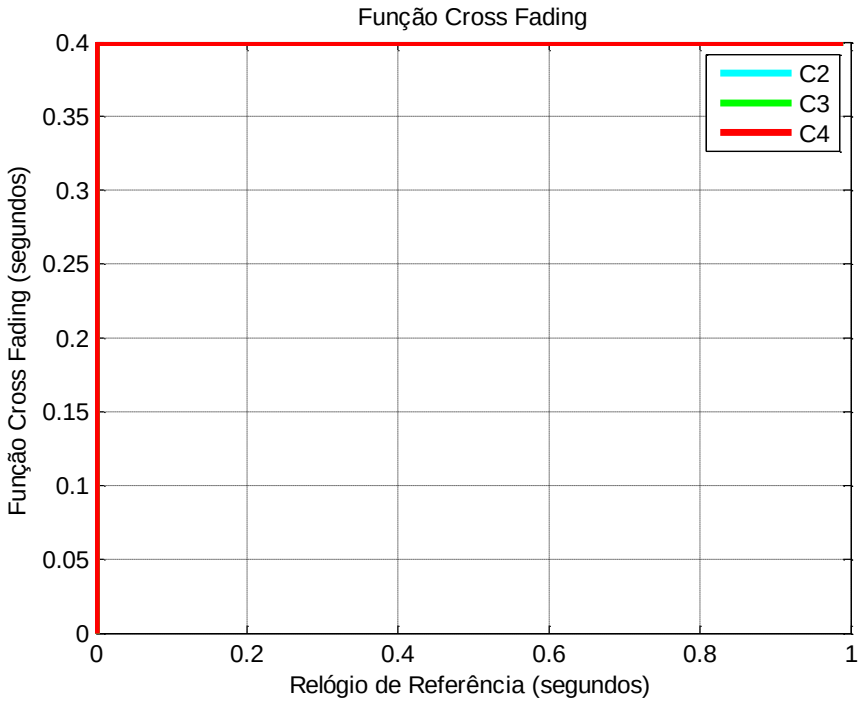


Figura 8.77 – Caso 15 - Função Cross-Fading Degrau.

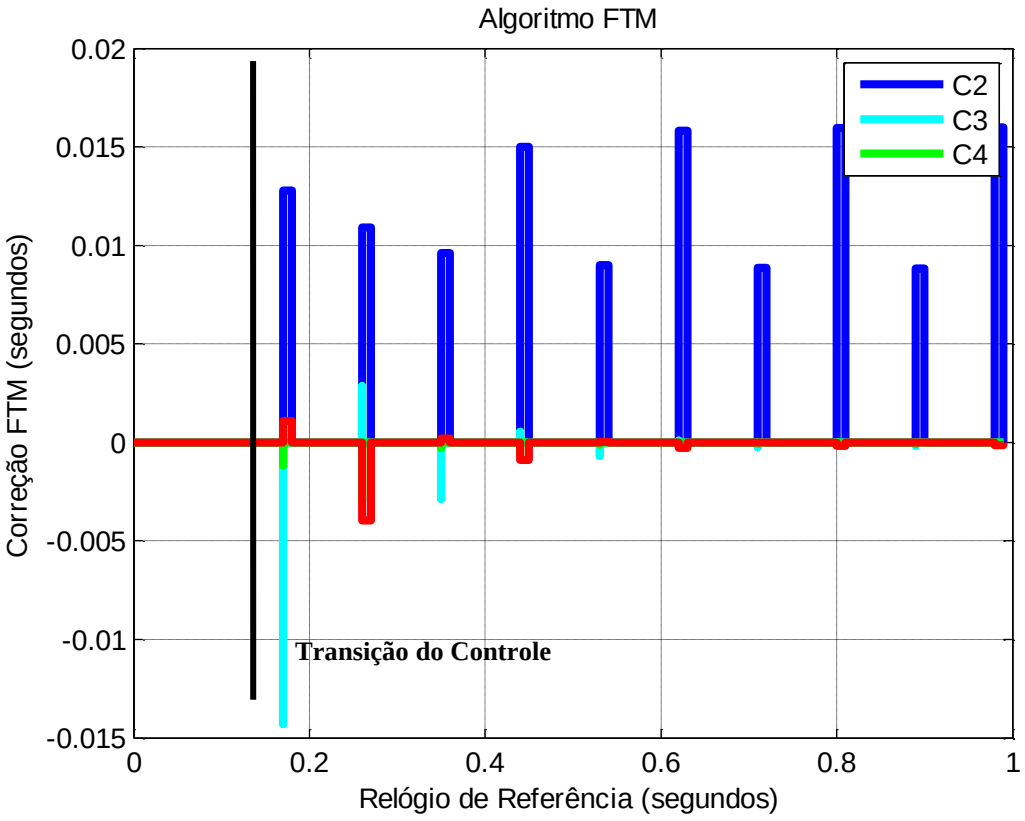


Figura 8.78 – Caso 15 - Correção NOMO - Algoritmo PReS Degrau.

Este caso mostra uma sincronização de relógios de métrica mista com o algoritmo PReS degrau. O algoritmo se mostrou eficiente, sendo que a principal vantagem é da sincronização modular a descontinuidade de tempo via uma função degrau, evitando excursões de tempo muito grande. A precisão foi prejudicada em relação ao caso anterior. No entanto, ajustando e otimizando os parâmetros é possível conseguir resultados melhores. A desvantagem é que o algoritmo precisa de um gerenciamento de transições e necessita atualizar a cada instante o valor de tempo. No entanto, o tempo da transição é conhecido, devido ao *deadbeat*, o que minimiza esta desvantagem.

8.18. Caso 16 - Algoritmo PReS Rampa

Este caso possui valores de viés inicial igual a: C1: 0; C2: 0.1 seg. ; C3: 0.01 seg. ; C4: -0.1 seg.; e deriva igual a: C1: 0; C2: 1%; C3: 0.01%; C4: -0.01%.

Neste caso, aplica-se um algoritmo com os dois modos de correção: 1) STM e o 2) NOMO, descritos no capítulo 6. Com isto, visa-se inicialmente corrigir o viés inicial com o controlador *deadbeat* e depois utiliza-se a técnica FTM em conjunto com uma função *cross-fading* rampa, definindo o algoritmo PReS rampa. A Figura 8.79 e Figura 8.80 mostram, respectivamente, os valores de macrotick e diferença entre os macroticks.

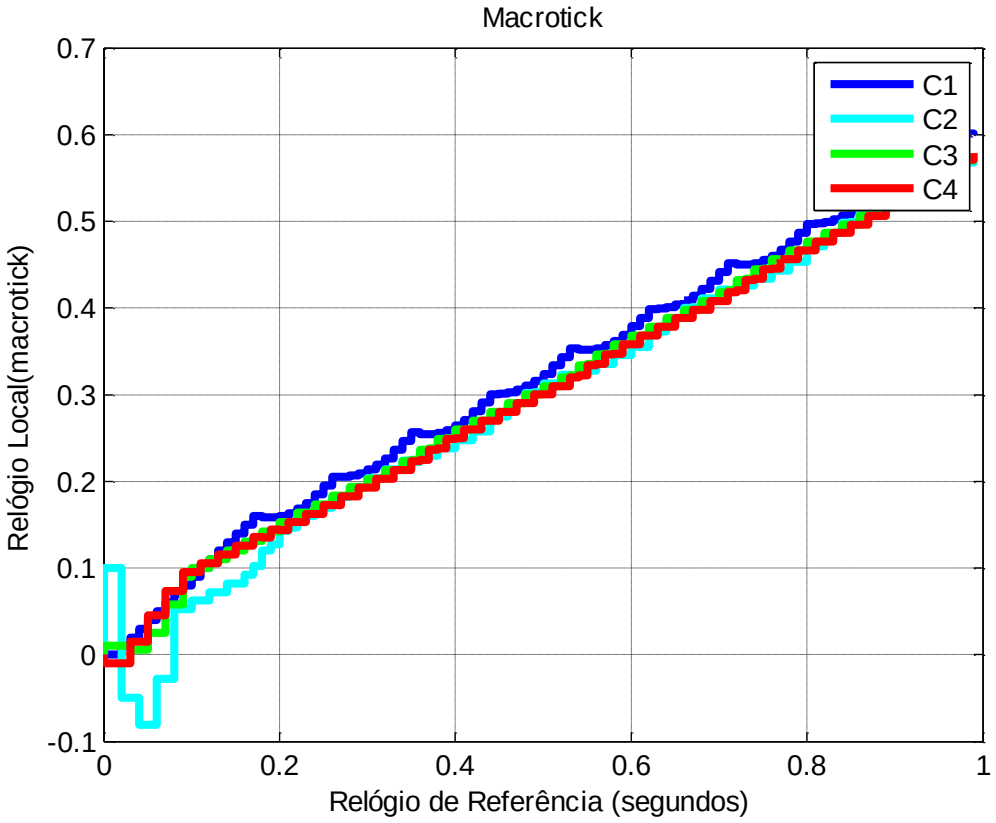


Figura 8.79 – Caso 16 - Macrotick.

A Figura 8.81 mostra o valor de correção do relógio calculado pelo controlador *deadbeat*. A Figura 8.82 mostra a função *cross-fading* rampa. A Figura 8.83 mostra a correção calculada pela função de convergência do algoritmo FTM mais a função *cross-fading* rampa.

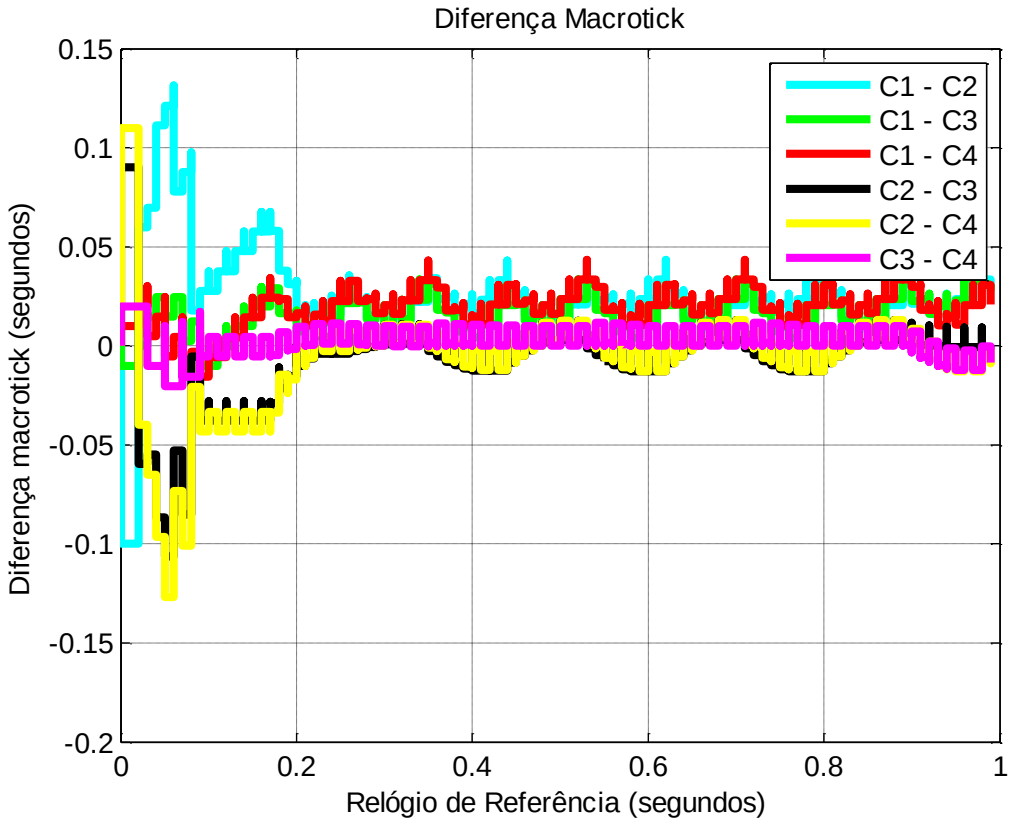


Figura 8.80 – Caso 16 - Diferença entre Macroticks.

O tempo de transição entre os modos de correção é de 10 macroticks vezes o maior MMCF e o período de re-sincronização é de 10 microticks, ou seja, após 10 macroticks vezes MMCF a correção, agora, é calculada pela função de convergência do algoritmo FTM. No entanto, com o uso da função rampa, a cada instante é aplicada uma correção multiplicada pelo valor da função *cross-fading*. Esta correção vai decaindo de valor de acordo com a função *cross-fading* implementada. No caso da reta, é necessário uma segunda transição para evitar que o valor da função *cross-fading* rampa fique negativa e inverta a fase do sistema tornando-o instável.

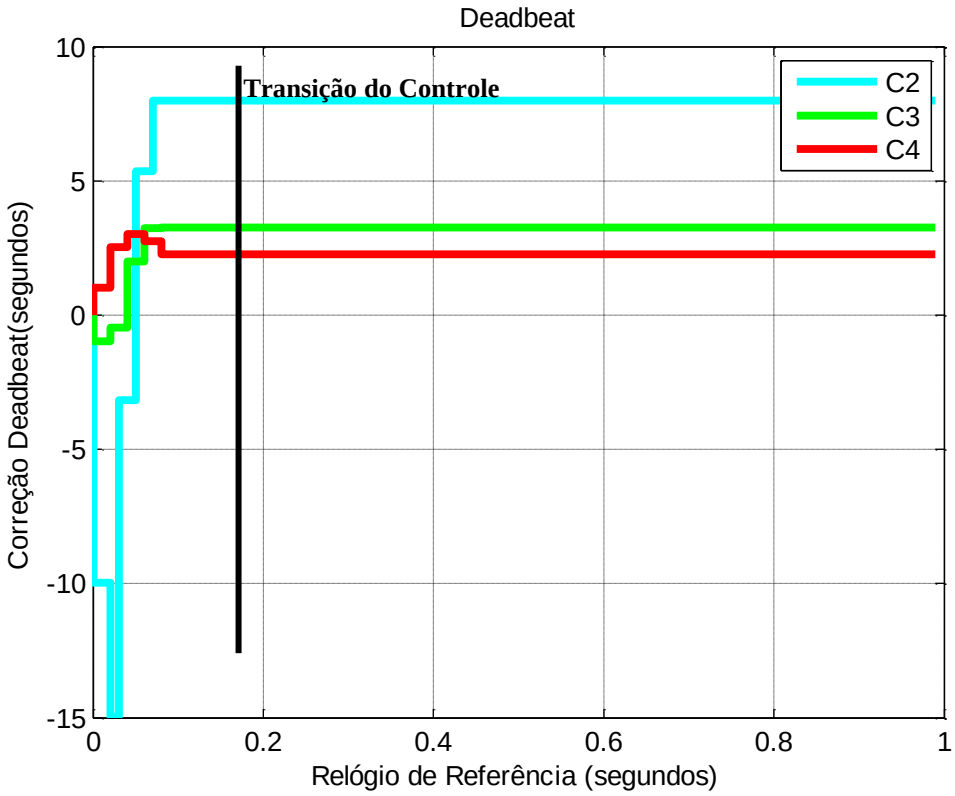


Figura 8.81 – Caso 16 - Correção STM - Algoritmo PReS Rampa.

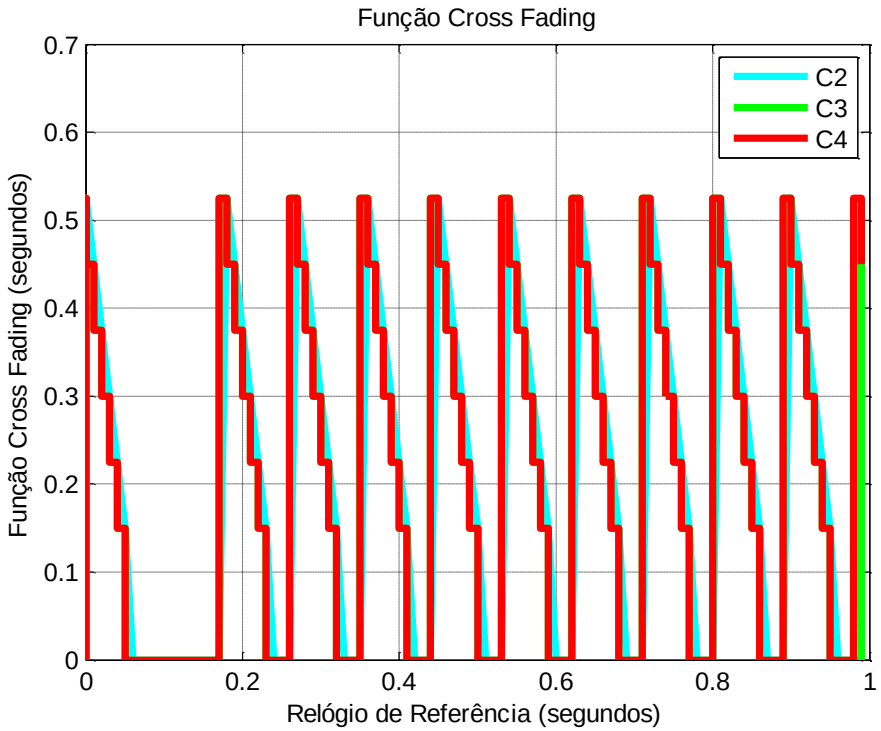


Figura 8.82 – Caso 16 - Função *Cross-Fading* Degrau.

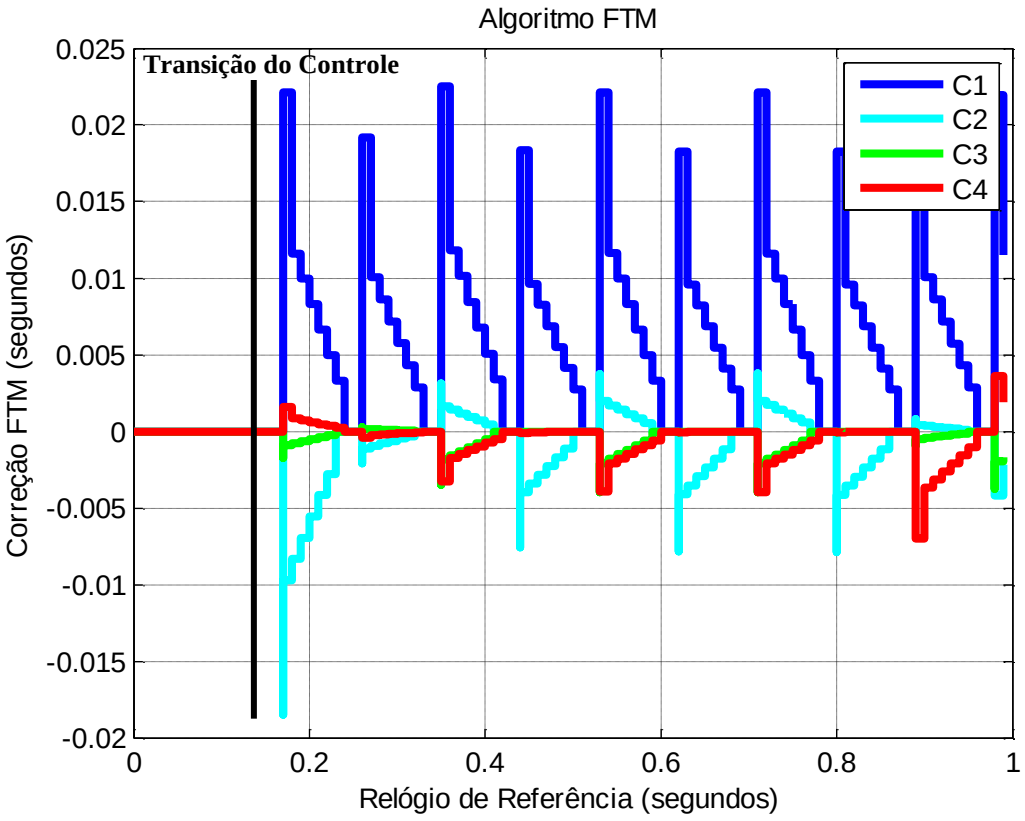


Figura 8.83 – Caso 16 - Correção NOMO - Algoritmo PReS - Rampa.

Este caso mostra uma sincronização de relógios de métrica mista com o algoritmo PReS rampa. O algoritmo se mostrou eficiente, sendo que a principal vantagem é da possibilidade de modular a descontinuidade de tempo devido à sincronização e ainda melhorar a precisão. Isto ocorre, devido à função cross-fading modular a correção do algoritmo ao longo de cada instante, diferente do algoritmo FTM que aplica toda a correção em um único instante. A desvantagem é que o algoritmo precisa de um gerenciamento de transições e necessita atualizar a cada instante o valor de tempo. No entanto, o tempo da transição é conhecido, devido ao *deadbeat*, o que minimiza esta desvantagem.

8.19. Caso 17 - PReS Exponencial

Este caso possui valores de viés inicial igual a: C1: 0; C2: 0.1 seg. ; C3: 0.01 seg. ; C4: -0.1 seg.; e deriva igual a: C1: 0; C2: 1%; C3: 0.01%; C4: -0.01%.

Neste caso, aplica-se um algoritmo com os dois modos de correção: 1) STM e o 2) NOMO, descritos no capítulo 6, definido pelo algoritmo PReS. Com isto, visa-se inicialmente corrigir o viés inicial com o controlador *deadbeat* e depois utiliza-se a técnica FTM em conjunto com uma função *cross-fading* exponencial, definindo o algoritmo PReS Exponencial. A Figura 8.84 e Figura 8.85 mostram, respectivamente, os valores de macrotick e diferença entre os macroticks.

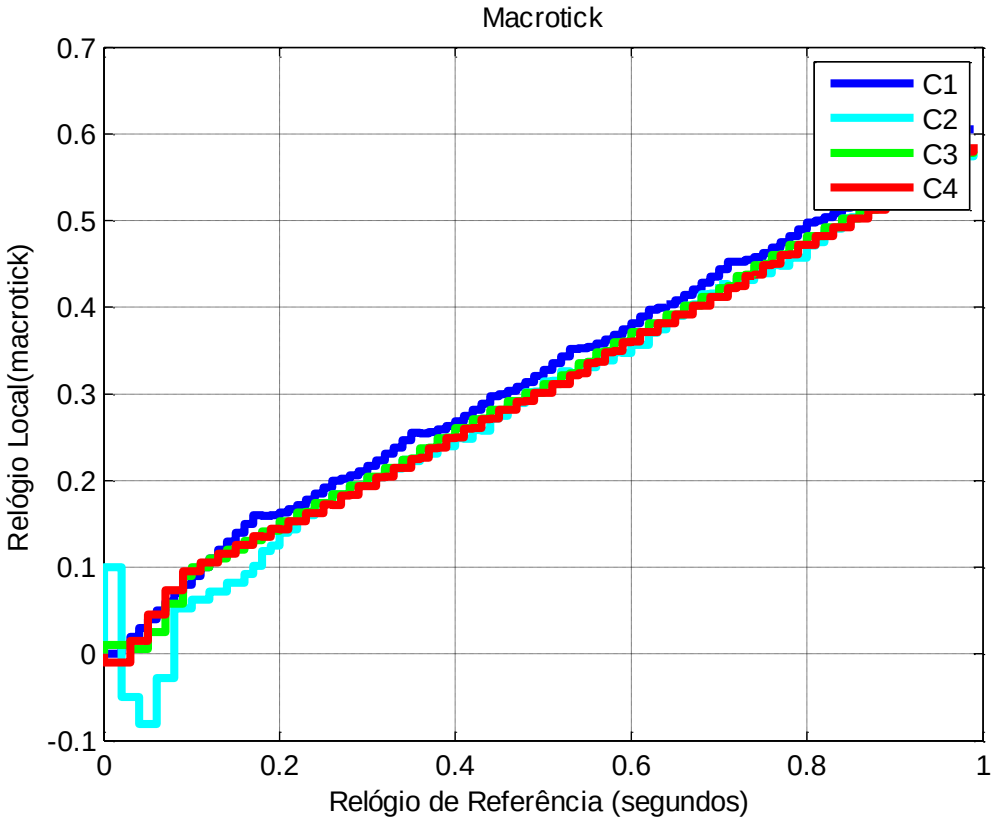


Figura 8.84 – Caso 17 - Macrotick.

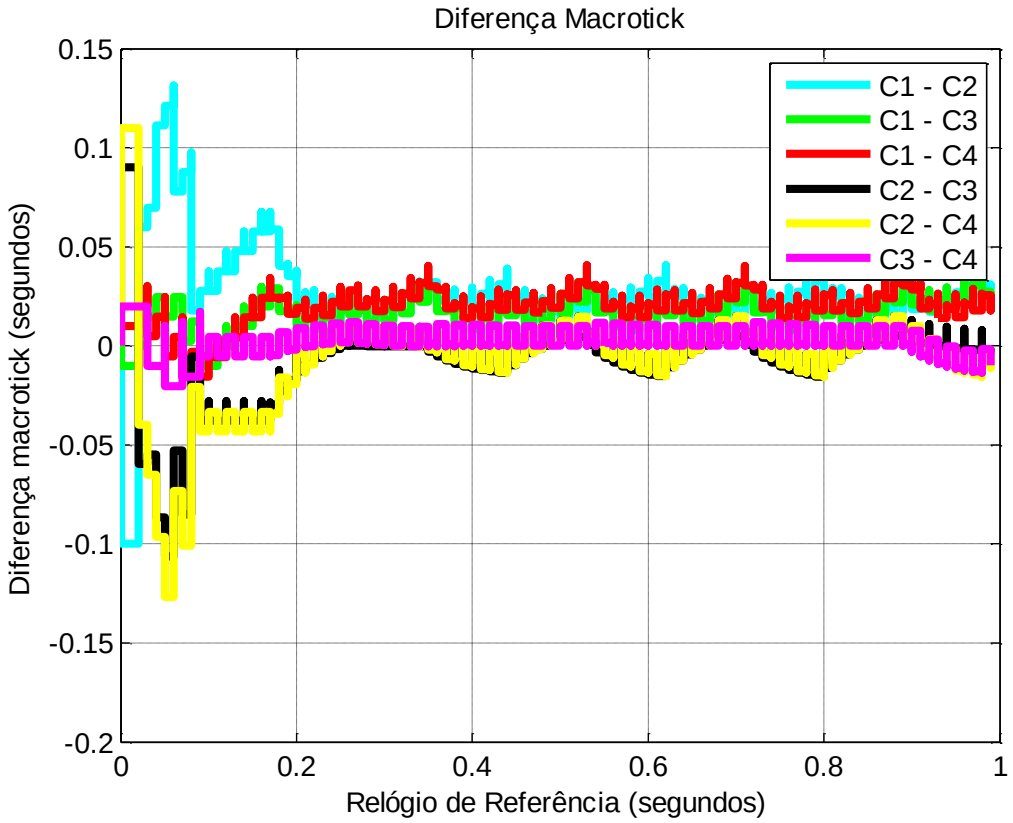


Figura 8.85 – Caso 17 - Diferença entre Macroticks.

A Figura 8.86 mostra o valor de correção do relógio calculado pelo controlador *deadbeat* no modo STM. A Figura 8.87 mostra a função *cross-fading* exponencial. A Figura 8.88 mostra a correção calculada pela função de convergência do algoritmo FTM vezes a função *cross-fading* exponencial no modo NOMO. O tempo de transição entre os modos de correção é de 10 macroticks vezes o maior MMCF e o período de re-sincronização é de 10 microticks, ou seja, após 10 macroticks, a correção é recalculada. No entanto, com o uso da função exponencial, a cada instante é aplicada a correção multiplicada pelo valor da função exponencial. Esta correção vai decaindo de valor de acordo com a função *cross-fading*. No caso da exponencial, como o valor da função tende à zero, não é necessário uma segunda transição para evitar que o valor da função *cross-fading* fique negativa.

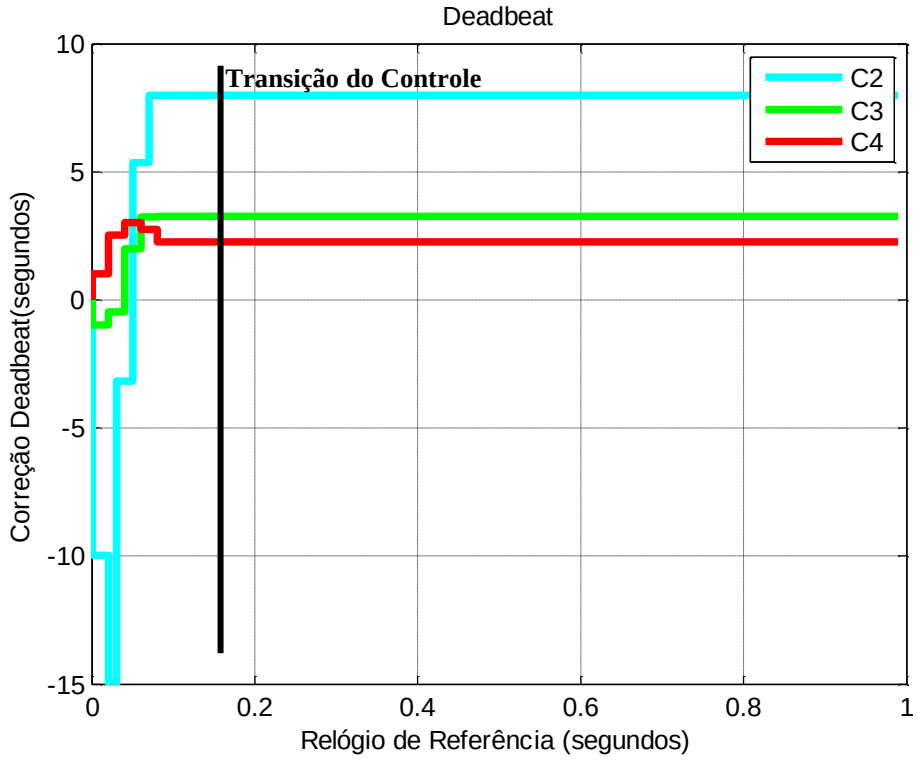


Figura 8.86 – Caso 17 - Correção STM - PReS Exponencial.

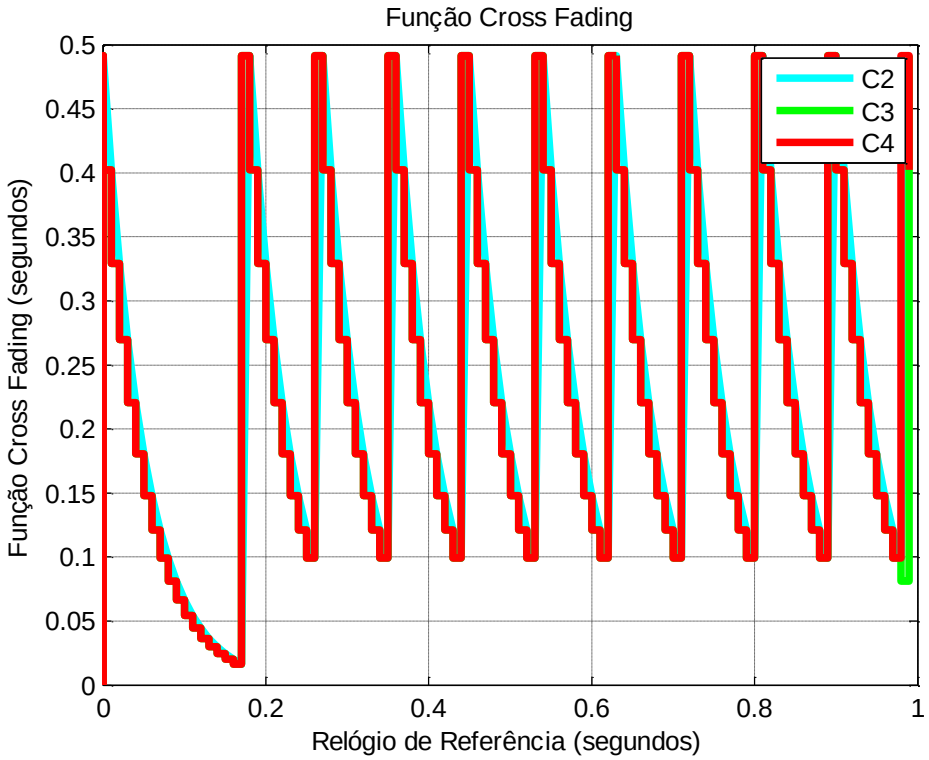


Figura 8.87 – Caso 17 - Função *Cross-Fading* Exponencial.

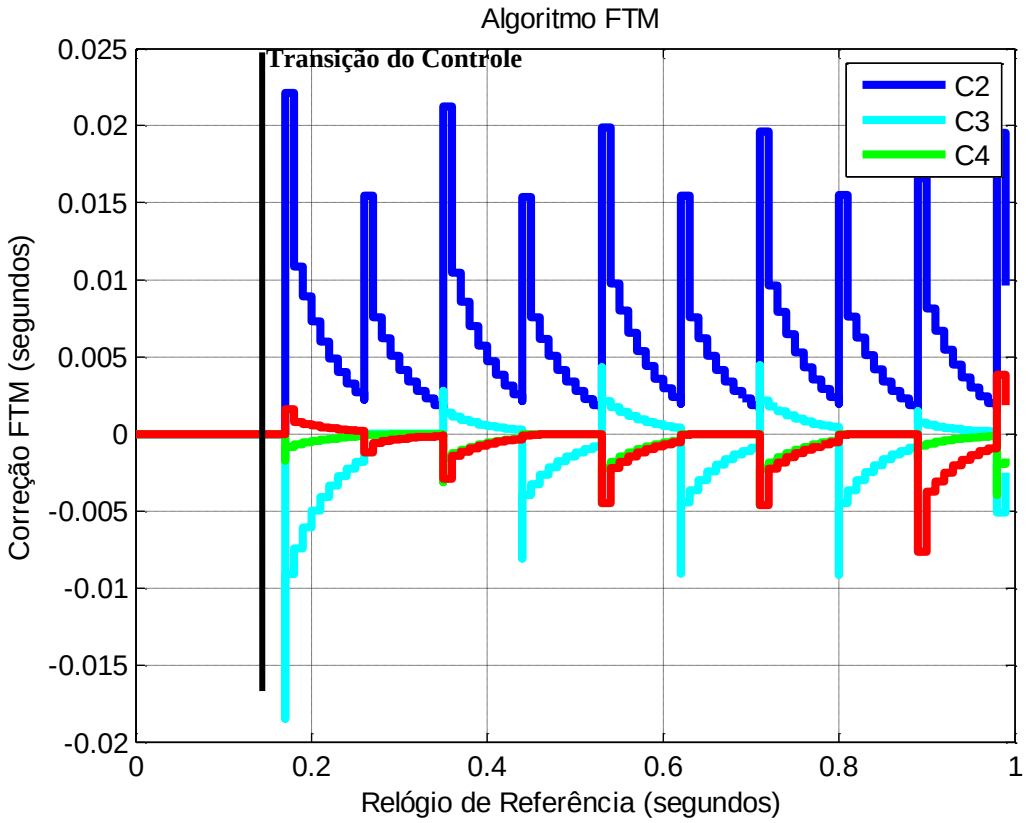


Figura 8.88 – Caso 17 - Correção NOMO - Algoritmo PReS - Exponencial.

Este caso mostra uma sincronização de relógios de métrica mista com o algoritmo PReS Exponencial. O algoritmo se mostrou eficiente, sendo que a principal vantagem é da possibilidade de modular a descontinuidade de tempo devido à sincronização via uma função exponencial e ainda melhorar a precisão. Isto ocorre, devido à função cross-fading modular a correção do algoritmo ao longo de cada instante, diferente do algoritmo PTP que aplica toda a correção em um único instante. A desvantagem é que o algoritmo precisa de um gerenciamento de transições e necessita atualizar a cada instante o valor de tempo. No entanto, o tempo da transição é conhecido, devido ao *deadbeat*, o que minimiza esta desvantagem.

8.20. Caso 18 - Algoritmo FTM - CSMA/CD

Este caso possui valores de viés inicial igual a: C1: 0; C2: 0.1 seg. ; C3: 0.01 seg. ; C4: -0.1 seg.; e deriva igual a: C1: 0; C2: 1%; C3: 0.01%; C4: -0.01%.

Neste caso, aplica-se o algoritmo FTM para corrigir o tempo sobre um sistema de controle por redes com uma rede CSMA/CD. A Figura 8.89 e Figura 8.90 mostram os valores de macrotick e diferença dos macroticks, respectivamente.

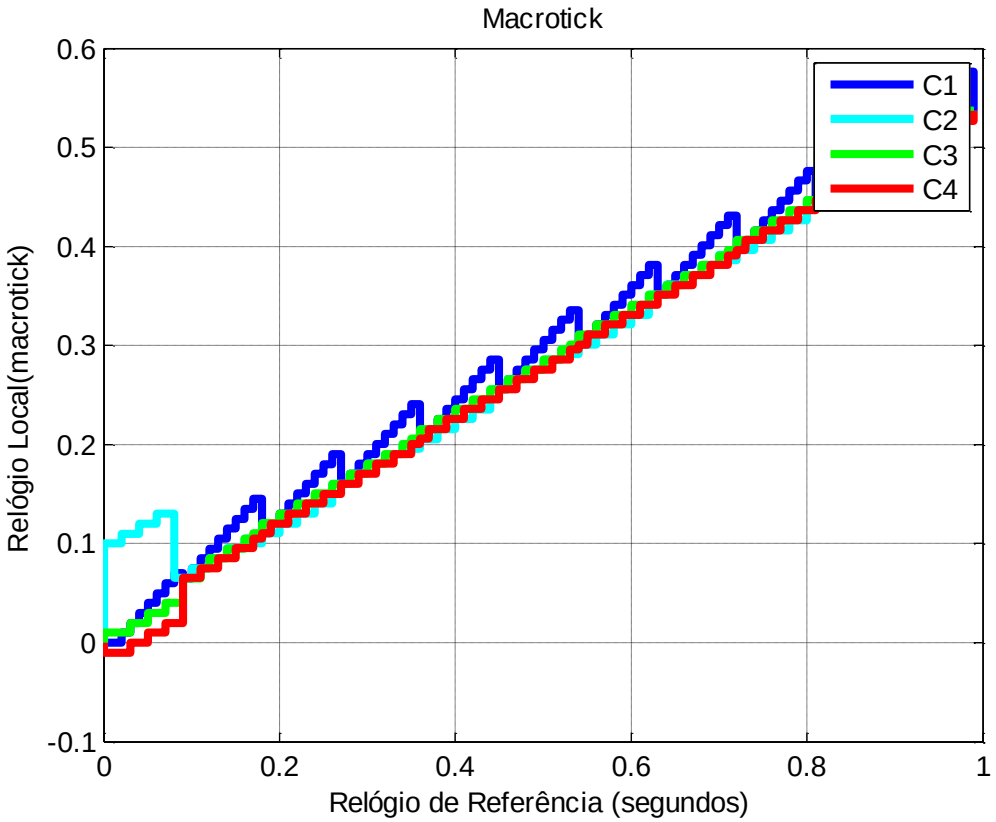


Figura 8.89 – Caso 18 - Macrotick.

A Figura 8.91 mostra o valor de correção do relógio calculado pela função de convergência do algoritmo FT. O período de re-sincronização é de 10 microticks, ou seja, a cada 10 microticks aplica-se o processo de sincronização do algoritmo FTM. A Figura 8.92 e a Figura 8.93 mostram a lei de controle PID e a resposta dinâmica do sistema de controle por rede em uma rede CSMA/CD.

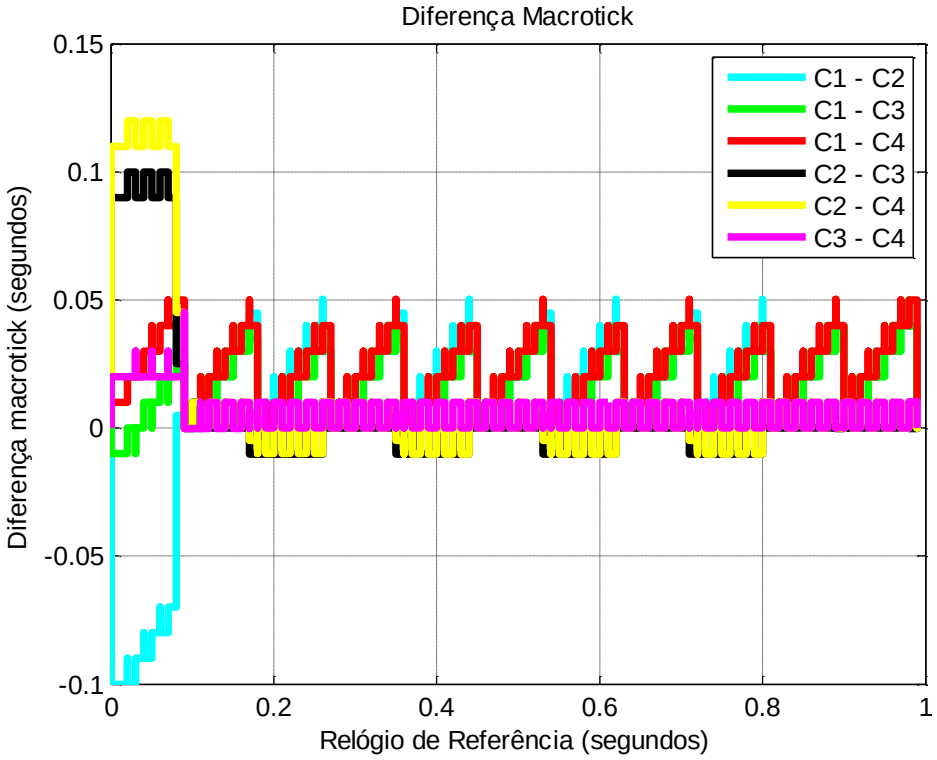


Figura 8.90 – Caso 18 - Diferença entre Macroticks.

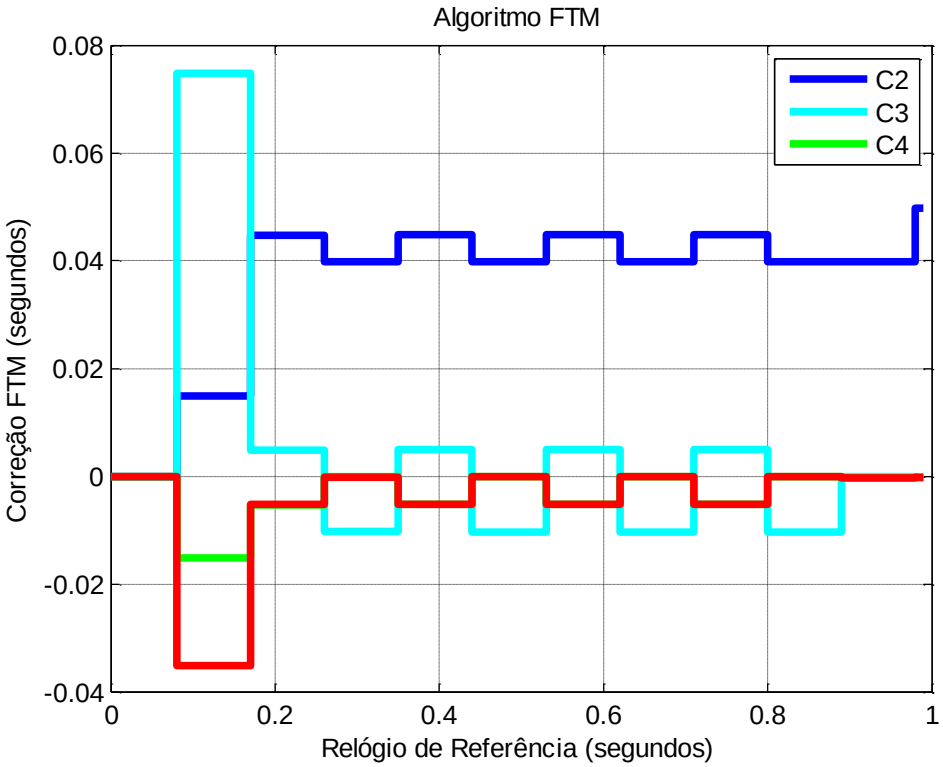


Figura 8.91 – Caso 18 - Correção Algoritmo FTM.

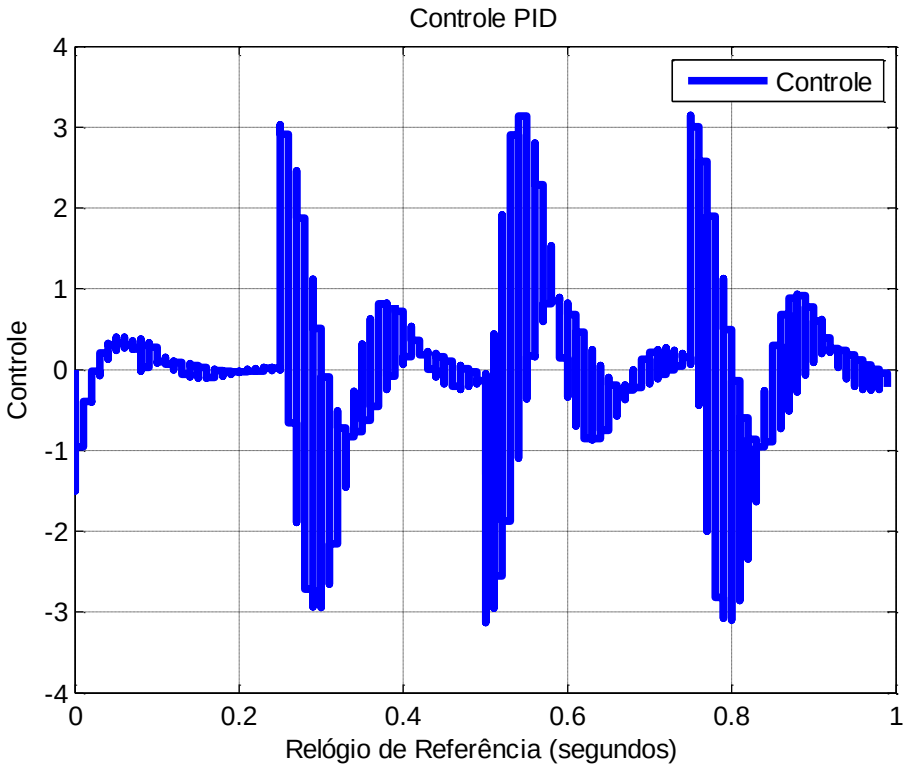


Figura 8.92 – Caso 18 - Controle PID.

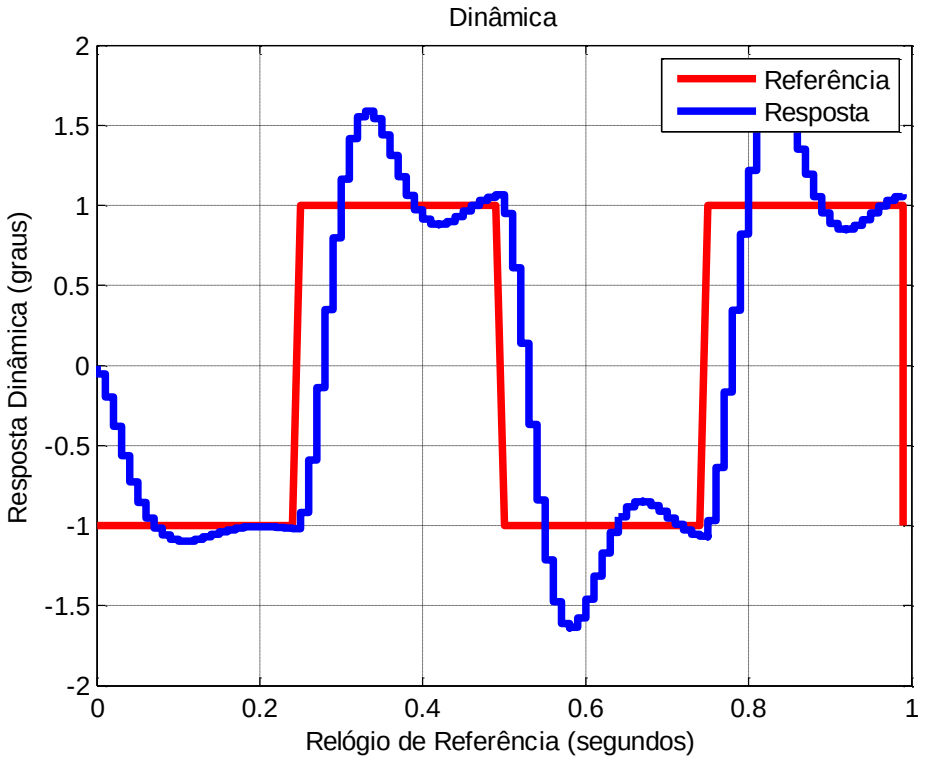


Figura 8.93 – Caso 18 - Resposta Dinâmica.

Este caso mostra uma sincronização de relógios com o algoritmo FTM sobre um sistema de controle por redes em uma rede CSMA/CD. O algoritmo de sincronização se mostrou eficiente para estabelecer a sincronização. No entanto, devido à grande excursão de tempo (descontinuidade de tempo) o algoritmo ao estabelecer a sincronização prejudica indiretamente o sistema de controle, causando sobressinal na resposta dinâmica. Isto ocorre, pois o algoritmo FTM não foi desenvolvido para levar em conta o sistema de controle e a de-sincronização inicial.

8.21. Caso 19 - Deadbeat - FTM - CSMA/CD

Este caso possui valores de viés inicial igual a: C1: 0; C2: 0.1 seg. ; C3: 0.01 seg. ; C4: -0.1 seg.; e deriva igual a: C1: 0; C2: 1%; C3: 0.01%; C4: -0.01%. Este caso aplica-se um algoritmo com os dois modos de correção: 1) STM e o 2) NOMO, descritos no capítulo 6. Com isto, visa-se inicialmente corrigir o viés inicial com o *deadbeat* e depois com o algoritmo FTM sobre um sistema de controle por redes sobre uma rede CSMA/CD. A Figura 8.94 e Figura 8.95 mostram, respectivamente, os valores de macrotick e diferença dos macroticks.

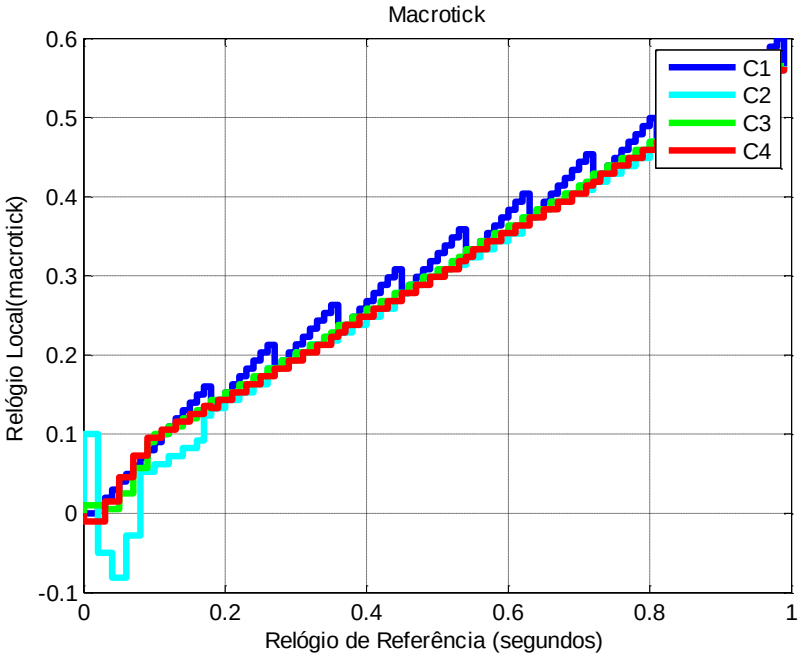


Figura 8.94 – Caso 19 - Macrotick.

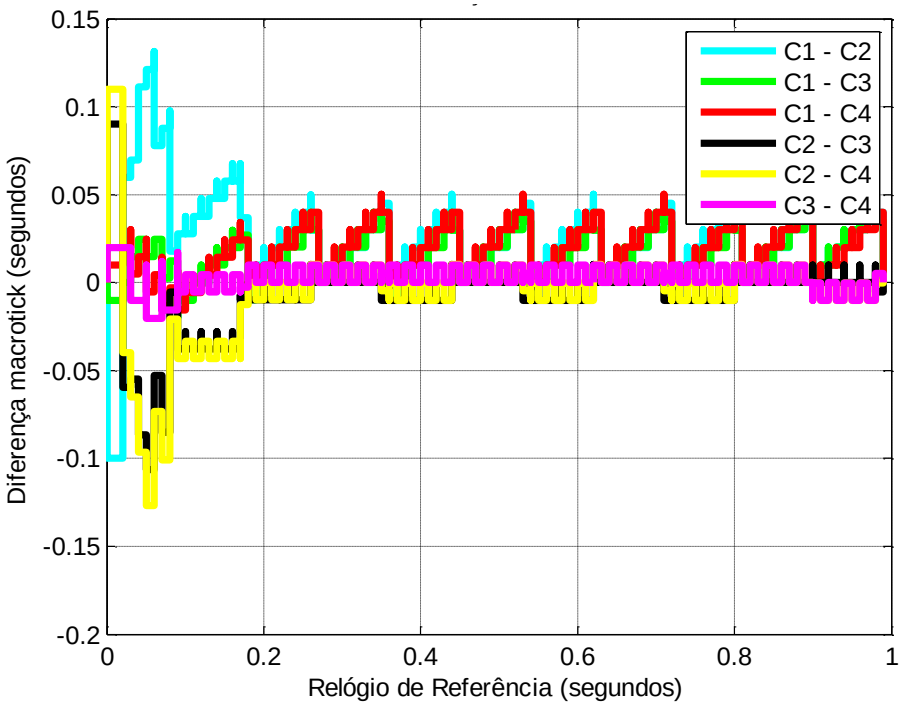


Figura 8.95 – Caso 19 - Diferença entre Macroticks.

A Figura 8.96 mostra o valor de correção do relógio calculado pelo controlador *deadbeat* no modo STM e a Figura 8.97 a correção calculada pela função de convergência do algoritmo PTP no modo NOMO.

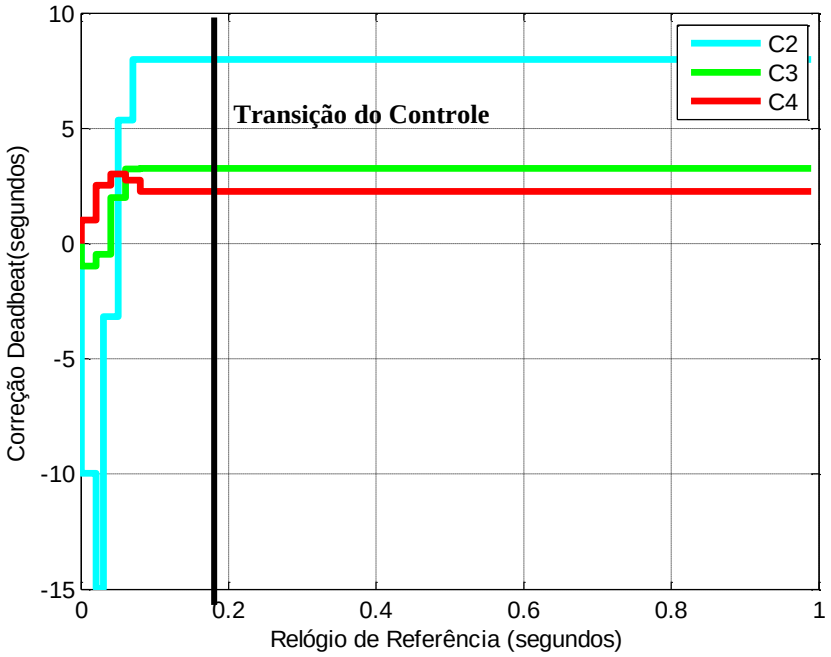


Figura 8.96 – Caso 19 - Modo STM - Controle *Deadbeat*.

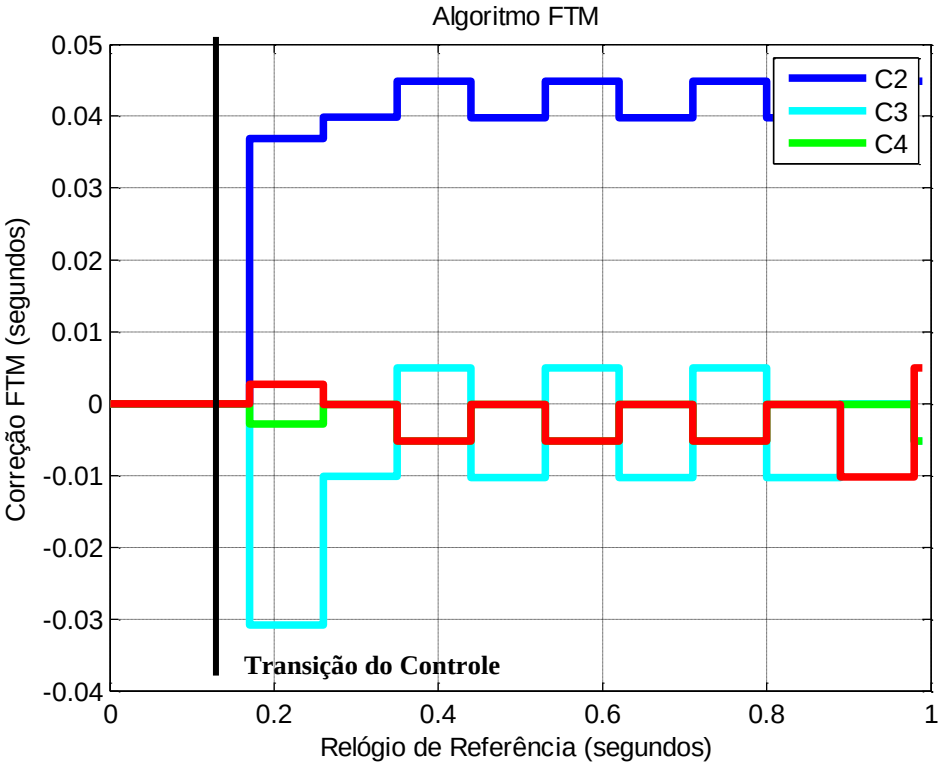


Figura 8.97 – Caso 19 - Correção modo NOMO - Algoritmo FTM.

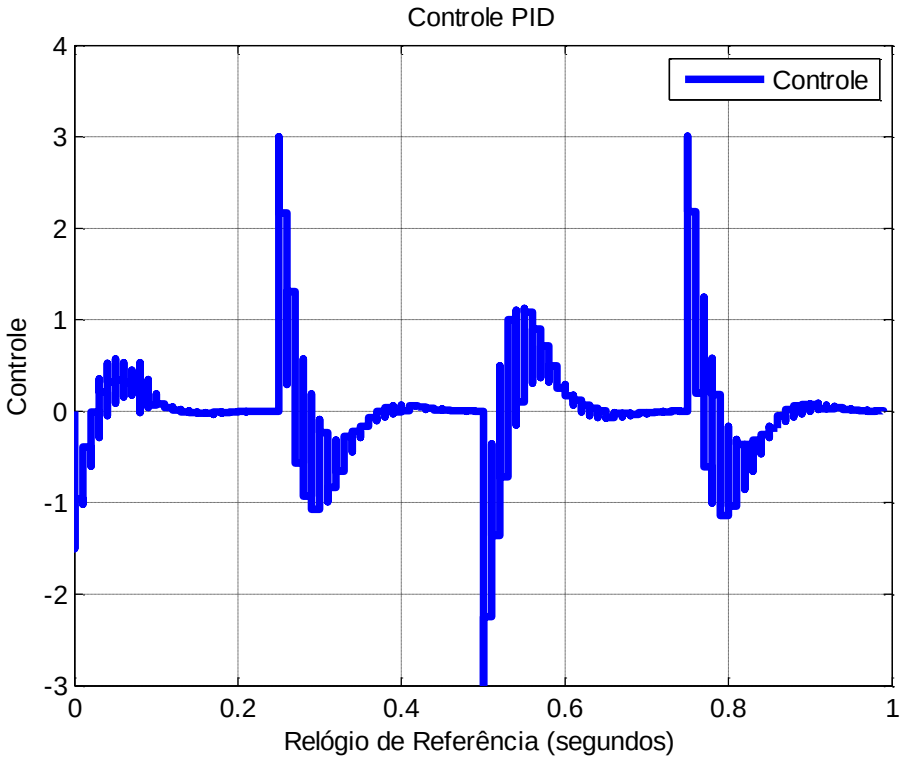


Figura 8.98 – Caso 19 - Controle PID.

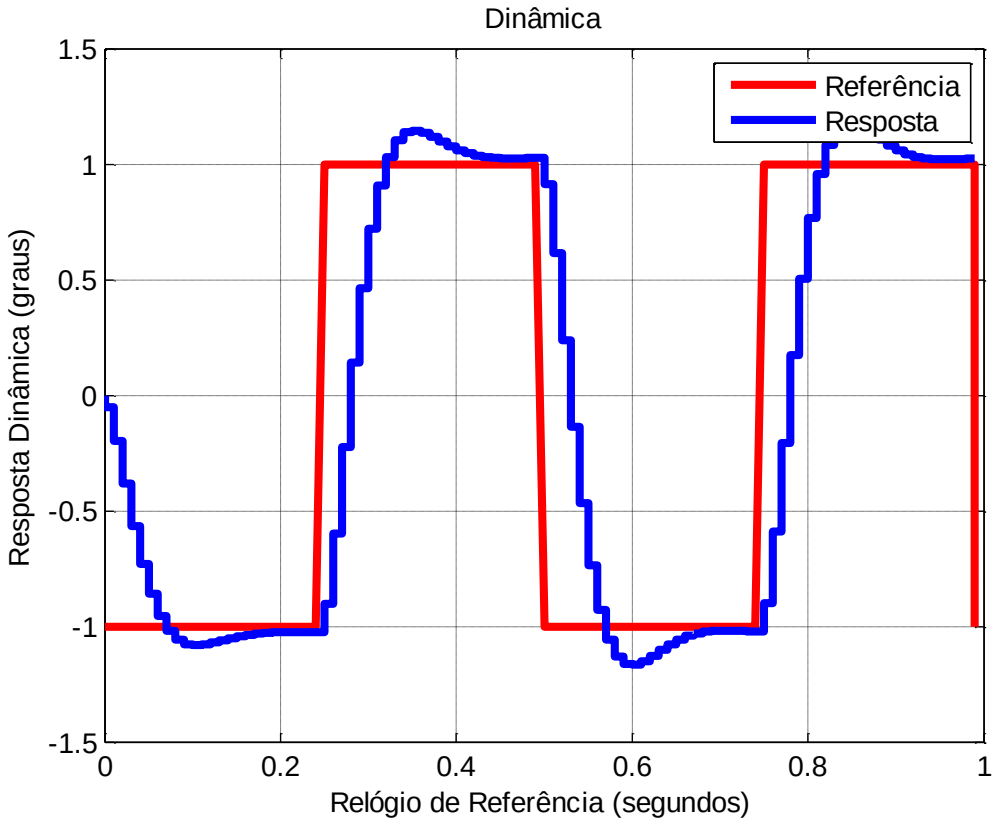


Figura 8.99 – Caso 19 - Resposta Dinâmica.

O tempo de transição dos modos de correção é de 10 macroticks e o período de re-sincronização é de 10 microticks, ou seja, após 10 macroticks, a correção passa a ser feita pelo PTP que a cada 10 microticks aplica-se o processo de sincronização do algoritmo.

Este caso mostra uma sincronização de relógios com o controlador *deadbeat* no modo STM e o algoritmo FTM no modo NOMO, sobre um sistema de controle por redes em uma rede CSMA/CD. No modo STM, não é realizado o controle da dinâmica, pois é um modo de inicialização de tempo. No modo NOMO o algoritmo de sincronização se mostrou eficiente para estabelecer a sincronização. Além disso, a sincronização de tempo melhorou, em relação ao caso anterior, a resposta dinâmica, apresentando um sobressinal médio (em torno de 20% a 40%). No sistema de controle. A de-sincronização inicial que antes levou o algoritmo a gerar sobre sinais altos no sistema de controle, agora

foi corrigida aplicando a reconfiguração dos algoritmos de tempo e utilizando o *deadbeat*. Esta técnica se mostrou eficiente e os resultados podem ser melhorados através da otimização dos parâmetros.

8.22. Caso 20 - Algoritmo PReS Degrau - CSMA/CD

Este caso possui valores de viés inicial igual a: C1: 0; C2: 0.1 seg. ; C3: 0.01 seg. ; C4: -0.1 seg.; e deriva igual a: C1: 0; C2: 1%; C3: 0.01%; C4: -0.01%. Neste caso, aplica-se um algoritmo com os dois modos de correção: 1) STM e o 2) NOMO, descritos no capítulo 6. Com isto, visa-se inicialmente corrigir o viés inicial com o controlador *deadbeat* e depois utiliza-se a técnica FTM em conjunto com uma função *cross-fading* degrau, definindo o algoritmo PReS degrau, sobre um sistema de controle por redes em uma rede CSMA/CD. A Figura 8.100 e Figura 8.101 mostram, respectivamente, os valores de macrotick e diferença entre o relógio de referência e os macroticks. A Figura 8.102 mostra o valor de correção do relógio calculado pelo controlador *deadbeat* no modo STM. A Figura 8.103 mostra a função *cross-fading* degrau.

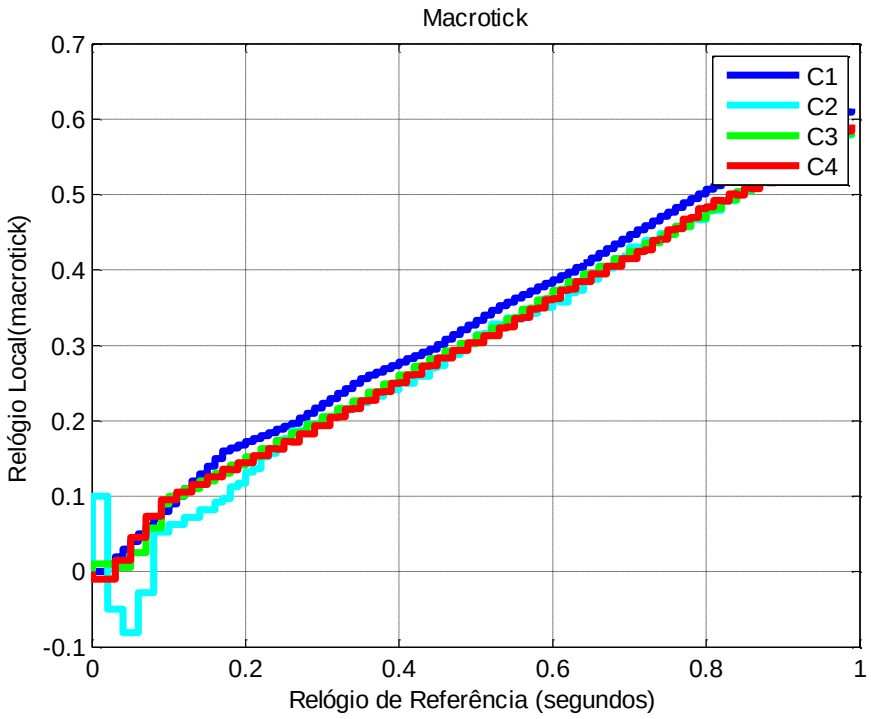


Figura 8.100 – Caso 20 - Macrotick.

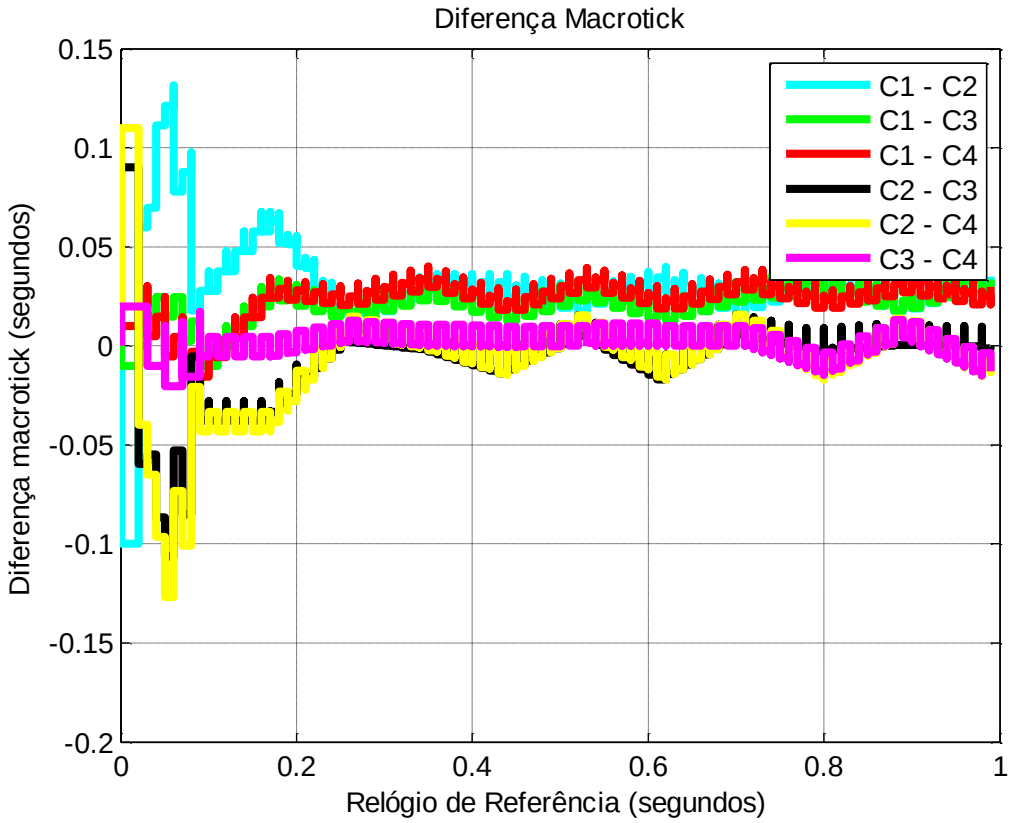


Figura 8.101 – Caso 20 - Diferença entre Macroticks.

A Figura 8.104 mostra a correção calculada pela função de convergência do algoritmo PTP mais a função *cross-fading* degrau. O tempo de transição entre os modos de correção é de 10 macroticks (existe uma transição entre o STM e NOMO após 10 macroticks) e o período de re-sincronização é de 10 microticks, ou seja, após 10 macroticks, a correção é recalculada. No entanto, com o uso da função degrau, a cada instante é aplicada uma correção multiplicada pelo valor da função *cross-fading*.

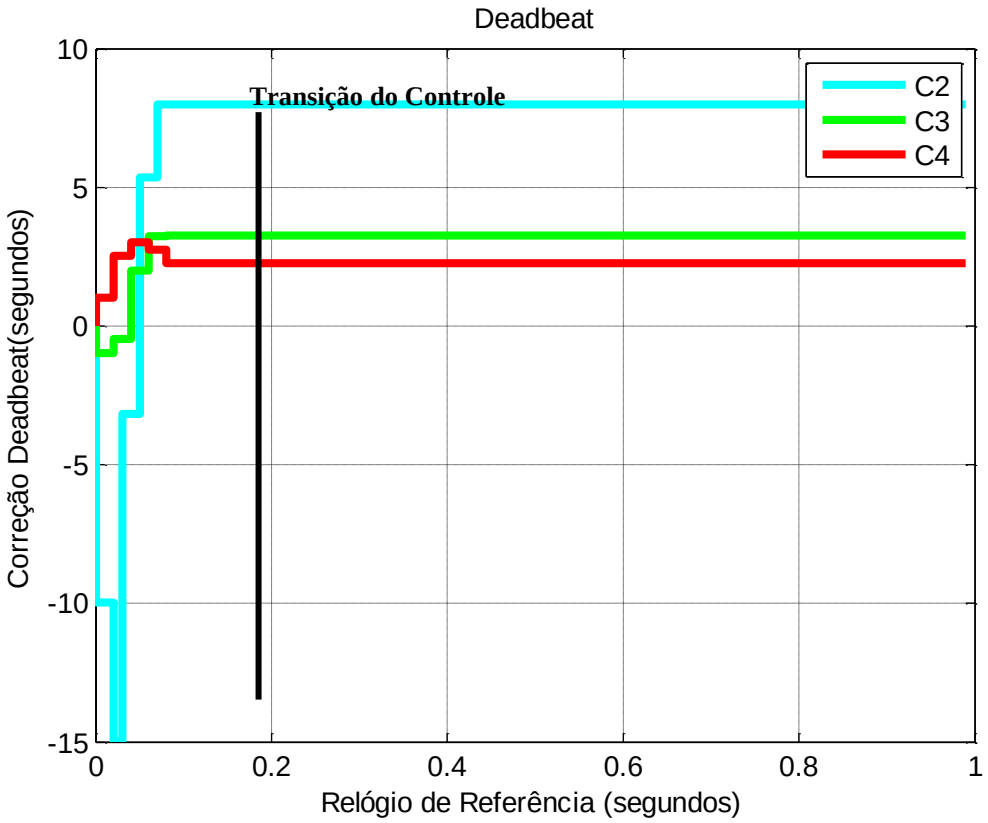


Figura 8.102 – Caso 20 - Modo STM - Algoritmo PReS Degrau.

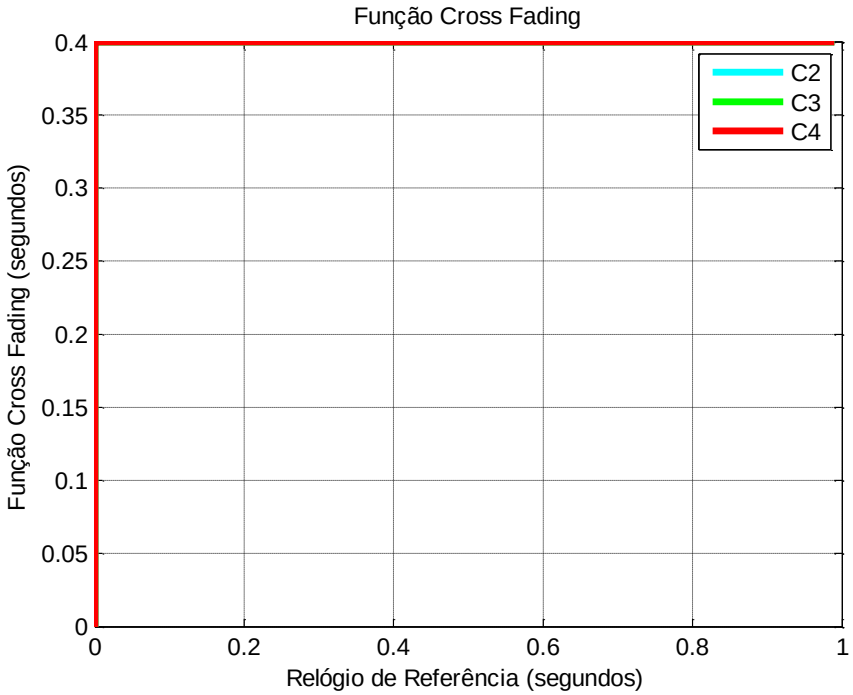


Figura 8.103 – Caso 20 - Função Cross-Fading Degrau.

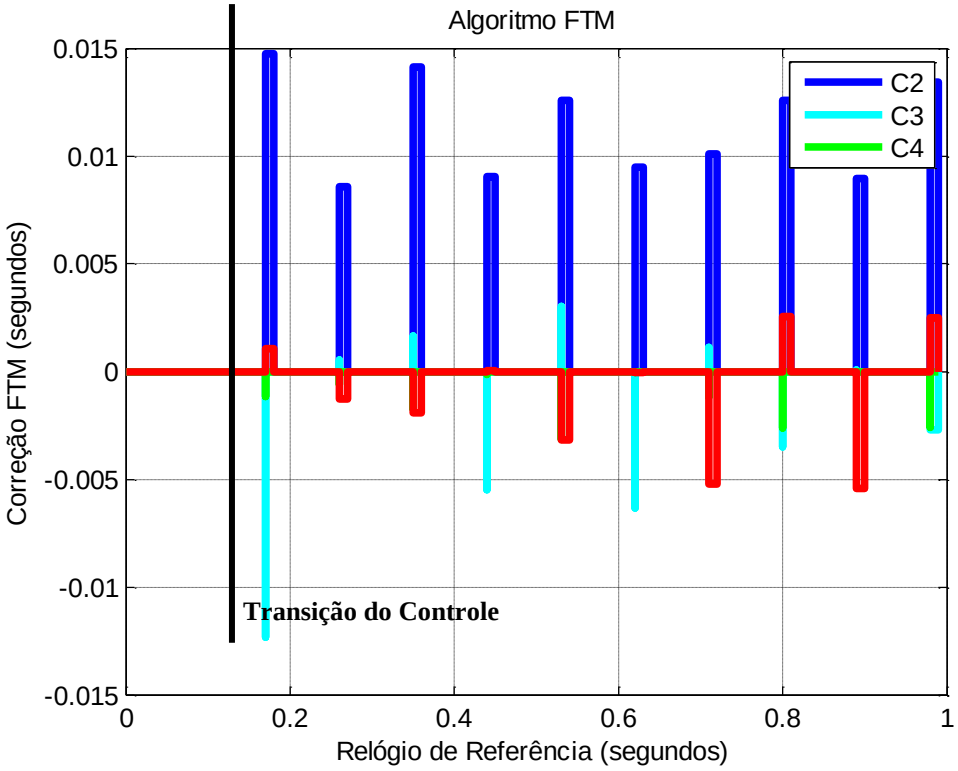


Figura 8.104 – Caso 20 - Correção modo NOMO - Algoritmo PReS Degrau.

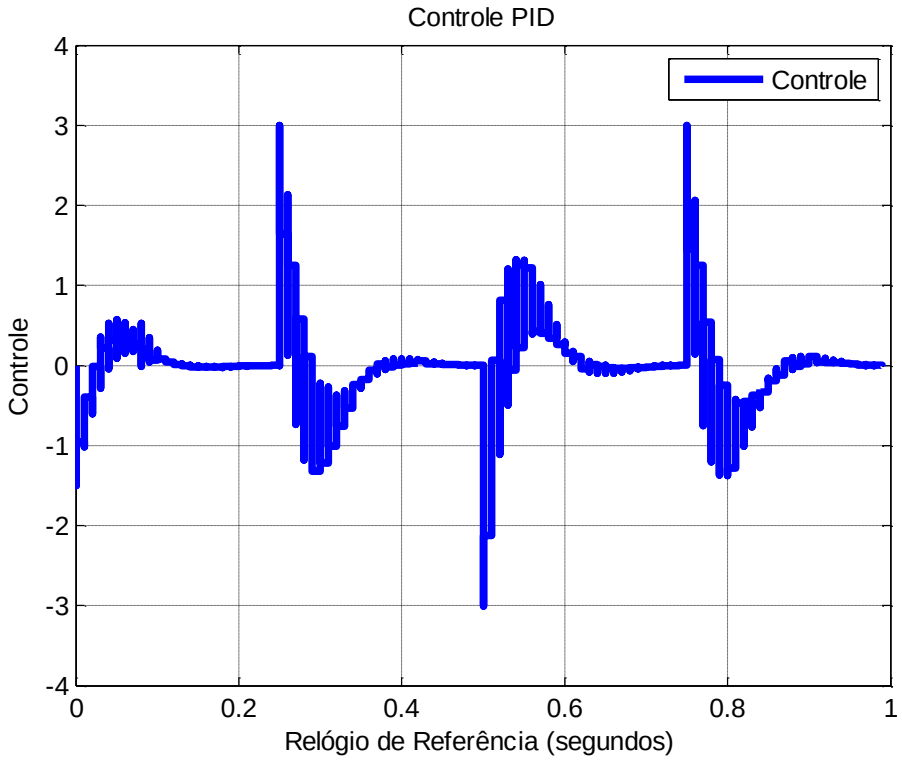


Figura 8.105 – Caso 20 - Controle PID.

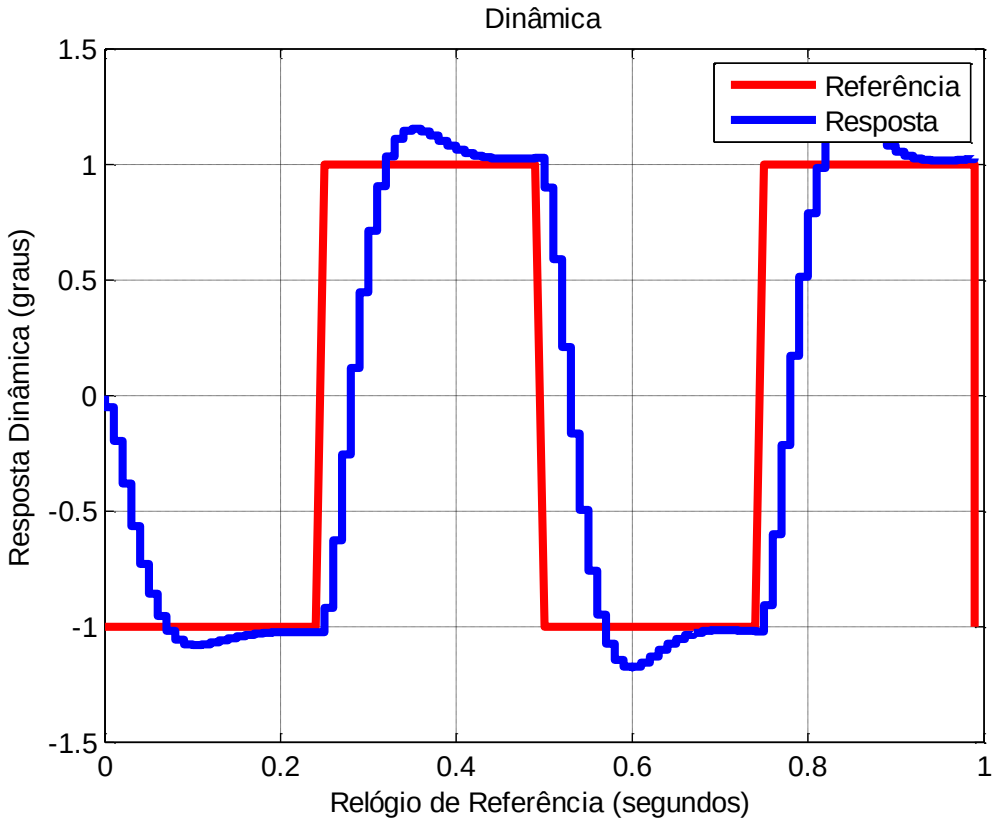


Figura 8.106 – Caso 20 - Resposta Dinâmica.

Este caso mostra uma sincronização de relógios com o algoritmo PReS degrau que utiliza um controlador *deadbeat* no modo STM e o algoritmo FTM com *cross-fading* degrau no modo NOMO, sobre um sistema de controle por redes em uma rede CSMA/CD. No modo STM, não é realizado o controle da dinâmica, pois é um modo de inicialização de tempo. No modo NOMO o algoritmo de sincronização se mostrou eficiente para estabelecer a sincronização. No entanto, este caso obteve uma resposta dinâmica melhor com um sobressinal do no sistema de controle menor que o caso anterior. Isto ocorre, pois a de-sincronização inicial que antes levou o algoritmo a gerar uma instabilidade no sistema de controle, agora foi corrigida aplicando a reconfiguração dos algoritmos de tempo; além disso, o algoritmo suaviza a descontinuidade de tempo. Esta técnica se mostrou eficiente para sincronizar relógios em sistemas de controle por redes e pode ser melhorada através da otimização dos parâmetros.

8.23. Caso 21 - Algoritmo PReS Rampa - CSMA/CD

Este caso possui valores de viés inicial igual a: C1: 0; C2: 0.1 seg. ; C3: 0.01 seg. ; C4: -0.1 seg.; e deriva igual a: C1: 0; C2: 1%; C3: 0.01%; C4: -0.01%.

Neste caso, aplica-se um algoritmo com os dois modos de correção: 1) STM e o 2) NOMO, descritos no capítulo 6. Com isto, visa-se inicialmente corrigir o viés inicial com o controlador *deadbeat* e depois utiliza-se a técnica FTM em conjunto com uma função *cross-fading* rampa, definindo o algoritmo PReS rampa, sobre um sistema de controle por redes em uma rede CSMA/CD.

A Figura 8.107 e Figura 8.108 mostram, respectivamente, os valores de macrotick e diferença entre o relógio de referência e os macroticks. A Figura 8.109 mostra o valor de correção do relógio calculado pelo controlador *deadbeat* no modo STM. A Figura 8.110 mostra a função *cross-fading* degrau.

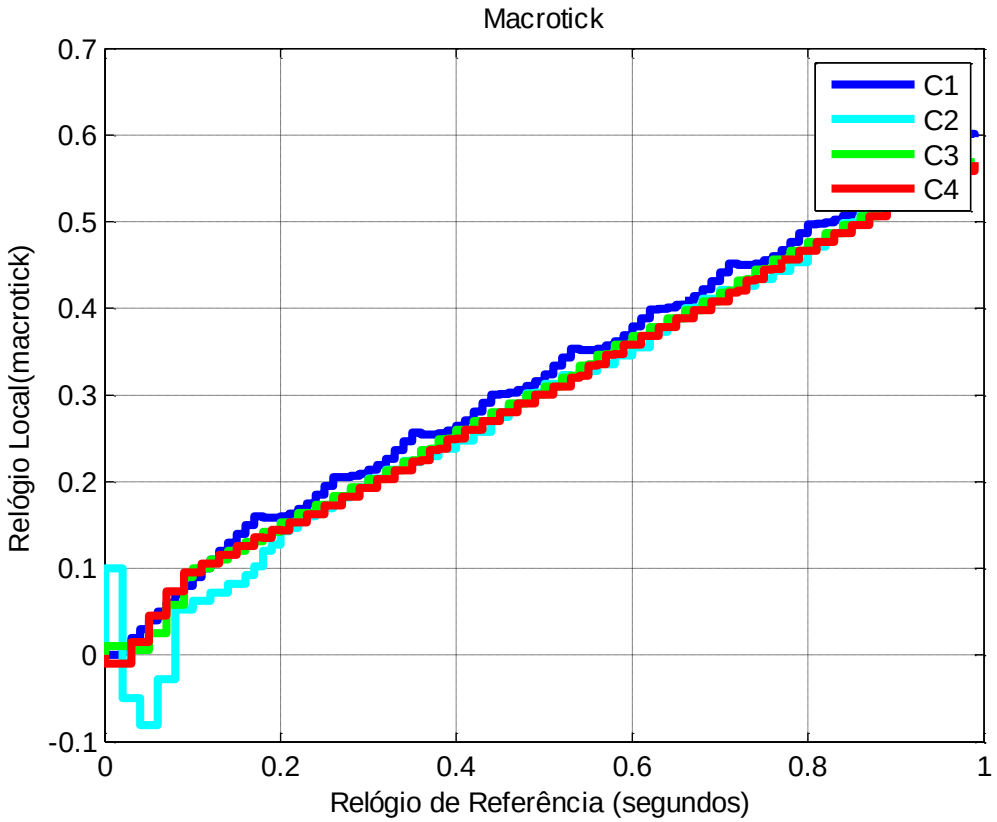


Figura 8.107 – Caso 21 - Macrotick.

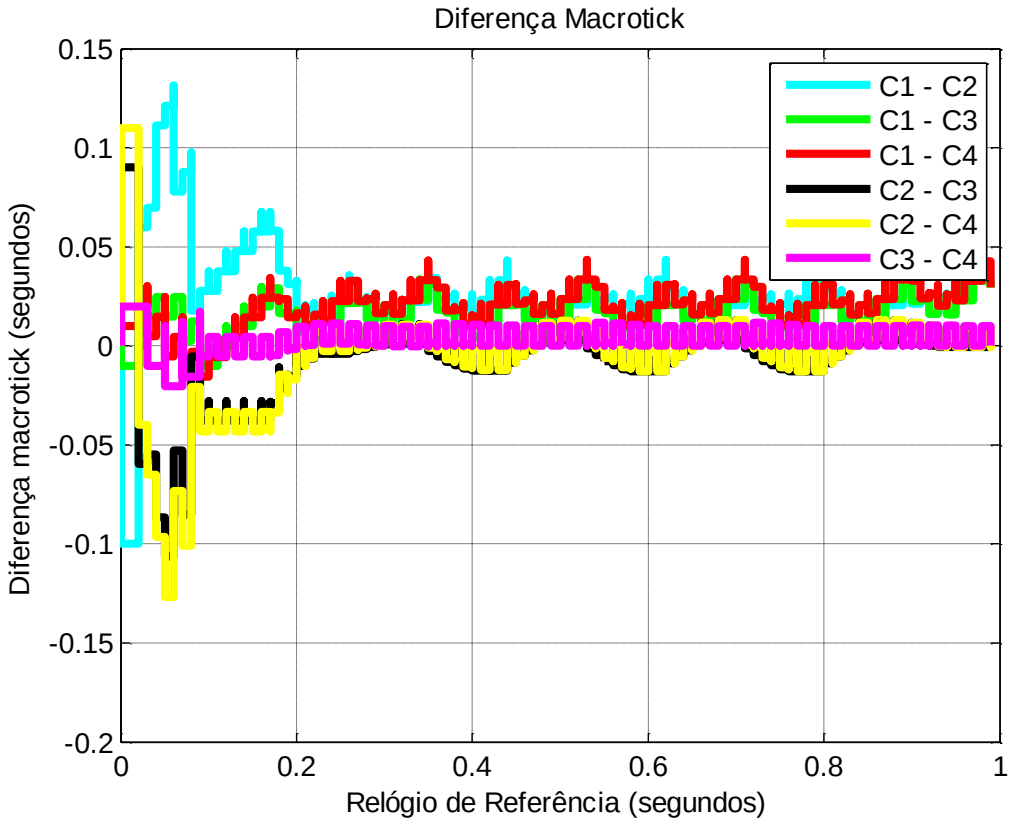


Figura 8.108 – Caso 21 - Diferença entre Macroticks.

A Figura 8.57 mostra a correção calculada pela função de convergência do algoritmo PTP mais a função *cross-fading* rampa. O tempo de transição entre os modos de correção é de 10 macroticks (existe uma transição entre o STM e NOMO após 10 macroticks) e o período de re-sincronização é de 10 microticks, ou seja, após 10 macroticks, a correção é recalculada. No entanto, com o uso da função rampa, a cada instante é aplicada uma correção multiplicada pelo valor da função *cross-fading*. No caso da reta, é necessária uma segunda transição para evitar que o valor da função *cross-fading* rampa fique negativa e inverta a fase do sistema tornando-o instável.

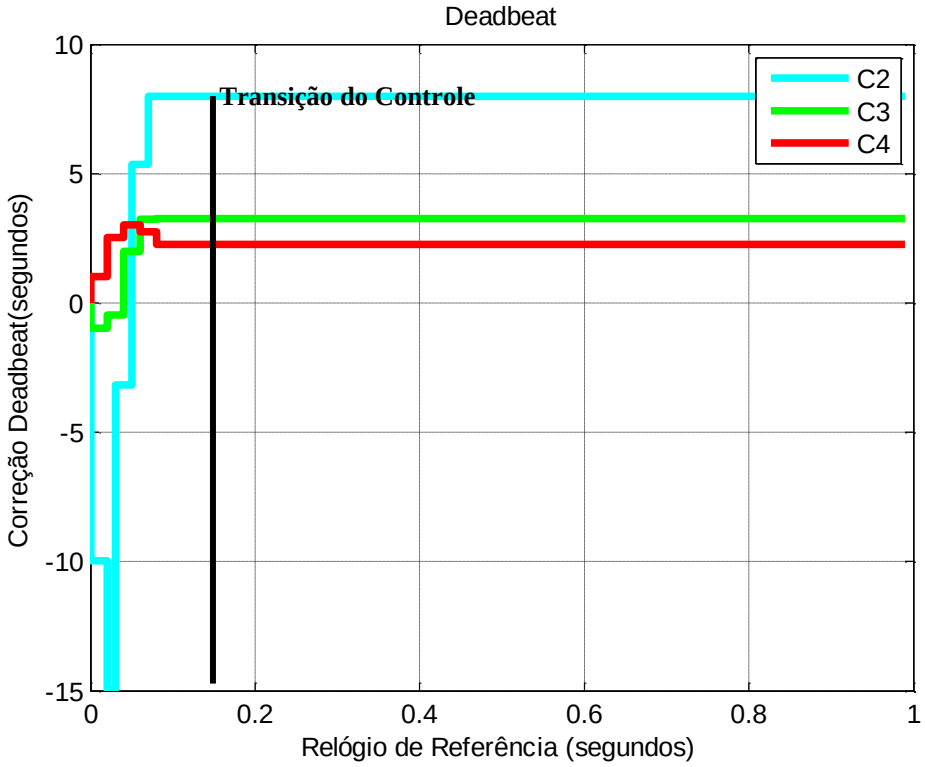


Figura 8.109 – Caso 21 - Modo STM - Algoritmo PReS Rampa.

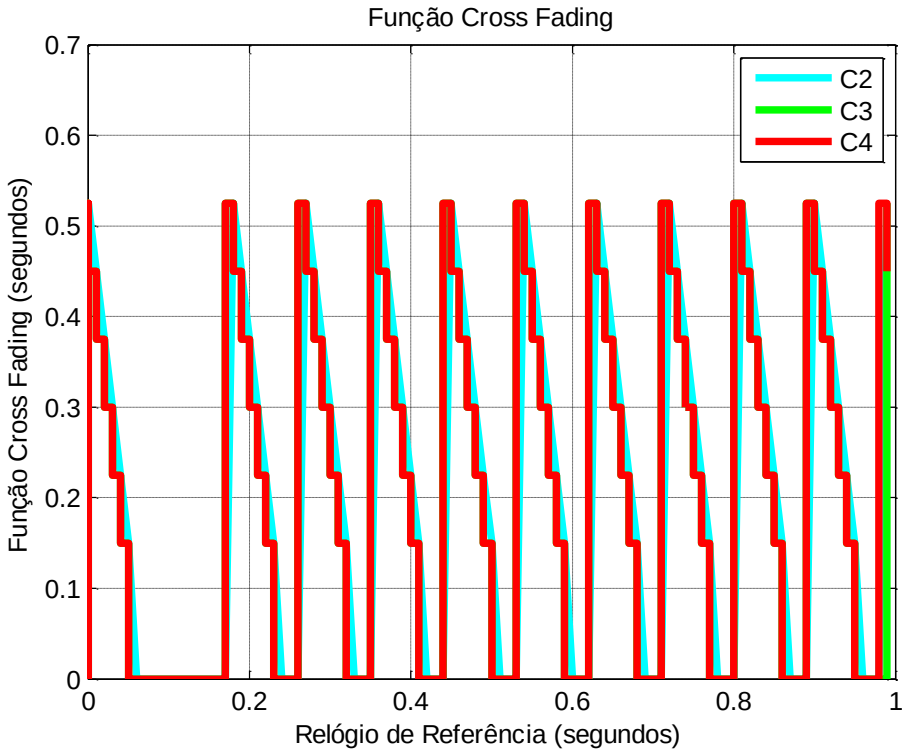


Figura 8.110 – Caso 21 - Função Cross-Fading Rampa.

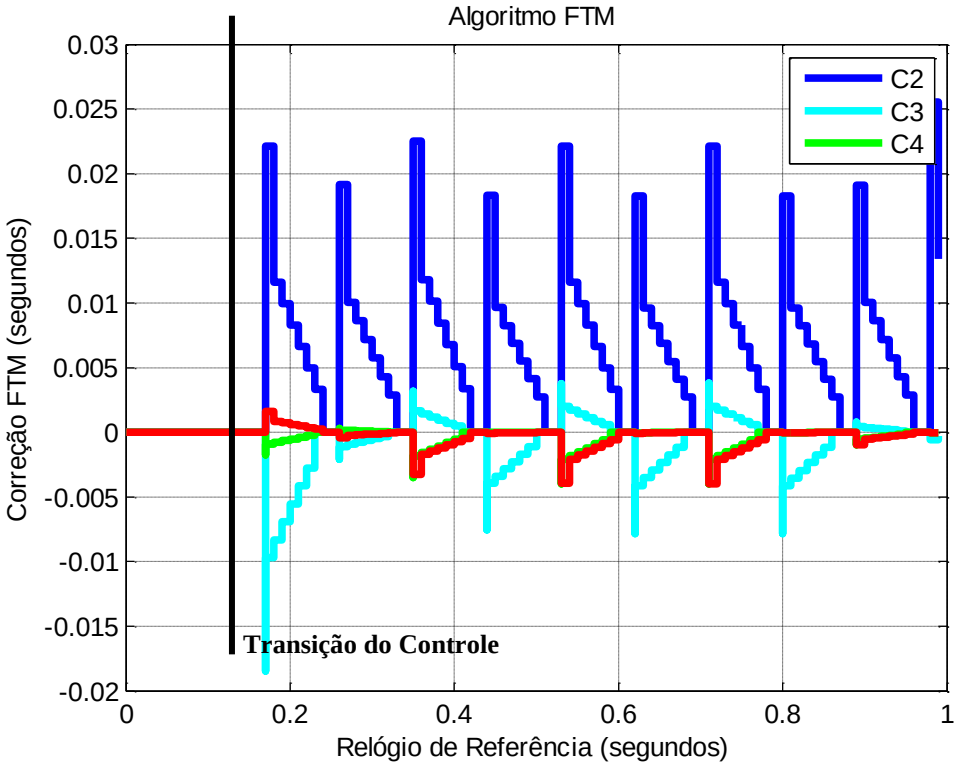


Figura 8.111 – Caso 21 - Correção modo NOMO - Algoritmo PReS Rampa.

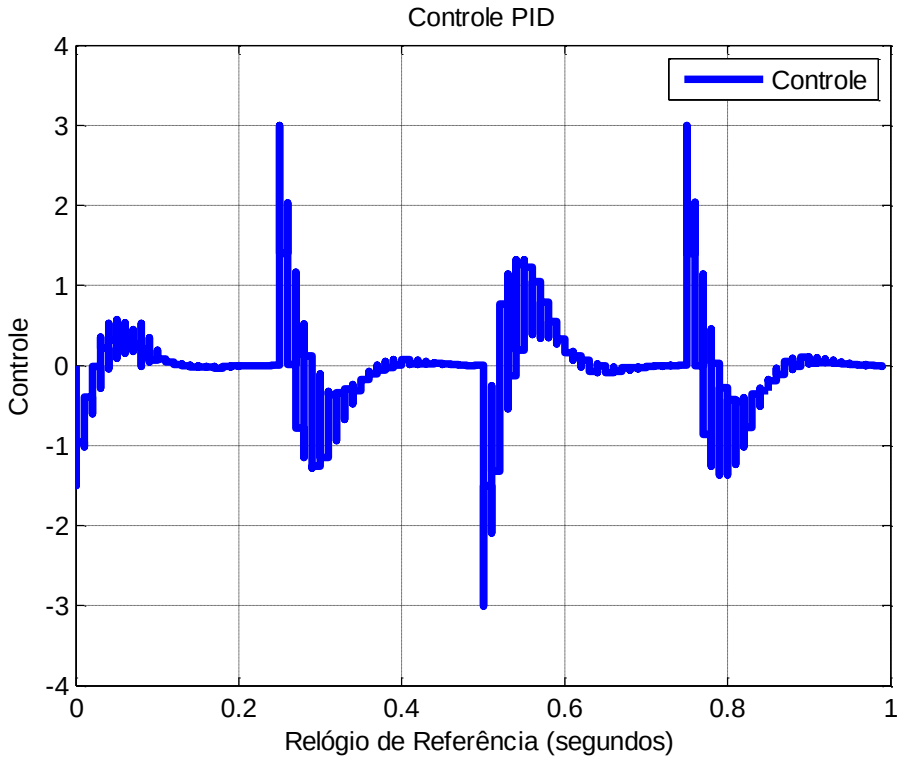


Figura 8.112 – Caso 21 - Controle PID.

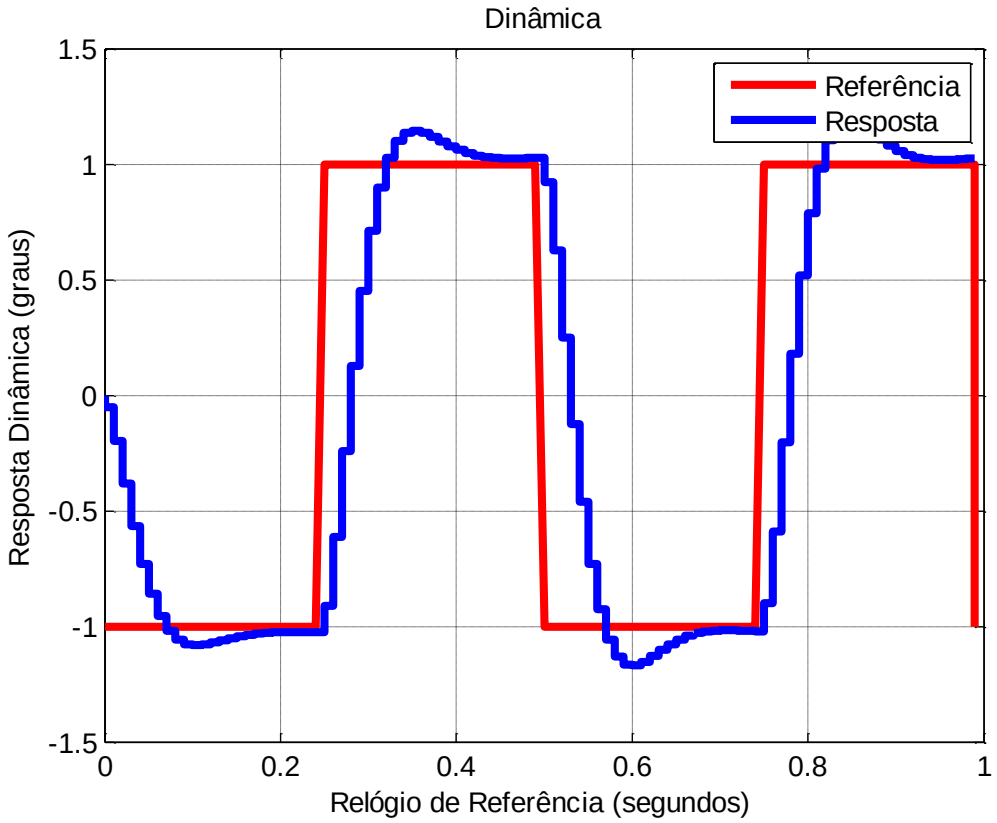


Figura 8.113 – Caso 21 - Resposta Dinâmica.

Este caso mostra uma sincronização de relógios com o algoritmo PReS rampa que utiliza um controlador *deadbeat* no modo STM e o algoritmo FTM com *cross-fading* rampa no modo NOMO, sobre um sistema de controle por redes em uma rede CSMA/CD. No modo STM, não é realizado o controle da dinâmica, pois é um modo de inicialização de tempo. No modo NOMO o algoritmo de sincronização se mostrou eficiente para estabelecer a sincronização. No entanto, devido à sincronização de tempo a resposta dinâmica apresentou uma melhora no sobressinal. Isto ocorre, pois o método suaviza a descontinuidade de tempo. A de-sincronização inicial que antes levou o algoritmo a gerar um sobre sinal no sistema de controle, agora foi corrigida aplicando a reconfiguração dos algoritmos de tempo e utilizando o *deadbeat*. Esta técnica se mostrou eficiente para sincronizar relógios em sistemas de controle por redes e pode ser melhorada através da otimização dos parâmetros.

8.24. Caso 22 - Algoritmo PReS Exponencial - CSMA/CD

Este caso possui valores de viés inicial igual a: C1: 0; C2: 0.1 seg. ; C3: 0.01 seg. ; C4: -0.1 seg.; e deriva igual a: C1: 0; C2: 1%; C3: 0.01%; C4: -0.01%.

Neste caso, aplica-se um algoritmo com os dois modos de correção: 1) STM e o 2) NOMO, descritos no capítulo 6. Com isto, visa-se inicialmente corrigir o viés inicial com o controlador *deadbeat* e depois utiliza-se a técnica FTM em conjunto com uma função *cross-fading* exponencial, definindo o algoritmo PReS exponencial, sobre um sistema de controle por redes em uma rede CSMA/CD.

A Figura 8.114 e Figura 8.115 mostram, respectivamente, os valores de macrotick e diferença entre os macroticks. A Figura 8.116 mostra o valor de correção do relógio calculado pelo controlador *deadbeat* no modo STM. A Figura 8.117 mostra a função *cross-fading* exponencial.

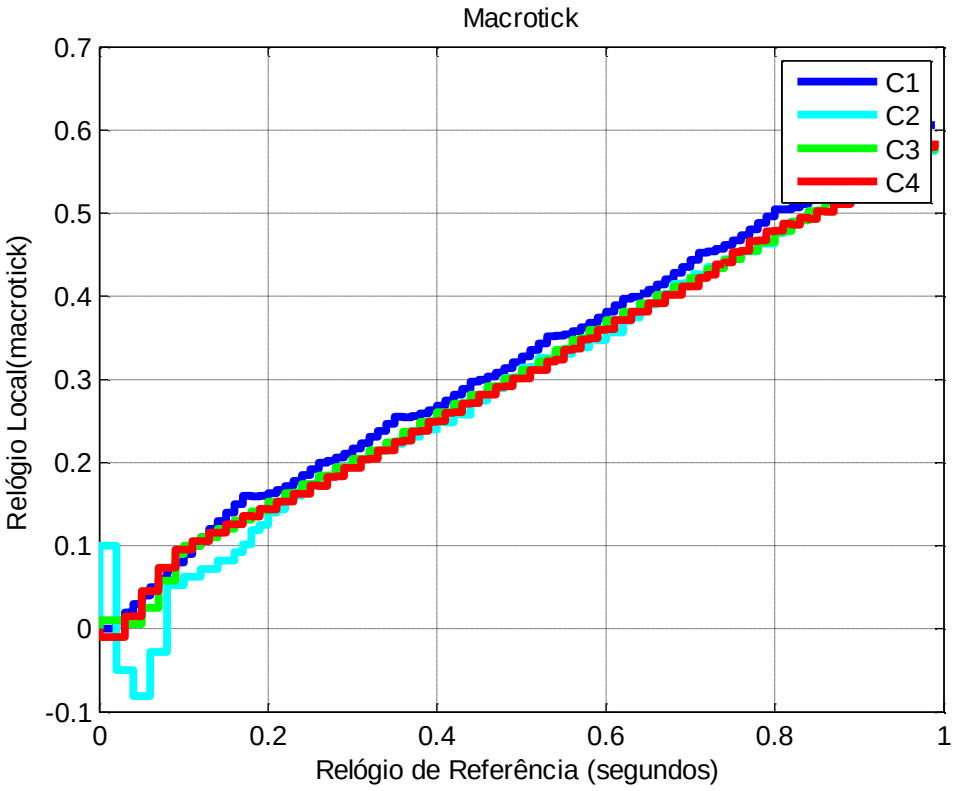


Figura 8.114 – Caso 22 - Macrotick.

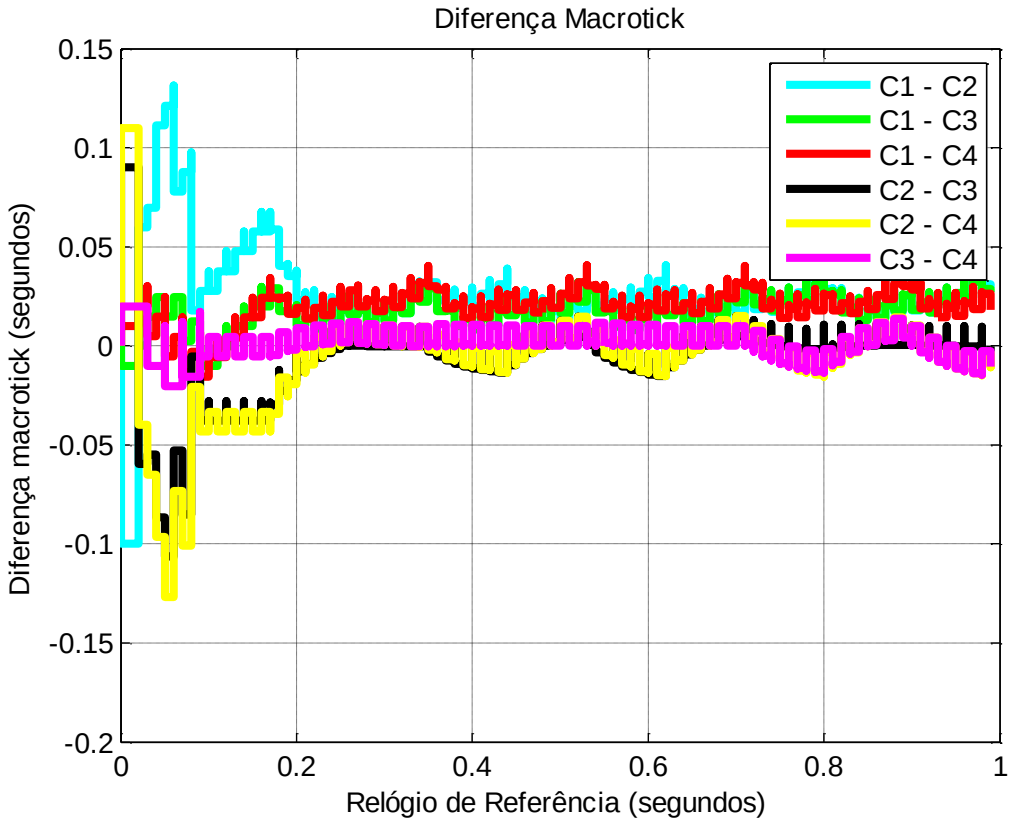


Figura 8.115 – Caso 22 - Diferença entre Macroticks.

A Figura 8.118 mostra a correção calculada pela função de convergência do algoritmo FTM mais a função *cross-fading* exponencial. O tempo de transição entre os modos de correção é de 10 macroticks (existe uma transição entre o STM e NOMO após 10 macroticks) e o período de re-sincronização é de 10 microticks, ou seja, após 10 macroticks, a correção é recalculada. No entanto, com o uso da função exponencial, a cada instante é aplicada uma correção multiplicada pelo valor da função *cross-fading*. No caso da exponencial, não é necessário uma segunda transição para evitar que o valor da função *cross-fading* exponencial fique negativa.

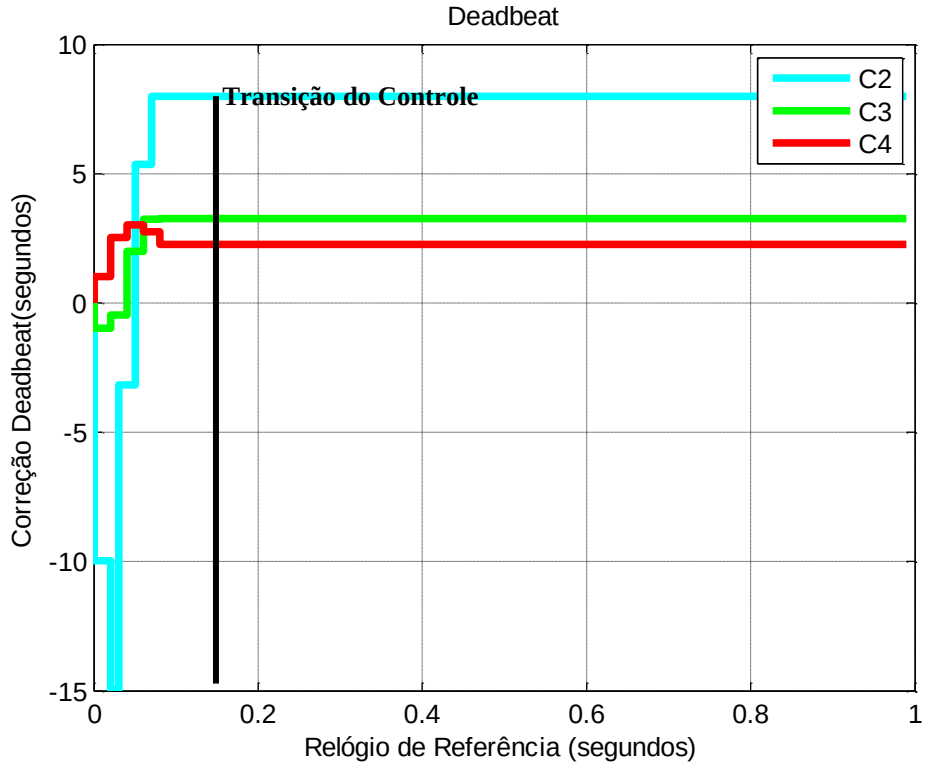


Figura 8.116 – Caso 22 - Modo STM - Algoritmo PReS Exponencial.

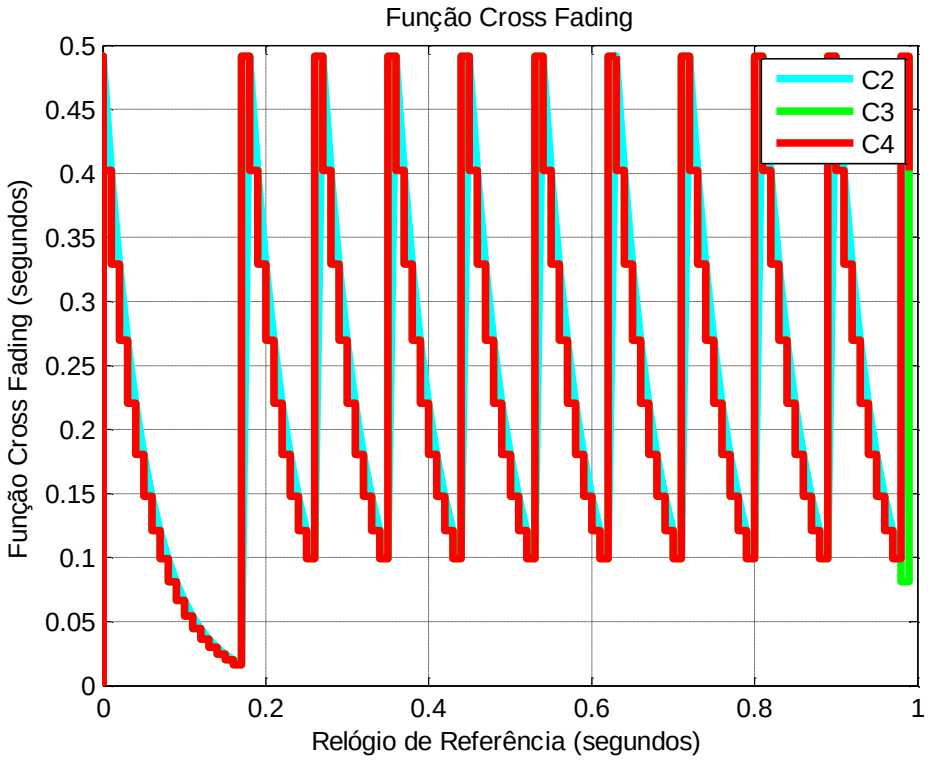


Figura 8.117 – Caso 22 - Função Cross-Fading Rampa.

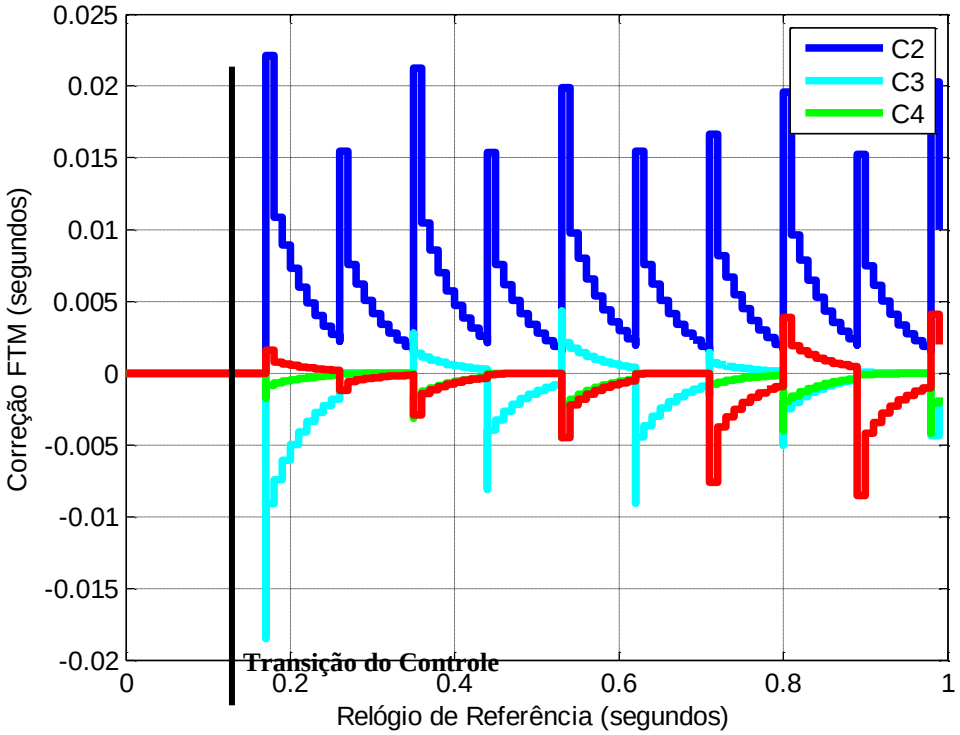


Figura 8.118 – Caso 22 - Correção modo NOMO - Algoritmo PReS Exponencial.

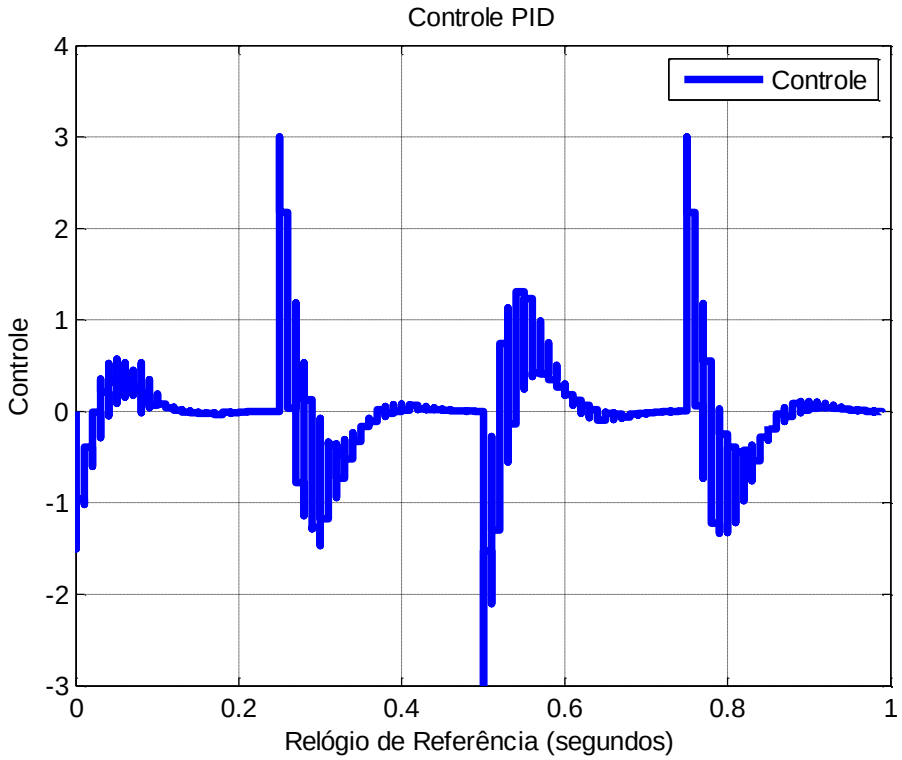


Figura 8.119 – Caso 22 - Controle PID.

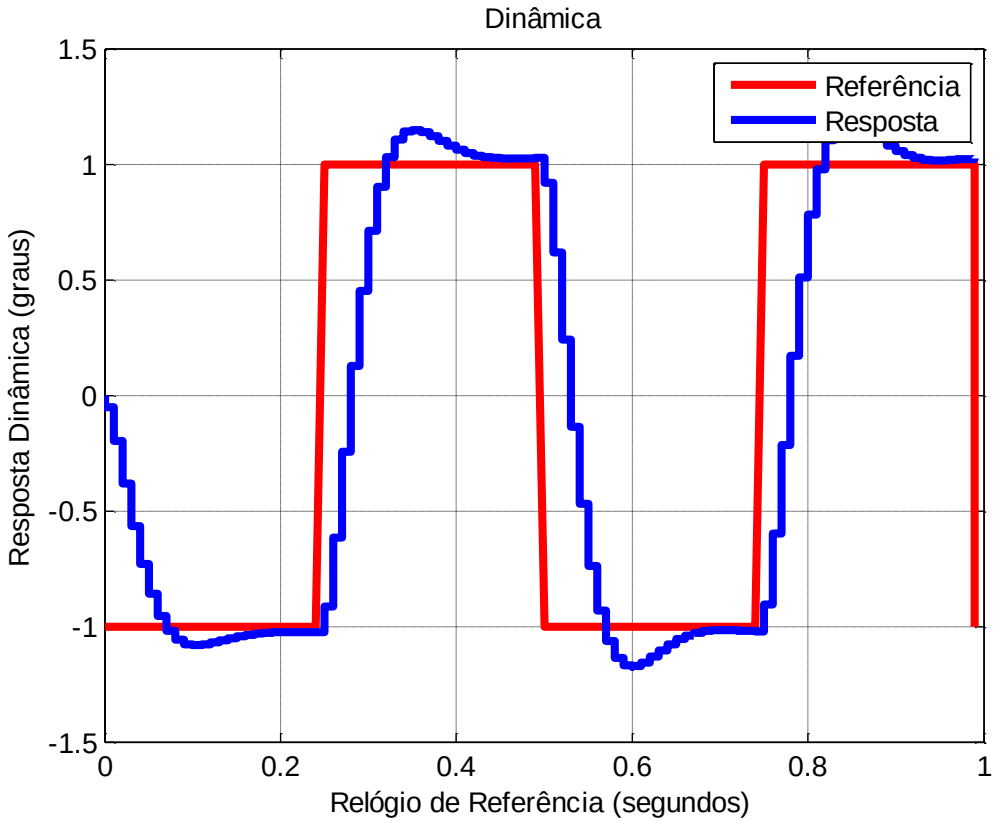


Figura 8.120 – Caso 22 - Resposta Dinâmica.

Este caso mostra uma sincronização de relógios com o algoritmo PReS exponencial que utiliza um controlador *deadbeat* no modo STM e o algoritmo PTP com *cross-fading* exponencial no modo NOMO, sobre um sistema de controle por redes em uma rede CSMA/CD. No modo STM, não é realizado o controle da dinâmica, pois é um modo de inicialização de tempo. No modo NOMO o algoritmo de sincronização se mostrou eficiente para estabelecer a sincronização. No entanto, devido à sincronização de tempo a resposta dinâmica apresentou um sobressinal. Isto ocorre, pois o método suaviza a descontinuidade de tempo. A de-sincronização inicial que antes levou o algoritmo a gerar uma instabilidade no sistema de controle, agora foi corrigida aplicando a reconfiguração dos algoritmos de tempo e utilizando o *deadbeat*. Esta técnica se mostrou eficiente para sincronizar relógios em sistemas de controle por redes e pode ser melhorada através da otimização dos parâmetros.

8.25. Caso 23 - Algoritmo SR

Este caso possui valores de viés inicial igual a: C1: 0; C2: 0.1 seg. ; C3: 0.01 seg. ; C4: -0.1 seg.; e deriva igual a: C1: 0; C2: 1%; C3: 0.01%; C4: -0.01%.

A Figura 8.121 e Figura 8.122 mostram os valores de macrotick e diferença dos macroticks entre si (precisão), respectivamente. Nota-se que nesta abordagem não existe a visão de um relógio de referência real e sim de um relógio virtual. Além disso, este algoritmo aplica duas instâncias de correção, uma do viés e outra do MMCF. Para sincronizar o MMCF, o relógio C1 é a referência para os outros relógios. Observa-se na Figura 8.123 que o relógio C1 possui grandes correções devido a fator de conversor MMCF ser diferente do restante do conjunto. No entanto, quando o MMCF é corrigido, a diferença do conjunto melhora (diminui).

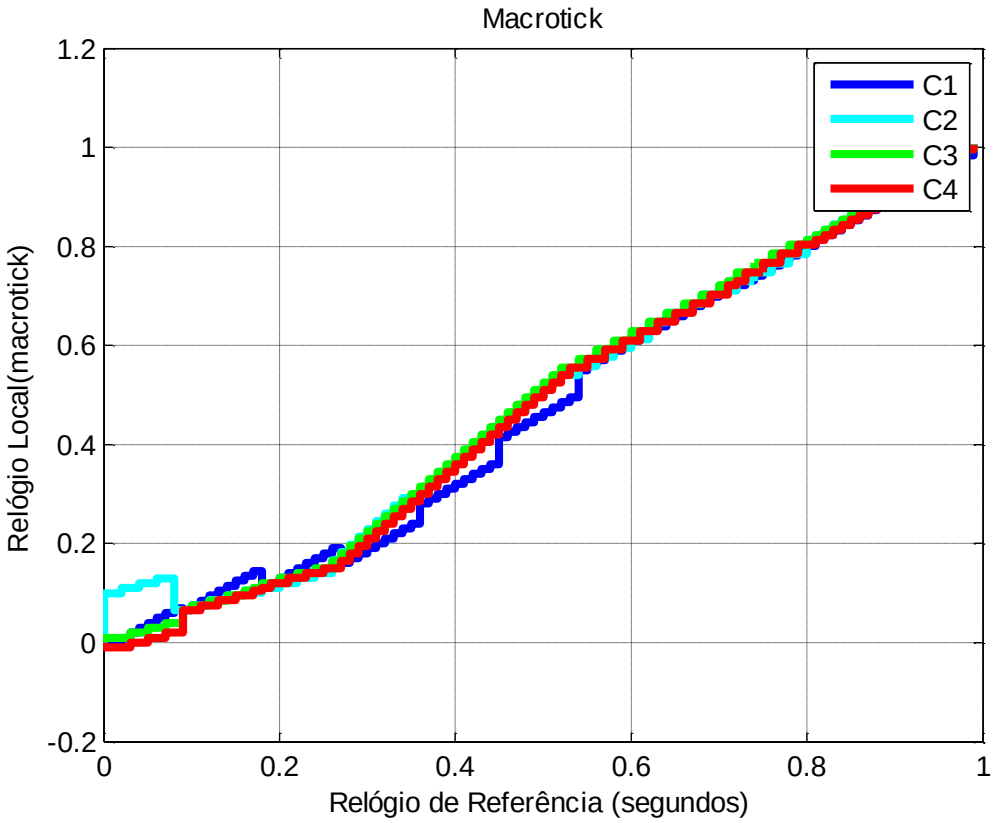


Figura 8.121 – Caso 23 - Macrotick.

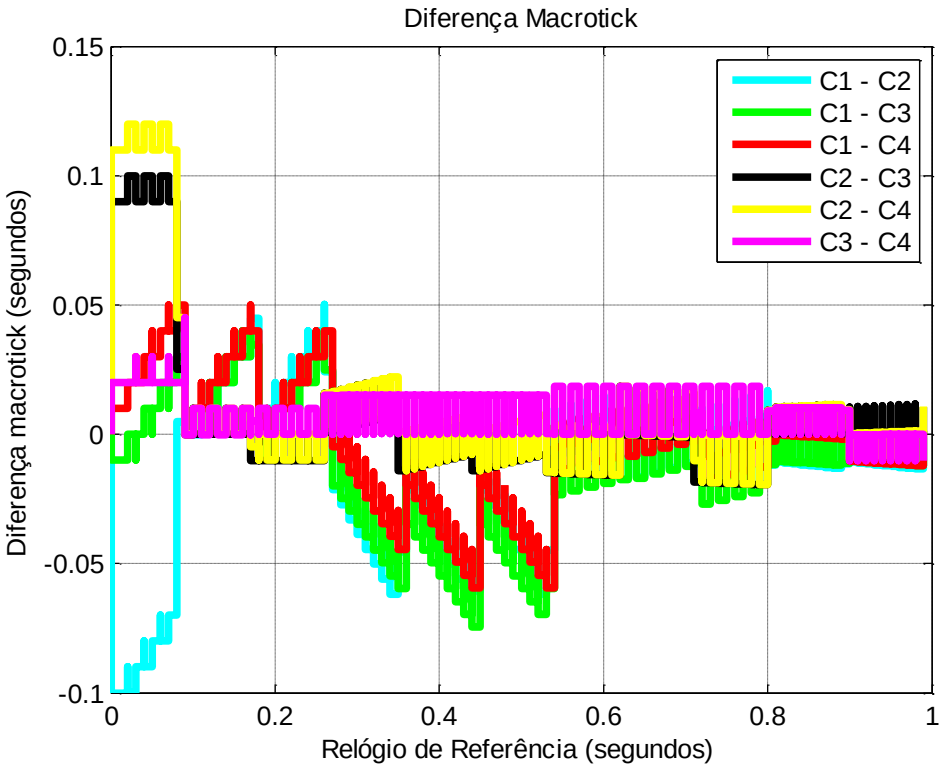


Figura 8.122 – Caso 23 - Diferença entre Macroticks - Precisão.

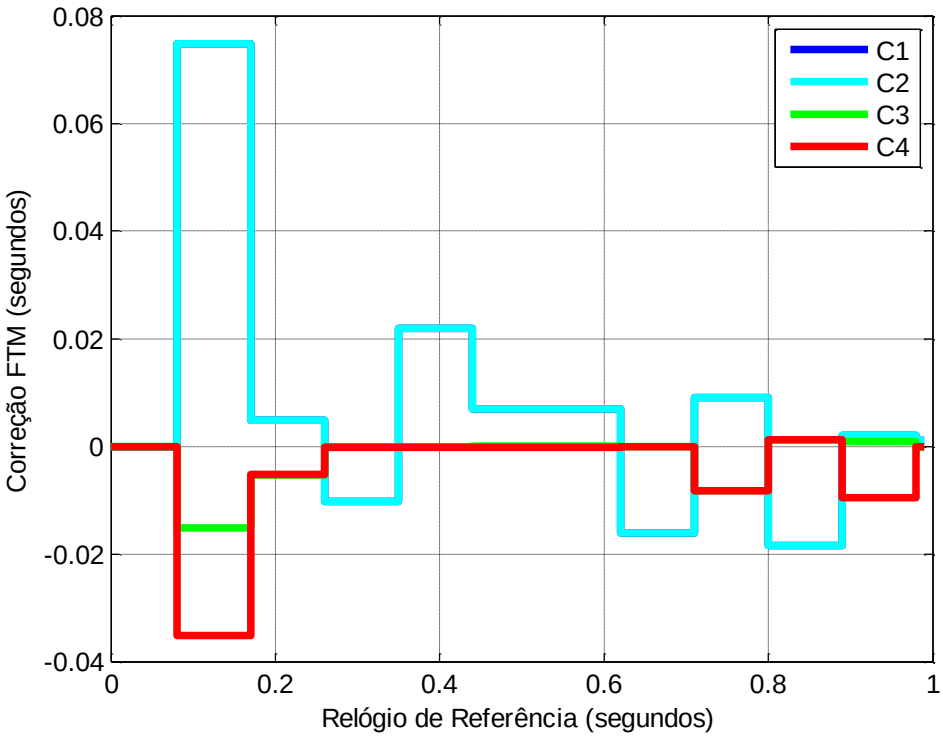


Figura 8.123 – Caso 23 - Correção viés - Algoritmo SR.

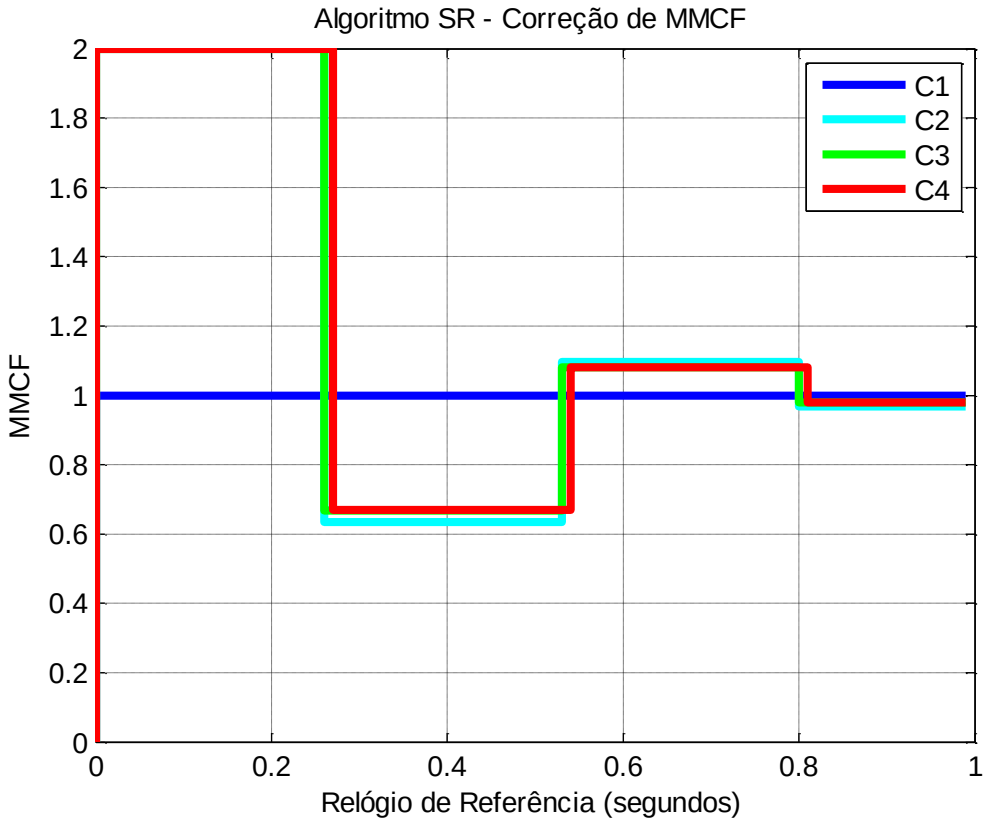


Figura 8.124 – Caso 23 - Correção MMCF - Algoritmo SR.

A Figura 8.123 mostra os valores de correção dos relógios calculado pela função de convergência do algoritmo SR. O período de re-sincronização é de 10 microticks, ou seja, a cada 10 microticks aplica-se o processo de sincronização do algoritmo FTM. A Figura 8.124 mostra os valores do MMCF corrigido pelo algoritmo SR. Ao sincronizar o MMCF, melhora bastante o resultado dos algoritmos de sincronização.

Este caso mostra uma sincronização de relógios com o algoritmo SR. O algoritmo se mostrou eficiente, sendo as principais vantagens a de a sincronização ser uma equação de convergência simples e estabelecer a sincronização do MMCF, melhorando a precisão. A desvantagem é que o algoritmo não possui gerenciamento da descontinuidade de tempo e com isso a descontinuidade de tempo e também não é preparado para levar em conta o viés inicial, apesar de corrigi-lo.

8.26. Caso 24 - Deadbeat - Algoritmo SR

Este caso possui valores de viés inicial igual a: C1: 0; C2: 0.1 seg. ; C3: 0.01 seg. ; C4: -0.1 seg.; e deriva igual a: C1: 0; C2: 1%; C3: 0.01%; C4: -0.01%.

Este caso aplica-se um algoritmo com os três modos de correção: 1) STM; 2) NOMO e o 3) NOMD, descritos no capítulo 6. Com isto, visa-se inicialmente corrigir o viés inicial com o controlador *deadbeat*; depois utiliza-se o algoritmo FTM para estabelecer a sincronização dentro de uma precisão; e enfim, corrige-se os valores de MMCF, utilizando C1 como referência. A Figura 8.125 e Figura 8.126 mostram, respectivamente, os valores de macrotick e diferença entre os macroticks.

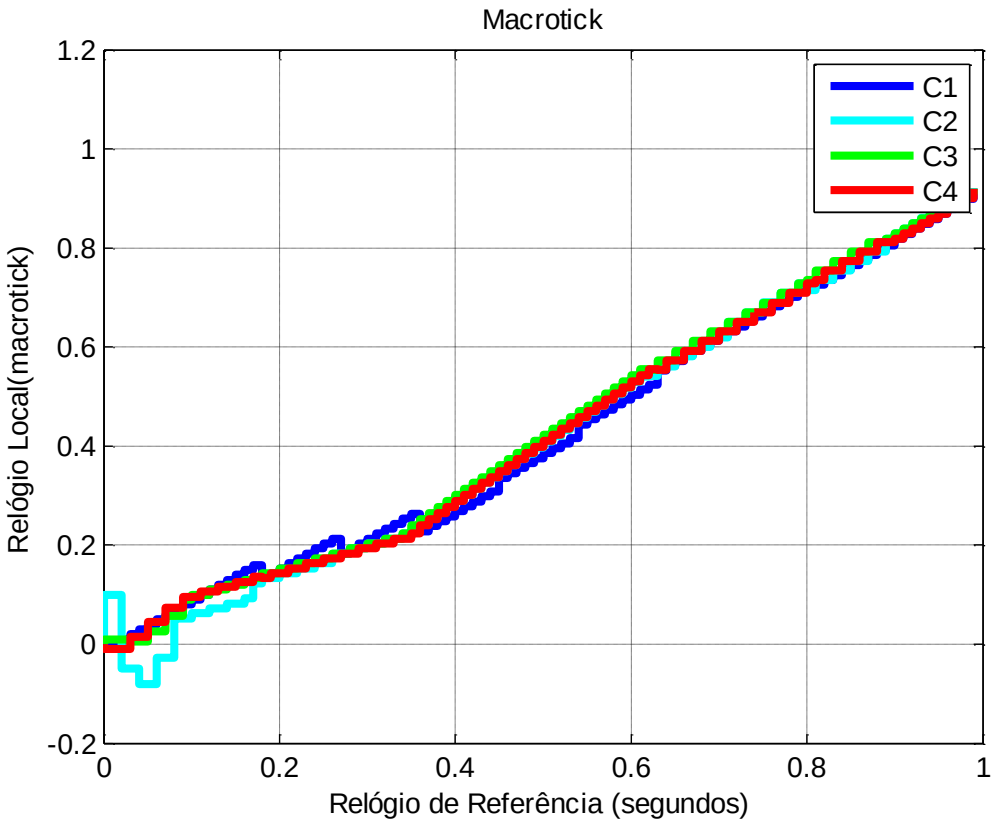


Figura 8.125 – Caso 24 - Macrotick.

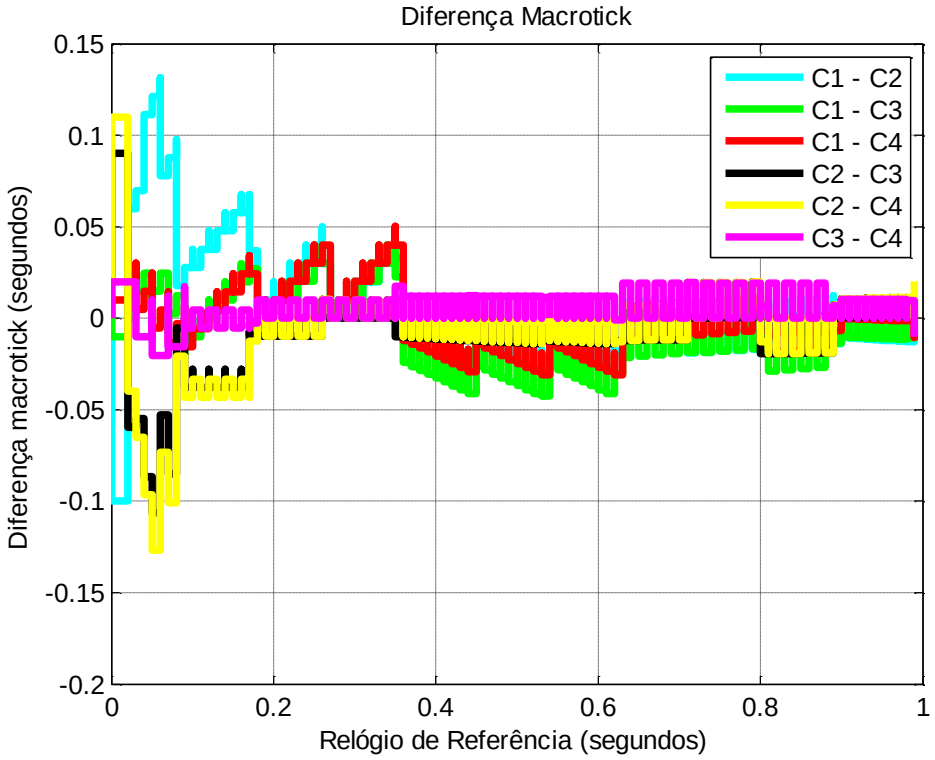


Figura 8.126 – Caso 24 - Diferença entre Macroticks - Precisão.

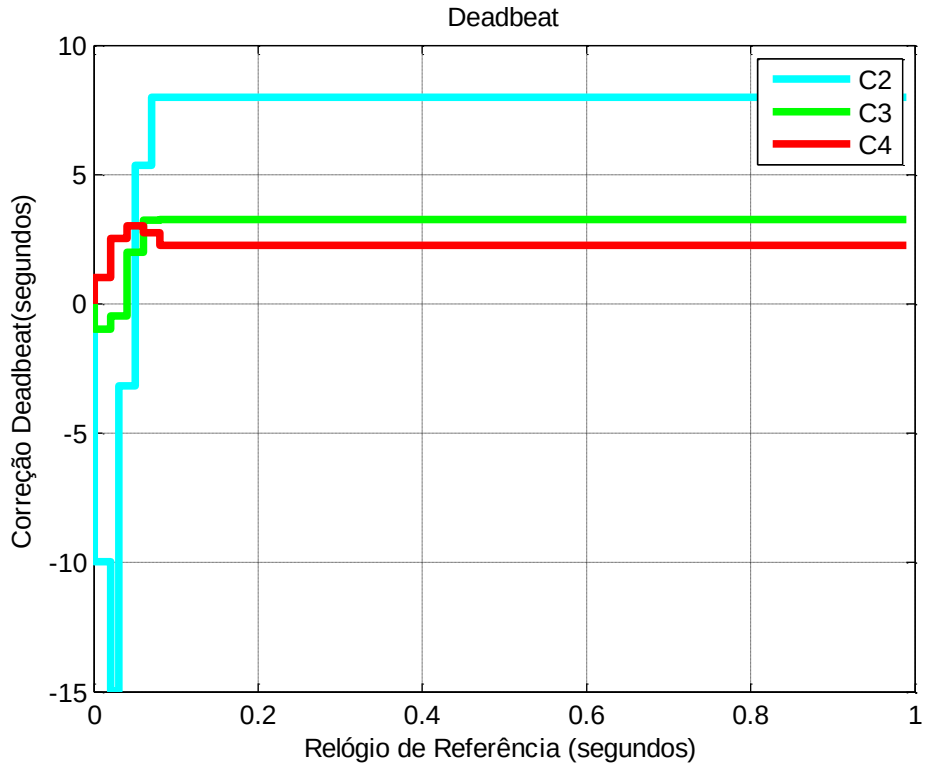


Figura 8.127 – Caso 24 - Correção STM - Controle *Deadbeat*.

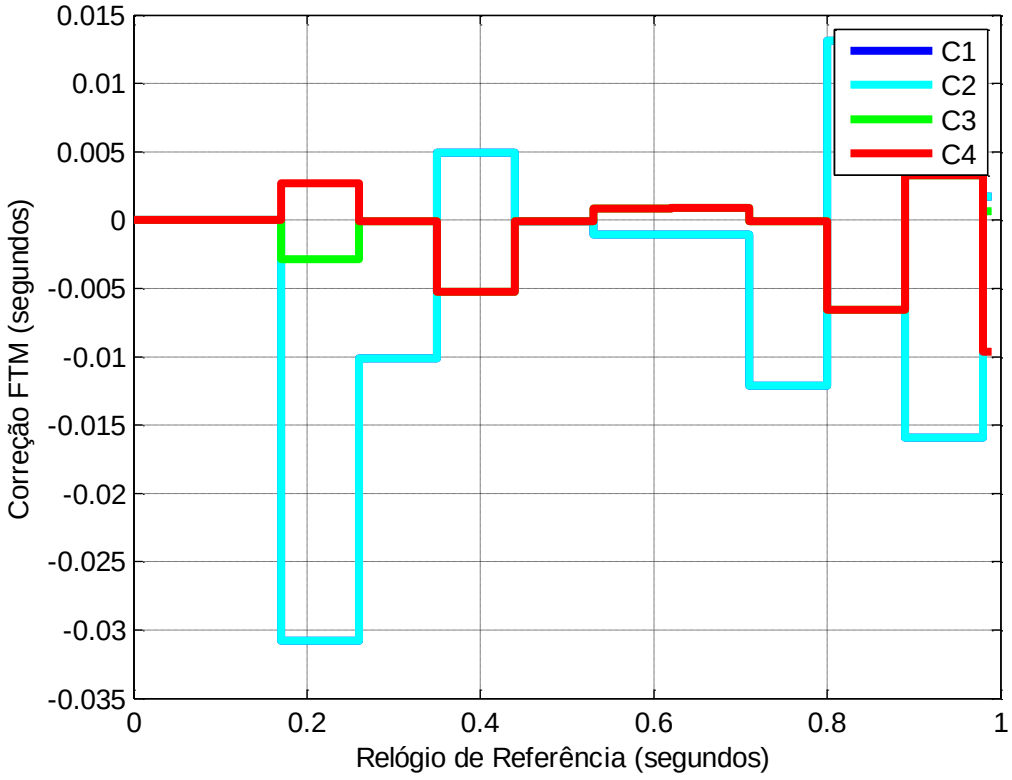


Figura 8.128 – Caso 24 - Correção NOMO - Algoritmo SR.

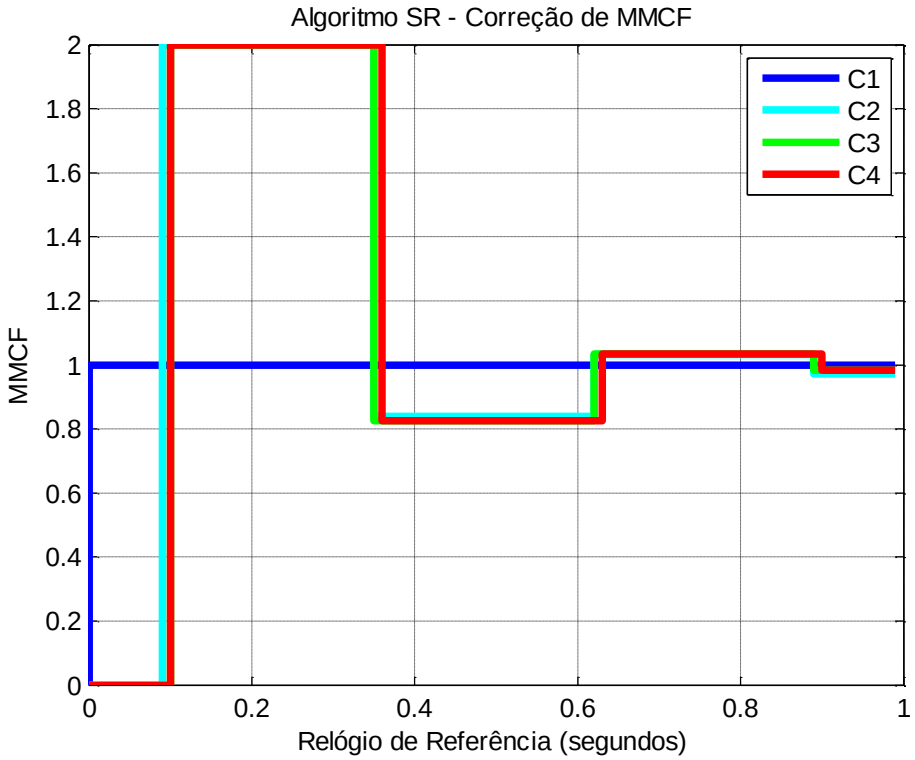


Figura 8.129 – Caso 24 - Correção NOMD - Algoritmo SR.

A Figura 8.127 mostra o valor de correção do relógio calculado pelo controlador *deadbeat* e a Figura 8.128 a correção calculada pelo modo NOMO do algoritmo SR. A Figura 8.129 mostra os valores corrigidos ao longo do tempo de MMCF, pelo modo NOMD. O tempo de transição do modo STM para NOMO é de 10 macroticks vezes o maior valor de MMCF, o de transição entre NOMO e NOMD é 3 vezes o valor do período de re-sincronização, e a re-sincronização ocorre a cada 10 microticks.

O algoritmo se mostrou eficiente, sendo que a principal vantagem é da sincronização sincronizar o viés instantâneo e o MMCF, com isto, melhorando a precisão e a exatidão. A desvantagem é que o algoritmo precisa de um gerenciamento de transições. No entanto, o tempo da transição é conhecido, devido ao *deadbeat*, o que minimiza esta desvantagem.

8.27. Caso 25 - Algoritmo PAReS Degrau

Este caso possui valores de viés inicial igual a: C1: 0; C2: 0.1 seg. ; C3: 0.01 seg. ; C4: -0.1 seg.; e deriva igual a: C1: 0; C2: 1%; C3: 0.01%; C4: -0.01%.

Neste caso, aplica-se um algoritmo com os três modos de correção: 1) STM; o 2) NOMO; e o 3) NOMD, descritos no capítulo 6. Com isto, visa-se inicialmente corrigir o viés inicial com o controlador *deadbeat*, depois utiliza-se a técnica FTM em conjunto com uma função *cross-fading* degrau; e depois sincroniza-se os MMCFs utilizando o relógio C1 como referência, definido como o algoritmo PAReS com uma função degrau. A Figura 8.130 e Figura 8.131 mostram, respectivamente, os valores de macrotick e diferença entre os macroticks.

A Figura 8.132 mostra o valor de correção do relógio calculado pelo controlador *deadbeat*.

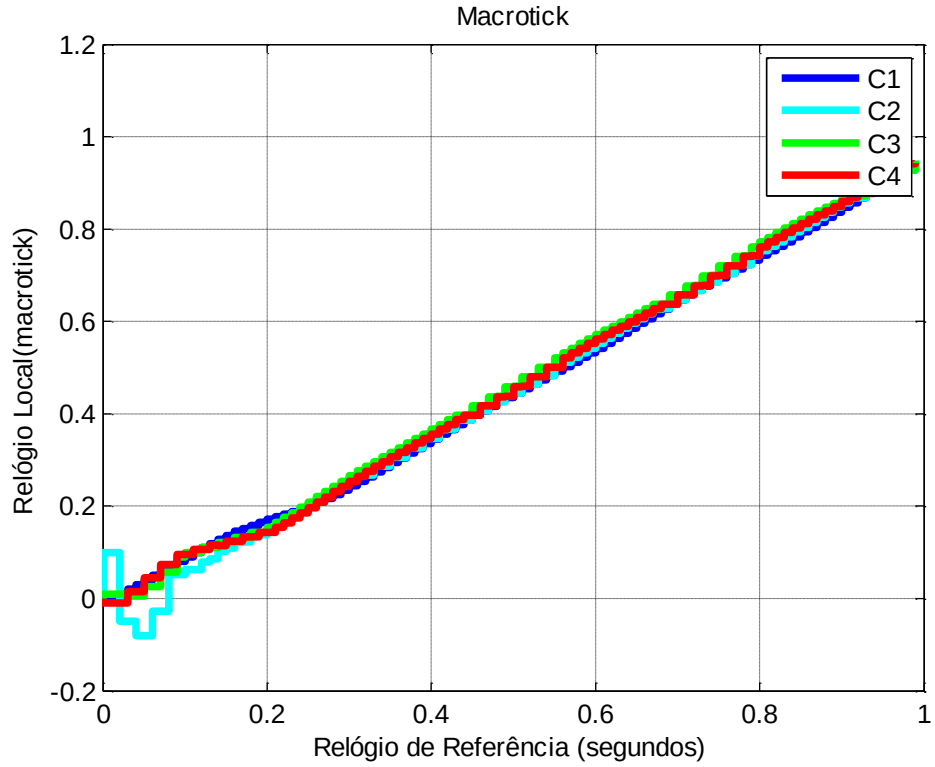


Figura 8.130 – Caso 25 - Macrotick.

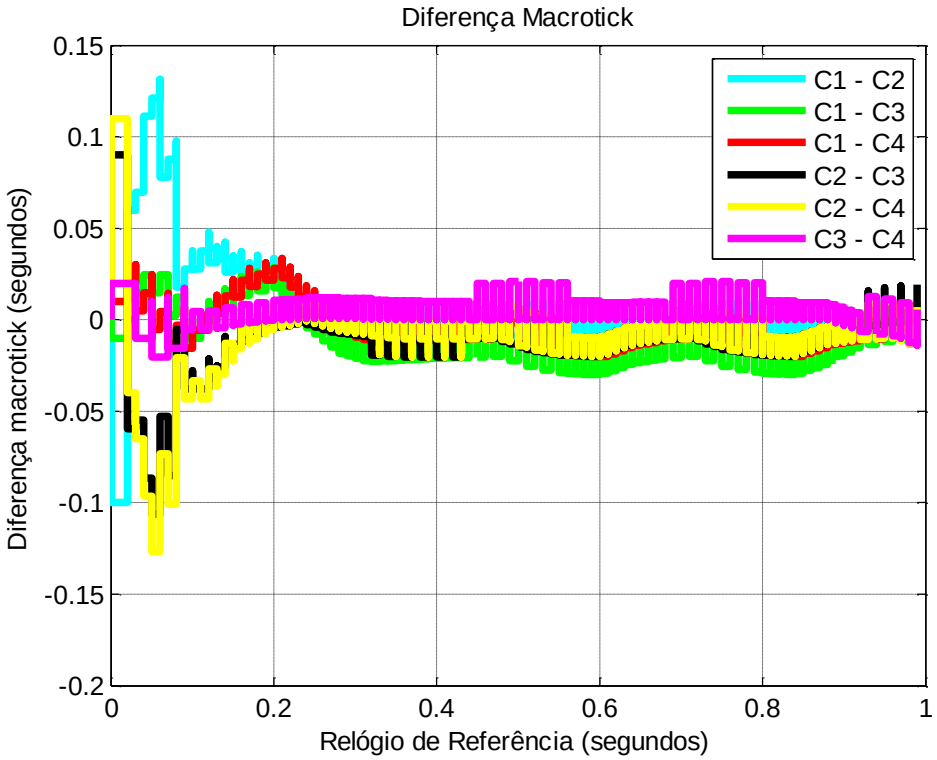


Figura 8.131 – Caso 25 - Diferença entre Macroticks.

A Figura 8.133 mostra a função *cross-fading* degrau. A Figura 8.134 mostra a correção calculada pela função de convergência do algoritmo FTM vezes a função *cross-fading* degrau. E a Figura 8.135 os valores de corrigidos de MMCFs. O tempo de transição entre os modos STM e NOMO de correção é de 10 macroticks vezes o maior MMCF, o período de re-sincronização é de 10 microticks, ou seja, após 10 microticks a correção é recalculada pela função de convergência do algoritmo FTM. No entanto, com o uso da função degrau, a cada instante (assim como no *deadbeat*) é aplicada uma correção multiplicada pelo valor da função *cross-fading*. Assim, a cada instante existe uma correção do algoritmo. E a transição entre os modos NOMO e NOMD ocorre a cada 3 re-sincronizações do NOMO. Corrigindo o MMCF.

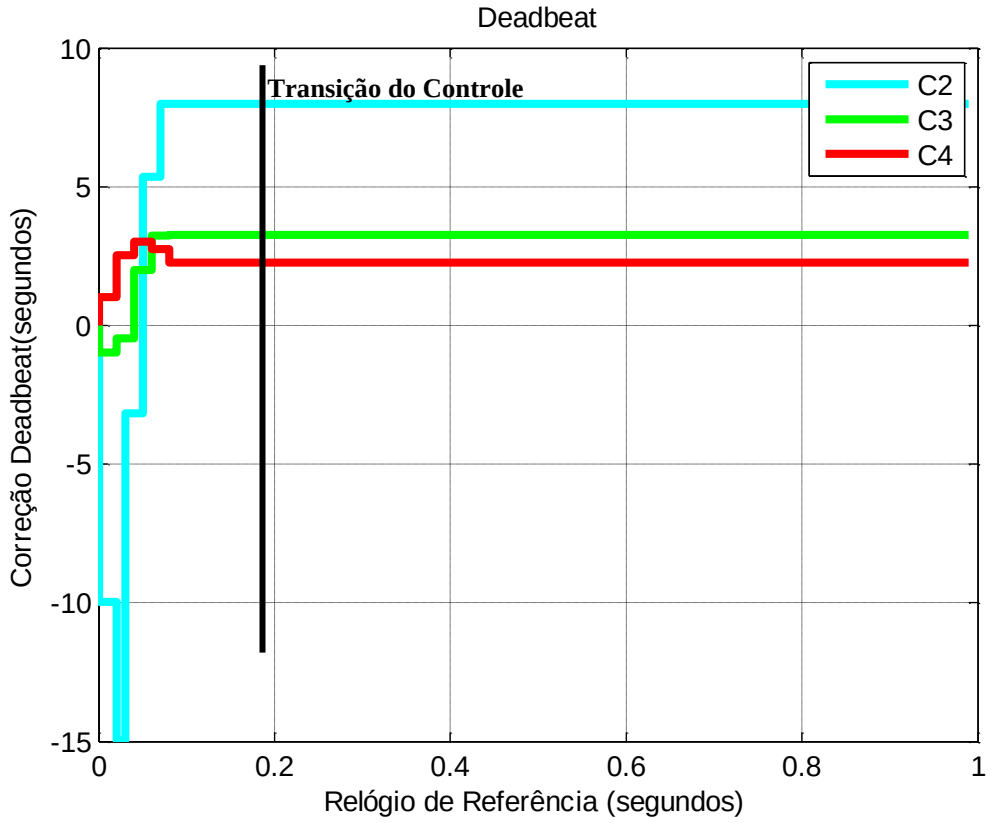


Figura 8.132 – Caso 25 - Correção STM - Algoritmo PAReS Degrau.

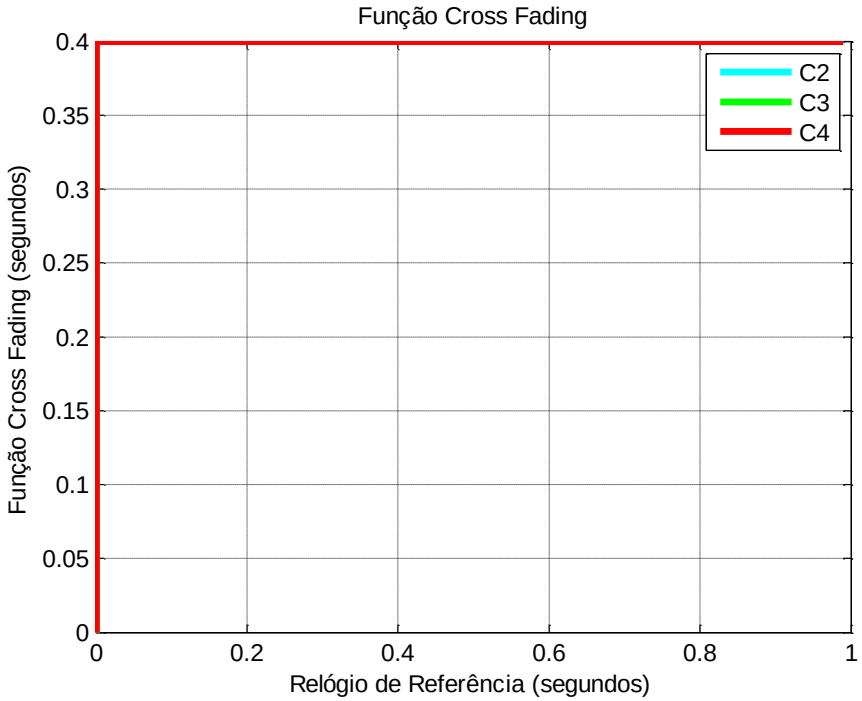


Figura 8.133 – Caso 25 - Função Cross-Fading Degrau.

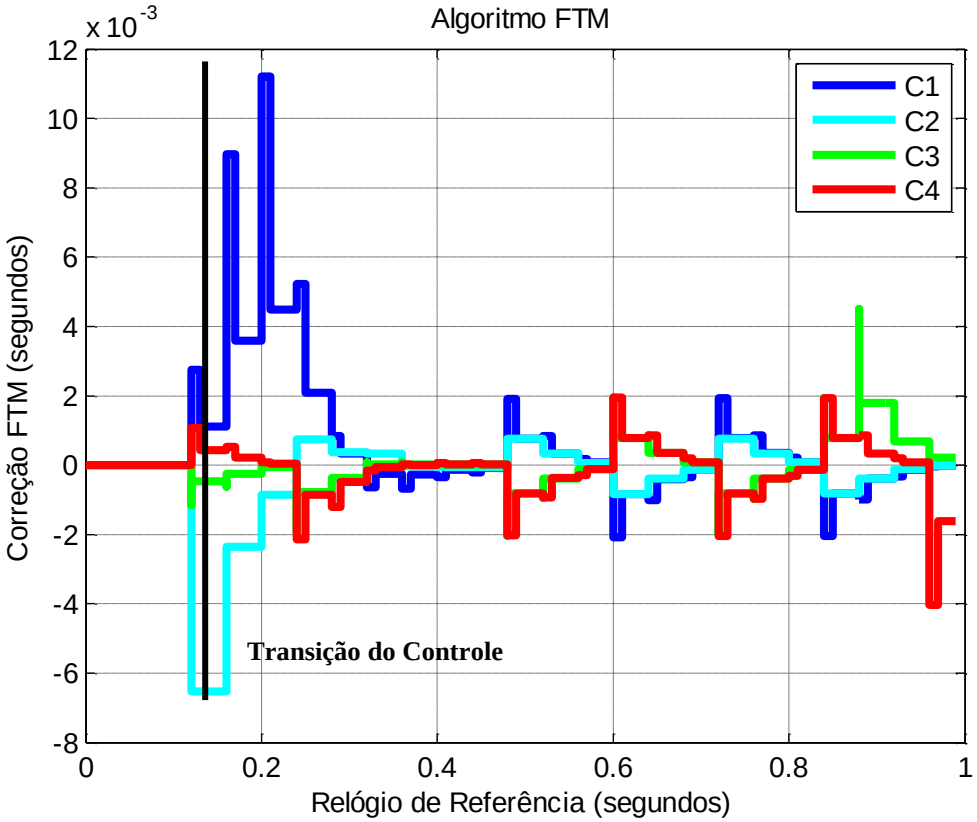


Figura 8.134 – Caso 15 - Correção NOMO - Algoritmo PAReS Degrau.

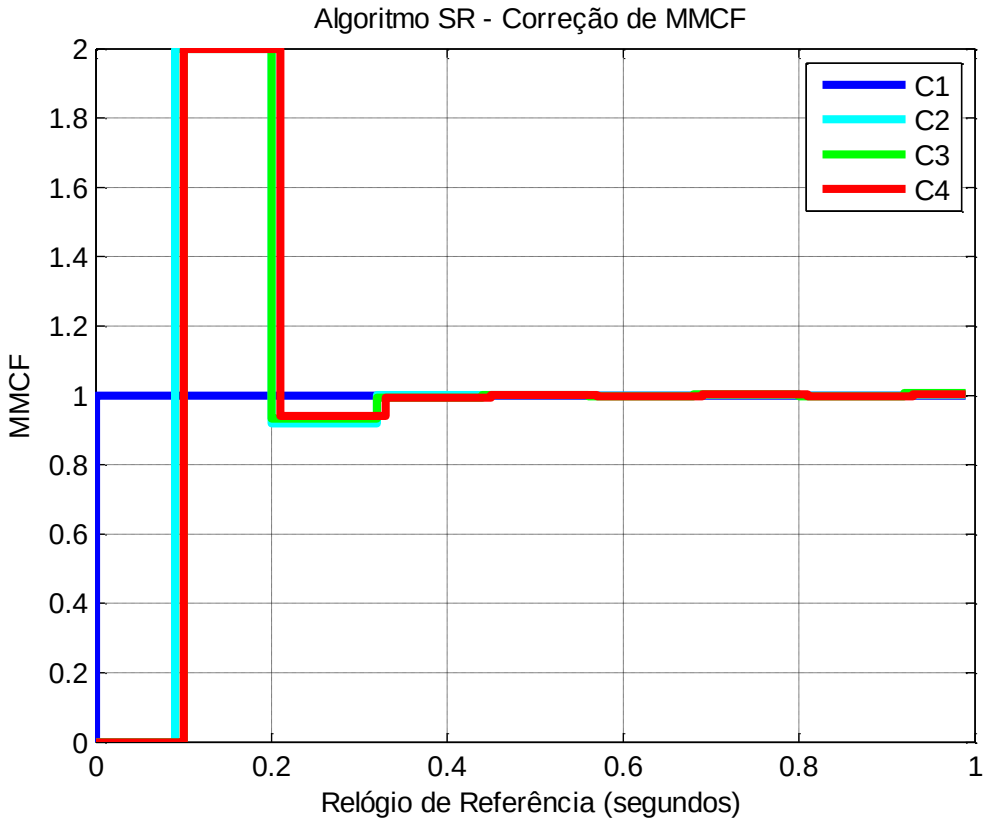


Figura 8.135 – Caso 25 - Correção NOMD - Algoritmo PAReS Degrau.

Este caso mostra uma sincronização de relógios de métrica mista com o algoritmo PReS degrau. O algoritmo se mostrou eficiente, sendo que a principal vantagem é da sincronização modular a descontinuidade de tempo via uma função degrau, evitando excursões de tempo muito grande. A precisão foi prejudicada em relação ao caso anterior. No entanto, ajustando e otimizando os parâmetros é possível conseguir resultados melhores. A desvantagem é que o algoritmo precisa de um gerenciamento de transições e necessita atualizar a cada instante o valor de tempo. No entanto, o tempo da transição é conhecido, devido ao *deadbeat*, o que minimiza esta desvantagem.

8.28. Caso 26 - Algoritmo PAReS Rampa

Este caso possui valores de viés inicial igual a: C1: 0; C2: 0.1 seg. ; C3: 0.01 seg. ; C4: -0.1 seg.; e deriva igual a: C1: 0; C2: 1%; C3: 0.01%; C4: -0.01%.

Neste caso, aplica-se um algoritmo com os três modos de correção: 1) STM; o 2) NOMO; e o 3) NOMD, descritos no capítulo 6. Com isto, visa-se inicialmente corrigir o viés inicial com o controlador *deadbeat*, depois utiliza-se a técnica FTM em conjunto com uma função *cross-fading* rampa; e depois sincroniza-se os MMCFs utilizando o relógio C1 como referência, definido como o algoritmo PAReS com uma função rampa. A Figura 8.136 e Figura 8.137 mostram, respectivamente, os valores de macrotick e diferença entre os macroticks. A Figura 8.138 mostra o valor de correção do relógio calculado pelo controlador *deadbeat*.

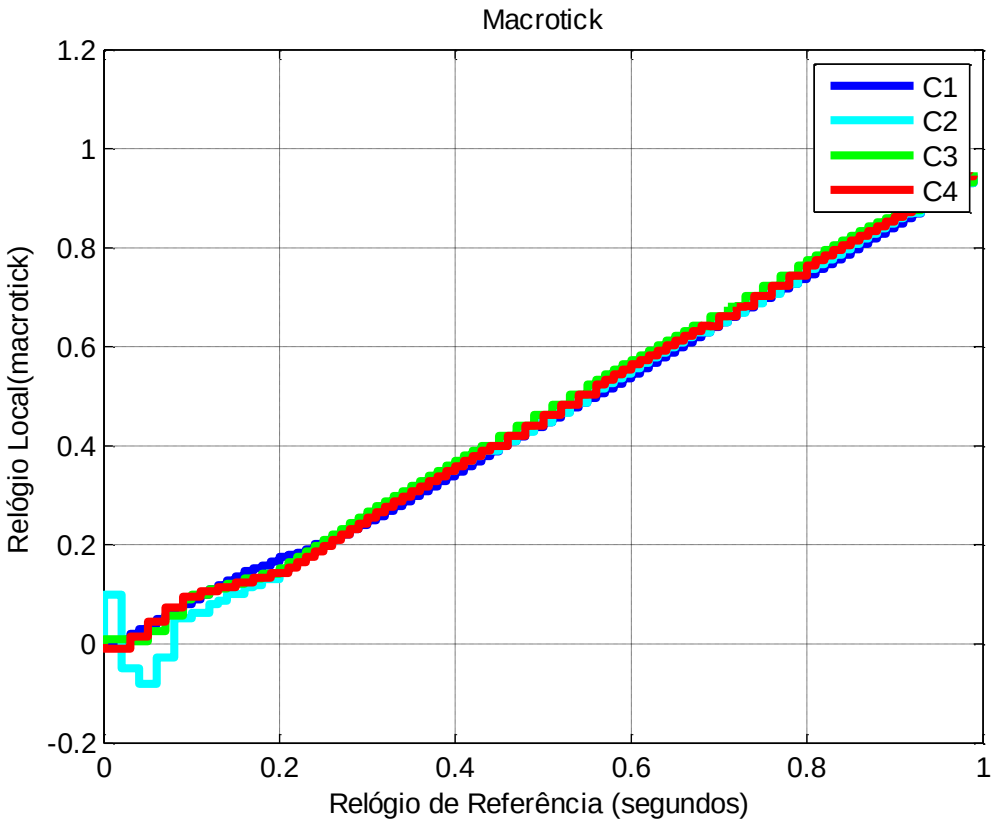


Figura 8.136 – Caso 26 - Macrotick.

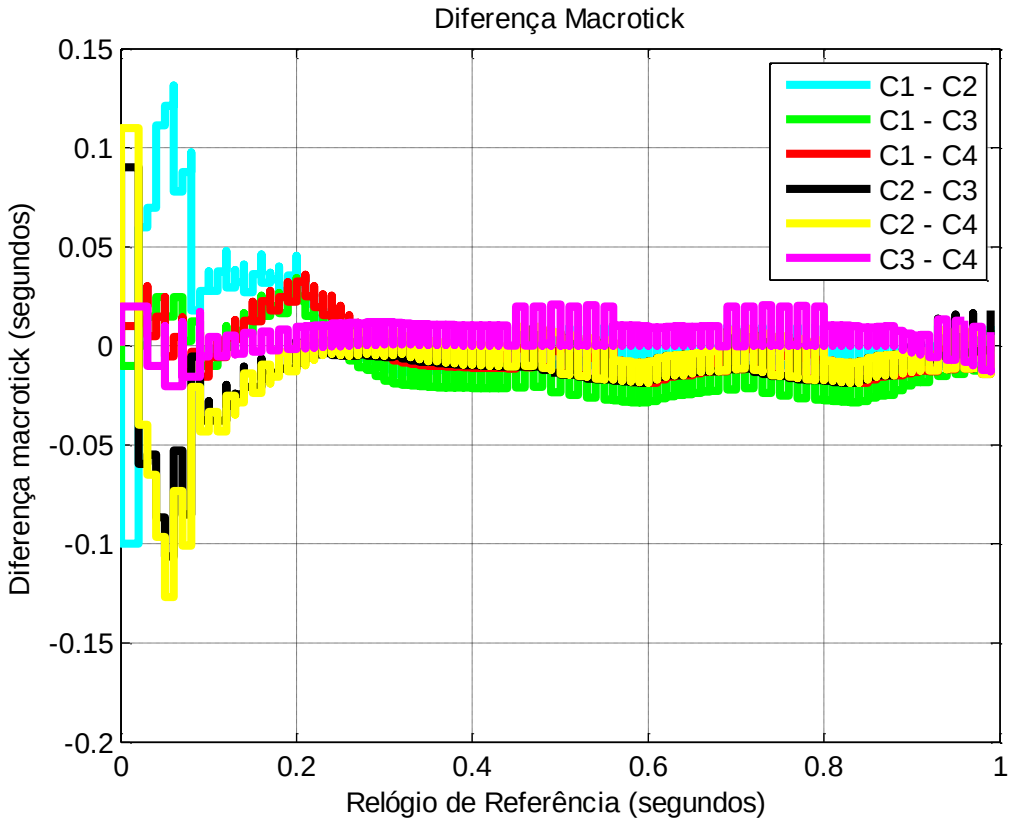


Figura 8.137 – Caso 25 - Diferença entre Macroticks.

A Figura 8.139 mostra a função *cross-fading* rampa. A Figura 8.140 mostra a correção calculada pela função de convergência do algoritmo FTM vezes a função *cross-fading* rampa. E a Figura 8.141 os valores de corrigidos de MMCFs. O tempo de transição entre os modos STM e NOMO de correção é de 10 macroticks vezes o maior MMCF, o período de re-sincronização é de 10 microticks, ou seja, após 10 microticks a correção é recalculada pela função de convergência do algoritmo FTM. No entanto, com o uso da função degrau, a cada instante (assim como no *deadbeat*) é aplicada uma correção multiplicada pelo valor da função *cross-fading*. Assim, a cada instante existe uma correção do algoritmo. E a transição entre os modos NOMO e NOMD ocorre a cada 3 re-sincronizações do NOMO. Corrigindo o MMCF.

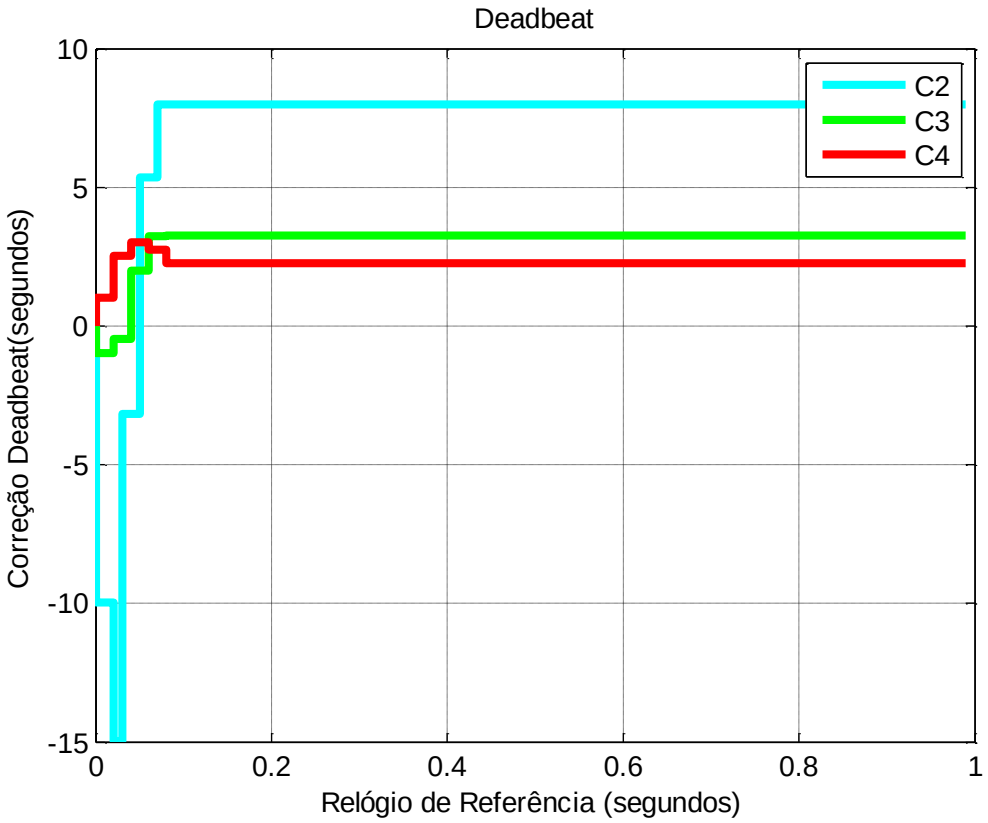


Figura 8.138 – Caso 26 - Correção STM - Algoritmo PAReS Rampa.

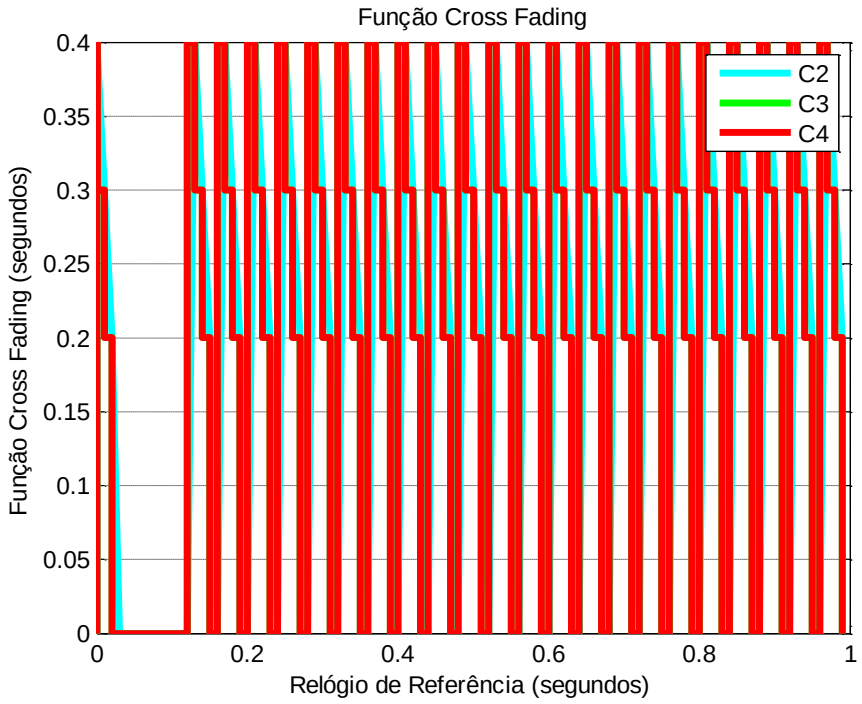


Figura 8.139 – Caso 26 - Função Cross-Fading Rampa.

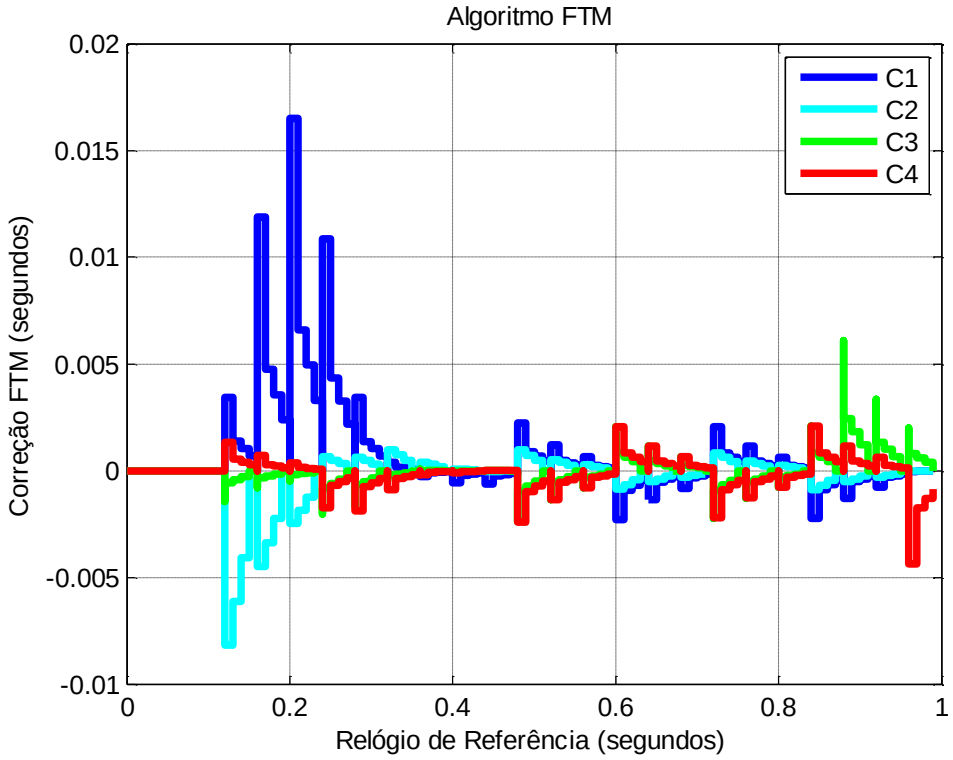


Figura 8.140 – Caso 26 - Correção NOMO - Algoritmo PAReS Rampa.

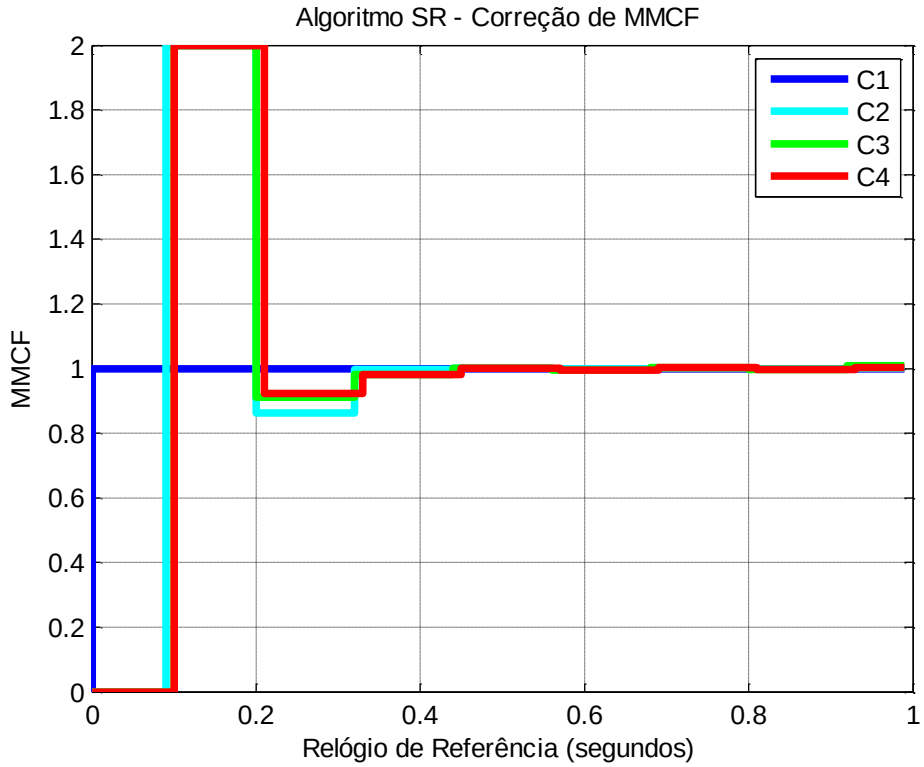


Figura 8.141 – Caso 26 - Correção NOMD - Algoritmo PAReS Rampa.

Este caso mostra uma sincronização de relógios de métrica mista com o algoritmo PAReS rampa. O algoritmo se mostrou eficiente, sendo que a principal vantagem é da sincronização modular a descontinuidade de tempo via uma função degrau, evitando excursões de tempo muito grande, e corrigir os valores de MMCF. A precisão e a exatidão foram melhoradas em relação ao caso anterior. No entanto, ajustando e otimizando os parâmetros é possível conseguir resultados melhores. A desvantagem é que o algoritmo precisa de um gerenciamento de transições e necessita atualizar a cada instante o valor de tempo. No entanto, o tempo da transição é conhecido, devido ao *deadbeat*, o que minimiza esta desvantagem. No caso da reta, é necessário uma segunda transição para evitar que o valor da função *cross-fading* rampa fique negativa e inverta a fase do sistema tornando-o instável.

8.29. Caso 27 - Algoritmo PAReS Exponencial

Este caso possui valores de viés inicial igual a: C1: 0; C2: 0.1 seg. ; C3: 0.01 seg. ; C4: -0.1 seg.; e deriva igual a: C1: 0; C2: 1%; C3: 0.01%; C4: -0.01%.

Neste caso, aplica-se um algoritmo com os três modos de correção: 1) STM; o 2) NOMO; e o 3) NOMD, descritos no capítulo 6. Com isto, visa-se inicialmente corrigir o viés inicial com o controlador *deadbeat*, depois utiliza-se a técnica FTM em conjunto com uma função *cross-fading* exponencial; e depois sincroniza-se os MMCFs utilizando o relógio C1 como referência, definido como o algoritmo PAReS com uma função exponencial. A Figura 8.142 e Figura 8.143 mostram, respectivamente, os valores de macrotick e diferença entre os macroticks. A Figura 8.144 mostra o valor de correção do relógio calculado pelo controlador *deadbeat*.

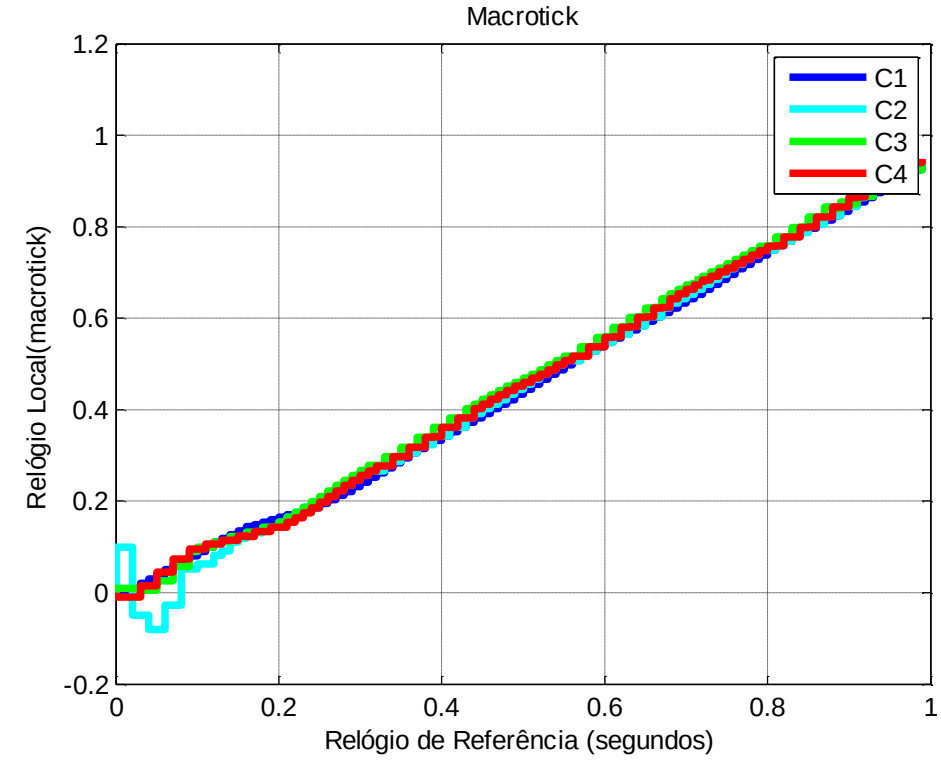


Figura 8.142 – Caso 27 - Macrotick.

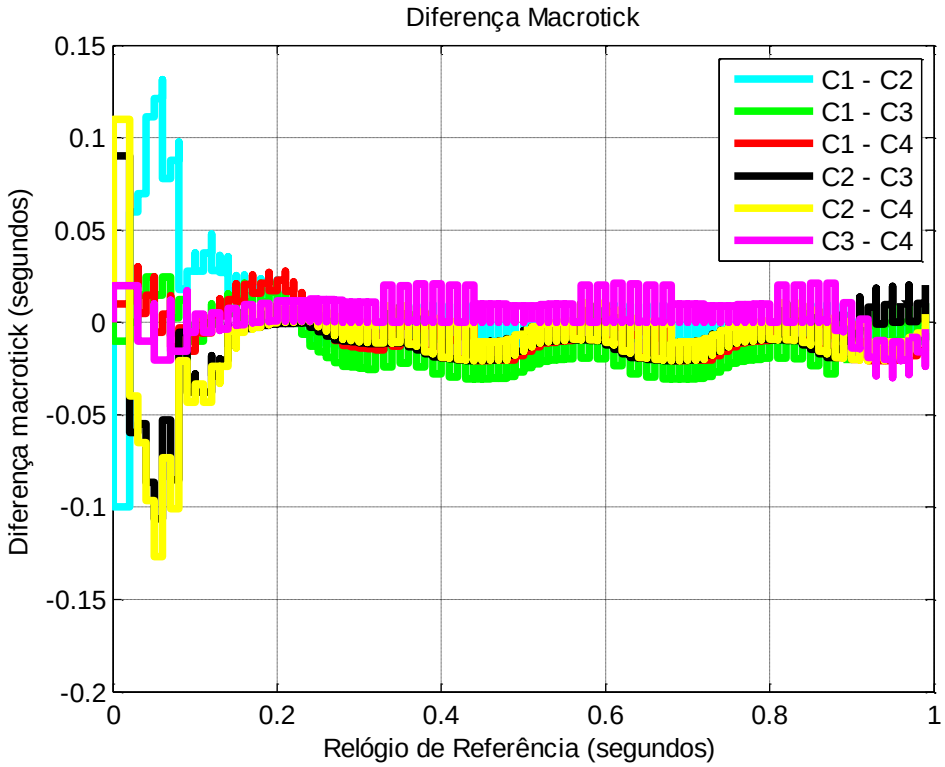


Figura 8.143 – Caso 27 - Diferença entre Macroticks.

A Figura 8.145 mostra a função *cross-fading* exponencial. A Figura 8.146 mostra a correção calculada pela função de convergência do algoritmo FTM vezes a função *cross-fading* exponencial. E a Figura 8.147 os valores de corrigidos de MMCFs. O tempo de transição entre os modos STM e NOMO de correção é de 10 macroticks vezes o maior MMCF, o período de re-sincronização é de 10 microticks, ou seja, após 10 microticks a correção é recalculada pela função de convergência do algoritmo FTM. No entanto, com o uso da função exponencial, a cada instante (assim como no *deadbeat*) é aplicada uma correção multiplicada pelo valor da função *cross-fading*. Assim, a cada instante existe uma correção do algoritmo. E a transição entre os modos NOMO e NOMD ocorre a cada 3 re-sincronizações do NOMO. Corrigindo o MMCF.

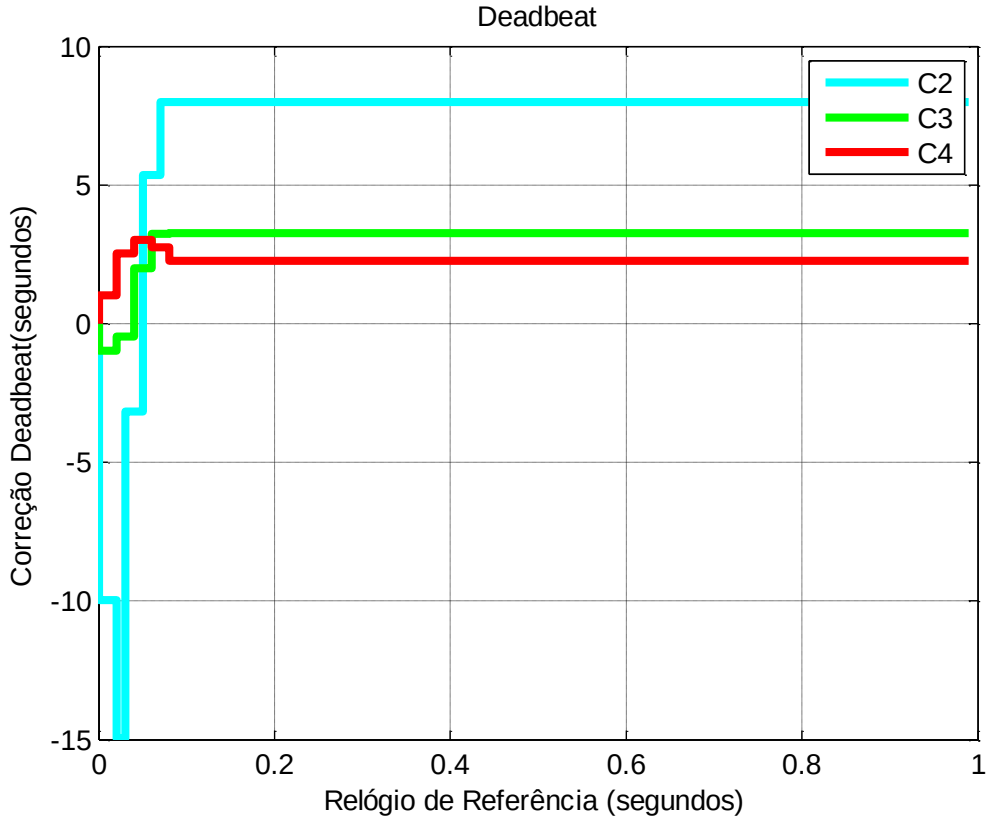


Figura 8.144 – Caso 27 - Correção STM - Algoritmo PAReS Exponencial.

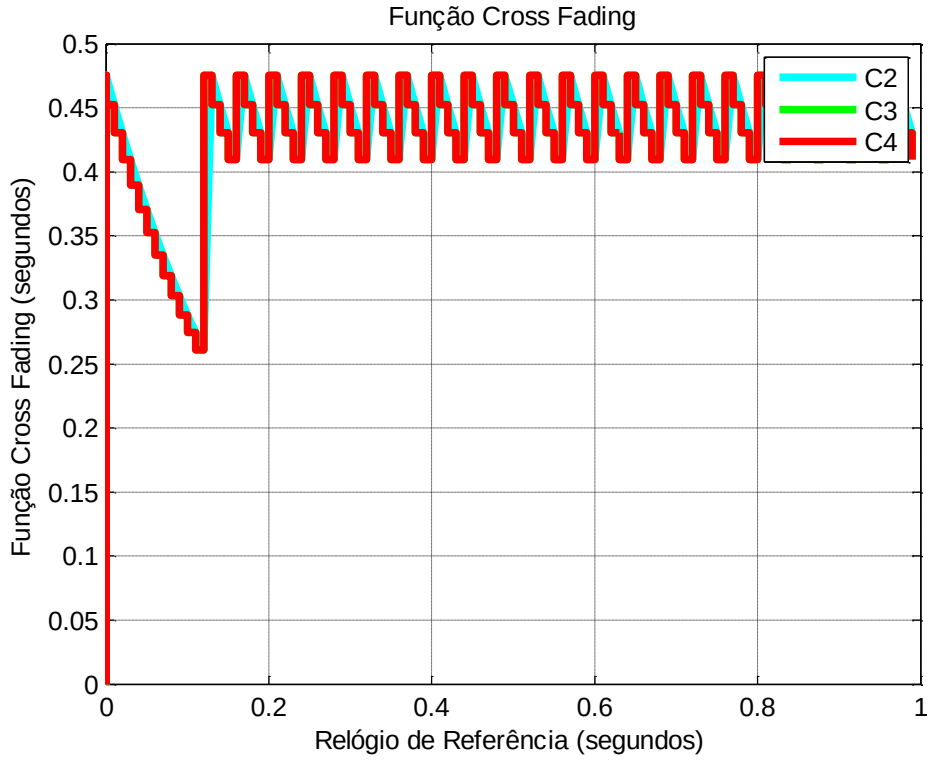


Figura 8.145 – Caso 27 - Função Cross-Fading Exponencial.

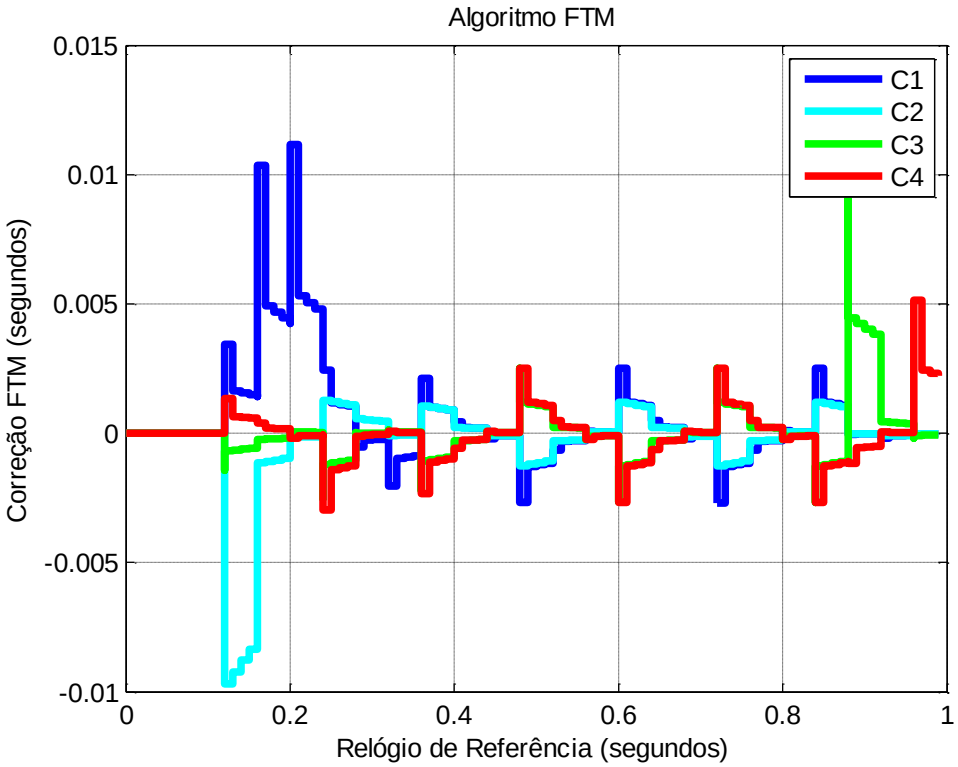


Figura 8.146 – Caso 27 - Correção NOMO - Algoritmo PAReS Exponencial.

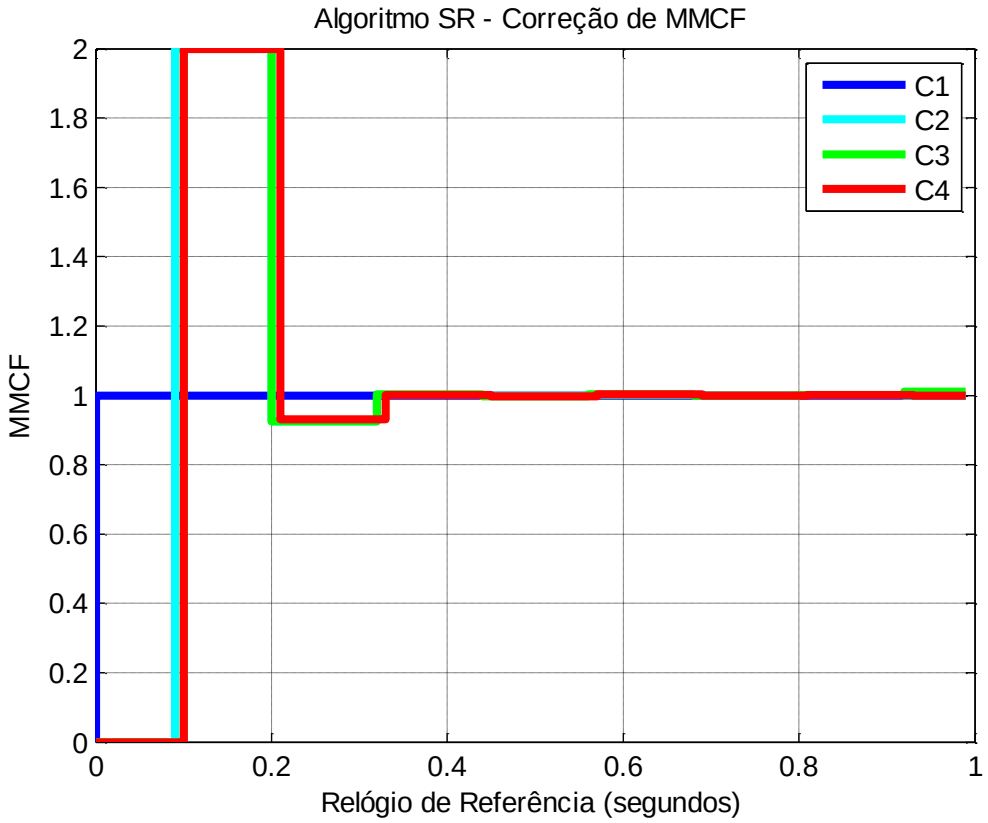


Figura 8.147 – Caso 27 - Correção NOMD - Algoritmo PAReS Exponencial.

Este caso mostra uma sincronização de relógios de métrica mista com o algoritmo PAReS exponencial. O algoritmo se mostrou eficiente, sendo que a principal vantagem é da sincronização modular a descontinuidade de tempo via uma função exponencial, evitando excursões de tempo muito grande, e corrigir os valores de MMCF. Ajustando e otimizando os parâmetros é possível conseguir resultados melhores. A desvantagem é que o algoritmo precisa de um gerenciamento de transições e necessita atualizar a cada instante o valor de tempo. No entanto, o tempo da transição é conhecido, devido ao *deadbeat*, o que minimiza esta desvantagem. No caso da exponencial, não é necessário uma segunda transição para evitar que o valor da função *cross-fading* exponencial fique negativa.

8.30. Caso 28 - SR - CSMA/CD

Este caso possui valores de viés inicial igual a: C1: 0; C2: 0.1 seg. ; C3: 0.01 seg. ; C4: -0.1 seg.; e deriva igual a: C1: 0; C2: 1%; C3: 0.01%; C4: -0.01%. Neste caso, aplica-se o algoritmo SR para corrigir o tempo sobre um sistema de controle por redes com uma rede CSMA/CD. A Figura 8.148 e Figura 8.149 mostram os valores de macrotick e diferença dos macroticks, respectivamente.

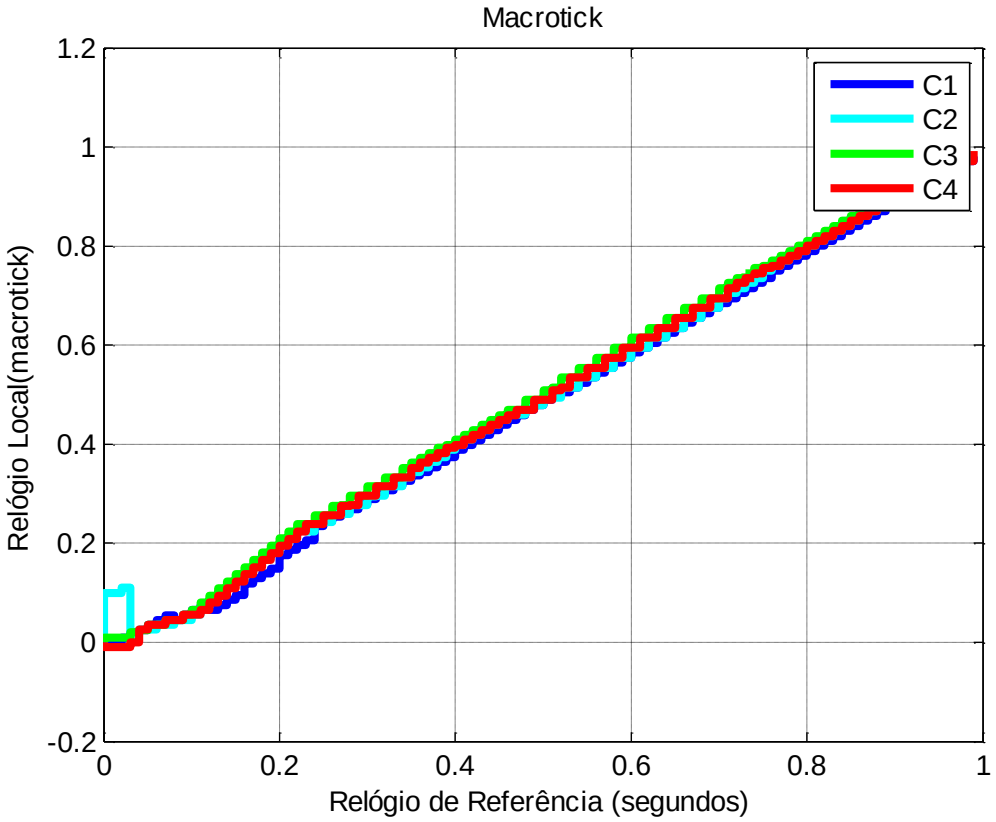


Figura 8.148 – Caso 28 - Macrotick.

A Figura 8.150 mostra o valor de correção do relógio calculado pela função de convergência do algoritmo FTM. O período de re-sincronização é de 10 microticks, ou seja, a cada 10 microticks aplica-se o processo de sincronização do algoritmo FTM. A cada 3 re-sincronizações é feito a sincronização do MMCF, como mostra a Figura 8.151. A Figura 8.152 e Figura 8.153 mostram a lei de controle PID e a resposta dinâmica do sistema de controle por rede em uma rede CSMA/CD.

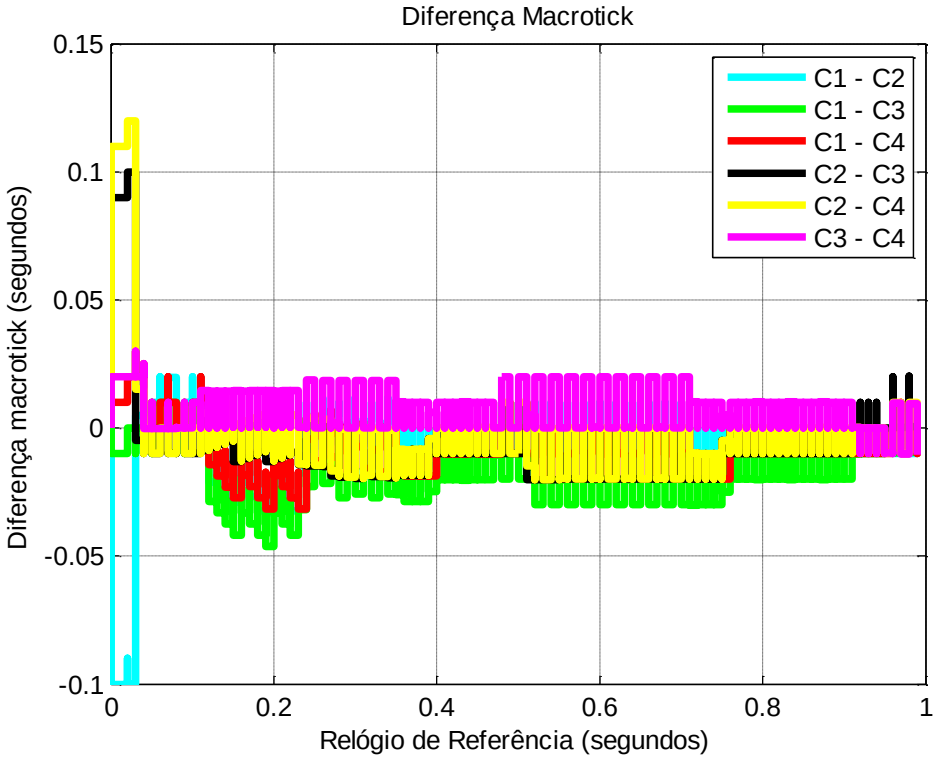


Figura 8.149 – Caso 28 - Diferença entre Macroticks.

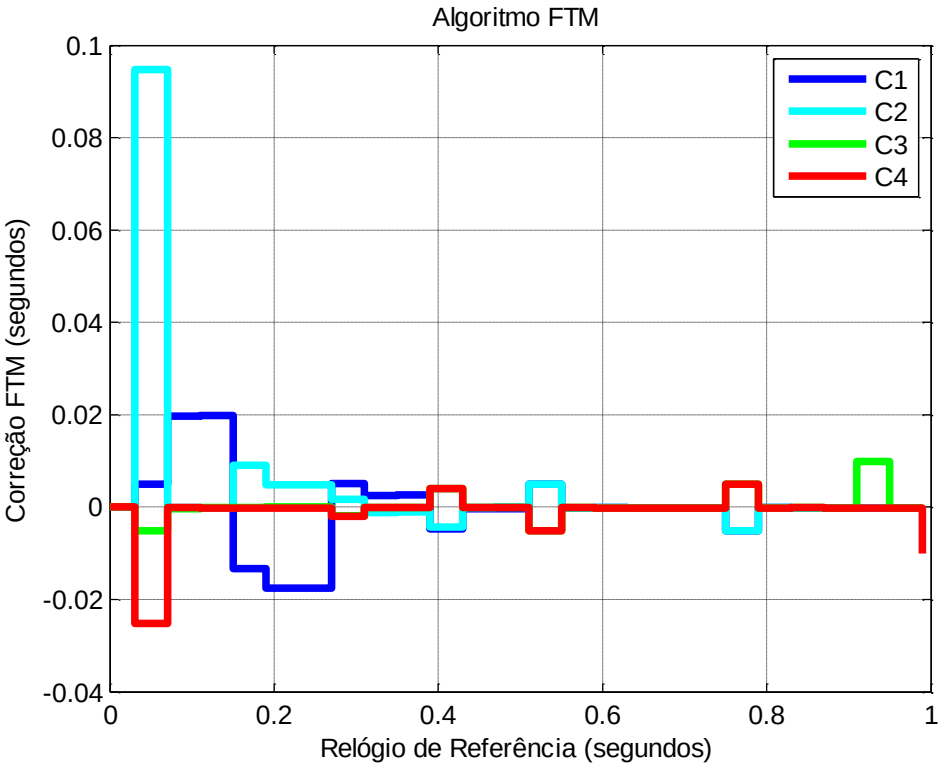


Figura 8.150 – Caso 28 - Correção viés Algoritmo SR.

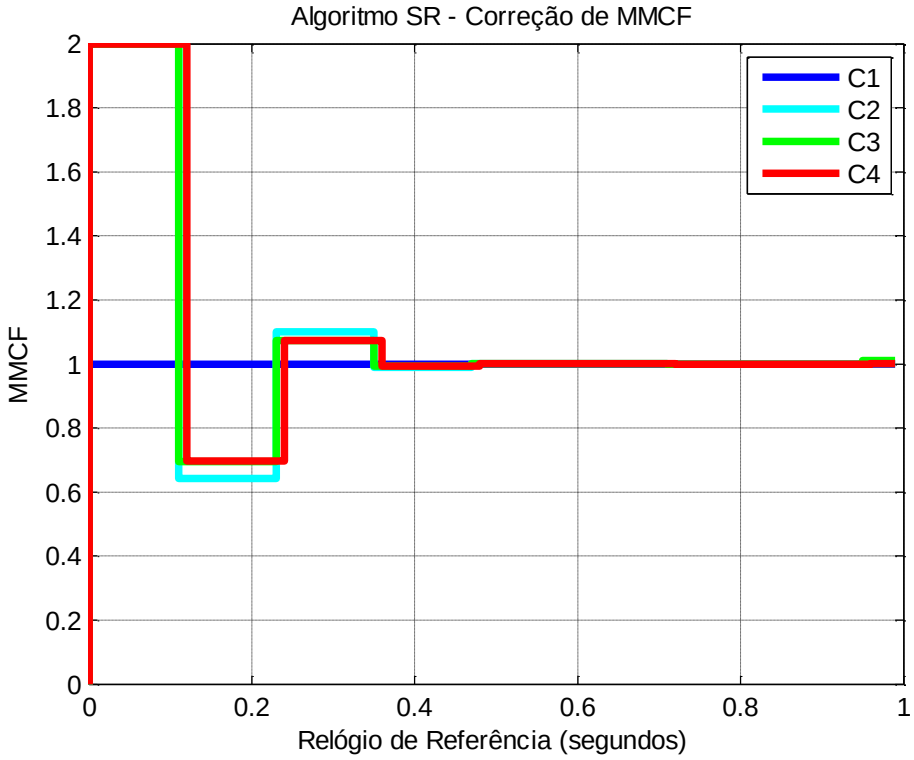


Figura 8.151 – Caso 28 - Correção MMCF Algoritmo SR.

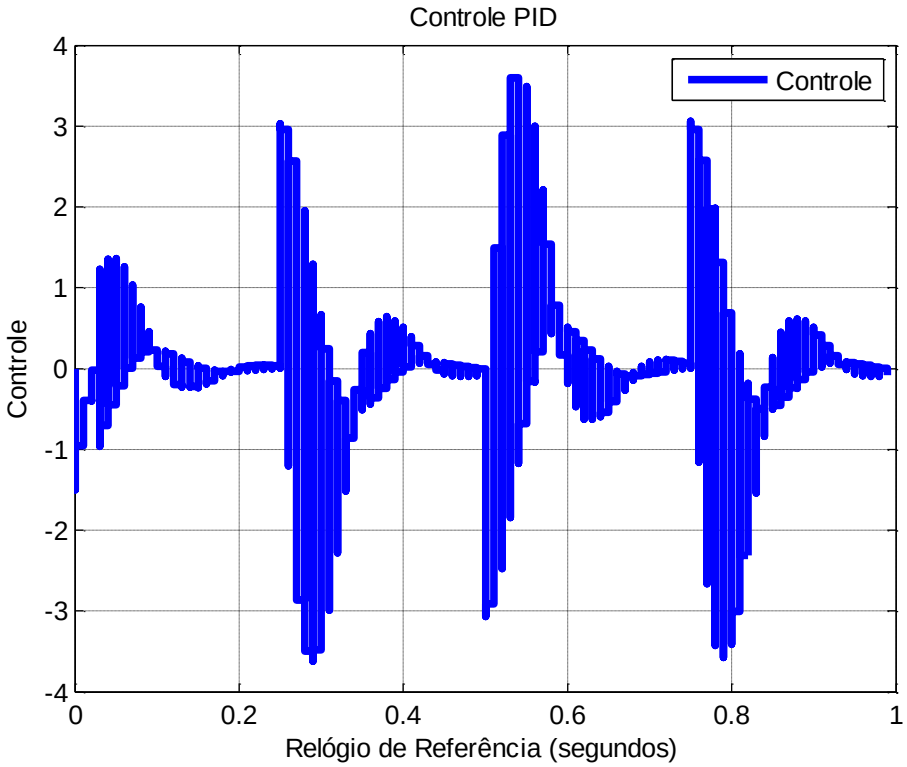


Figura 8.152 – Caso 28 - Controle PID.

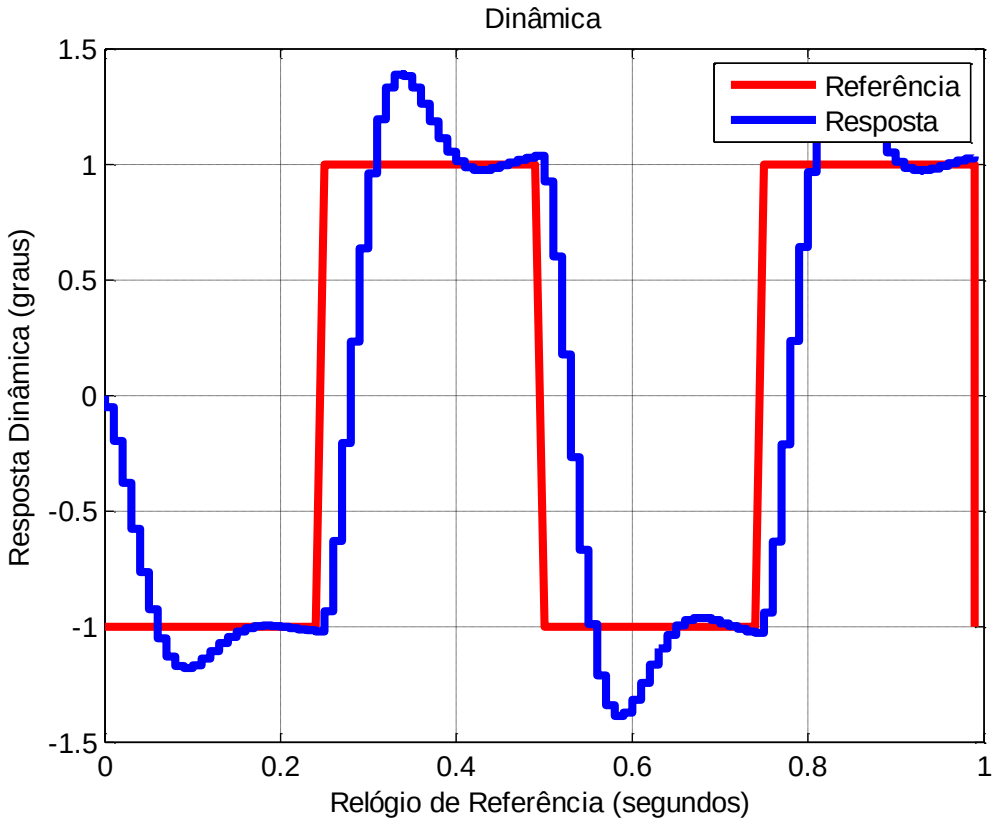


Figura 8.153 – Caso 28 - Resposta Dinâmica.

Este caso mostra uma sincronização de relógios com o algoritmo SR sobre um sistema de controle por redes em uma rede CSMA/CD. O algoritmo de sincronização se mostrou eficiente para estabelecer a sincronização. No entanto, devido à grande excursão de tempo (descontinuidade de tempo) o algoritmo ao estabelecer a sincronização prejudica indiretamente o sistema de controle, causando sobressinal na resposta dinâmica. Isto ocorre, pois o algoritmo SR não foi desenvolvido para levar em conta o sistema de controle e a de-sincronização inicial.

8.31. Caso 29 - DeadBeat - Algoritmo SR - CSMA/CD

Este caso possui valores de viés inicial igual a: C1: 0; C2: 0.1 seg. ; C3: 0.01 seg. ; C4: -0.1 seg.; e deriva igual a: C1: 0; C2: 1%; C3: 0.01%; C4: -0.01%. Este caso aplica-se um algoritmo com os três modos de correção: 1) STM; o 2) NOMO; e 3) NOMD, descritos no capítulo 6. Com isto, visa-se inicialmente

corrigir o viés inicial com o *deadbeat* e depois com o algoritmo SR corrigir o viés e o MMCF, utilizando C1 como referência, sobre um sistema de controle por redes sobre uma rede CSMA/CD. A Figura 8.154 e Figura 8.155 mostram, respectivamente, os valores de macrotick e diferença dos macroticks.

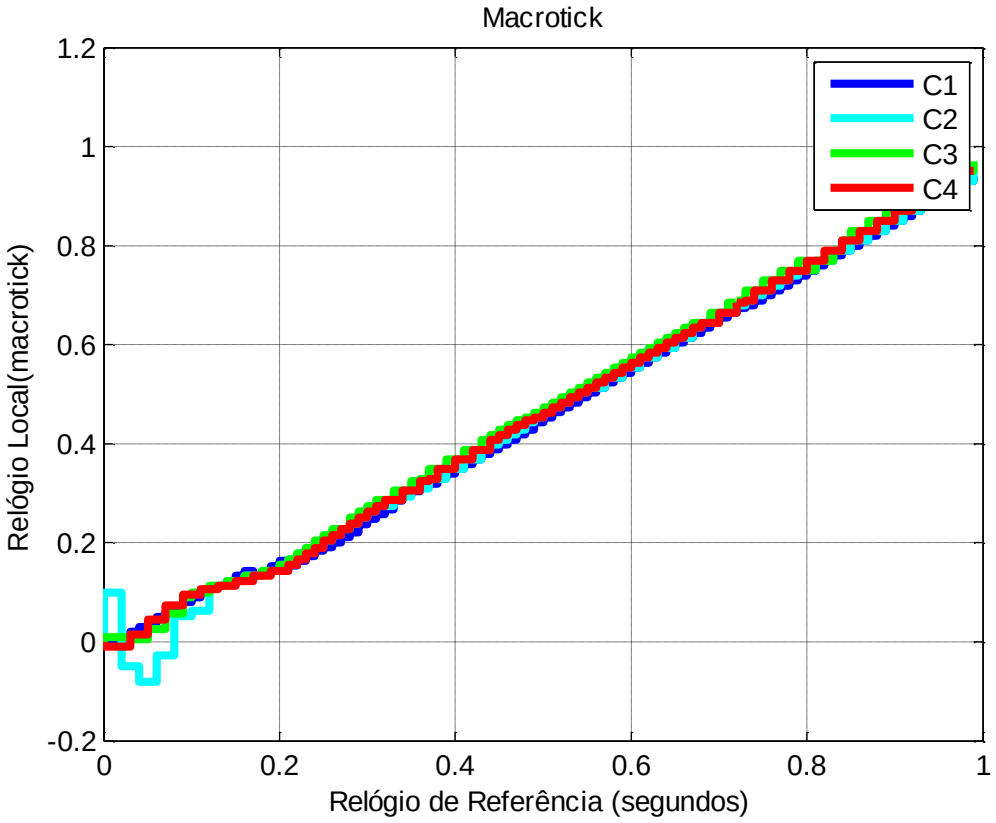


Figura 8.154 – Caso 29 - Macrotick.

A Figura 8.156 mostra o valor de correção do relógio calculado pelo controlador *deadbeat* no modo STM, a Figura 8.157 a correção de viés calculada pela função de convergência do algoritmo SR no modo NOMO; e a Figura 8.158 a correção calculada pelo modo NOMD, de correção do MMCF.

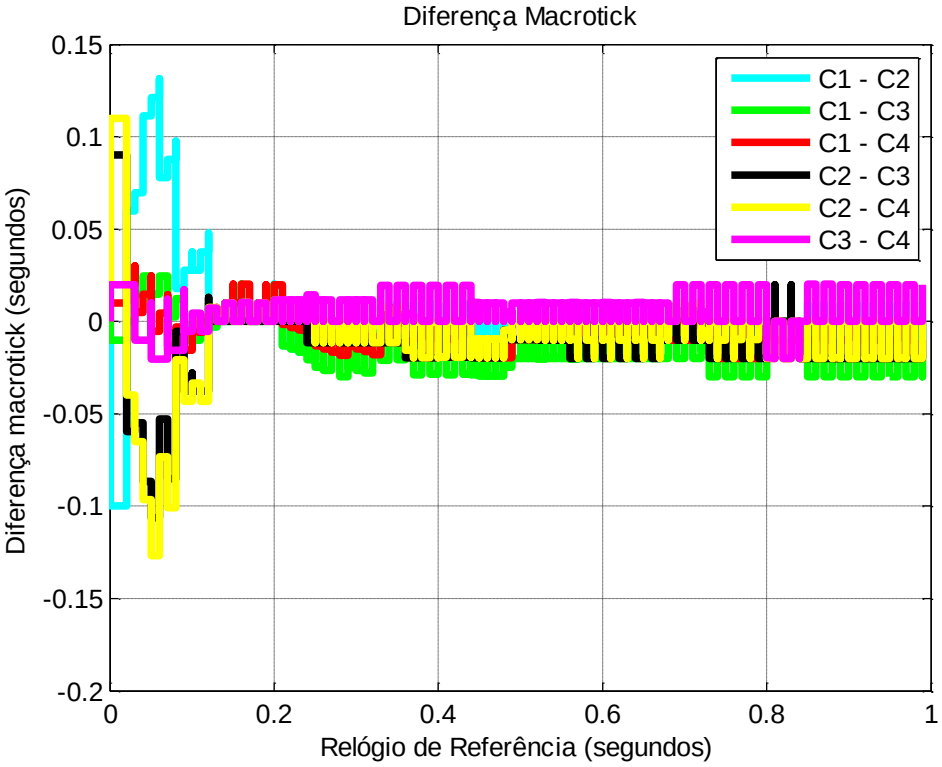


Figura 8.155 – Caso 29 - Diferença entre Macroticks.

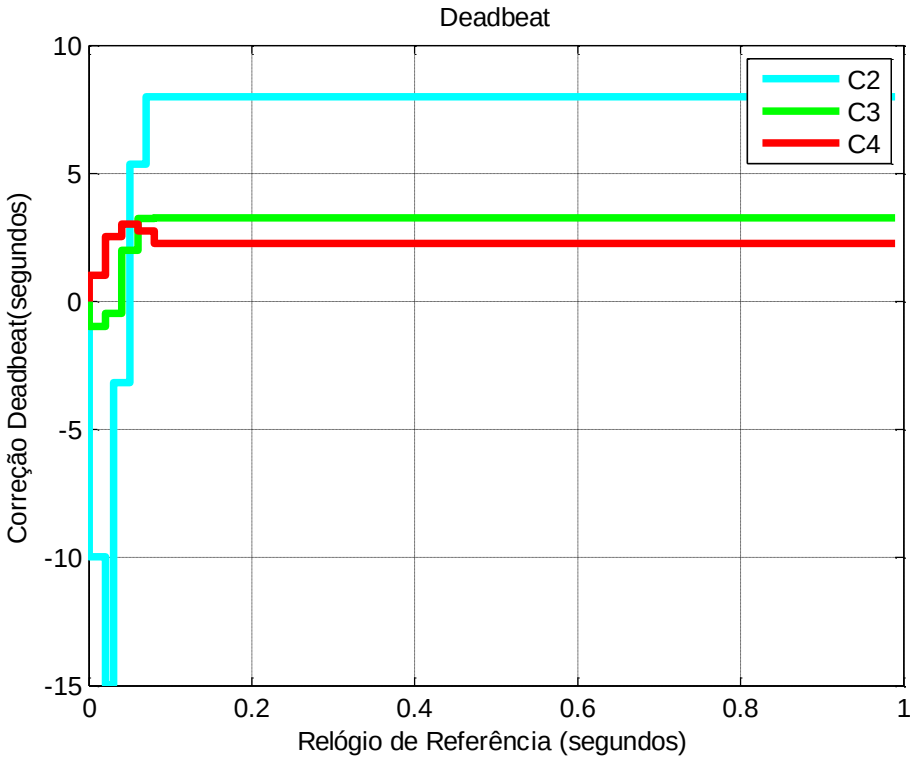


Figura 8.156 – Caso 29 - Modo STM - Controle *Deadbeat*.

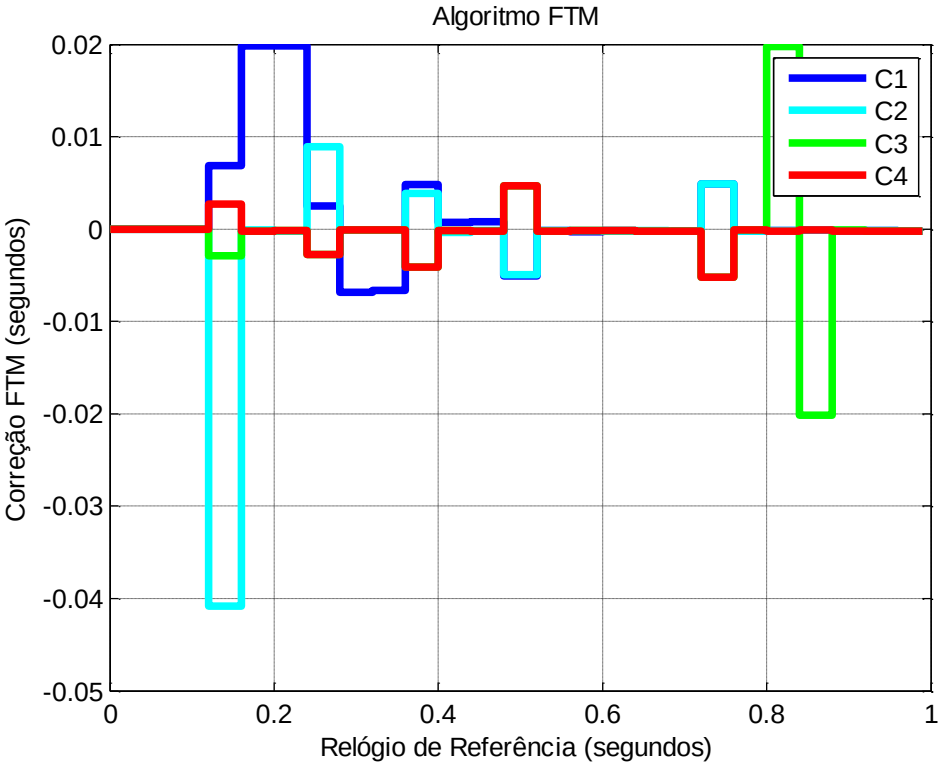


Figura 8.157 – Caso 29 - Correção viés modo NOMO - Algoritmo SR.

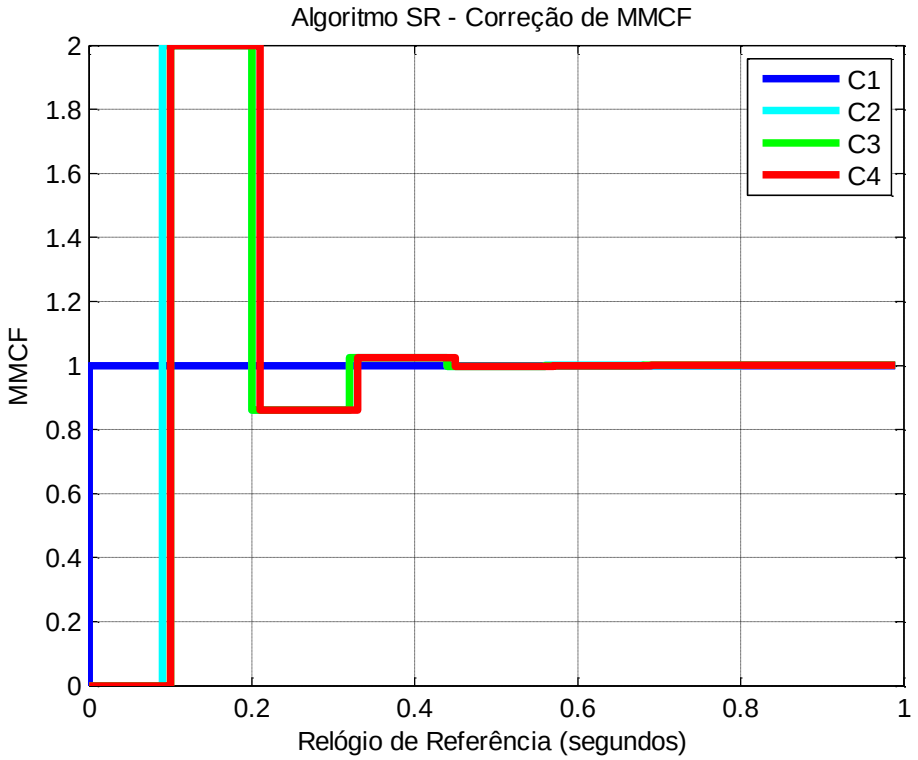


Figura 8.158 – Caso 29 - Correção MMCF modo NOMD - Algoritmo SR.

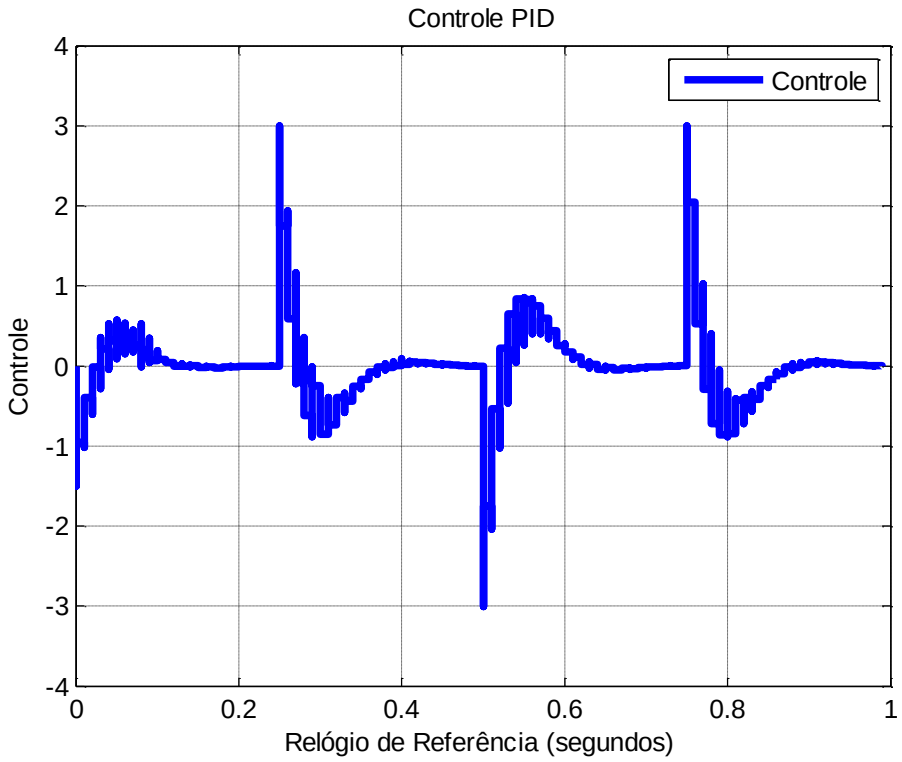


Figura 8.159 – Caso 29 - Controle PID.

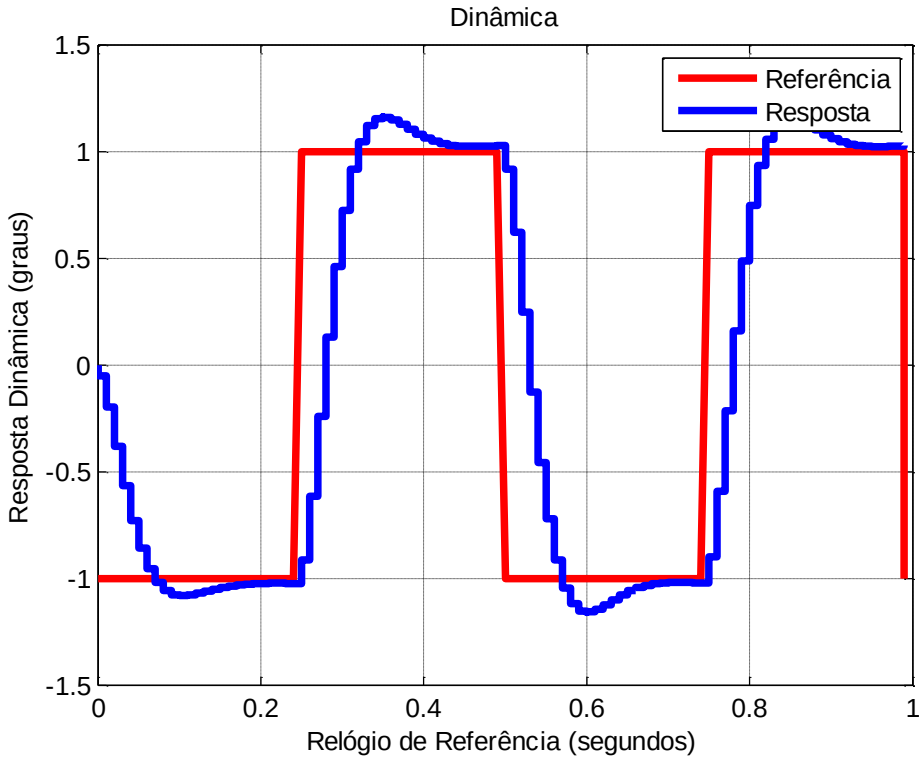


Figura 8.160 – Caso 29 - Resposta Dinâmica.

O tempo de transição dos modos de correção é de 10 macroticks e o período de re-sincronização é de 10 microticks, ou seja, após 10 macroticks, a correção passa a ser feita pelo PTP que a cada 10 microticks aplica-se o processo de sincronização do algoritmo.

Este caso mostra uma sincronização de relógios com o controlador *deadbeat* no modo STM e o algoritmo FTM no modo NOMO, sobre um sistema de controle por redes em uma rede CSMA/CD. No modo STM, não é realizado o controle da dinâmica, pois é um modo de inicialização de tempo. No modo NOMO o algoritmo de sincronização se mostrou eficiente para estabelecer a sincronização. Além disso, a sincronização de tempo melhorou, em relação ao caso anterior, a resposta dinâmica, apresentando um sobressinal médio (em torno de 20% a 40%). No sistema de controle. A de-sincronização inicial que antes levou o algoritmo a gerar sobre sinais altos no sistema de controle, agora foi corrigida aplicando a reconfiguração dos algoritmos de tempo e utilizando o *deadbeat*. Esta técnica se mostrou eficiente e os resultados podem ser melhorados através da otimização dos parâmetros.

8.32. Caso 30 - Algoritmo PAReS Degrau - CSMA/CD

Este caso possui valores de viés inicial igual a: C1: 0; C2: 0.1 seg. ; C3: 0.01 seg. ; C4: -0.1 seg.; e deriva igual a: C1: 0; C2: 1%; C3: 0.01%; C4: -0.01%. Neste caso, aplica-se um algoritmo com os três modos de correção: 1) STM; 2) NOMO; e 3) NOMD, descritos no capítulo 6. Com isto, visa-se inicialmente corrigir o viés inicial com o controlador *deadbeat*, depois utiliza-se a técnica FTM em conjunto com uma função *cross-fading* degrau e depois corrige-se o MMCF com o C1 de referência, definindo o algoritmo PAReS degrau, sobre um sistema de controle por redes em uma rede CSMA/CD. A Figura 8.161 e Figura 8.162 mostram, respectivamente, os valores de macrotick e diferença entre o relógio de referência e os macroticks. A Figura 8.163 mostra o valor de correção do relógio calculado pelo controlador *deadbeat* no modo STM. A Figura 8.164 mostra a função *cross-fading* degrau.

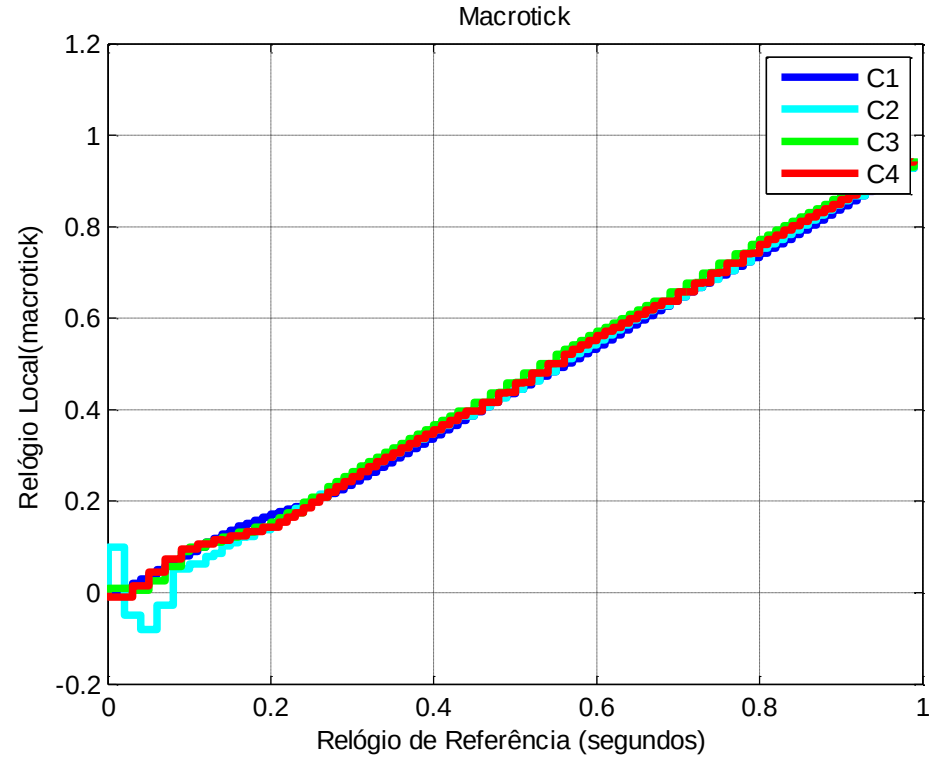


Figura 8.161 – Caso 30 - Macrotick.

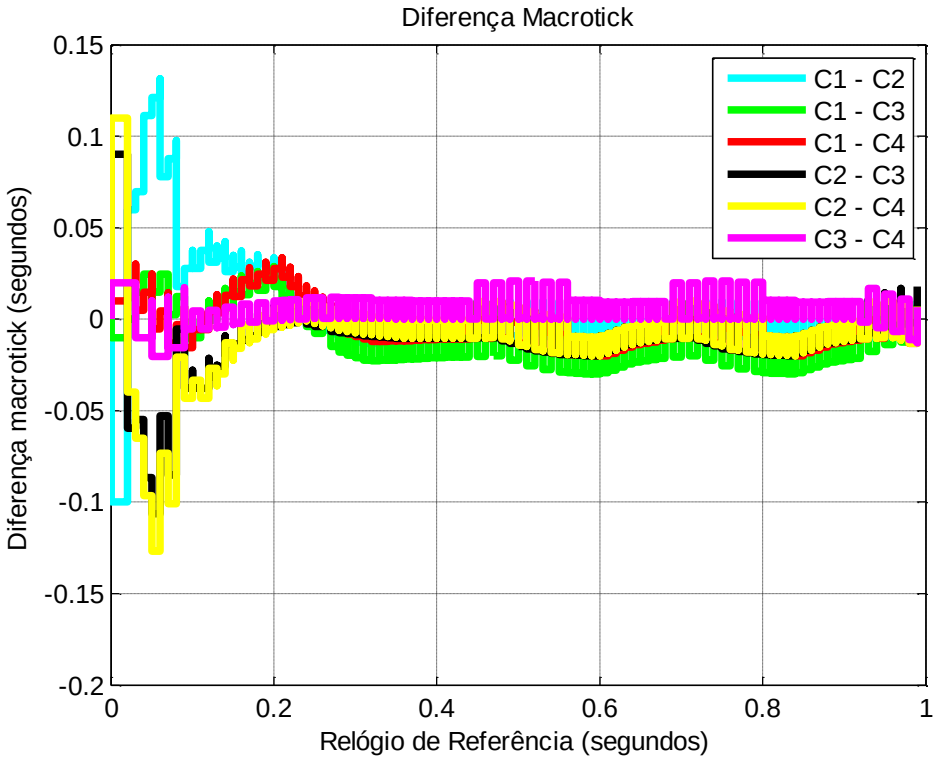


Figura 8.162 – Caso 30 - Diferença entre Macroticks.

A Figura 8.165 mostra a correção do modo NOMO calculada pela função de convergência do algoritmo SR vezes a função *cross-fading* degrau. A Figura 8.166 mostra a correção do MMCF, feito pelo modo NOMD. O tempo de transição entre os modos STM e NOMO de correção é de 10 macroticks, entre o NOMO e o NOMD é a cada 3 períodos de re-sincronizações do modo NOMO e o período de re-sincronização é de 10 microticks, ou seja, após 10 microticks a correção é recalculada. No entanto, com o uso da função degrau, a cada instante é aplicada uma correção multiplicada pelo valor da função *cross-fading*.

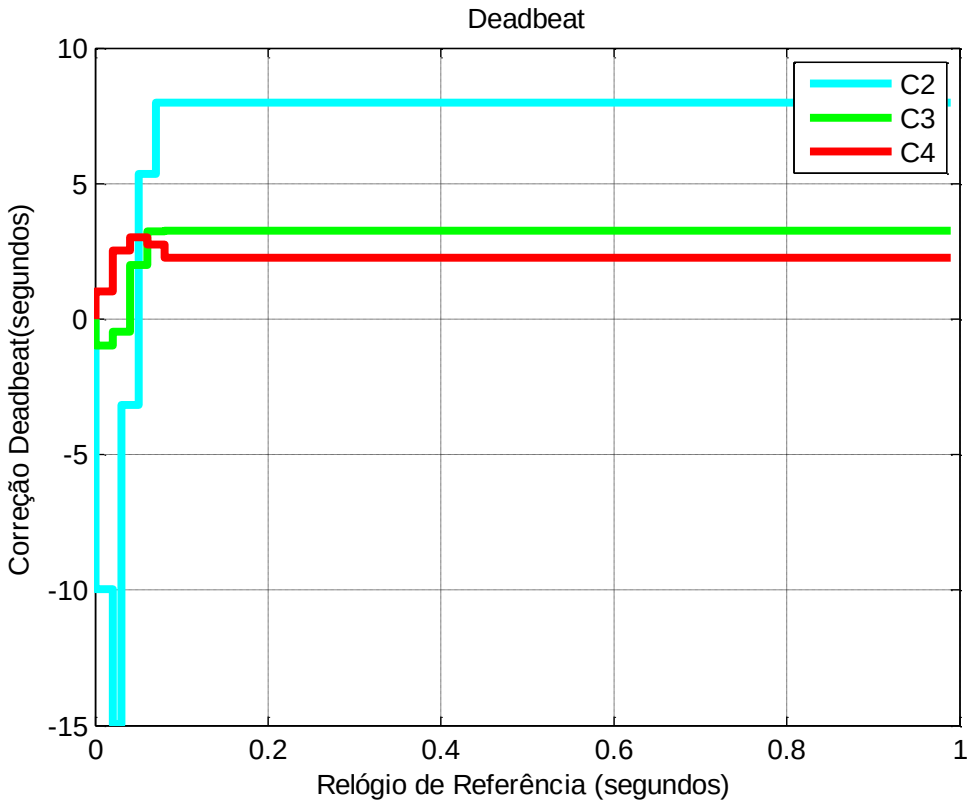


Figura 8.163 – Caso 30 - Modo STM - Algoritmo PAReS Degrau.

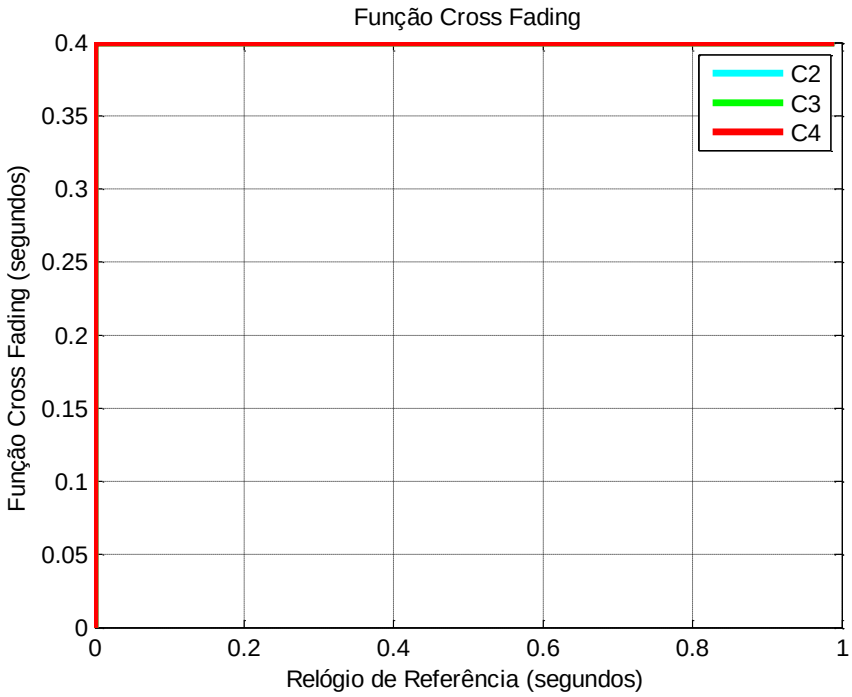


Figura 8.164 – Caso 30 - Função Cross-Fading Degrau.

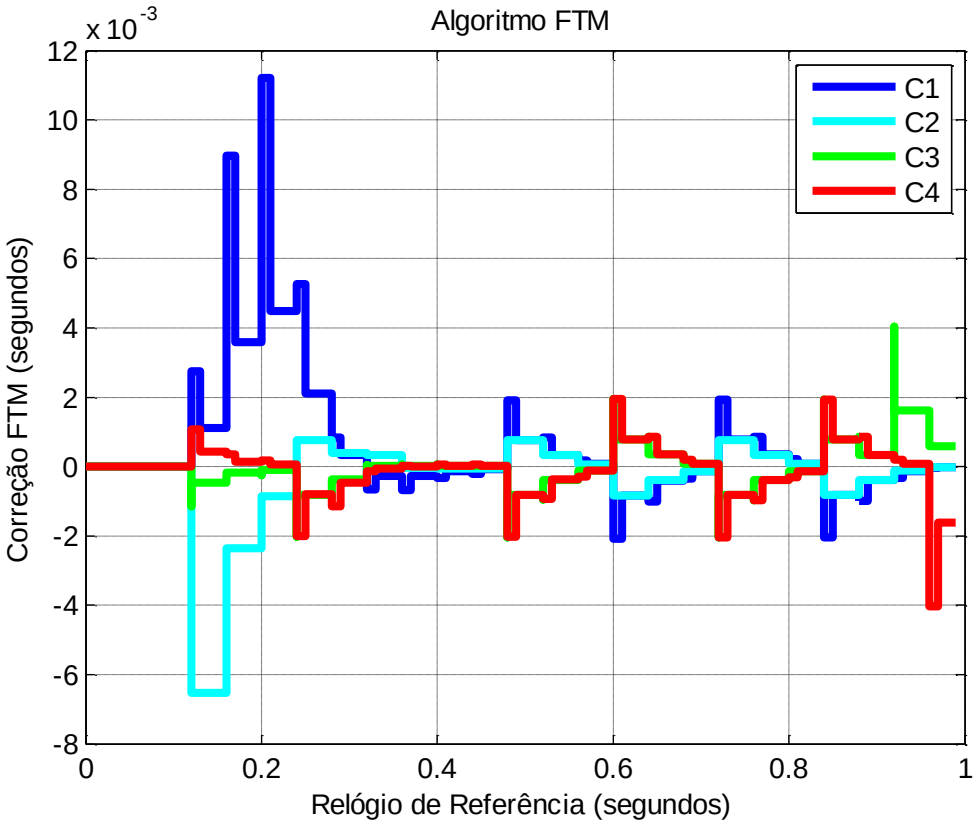


Figura 8.165 – Caso 30 - Correção modo NOMO - Algoritmo PAReS Degrau.

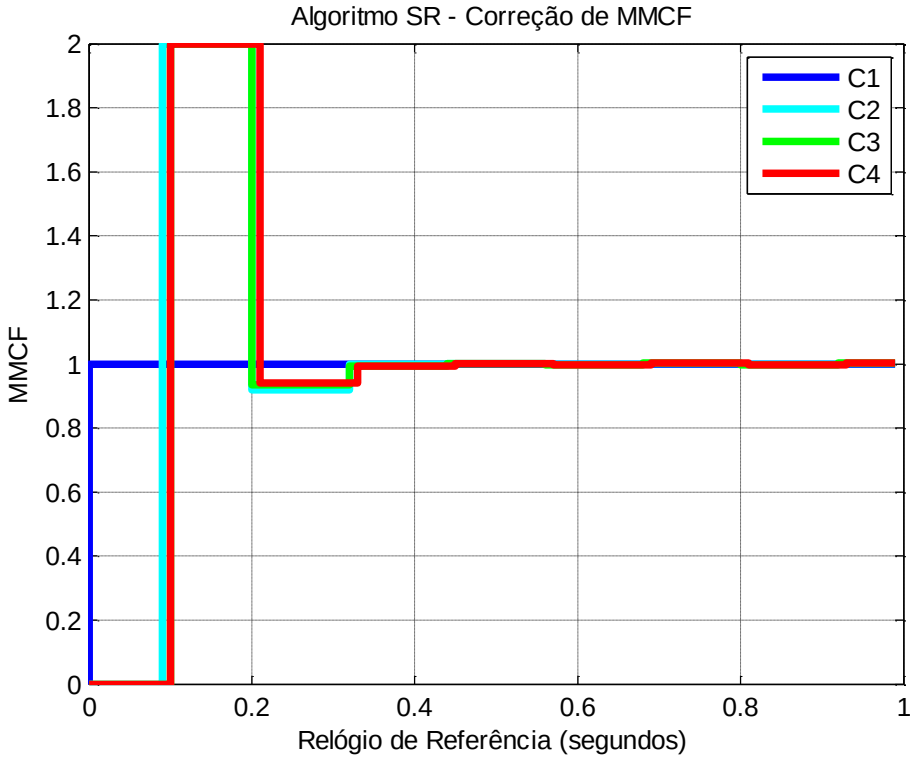


Figura 8.166 – Caso 30 - Correção modo NOMD - Algoritmo PAReS Degrau.

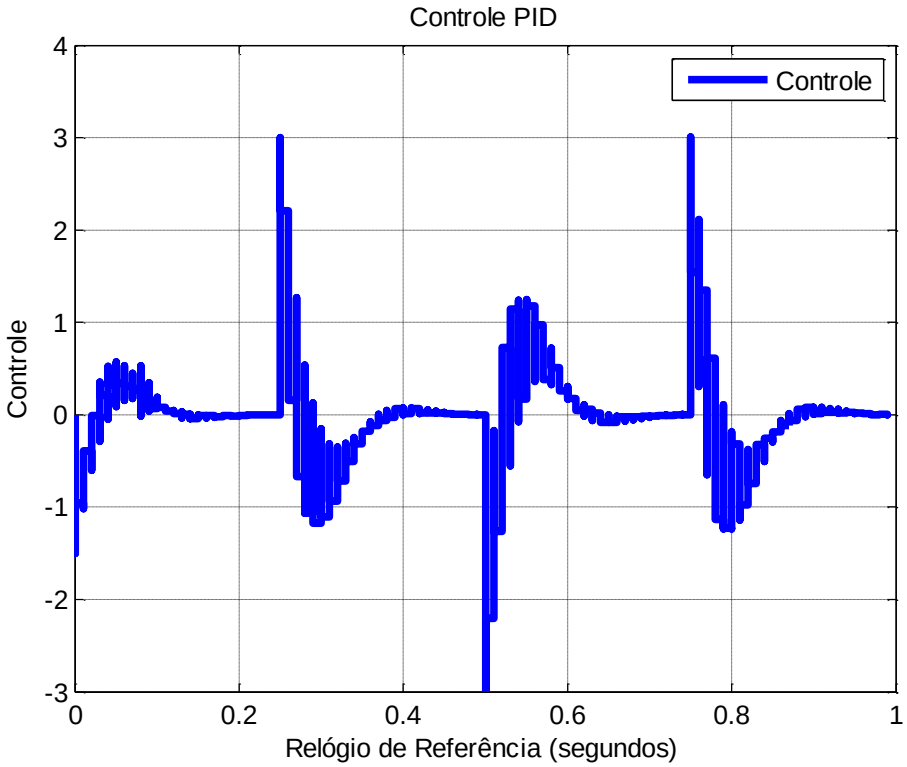


Figura 8.167 – Caso 30 - Controle PID.

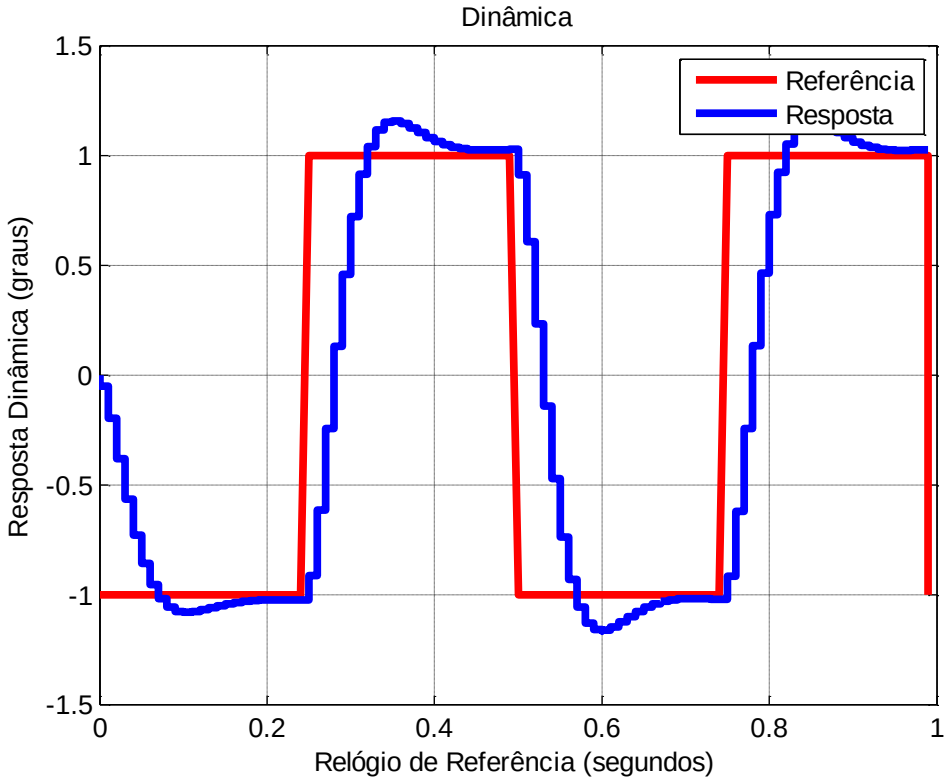


Figura 8.168 – Caso 30 - Resposta Dinâmica.

Este caso mostra uma sincronização de relógios com o algoritmo PAReS degrau que utiliza um controlador *deadbeat* no modo STM, a função de convergência do algoritmo FTM com *cross-fading* degrau no modo NOMO, e a correção do MMCF em relação a uma referência (no caso C1) no modo NOMD sobre um sistema de controle por redes em uma rede CSMA/CD. No modo STM, não é realizado o controle da dinâmica, pois é um modo de inicialização de tempo. No modo NOMO o algoritmo de sincronização se mostrou eficiente para estabelecer a sincronização. O modo NOMD converge os MMCFs sem problemas. Além disso, o caso obteve uma boa resposta dinâmica com um sobressinal médio. A de-sincronização inicial que antes degradava o controle agora foi corrigida aplicando a reconfiguração dos algoritmos de tempo; além disso, o algoritmo suaviza a descontinuidade de tempo. E busca obter a sincronização precisa e exata. Esta técnica se mostrou eficiente para sincronizar relógios em sistemas de controle por redes e pode ser melhorada através da otimização dos parâmetros.

8.33. Caso 31 - Algoritmo PAReS Rampa - CSMA/CD

Este caso possui valores de viés inicial igual a: C1: 0; C2: 0.1 seg. ; C3: 0.01 seg. ; C4: -0.1 seg.; e deriva igual a: C1: 0; C2: 1%; C3: 0.01%; C4: -0.01%. Neste caso, aplica-se um algoritmo com os três modos de correção: 1) STM; 2) NOMO; e 3) NOMD, descritos no capítulo 6. Com isto, visa-se inicialmente corrigir o viés inicial com o controlador *deadbeat*, depois utiliza-se a técnica FTM em conjunto com uma função *cross-fading* rampa e depois corrige-se o MMCF com o C1 de referência, definindo o algoritmo PAReS rampa, sobre um sistema de controle por redes em uma rede CSMA/CD. A Figura 8.169 e Figura 8.170 mostram, respectivamente, os valores de macrotick e diferença entre o relógio de referência e os macroticks. A Figura 8.171 mostra o valor de correção do relógio calculado pelo controlador *deadbeat* no modo STM. A Figura 8.172 mostra a função *cross-fading* rampa.

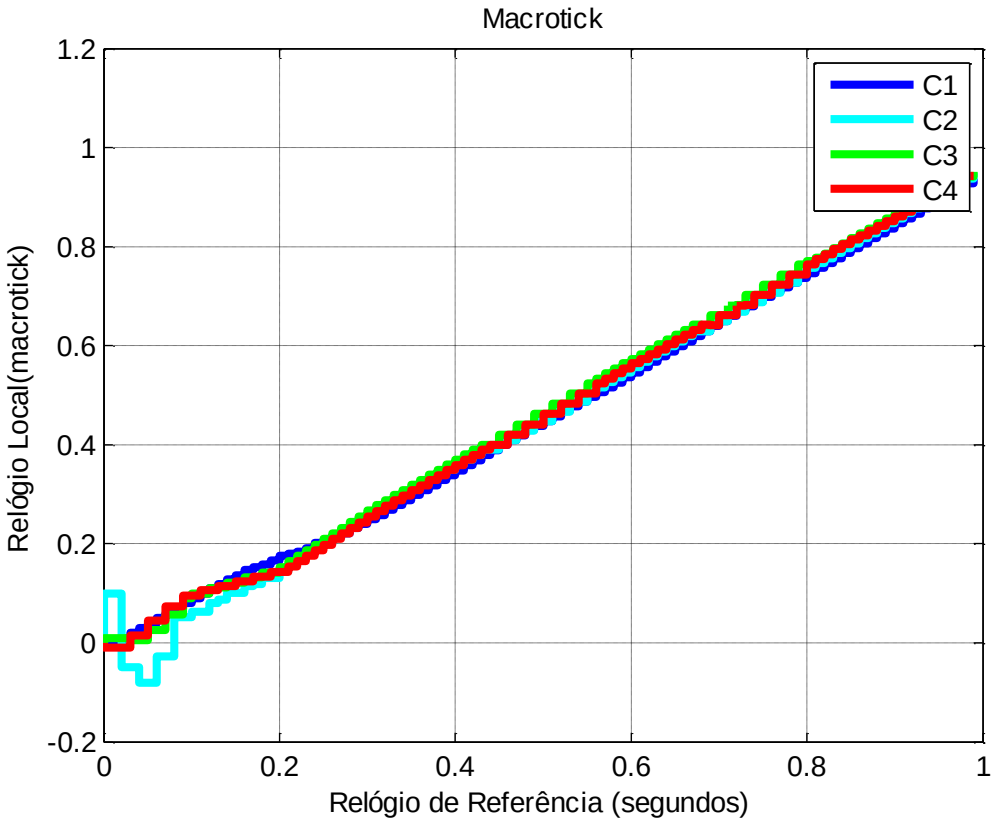


Figura 8.169 – Caso 31 - Macrotick.

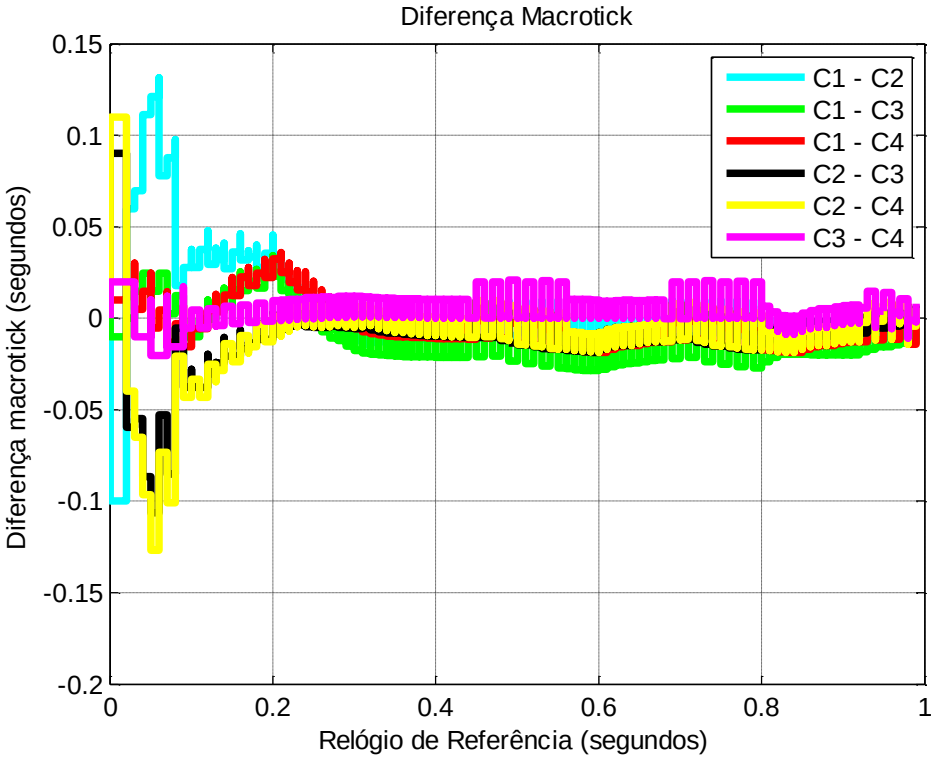


Figura 8.170 – Caso 31 - Diferença entre Macroticks.

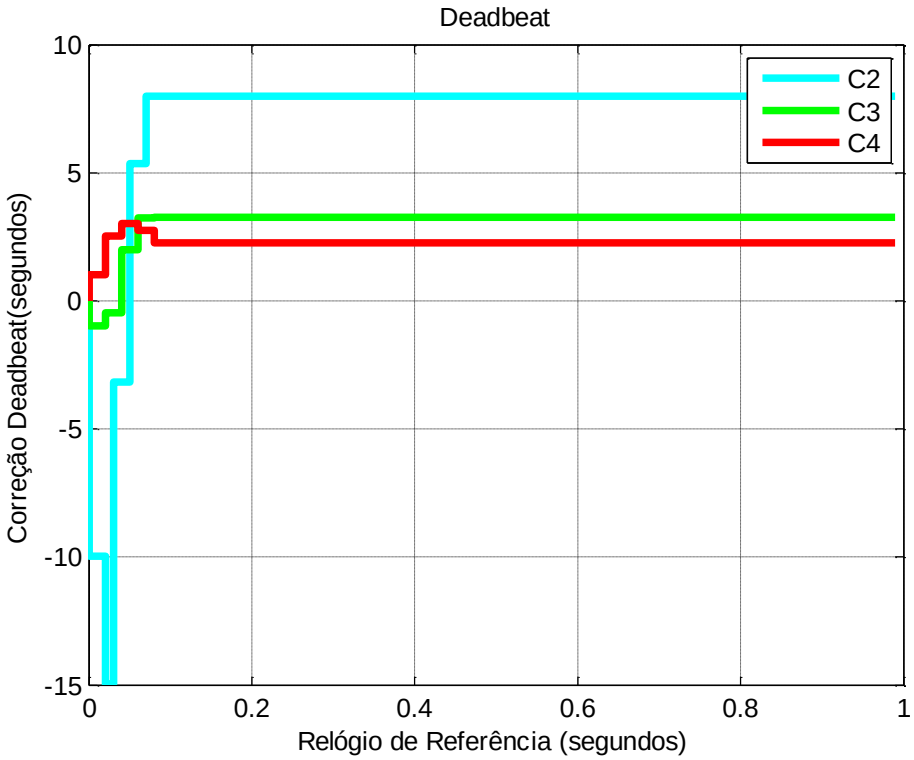


Figura 8.171 – Caso 31 - Modo STM - Algoritmo PAReS Rampa.

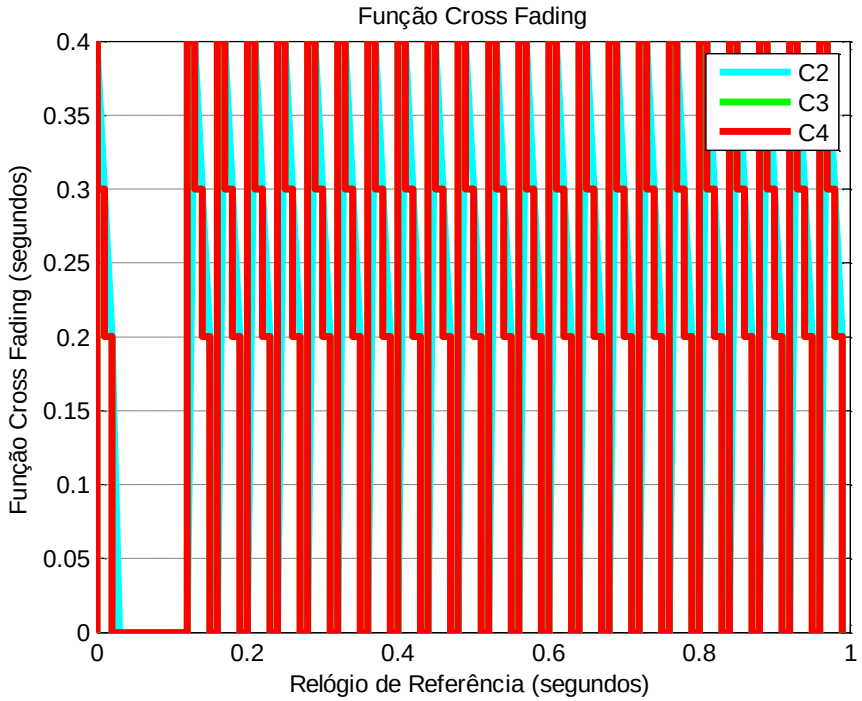


Figura 8.172 – Caso 30 - Função Cross-Fading Rampa.

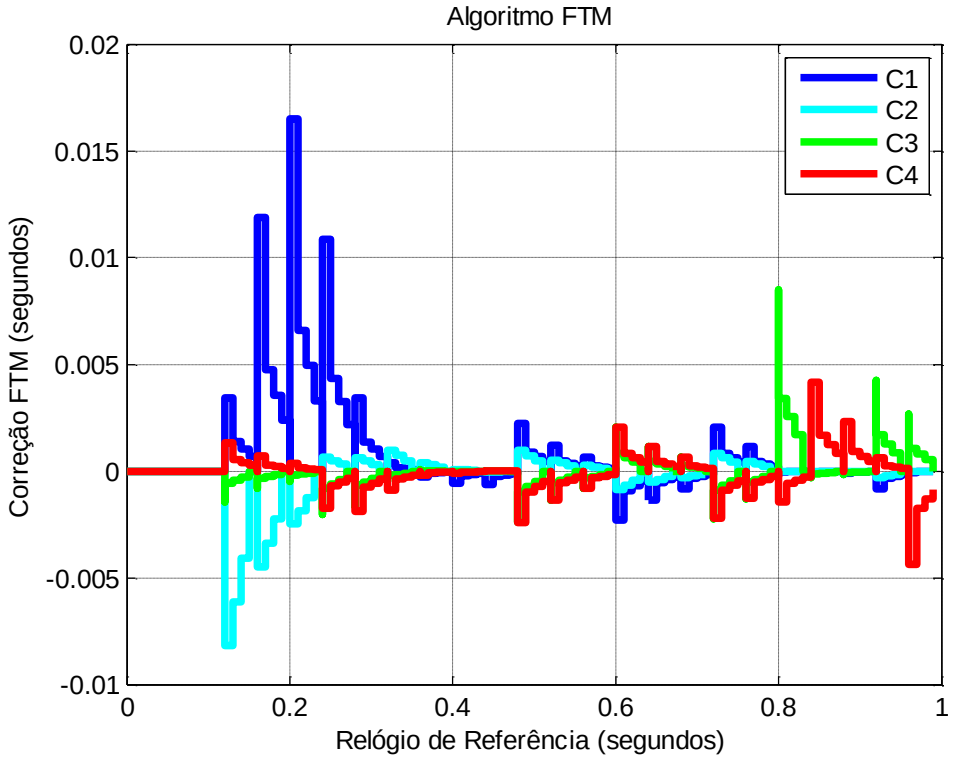


Figura 8.173 – Caso 31 - Correção modo NOMO - Algoritmo PAReS Rampa.

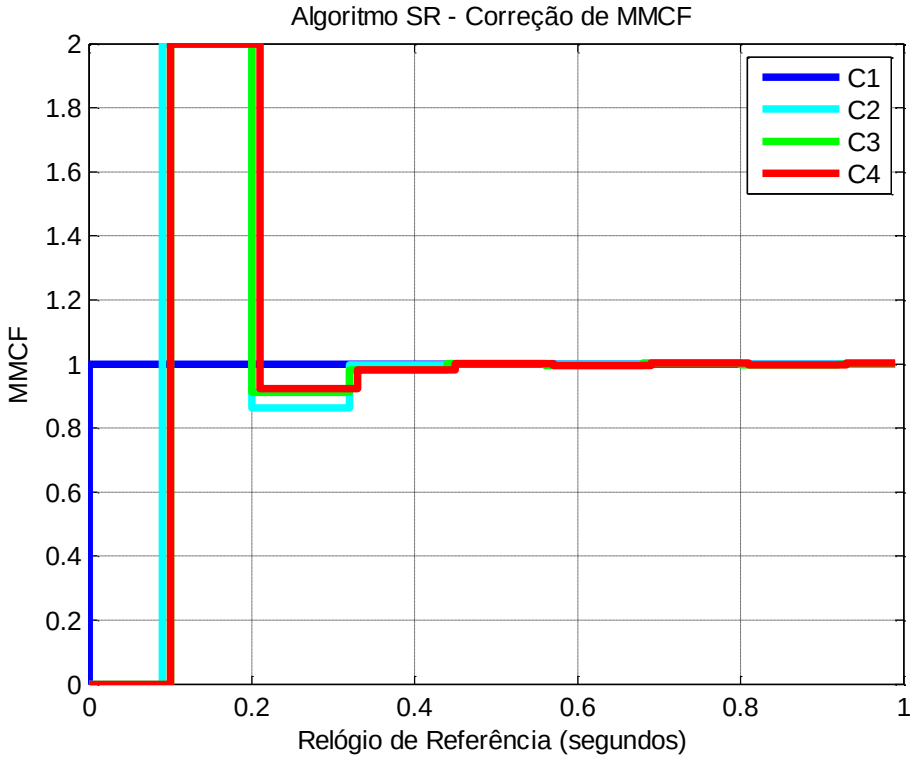


Figura 8.174 – Caso 31 - Correção modo NOMD - Algoritmo PAReS Rampa.

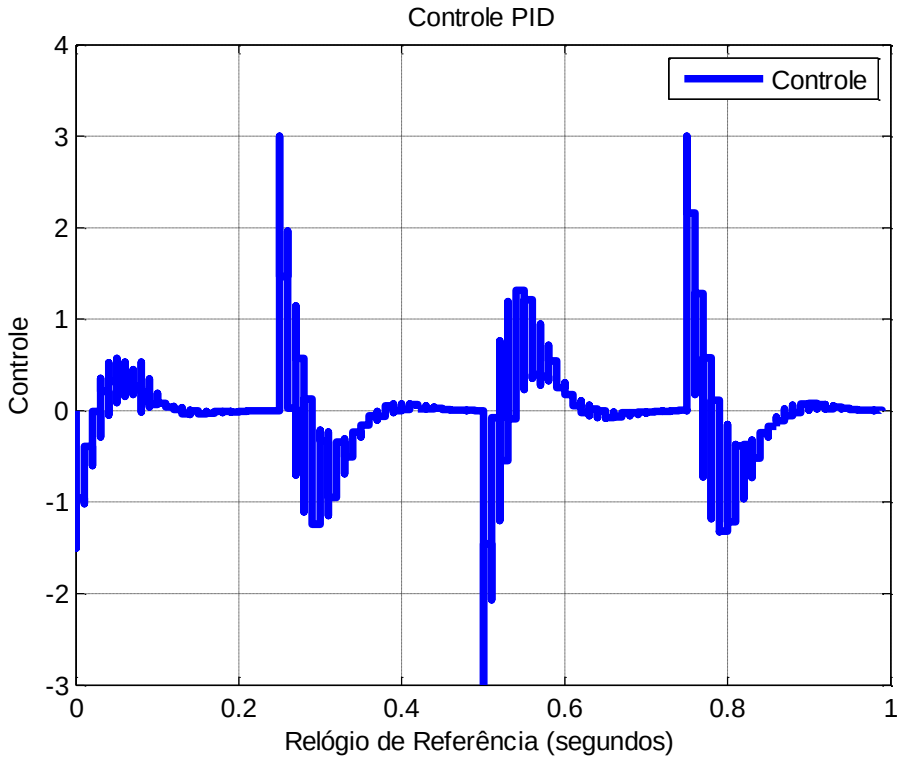


Figura 8.175 – Caso 31 - Controle PID.

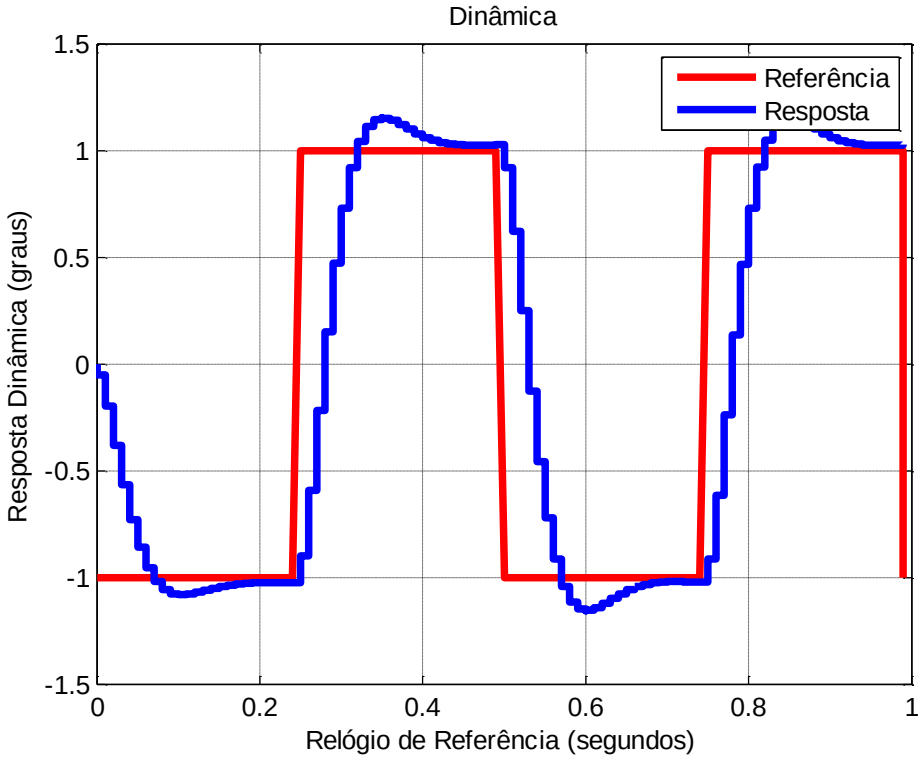


Figura 8.176 – Caso 31 - Resposta Dinâmica.

A Figura 8.173 mostra a correção do modo NOMO calculada pela função de convergência do algoritmo SR vezes a função *cross-fading* rampa. A Figura 8.174 mostra a correção do MMCF, feito pelo modo NOMD. O tempo de transição entre os modos STM e NOMO de correção é de 10 macroticks, entre o NOMO e o NOMD é a cada 3 períodos de re-sincronizações do modo NOMO e o período de re-sincronização é de 10 microticks, ou seja, após 10 microticks a correção é recalculada. No entanto, com o uso da função rampa, a cada instante é aplicada uma correção multiplicada pelo valor da função *cross-fading*.

Este caso mostra uma sincronização de relógios com o algoritmo PAReS degrau que utiliza um controlador *deadbeat* no modo STM, a função de convergência do algoritmo FTM com *cross-fading* rampa no modo NOMO, e a correção do MMCF em relação a uma referência (no caso C1) no modo NOMD sobre um sistema de controle por redes em uma rede CSMA/CD. No modo STM, não é realizado o controle da dinâmica, pois é um modo de inicialização

de tempo. No modo NOMO o algoritmo de sincronização se mostrou eficiente para estabelecer a sincronização. O modo NOMD converge os MMCFs sem problemas. Além disso, o caso obteve uma boa resposta dinâmica com um sobressinal médio. A de-sincronização inicial que antes degradava o controle agora foi corrigida aplicando a reconfiguração dos algoritmos de tempo; além disso, o algoritmo suaviza a descontinuidade de tempo. Mas, no caso da reta, é necessário uma segunda transição para evitar que o valor da função *cross-fading* rampa fique negativa e inverta a fase do sistema tornando-o instável. E busca obter a sincronização precisa e exata. Esta técnica se mostrou eficiente para sincronizar relógios em sistemas de controle por redes e pode ser melhorada através da otimização dos parâmetros.

8.34. Caso 32 - Algoritmo PAReS Exponencial - CSMA/CD

Este caso possui valores de viés inicial igual a: C1: 0; C2: 0.1 seg. ; C3: 0.01 seg. ; C4: -0.1 seg.; e deriva igual a: C1: 0; C2: 1%; C3: 0.01%; C4: -0.01%. Neste caso, aplica-se um algoritmo com os três modos de correção: 1) STM; 2) NOMO; e 3) NOMD, descritos no capítulo 6. Com isto, visa-se inicialmente corrigir o viés inicial com o controlador *deadbeat*, depois utiliza-se a técnica FTM em conjunto com uma função *cross-fading* exponencial e depois corrige-se o MMCF com o C1 de referência, definindo o algoritmo PAReS exponencial, sobre um sistema de controle por redes em uma rede CSMA/CD. A Figura 8.177 e Figura 8.178 mostram, respectivamente, os valores de macrotick e diferença entre o relógio de referência e os macroticks. A Figura 8.179 mostra o valor de correção do relógio calculado pelo controlador *deadbeat* no modo STM. A Figura 8.180 mostra a função *cross-fading* exponencial.

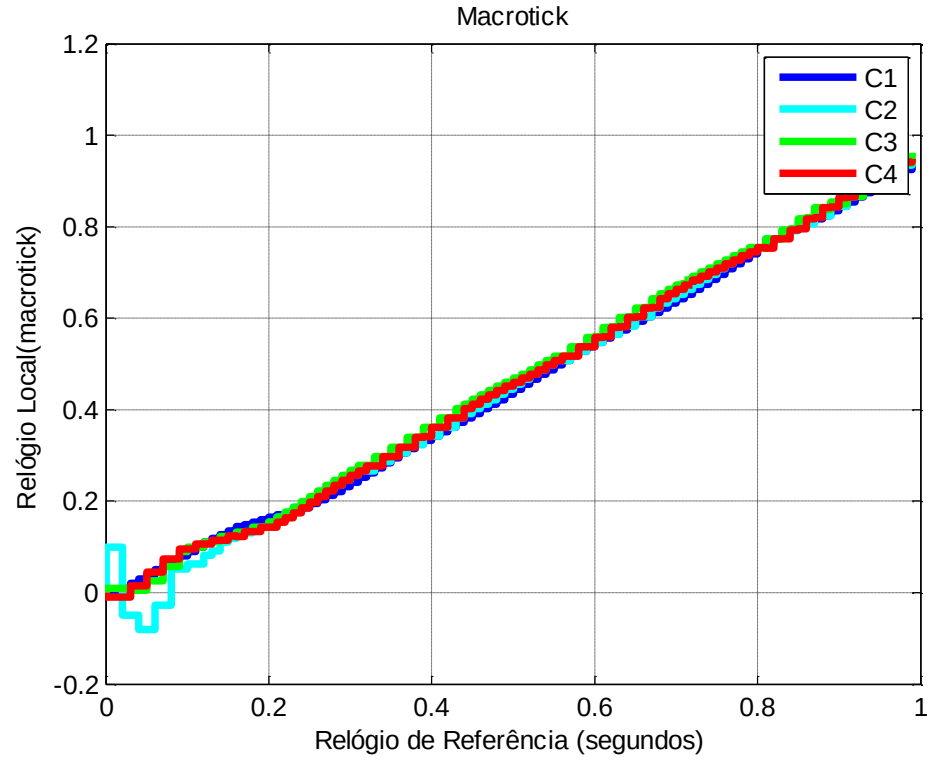


Figura 8.177 – Caso 32 - Macrotick.

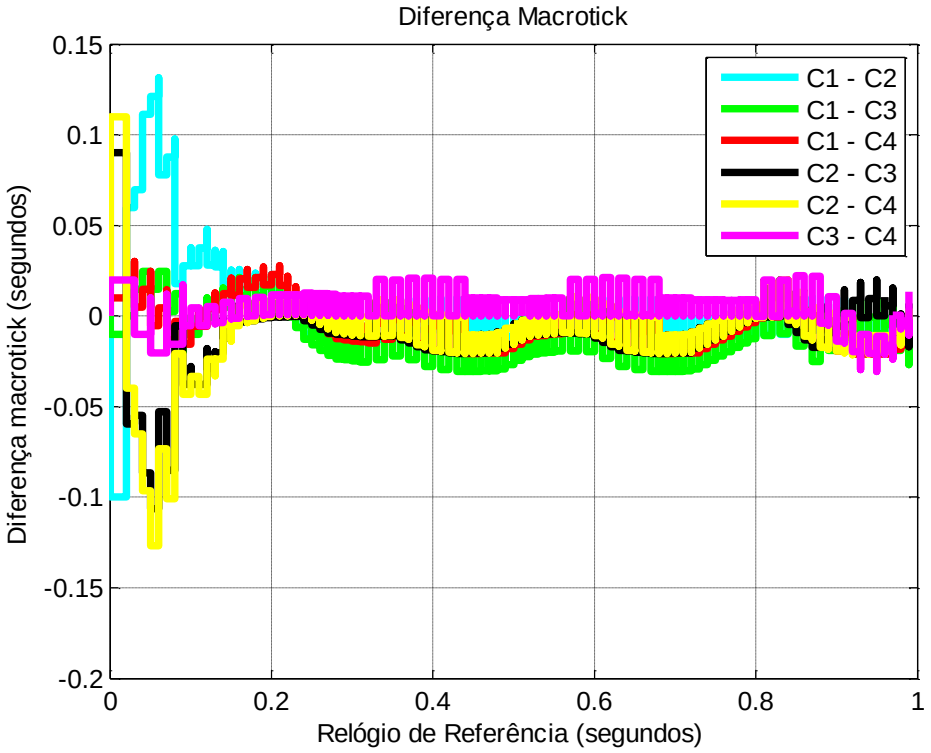


Figura 8.178 – Caso 32 - Diferença entre Macroticks.

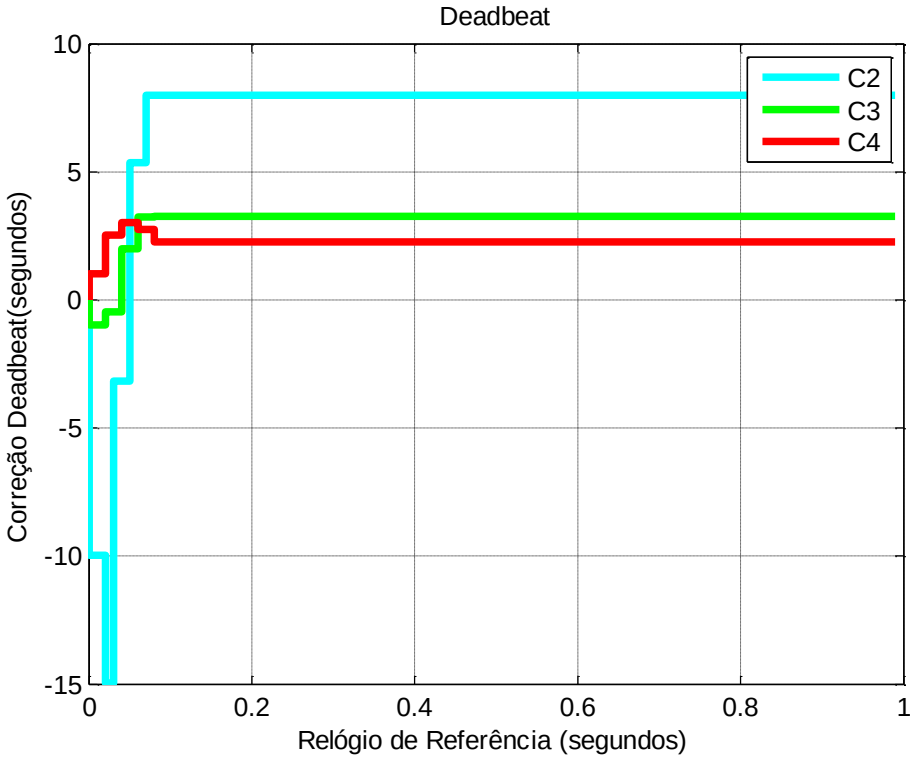


Figura 8.179 – Caso 32 - Modo STM - Algoritmo PAReS Exponencial.

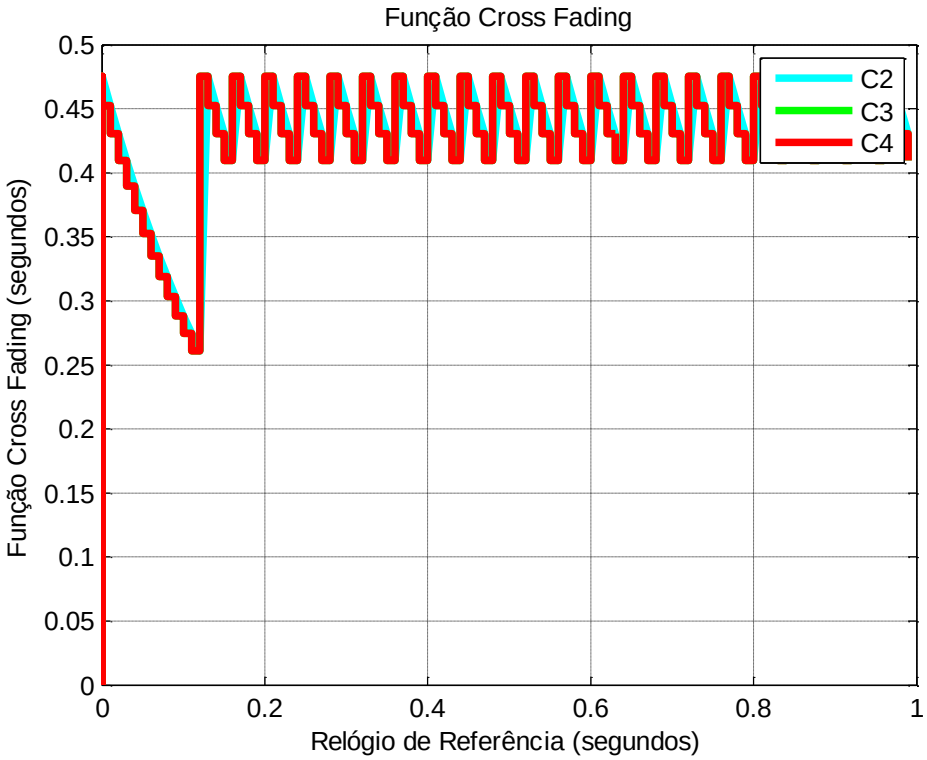


Figura 8.180 – Caso 32 - Função Cross-Fading Exponencial.

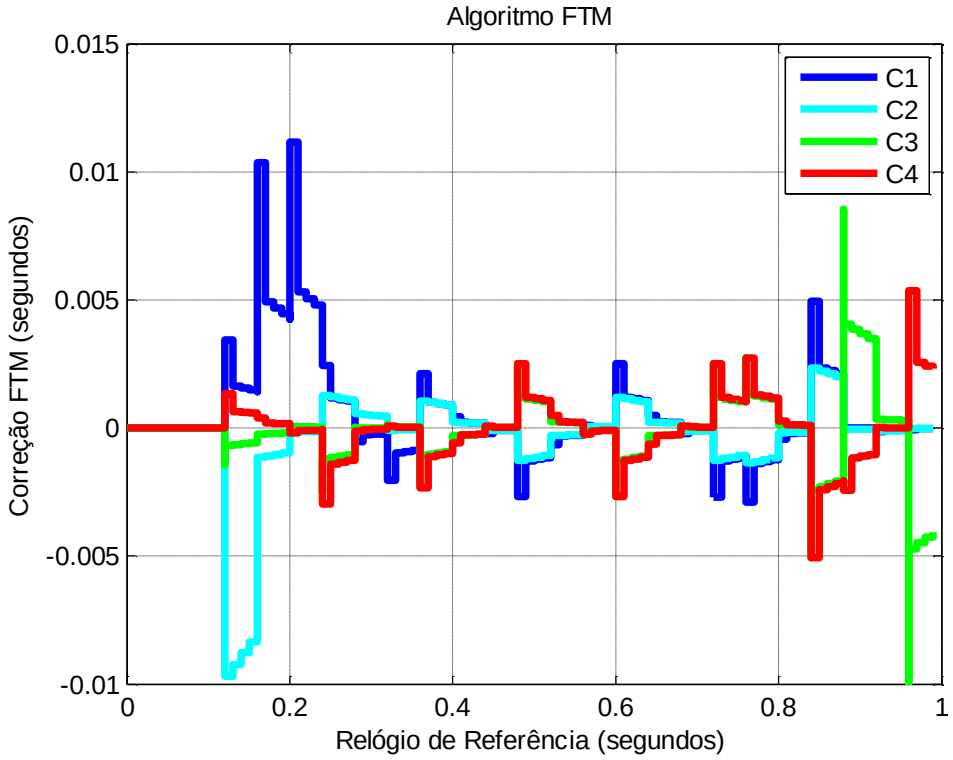


Figura 8.181 – Caso 32 - Correção modo NOMO - Algoritmo PAReS Exponencial.

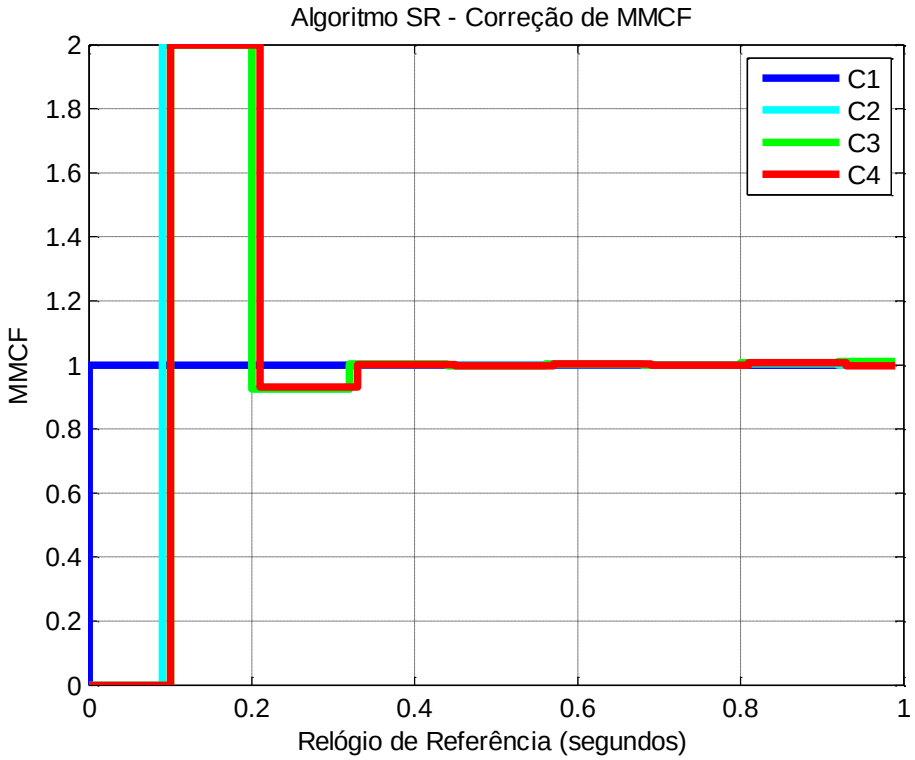


Figura 8.182 – Caso 32 - Correção modo NOMD - Algoritmo PAReS Exponencial.

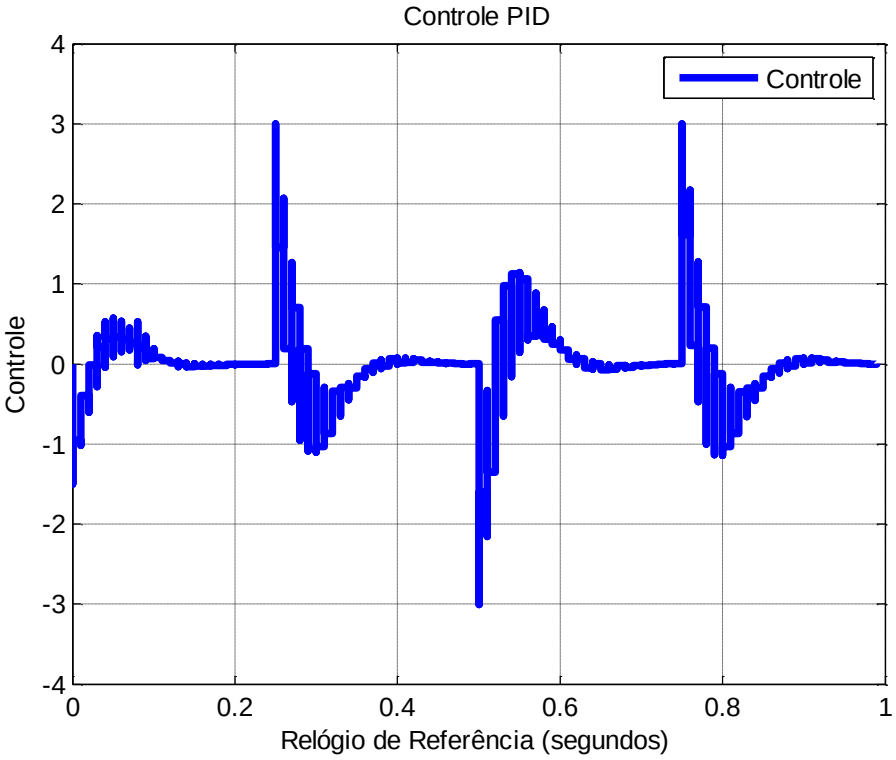


Figura 8.183 – Caso 32 - Controle PID.

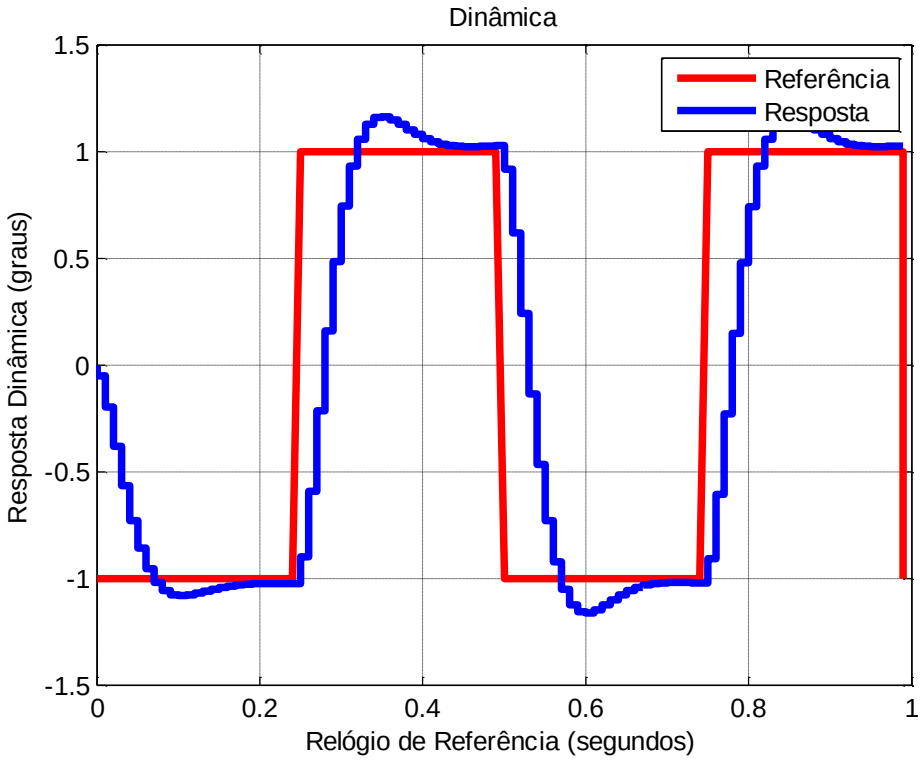


Figura 8.184 – Caso 32 - Resposta Dinâmica.

A Figura 8.181 mostra a correção do modo NOMO calculada pela função de convergência do algoritmo SR vezes a função *cross-fading* exponencial. A Figura 8.182 mostra a correção do MMCF, feito pelo modo NOMD. O tempo de transição entre os modos STM e NOMO de correção é de 10 macroticks, entre o NOMO e o NOMD é a cada 3 períodos de re-sincronizações do modo NOMO e o período de re-sincronização é de 10 microticks, ou seja, após 10 microticks a correção é recalculada. No entanto, com o uso da função exponencial, a cada instante é aplicada uma correção multiplicada pelo valor da função *cross-fading*.

Este caso mostra uma sincronização de relógios com o algoritmo PAReS exponencial que utiliza um controlador *deadbeat* no modo STM, a função de convergência do algoritmo FTM com *cross-fading* exponencial no modo NOMO, e a correção do MMCF em relação a uma referência (no caso C1) no modo NOMD sobre um sistema de controle por redes em uma rede CSMA/CD. No modo STM, não é realizado o controle da dinâmica, pois é um modo de inicialização de tempo. No modo NOMO o algoritmo de sincronização se mostrou eficiente para estabelecer a sincronização. O modo NOMD converge os MMCFs sem problemas. Além disso, o caso obteve uma boa resposta dinâmica com um sobressinal médio. A de-sincronização inicial que antes degradava o controle agora foi corrigida aplicando a reconfiguração dos algoritmos de tempo; além disso, o algoritmo suaviza a descontinuidade de tempo. Mas, no caso da exponencial, não é necessário uma segunda transição para evitar que o valor da função *cross-fading* exponencial fique negativa. O algoritmo busca obter a sincronização precisa e exata. Esta técnica se mostrou eficiente para sincronizar relógios em sistemas de controle por redes e pode ser melhorada através da otimização dos parâmetros.

8.35. Caso 33 - Algoritmo AReS Degrau - TDMA

Os parâmetros e a sincronização de relógios é o mesmo da seção 8.12. A diferença é que este caso faz toda a simulação sobre uma rede TDMA. Assim, tem-se o seguinte resultado.

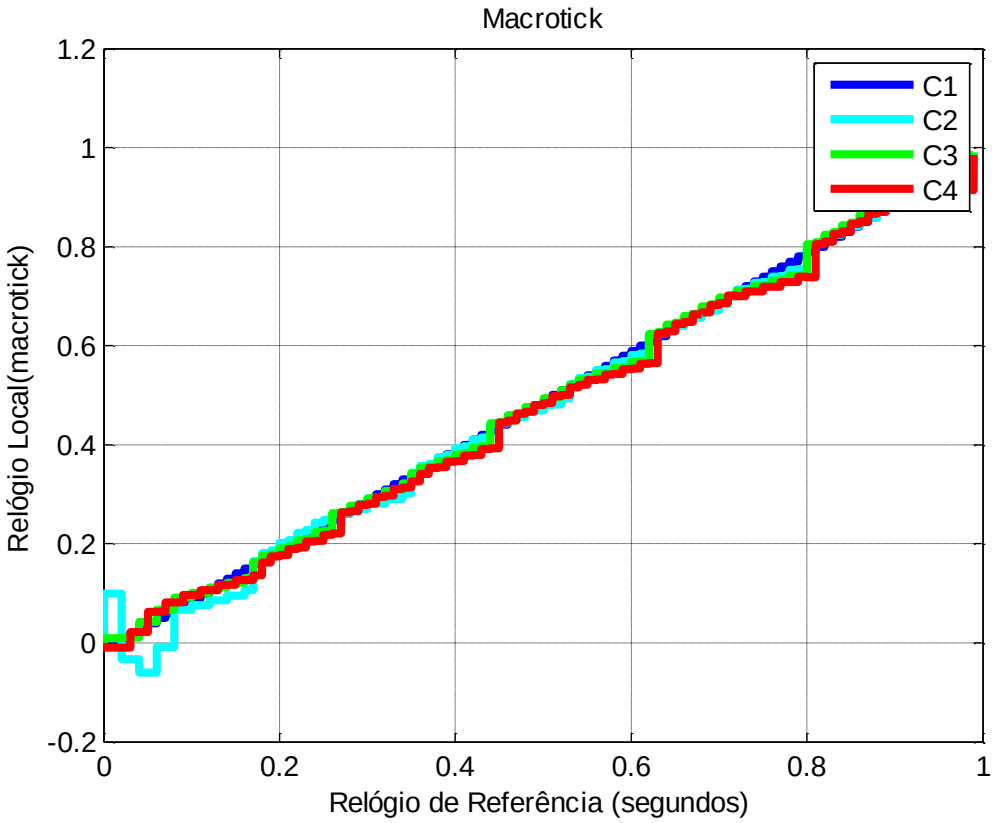


Figura 8.185 – Caso 33 - Macrotick.

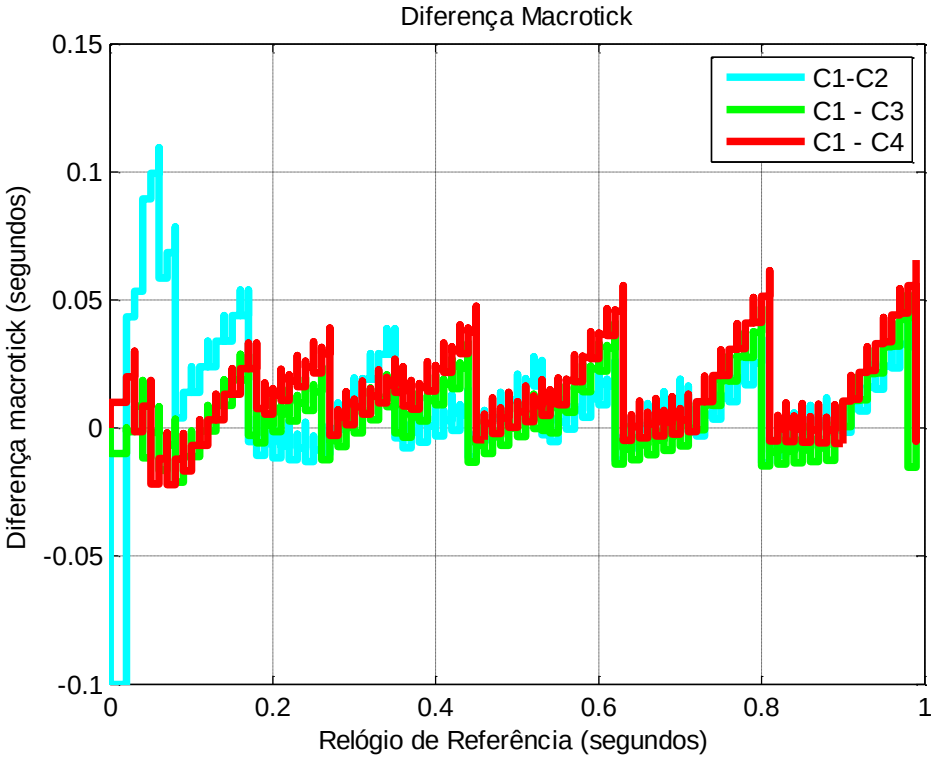


Figura 8.186 – Caso 33 - Diferença entre Macroticks.

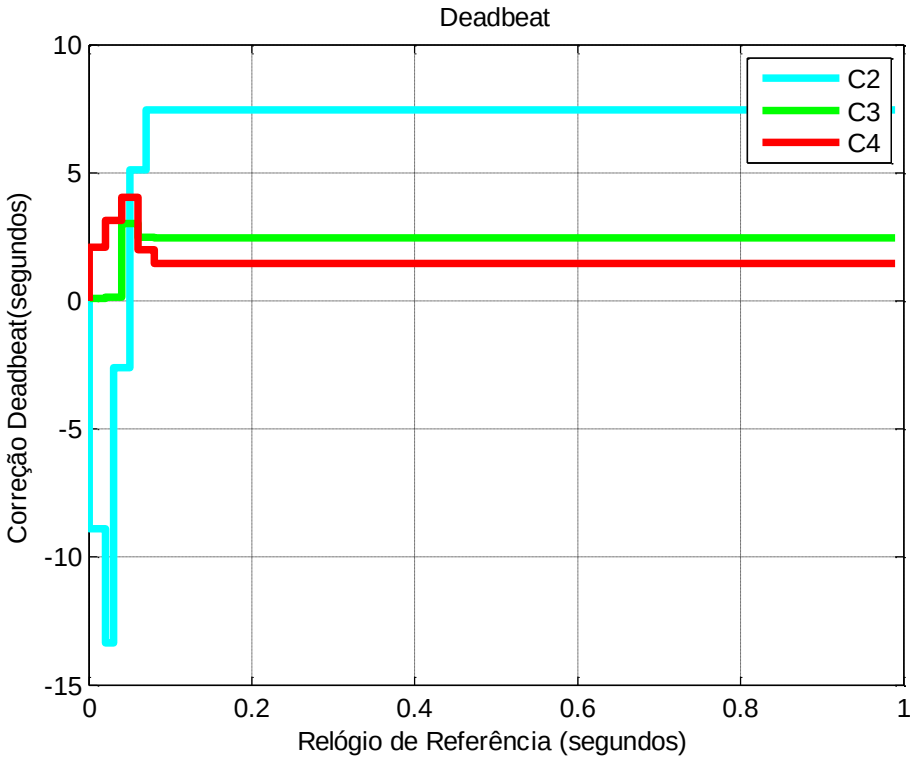


Figura 8.187 – Caso 33 - Modo STM - Algoritmo AReS Degrau - TDMA.

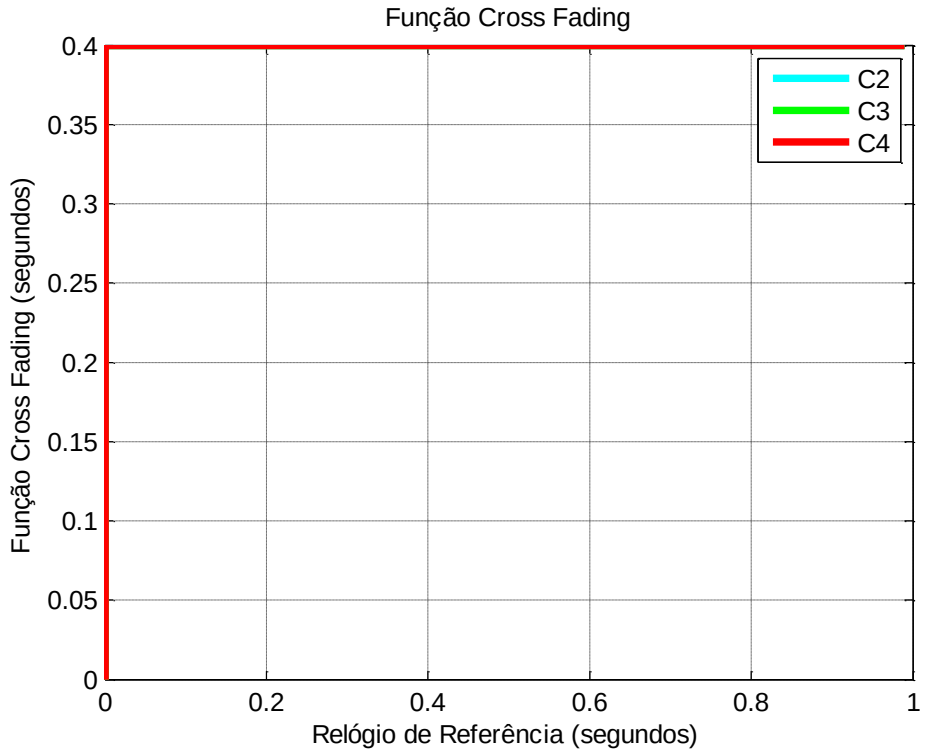


Figura 8.188 – Caso 33 - Função Cross-Fading Degrau.

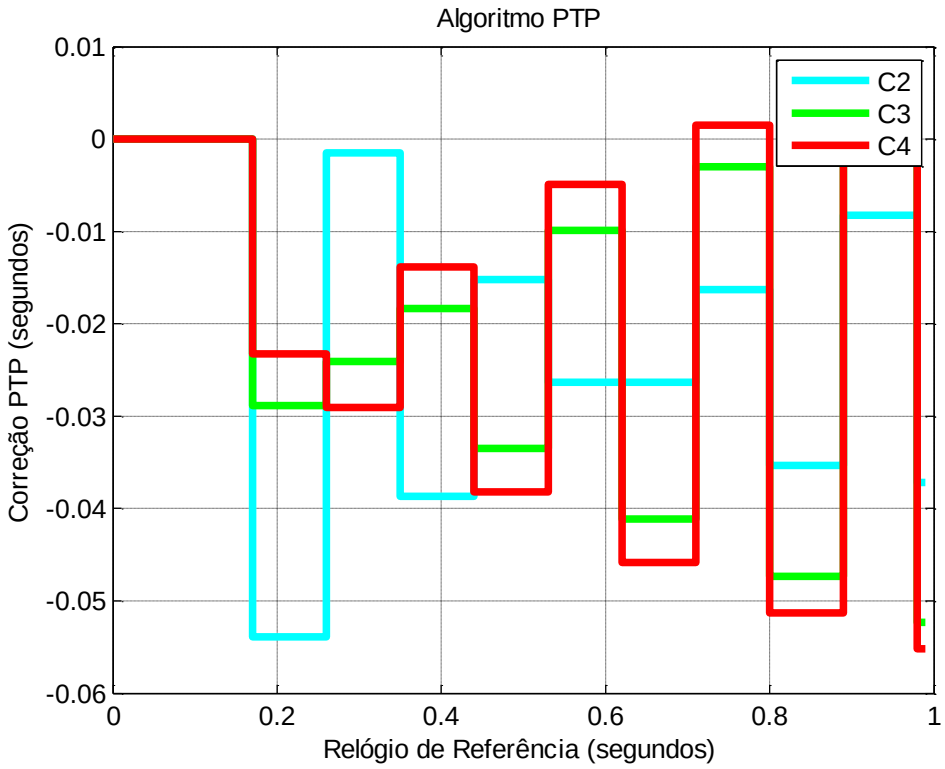


Figura 8.189 – Caso 33 - Correção modo NOMO - Algoritmo AReS Degrau TDMA.

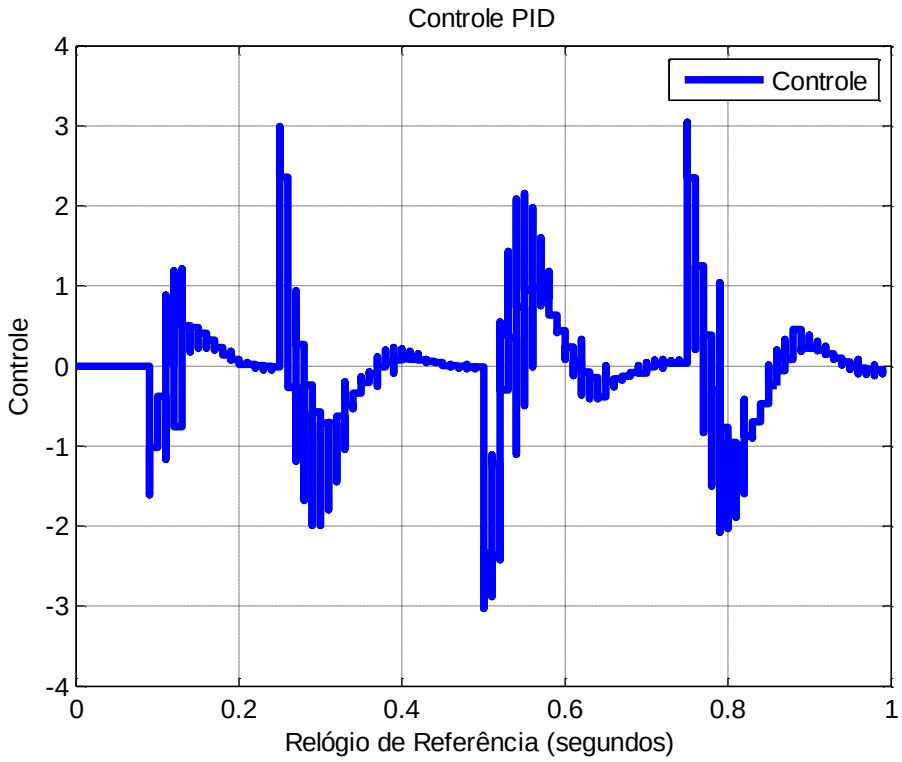


Figura 8.190 – Caso 33 - Controle PID.

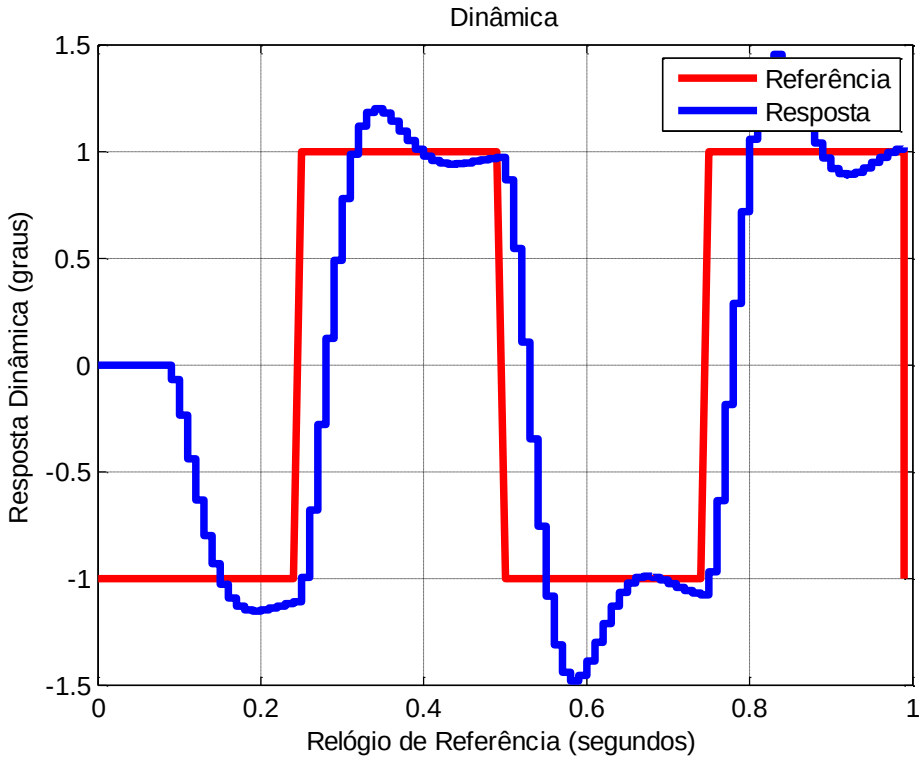


Figura 8.191 – Caso 33 - Resposta Dinâmica.

A principal diferença deste caso para o caso da seção 8.12 é que o gerenciamento de redes foi modificado. Em uma rede TDMA é possível estimar o pior caso de atraso de rede. Além disso, a rede TDMA aumenta o atraso de rede e por isso este caso teve um sobre sinal maior na resposta dinâmica e também uma exatidão pior em relação ao caso da seção 8.12.

8.36. Caso 34 - Algoritmo AReS Rampa - TDMA

Os parâmetros e a sincronização de relógios é o mesmo da seção 8.13. A diferença é que este caso faz toda a simulação sobre uma rede TDMA. Assim, tem-se o seguinte resultado.

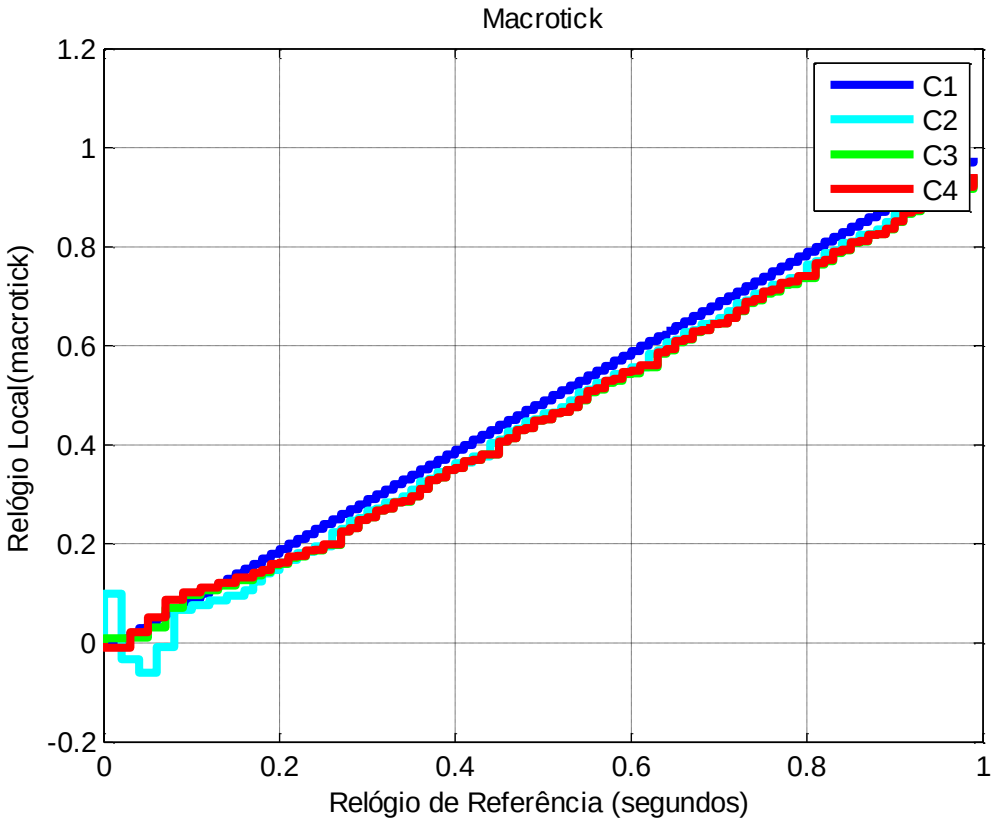


Figura 8.192 – Caso 34 - Macrotick.

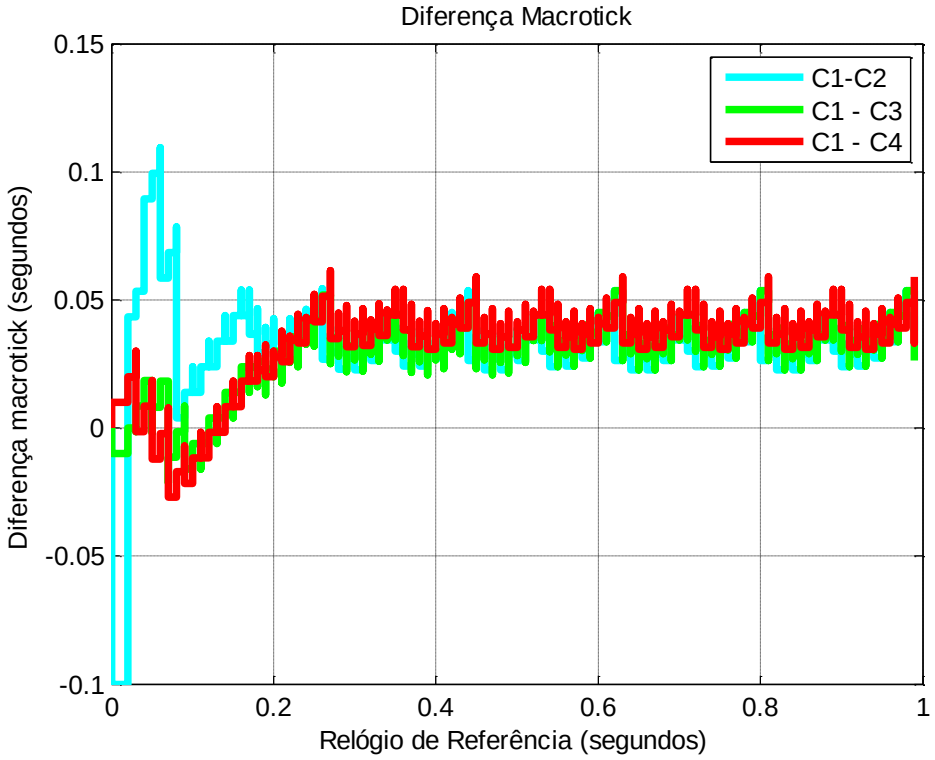


Figura 8.193 – Caso 34 - Diferença entre Macroticks.

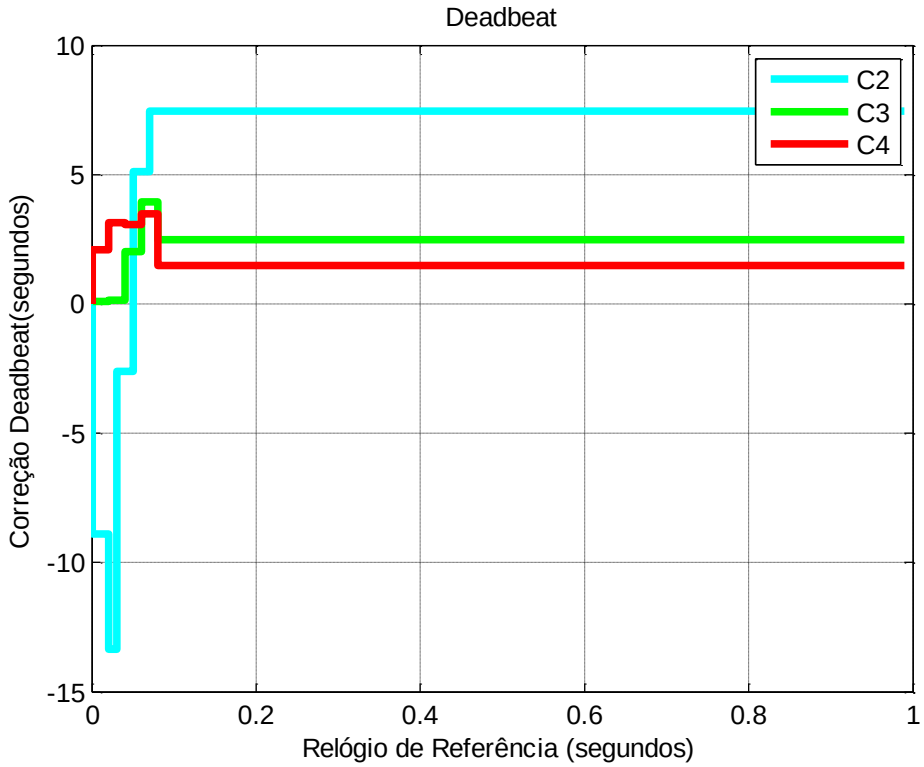


Figura 8.194 – Caso 34 - Modo STM - Algoritmo AReS Rampa - TDMA.

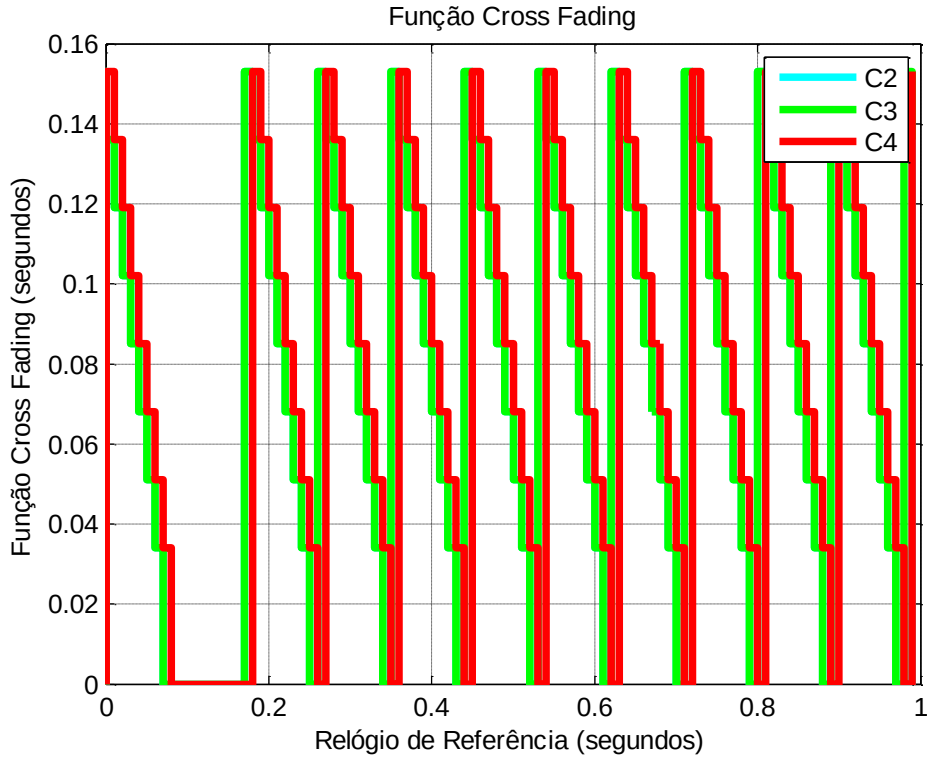


Figura 8.195 – Caso 34 - Função Cross-Fading Rampa.

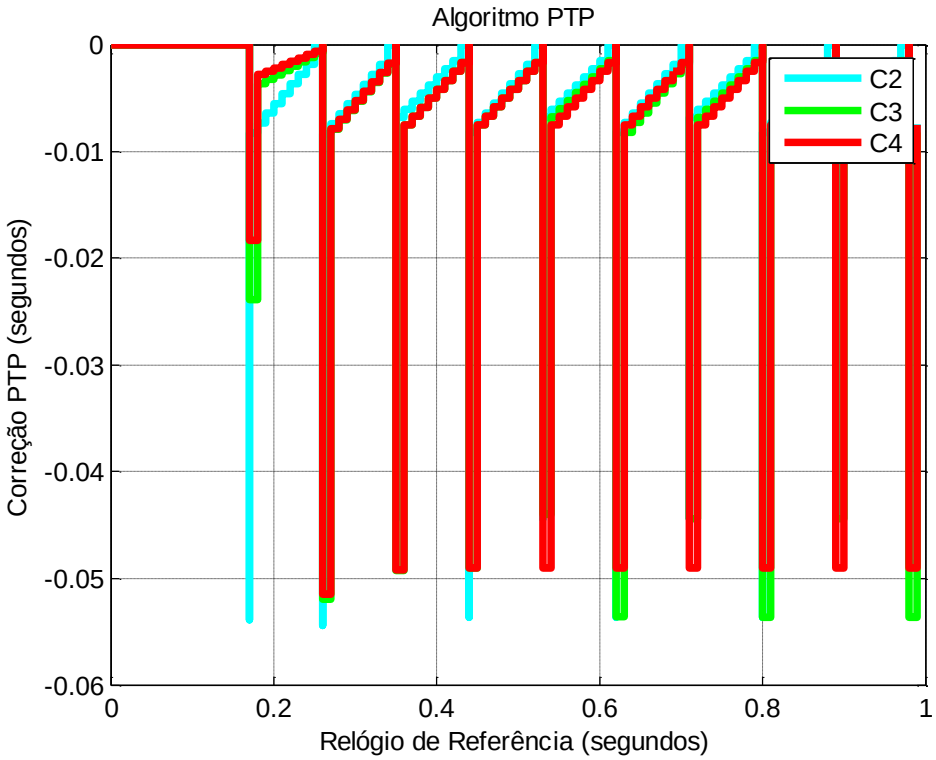


Figura 8.196 – Caso 34 - Correção modo NOMO - Algoritmo AReS Rampa TDMA.

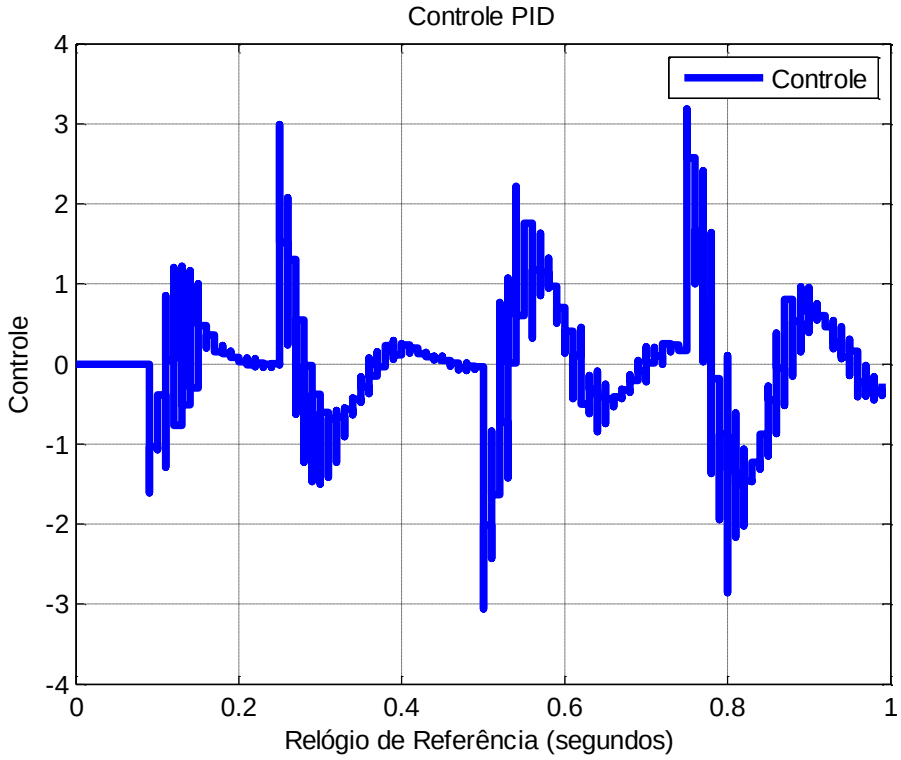


Figura 8.197 – Caso 34 - Controle PID.

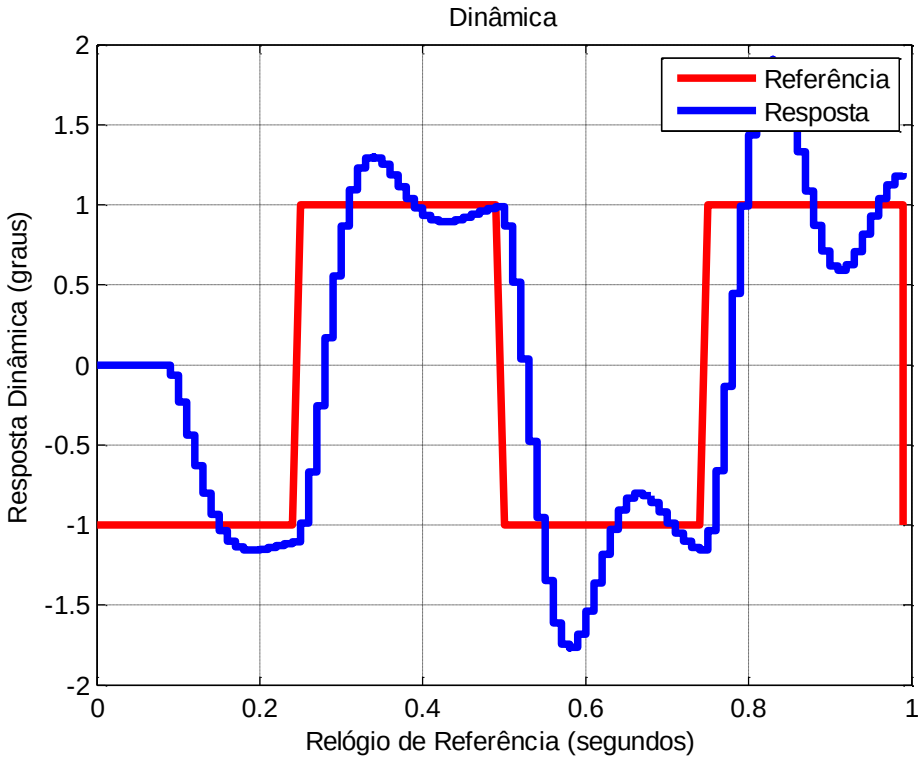


Figura 8.198 – Caso 34 - Resposta Dinâmica.

A principal diferença deste caso para o caso da seção 8.13 é que o gerenciamento de redes foi modificado. Em uma rede TDMA é possível estimar o pior caso de atraso de rede. Além disso, a rede TDMA aumenta o atraso de rede e por isso este caso teve um sobre sinal um pouco maior na resposta dinâmica e também uma exatidão pior em relação ao caso da seção 8.13.

8.37. Caso 35 - Algoritmo AReS Exponencial - TDMA

Os parâmetros e a sincronização de relógios é o mesmo da seção 8.14. A diferença é que este caso faz toda a simulação sobre uma rede TDMA. Assim, tem-se o seguinte resultado.

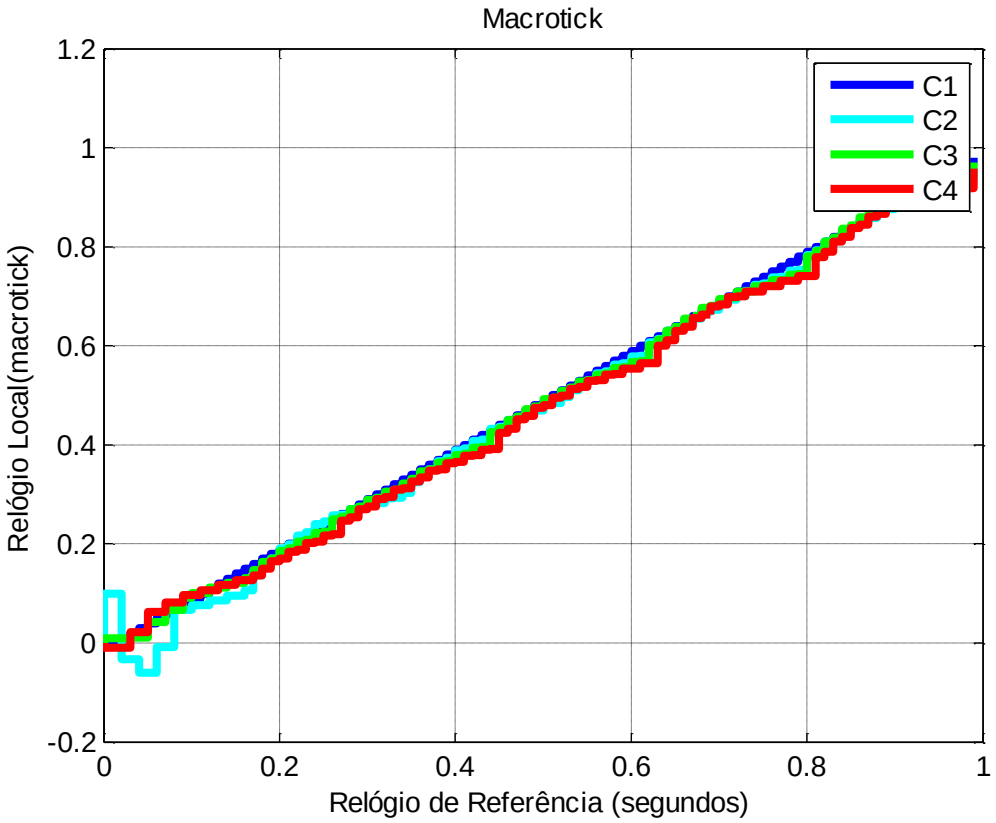


Figura 8.199 – Caso 35 - Macrotick.

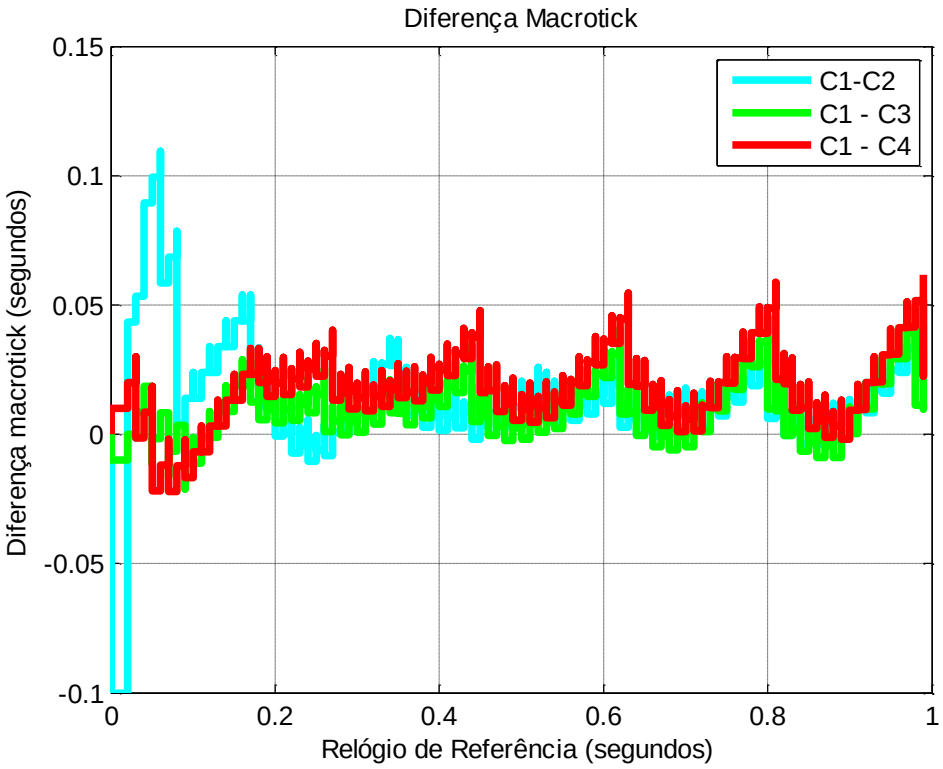


Figura 8.200 – Caso 35 - Diferença entre Macroticks.

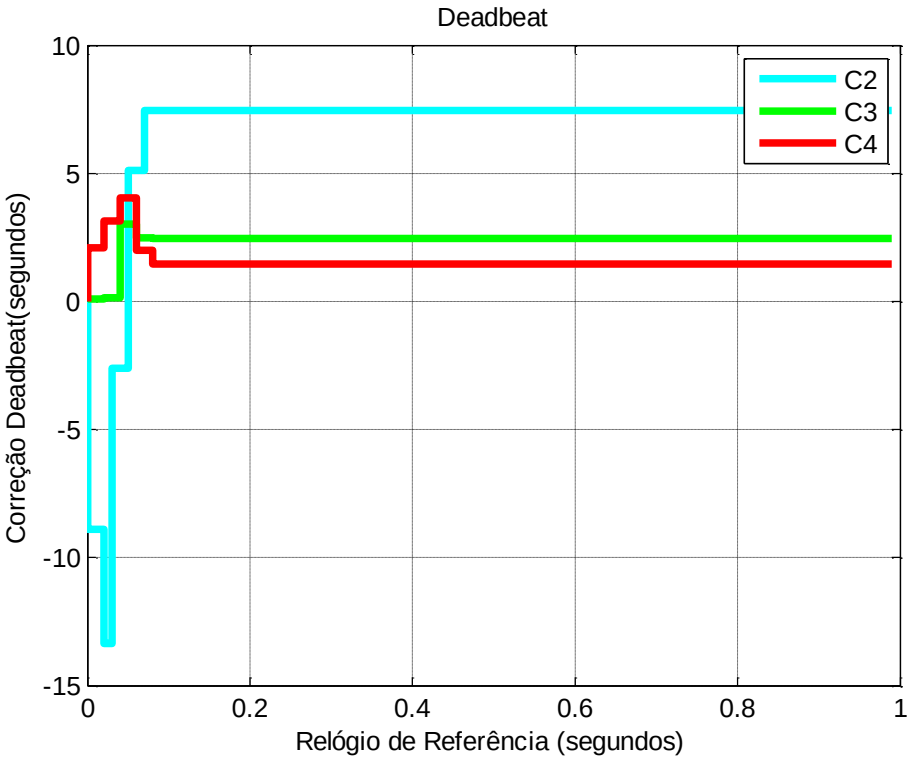


Figura 8.201 – Caso 35 - Modo STM - Algoritmo AReS Exp. - TDMA.

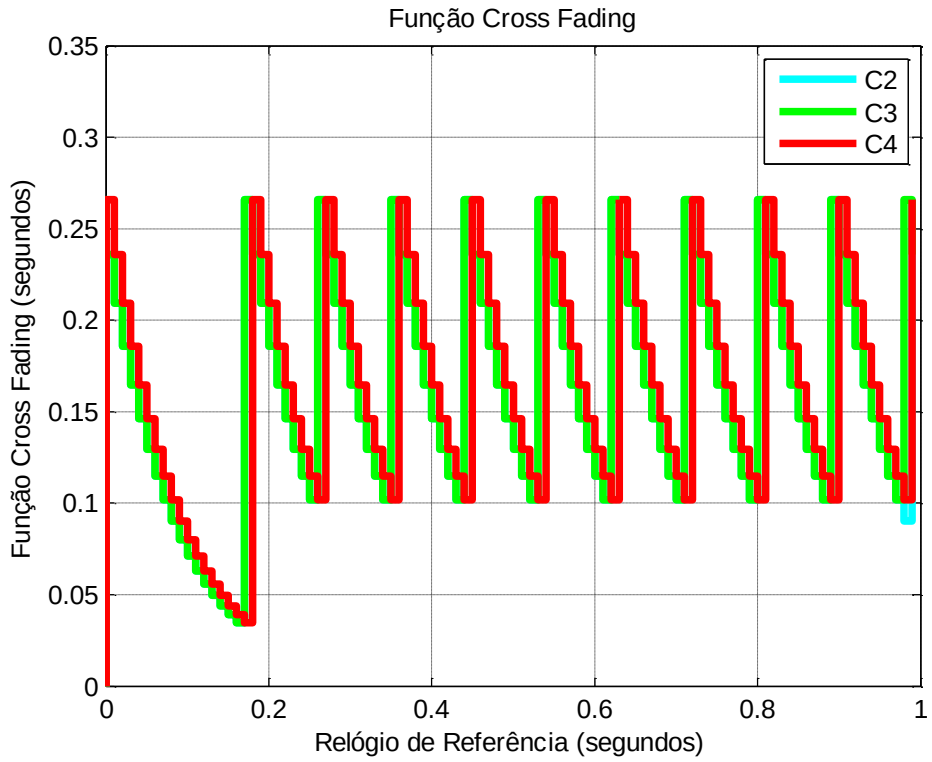


Figura 8.202 – Caso 35 - Função Cross-Fading Exponencial.

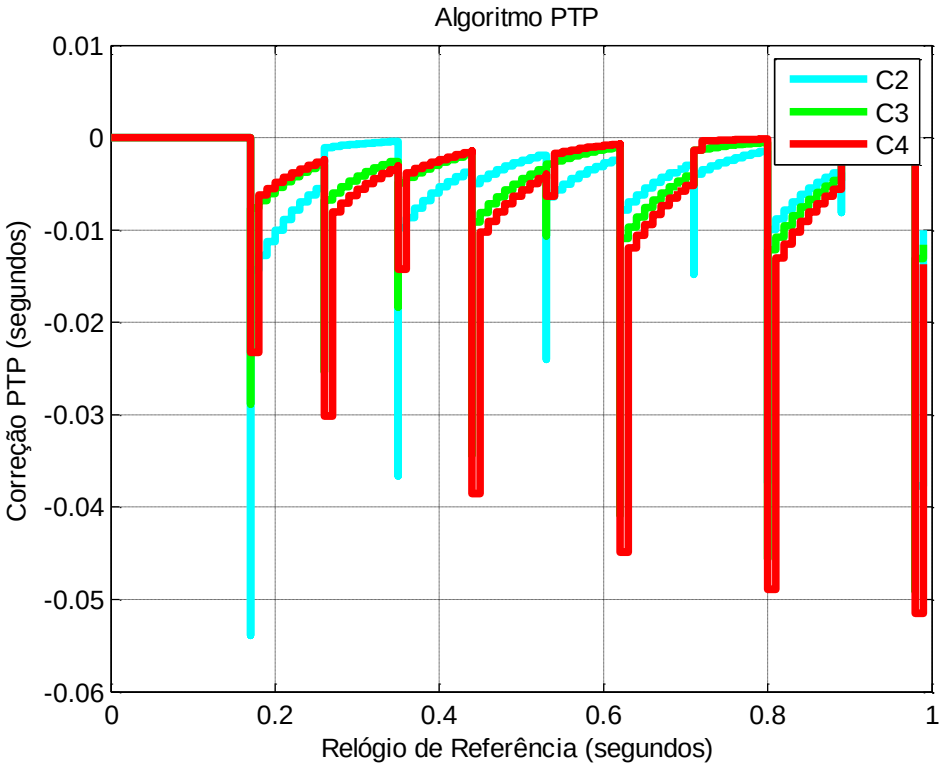


Figura 8.203 – Caso 35 - Correção modo NOMO - Algoritmo AReS Exp. TDMA.

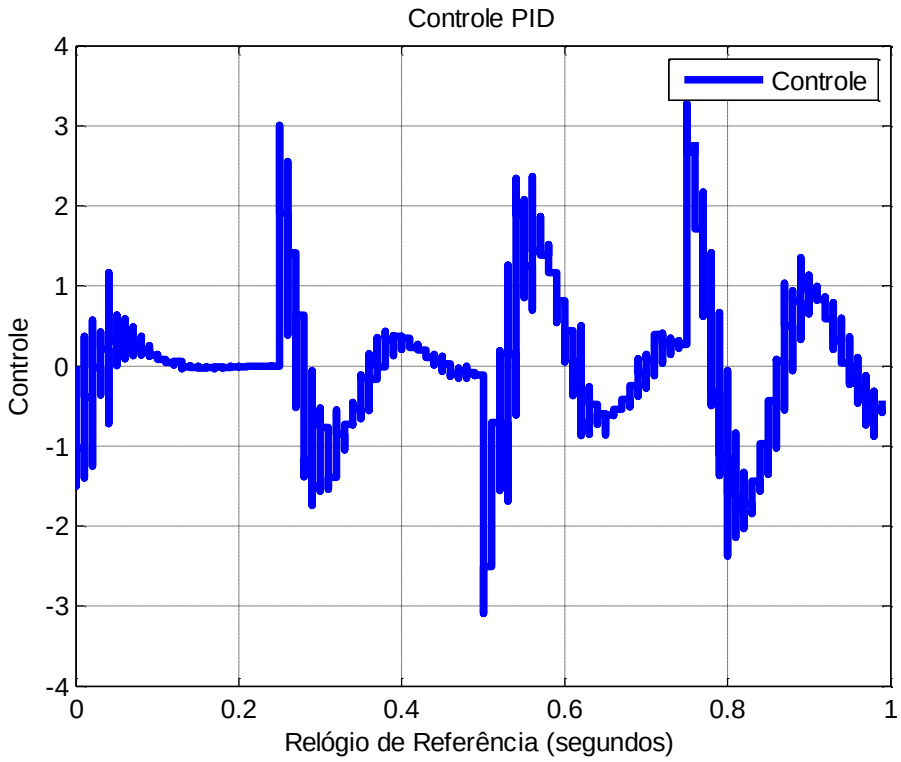


Figura 8.204 – Caso 35 - Controle PID.

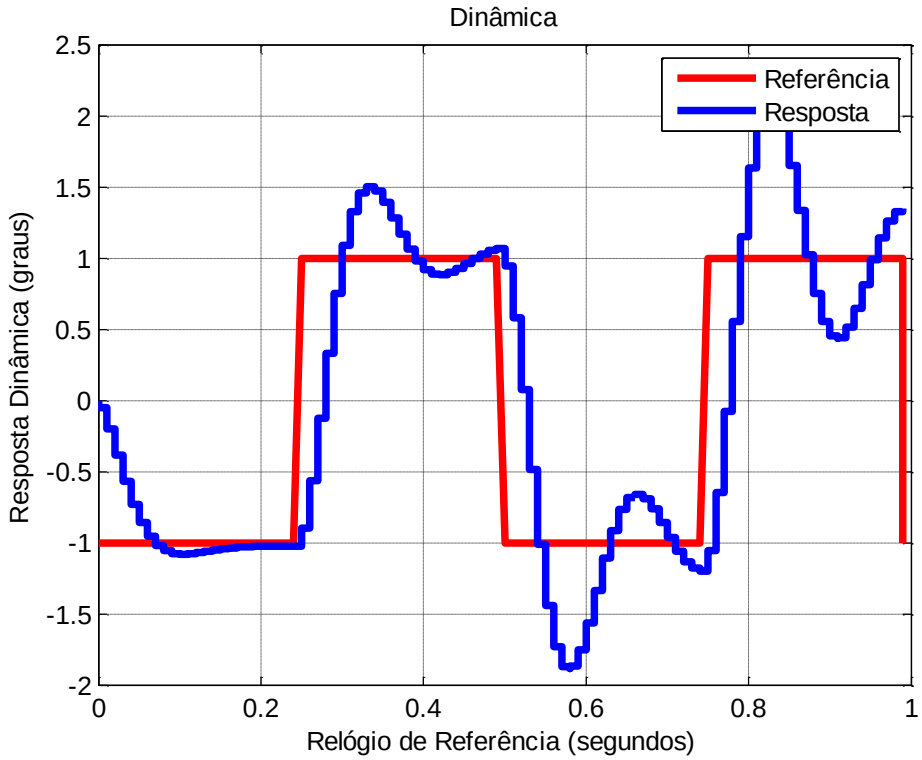


Figura 8.205 – Caso 35 - Resposta Dinâmica.

A principal diferença deste caso para o caso da seção 8.14 é que o gerenciamento de redes foi modificado. Em uma rede TDMA é possível estimar o pior caso de atraso de rede. Além disso, a rede TDMA aumenta o atraso de rede e por isso este caso teve um sobre sinal um pouco maior na resposta dinâmica e também uma exatidão pior em relação ao caso da seção 8.14.

8.38. Caso 36 - Algoritmo PReS Degrau - TDMA

Os parâmetros e a sincronização de relógios é o mesmo da seção 8.22. A diferença é que este caso faz toda a simulação sobre uma rede TDMA. Assim, tem-se o seguinte resultado.

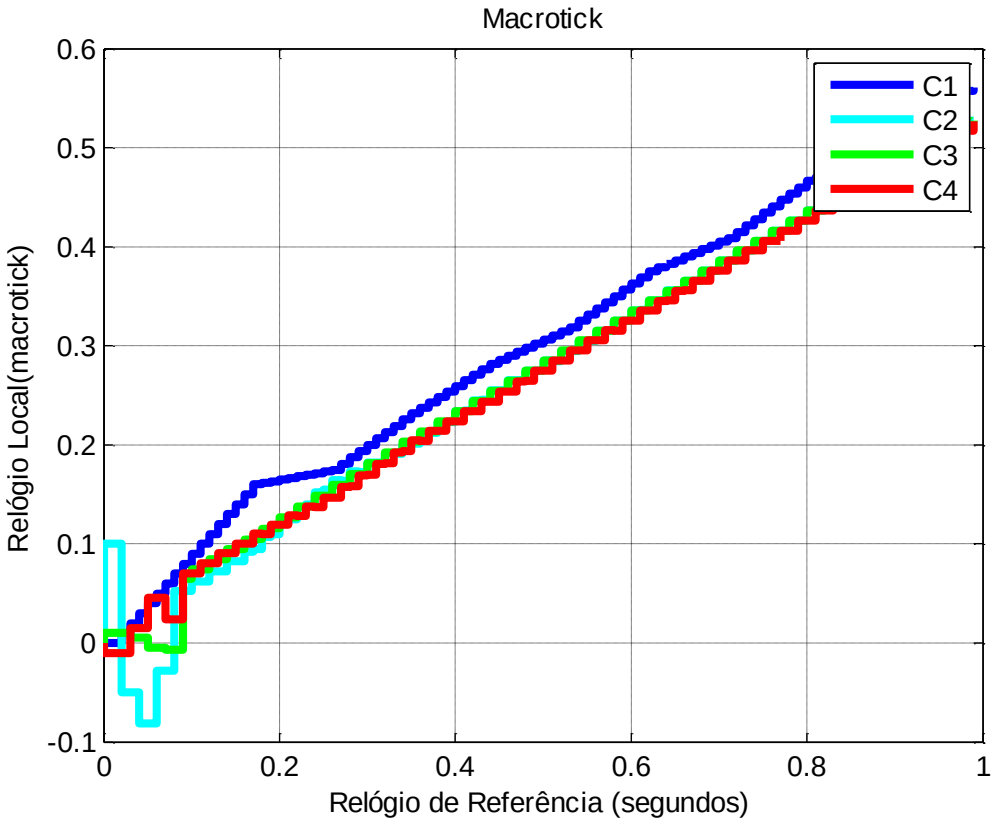


Figura 8.206 – Caso 36 - Macrotick.

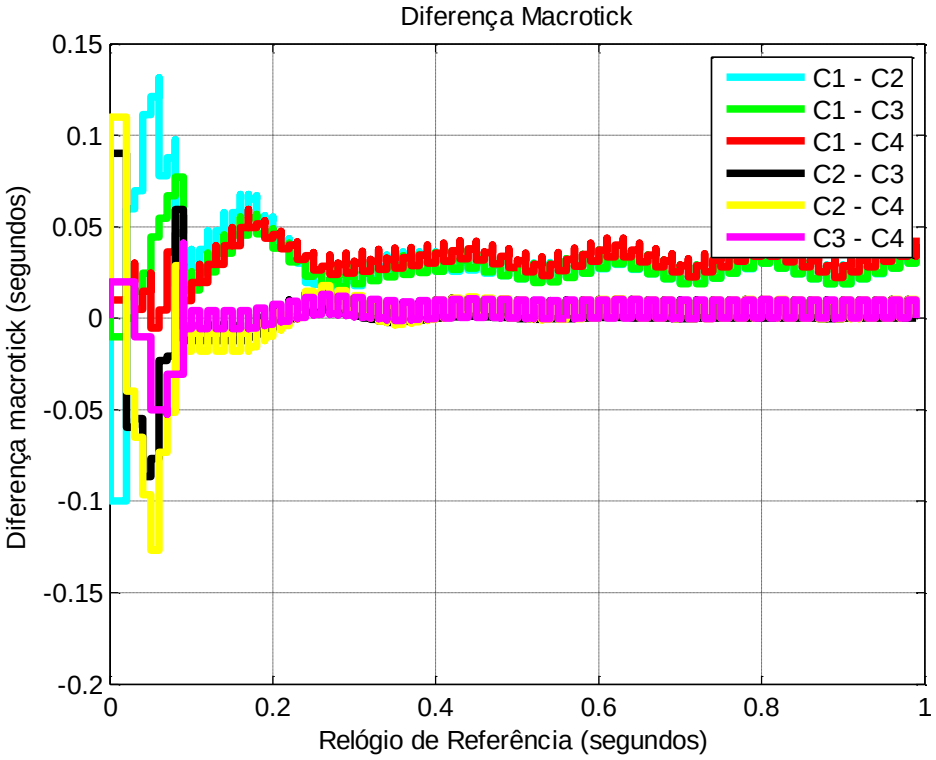


Figura 8.207 – Caso 36 - Diferença entre Macroticks.

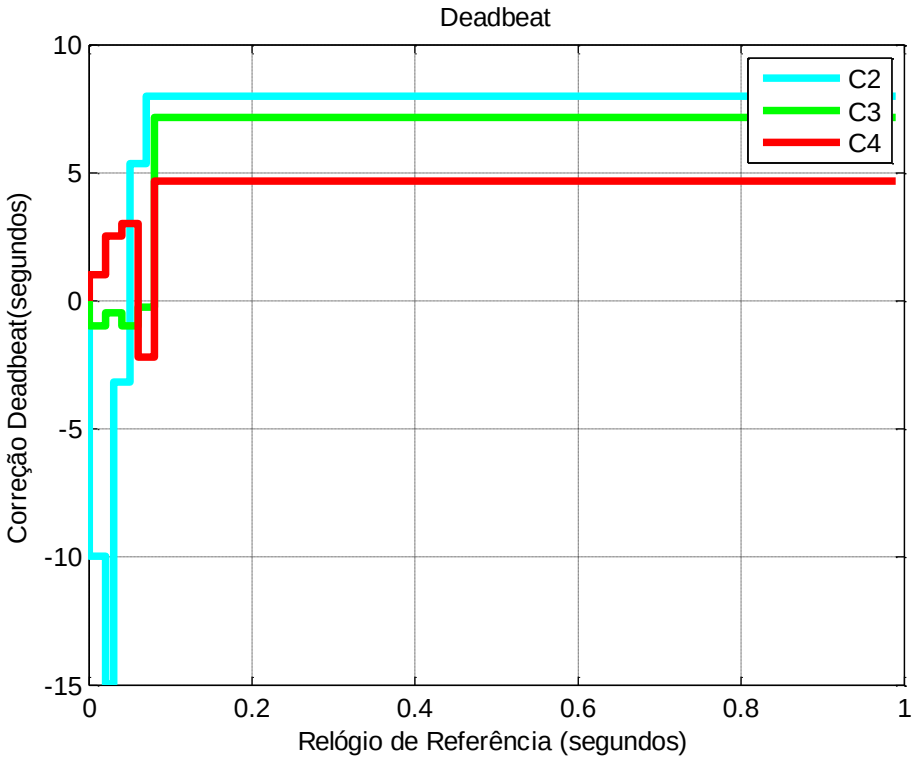


Figura 8.208 – Caso 36 - Modo STM - Algoritmo AReS Degrau - TDMA.

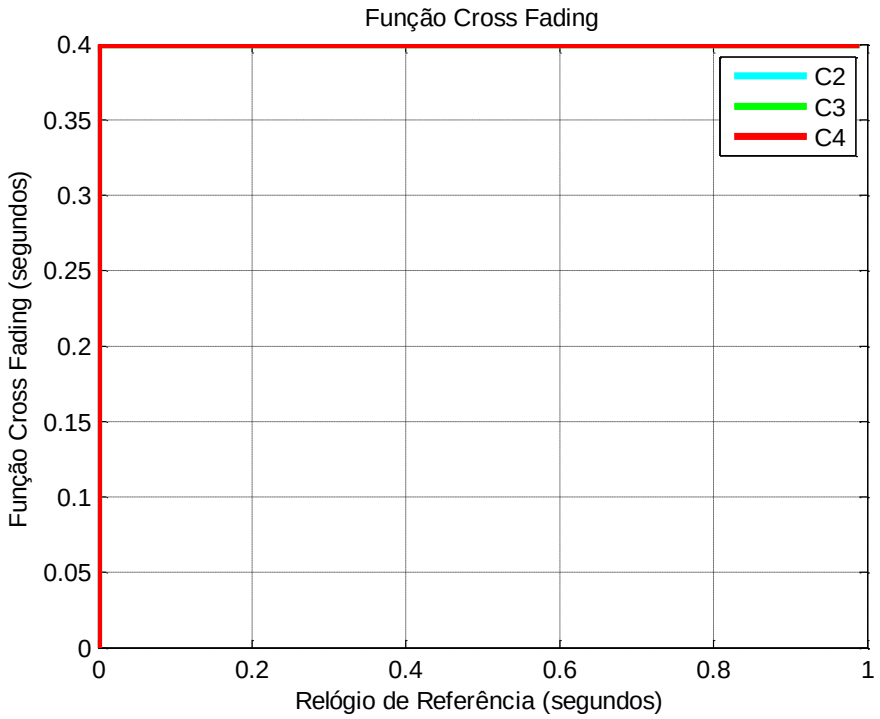


Figura 8.209 – Caso 36 - Função Cross-Fading Degrau.

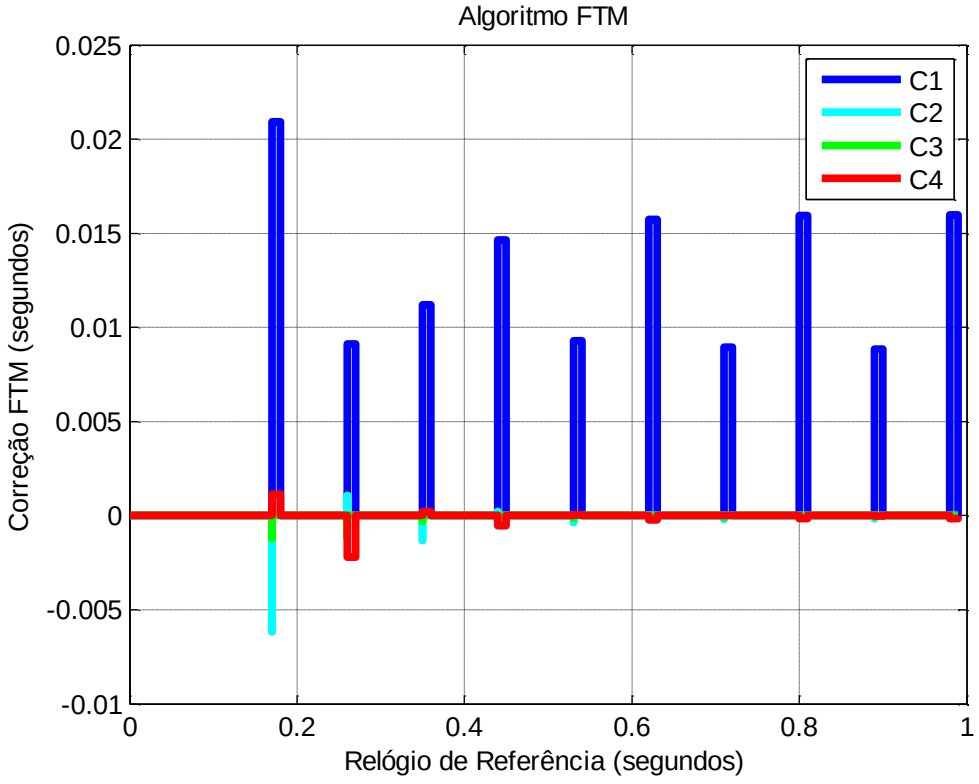


Figura 8.210 – Caso 36 - Correção modo NOMO - Algoritmo PReS Degrau TDMA.

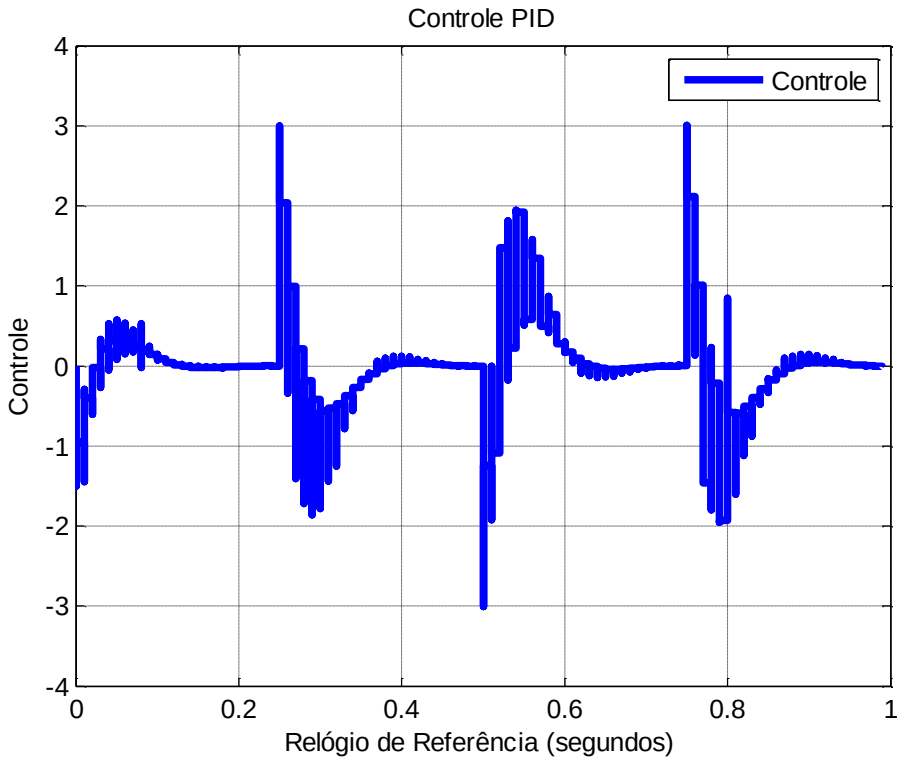


Figura 8.211 – Caso 36 - Controle PID.

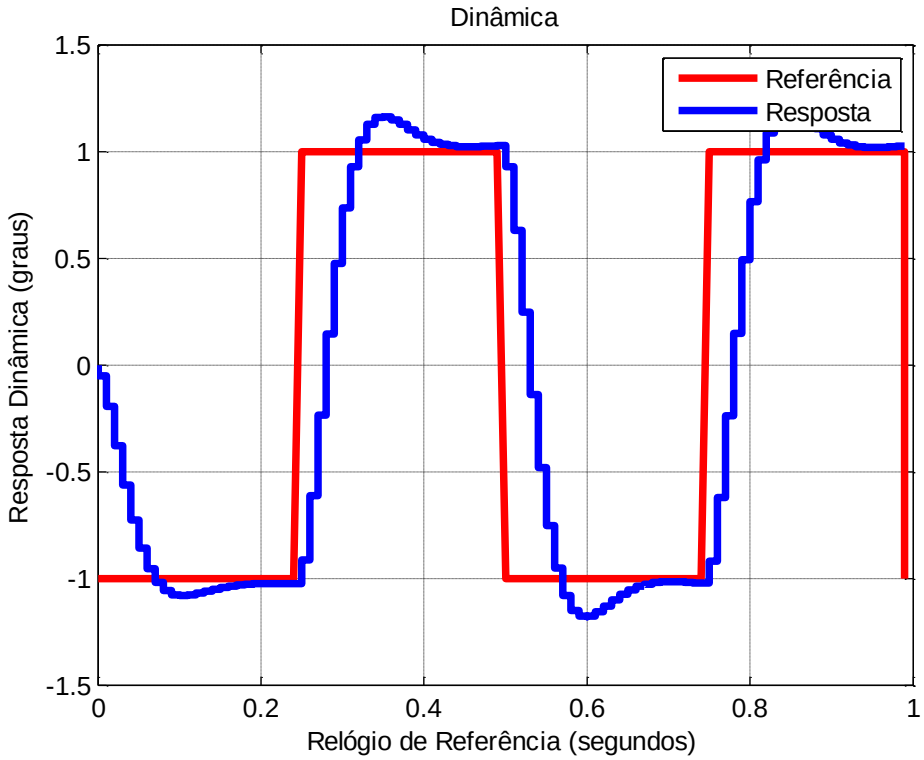


Figura 8.212 – Caso 36 - Resposta Dinâmica.

A principal diferença deste caso para o caso da seção 8.22 é que o gerenciamento de redes foi modificado. Em uma rede TDMA é possível estimar o pior caso de atraso de rede. Além disso, a rede TDMA aumenta o atraso de rede. Este caso teve um sobre sinal um pouco menor na resposta dinâmica e também uma precisão um pouco pior em relação ao caso da seção 8.22.

8.39. Caso 37 - Algoritmo PReS Rampa - TDMA

Os parâmetros e a sincronização de relógios é o mesmo da seção 8.23. A diferença é que este caso faz toda a simulação sobre uma rede TDMA. Assim, tem-se o seguinte resultado.

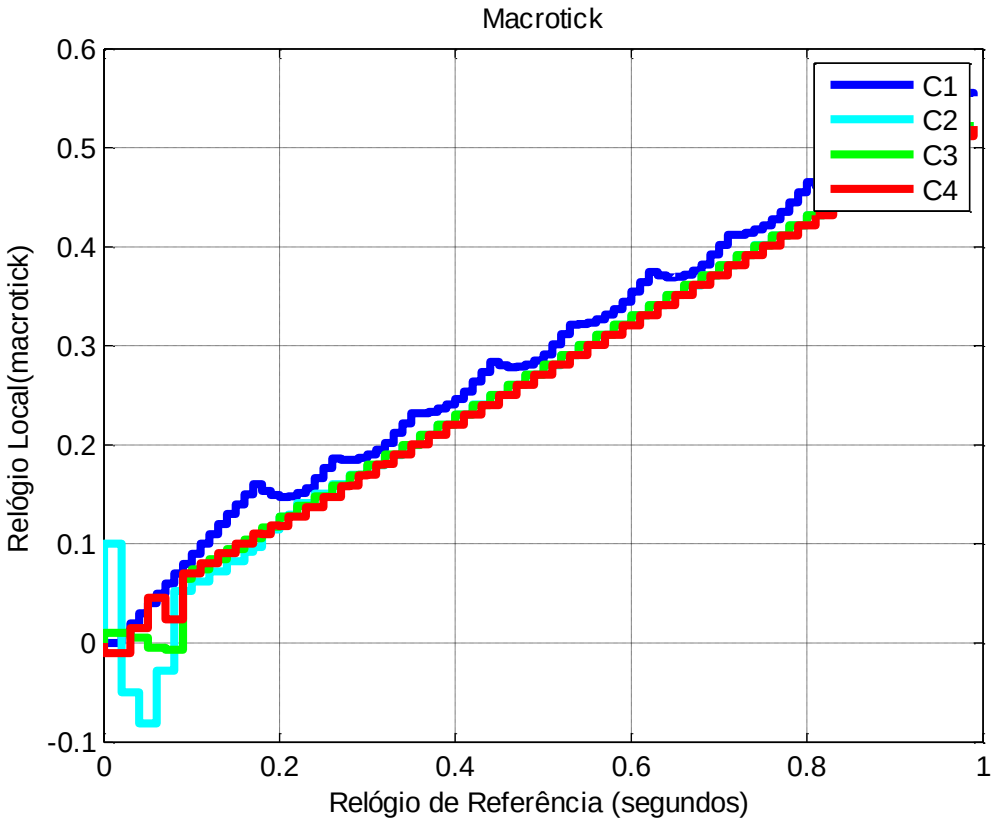


Figura 8.213 – Caso 37 - Macrotick.

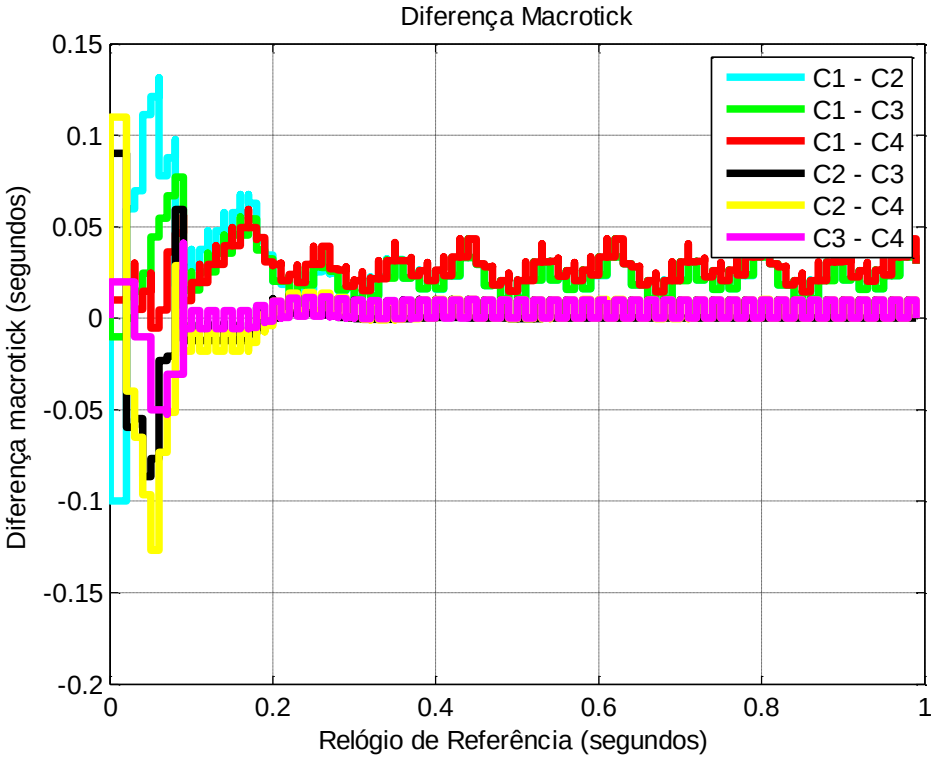


Figura 8.214 – Caso 37 - Diferença entre Macroticks.

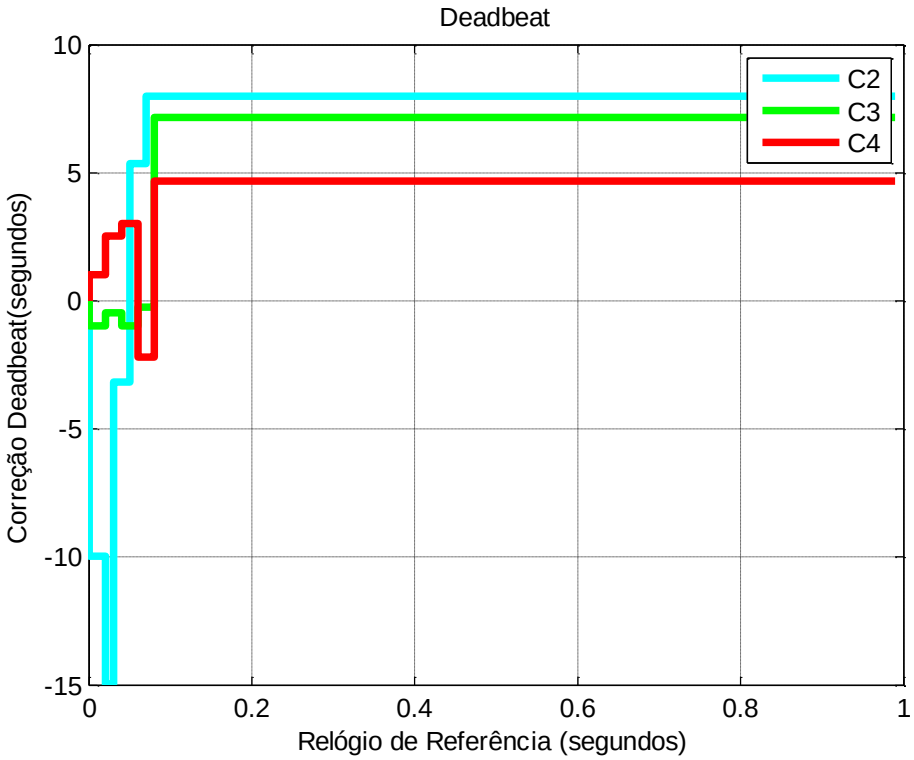


Figura 8.215 – Caso 37 - Modo STM - Algoritmo PReS Rampa - TDMA.

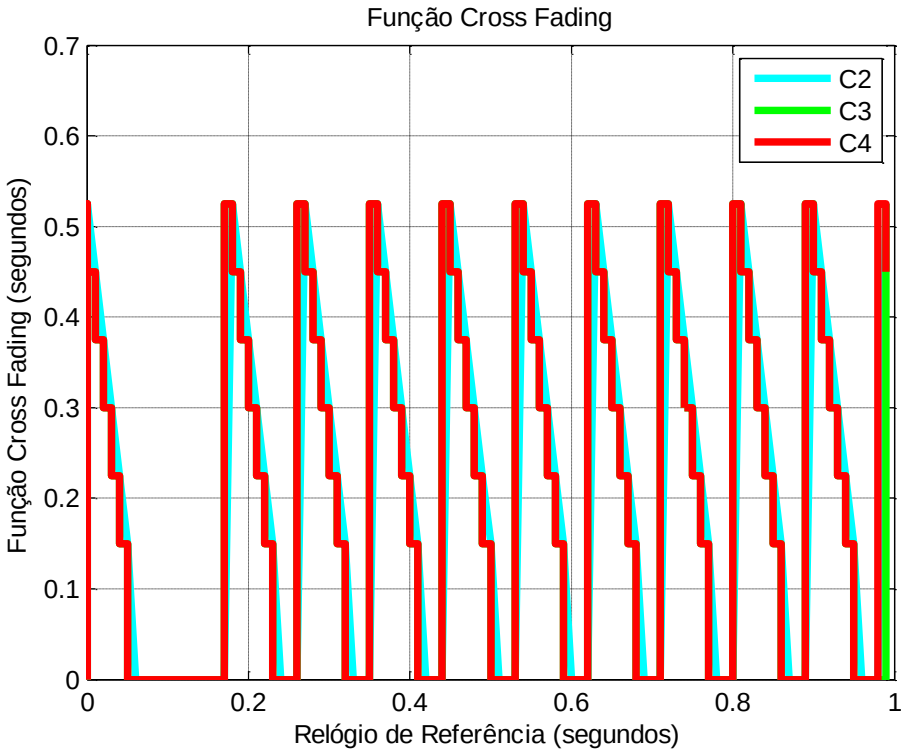


Figura 8.216 – Caso 37 - Função Cross-Fading Rampa.

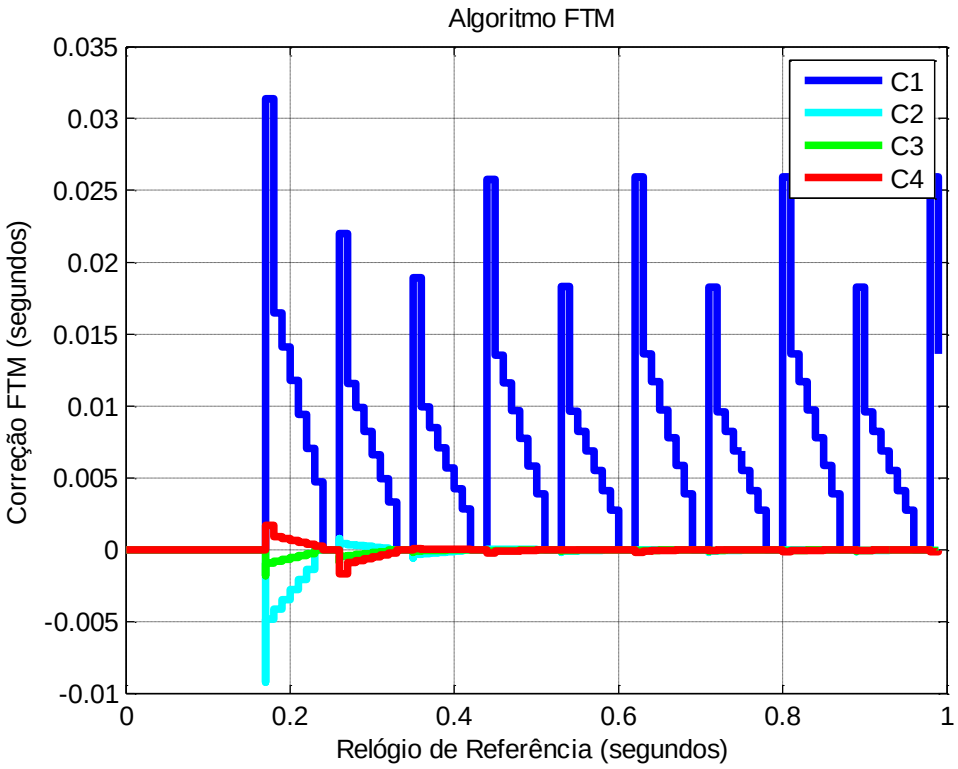


Figura 8.217 – Caso 37 - Correção modo NOMO - Algoritmo PReS Rampa TDMA.

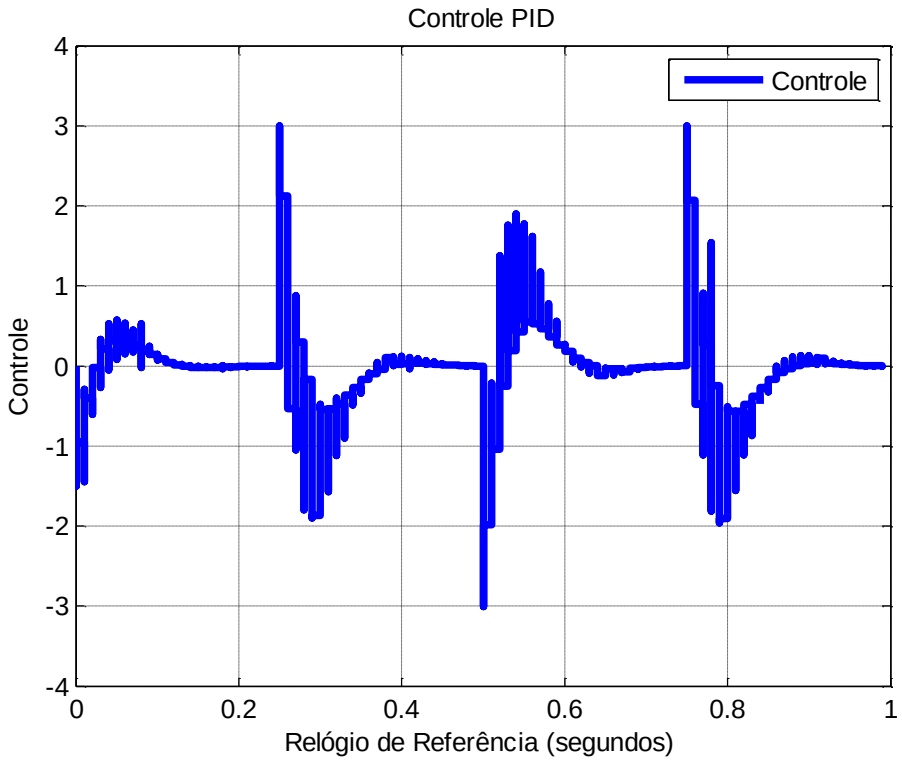


Figura 8.218 – Caso 37 - Controle PID.

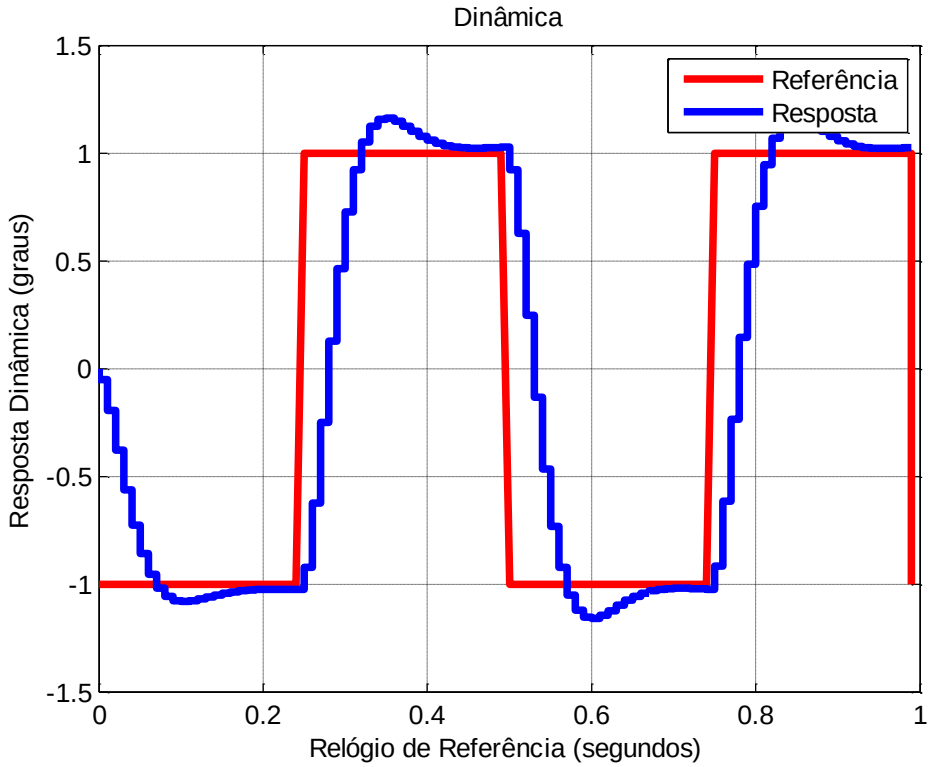


Figura 8.219 – Caso 37 - Resposta Dinâmica.

A principal diferença deste caso para o caso da seção 8.23 é que o gerenciamento de redes foi modificado. Em uma rede TDMA é possível estimar o pior caso de atraso de rede. Além disso, a rede TDMA aumenta o atraso de rede. Este caso teve um sobre sinal um pouco menor na resposta dinâmica, uma melhora da precisão em relação ao caso da seção 8.23. Devido ao atraso de rede ser sempre constante era esperado que os casos TDMA obtivessem um melhor resultado em relação à rede CSMA/CD.

8.40. Caso 38 - Algoritmo PReS Exponencial - TDMA

Os parâmetros e a sincronização de relógios é o mesmo da seção 8.24. A diferença é que este caso faz toda a simulação sobre uma rede TDMA. Assim, tem-se o seguinte resultado.

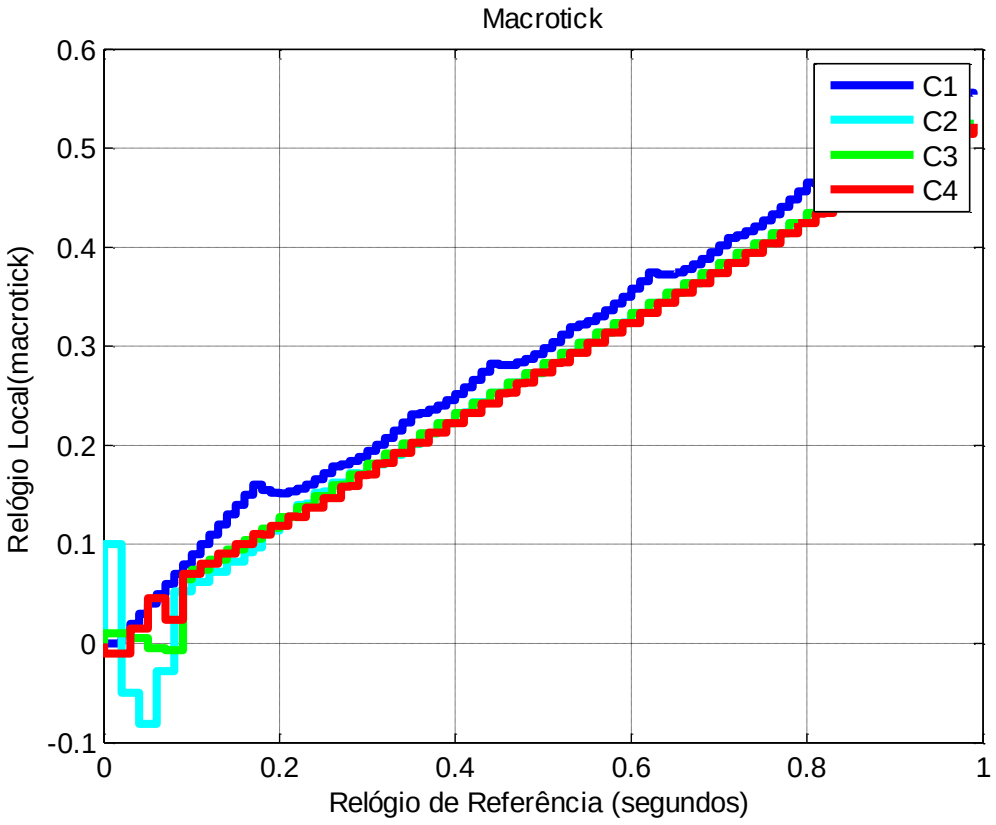


Figura 8.220 – Caso 38 - Macrotick.

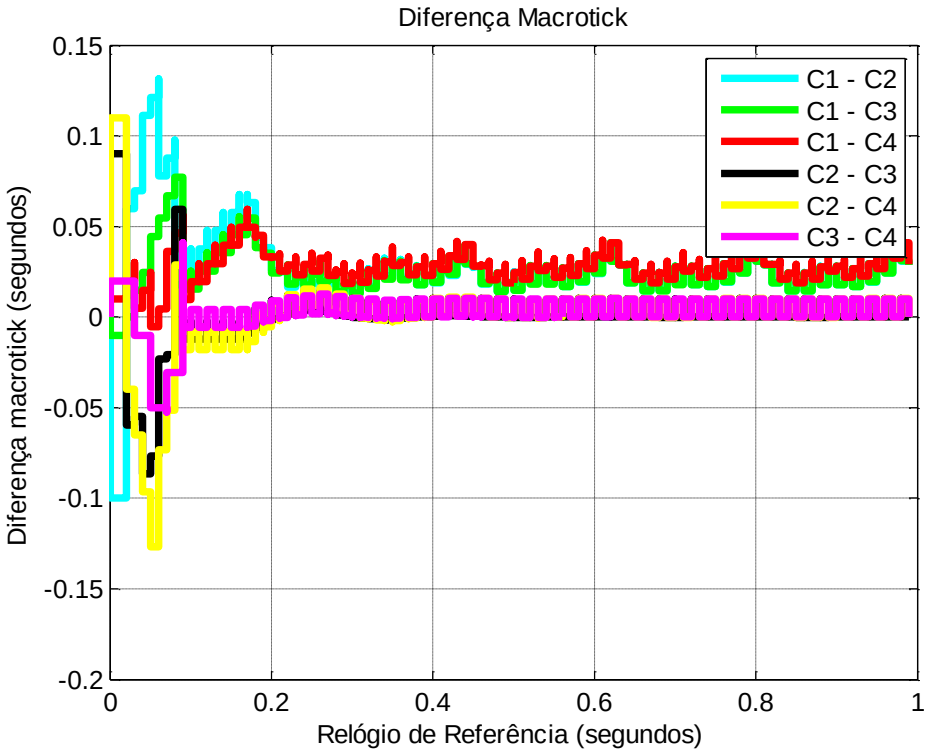


Figura 8.221 – Caso 38 - Diferença entre Macroticks.

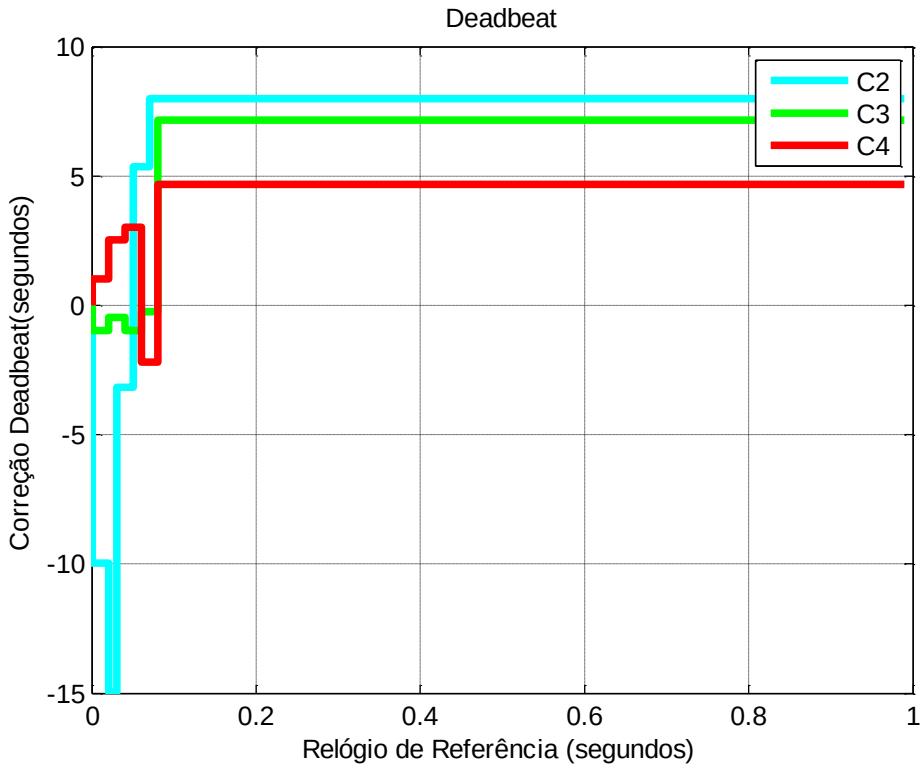


Figura 8.222 – Caso 38 - Modo STM - Algoritmo PReS Exp. - TDMA.

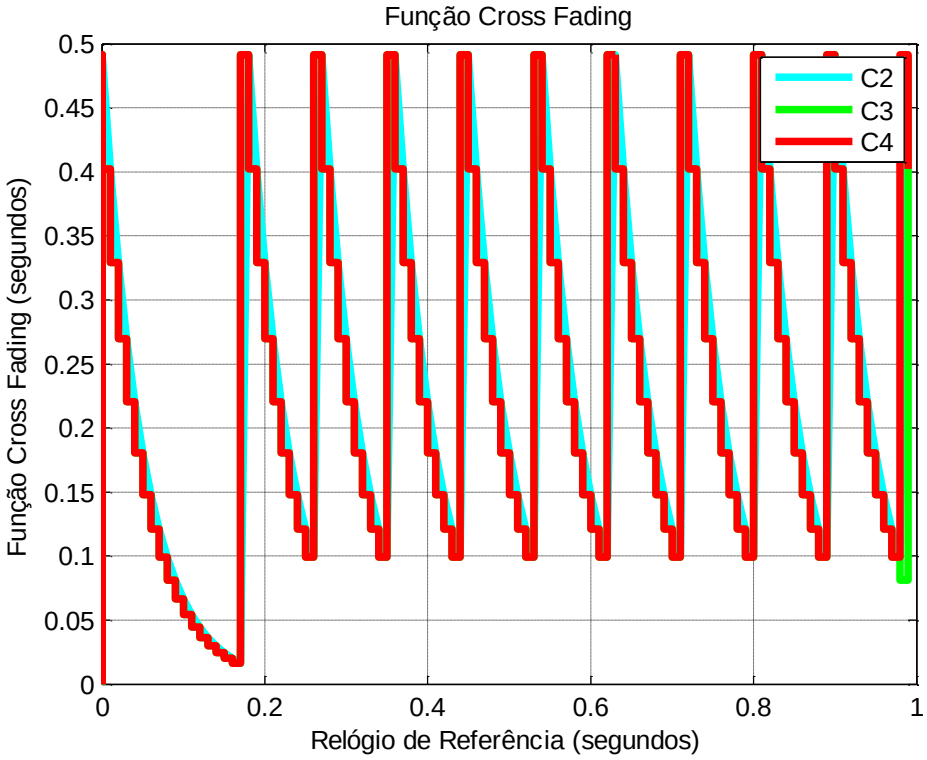


Figura 8.223 – Caso 38 - Função Cross-Fading Exponencial.

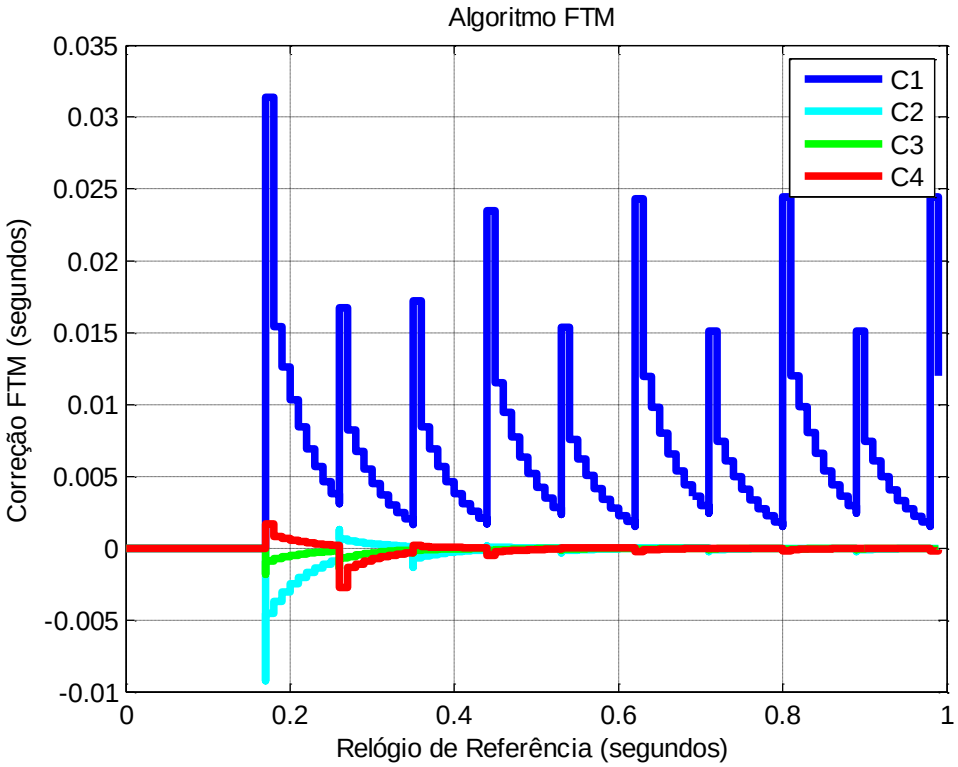


Figura 8.224 – Caso 38 - Correção modo NOMO - Algoritmo PReS Exp. TDMA.

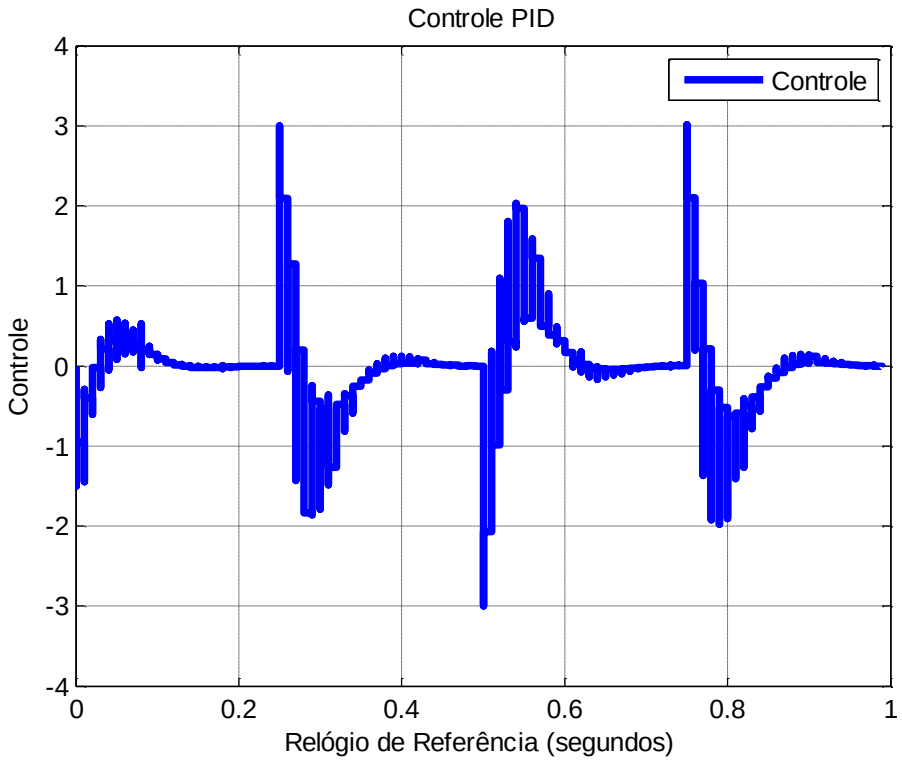


Figura 8.225 – Caso 38 - Controle PID.

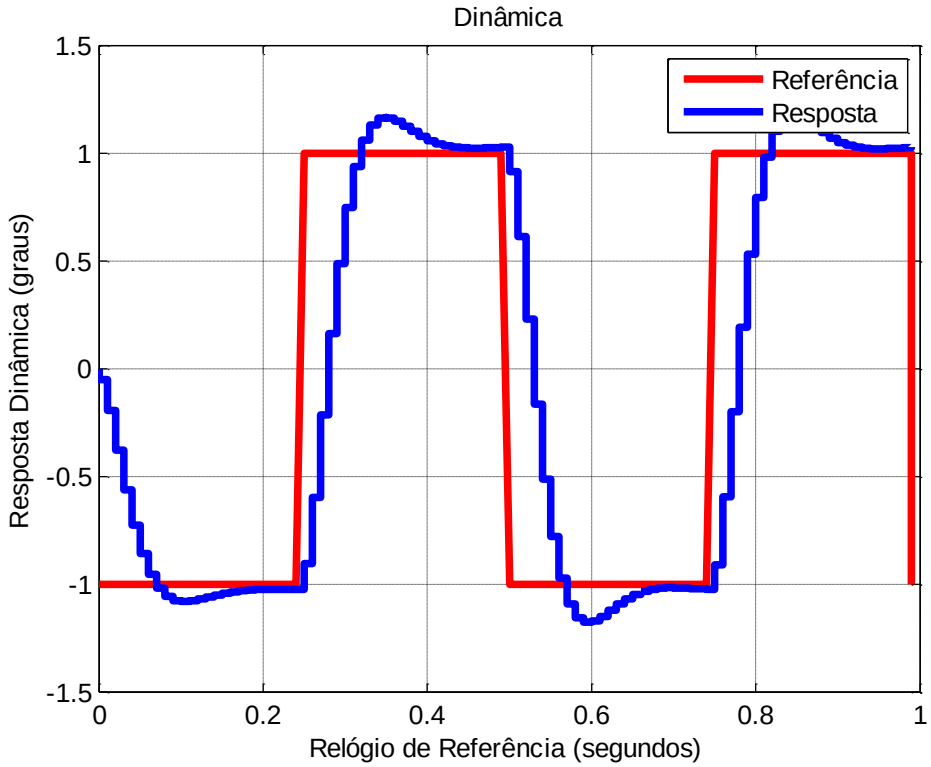


Figura 8.226 – Caso 38 - Resposta Dinâmica.

A principal diferença deste caso para o caso da seção 8.24 é que o gerenciamento de redes foi modificado. Em uma rede TDMA é possível estimar o pior caso de atraso de rede. Além disso, a rede TDMA aumenta o atraso de rede. Este caso teve um sobre sinal um pouco menor na resposta dinâmica, uma melhora da precisão em relação ao caso da seção 8.24. Devido ao atraso de rede ser sempre constante era esperado que os casos TDMA obtivessem um melhor resultado em relação à rede CSMA/CD.

8.41. Caso 39 - Algoritmo PAReS Degrau - TDMA

Os parâmetros e a sincronização de relógios é o mesmo da seção 8.32. A diferença é que este caso faz toda a simulação sobre uma rede TDMA. Assim, tem-se o seguinte resultado.

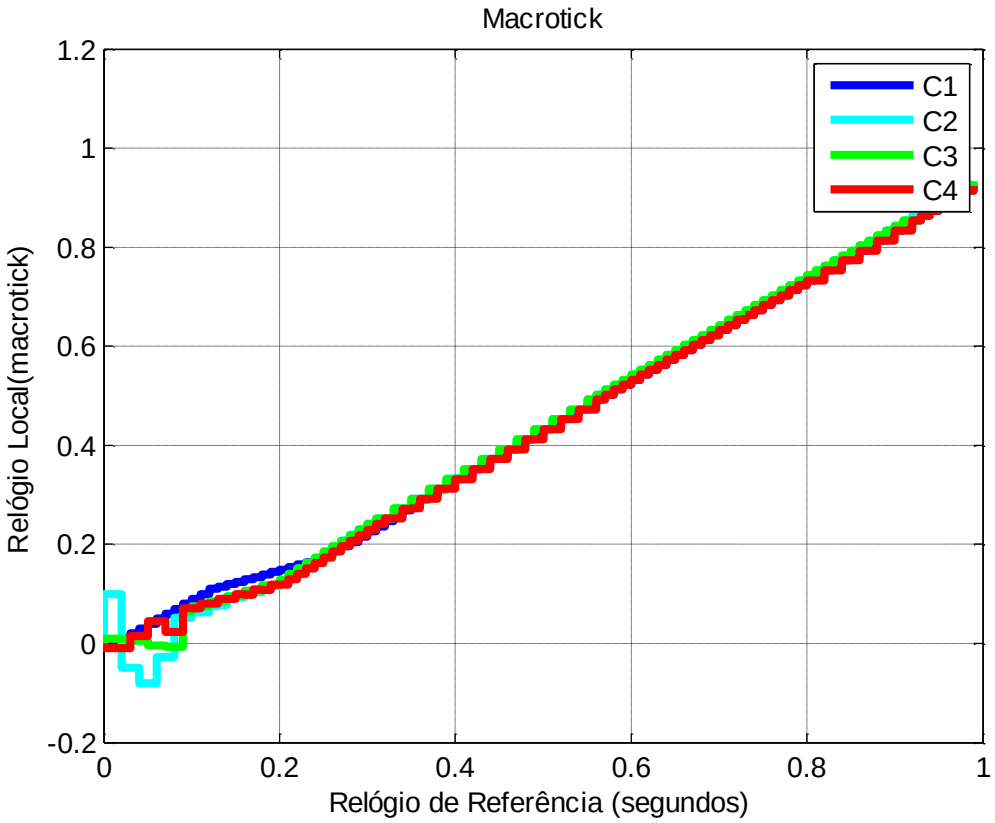


Figura 8.227 – Caso 39 - Macrotick.

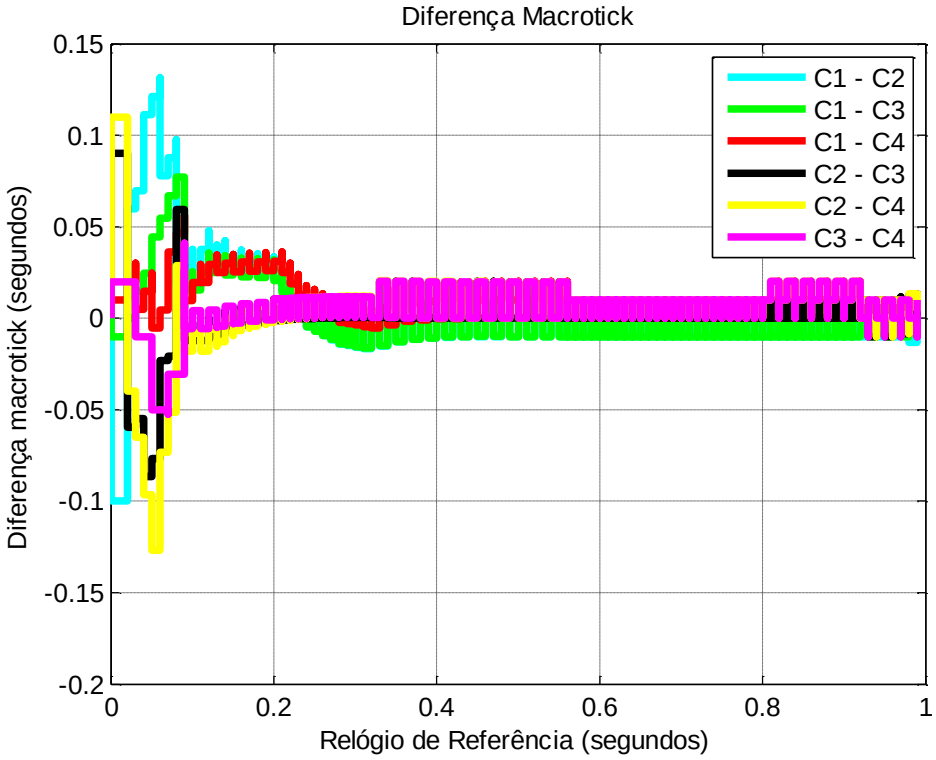


Figura 8.228 – Caso 39 - Diferença entre Macroticks.

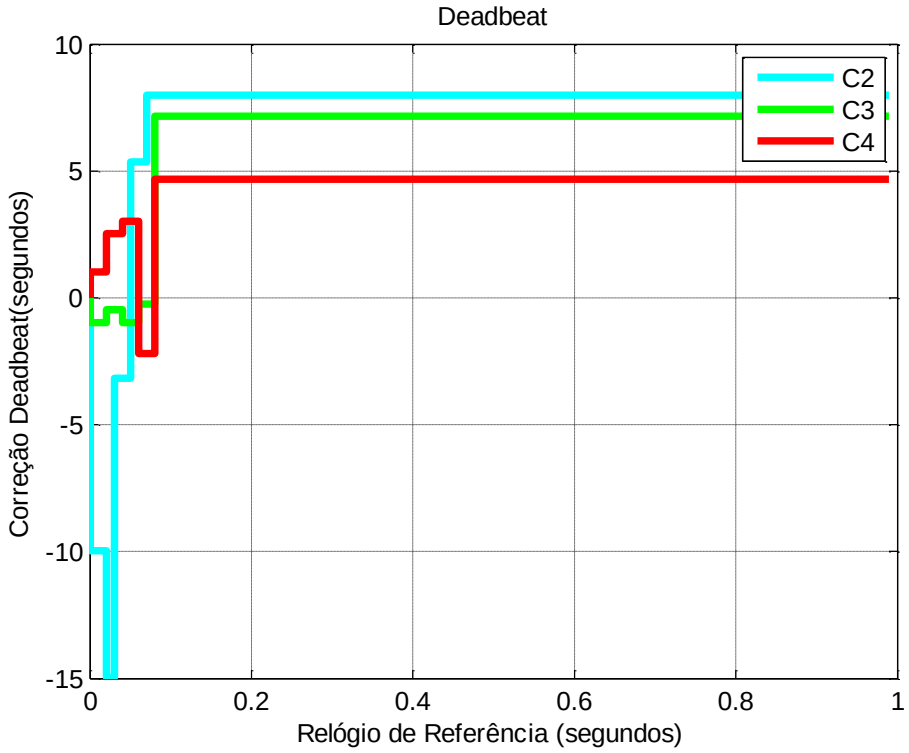


Figura 8.229 – Caso 39 - Modo STM - Algoritmo PAReS Degrau. - TDMA.

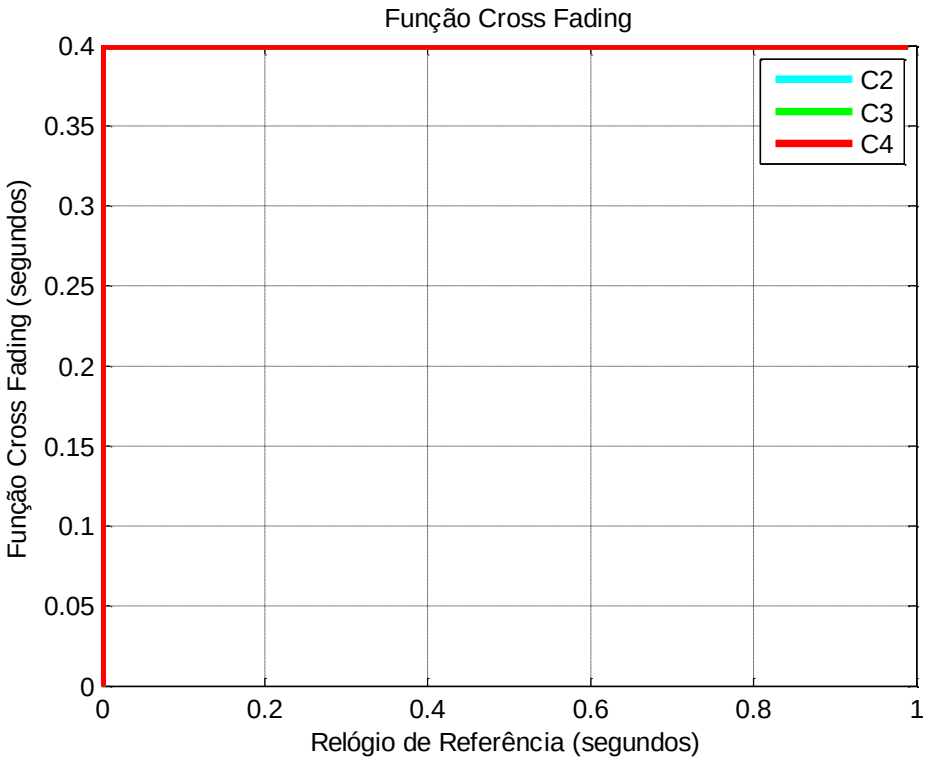


Figura 8.230 – Caso 39 - Função Cross-Fading Degrau.

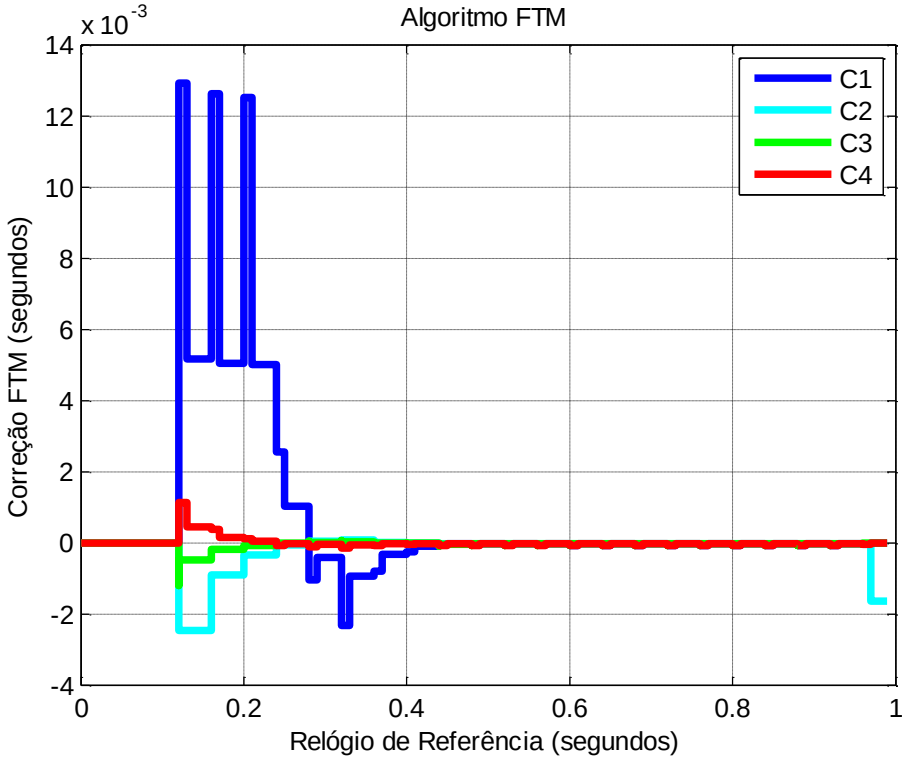


Figura 8.231 – Caso 39 - Correção modo NOMO - Algoritmo PAReS Deg. TDMA.

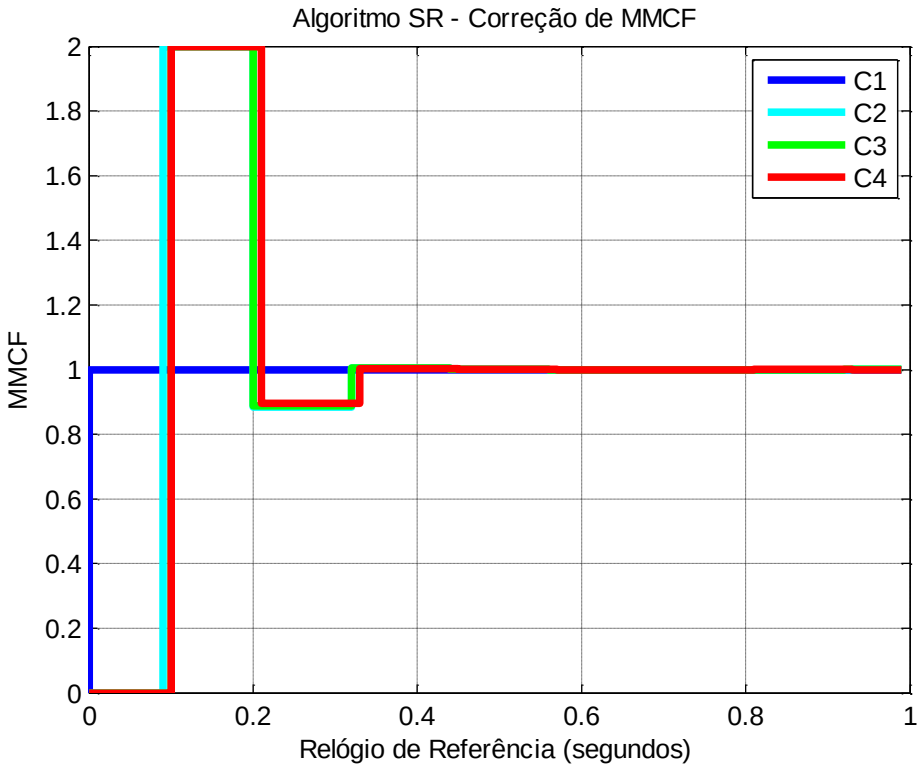


Figura 8.232 – Caso 39 - Correção modo NOMD - Algoritmo PAReS Deg. TDMA.

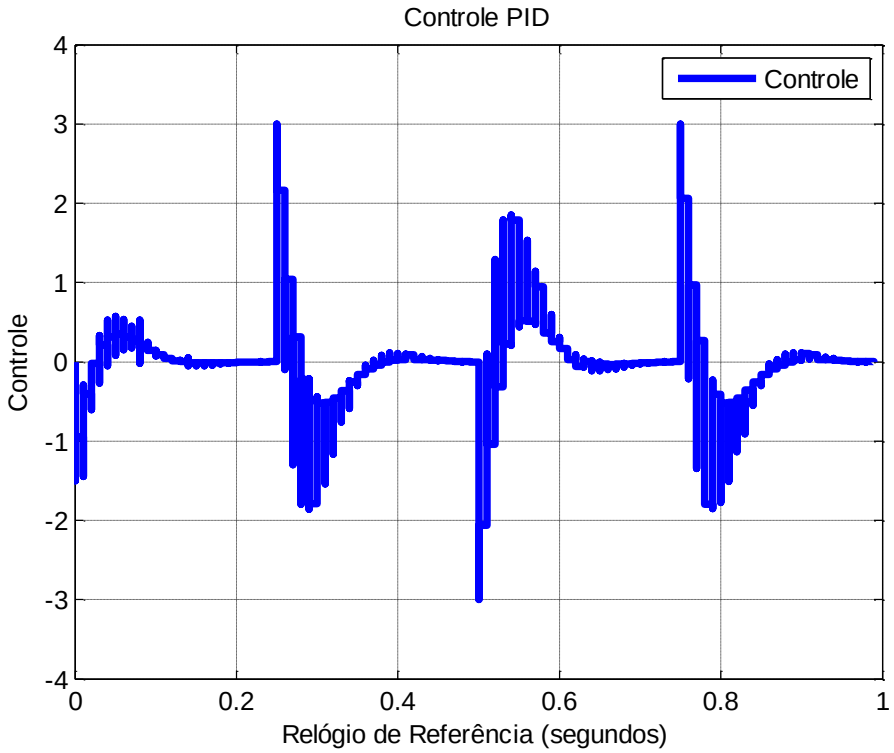


Figura 8.233 – Caso 39 - Controle PID.

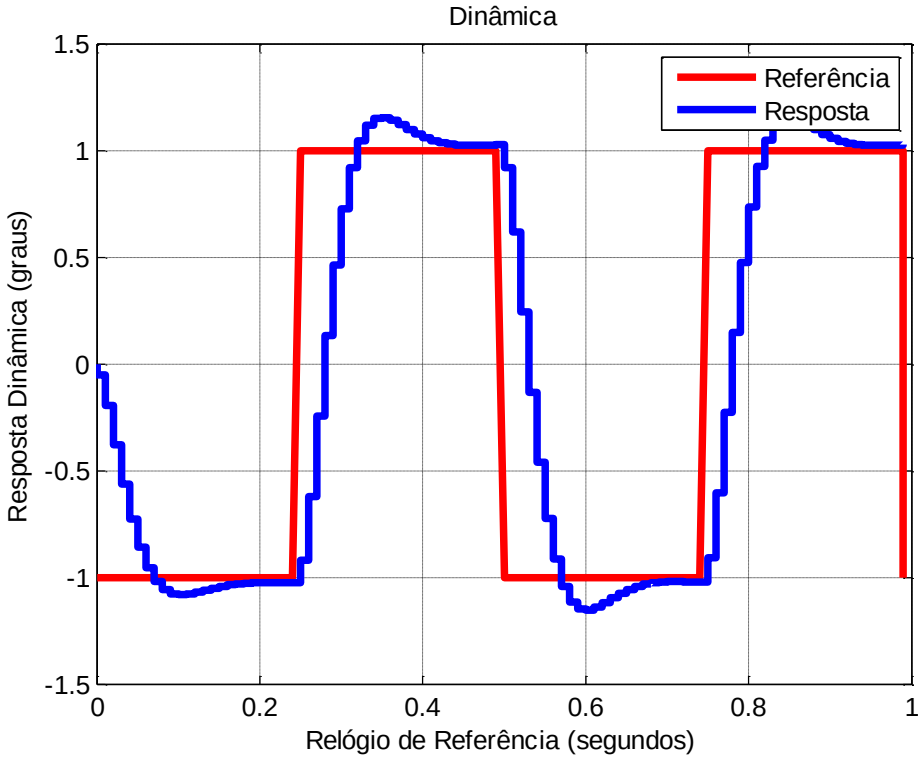


Figura 8.234 – Caso 39 - Resposta Dinâmica.

A principal diferença deste caso para o caso da seção 8.32 é que o gerenciamento de redes foi modificado. Em uma rede TDMA é possível estimar o pior caso de atraso de rede. Além disso, a rede TDMA aumenta o atraso de rede. Este caso teve um sobre sinal um pouco menor na resposta dinâmica, uma melhora da precisão e exatidão em relação ao caso da seção 8.32. Devido ao atraso de rede ser sempre constante era esperado que os casos TDMA obtivessem um melhor resultado em relação à rede CSMA/CD.

8.42. Caso 40 - Algoritmo PAReS Rampa - TDMA

Os parâmetros e a sincronização de relógios é o mesmo da seção 8.33. A diferença é que este caso faz toda a simulação sobre uma rede TDMA. Assim, tem-se o seguinte resultado.

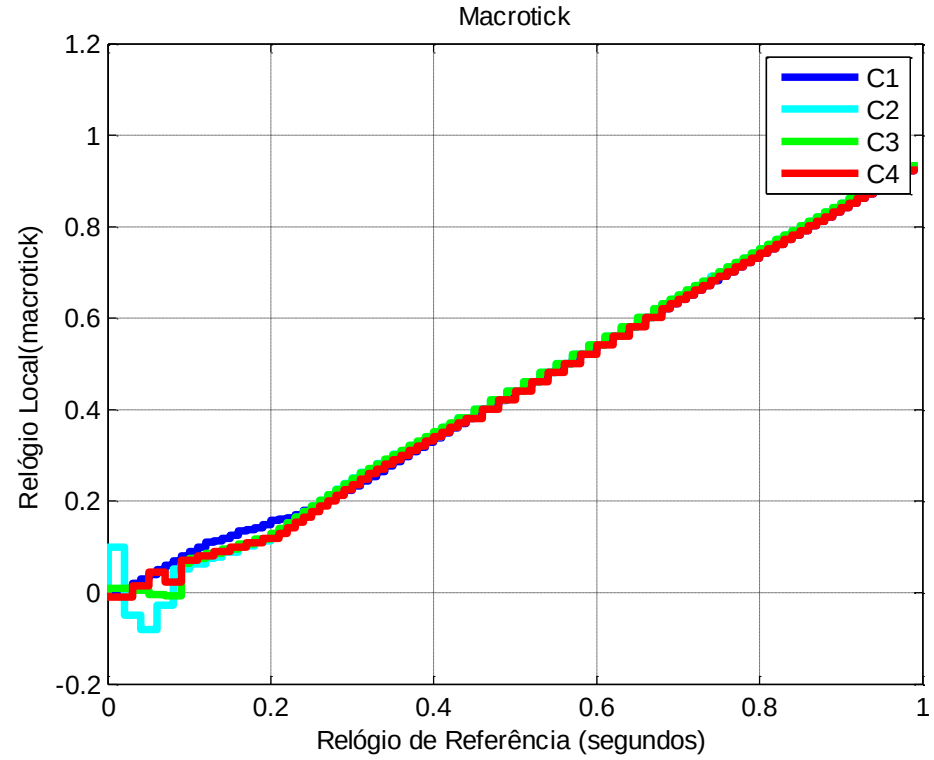


Figura 8.235 – Caso 40 - Macrotick.

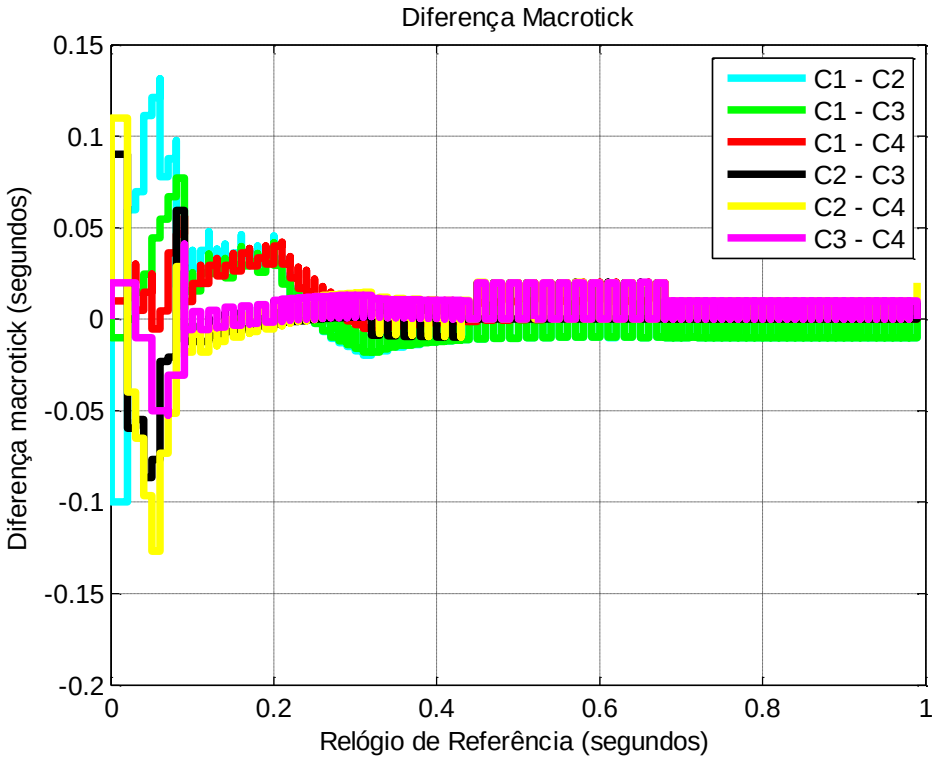


Figura 8.236 – Caso 40 - Diferença entre Macroticks.

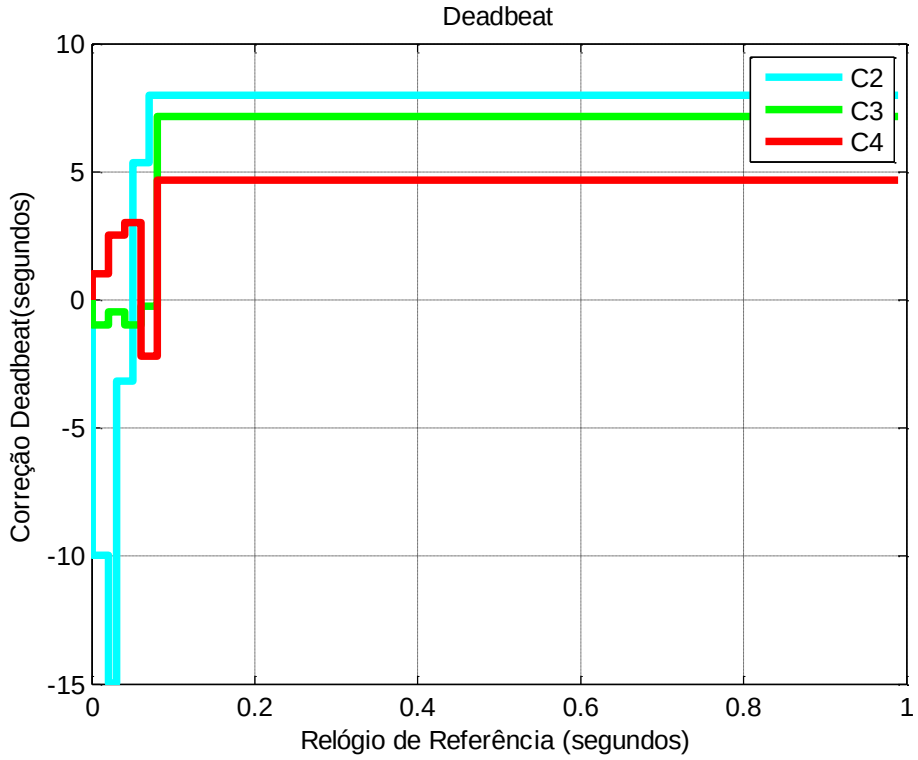


Figura 8.237 – Caso 40 - Modo STM - Algoritmo PAReS Rampa. - TDMA.

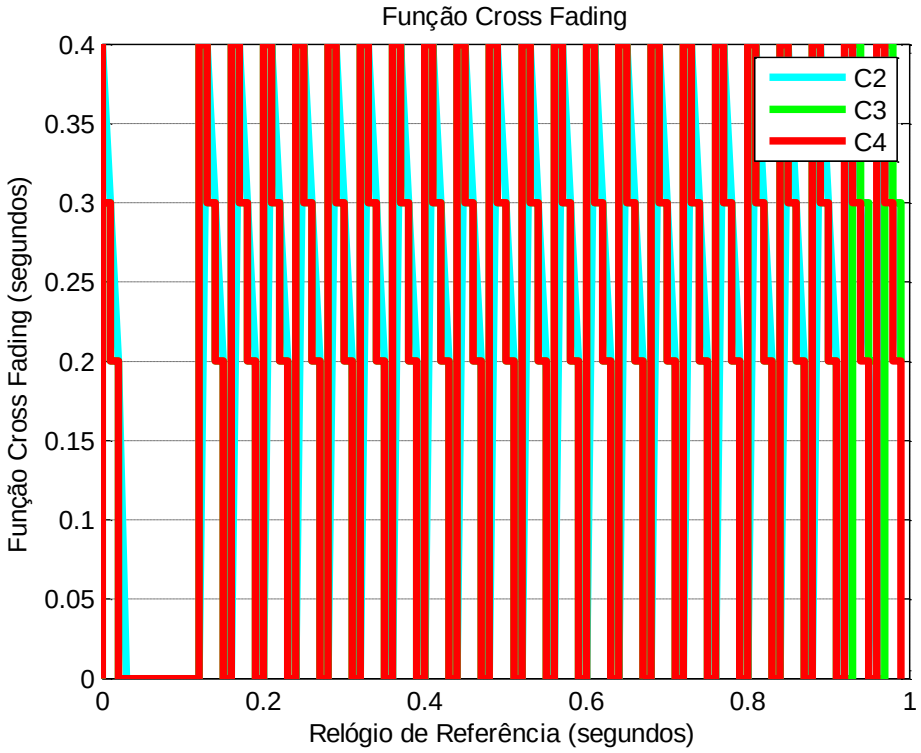


Figura 8.238 – Caso 40 - Função Cross-Fading Rampa.

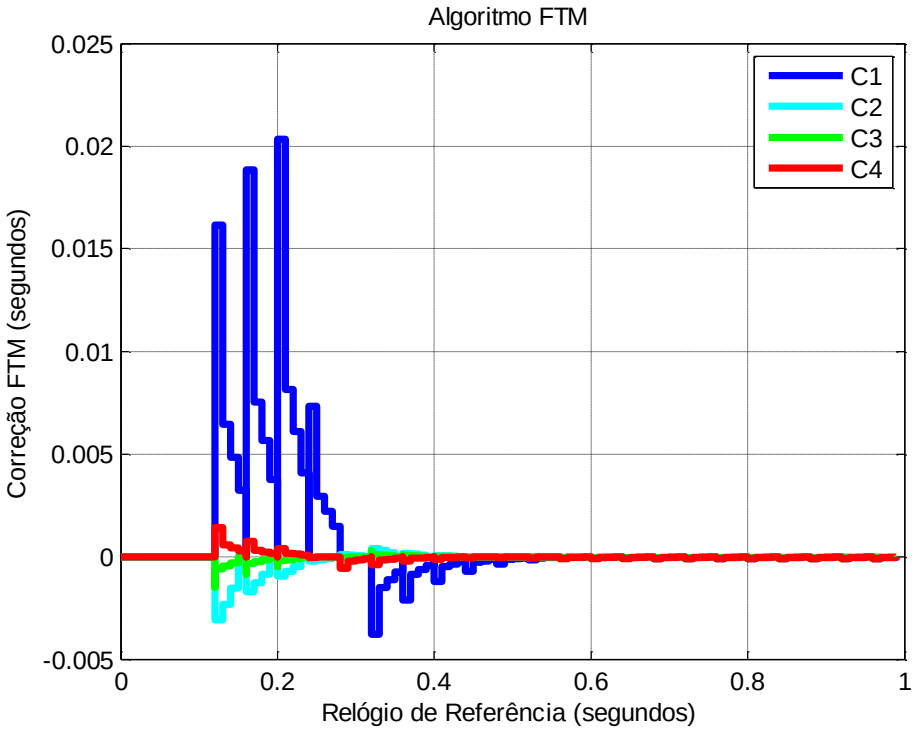


Figura 8.239 – Caso 40 - Correção modo NOMO - Algoritmo PAReS Rampa TDMA.

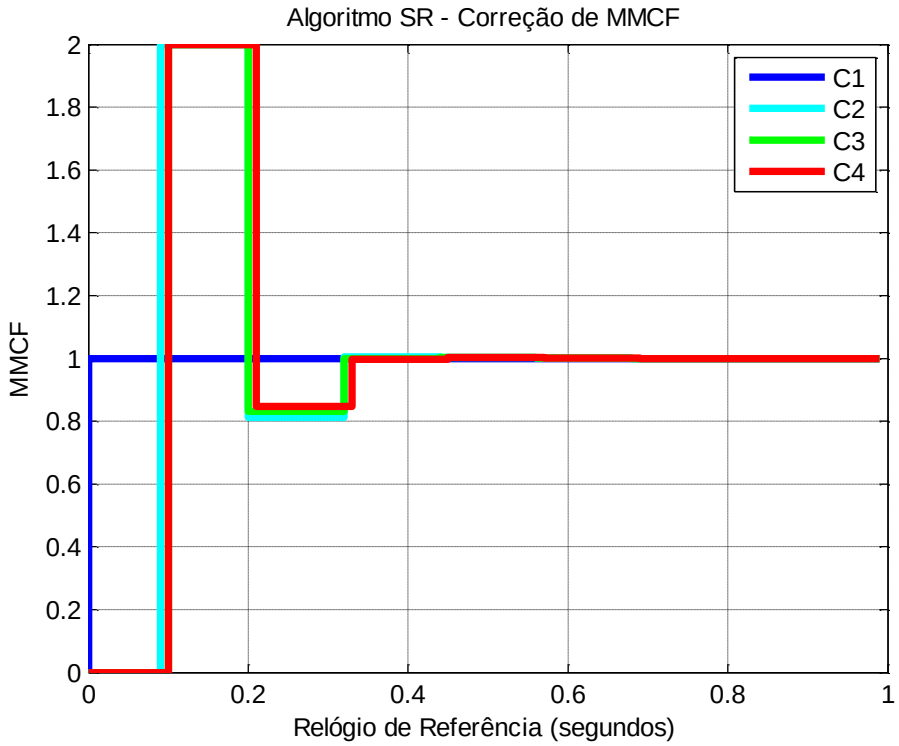


Figura 8.240 – Caso 40 - Correção modo NOMD - Algoritmo PAReS Ramp. TDMA.

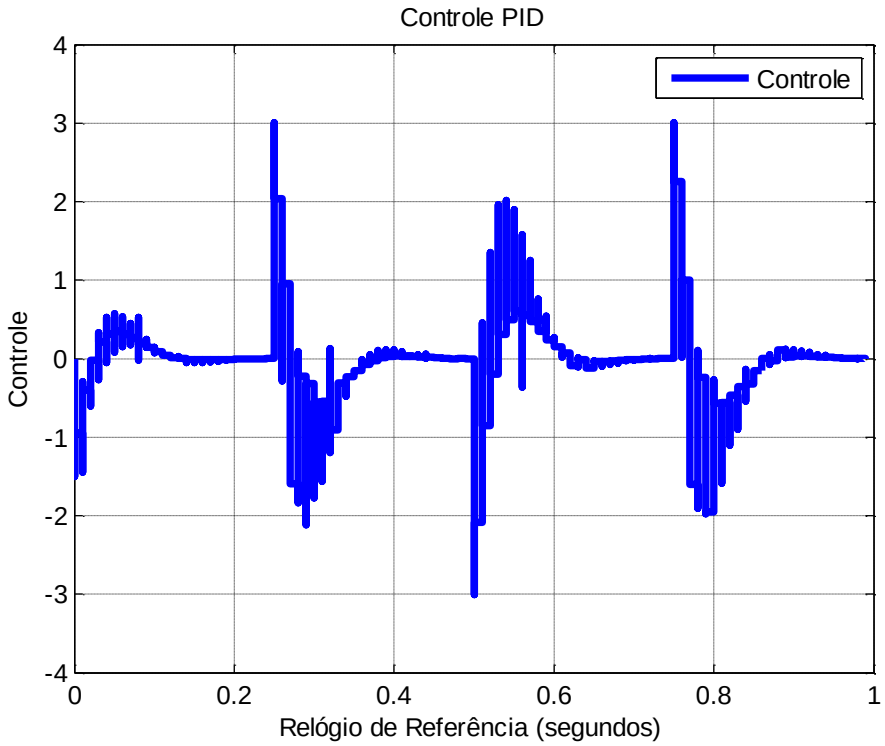


Figura 8.241 – Caso 40 - Controle PID.

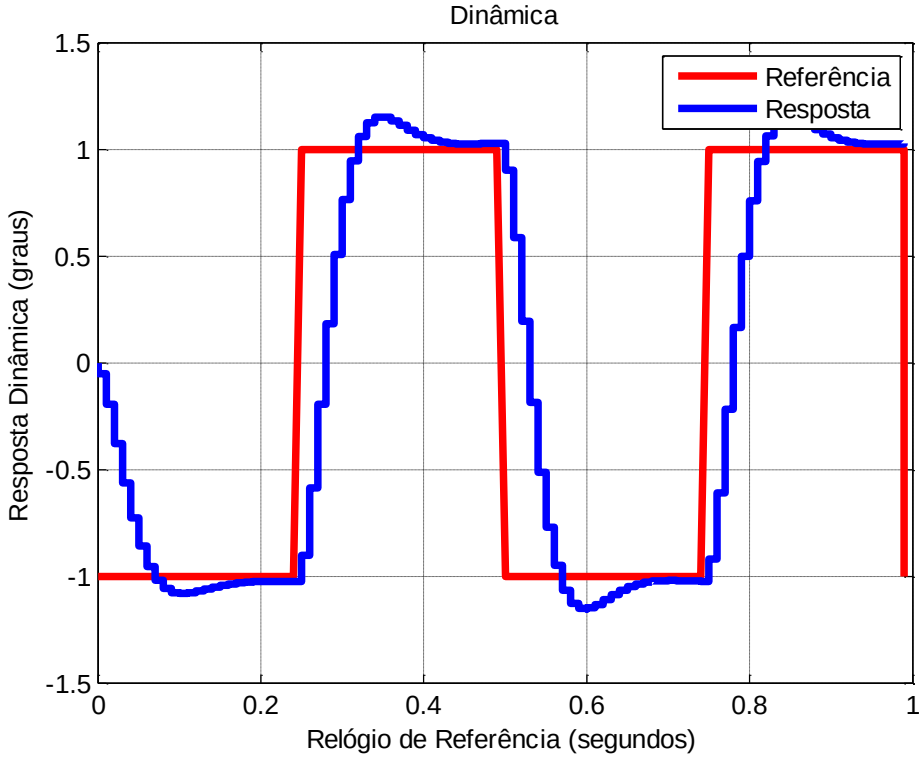


Figura 8.242 – Caso 40 - Resposta Dinâmica.

A principal diferença deste caso para o caso da seção 8.33 é que o gerenciamento de redes foi modificado. Em uma rede TDMA é possível estimar o pior caso de atraso de rede. Além disso, a rede TDMA aumenta o atraso de rede. Este caso teve um sobre sinal um pouco menor na resposta dinâmica, uma melhora da precisão e exatidão em relação ao caso da seção 8.33. Devido ao atraso de rede ser sempre constante era esperado que os casos TDMA obtivessem um melhor resultado em relação à rede CSMA/CD.

8.43. Caso 41 - Algoritmo PAReS Exponencial - TDMA

Os parâmetros e a sincronização de relógios é o mesmo da seção 8.34. A diferença é que este caso faz toda a simulação sobre uma rede TDMA. Assim, tem-se o seguinte resultado.

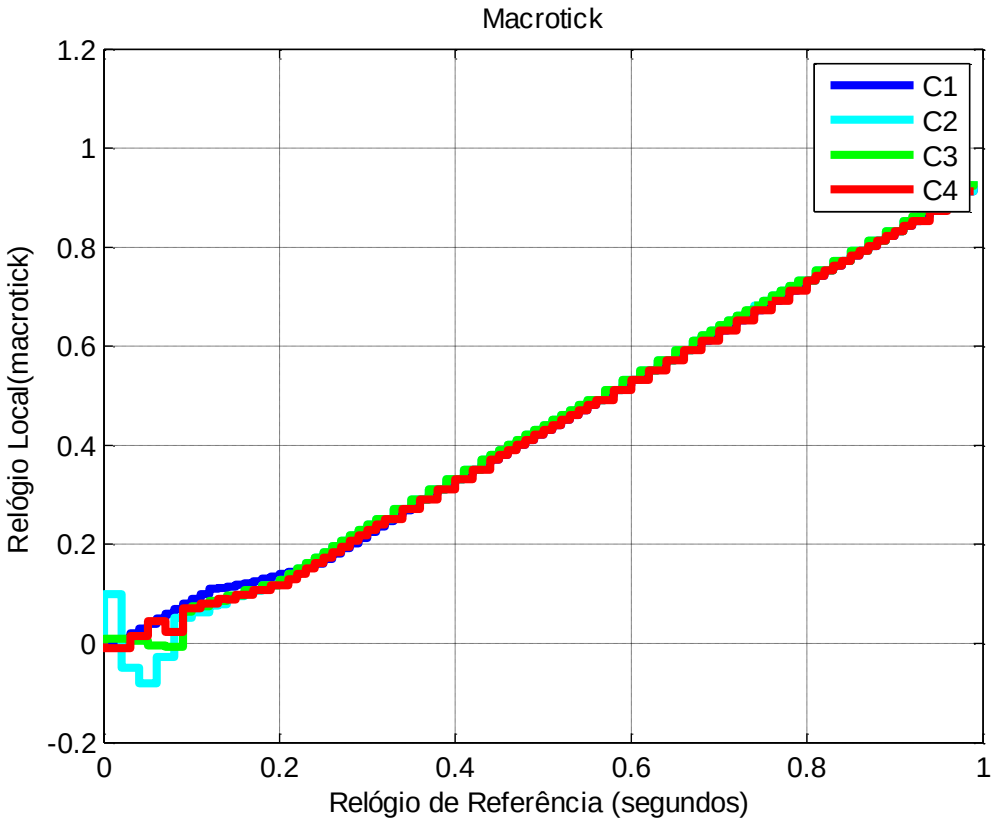


Figura 8.243 – Caso 41 - Macrotick.

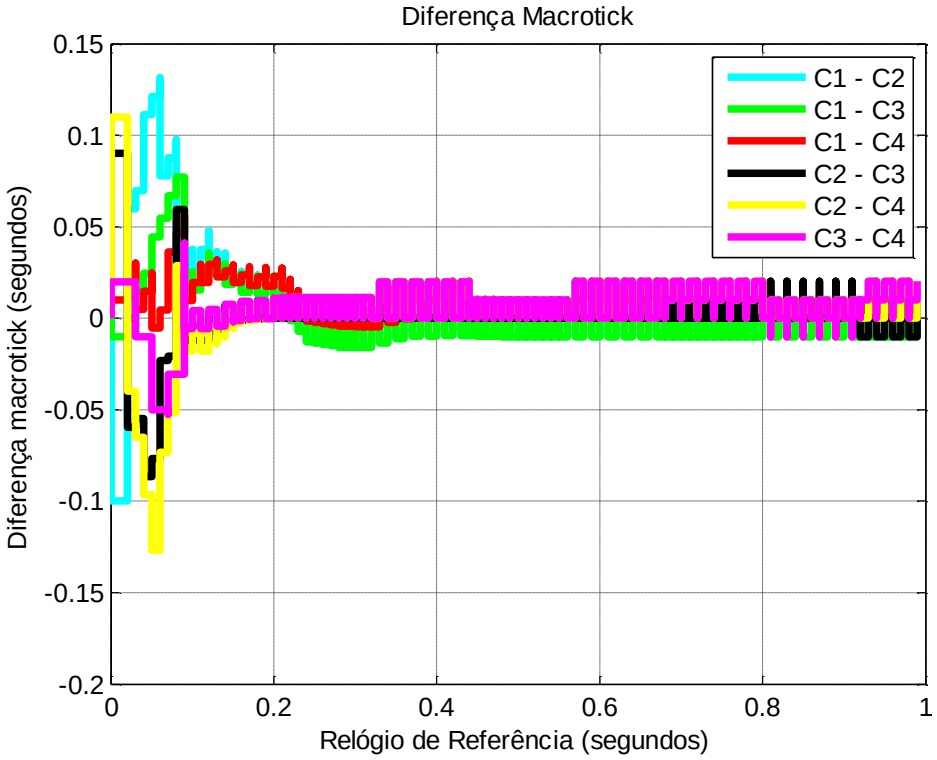


Figura 8.244 – Caso 41 - Diferença entre Macroticks.

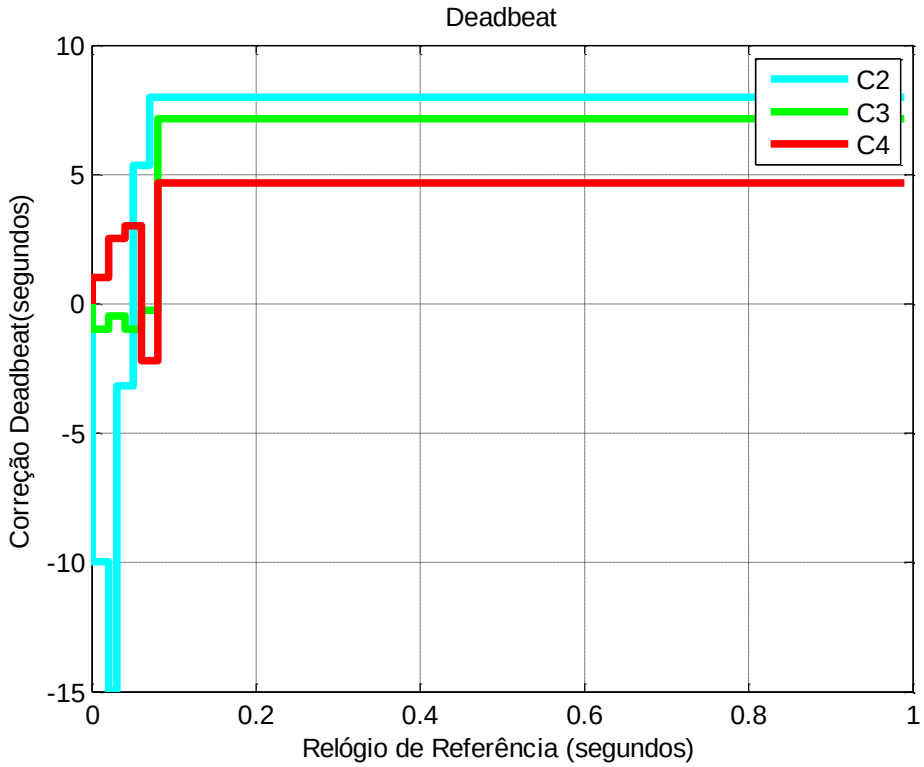


Figura 8.245 – Caso 41 - Modo STM - Algoritmo PAReS Exponencial - TDMA.

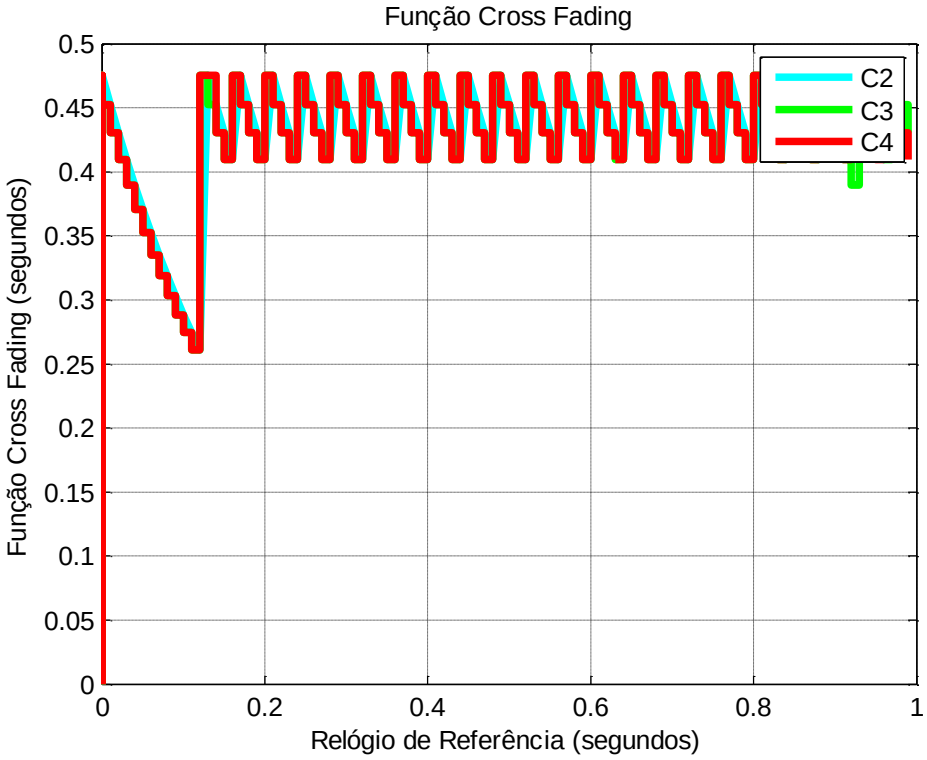


Figura 8.246 – Caso 41 - Função Cross-Fading Exponencial.

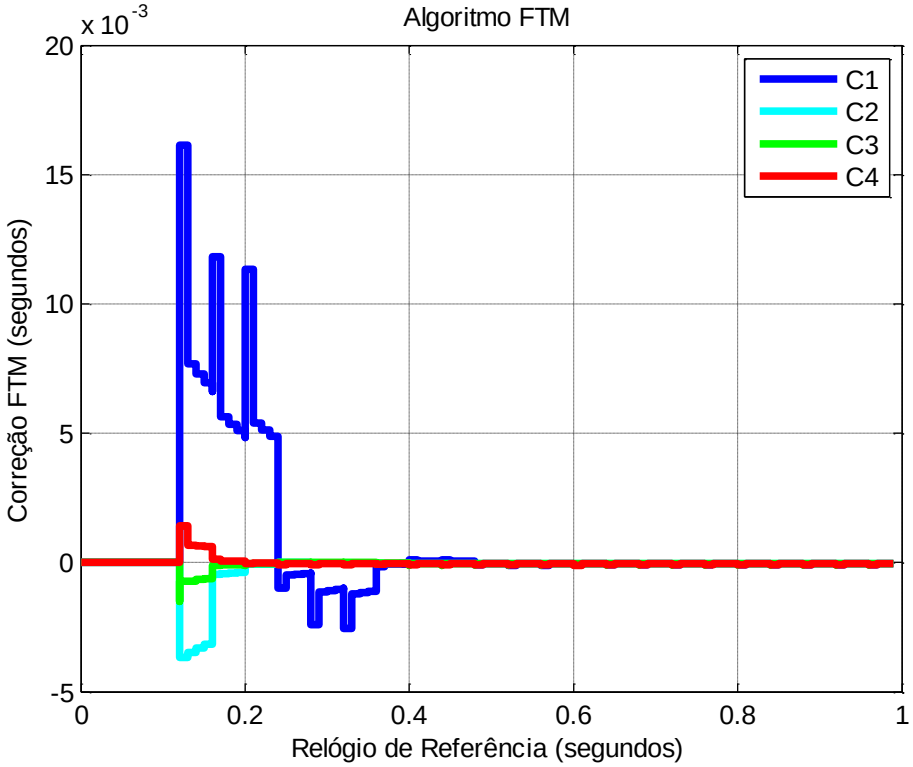


Figura 8.247 – Caso 41 - Correção modo NOMO - Algoritmo PAReS Exp. TDMA.

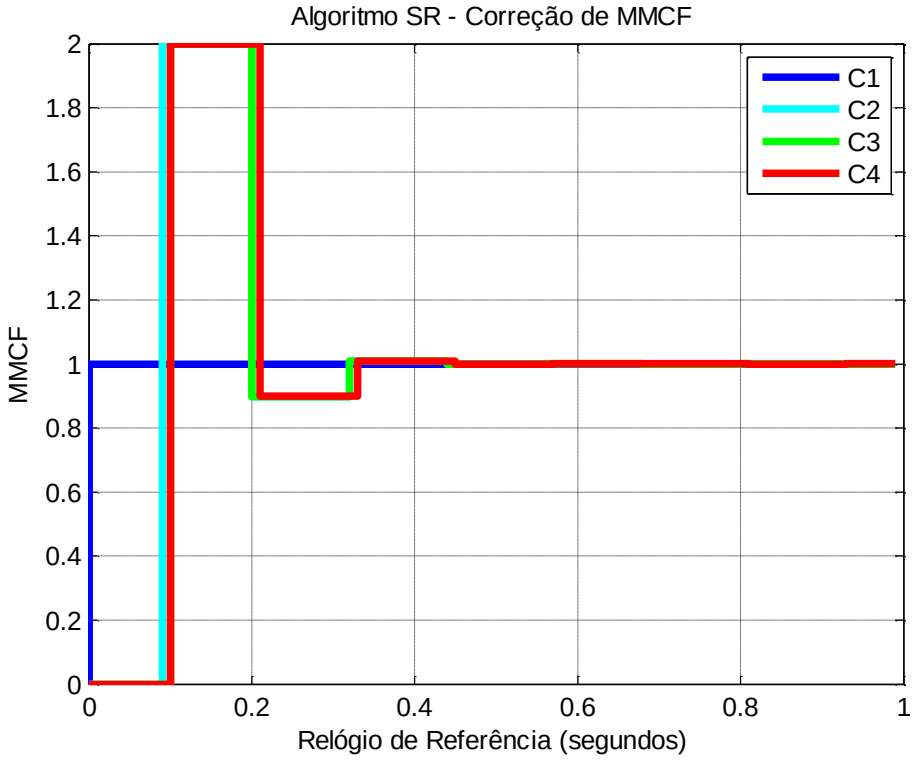


Figura 8.248 – Caso 41 - Correção modo NOMD - Algoritmo PAReS Exp. TDMA.

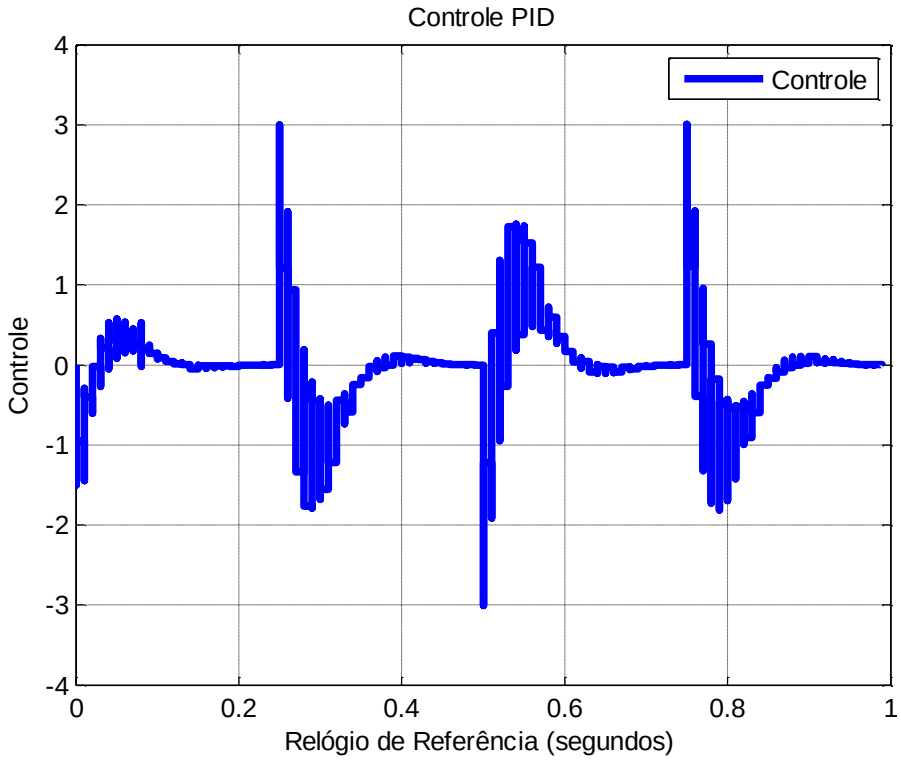


Figura 8.249 – Caso 41 - Controle PID.

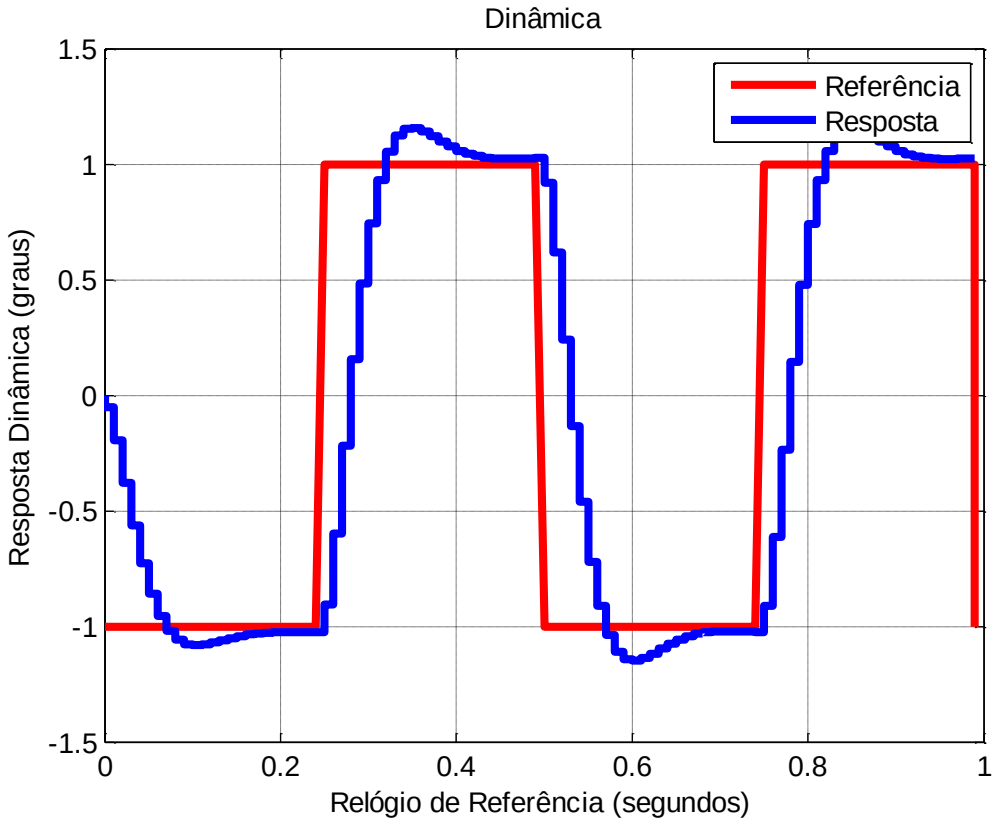


Figura 8.250 – Caso 41 - Resposta Dinâmica.

A principal diferença deste caso para o caso da seção 8.34 é que o gerenciamento de redes foi modificado. Em uma rede TDMA é possível estimar o pior caso de atraso de rede. Além disso, a rede TDMA aumenta o atraso de rede. Este caso teve um sobre sinal um pouco menor na resposta dinâmica, uma melhora da precisão e exatidão em relação ao caso da seção 8.34. Devido ao atraso de rede ser sempre constante era esperado que os casos TDMA obtivessem um melhor resultado em relação à rede CSMA/CD.

8.44. Indicadores de Desempenho

Os resultados gráficos são muito bons para verificar e validar os algoritmos sobre um sistema de controle por redes. No entanto, por meios gráficos é difícil analisar os indicadores de desempenho.

Para tanto, este trabalho escolheu 3 indicadores de desempenho para analisar os casos.

a) **Média das diferenças de macroticks:** A média é um ótimo indicador, pois indica o quão exato o conjunto está. Como temos várias séries temporais, para calcular este indicador este trabalho calcula a média de todas as séries temporais de diferenças de macroticks. Com isso, o maior valor do módulo das médias (ou seja, o conjunto que mais se afasta de zero) é o indicador de exatidão. Quanto mais próximo de zero o valor da média, mais exato está o conjunto.

b) **Desvio Padrão:** Aqui faz-se o mesmo procedimento realizado para a média nas séries temporais de diferenças de macroticks para encontrar o desvio padrão. No entanto, o desvio padrão é o indicador de precisão. Quanto menor o valor do desvio padrão, mais preciso (mais próximo entre si estão os valores de tempo) está o conjunto.

c) **Critério integral do erro quadrático (ISE):** este critério indica o quão próximo da referência esta à resposta dinâmica do seu sistema. Para tanto, foram gerados dois casos de simulação: 1) Com CSMA/CD e com 2) TDMA; sem a influência dos algoritmos de sincronização. Com isto, estes dois casos são os casos de referência dinâmica e servem como comparação para os casos com sincronização de relógios. Assim, para todas as respostas dinâmicas, foram calculados os critérios integrais utilizando a seguinte formulação:

$$ISE = \int_0^{\infty} erro^2 dt \tag{8.1}$$

$$erro = ref(t) - y(t) \tag{8.2}$$

Em que: ref(t) é o valor da função de referência e y(t) o valor da resposta dinâmica. No entanto, como todo o sistema é discreto, utilizou-se a regra do trapézio para a integração numérica.

8.45. Síntese das Simulações

A Tabela 8.2 apresenta os resultados de simulação com os indicadores de desempenho.

Para melhor entender a Tabela 8.2, a seguinte nomenclatura foi utilizada:

a) "NomeDoAlgoritmo": Somente o nome do algoritmo significa que ele foi utilizado de forma pura, como definido: Ex.: PTP.

b) "NomeDoAlgoritmo + Função Cross-Fading": indica que o algoritmo utilizado e a função cross-fading utilizada: Ex.: AReS Degrau - Algoritmo AReS com a função cross-fading degrau.

c) "*Deadbeat* + NomeDoAlgoritmo": indica que antes do algoritmo, foi aplicada uma sincronização inicial (Modo STM) utilizando o controle *deadbeat*.

Tabela 8.2 - Indicadores de desempenho - Simulação dos algoritmos de sincronização.

Casos	Desvio Padrão	Média	Critério Integral do Erro Quadrático (ITSE) - Dinâmica de Referência	Critério Integral do Erro Quadrático (ITSE) - Dinâmica com Sincronização	Algoritmos	Rede
Caso 1	0.00000000	0.00000000	0.00000000	0.00000000	Ideal	-
Caso 2	0.03020921	0.00307050	0.00000000	0.00000000	Deadbeat	-
Caso 3	0.03411767	0.01666922	0.00000000	0.00000000	PTP	-
Caso 4	0.03683457	0.02245300	0.00000000	0.00000000	Deadbeat - PTP	-
Caso 5	0.03160719	0.01352334	0.00000000	0.00000000	AReS Degrau	-
Caso 6	0.02980702	0.03481900	0.00000000	0.00000000	AReS Rampa	-
Caso 7	0.03092307	0.01680552	0.00000000	0.00000000	AReS Rampa	-
Caso 8	0.02247274	0.01827090	0.36203441	1587,52494841	PTP	CSMA/CD
Caso 9	0.02328796	0.02053082	0.36203441	0.53016632	Deadbeat - PTP	CSMA/CD
Caso 10	0.02594739	0.01037445	0.36203441	0.45361669	AReS Degrau	CSMA/CD
Caso 11	0.02084018	0.03353262	0.36203441	0.48545149	AReS Rampa	CSMA/CD
Caso 12	0.02177582	0.01406182	0.36203441	0.41872875	AReS Exponencial	CSMA/CD
Caso 13	0.03404249	0.01546472	0.00000000	0.00000000	FTM	-
Caso 14	0.03682519	0.02177861	0.00000000	0.00000000	Deadbeat - FTM	-
Caso 15	0.03426164	0.03115537	0.00000000	0.00000000	PReS Degrau	-
Caso 16	0.03750091	0.02815829	0.00000000	0.00000000	PReS Rampa	-
Caso 17	0.03759079	0.02806664	0.00000000	0.00000000	PReS Exponencial	-
Caso 18	0.03351872	0.01232386	0.36203441	0.38496132	FTM	CSMA/CD
Caso 19	0.03004284	0.02497284	0.36203441	0.35071464	Deadbeat - FTM	CSMA/CD
Caso 20	0.03134342	0.03054203	0.36203441	0.35217439	PReS Degrau	CSMA/CD
Caso 21	0.03128230	0.02630475	0.36203441	0.35351948	PReS Rampa	CSMA/CD

Caso 22	0.03110070	0.02666633	0.36203441	0.35400987	PReS Exponencial	CSMA/CD
Caso 23	0.03366050	-0.01320109	0.00000000	0.00000000	SR	-
Caso 24	0.04062038	0.00852185	0.00000000	0.00000000	Deabeat - SR	-
Caso 25	0.03445895	0.00628870	0.00000000	0.00000000	PAReS Degrau	-
Caso 26	0.03507067	0.00670348	0.00000000	0.00000000	PAReS Rampa	-
Caso 27	0.03442285	0.00469874	0.00000000	0.00000000	PAReS Exponencial	-
Caso 28	0.02699320	-0.00792600	0.36203441	0.35965438	SR	CSMA/CD
Caso 29	0.02455384	0.00036928	0.36203441	0.35394183	Deabeat - SR	CSMA/CD
Caso 30	0.03014176	0.00387957	0.36203441	0.35569277	PAReS Degrau	CSMA/CD
Caso 31	0.03112148	0.00444227	0.36203441	0.35403245	PAReS Rampa	CSMA/CD
Caso 32	0.02920917	0.00267943	0.36203441	0.35217343	PAReS Exponencial	CSMA/CD
Caso 33	0.02629688	0.01050516	0.35498787	0.45177425	AReS Degrau	TDMA
Caso 34	0.02145692	0.03333821	0.35498787	0.49723351	AReS Rampa	TDMA
Caso 35	0.02225724	0.01392661	0.35498787	0.44120419	AReS Exponencial	TDMA
Caso 36	0.02925873	0.03111299	0.35498787	0.35635858	PReS Degrau	TDMA
Caso 37	0.02967778	0.02686567	0.35498787	0.35721012	PReS Rampa	TDMA
Caso 38	0.02936551	0.02726333	0.35498787	0.34862219	PReS Exponencial	TDMA
Caso 39	0.03221836	0.00413843	0.35498787	0.35533770	PAReS Degrau	TDMA
Caso 40	0.03279156	0.00488695	0.35498787	0.34581259	PAReS Rampa	TDMA
Caso 41	0.03119883	0.00355690	0.35498787	0.35318225	PAReS Exponencial	TDMA

Da Tabela 8.2, em verde estão os melhores casos de cada coluna e em vermelho os piores casos.

Para uma melhor visualização, resumi-se a Tabela 8.2 com os melhores e piores casos e também dos algoritmos FTM, PTP e SR como comparação na seguinte tabela:

Tabela 8.3 - Melhores e Piores Casos da Simulação dos algoritmos de sincronização.

Algoritmo	Desvio Padrão	Média	Critério Integral	Rede	
PTP	0.03411767	0.01666922	-	-	
FTM	0.03404249	0.01546472	-	-	
SR	0.03366050	-0.01320109	-	-	
PTP	0.02247274	0.01827090	1587,524948	CSMA/CD	Pior Critério
FTM	0.03351872	0.01232386	0.38496132	CSMA/CD	
SR	0.02699320	-0.00792600	0.35965438	CSMA/CD	
AReS Rampa	0.02084018	0.03353262	0.48545149	CSMA/CD	Melhor Desvio Padrão
Deabeat - SR	0.02455384	0.00036928	0.35394183	CSMA/CD	Melhor Média
Deadbeat - FTM	0.03004284	0.02497284	0.35071464	CSMA/CD	Melhor Critério CSMA/CD
PAReS Rampa	0.03279156	0.00488695	0.34581259	TDMA	Melhor Critério
Deabeat - SR	0.04062038	0.00852185	-	-	Pior Desvio Padrão
AReS Rampa	0.02980702	0.03481900	-	-	Pior Média
AReS Rampa	0.02145692	0.03333821	0.49723351	TDMA	Pior Critério TDMA

O caso 1 é o ideal. Baseado nisso, nota-se então que os melhores indicadores são os indicadores que são o mais próximo de zero, possível. Observa-se no do PTP com sistema de controle em CSMA/CD a sincronização foi eficiente, mas o sistema de controle instabilizou sendo o pior critério integral de todas as simulações.

Com o caso AReS Rampa com controle em rede CSMA/CD, obteve-se o melhor desvio padrão da sincronização dos relógios. No entanto, o desempenho do controle foi degradado. No caso Deadbeat com SR e sistema de controle CSMA/CD, obteve-se a melhor média da sincronização e além

disso houve uma melhora pequena, em relação ao caso de referência, na resposta do controle (critério integral).

O *Deadbeat* com FTM, obteve o melhor resultado com rede CSMA/CD para o critério integral.

O Algoritmo PAReS Rampa obteve o melhor critério integral e portanto o melhor desempenho do sistema de controle, pois melhorou a resposta em relação ao caso nominal.

O *deadbeat* com SR e o Algoritmo AReS sem controle e o algoritmo AReS Rampa com rede TDMA, obtiveram os piores indicadores de desvio padrão, média e critério integral, respectivamente.

Observa-se portanto que o conjunto de algoritmos ReS e os procedimentos propostos melhoraram o desempenho tanto da sincronização e principalmente do sistema de controle. As métricas mistas de precisão, exatidão sobre um algoritmo de reconfiguração e suavizável demonstram uma abordagem interessante no projeto de sistemas de controle.

Com otimização e ajustes de parâmetros é possível chegar a resultados muito melhores.

9 ANÁLISE DE ESTABILIDADE DOS MODELOS MATEMÁTICOS E ALGORITMOS DE SINCRONIZAÇÃO

Uma das grandes vantagens da abordagem matemática por modelos é a possibilidade da análise matemática das habilidades de um sistema, tal como a estabilidade. Neste Capítulo 9 é apresentada a análise de estabilidade dos modelos matemáticos em transformada Z e equações de diferenças, apresentados no Capítulo 4, além dos algoritmos de sincronização propostos.

9.1. Análise de Estabilidade do Microtick

O modelo matemático do microtick na transformada Z é dado pela equação (4.17), reescrita abaixo. Com isso, tem-se:

$$MiT_i(z) = \frac{\rho_i T}{z-1} \tag{9.1}$$

Assim, para um período de amostragem de 0.01 segundos e uma deriva de 0.01 (seg./seg), a Figura 9.1 mostra o diagrama de pólos e zeros do microtick e a Figura 9.2 mostra o lugar das raízes.

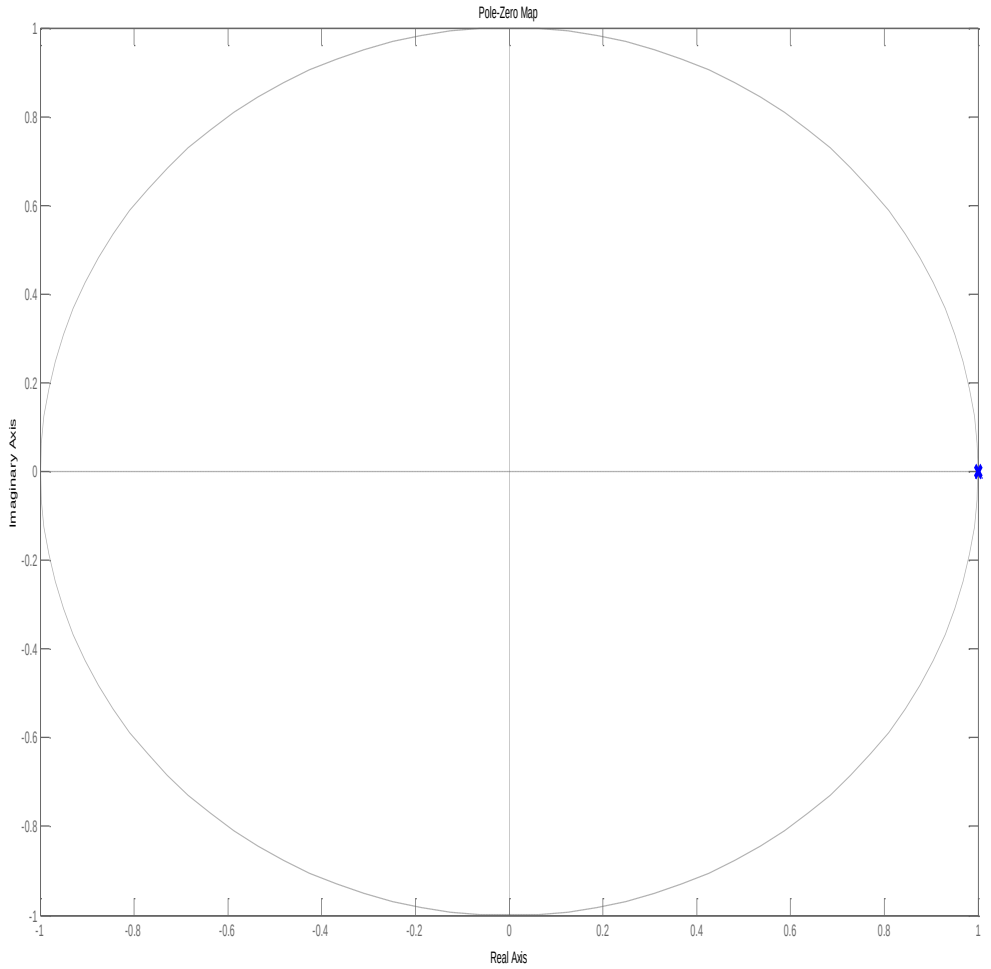


Figura 9.1 – Diagrama de pólos e zeros do microtick.

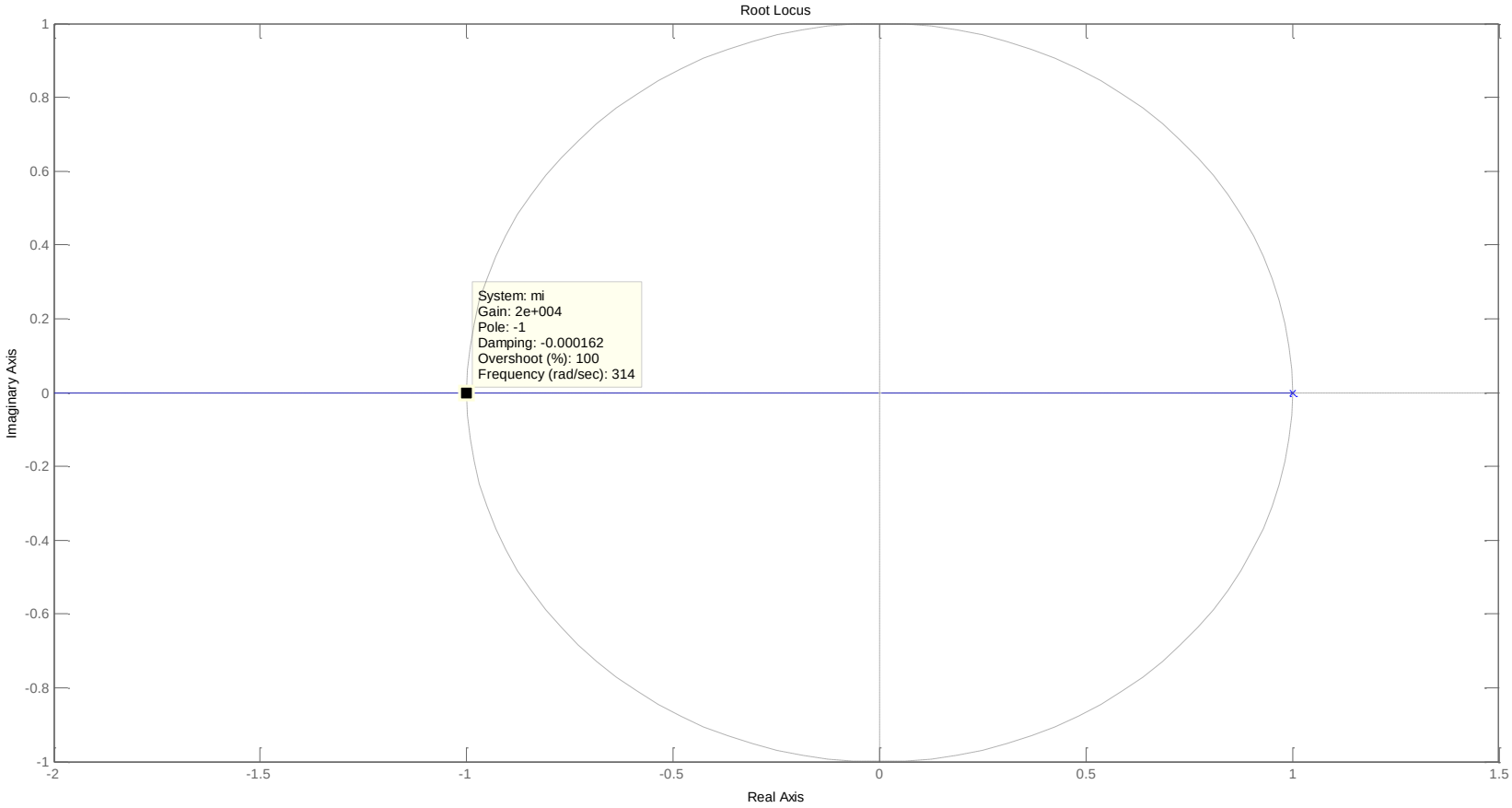


Figura 9.2 – Diagrama do lugar das raízes do microtick.

O diagrama de pólos e zeros possui um pólo sobre o círculo unitário, indicando que o microtick é marginalmente estável em malha aberta. Já o lugar das raízes, demonstra que ele é estabilizável em malha fechada e que o limite de estabilidade do sistema para o projeto de sistemas de controle é elevado. Como o ganho é muito alto para o domínio dos relógios, dificilmente um sistema de controle em malha fechada tornará o sistema instável.

9.2. Análise de Estabilidade do Macrotick

O modelo matemático do macrotick na transformada Z é dado pela equação (4.18). Reorganizando, incluindo a equação do microtick e considerando a função Ш igual a 1 em todos os períodos de amostragem, ou seja admite-se que o macrotick tem algum incremento devido ao microtick em todos os instantes de amostragem. Com isso, tem-se:

$$MAT_i(z) = \frac{\rho_i T\left(1 - z^{-(1+MMCF_i)}\right)}{z^{-1}(z-1)^2 MMCF_i} \qquad (9.2)$$

Para fazer a análise de estabilidade, 3 parâmetros podem ser variados (MMCF, ρ e T). No caso do MMCF, ao se variar o MMCF a função de transferência se modifica. Portanto, nesta tese simula-se o MMCF variando de 1, 10 e 20. Para cada um destes MMCF´s, aplica-se para análise de estabilidade pelo plano de pólos e zero um período de amostragem (T) de 0.01 segundos e uma deriva de 0.01 (seg/seg), verificando a estabilidade do macrotick em malha aberta.

Depois de fazer a análise do plano de pólos e zeros, aplica-se o lugar das raízes (*root-locus*), em que se varia o termo $\rho_i T$, verificando a estabilidade do macrotick em malha fechada.

a) Para MMCF=1: A Figura 9.3 mostra o diagrama de pólos e zeros do macrotick, com 2 pólos na origem e 2 sobre o circulo unitário. Observa-se que o sistema é marginalmente estável em malha aberta. A Figura 9.4 mostra o lugar das raízes.

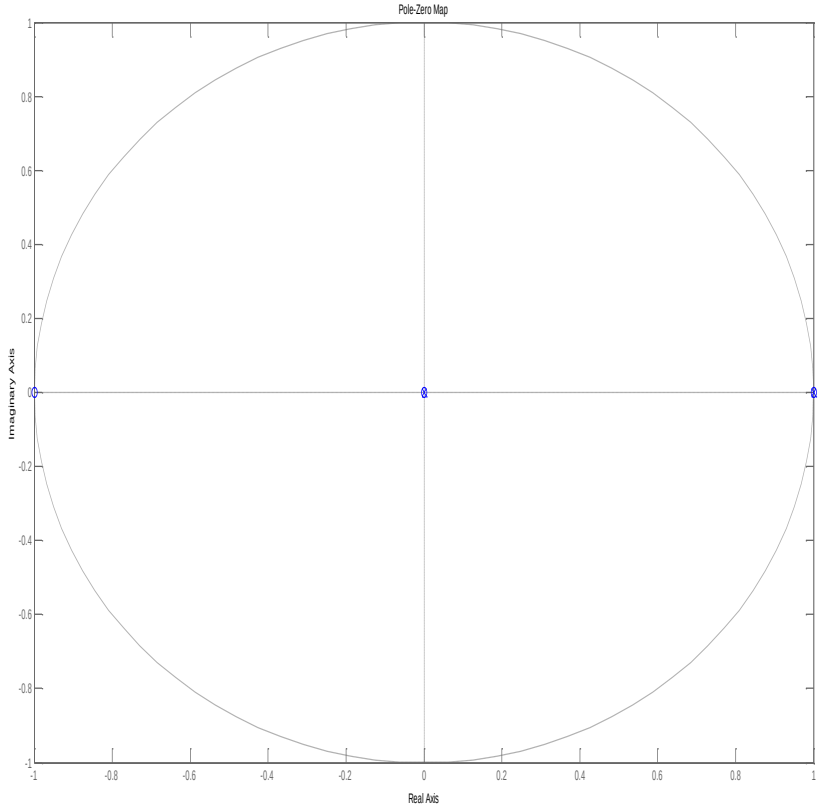


Figura 9.3 – Diagrama de pólos e zeros do macrotick.

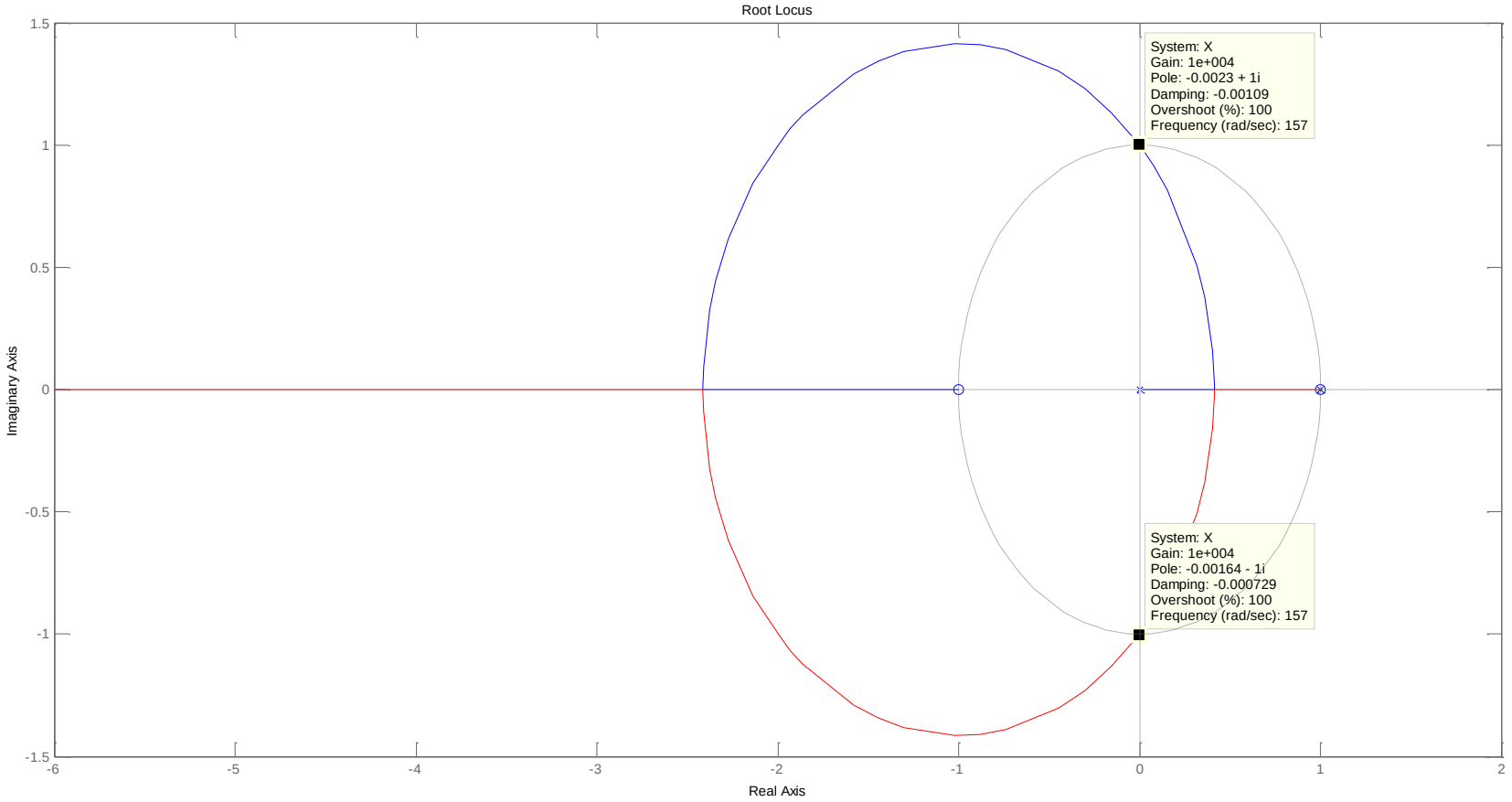


Figura 9.4 – Diagrama do lugar das raízes do macrotick - MMCF=1.

b) Para MMCF=10: A Figura 9.5 mostra o diagrama de pólos e zeros com 11 pólos na origem e 2 no circulo unitário e 1 zero na origem e outros 11 zeros sobre o circulo unitário. Para estes parâmetros, o sistema é marginalmente estável em malha aberta. A Figura 9.6 (a) sem informações dos ganhos e (b) Com informações dos ganhos, mostra o lugar das raízes. Assim, observa-se que variando o MMCF e também o termo $\rho_i T$ o sistema torna-se condicionalmente instável, contendo algumas zonas que podem levar o sistema a instabilidade.

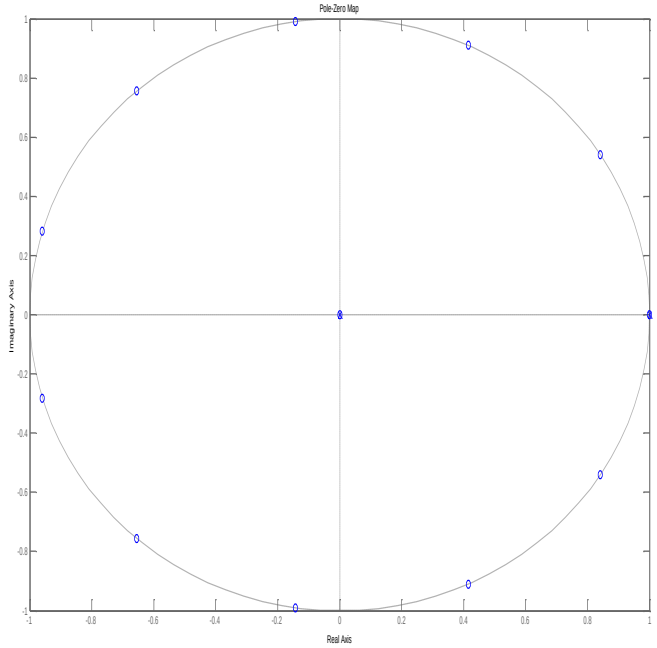
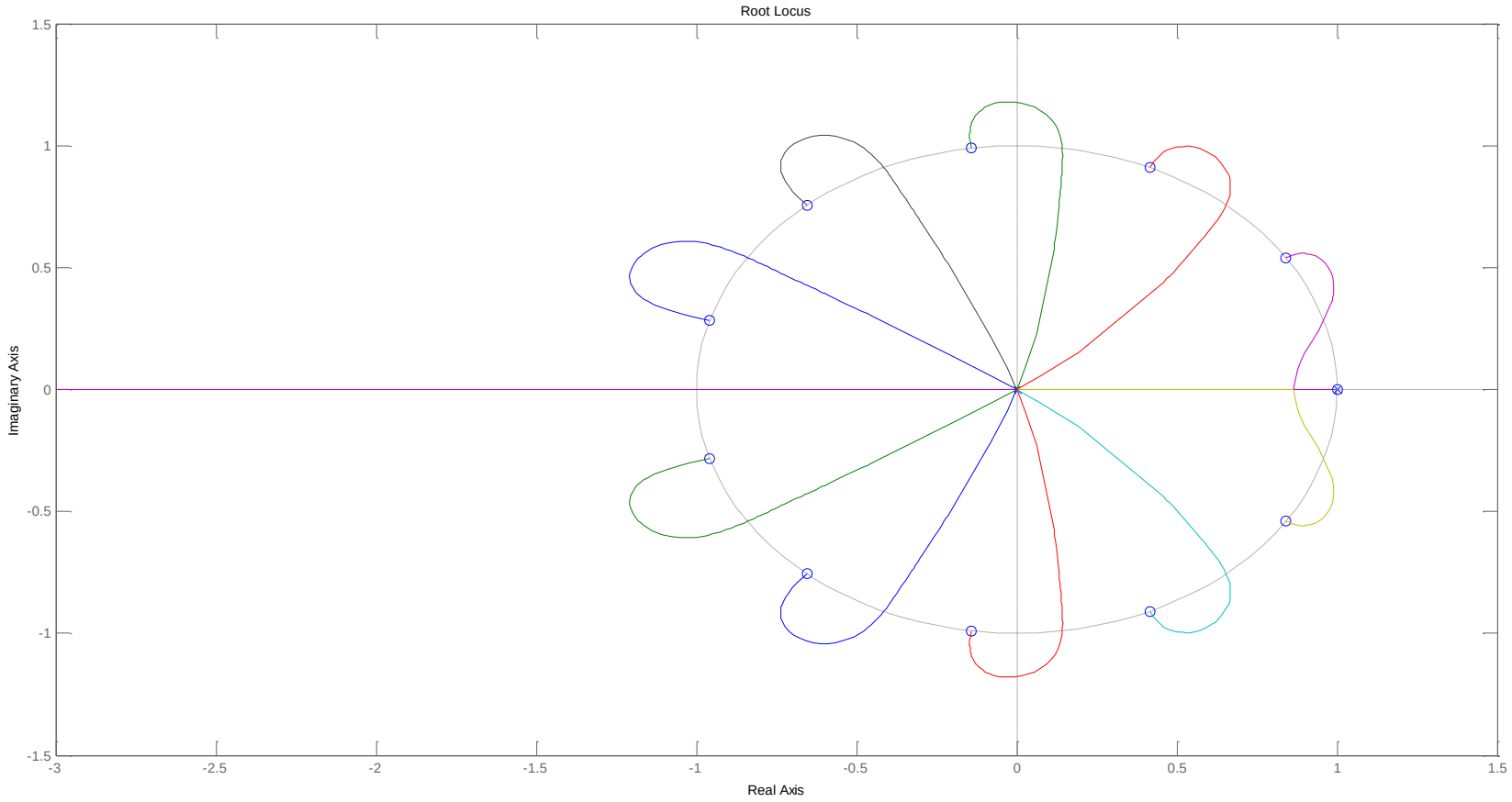
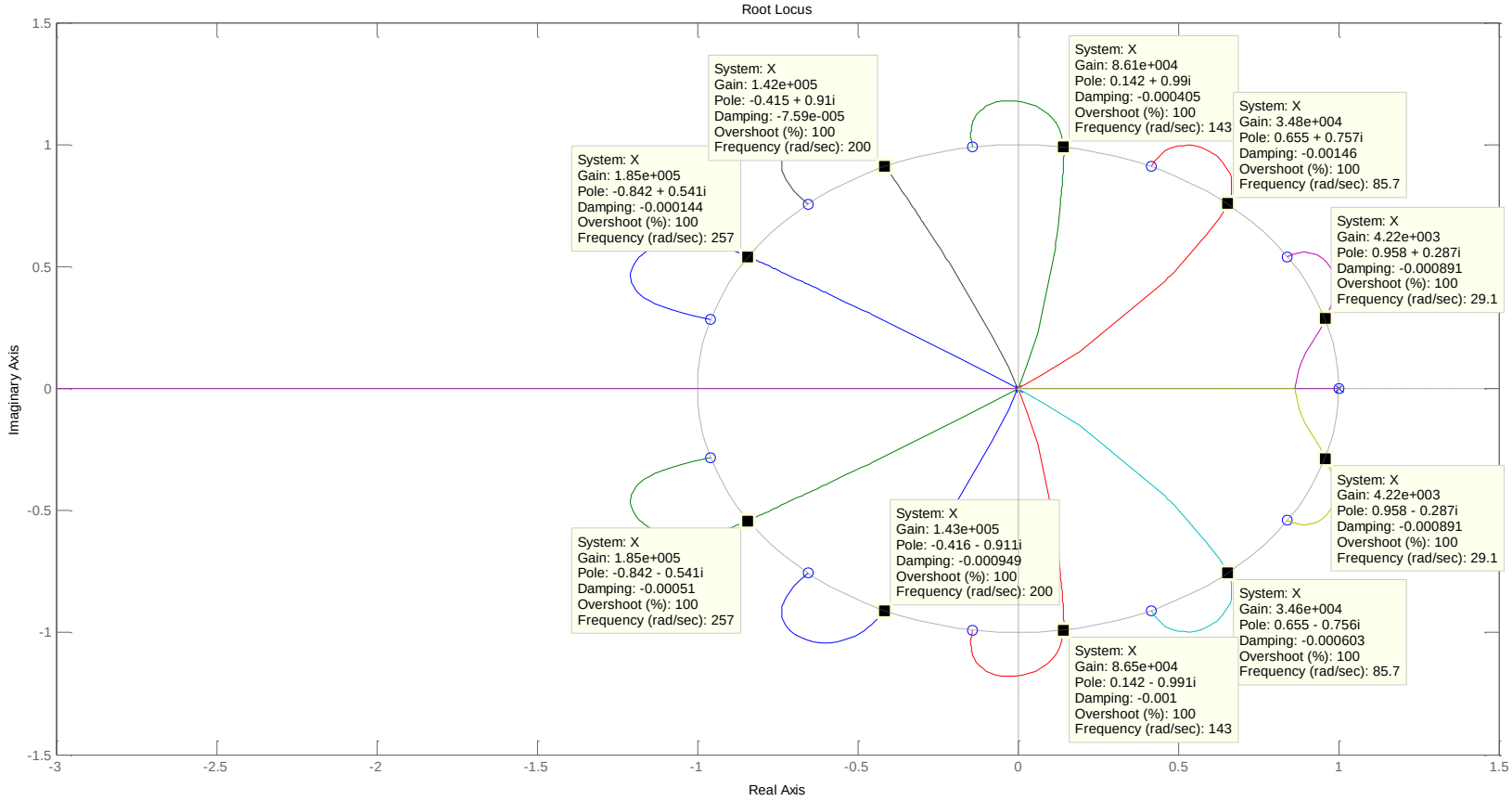


Figura 9.5 – Diagrama de pólos e zeros do macrotick - MMCF = 10.


Root Locus
Imaginary Axis
Real Axis
1.5
1
0.5
0
-0.5
-1
-1.5
-3
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5

(a)



(b)

Figura 9.6 – Diagrama do lugar das raízes do macrotick - MMCF = 10: (a) Sem Ganhos; (b) Com ganhos.

c) MMCF=20; A Figura 9.7 mostra o diagrama de pólos e zeros com 21 pólos na origem e 2 no circulo unitário, e 1 zero na origem e 21 zeros sobre o circulo unitário. Para estes parâmetros o sistema é marginalmente estável em malha aberta. A Figura 9.8 mostra o lugar das raízes. Assim como anteriormente, observa-se que variando o MMCF e o termo $\rho_i T$ o sistema torna-se condicionalmente instável, contendo zonas que podem levar o sistema à instabilidade. Comparando com o caso anterior, observa-se nas figuras do lugar da raízes que quanto maior o MMCF e o termo $\rho_i T$, maior é a possibilidade de existirem caminhos instáveis para o controle. Contudo, ao continuar aumentando o MMCF e o termo $\rho_i T$, o sistema volta à estabilidade. Este efeito foi demonstrado em Tredinnick (2009), em que verificou-se a presença de regiões de instabilidade e estabilidade do sistema do controle discreto de planta analógica como função do período de amostragem T. Este efeito também é observado por esta tese, atráves da variação do MMCF e do termo $\rho_i T$, em que de fato é a variação do período de amostragem da planta do macrotick. A variação de deriva ρ_i é muito pequena (da ordem de 1e-3 seg/seg em relógios falhados que normalmente são descartados da sincronização) e portanto, considerando para esta análise a mesma constante. No entanto, observa-se que se o relógio falhado estiver participando da malha de controle e a deriva não for tão pequena em relação ao período de amostragem, a variação da deriva também pode gerar o mesmo efeito.

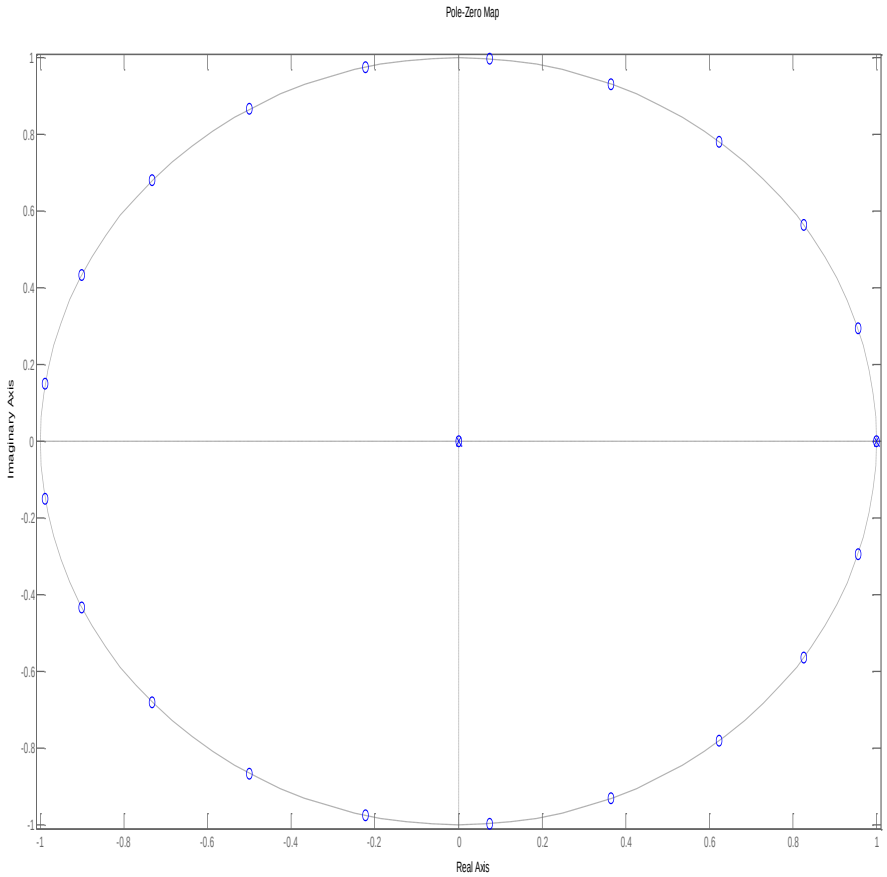


Figura 9.7 – Diagrama de pólos e zeros do macrotick.

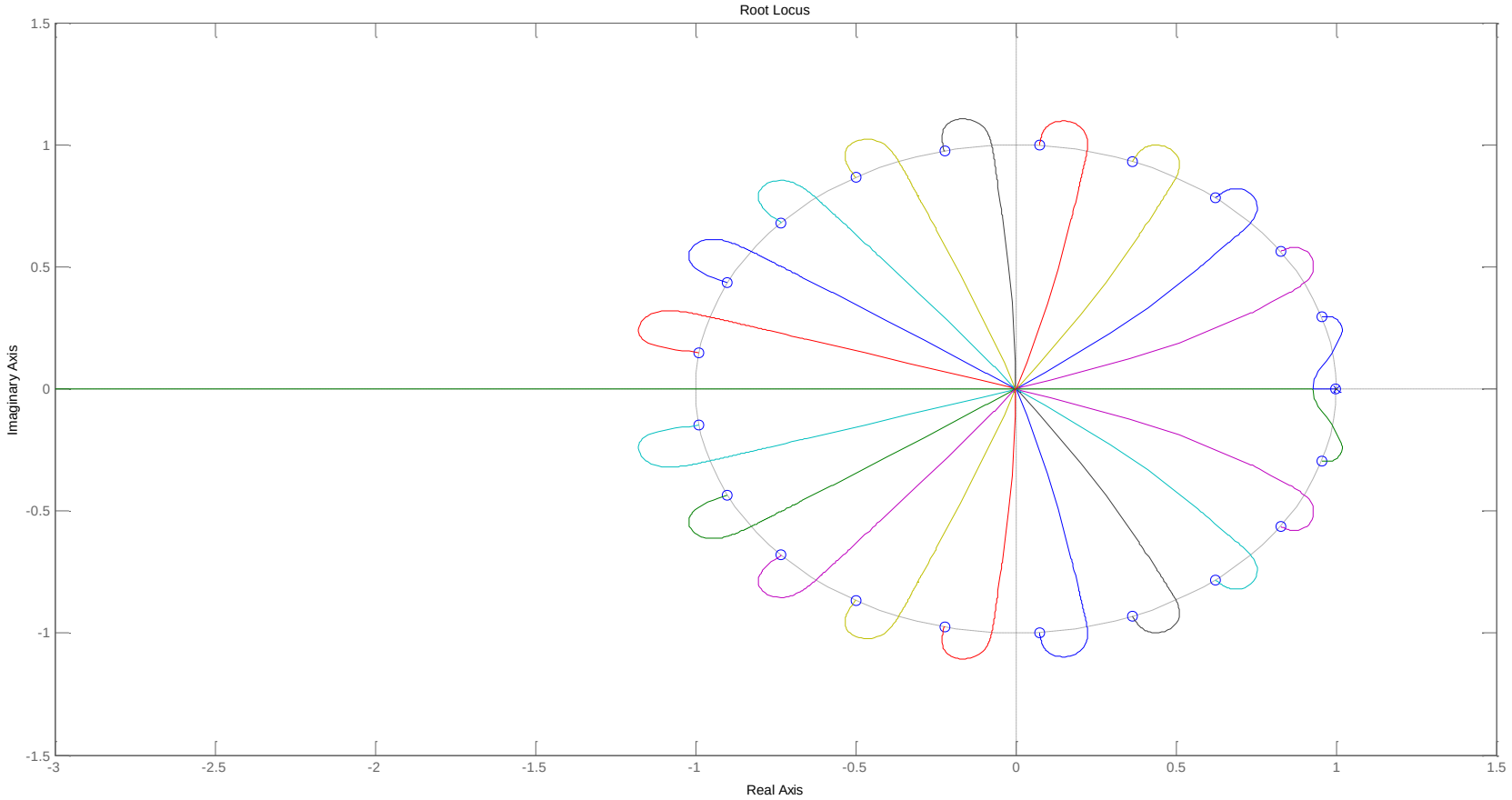


Figura 9.8 – Diagrama do lugar das raízes do macrotick - MMCF =20.

9.3. Análise de Estabilidade do Deadbeat

Os algoritmos ReS projetam um controlador *deadbeat* em malha fechada para uma entrada rampa em seu modo STM. Portanto, será feito na sequência a análise da estabilidade do sistema.

A equação (6.1) mostra a equação em malha fechada, reescrita abaixo:

$$\frac{C(z)}{R(z)} = M(z) = \frac{D(z)\mu T(z)}{1 + D(z)\mu T(z)} \tag{9.3}$$

Sabendo que μT(z) é dado pela a equação (9.1), idêntica à equação da dinâmica e o D(z) é dado pela equação (6.16), reescrita abaixo:

$$D(z) = \frac{1}{\left(\rho T - \mu t(0)\right)} \frac{(2z^2 - 3z + 1)z^{-2}}{(z^2 - 2z + 1)z^{-2}} \tag{9.4}$$

Com isso, é possível fazer a análise em malha fechada da estabilidade do sistema para diferentes parâmetros. Consideraremos μT(z) primeiramente como a equação do microtick (equação (9.1)) e depois para o macrotick (equação (9.2)).

Para tanto, utilizaram-se os seguintes parâmetros: 1) Período de amostragem T = 0.01 seg.; 2) Deriva ρ = 0.01 (seg./seg); 3) Condição inicial = 0.1 (seg.); 4) MMCF = 1, MMCF=2 e MMCF = 10;

Tem-se portanto:

a) Microtick: A Figura 9.9 mostra o diagrama de pólos e zeros da malha fechada do *deadbeat* sobre um macrotick com MMCF=1. Observa-se 8 pólos na origem, 3 pólos dentro do círculo unitário e 3 pólos fora do circulo unitário. E 8 zeros na origem, 2 dentro do circulo unitário e 2 fora do circulo unitário. Observa-se que existem pólos e também zeros fora do círculo unitário. No entanto, os pólos que estão fora do círculo são tão pequenos (ordem de 1e-3 a 1e-2 de diferença para 1)

o que dificulta a analisar se este fato é devido a erros numéricos ou instabilidade do sistema. No entanto, é conhecido que a funções do Matlab usadas (pzmap, pole, zero) para analisar a estabilidade possuem limitações, demonstradas no Apêndice B. Portanto, para este caso, considera-se que a instabilidade é devido a erros numéricos e portanto o sistema é marginalmente estável em malha aberta. A Figura 9.11 mostra o diagrama do lugar das raízes de uma função D(z)μT(z) (malha aberta do sistema com *deadbeat*). Observa-se que em malha fechada, o controle com *deadbeat*, é marginalmente estável. Observa-se, os mesmos erros numéricos da Figura 9.9 devidos as limitações demonstradas no Apêndice B.

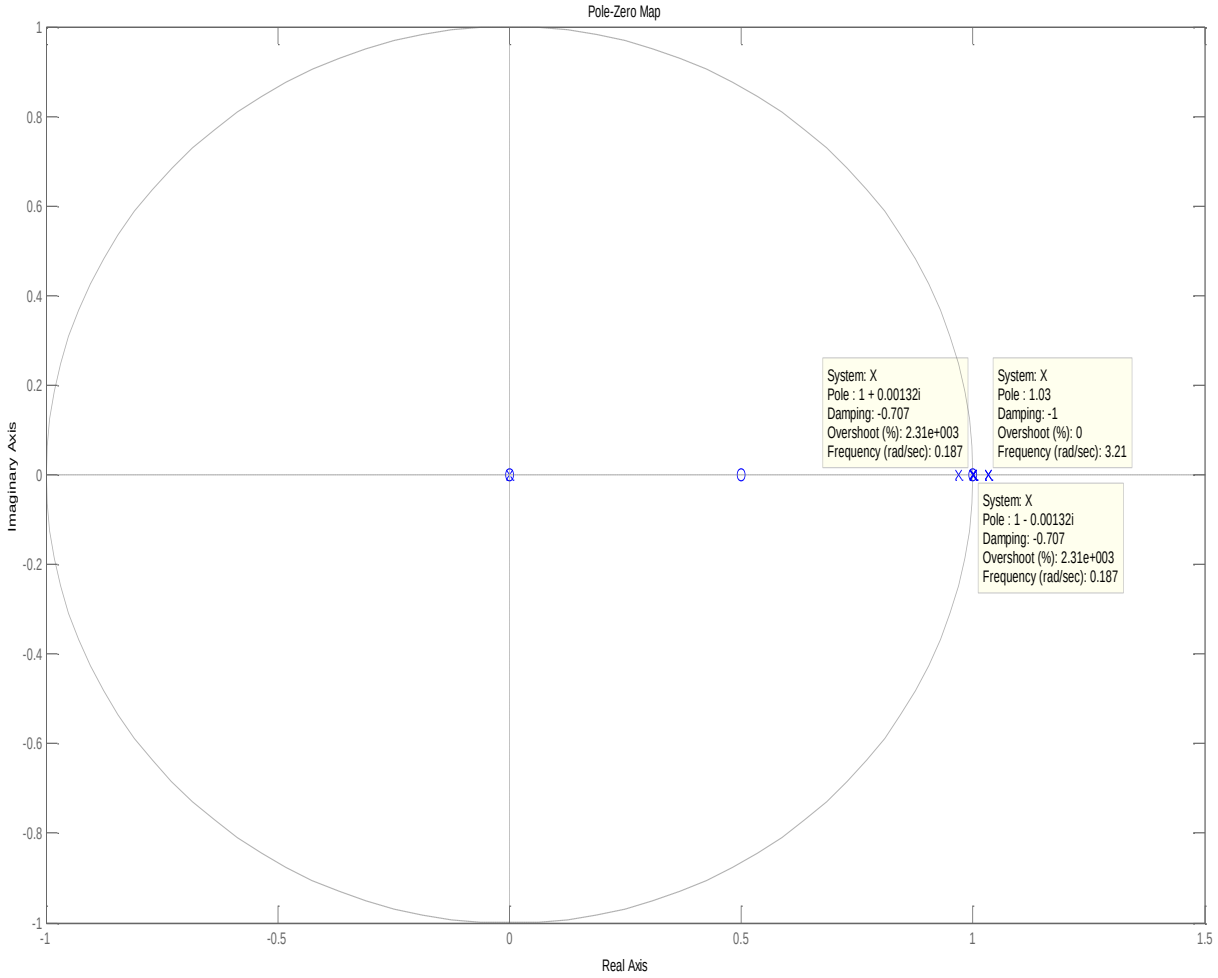


Figura 9.9 – Diagrama de pólos e zeros do *deadbeat* - microtick - Malha fechada.

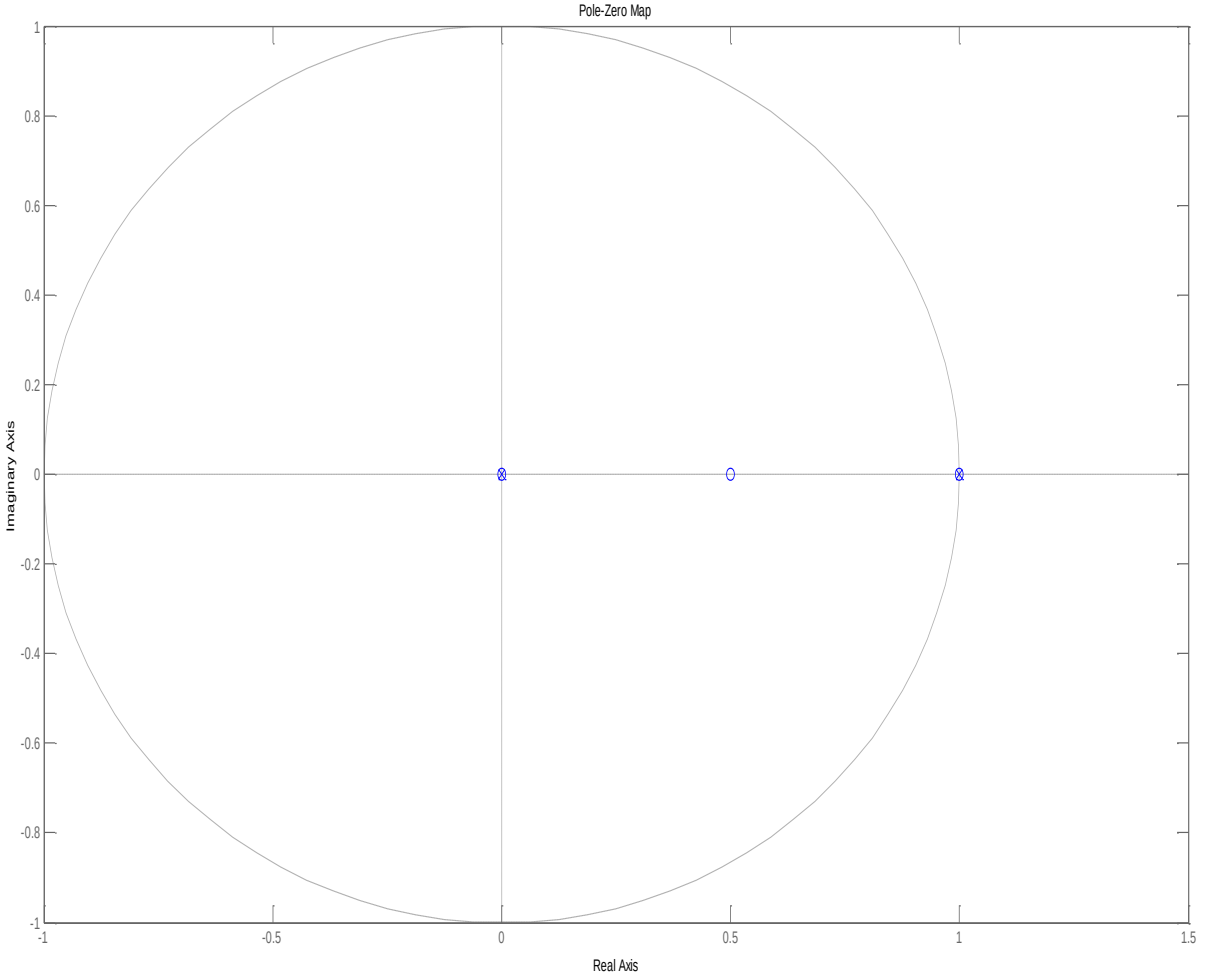
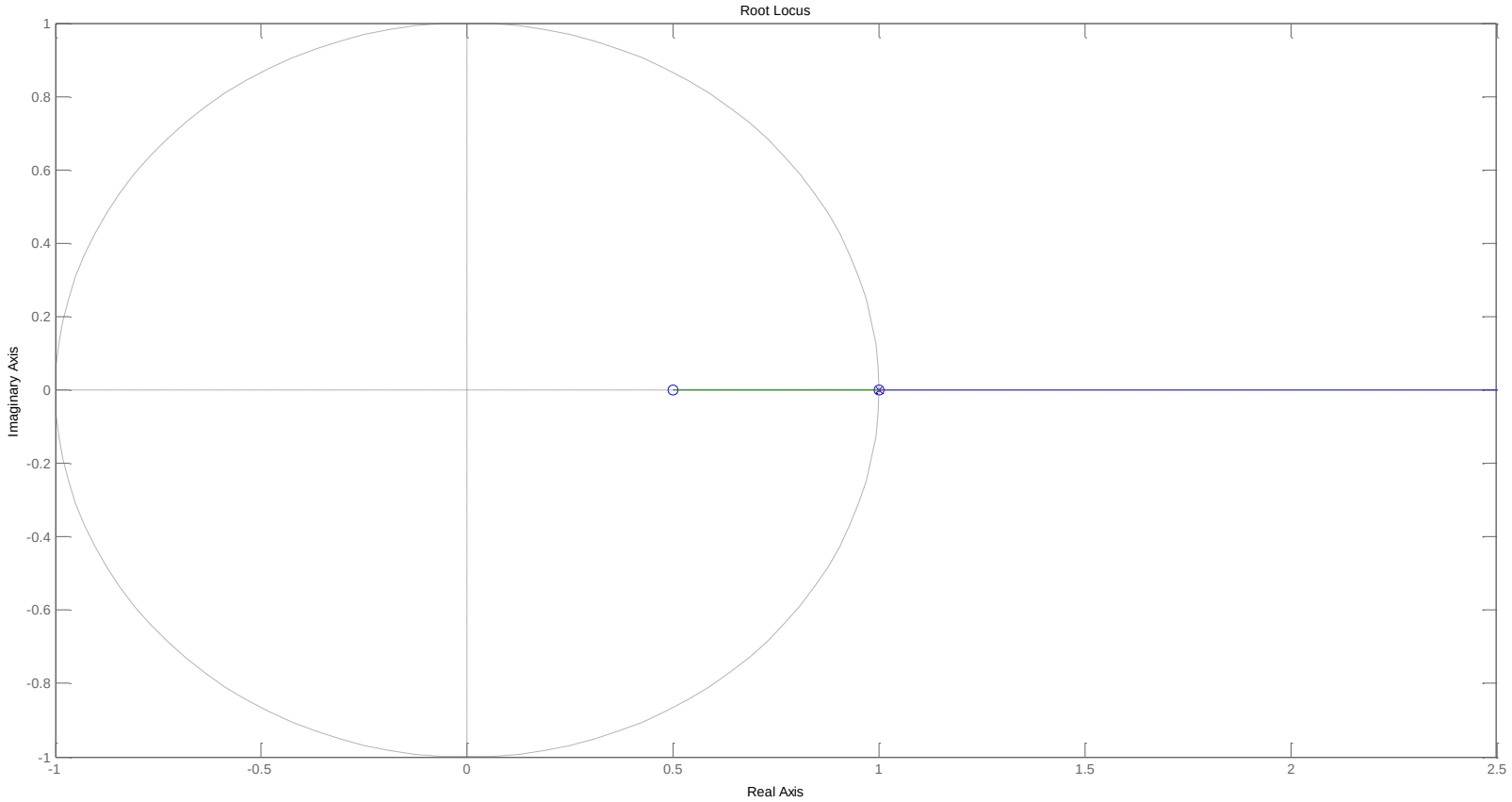
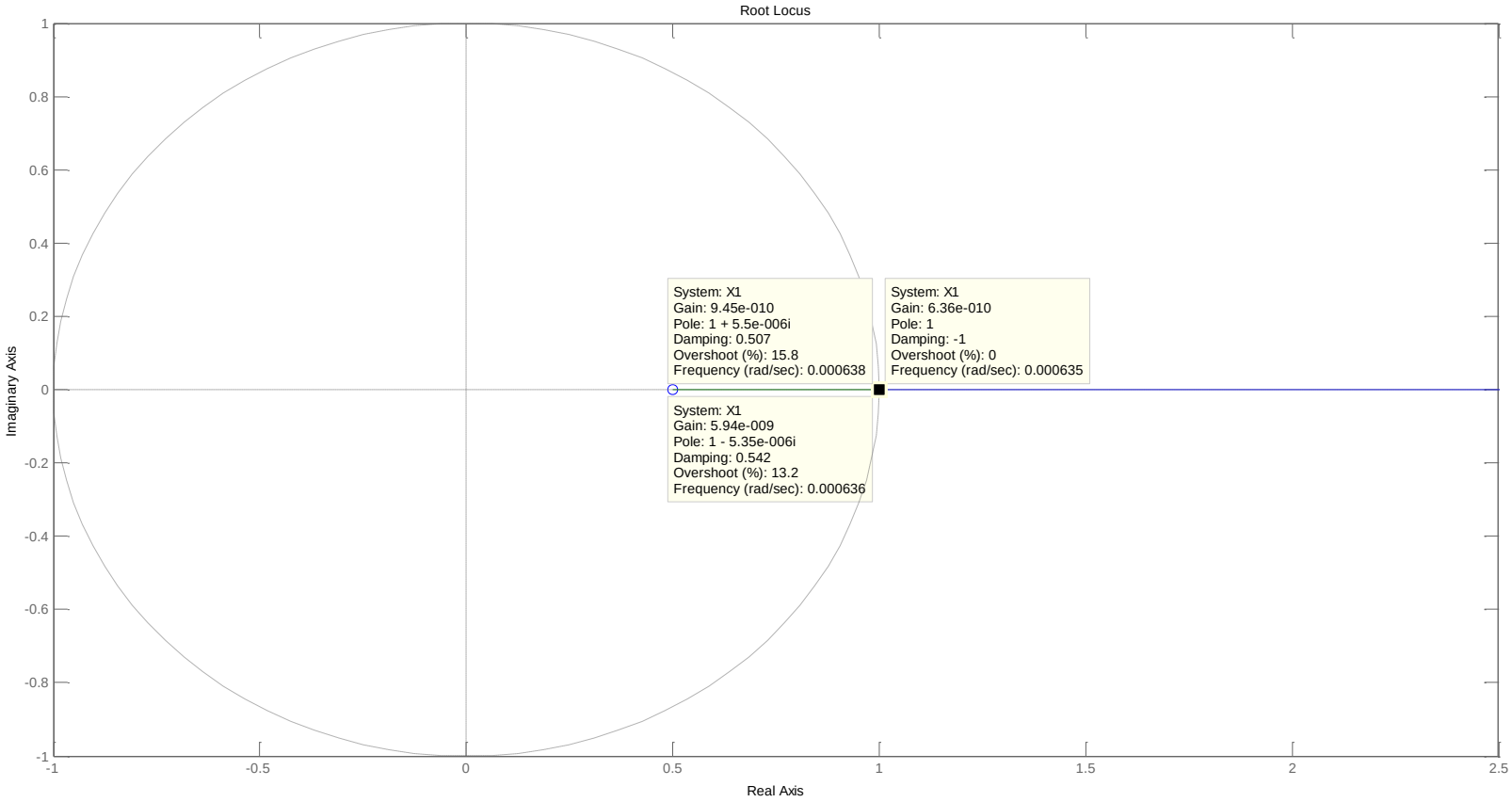


Figura 9.10 – Diagrama de pólos e zeros do *deadbeat* - microtick - Malha aberta.


Root Locus
Imaginary Axis
Real Axis
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1
1.5
2
2.5

(a)



(b)

Figura 9.11 – Diagrama de lugar das raízes do *deadbeat* - microtick (a) Sem informação de ganhos; (b) Com informação de ganhos.

b) Macrotick com MMCF=1: A Figura 9.12 e Figura 9.13 mostram o diagrama de pólos e zeros da malha fechada e malha aberta, respectivamente, do *deadbeat* sobre um macrotick com MMCF=1. Observa-se que existem 10 pólos na origem, 4 pólos fora do circulo unitário e 4 pólos dentro do circulo unitário. Na malha aberta, observa-se 3 pólos fora do circulo unitário. No entanto, os 4 pólos, em malha fechada, e os 3, em malha aberta, que estão fora do circulo e os pólos que estão dentro do circulo unitário, tem uma diferença pequena (da ordem de 1e-2 de diferença para 1) o que dificulta a analisar se este fato é devido a erros numéricos ou instabilidade do sistema. No entanto, devido à pequena diferença e dado o conhecimento das limitações das funções do Matlab usadas (pzmap, pole, zero) para analisar a estabilidade possuem limitações, demonstradas no Apêndice B, mesmo o erro sendo maior que o caso anterior (a) *Deadbeat* - Microtick), o número de pólos múltiplos também cresceu e, portanto, para este caso, considera-se que a instabilidade é devido a erros numéricos e portanto o sistema é marginalmente estável em malha aberta. A Figura 9.14 mostra o diagrama do lugar das raízes de uma função D(z)μT(z) (malha aberta do sistema com *deadbeat*). Observa-se que em malha fechada, o controle com *deadbeat*, é marginalmente estável. Observa-se, os mesmos erros numéricos da Figura 9.12 devidos as limitações demonstradas no Apêndice B.

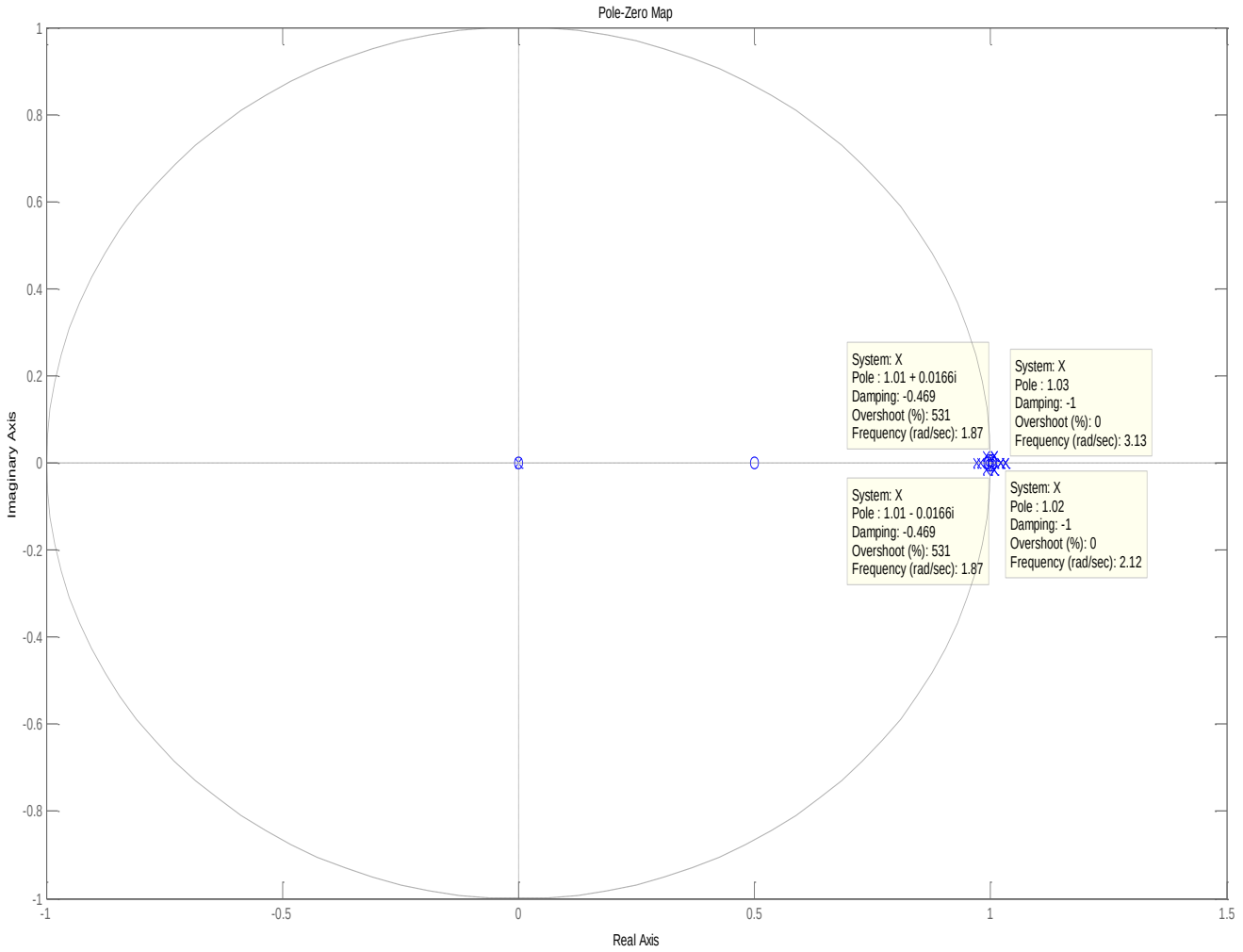


Figura 9.12 – Diagrama de pólos e zeros do *deadbeat* - macrotick - MMCF = 1 - malha fechada.

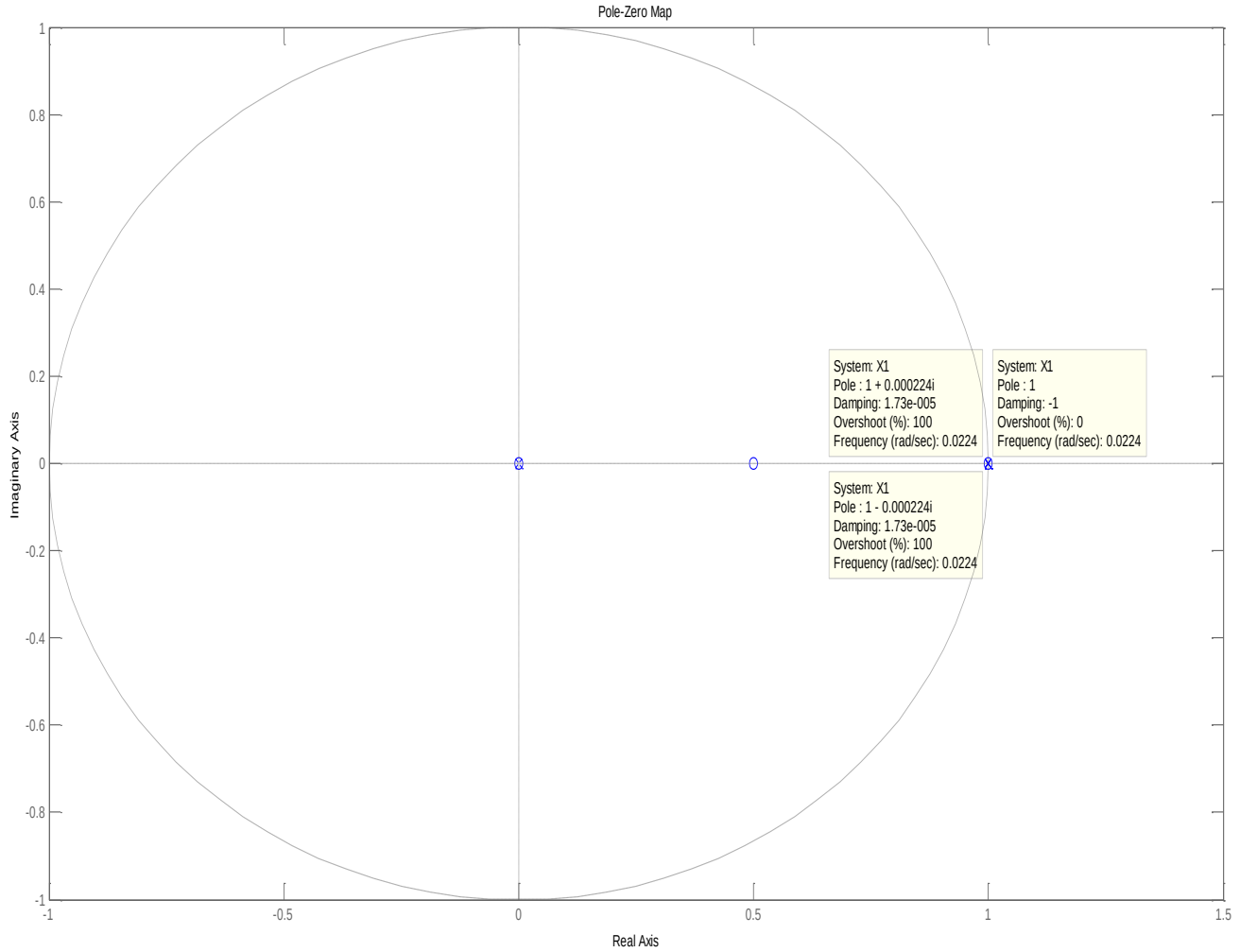
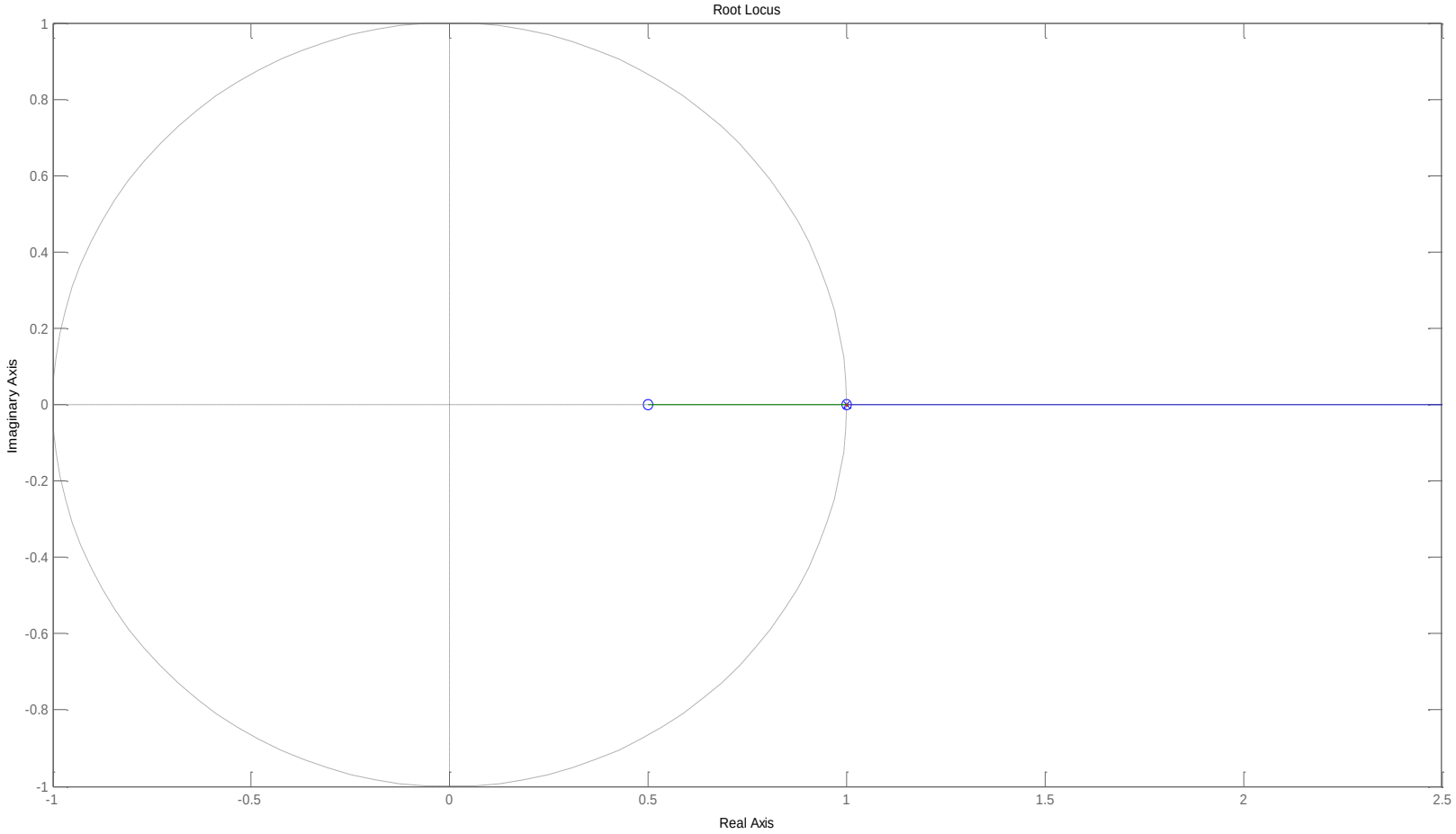
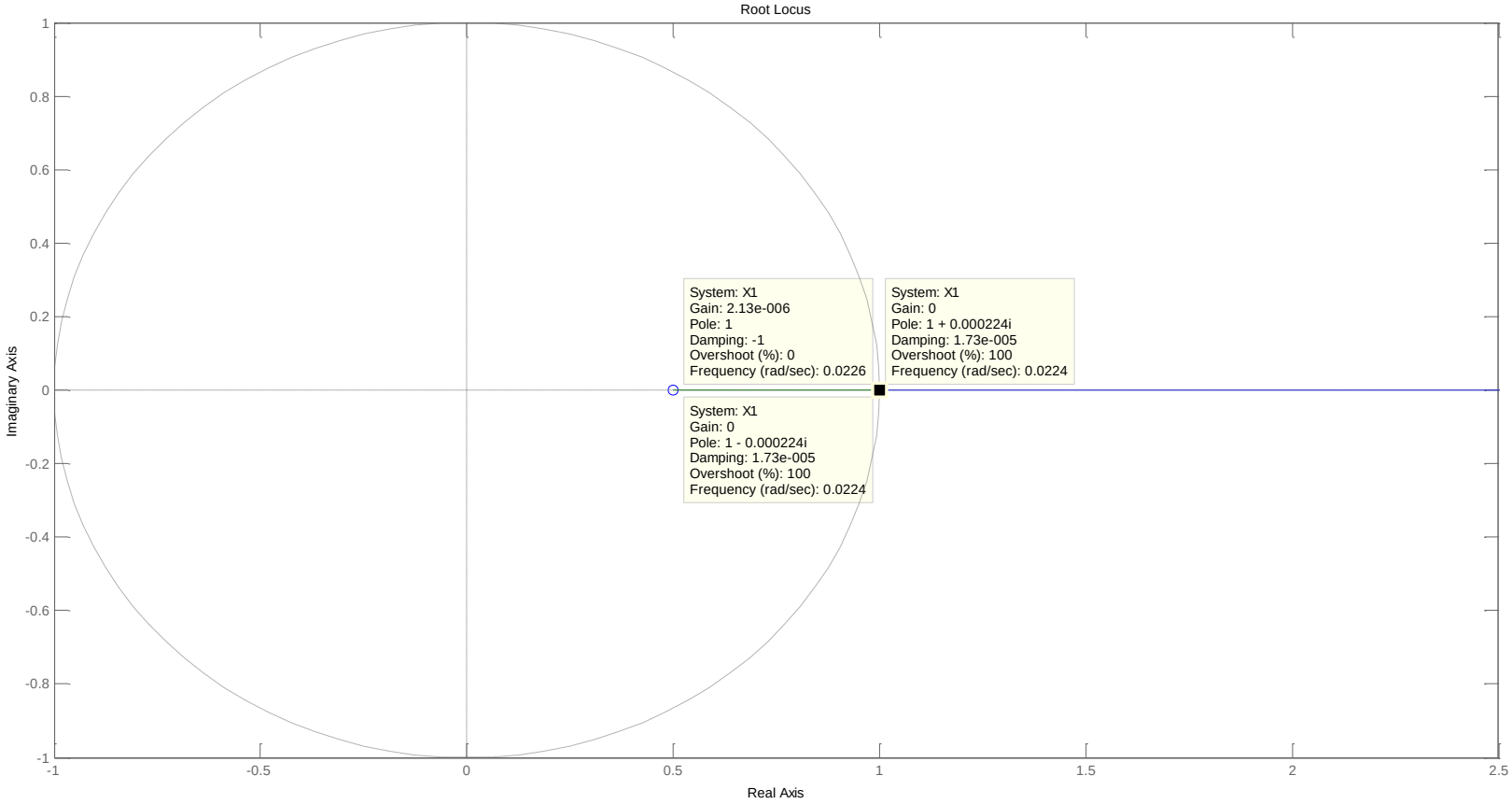


Figura 9.13 – Diagrama de pólos e zeros do *deadbeat* - macrotick - MMCF = 1 - malha aberta.


Root Locus
Imaginary Axis
Real Axis
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1
1.5
2
2.5

(a)



(b)

Figura 9.14 – Diagrama de lugar das raízes do *deadbeat* - macrotick - MMCF = 1. (a) Sem informação de ganhos. (b) Com informação de ganhos.

c) Macrotick com MMCF = 2: A Figura 9.15 e Figura 9.16 mostram o diagrama de pólos e zeros da malha fechada e malha aberta, respectivamente, do *deadbeat* sobre um macrotick com MMCF=2. Observa-se, na malha fechada, que existem pólos fora do circulo unitário, assim como o caso anterior. São 11 pólos na origem, 4 pólos fora do circulo unitário, 5 pólos dentro do circulo unitário (1 perto da origem e os outros perto do circulo unitário). Na malha aberta, observa-se 3 polos fora do circulo unitário. No entanto, ao aumentar o MMCF o erro dos pólos fora do circulo unitário também crescem, entrando no limiar do erro numérico e da estabilidade. A Figura 9.17 mostra o lugar das raízes, demonstrando a estabilidade em malha fechada. Aqui ainda se considera, tal como o caso anterior, erros devidos numéricos devido a limitações do Matlab e, portanto, o sistema é marginalmente estável em malha fechada. Era esperado que uma vez o *deadbeat* sendo projetado para uma entrada rampa e uma planta igual ao microtick que o sistema fosse instável com um Macrotick com MMCF maior que 1, pois o *deadbeat* é um controlador ótimo e portanto sensível à variação de parâmetros. Para utilizar o *deadbeat* em plantas com macrotick, o valor de deriva deve ser bem determinado. No entanto, aqui observa-se que em MMCF=2 ainda está no limiar da estabilidade. Todas as simulações anteriores deste trabalho com o *deadbeat* foram utilizando um MMCF = 1 já que no modo STM dos algoritmos ReS o *deadbeat* era aplicado a todos os instantes de microtick. Para garantir a estabilidade em malha fechada, para os valores maiores que 1 do MMCF, deve-se ter um método de estimação de deriva para evitar as instabilidades da planta.

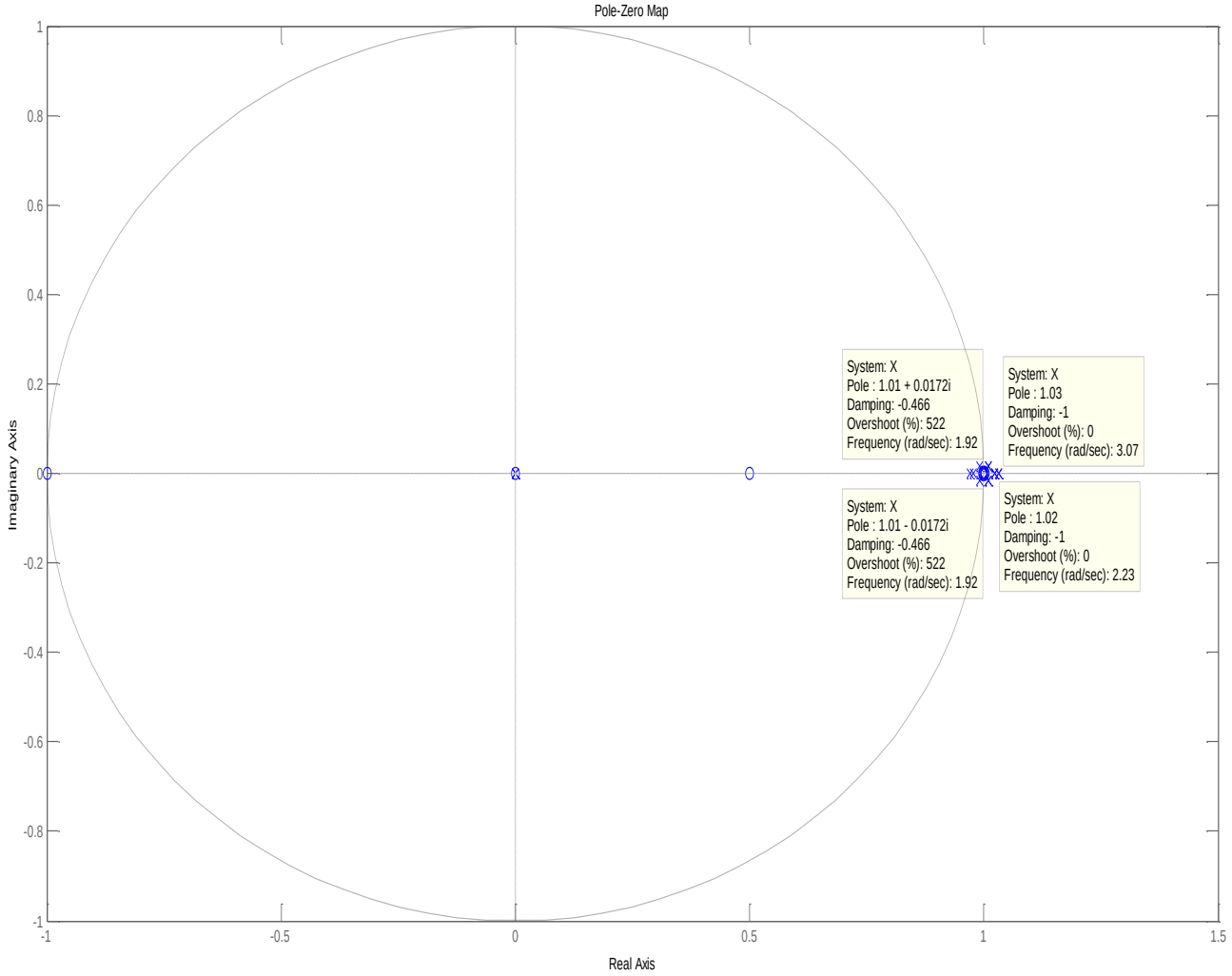


Figura 9.15 – Diagrama de pólos e zeros do *deadbeat* - MMCF = 2 - malha fechada.

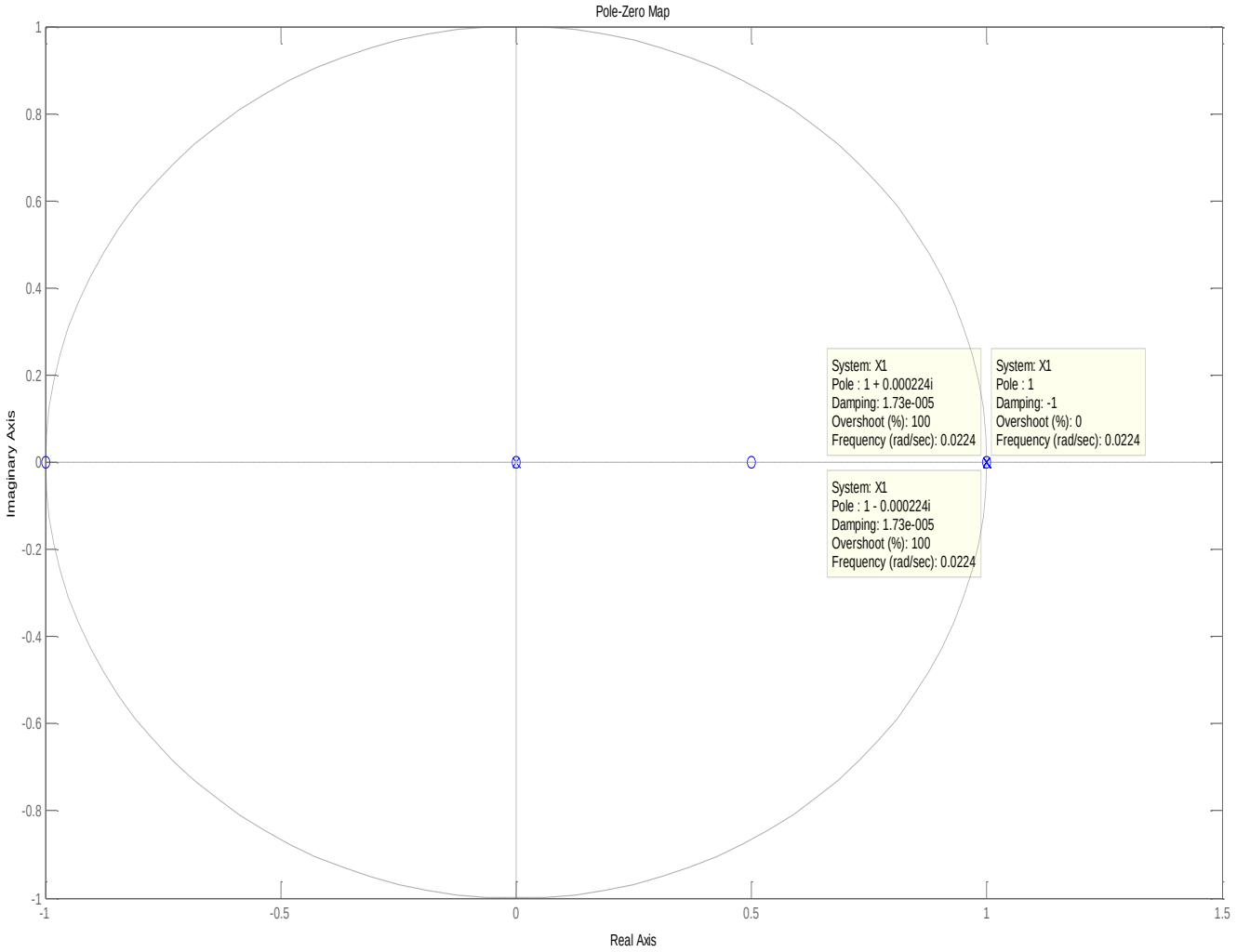
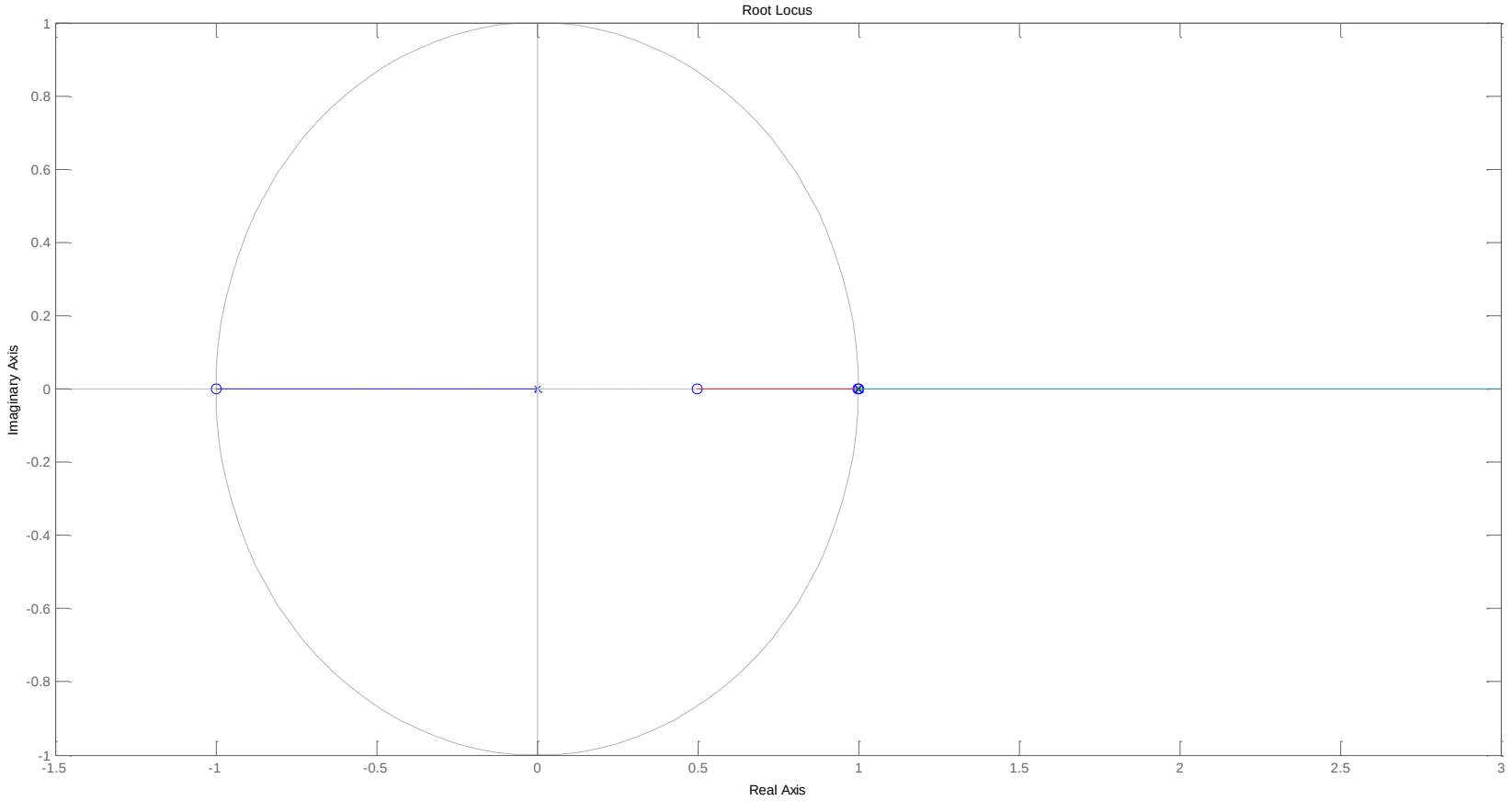
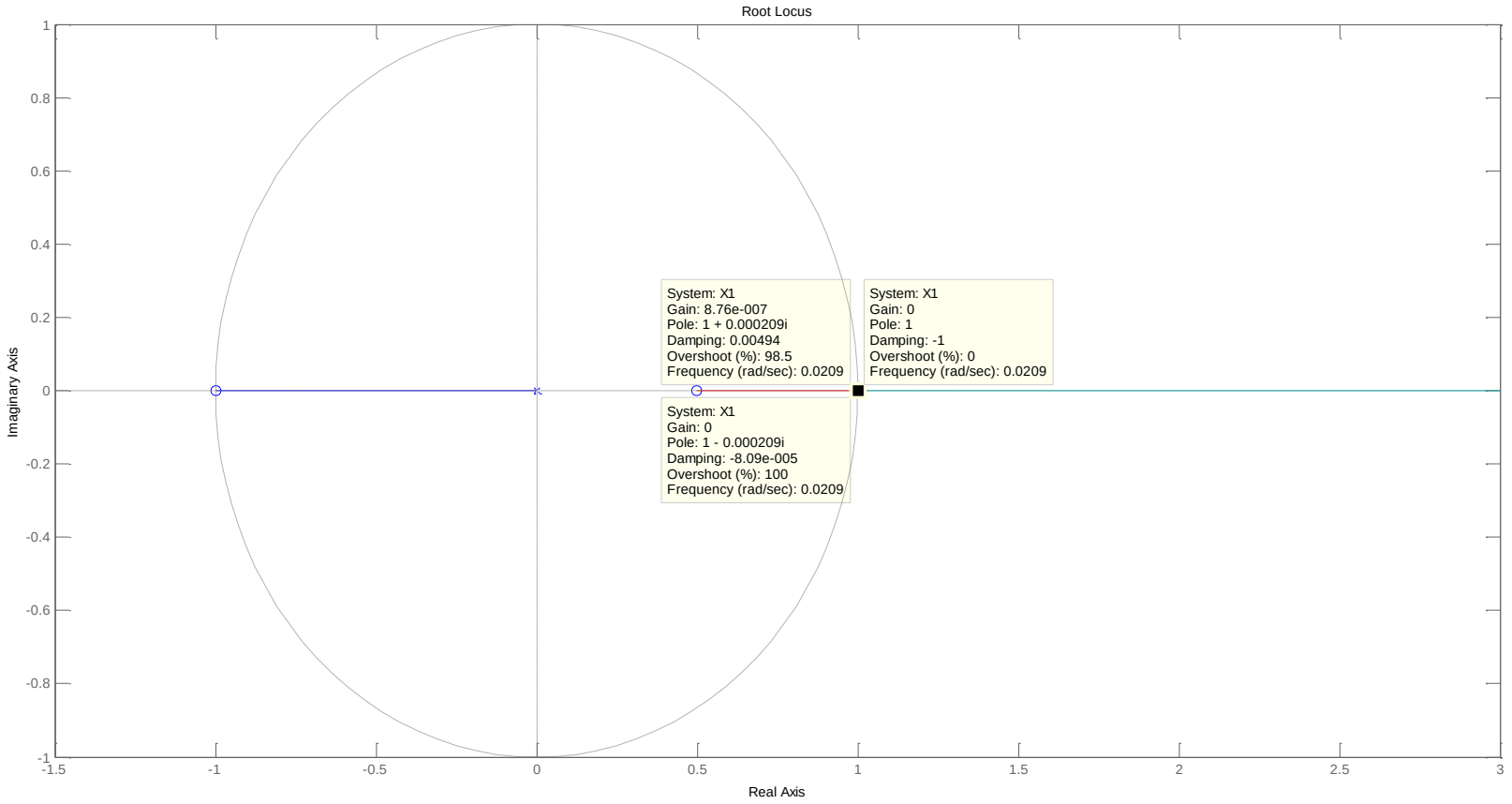


Figura 9.16 – Diagrama de pólos e zeros do *deadbeat* - MMCF = 2 - malha aberta.


Root Locus
Imaginary Axis
Real Axis
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5
3

(a)



(b)

Figura 9.17 – Diagrama de lugar das raízes do *deadbeat* - MMCF = 2. (a) Sem informação de ganhos. (b) Com informação de ganhos.

d) Macrotick com MMCF = 10: A Figura 9.18 e Figura 9.19 mostra o diagrama de pólos e zeros da malha fechada e malha aberta, respectivamente, do *deadbeat* sobre um macrotick com MMCF=10. Observa-se, na malha fechada que existem 19 pólos na origem, 1 pólo fora do circulo unitário, 4 pólos complexos conjugados fora do circulo unitário mas muito próximos ao circulo e 12 pólos dentro do circulo unitário. Portanto o sistema é instável em malha fechada. A Figura 9.20 mostra o lugar das raízes que demonstra que, realmente, existem ganhos em malha fechada que instabilizam a planta. Isto era esperado, uma vez que o *deadbeat* foi projetado para uma entrada rampa e uma planta igual ao microtick. O *deadbeat* é um controlador ótimo e portanto sensível à variação de parâmetros. Para utilizar o *deadbeat* em plantas com macrotick, o valor de deriva e MMCF devem ser bem determinados. Como observado no caso anterior, para garantir a estabilidade em malha fechada, para os valores maiores que 1 do MMCF, deve-se ter um método de estimação de deriva para evitar as instabilidades da planta.

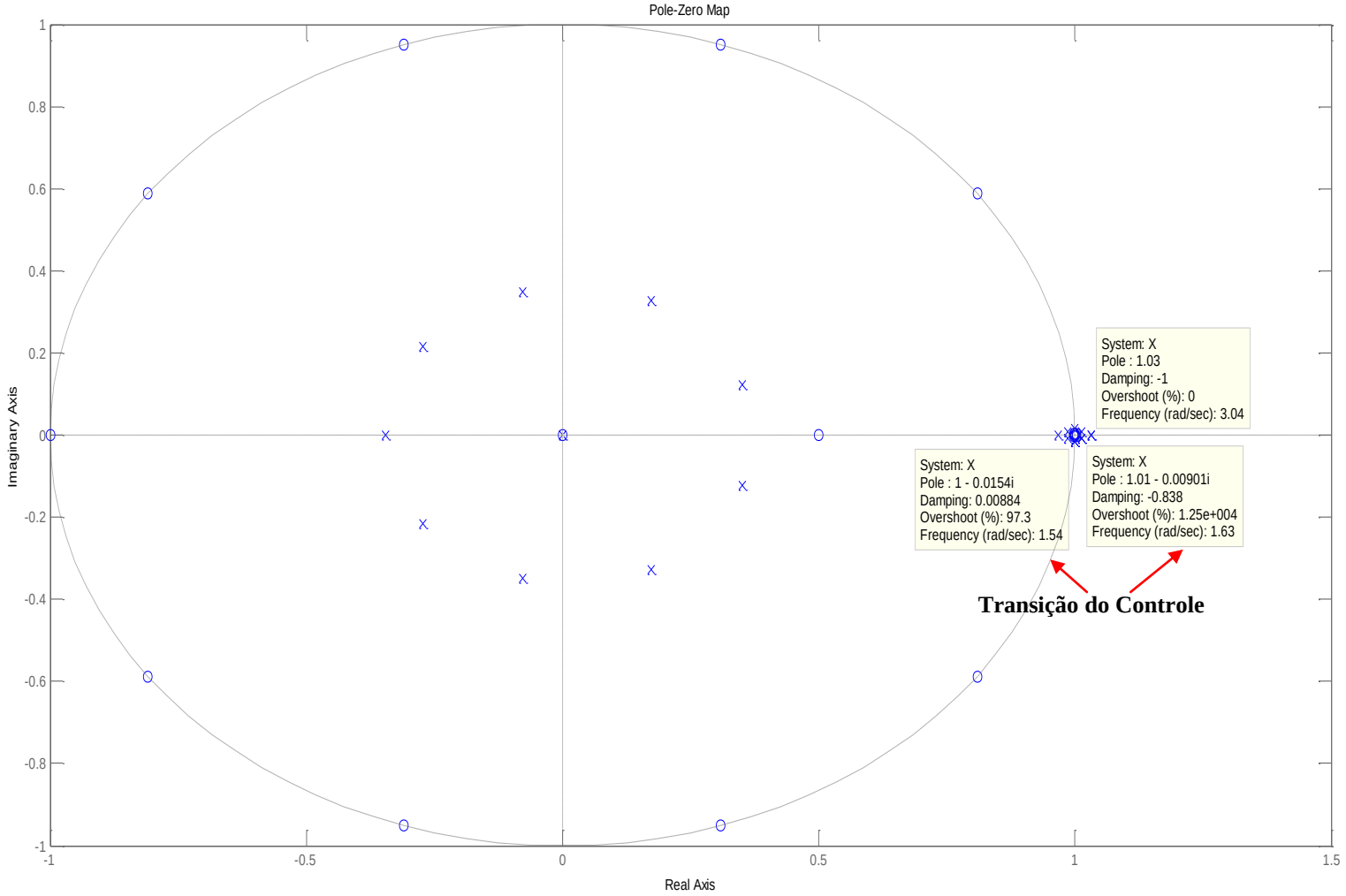


Figura 9.18 – Diagrama de pólos e zeros do *deadbeat* - Macrotick - MMCF = 10 - malha fechada.

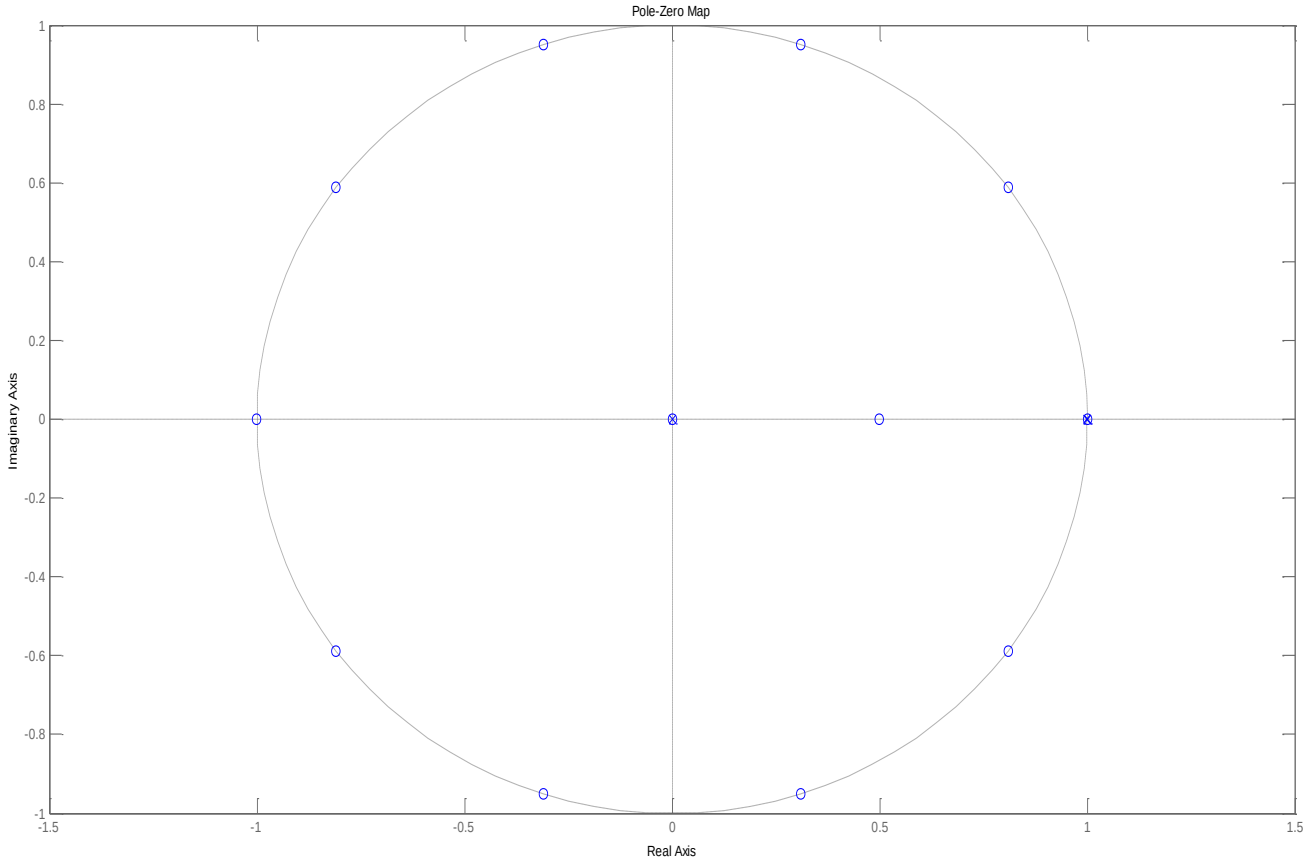


Figura 9.19 – Diagrama de pólos e zeros do *deadbeat* - Macrotick - MMCF = 10 - malha aberta.

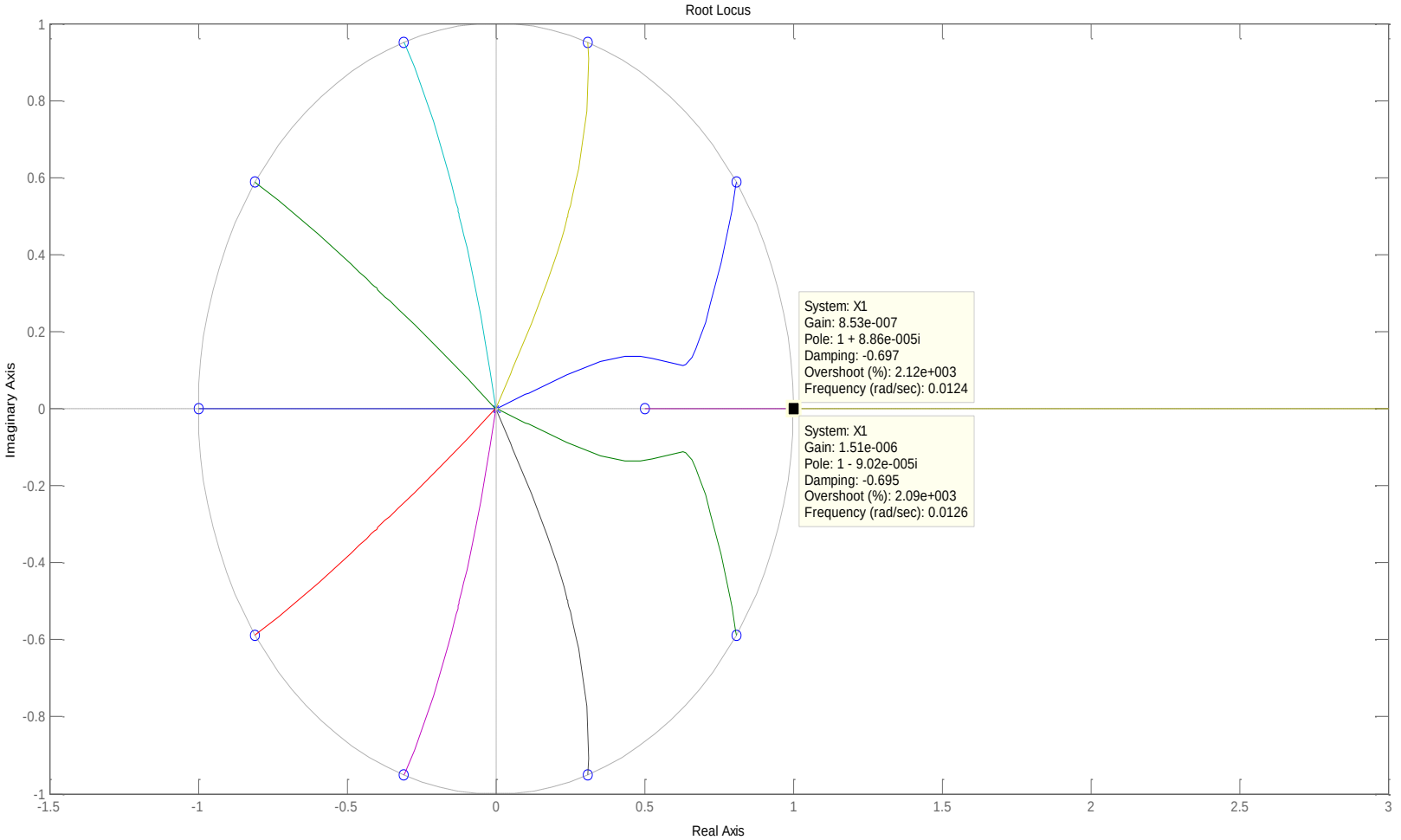


Figura 9.20 – Diagrama de lugar das raízes do *deadbeat* - Macrotick - MMCF = 10.

9.4. Análise de Estabilidade do Modo NOMO

Os algoritmos ReS em seu modo NOMO, utilizam-se de 2 tipos de função de convergência. No entanto, observa-se que ambas as equações de convergência, a grosso modo, podem ser consideradas como uma equação de média com 2 medidas de relógios, com a diferença que na abordagem centralizada se utiliza 2 relógios (1 de referência) e quatro medidas para se definir a diferença de tempo e na abordagem distribuída utilizam-se 4 relógios (não existe referência real), mas devido ao teorema dos generais bizantinos, dois relógios são descartados. Portanto, para analisar a estabilidade, esta tese faz a análise utilizando uma função de correção de média de 2 macroticks. A equação de correção portanto fica:

$$corr(k) = \frac{[x1(k) - x1(k-1)] + [x1(k) - x2(k-1)]}{2} \tag{9.5}$$

Aplicando a transformada Z à equação (9.5), tem-se:

$$CORR(z) = \frac{X1(z)[2 - z^{-1}] - z^{-1}X2(z)}{2} \tag{9.6}$$

Sendo X1(z) e X2(z) conhecidas, como microtick e/ou macrotick. Sendo o intuito corrigir X1 em relação a X2, tem-se portanto (utilizando-se o teorema de linearidade) a seguinte função em transformada Z:

$$F(z) = X1(z) - CORR(z) \tag{9.7}$$

Assim, com a equação (9.7), as seguintes análises de estabilidade através do diagrama de pólos e zero, considerando F(z) em malha aberta; e o diagrama de lugar das raízes, considerando o sistema D(z)*F(z), em que D(z) é o ganho de controle que fecha a malha do sistema.

As seguintes análise são realizadas:

a) **Análise 1:** X1 e X2 são funções microtick com os parâmetros ρ1 = 0.01, ρ2 = 0.0001 e T = 0.01, em que, respectivamente, ρ1 e ρ2 são as derivas das funções X1 e X2 e T o período de amostragem. A Figura 9.21 mostra o diagrama de pólos e zeros do sistema em questão. Observa-se que o sistema possui 2 pólos na origem e 3 pólos sob o circulo unitário. Portanto o sistema é marginalmente estável em malha aberta. A Figura 9.22 mostra o lugar das raízes do sistema em questão. Observa-se que, em caso de fechamento de malha, o ganho para se levar o sistema à instabilidade é muito alto, o que é fisicamente não realizável. Portanto, confirma-se também o sistema marginalmente estável em malha fechada.

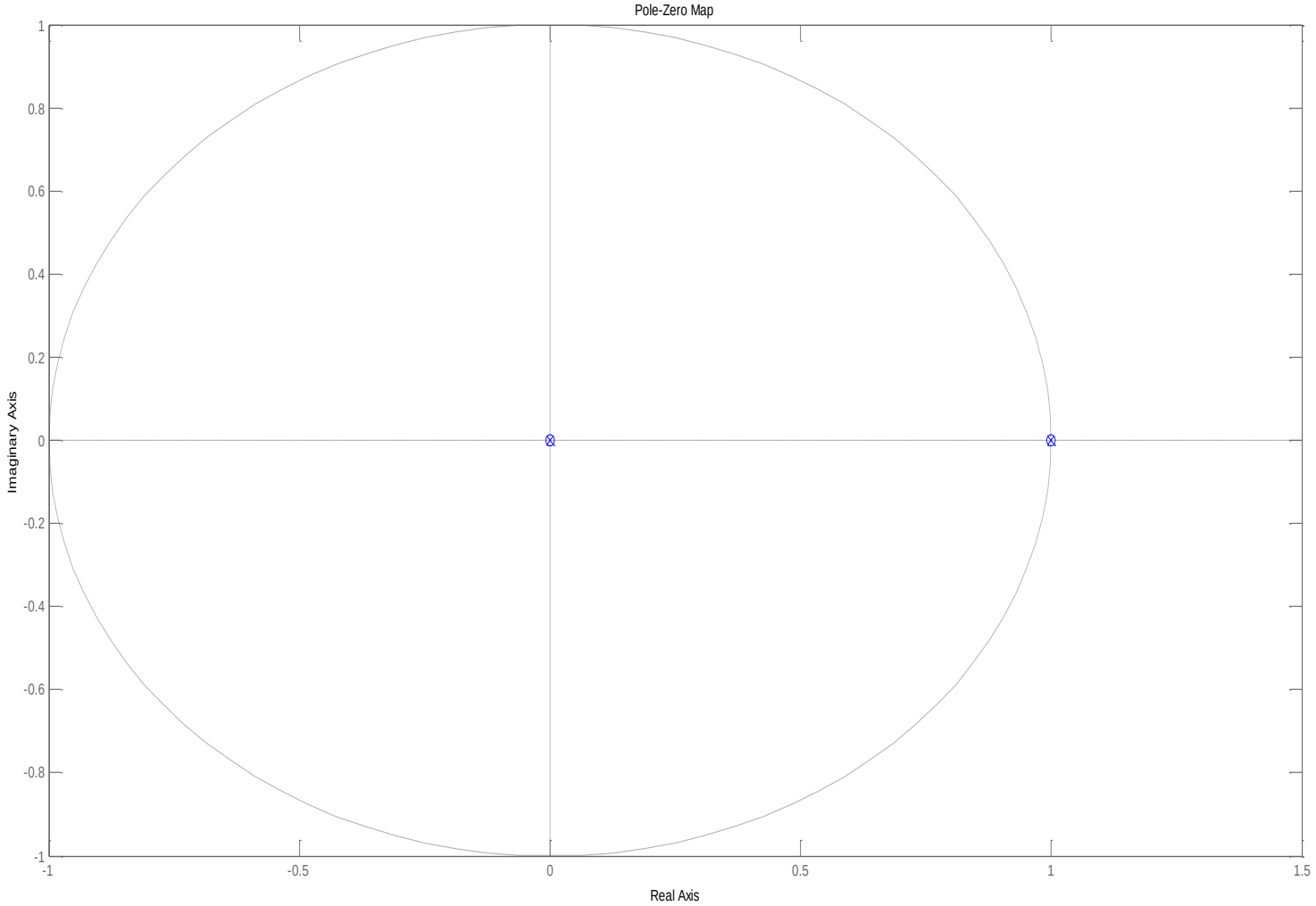


Figura 9.21 – Diagrama de pólos e zeros do modo NOMO - Microtick.

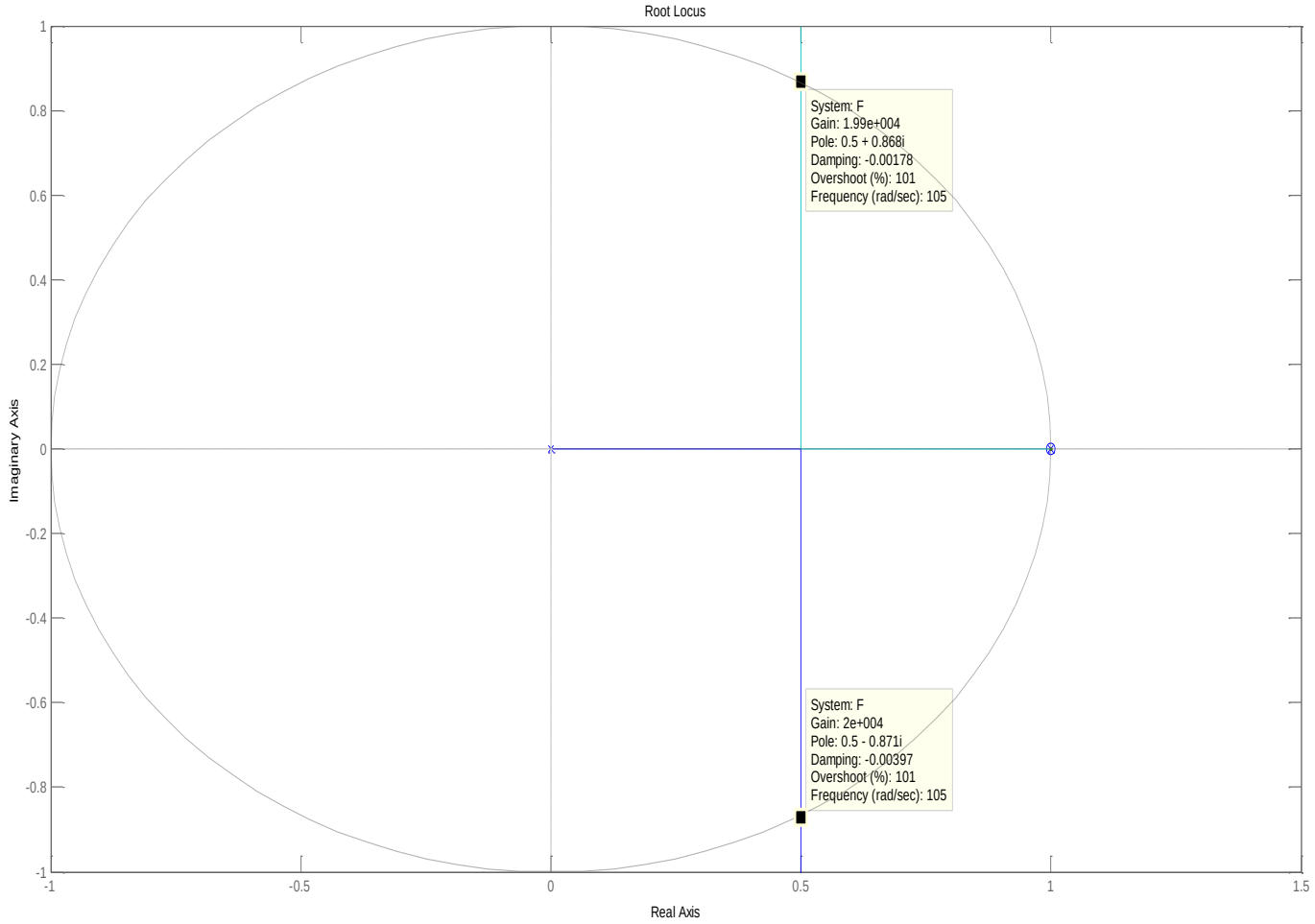


Figura 9.22 – Diagrama de lugar das raízes do modo NOMO - Microtick.

b) **Análise 2:** X1 e X2 são funções macrotick com os parâmetros ρ1 = 0.01, ρ2 = 0.0001, MMCF1 =1, MMCF2=1, T = 0.01, em que, respectivamente, ρ1 e ρ2 são as derivas das funções X1 e X2, MMCF1 e MMCF2 são os fatores de conversão *microtick-macrotick* e T o período de amostragem. A Figura 9.23 mostra o diagrama de pólos e zeros do sistema em questão. Observa-se que o sistema possui 5 pólos na origem e 6 pólos perto de 1. Alguns pólos estão fora do circulo unitário (3 pólos). No entanto, é conhecido que a funções do Matlab usadas (pzmap, pole, zeros) para analisar a estabilidade possuem limitações, como demonstradas no Apêndice B. Portanto, para este caso, devido ao erro em relação a 1 dos pólos serem muito pequenos (dentro do limite de erro), considera-se que a instabilidade é devido a erros numéricos e portanto o sistema é marginalmente estável em malha aberta. A Figura 9.24 mostra o lugar das raízes do sistema em questão. Assim como o caso anterior, observa-se, sendo os pólos de fora do circulo como erros numéricos, que o sistema também é marginalmente estável em malha fechada.

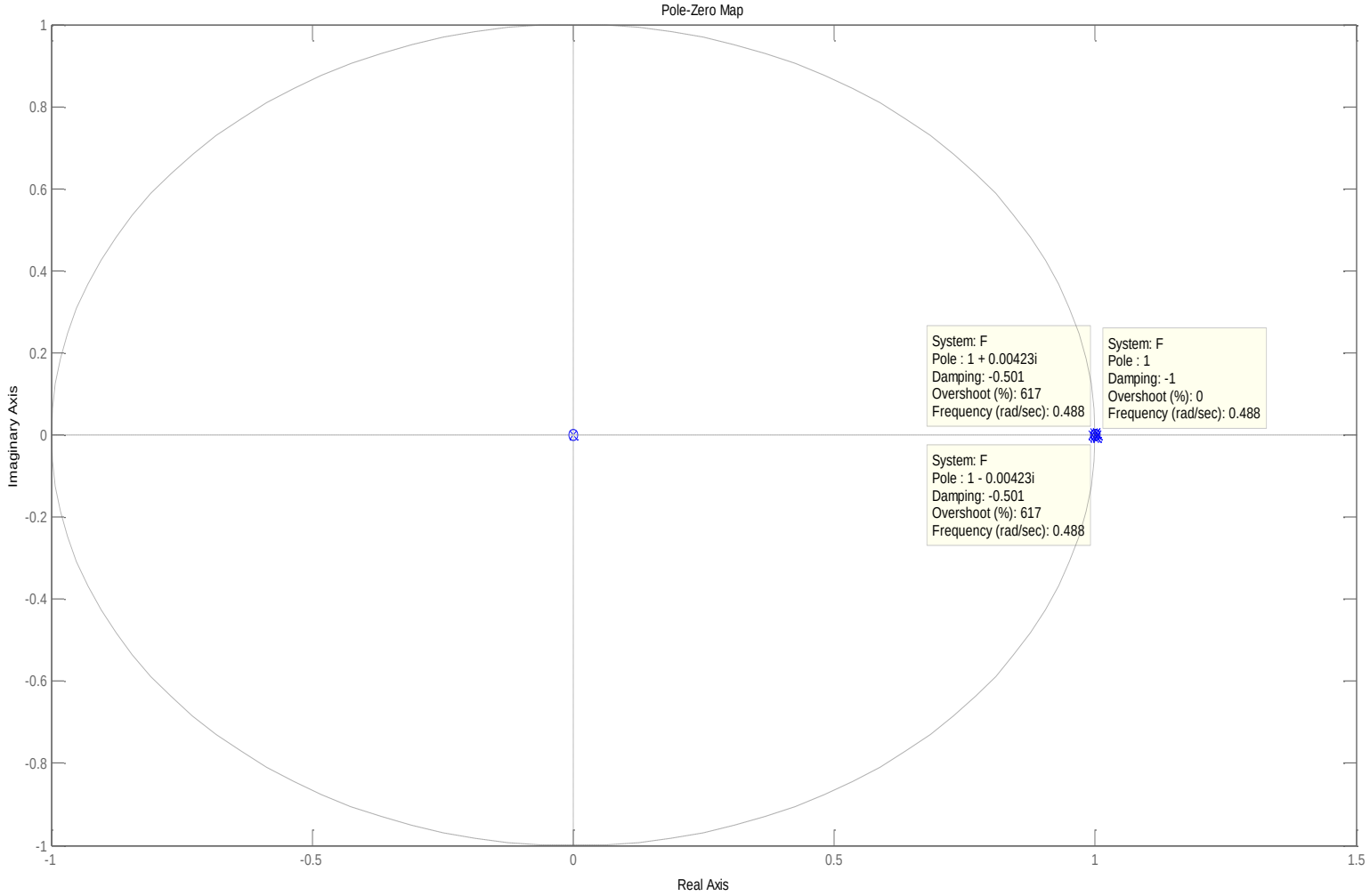


Figura 9.23 – Diagrama de pólos e zeros do modo NOMO - Macrotick - MMCF1 = 1 e MMCF2 = 1.

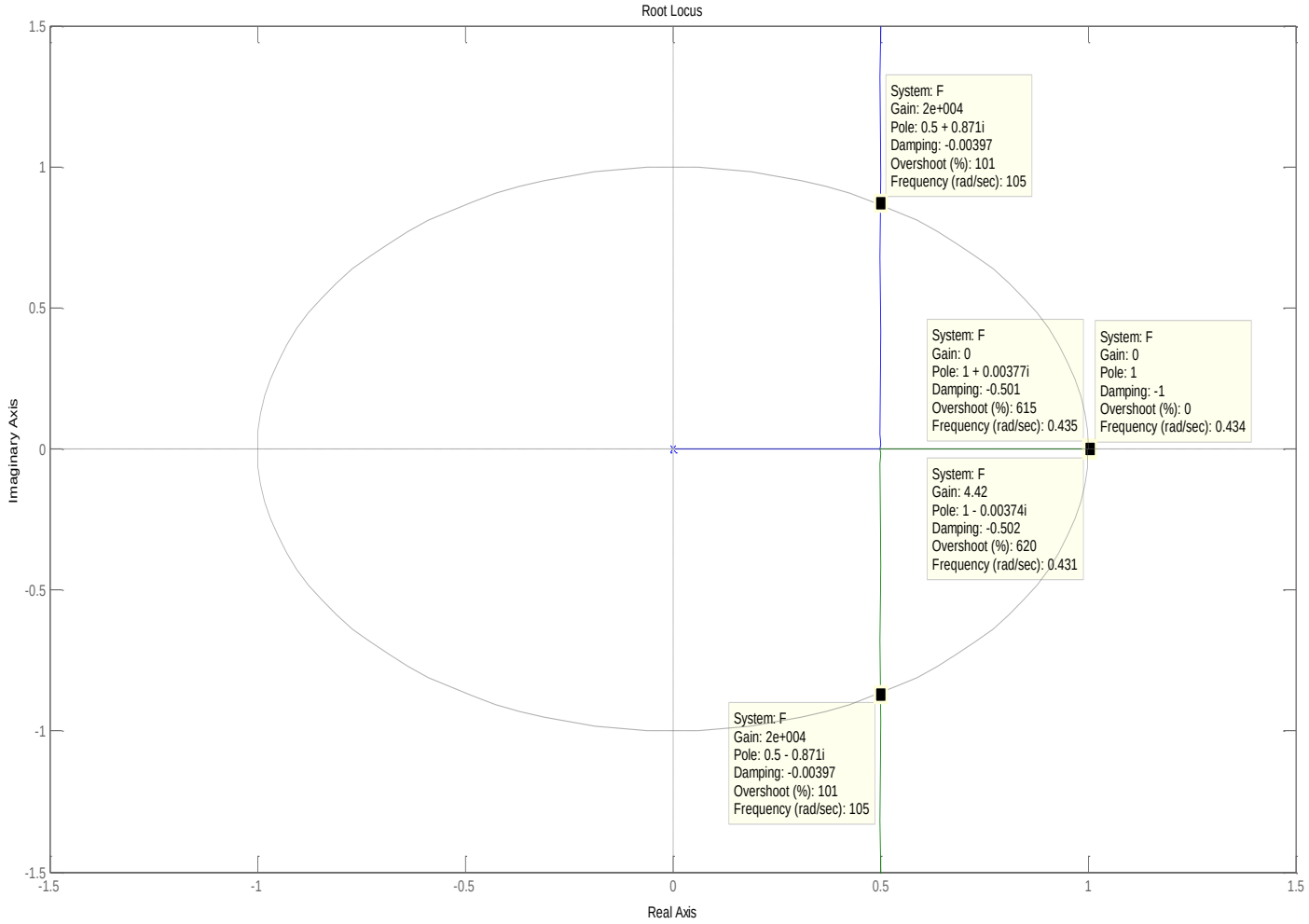


Figura 9.24 – Diagrama de lugar das raízes do modo NOMO - Macrotick - MMCF1 = 1 e MMCF2 = 1.

c) **Análise 3:** X1 e X2 são funções macrotick com os parâmetros ρ1 = 0.01, ρ2 = 0.0001, MMCF1 =1, MMCF2=2, T = 0.01. A Figura 9.25 mostra o diagrama de pólos e zeros do sistema em questão. Observa-se que o sistema possui 6 pólos na origem e 6 pólos perto de 1. Alguns pólos estão fora do circulo unitário (3 pólos). No entanto, assim como nos casos anteriores, os pólos que estão fora do circulo são devido a erros numéricos, como demonstrado no Apêndice B. Portanto, consideraremos que o sistema é marginalmente estável em malha aberta. A Figura 9.26 mostra o diagrama de lugar das raízes do sistema em questão. Também, devido a erros numéricos, possui pólos fora do circulo unitário. Portanto, o sistema é marginalmente estável em malha fechada.

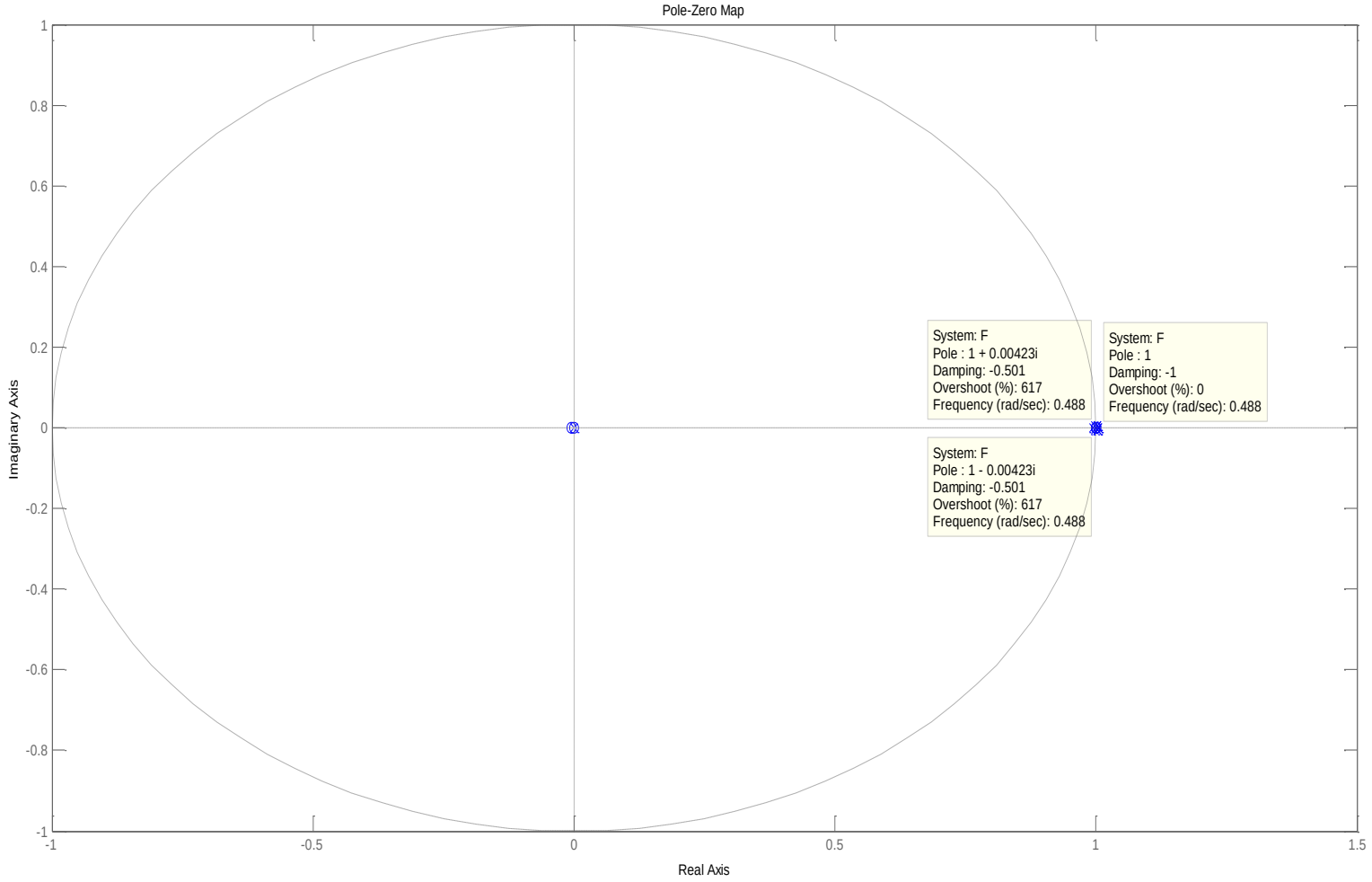


Figura 9.25 – Diagrama de pólos e zeros do modo NOMO - Macrotick - MMCF1 = 1 e MMCF2 = 2.

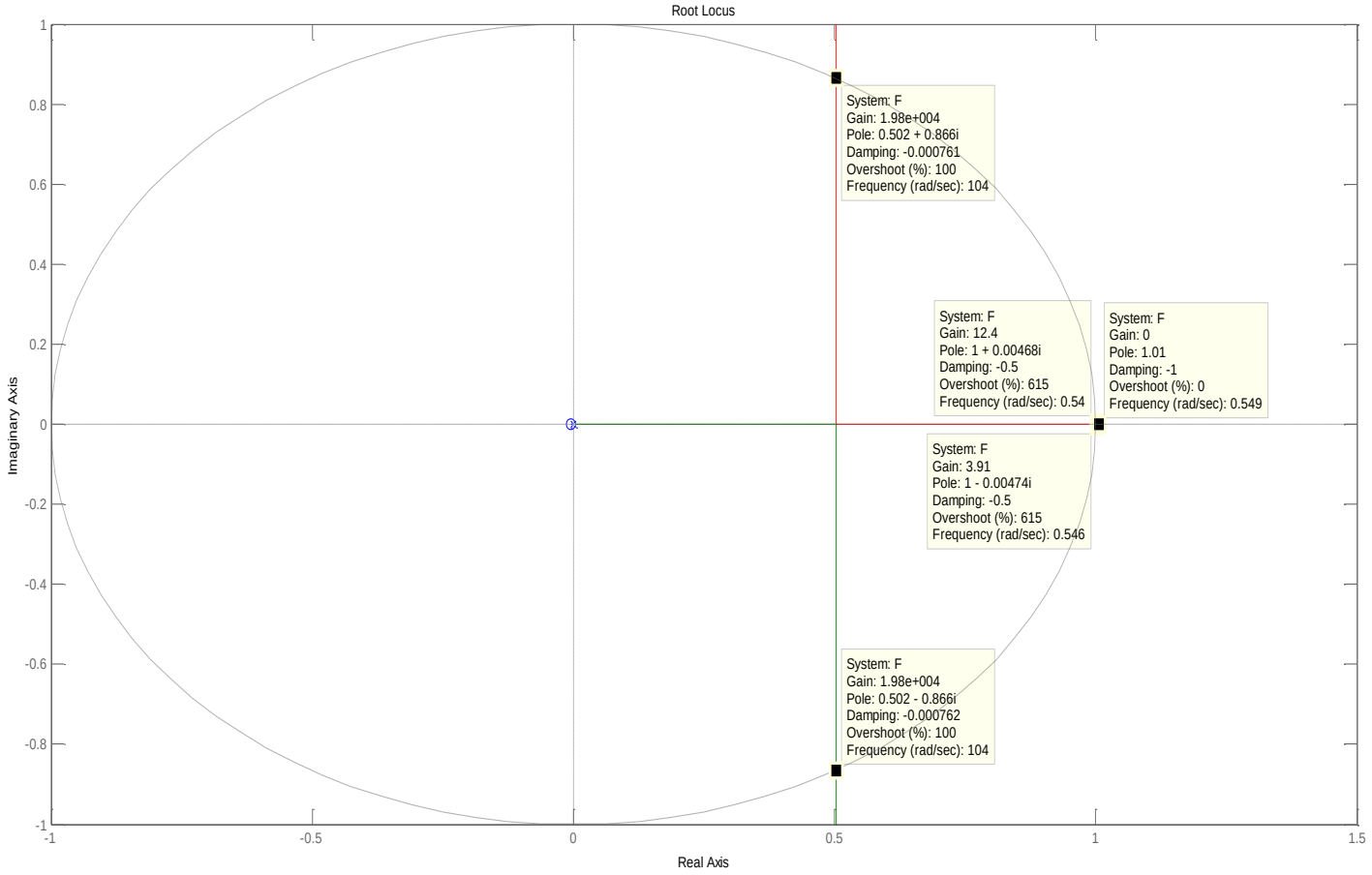


Figura 9.26 – Diagrama de lugar das raízes do modo NOMO - Macrotick - MMCF1 = 1 e MMCF2 = 2.

d) **Análise 4:** X1 e X2 são funções macrotick com os parâmetros ρ1 = 0.01, ρ2 = 0.0001, MMCF1 =2, MMCF2=2, T = 0.01. A Figura 9.27 mostra o diagrama de pólos e zeros do sistema em questão. Observa-se que o sistema possui 8 pólos na origem e 6 pólos perto de 1. Alguns pólos estão fora do circulo unitário (3 pólos). No entanto, assim como nos casos anteriores, os pólos que estão fora do circulo são devido a erros numéricos, demonstrado no Apêndice B. Portanto, consideraremos que o sistema é marginalmente estável em malha aberta. A Figura 9.28 mostra o lugar das raízes. Observa-se que o sistema é marginalmente estável em malha fechada. No entanto, é importante destacar dois pontos: 1) Com o aumento do MMCF1 e MMCF2 o sistema tende a se tornar mais suscetível a instabilidade em malha fechada; 2) Ao adicionar o termo de Corr(z), o sistema se tornou mais robusto em relação à estabilidade. Este termo se refere ao algoritmo de sincronização de relógios. Isto era esperado, já que o algoritmo opera em malha aberta e torna o sistema estável em malha aberta. Já em malha fechada, indica a aplicação de um controle que neste caso é representado pela função amortizadora. O que indica que a função amortizadora (cross-fading) pode instabilizar o sistema se não for bem projetada e devido a MMCF´s e períodos de amostragem grandes.

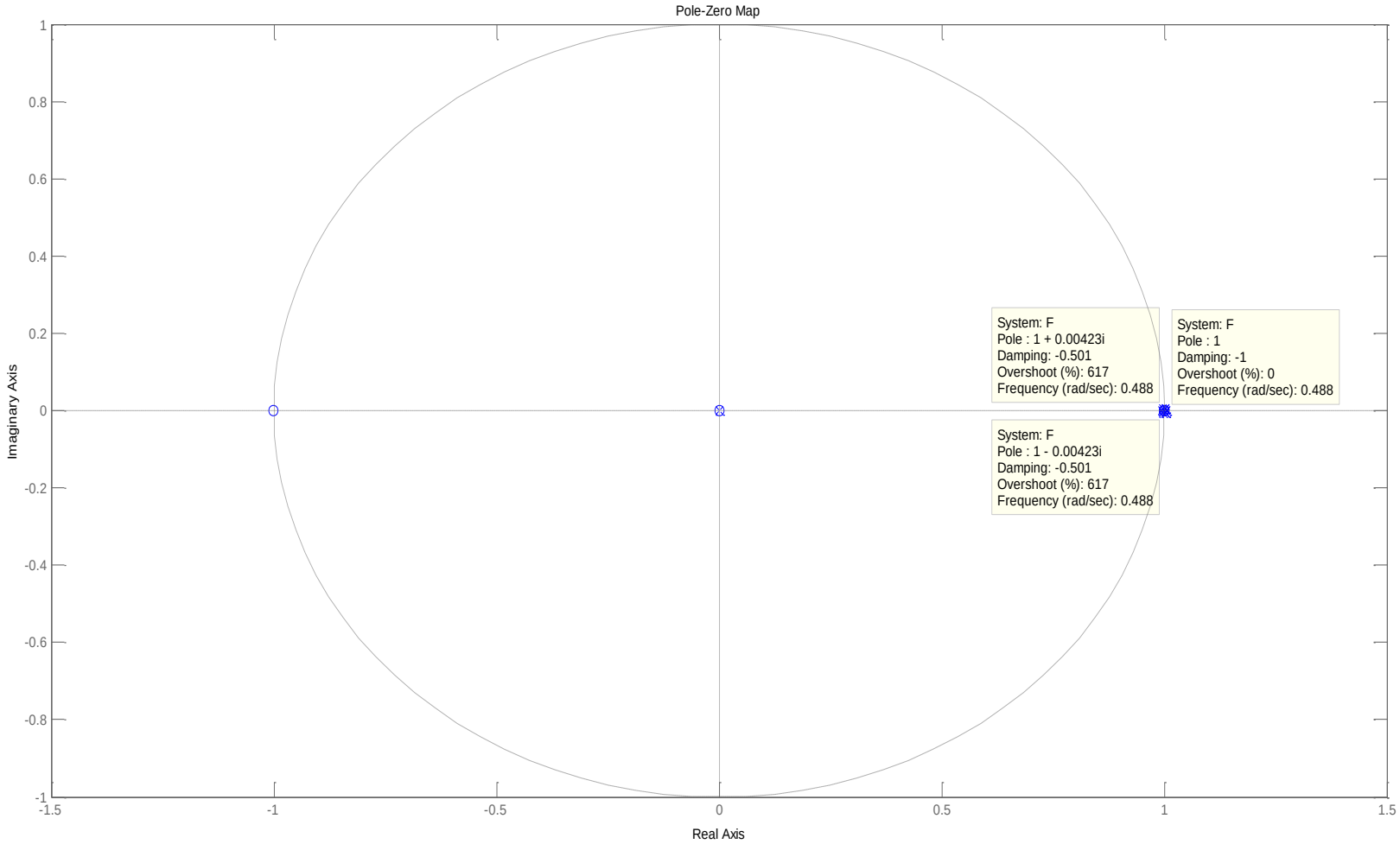


Figura 9.27 – Diagrama de pólos e zeros do modo NOMO - Macrotick - MMCF1 = 2 e MMCF2 = 2.

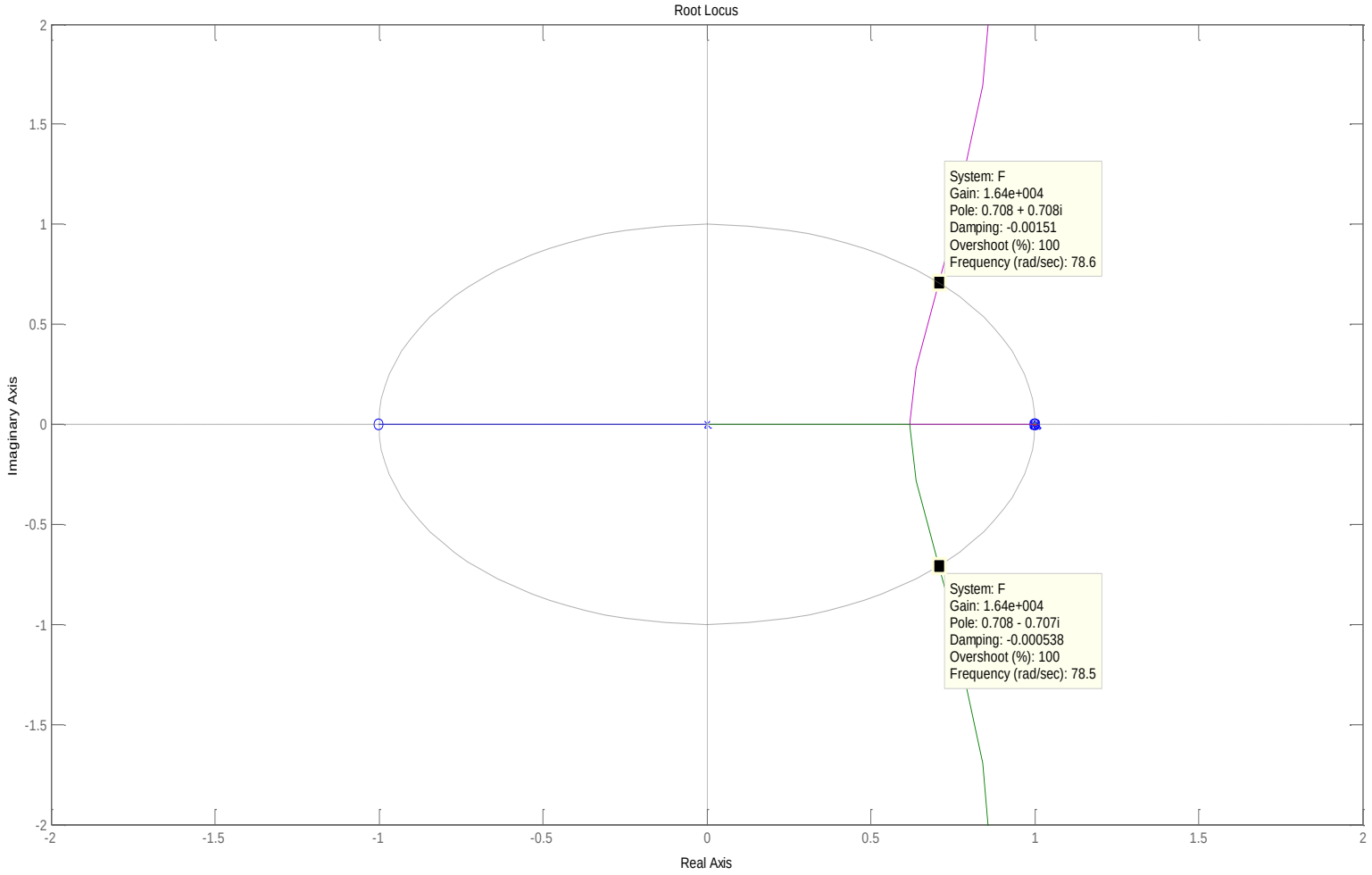


Figura 9.28 – Diagrama lugar das raízes do modo NOMO - Macrotick - MMCF1 = 2 e MMCF2 = 2.

9.5. Estabilidade modo NOMO com Função Amortizadora

No modo NOMO, também se implementam as funções amortizadoras (*cross-fading)* multiplicando-se uma função u(k) ao valor de correção. Portanto, a equação (9.7) fica:

$$F(z) = X1(z) - \ CORR(z) * U(z) \tag{9.8}$$

Em que * indica a convolução de U(z) com a função de correção CORR(z).

As funções utilizadas para amortização (*cross-fading*) neste trabalho foram: 1) Degrau; 2) Rampa; 3) Exponencial.

A função degrau foi modelada como uma constante. Sendo uma constante, e portanto um escalar, o teorema da convolução não é necessário e assim utilizando a propriedade da linearidade, tem-se:

$$F(z) = X1(z) - u\ CORR(z) \tag{9.9}$$

Em que u é o escalar constante do degrau. Assim, o degrau não vai introduzir nenhum pólo que possa instabilizar o sistema e portanto o sistema continua com as mesmas condições de estabilidade da seção 9.4, alterando somente os zeros.

A função rampa cresce indefinidamente e portanto para fins práticos sua análise não é útil, pois não reflete a realidade. No entanto a função rampa pode afetar os pólos e portanto afetar a estabilidade do sistema.

A função exponencial converge a um valor, portanto para fins práticos sua análise algébrica é útil e tem uma maior fidelidade com a realidade. No entanto a função exponencial pode afetar os pólos e portanto afetar a estabilidade do sistema.

No entanto, as análises de lugar das raízes da Seção 9.4, demonstram de modo geral como as funções amortizadoras, quando tratadas como um controlador de malha fechada, podem influenciar a estabilidade do sistema.

9.6. Análise de Estabilidade do Modo NOMD

O modo NOMD não possui funções. O modo NOMD nos algoritmos ReS é um simples ajuste estático do parâmetro MMCF. A mudança do MMCF afeta a estabilidade do sistema, como foi observado anteriormente. E portanto, não há motivos para fazer a análise da estabilidade do ajuste do MMCF.

10 CONCLUSÕES

Uma arquitetura de sistemas de controle por rede integra elementos de pelo menos três domínios de conhecimento, tais como controle, comunicação e computação. Estes sistemas que envolvem computação, comunicação e controle, chamados atualmente de sistemas cibernético-físicos (em inglês, Cyber Physical Systems - CPS), possuem funcionalidades básicas tais como agendadores de tarefas, gerenciadores de tempo, serviço de comunicação e outros que juntos formam um conjunto de serviços essenciais para que o sistema de controle tenha uma correta operação durante o seu ciclo de vida. Estes serviços tornam-se mais especiais em sistemas espaciais, como satélites e robôs, pois todo o sistema opera continuamente durante um ciclo de vida extenso sob um ambiente extremamente agressivo. A implementação distribuída, a integração de sistemas, a complexidade e o aumento do requisito de confiabilidade tornam relevantes diversos problemas que antes não o eram, demandando o desenvolvimento de novas tecnologias, técnicas e algoritmos. Dentre estes problemas está à sincronização de tempo entre diversas entidades de uma implementação distribuída e como estas soluções influenciam o desempenho de sistemas de controle por rede. Esta tese estudou os efeitos da sincronização de relógios sobre o transitório e a estabilidade de sistemas de controle por rede. O trabalho propôs um conjunto de algoritmos de sincronização com métrica mista (Precisão, Exatidão e Controle) corrigindo os vieses iniciais, os vieses instantâneos causados pela deriva dos relógios locais utilizando técnicas de amortização das descontinuidades de tempo durante o ajuste dos relógios. Também analisou e validou o conjunto de algoritmos de sincronização de relógios sobre um NCS comparando os resultados com os algoritmos FTM, PTP e SR. Tudo, utilizando teoria, análise (quando aplicável), controle e simulação como abordagens.

Os algoritmos FTM, PTP e SR provêem boas técnicas para alcançar a sincronização de relógios, cada um seguindo uma métrica, dentro de uma tolerância. Mesmo sob grandes derivas ou falhas de um relógio é possível chegar a uma sincronização. De fato, em caso de falhas, a precisão/exatidão da sincronização é degradada, mas sincronizada. Contudo, os efeitos de descontinuidade de tempo afetam a malha de controle conectada pela rede, como mostraram alguns casos simulados na seção 7.1. Ao corrigir o tempo, fazendo a volta ou o avanço no tempo, o algoritmo de sincronização faz com que todo o contexto temporal se reajuste, modificando assim períodos de tarefas, atraso computacional e de rede e, por consequência, o sistema de controle e a resposta dinâmica.

O passo inicial foi determinar o modelo matemático da dinâmica de um relógio. Para tanto, todas as equações foram modeladas em um domínio discreto utilizando equações de diferenças, transformada Z e espaço de estados. Sendo a dinâmica de um relógio crescente e desejavelmente uma reta, quatro modelos dentro desta abordagem foram desenvolvidos e propostos, com todos os modelos descrevendo o microtick e o macrotick.

O primeiro modelo proposto é um modelo matemático por equação de diferenças. A equação de diferenças permite modelar um sistema dinâmico representado por uma sequência de pontos representados de maneira discreta e equiespaçadas no tempo indexado por k. A vantagem do uso da equação de diferenças é que computadores e processadores são dispositivos discretos, fazendo com que o modelo seja uma boa opção para este tipo de utilização. Este modelo foi usado durante toda a tese para simular, verificar e validar os algoritmos de sincronização de relógios.

O segundo modelo proposto foi um modelo por equação de diferenças utilizando PWM. O modelo do macrotick definido na equação (4.13) possui um trem de impulsos que amostra a segunda parte da equação. Este modelo possui o termo MMCF que é o fator de conversão do Microtick para o Macrotick, em outras palavras é o divisor de frequências. Baseado nisso, este

modelo propõe uma maneira de modelar a divisão de frequências através do uso da Modulação por Largura de Pulso (PWM, em inglês *Pulse Width Modulation*). Este modelo se mostra promissor para fins de futuras implementações, no entanto para análise matemática, preferiu-se o uso do primeiro modelo, pois o PWM é um modelo não linear.

O terceiro modelo representa as equações de diferenças de um relógio com deriva constante no domínio Z, utilizando a transformada Z. Para fins de análise, este modelo se mostra promissor e muito mais vantajoso, já que existe uma gama de ferramentas matemáticas que auxiliam esta análise. As análise de estabilidade realizadas nesta tese foram todas baseadas neste modelo.

Um quarto modelo foi proposto utilizando a Teoria de Controle, utilizando um modelo por espaço de estados discretos. Este é o primeiro modelo que faz a ponte entre o mundo do controle e o mundo da sincronização de relógios. Apesar de este modelo não ter sido muito explorado nesta tese, o modelo por equação de estados fornece uma nova visão sobre a teoria de sincronização de relógios. Além disso, com este modelo, é possível aplicar toda a teoria de controle discreto moderno, abrindo assim várias possibilidades para análise e projeto de sincronização de relógios.

O desenvolvimento destes modelos teve como contribuições:

a) Quatro modelos que descrevem a dinâmica de um relógio (microtick e macrotick);

b) Uso dos modelos de equação de diferenças e transformada Z para o desenvolvimento, baseado em técnicas de controle, estudos e novas técnicas para a sincronização de relógios sobre sistemas de controle por redes;

c) Um modelo de implementação utilizando PWM;

d) Um modelo em espaço de estados discretos que faz a ponte entre o mundo do controle e o mundo da sincronização de relógios.

Com base no modelo por equação de diferenças (e seguindo a técnica "*cross-fading*"), foram propostas três técnicas de amortização para minimizar descontinuidades de tempo no instante de correção dos algoritmos de relógios.

A primeira técnica proposta foi utilizando uma função rampa. Para verificar a técnica, foram feitos vários casos de simulação utilizando dois algoritmos: 1) de média com 2 relógios; e 2) FTM com 4 relógios. Os casos mostraram que ajustando as métricas do projeto adequadamente é possível melhorar o desempenho da sincronização de relógios, *i.e*, as precisões do algoritmo FTM foram melhoradas. As vantagens desta técnica são: 1) Nenhum caso simulado causou instabilidade no sistema; 2) Em um simples ajuste das métricas, é possível se igualar ao caso sem a amortização. As desvantagens desta técnica são: 1) a necessidade de 2 chaveamentos: a) quando a função atinge o valor zero(nG); b) no próximo instante de re-sincronização; e 2) a dificuldade em se desenvolver uma análise algébrica útil.

A segunda técnica proposta foi utilizando uma função exponencial. Para verificar a técnica, foram feitos vários casos de simulação utilizando dois algoritmos: 1) de média com 2 relógios; e o 2) FTM com 4 relógios. Os casos mostraram que ajustando as métricas do projeto adequadamente é possível melhorar os resultados, *i.e.*, as precisões do algoritmo foram melhoradas. As vantagens desta técnica são: 1) Não necessita de 2 chaveamentos, somente um no instante de re-sincronização; 2) facilita no desenvolvimento de uma análise algébrica. As desvantagens desta técnica são: 1) pode instabilizar o sistema se as métricas forem mal ajustadas; 2) a função exponencial tem um "*overshoot*" inicial maior.

Uma terceira técnica é possível com o uso de uma função degrau. No entanto a função degrau, de fato, é um caso particular da função rampa.

O estudo da descontinuidade teve como contribuições:

a) Três técnicas de amortização propostas: 1) uma baseada na função rampa; 2) uma baseada na função exponencial; 3) e a terceira baseada em uma função degrau.

b) Desenvolvimento analítico de técnicas de amortização para algoritmos de sincronização de relógios. Com estas técnicas, foi possível melhorar o desempenho do algoritmo FTM e PTP utilizando as funções amortizadoras propostas (técnica “*cross-fading*”).

Com base no modelo por equação de diferenças (e seguindo a técnica "cross-fading"), foram propostas três técnicas de amortização para minimizar descontinuidades de tempo no instante de correção dos algoritmos de relógios.

Os algoritmos tradicionais de sincronização de relógios existentes não foram desenvolvidos para trabalhar para o sistema de controle. Ou seja, os algoritmos de sincronização, antes, desenvolvidos para trabalharem para ter a melhor convergência possível, devido a novos requisitos tais como integração do sistema, agora devem considerar um nova métrica para que, ao estabelecer a sincronização de relógios, o algoritmo tenha uma mínima influência sobre o controle. Portanto, a primeira proposta foi estabelecer uma nova métrica, chamada aqui nesta tese de **controle**. A métrica **controle** é a métrica que indica que o algoritmo de sincronização deve levar em consideração, além da **precisão** e **exatidão**, também o **controle** e a resposta dinâmica. Com as métricas definidas, os estudos sobre o sistema de controle por redes e as descontinuidades de tempo dos algoritmos indicaram que o viés inicial e descontinuidade de tempo são as duas imperfeições que mais influenciam o controle e sua resposta dinâmica. Portanto, assim como na métrica, introduziram-se as imperfeições de viés inicial e descontinuidade de tempo como imperfeições a serem corrigidas (ou levadas em consideração) pelo algoritmo de sincronização. Este novo modelo de métricas e imperfeições não exclui as antigas relações na qual os algoritmos de sincronização são

projetados, de fato, introduz novas relações possíveis no desenvolvimento dos algoritmos de sincronização. No entanto, os objetivos conflitantes, tanto das métricas quanto das imperfeições, dificultam o desenvolvimento de um algoritmo com uma lógica simples, levando a necessidade do desenvolvimento de um algoritmo reconfigurável. Assim, esta tese desenvolveu o conjunto de algoritmos de sincronização de relógios ReS (do inglês, *Reconfigurable Smoothable*). Esta tese propôs, então, quatro algoritmos de sincronização:

a) **Algoritmo ReS:** é um algoritmo de sincronização de tempo reconfigurável e suavizável, em que a métrica principal não é especificada e portanto, deve ser especificada.

b) **Algoritmo AReS:** é um algoritmo de sincronização de tempo reconfigurável e suavizável, onde a métrica principal de sincronização é a **exatidão** (do inglês, ***accurate***).

c) **Algoritmo PReS:** é um algoritmo de sincronização de tempo reconfigurável e suavizável, onde a métrica principal de sincronização é a **precisão** (do inglês, ***precise***).

d) **Algoritmo PAReS:** é um algoritmo de sincronização de tempo reconfigurável e suavizável, onde as métricas principais de sincronização são a **exatidão** e **precisão**.

O conjunto de algoritmos de sincronização de tempo se propõem a serem reconfiguráveis e suavizáveis, com métricas mistas de exatidão, precisão e controle. A reconfiguração é feita baseada em três modos de operação, conforme o diagrama de modos da Figura 6.3. Sendo os modos de operação:

a) **Modo STM:** No modo de inicialização (em Inglês, *Startup Mode*) todos os algoritmos devem seguir como métrica principal a **exatidão** e a imperfeição a ser corrigida é o **viés inicial**.

b) **Modo NOMO:** No Modo Nominal de Viés Instantâneo (em inglês, *Nominal Offset Mode*) as métricas e imperfeições dos algoritmos não são especificadas. No entanto, o objetivo deste modo é corrigir de forma eficaz os **vieses instantâneos** do macrotick gerados pela deriva dos microtick dos relógios.

c) **Modo NOMD:** No Modo Nominal de Deriva (em inglês, *Nominal Drift Mode*) a métrica e imperfeição dos algoritmos são respectivamente **exatidão** e **deriva**. O modo visa corrigir de forma eficaz a curvatura do macrotick corrigindo os fatores de geração de macrotick em relação ao microtick de cada relógio.

A suavização foi desenvolvida introduzindo nas funções de convergência a função amortizadora que multiplica a função de convergência. Foram usadas as funções amortizadoras descritas no capítulo 5, no entanto, nada impede o uso de outros tipos de funções amortizadoras.

Além dos algoritmos, foi estabelecido um procedimento para o projeto da sincronização de relógios, em que utilizando os procedimentos descritos pelos fluxogramas e alterando algumas características é possível de fato utilizar diferentes algoritmos dentro deste procedimento, inclusive os algoritmos FTM, PTP e SR. Este procedimento permite que os algoritmos de sincronização trabalhem com diferentes métricas e imperfeições, estabelecendo assim um conjunto de algoritmos reconfiguráveis e suavizáveis de métrica mista (precisão, exatidão e controle) para uso em sistemas de controle por rede e/ou sistemas complexos e altamente integrados. Todos os procedimentos foram estabelecidos para sincronização de relógios lógicos (macrotick). No entanto, é possível aplicar o mesmo procedimento sobre relógios físicos (microtick) com o mínimo de alterações.

Para a verificação e validação sobre o sistema de controle por redes, foram simulados 41 casos, utilizando diferentes algoritmos de sincronização e duas redes de comunicação TDMA e CSMA/CD. Para uma melhor análise, foram

especificados 3 indicadores de desempenho: Desvio Padrão, Média, e Critério Integral do Erro Quadrático (ISE).

Das simulações, no algoritmo PTP com um sistema de controle em CSMA/CD a sincronização foi eficiente, mas o sistema de controle instabilizou-se sendo o pior critério integral de todas as simulações. O FTM teve uma boa média e desvio padrão mas degradou um pouco o controle. Já o algoritmo SR teve um bom desempenho sobre os 3 indicadores, inclusive melhorando o controle.

O algoritmo AReS com uma função amortizadora rampa com controle em rede CSMA/CD, obteve o melhor desvio padrão da sincronização dos relógios.O caso *Deadbeat* com SR e sistema de controle CSMA/CD, obteve a melhor média da sincronização. O *Deadbeat* com FTM, obteve o melhor resultado com rede CSMA/CD para o critério integral.O Algoritmo PAReS Rampa obteve o melhor critério integral e portanto o melhor desempenho do sistema de controle, pois melhorou a resposta em relação ao caso nominal além de manter a média e desvio padrão baixos. Após, foram realizadas análises de estabilidade dos modelos propostos com alguns parâmetros. Nem todos os casos foram estáveis, mas foi demonstrado através da análise dos pólos e zeros que os modelos e algoritmos de sincronização são marginalmente estáveis em malha aberta, com somente um caso instável.

Conclui-se, portanto, que o conjunto de algoritmos ReS melhoraram o desempenho tanto da sincronização quanto do sistema de controle. As métricas mistas de precisão, exatidão e controle sobre um algoritmo de reconfiguração e suavizável demonstraram ser uma abordagem a ser seguida no projeto de algoritmos de sincronização de relógios para sistemas de controle por redes. Além disso, com otimização e ajustes de parâmetros é possível chegar a resultados muito melhores.

As contribuições do estudo do transitório e estabilidade dos algoritmos de sincronização de relógios sobre um sistema de controle foram:

a) Um nova abordagem métrica e de imperfeições para o projeto de algoritmos de sincronização de relógios;

b) Um procedimento para o projeto da sincronização de relógios

c) 4 Algoritmos de sincronização (ReS, PReS, AReS, PAReS) reconfiguráveis e suavizáveis;

d) Uso de 3 indicadores de desempenho (Desvio Padrão, Média e ISE) para analisar a qualidade dos algoritmos de sincronização de relógios sobre sistemas de controle por redes;

e) Modelo por equação de diferenças, transformada Z, espaço de estados e utilizando PWM, para análise e projeto de algoritmos de sincronização de relógios;

f) Modelo de simulação baseado no Matlab/Simulink/TrueTime para projeto e análise, tanto de sistemas de controle por redes, quanto de sincronização de relógios.

10.1. Sugestões de Trabalhos Futuros

a) Aplicar os algoritmos propostos para a sincronização de relógios físicos

b) Desenvolver novos indicadores de desempenho tanto para analise de qualidade, quanto de projeto de algoritmos de sincronização de relógios

c) Aplicar todo o procedimento utilizando outros tipos de controladores para as funções de convergência

d) Desenvolver todo o procedimento utilizando o modelo de espaço de estados

e) Analisar o modelo de espaço de estados e estabelecer os limites de suas habilidades (estabilidade, controlabilidade, observabilidade...)

f) Desenvolver um modelo de simulação sobre plantas sensíveis a variação temporal, tal como plantas com modos flexíveis.

REFERÊNCIAS BIBLIOGRÁFICAS

AMARAL, J. C. **Estudo dos efeitos da reconfiguração sobre o transitório e a estabilidade de sistemas de controle reconfiguráveis**. 2013. 192 p. (sid.inpe.br/mtc-m19/2013/07.25.18.55-TDI). Tese (Doutorado em Mecânica Espacial e Controle) - Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos, 2013. Disponível em: <http://urlib.net/8JMKD3MGP7W/3EGT738>. Acesso em: 30 jul. 2015.

ATTIYA, H.; HAY, D.; WELCH, J. L. Optimal clock synchronization under energy constraints in wireless ad-hoc networks. In:INTERNATIONAL CONFERENCE, OPODIS, 9., 2006, Pisa, Italia. **Proceedings…** Springer. 2006. p. 221-234.

CARPI, A.; EGGER, A. **Uncertainty, error, and confidence**. VisionLearning, 2008. Disponivel em: <http://www.visionlearning.com/en/library/Process-of-Science/49/Uncertainty-Error-and-Confidence/157>. Acesso em: 26 Agosto 2014.

CRISTIAN, F. Probabilistic clock synchronization. **Distributed Computing**, v. 3, p.146-158,1989.

DAVID, A. W.; NEIL, A.; CLIFFORD, H. C. **The science of timekeeping**. Agilent Technologies. [S.l.], p. 88. 1997. (Agilent AN 1289).

DUTERTRE, B. **The Welch-Lynch clock synchronization algorithm**. University of London. Londres, p. 24. 1998. (747).

EIDSON, J. C. **Measurement, control, and communication using IEEE 1588**. [S.l.]: Springer, 2006. 283 p. ISBN ISBN 978-1-84628-251-5. Advances in Industrial Control.

EIDSON, J. C. **Measurement, control, and communication using IEEE 1588**. New York: Springer-Verlag, 2006. 283 p. ISBN ISBN-10: 1846282500.

GWALTNEY, D. A.; BRISCOE, J. M. **Comparison of communication architectures for spacecraft modular avionics systems**. Alabama: Nasa. Marshall Space Flight Center, 2006. 32p. (NASA/TM-2006-214431).

HANZLIK, A.; ADEMAJ, A.; KOPETZ, H. Combination of clock-state and clock-rate correction in fault-tolerant distributed systems. **Journal Real-Time Systems**, v. 33, n. 1-3, p. 139-173,1 Julho, 2006.

HENRIKSSON, D.; CERVIN, A.; ARZÉN, K. Truetime: simulation of control loops under shared computer resources. In: IFAC WORLD CONGRESS ON AUTOMATIC CONTROL, 15., 2002,. Barcelona. **Proceedings…** : Barcelona: International Federation of Automatic Control, 2002.

HIESH, G.; HUNG, J. C. Phase-Locked Loop Techniques - A Survey. **IEEE Transactions on Industrial Electronics**, 43, Dezembro 1996. 609-615.

KIHARA, M.; ESKELINEN, P.; ONO, S. **Digital clocks for synchronization and communications**. Boston: Artech House, 2003. 274 p. ISBN ISBN-13: 978-1580535069.

KOPETZ, H. Sparse time versus dense time in distributed real-time systems. In: INTERNATIONAL CONFERENCE ON DISTRIBUTED COMPUTING SYSTEMS, 12., 1992, Yokohama, Japão. **Proceedings…** 10.1109/ICDCS.1992.235008. 1992. p. 460-467.

KOPETZ, H. **Real time systems:** design principles for distributed embedded applications. 1. ed. Norwell: Kluwer Academic Publishers, 1997. 378 p. ISBN 978-1-4419-8237-7.

KOPETZ, H. et al. **A synchronization strategy for a TTP/C controller**. SAE International. 1997. SAE Technical Paper 960120, 1996, doi:10.4271/960120.

KOPETZ, H.; BAUER, G. The time-triggered architecture. **Proceedings of the IEEE**, 91, n. 1, Janeiro 2003. 112-126.

KOPETZ, H.; OCHSENREITER, W. Clock synchronization in distributed real-time systems. **IEEE Transaction on Computers**, C-36, n. 8, Agosto 1987. 933-940.

KUO, B. C. **Digital control systems**. Champaign: SRL Publishing Company, 1977. ISBN 0-918152-01-1.

LAMPORT, L. Time, clocks, and the ordering of events in a distributed system. **Coomunications of the ACM**, 21, n. 7, Julho 1978. 558-565.

LINDSEY, W. C. **Synchronization systems in communication and control**. New Jersey: Prentice-Hall Inc., Englewood Cliffs, 1972. ISBN ISBN: 0-13-879957-1.

LUSTOSA, H. D. **Influência de tipos de barramentos e de suas características de alto nível em sistemas de controle por rede**. 2009. 245 p. (INPE-15781-TDI/1524). Dissertação (Mestrado em Mecânica Espacial e Controle) - Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2009. Disponível em: <http://urlib.net/8JMKD3MGP8W/34RQSJ2>. Acesso em: 30 jul. 2015.

LUSTOSA, H. D.; SOUZA, M. L. O. Influences of data bus protocols on an aircraft fly-by-wire networked control systems. In: CONGRESSO E EXPOSIÇÃO INTERNACIONAIS DE TECNOLOGIA DA MOBILIDADE, 2008, São Paulo. **Proceedings...** São Paulo: SAE Brasil, 2008. p. 54.

MALEKPOUR, M. R. A Byzantine-fault tolerant self-stabilizing protocol for distributed clock synchronization systems. In: INTERNATIONAL

CONFERENCE ON STABILIZATION, SAFETY AND SECURITY OF DISTRIBUTED SYSTEMS, 8., 2006, Berlin. **Proceedings…** Springer-Verlag. 2006. p. 411-427.

MARTÍ, P.; LOZOYA, C.; FUERTES, J. M. **Clock synchronization for networked control systems using low-cost microcontrollers**. Catalonia, Espanha: Technical University of Catalonia, p. 18. 2008. (ESAII-RR-08-02).

MATHWORKS. Matlab Help - Control System Toolbox Documentation. **Matlab Help**, 2010. Disponível em: <http://www.mathworks.com/help/control/ref/pole.html>. Acesso em: 22 jul. 2015.

MEYR, H.; ASCHEID, G. **Synchronization in digital communications:** v1-phase, frequency-locked loops and amplitude control. USA: John Wiley & Sons, v. 1, 1990. ISBN ISBN --471-50193-x.

MOREIRA, M. L. B. **Projeto e simulação de um controle discreto para a plataforma multi-missão e sua migração para um sistema operacional de tempo real**. Instituto Nacional de Pesquisas Espaciais. São José dos Campos, p. 181. 2006. (INPE-14202-TDI/1103).

OKLOBDZIJA, V. G. et al. **Digital System Clocking High-Performance and Low-Power Aspects**. Haboken: John Wiley & Sons, 2003. 264 p. ISBN ISBN: 978-0-471-27447-6.

RAMANATHAN, P.; SHIN, K. G.; BUTLER, R. W. Fault-tolerant clock synchronization in distributed systems. **Journal Computer**, Los Alamitos, CA, EUA, 23, n. 10, Outubro 1990. 33-42.

RYU, M.; PARK, J.; HONG, S. Timing constraint remapping to avoid time discontinuities in distributed real-time systems. In: IEEE Real-Time Technology and Applications Symposium, 5., 1999, Vancouver, BC. **Proceedings…** IEEE. 1999. p. 89-98.

SCHNEIDER, F. B. **A Paradigm for reliable clock synchronization**. Advanced Seminar of Local Area Network. Bandol, França: [s.n.]. 1986.

TREDINNICK, M. R. A. C. **Análise da estabilidade de sistemas de controle análogicos-digitais em função do período de amostragem**. 2009. 583 p. (INPE-15732-TDI/1478). Tese (Doutorado em Mecânica Espacial e Controle) - Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2009. Disponível em: <http://urlib.net/8JMKD3MGP8W/34NCHQL>. Acesso em: 30 jul. 2015.

TTTECH COMPUTERTECHNIK AG. **TTEthernet** - a powerful network solution for all purposes. Viena: Site da TTTEch, 2010. DisponÍvel em: <http://www.tttech.com/technologies/ttethernet/>. Acesso em: 26 Agosto 2014.

VALDIVIA, R. H. V. **Influência dos agendadores da computação (RMS) e da comunicação (TDMA) na estabilidade de um sistema de controle por rede**. 2009. 168 p. (INPE-15692-TDI/1466). Tese (Doutorado em Mecânica Espacial e Controle) - Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2009. Disponível em: <http://urlib.net/8JMKD3MGP8W/34MSR7S>. Acesso em: 30 jul. 2015.

WELCH, J. L.; LYNCH, N. A new fault-tolerant algorithm for clock synchronization. **Information and Computation**, v. 77, n. 1, p. 1-36, April 1988. doi>10.1016/0890-5401(88)90043-0.

ZHANG, W.; BRANICKY, M. S.; PHILLIPS, S. M. Stability of networked control systems. **IEEE Control Systems**, v. 21, n. 1, p. 84 - 99, Feb. 2001. ISSN ISSN:1066-033X.

APÊNDICE A - DETALHAMENTO DO DESENVOLVIMENTO ANALÍTICO PARA O EQUACIONAMENTO DO MACROTICK

Equacionamento detalhado do macrotick:

Dada a equação (4.6) do microtick, repetindo a mesma abaixo:

$$\mu T_i(k+1) = \mu T_i(k) + \rho_i T \tag{A.1}$$

Dada a equação (4.7) do macrotick, repetindo a mesma abaixo:

$$mT_i(k) = \frac{\mu T_i(k)}{MMCF_i} \tag{A.2}$$

Passando a equação (A.2) de k para k+1, tem-se:

$$mT_i(k+1) = \frac{\mu T_i(k+1)}{MMCF_i} \tag{A.3}$$

Combinando as equações (A.1) e (A.3), chega-se a:

$$mT_i(k+1) = \frac{\mu T_i(k) + \rho_i T}{MMCF_i} \tag{A.4}$$

Separando os termos de (A.4), tem-se:

$$mT_i(k+1) = \frac{\mu T_i(k)}{MMCF_i} + \frac{\rho_i T}{MMCF_i} \tag{A.5}$$

Substituindo o primeiro termo da equação (A.5) pela equação (A.2), tem-se:

$$mT_i(k+1) = mT_i(k) + \frac{\rho_i T}{MMCF_i} \tag{A.6}$$

Isolando o termo ρT da equação (A.1), encontra-se:

$$\rho_i \mathrm{T} = \mu T_i(k+1) - \mu T_i(k) \tag{A.7}$$

Substituindo (A.7) em (A.6), tem-se:

$$mT_i(k+1) = mT_i(k) + \frac{\mu T_i(k+1) - \mu T_i(k)}{MMCF_i} \tag{A.8}$$

No entanto, o macrotick é um relógio que é amostrado em uma frequência diferente da do microtick, por isso a existência do MMCF que é um divisor de frequências. Para uma modelagem mais fiel ao que ocorre com o macrotick em computadores digitais, é necessário adicionar à equação (A.8) uma chave que amostra o sinal em uma frequência diferente da do microtick, de acordo com o MMCF definido. Para isso, define-se um trem de impulsos Ш da seguinte maneira:

$$Ш = \begin{cases} \sum_{k=1}^{N} \delta(t - kT \cdot MMCF_z) \\ 0, \qquad se\ k \leq 0 \end{cases} \tag{A.9}$$

Portanto, utilizando a equação (A.9) em (A.6), a equação geral do macrotick é definido como:

$$mT_i(k+1) = mT_i(k) + \frac{\left[\sum_{\mathrm{k-MMCF_z}}^{\mathrm{k+1}} \rho_i T\right] \cdot Ш}{MMCF_i} \tag{A.10}$$

No entanto considerando-se a deriva constante, chega-se à seguinte conclusão:

$$\sum_{\mathrm{k-MMCF_z}}^{\mathrm{k+1}} \rho_i T = (1 + \mathrm{MMCF_z})\rho_i T \tag{A.11}$$

Ou que pode ser reescrito da seguinte maneira:

$$\sum_{\mathrm{k-MMCF_z}}^{\mathrm{k+1}} \rho_i T = (\rho_{k+1} - \rho_{\mathrm{k-MMCF_z}})T \tag{A.12}$$

Assim, baseado na equação (A.7), tem-se que:

$$\rho_{k+1} T = \mu T_i(k+1) - \mu T_i(k) \tag{A.13}$$

$$\rho_{k-MMCF_z} T = \mu T_i(\mathrm{k} - MMCF_z) - \mu T_i(k)$$

Observando que ρ é um delta, e portanto o instante (k-MMCFz), apesar do sinal negativo, é um instante superior ao instante k para aquele ponto, devido ao trem de impulsos. Assim, substituindo (A.13) em (A.12), tem-se:

$$\sum_{\mathrm{k-MMCF_z}}^{\mathrm{k+1}} \rho_i T = (\mu T_i(k+1) - \mu T_i(\mathrm{k} - MMCF_z))T \qquad \text{(A.14)}$$

Substituindo (A.14) em (A.10), chega-se a equação geral do macrotick para deriva constante em todos os instantes de amostragem:

$$mT_i(k+1) = mT_i(k) + \frac{[\mu T_i(k+1) - \mu T_i(k - MMCF_i)] \cdot \text{Ш}}{MMCF_i} \qquad \text{(A.15)}$$

Condições de existência do MMCF

A equação (A.2) demonstra que existe uma razão definida entre o microtick e o macrotick. Esta razão é proveniente do divisor de frequências. Chega-se portanto a:

$$\alpha = \frac{1}{MMCF_i} \qquad \text{(A.16)}$$

Assim, substituindo (A.16) em (A.2), chega-se:

$$mT_i(k) = \alpha \mu T_i(k) \qquad \text{(A.17)}$$

Em que, tem-se:

$$\alpha = \frac{mT_i(k)}{\mu T_i(k)} \qquad \text{(A.18)}$$

Em que se definem as razões matemáticas e seguintes condições de existência:

a) α=0: portanto, $mT_i(k) = 0$. Isto signfica que para a razão de 0, não existe possibilidade de existência do macrotick;

b) α < 0: para esta razão o macrotick ou microtick são negativos. Dado que não existe tempo negativo, esta razão só pode ocorrer caso haja uma inversão de fase devido a descontinuidades de tempo. Havendo inversão de fase do sistema, o sistema torna-se instável e, portanto, esta razão deve ser evitada.

c) 0 < α < 1: para esta razão o $mT_i(k) < \mu T_i(k)$. Significando que o macrotick possui um período de amostragem maior que o microtick.

d) α = 1: para esta razão $mT_i(k) = \mu T_i(k)$. Significando que o macrotick e microtick possuem o mesmo período de amostragem.

e) α > 1: para esta razão o $mT_i(k) > \mu T_i(k)$. Significando que o macrotick possui um período de amostragem menor que o microtick.

APÊNDICE B - LIMITAÇÕES DEVIDO A FUNÇÕES DE PÓLOS E ZEROS DO MATLAB

De acordo com Mathworks (2010), utilizando-se da função "pole" e consequentemente de suas derivações "zero" e "pzmap", caso o sistema tenha atrasos internos, os pólos são obtidos ajustando, primeiramente, todos os atrasos internos a zero (criando uma aproximação de Padé de ordem zero), de modo que o sistema tenha um número finito de zeros. Para alguns sistemas, estabelecendo atrasos iguais a zero, criam-se laços algébricos singulares que resultam em inadequados ou mal definidas aproximações. Para estes sistemas, o pólo retorna um erro. Este erro não implica um problema com o modelo em si, e sim com a função definida pelo Matlab. Além disso, ainda de acordo com Mathworks (2010), pólos multiplos são numericamente sensiveis e não podem ser computados com alta precisão e fidelidade. Assim, um pólo λ com multiplicadade m, tipicamente dá origem a um conjunto de pólos computados distribuídos ao longo de um círculo com o centro em λ e raio da ordem de:

$$\partial = \varepsilon^{\frac{1}{m}} \tag{B.1}$$

Em que, ε é o valor da precisão relativa da máquina (eps) e m a multiplicidade dos pólos. Portanto, dado que a eps da máquina usada é da ordem de 2.2204e-016, tem-se portanto para diferentes multiplicidades de pólos os seguintes valores:

m=1: $\partial = 2.2204\text{e}-016$
m=2: $\partial = 1.4901\text{e}-008$
m=3: $\partial = 6.0555\text{e}-006$
m=4: $\partial = 1.2207\text{e}-004$
m=5: $\partial = 7.4010\text{e}-004$
m=6: $\partial = 0.0025$

Estes resultados, explicam os erros que causam algumas instabilidades no Capitulo 9.