UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Câmpus de Bauru - SP

RANULFO ACIR DE OLIVEIRA RESENDE

Estudo da Dinâmica de Populações Utilizando a Técnica de Polinômios de Colocação

Bauru - SP
2015

RANULFO ACIR DE OLIVEIRA RESENDE

Estudo da Dinâmica de Populações Utilizando a Técnica de Polinômios de Colocação

Dissertação apresentada à Faculdade de Engenharia de Bauru - UNESP como parte dos requisitos para obtençao do Título de MESTRE EM ENGENHARIA ELÉTRICA.

Prof. Dr. José Manoel Balthazar
Orientador
Profa. Dra. Edilaine Martins Soler
Coorientadora

Bauru - SP
2015

Resende, Ranulfo Acir de Oliveira.
Estudo da dinâmica de populações utilizando a técnica de polinômios de colocação / Ranulfo Acir de Oliveira Resende, 2015
214 f.

Orientador: José Manoel Balthazar
Coorientadora: Edilaine Martins Soler

Dissertação (Mestrado)–Universidade Estadual Paulista. Faculdade de Engenharia, Bauru, 2015

1. Otimização. 2. Controle ótimo. 3. Modelagem matemática. 4. Simulação numérica. 5. Dengue. I. Universidade Estadual Paulista. Faculdade de Engenharia. II. Título.

UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
CAMPUS DE BAURU
FACULDADE DE ENGENHARIA DE BAURU

ATA DA DEFESA PÚBLICA DA DISSERTAÇÃO DE MESTRADO DE RANULFO ACIR DE OLIVEIRA RESENDE, DISCENTE DO PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA, DO(A) FACULDADE DE ENGENHARIA DE BAURU.

Aos 23 dias do mês de outubro do ano de 2015, às 14:00 horas, no(a) Anfiteatro da Diretoria Técnica de Informática, reuniu-se a Comissão Examinadora da Defesa Pública, composta pelos seguintes membros: Prof. Dr. JOSE MANOEL BALTHAZAR do(a) Professor Voluntário do Departamento de Engenharia Mecânica, da Faculdade de Engenharia, Prof. Dr. DIEGO COLON do(a) PTC/EPUSP/USP, Prof. Dr. ÁTILA MADUREIRA BUENO do(a) Engenharia de Controle e Automação/ Unidade de Sorocaba, sob a presidência do primeiro, a fim de proceder a arguição pública da DISSERTAÇÃO DE MESTRADO de RANULFO ACIR DE OLIVEIRA RESENDE, intitulado "ESTUDO DA DINÂMICA DE POPULAÇÕES UTILIZANDO A TÉCNICA DE POLINÔMIOS DE COLOCAÇÃO". Após a exposição, o discente foi arguido oralmente pelos membros da Comissão Examinadora, tendo recebido o conceito final:_ APROVADO_ _ _ . Nada mais havendo, foi lavrada a presente ata, que, após lida e aprovada, foi assinada pelos membros da Comissão Examinadora.

JM -.

Prof. Dr. JOSE MANOEL BALTHAZAR

Diego Colon

Prof. Dr. DIEGO COLON

Prof. Dr. ÁTILA MADUREIRA BUENO

AGRADECIMENTOS

Meus agradecimentos:

- A Deus, por ter-me dado força e saúde para realizar este trabalho;
- À minha família, pelo apoio e incentivo;
- Ao Prof. Dr. José Manoel Balthazar e à Profa. Dra. Edilaine Martins Soler, pelas orientações, confiança e incentivo à pesquisa;
- Ao Prof. Dr. José Alfredo Covolan Ulson, pela coorientação;
- Ao Prof. Dr. Leonardo Nepomuceno, pela interação na pesquisa; e
- À Administração da FEB-Unesp, seu corpo docente e funcionários, pelo apoio a esta pesquisa.

"Paz não é a ausência de guerra. É uma virtude, um estado mental, uma disposição para a benevolência, confiança e justiça."

Baruch Spinoza (1632-1677)

À minha família, com quem aprendemos e ensinamos o amor, o perdão, a esperança e a fé.

A todos que ensinam e fazem da educação a alavanca para mover o mundo.

RESUMO

Este trabalho abrange o estudo de um sistema de controle na sua forma mais simples, ou seja, controlador e processo. A otimização de uma função objetivo definida por um determinado índice de desempenho, que tem como variáveis o estado do sistema e os sinais de controle, busca atingir o controle ótimo que minimiza este índice.

A pesquisa levantou os principais conceitos afetos à teoria do controle ótimo e algumas das principais técnicas de solução do que constitui o conceito central da teoria, o Problema de Controle Ótimo, ou PCO.

Em especial, foi possível implementar a técnica de otimização dinâmica por pontos de colocação, aplicada em um ambiente integrado constituído de ferramentas de modelagem, JModelica, e de otimização, IPOPT.

Como principal referência foi utilizado o PCO de um modelo matemático da transmissão da dengue, com auxílio do qual alguns pesquisadores tentaram calcular, em função dos custos relativos de cada técnica de combate à doença, inseticidas e liberação no ambiente de machos estéreis, as melhores estratégias para diminuir a população de fêmeas fertilizadas, diretamente relacionadas aos casos da doença.

Os trabalhos originais que pesquisaram o mesmo problema abordaram a questão pela formulação de um PCO com solução pelo Princípio do Máximo de Pontryagin, em procedimento sequencial e aproximado. Outros utilizaram algoritmos genéticos e otimização multiobjetivo.

Os resultados desta pesquisa que podem ser destacados são inicialmente a própria revisão da teoria de controle ótimo, a instalação e a operação do referido ambiente integrado de modelagem, simulação e otimização, onde diversos PCO foram e podem ser solucionados, bem como o programa em linguagem Python que solucionou estes PCO e, finalmente, as considerações teóricas sobre o PCO do modelo de transmissão da dengue, permitindo sugerir alterações no cálculo da função objetivo para melhor utilização dos sinais de controle.

Palavras-chave: otimização, controle ótimo, modelagem matemática, simulação numérica, dengue

ABSTRACT

This work comprises the study of a control system in its most simple format, that is, a controller and a process or a plant. The optimization of a cost function, with the system state as variables, aims to achieve the optimal control.

Starting from the concepts presented in the previous works, we studied alternatives optimization techniques suited to the optimal control.

We used as experimental basement a mathematical model of dengue disease's transmission, the same model adopted by some researchers to help to assess the best strategies to vanish the fertilized females, an index that is directly related to the illness cases, as function of the relative costs of each technique available to combat the disease: the use of insecticides and the release of sterile males to the environment.

The original researches that studied the same problem approached the matter by the formulation of an optimal control problem solved by Pontryagin Maximum Principle, in a sequential and approximated procedure. Others adopted genetic algorithm and multi-objective optimization.

This research shows the results of the approach by dynamic optimization with collocation points in a integrated environment with modeling and simulation, JModelica, and optimization, IPOPT, tools.

The application of the dynamic optimization with collocation points technique to the studied model allowed, by better observation of control signals and the assessment of the cost function, to suggest changes to a more efficient control.

Keywords: optimization, optimal control, mathematical modeling, numeric simulation, dengue

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIAÇÕES E SIGLAS

EDO	Equação Diferencial Ordinária
EDP	Equação Diferencial Parcial
HJB	Hamilton-Jacobi-Bellman (Equação de)
IPOPT	Interior-Point Optimizer (Ferramenta de Otimização)
MPC	Controle por Predição do Modelo
NMPC	Controle por Predição do Modelo Nao Linear
PCO	Problema de Controle Ótimo
PMP	Princípio do Mínimo (ou do Máximo) de Pontryagin
PNL	Programa Não Linear

LISTA DE SÍMBOLOS

$\boldsymbol{u}_0$	Controle inicial
$\boldsymbol{u}^*$	Controle ótimo[1]
$\bar{\boldsymbol{u}}$	Controle predito[2]
$\boldsymbol{x}_0$	Estado inicial
$\boldsymbol{x}^*$	Estado ótimo[3]
$\bar{\boldsymbol{x}}$	Estado predito[2]
C	Fator que caracteriza o meio quanto a nutrientes, espaço, etc
T_c	Horizonte de controle[2]
T_p	Horizonte de predição[2]
t_0	Instante de tempo inicial
t_f	Instante de tempo final
T	Instante de tempo final
c_4	Índice de benefício pelos machos estéreis
c_1	Índice de custo relacionado ao controle por inseticidas
c_2	Índice de custo relacionado ao controle por mosquitos estéreis
c_3	Índice de custo social
$J[\boldsymbol{x}, \boldsymbol{u}; t_0, T]$	Índice de desempenho, função custo ou função objetivo
n_e	Número de elementos para discretização do PCO[4]
n_c	Número de pontos de colocação em cada elemento[4]
$\tau_{i,k}$	Ponto de colocação 'k' no elemento 'i'[4]
$\tilde{\ell}_k$	Polinômios base de Lagrange com interpolação no ponto $\tau_{i,0}$[4]
ℓ_k	Polinômios base de Lagrange sem interpolação no ponto $\tau_{i,0}$[4]
r	Proporção de fêmeas que passam para a fase adulta
g_i	Restrições de desigualdade[4]
g_e	Restrições de igualdade[4]
G_i	Restrições pontuais de desigualdade[4]
G_e	Restrições pontuais de igualdade[4]

[1] Pode corresponder a um sinal de controle, não um valor instantâneo.
[2] Aplicável no MPC.
[3] Pode corresponder a um caminho ótimo, não um valor instantâneo.
[4] Aplicável na técnica do otimização dinâmica por pontos de colocação.

A(t)	População na fase aquática
F(t)	População de fêmeas fertilizadas (depois de acasalar)
I(t)	População de fêmeas imaturas (antes de acasalar)
U(t)	População de fêmeas não fertilizadas (depois de acasalar)
S(t)	População de machos estéreis.
M(t)	População de machos naturais (férteis)
u_1	Sinal de controle relacionado aos inseticidas
u_2	Sinal de controle relacionado à inserção de machos estéreis
β_S	Taxa de acasalamento dos machos estéreis
β	Taxa de acasalamento dos machos naturais
α	Taxa de inserção de mosquitos estéreis
ϕ	Taxa de oviposição intrínseca
γ	Taxa de mosquitos que passam da fase aquática para a adulta
μ_X	Taxa de mortalidade da população X
$\boldsymbol{u}$	Variável de controle
$\boldsymbol{x}$	Variável de estado
$\boldsymbol{Z}$	Variável no PNL, com todas as variáveis dos pontos de colocação de PCO discretizado[4]
$\boldsymbol{z}$	Variável que integra todas as variáveis dependentes do tempo em um PCO[4]

SUMÁRIO

1 INTRODUÇÃO

A Otimização corresponde à ciência que busca encontrar o valor máximo (ou mínimo) de uma determinada função, geralmente multivariável e com restrições sobre estas variáveis.

Os problemas de otimização aparecem em diversas áreas: na logística, na economia, na indústria, engenharia, etc. Aparecem em qualquer situação em que, a partir de um determinado ponto da função, descobrir o seu ponto de máximo (ou de mínimo) pode representar resultados mais eficientes, lucrativos ou mesmo a relevante economia de energia.

As técnicas de otimização foram desenvolvidas inicialmente na Segunda Guerra Mundial, em problemas logísticos de guerra, e continuaram crescendo em número e complexidade, passando-se dos iniciais problemas lineares, quadráticos, para os não lineares e com restrições de igualdade e de desigualdade, conforme (LUENBERGER, 1984).

O Controle, ou Engenharia de Controle, busca, partindo-se do estudo das características de um determinado sistema (mecânico, elétrico, térmico, químico, etc) e conhecendo-se a saída do sistema, atuar sobre o sistema de forma a manter constante ou com mínimas variações a atual saída ou, tendo-se uma referência desejada, levar o sistema da situação atual para a desejada, conforme (OGATA, 1993).

A ideia mais simples de um Sistema de Controle abrange um processo controlado e o seu controlador. Na realidade, os sistemas de controle podem ser muito complexos e com múltiplos controladores e processos se retroalimentando.

O Controle envolve sensores para conhecer os valores das variáveis controladas (do processo), os atuadores, para agir sobre processo (variáveis manipuladas) e o conhecimento de como o processo responde em suas variáveis controladas (medidas pelos sensores) às alterações nas variáveis manipuladas (aplicadas pelos atuadores). Este conhecimento do relacionamento entre variáveis controladas e manipuladas pode ser entendido como função de transferência do processo.

(HEMERLY, 1996) destaca que, no final da década de 50, a teoria de controle baseada em funções de transferência já estava bastante solidificada, havendo inovação na década de 60 com a introdução de variáveis de estado para representar sistemas dinâmicos.

Segundo (FRANKLIN et al., 1991, pp. 11), a utilização de EDO como modelos para sistemas de controle iniciou nos anos 50 com Bellman e Kalman nos Estados Unidos e Pontryagin na então União Soviética (URSS).

Um importante recurso no estudo de sistemas de controle é a simulação. Assim sendo, um sistema real é inicialmente modelado e o seu comportamento é estudado e analisado com recursos computacionais, analógicos ou digitais, sendo possível, muitas vezes, prever este comportamento diante de variações em parâmetros do modelo ou de distúrbios nos sinais de controle. A simulação, portanto, é uma importante ferramenta para atingir o aprimoramento dos sistemas, garantindo maior eficiência no controle.

O escopo deste trabalho abrange o estudo e a simulação de um sistema de controle na sua forma mais simples, ou seja, controlador e processo, sem realimentação. A otimização de uma função objetivo que tem como variáveis as variáveis de estado do sistema (e outras variáveis que forem necessárias, como as próprias variáveis de controle) e suas restrições, buscará atingir um controle ótimo.

Foi estudada a técnica de otimização dinâmica por pontos de colocação, utilizada integradamente nas ferramentas de modelagem, JModelica (ÅKESSON et al., 2010), (MAGNUSSON; ÅKESSON, 2012),i (ANDERSSON et al., 2011), e de otimização, IPOPT (WÄCHTER; BIEGLER, 2006).

Neste sentido, o referido ambiente integrado baseado em aplicações de *software* livre foi instalado e operado. Com o objetivo de "aferir" a técnica e o ambiente, foram estudados alguns PCO apresentados em referências clássicas de controle ótimo e, para subsidiar a pesquisa com cálculos de simulação mais expressivos, foi estudado o PCO relacionado ao modelo de transmissão da dengue (ESTEVA; YANG, 2005), (THOMÉ, 2007) e (THOMÉ et al., 2010). Tal modelo foi escolhido não somente pela relevância social do tema, que demanda urgência na aplicação de controles efetivos para a diminuição dos casos da doença, mas também por apresentar a complexidade matemática necessária para avaliação da técnica: o sistema do modelo é constituído de cinco equações diferenciais não lineares, algumas acopladas às outras.

Resultados de simulações expressivos foram alcançados pela programação de um *script* em linguagem Python que gerou de forma automatizada os dados obtidos nas soluções de todos os PCO estudados. Um outro *script* em linguagem Python foi utilizado na análise destes dados, permitindo avançar também em conceitos e na formulação teórica do principal PCO estudado.

1.1 OBJETIVOS DA PESQUISA

Foram objetivos generalizados da pesquisa:

- Investigar e conhecer os principais conceitos afetos ao controle ótimo;
- Investigar a aplicação prática da otimização em controle;
- Utilizar eficiente e eficazmente uma técnica de otimização em um problema de controle

ótimo conhecido; e

- Pesquisar possíveis contribuições teóricas e práticas ao PCO e à técnica estudada.

Para atingir estes objetivos, o capítulo 2 apresenta a metodologia adotada no desenvolvimento da pesquisa e os fundamentos teóricos nos quais se baseou. São descritos, em especial, os procedimentos para obtenção da fundamentação teórica, bem como para a realização das simulações e a análise dos resultados.

O capítulo 3 apresenta uma discussão sobre as principais técnicas empregadas no controle ótimo, com ênfase na otimização dinâmica por pontos de colocação, empregada nas simulações.

A seguir, no capítulo 4, são descritas as ferramentas utilizadas na pesquisa, especificamente as ferramentas integradas de modelagem e simulação, JModelica, e de otimização, IPOPT.

Finalmente, no capítulo 5 são apresentados os resultados da aplicação da técnica de otimização dinâmica por pontos de colocação ao PCO relacionado ao modelo de transmissão da dengue (ESTEVA; YANG, 2005), (THOMÉ, 2007), (THOMÉ et al., 2010), (BARSANTE et al., 2013).

Ao final da dissertação, no apêndice A, são apresentados o modelo em linguagem Modelica (extensão Optimica para controle) e os *scripts* Python utilizados nas otimizações, simulações e análises.

Os resultados gráficos do estudo variacional baseado em simulações foram apresentados nos apêndices B a E.

O próximo capítulo apresenta a a metodologia e os fundamentos teóricos da pesquisa.

2 METODOLOGIA E FUNDAMENTOS TEÓRICOS

Neste capítulo será apresentada a metodologia adotada e os procedimentos para obtenção, assimilação, entendimento e aplicação dos conhecimentos já estabelecidos sobre o tema da pesquisa.

Com o mesmo cuidado, aqui são descritos os procedimentos para a obtenção dos dados apresentados na seção 5, onde são discutidos os resultados das simulações.

2.1 METODOLOGIA

A pesquisa iniciou com o estudo dos principais conceitos de otimização e controle apresentados nas próprias disciplinas cursadas na FEB-Unesp, *Controle Digital*, *Otimização Não Linear*, *Tópicos em Otimização Linear, Inteira e Quadrática* e *Sistemas Mecatrônicos*.

Posteriormente, os conceitos básicos de otimização, como: otimização irrestrita e com restrições de igualdade e de desigualdade, método de pontos interiores, multiplicadores de Lagrange, referenciados na bibliografia, (LUENBERGER, 1984), (BAZARAA; SHETTY, 1979) e (BAZARAA et al., 2009), juntamente com os conceitos de controle ótimo, abrangendo variáveis de estado, controle realimentado, critério de estabilidade de Lyapunov, Princípio do Mínimo de Pontryagin e Equação de Hamilton-Jacobi-Bellman, (GOLNARAGHI; KUO, 2009), (OGATA, 1993), (OGATA, 1995), (LEWIS, 1986), (KIRK, 1970), (BRYSON, 1975), (ATHANS, 2007), foram compilados em um texto de fundamentos teóricos que é apresentado nas próximas seções.

Como citado anteriormente, esta fundamentação teórica teve como fontes não somente as referências bibliográficas apresentadas, mas também notas de aula das disciplinas cursadas na graduação e na pós-graduação.

Partindo-se desta fundamentação teórica, buscou-se na literatura trabalhos relacionados que apresentassem técnicas e ferramentas que aplicassem estes princípios em solução do problema de controle ótimo. Foram estudados alguns modelos de sistemas a serem controlados, bem como técnicas de otimização pela aplicação sequencial do PMP, por algoritmos genéticos e otimização multiobjetivo. Em especial, a técnica de otimização dinâmica por pontos de colocação baseada nas ferramentas de modelagem, JModelica (ÅKESSON et al., 2010), (MAGNUSSON; ÅKESSON, 2012), e de otimização, IPOPT, foi implementada e comparada com os casos das referências.

Como uma espécie de aferição deste ambiente, foi utilizado um modelo de PCO para um sistema de segunda ordem apresentado na literatura clássica de controle ótimo (KIRK, 1970, pp. 216), cujas soluções pelas diferentes técnicas foram comparadas.

O referido ambiente permitiu estudar um problema de controle ótimo aplicado a um modelo de transmissão da dengue (ESTEVA; YANG, 2005), (THOMÉ, 2007), (BARSANTE et al., 2013), (BANNWART, 2013), obtendo-se dados que evidenciaram a necessidade de alterar as condiçoes do problema estudado de forma a obter um controle mais efetivo.

Os dados e figuras foram obtidos automaticamente pela execução de um *script* em um ambiente integrado de modelagem, otimização e simulação. Este *script* foi apresentado e pode ser facilmente executado por outros pesquisadores em ambientes equivalentes próprios.

Todos os resultados, i.e, figuras, variáveis, modelos fontes e compilados, foram salvos em formatos usuais para fácil acesso e verificação. Figuras foram armazenadas em formatos "jpeg", "eps"e "png". Dados foram armazenados em formatos ".mat", no padrão do MATLAB®. Todas as saídas do otimizador foram armazenadas em arquivo de registro específico para a otimização executada. Todas os parâmetros relevantes do ambiente computacional foram armazenados em arquivo de registro próprio. Todos os dados, figuras e arquivos de registro foram agrupados sistematicamente em função do caso simulado e dos principais parâmetros de cálculos utilizados (no caso o fator de ajuste "k" e o período para controles constantes "d", em que d=0 significa sinais de controle livres e d=10 significa sinais de controle constantes por 10 dias).

2.2 Definições, Conceitos e Classificações

São apresentados, a seguir, os fundamentos teóricos que serviram de base para a pesquisa, cujas fontes foram não somente as referências bibliográficas apresentadas, mas também notas de aula das disciplinas cursadas na graduação e na pós-graduação.

Inicialmente, é conveniente definir conceitos importantes no estudo do controle ótimo de sistemas, conforme apresentados nos textos sobre o tema, (OGATA, 1993), (GOLNARAGHI; KUO, 2009), (FRANKLIN; POWELL, 1980), (FRANKLIN et al., 1991).

Definição 2.1. ***Sistema****: É a combinação de componentes que agem em conjunto no desempenho de uma função que não pode ser obtida se uma ou mais partes for retirada do conjunto.*

Esta definição enfatiza a importância da função implementada pelo sistema para a correta identificação dos seus componentes. Esta função será a “lupa” que ampliará ou reduzirá o foco sobre os componentes atuando em conjunto para delimitar o sistema nas suas interações com o ambiente.

Outra consideração é que a definição de sistema não se limita a algo físico, mas abrange

sistemas biológicos, econômicos, de informação e outros.

Uma definição alternativa e simples é que "sistema corresponde à uma parte do universo que interage com o meio ambiente, havendo fluxo de informação, matéria e/ou energia entre ambos".

O controle de um sistema objetiva assegurar que este "fluxo" entre o sistema e o meio ambiente obedeça a determinados limites, restrições ou requisitos previamente especificados.

Definição 2.2. ***Planta****: Corresponde a uma parte ou mesmo um conjunto de itens ou componentes de uma máquina ou equipamento, que operam conjuntamente com a finalidade de desempenhar uma operação.*

A planta é algo essencialmente físico, que geralmente é controlado na execução da operação. Exemplos de plantas são fornos, caldeiras, reatores, e máquinas como veículos, aviões, etc.

O controle de uma planta, em geral, tem o objetivo de assegurar o desempenho da função nas condições e restrições pertinentes a cada requisito da planta. Eventualmente podem existir requisitos de operação nominal dos componentes, de economia, de segurança, etc.

Definição 2.3. ***Processo****: É uma operação natural ou artificial que evolui progressivamente por mudanças graduais sucessivas, conduzindo a um resultado ou finalidade particular.*

Simplificadamente, entende-se como processo a operação a ser controlada.

Definição 2.4. ***Controle****: É um sistema externo que tem a função específica de manter o funcionamento do sistema controlado dentro das especificações para a sua função.*

O controle assegura que o referido fluxo trocado entre o sistema controlado e o meio ambiente citado na definição 2.1 obedece aos requisitos previamente definidos. Neste caso, o controle é um sistema que tem a função específica de controlar outro sistema.

Vale observar que do ponto de vista do sistema controlado, o controle pertence ao meio ambiente, o controle é externo.

Outra definição para controle é a própria função de controlar. Neste caso o termo controle representa uma função, não um sistema.

Definição 2.5. ***Controlador****: É o objeto que tem a função de controlar um outro ente, seja este ente outro sistema, processo ou planta.*

O controlador, assim como a planta, tem característica essencialmente física.

Definição 2.6. ***Variável Controlada****: Corresponde à grandeza ou condição que é medida e controlada pelo controle.*

Esta grandeza pode estar associada a uma das grandezas integrantes do fluxo de saída do sistema (geralmente denotada por 'y') ou a uma grandeza ou condição interna do sistema que necessita ser controlada (geralmente denotada por 'x').

Definição 2.7. ***Variável Manipulada***: *Corresponde à grandeza ou condição sobre a qual o controle atua, alterando seu valor, de forma a afetar o valor da variável controlada.*

Esta grandeza ou condição está associada ao fluxo de entrada no sistema (geralmente denotada por 'u').

Definição 2.8. ***Distúrbio, Perturbação ou Ruído***: *Corresponde a variações espúrias e consequentemente não previstas nas entradas do sistema ('u') ou no seu estado interno ('x'), que afetam adversamente uma ou mais variáveis controladas.*

Nesta definição, considera-se que distúrbios previsíveis e conhecidos podem ser compensados dentro do sistema.

Definição 2.9. ***Sistema de Controle de Malha Aberta*** : *É aquele em que as saídas do sistema não tem qualquer efeito sobre a ação de controle. Espera-se que as saídas do sistema estejam dentro dos valores nominais, mas o controlador atua no sistema para que este opere com base em calibrações ou temporização (figura 1).*

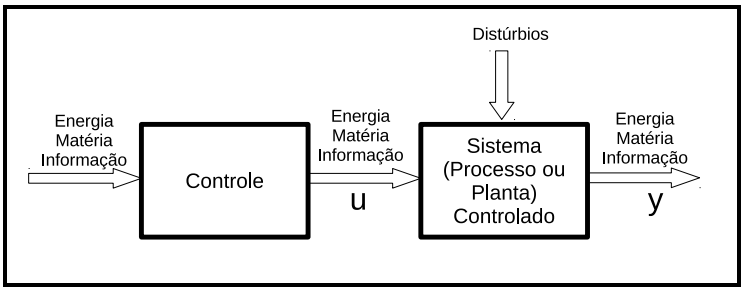

Figura 1: Sistema de Controle de Malha Aberta. Fonte: o autor.

Definição 2.10. ***Sistema de Controle de Malha Fechada (ou Realimentado)*** : *É aquele que possui uma relação entre uma ou mais variáveis controladas e as respectivas referências (figura 2).*

As definições a seguir abordam diversas classificações de sistemas.

Definição 2.11. ***Sistema Contínuo***[1]: *Um sistema é chamado de contínuo quando os valores de suas grandezas são definidas em tempo contínuo e são contínuas em amplitude, podendo assumir qualquer valor intermediário dentro de uma faixa de operação, conforme a figura 3.*

[1] As figuras 3 a 6 são usuais nas aulas teóricas sobre controle. Mesmo pesquisando, não conseguimos encontrar na literatura uma fonte específica destas figuras, contudo, (GOLTEN; VERWER, 1992, pp 241.) apresenta uma interessante figura descritiva dos sinais que caracterizam estes sistemas.

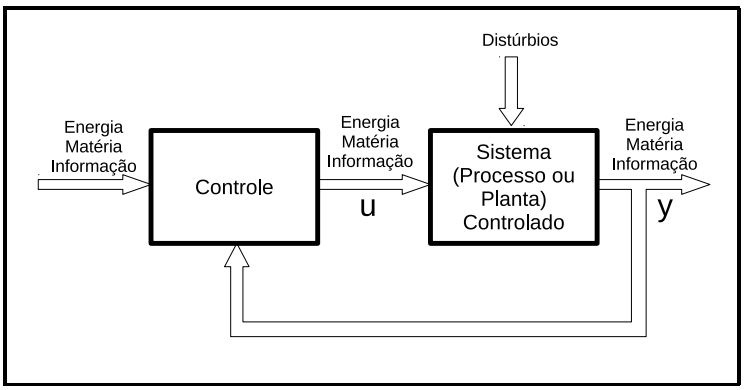

Figura 2: Sistema de Controle Realimentado. Fonte: o autor.

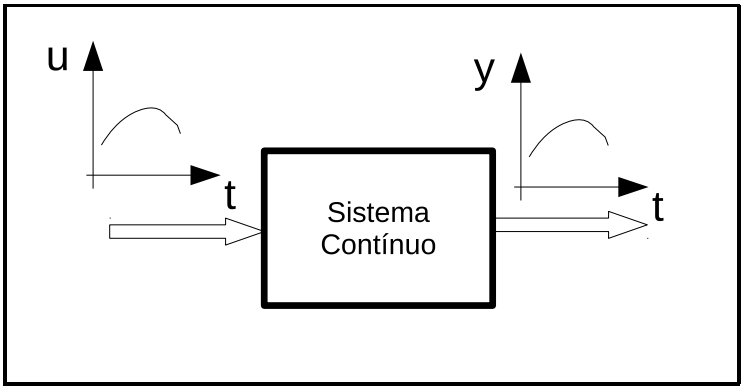

Figura 3: Sistema Contínuo.

Definição 2.12. ***Sistemas Quantizados de Tempo Contínuo (ou Sistema Quantizado)****: Corresponde ao sistema que é quantizado em amplitude, mas não no tempo. Suas grandezas podem assumir apenas alguns valores de amplitudes, mas podem ser observadas em qualquer instante no tempo, como na figura 4.*

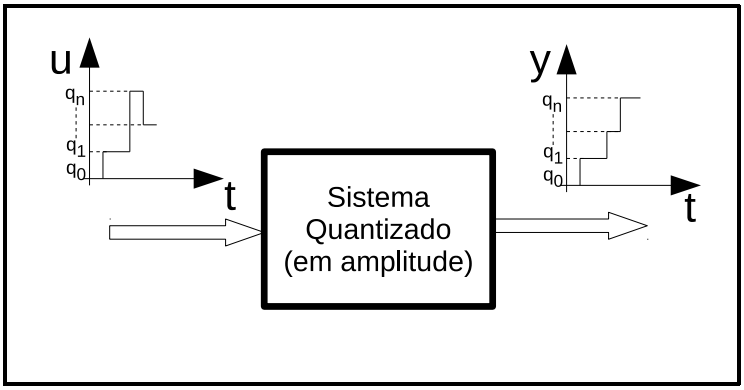

Figura 4: Sistema Quantizado.

Definição 2.13. ***Sistemas de Tempo Discreto****: Corresponde a um sistema amostrado, ou seja, ele possui grandezas que são observadas ou definidas apenas em instantes específicos no tempo (figura 5).*

O importante nestes sistemas é que o período entre estes instantes vai definir o período de amostragem.

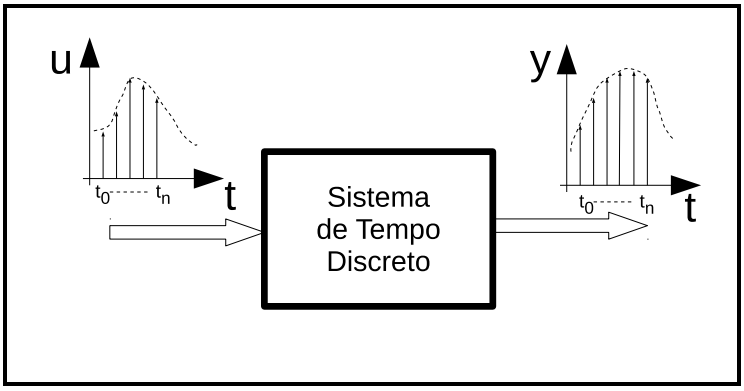

Figura 5: Sistema de Tempo Discreto.

Definição 2.14. ***Sistemas Quantizados de Tempo Discreto (ou Sistemas Digitais)****: São aqueles em que as grandezas são quantizadas em amplitude e são definidas apenas em instantes específicos no tempo (figura 6).*

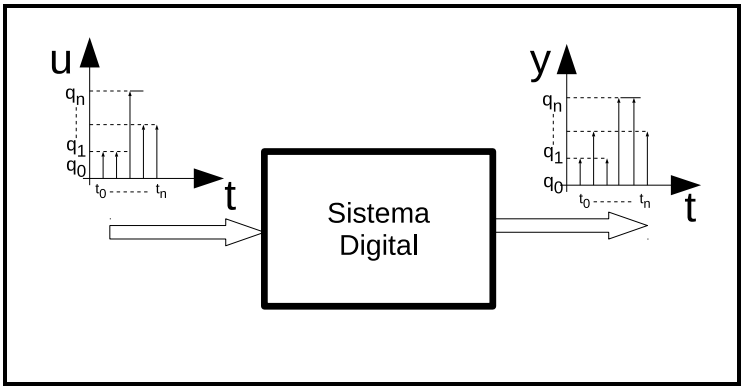

Figura 6: Sistema Digital.

Os sistemas contínuos são definidos por equações diferenciais, já os sistemas discretos são definidos por equações discretas ou a diferenças.

Os sistemas também podem ser classificados quanto ao número de variáveis de entrada e de saída, como a seguir:

Definição 2.15. ***Sistema SISO (Single Input Single Output)****: É aquele que possui uma única entrada e uma única saída.*

Definição 2.16. ***Sistema MIMO (Multiple Input Multiple Output)****: É aquele que possui múltiplas entradas e múltiplas saída.*

Outra classificação considera a previsibilidade das saídas.

Definição 2.17. ***Sistema Determinístico****: É aquele em que, se forem mantidas as mesmas condições iniciais do sistema, a repetição da entrada implicará a mesma saída.*

Corolário 2.1. *A saída de um sistema determinístico num determinado instante pode ser prevista ao se conhecer as condições iniciais e a entrada no mesmo instante.*

Definição 2.18. ***Sistema Estocástico ou Não Determinístico****: É aquele em que mesmo quando mantidas as mesmas condições do sistema, a repetição da entrada não garante a mesma saída, sendo, em geral, diferente.*

Neste sistema, as variáveis são caracterizadas em termos probabilísticos como média, variança, desvio padrão, etc.

O termo "realização", no contexto de sistema estocástico, refere-se a uma das possíveis evoluções do processo estocástico.

Outras classificações consideram a memória do sistema:

Definição 2.19. ***Sistema Instantâneo ou Sem Memória****: É aquele em que, mantidas as condições iniciais, as saídas dependem apenas dos valores das entradas no mesmo instante.*

Os sistemas instantâneos são regidos por equações algébricas:

$$y(t) = f(u(t)) \tag{1}$$

A função 'f' em (1) corresponde a uma expressão algébrica.

Definição 2.20. ***Sistema com Memória ou Dinâmico****: É aquele em que a saída depende não apenas das condições iniciais e do valor da entrada no mesmo instante, mas também de valores passados da própria saída e/ou da entrada e, obviamente, das condições iniciais.*

Os sistemas dinâmicos são regidos por equações diferenciais. As condições iniciais do sistema constituem a memória inicial.

Definição 2.21. ***Sistema Concentrado****: É aquele em que os parâmetros dos componentes do sistema não dependem do espaço.*

Um exemplo de sistema concentrado é o conhecido sistema mecânico do tipo massa-mola, em que a mola possui massa desprezível.

Os sistemas concentrados são descritos por equações diferenciais ordinárias.

Definição 2.22. ***Sistema Distribuído****: É aquele em que os parâmetros de pelo menos um componente do sistema depende do espaço.*

Um exemplo de sistema distribuído é o conhecido sistema mecânico do tipo massa-mola, em que a mola possui massa considerável.

Os sistemas distribuídos são descritos por equações diferenciais parciais.

Definição 2.23. ***Sistemas Relaxados****: É aquele em que as condições iniciais do sistema são nulas ou matematicamente não interferem nas saídas do sistema.*

Estes sistemas são de especial interesse por permitirem a utilização de função de transferência.

Definição 2.24. ***Sistema Causal (Físico ou Não Antecipativo)****: É aquele em que a saída num determinado instante não depende de valores futuros da entrada, mas apenas do valor no mesmo instante e, eventualmente, de valores passados.*

Definição 2.25. ***Sistema Não Causal (Não Físico ou Antecipativo)***[2]*: É aquele em que a saída num determinado instante depende de algum valor futuro da entrada, não apenas do valor no mesmo instante ou de valores passados.*

Definição 2.26. ***Sistema Linear****: Definindo-se que sobre uma entrada u(t), o sistema opera uma transformação $T\{\bullet\}$, de forma que a saída é dada por:*

$$y(t) = T\{u(t)\} \tag{2}$$

Sistema Linear é aquele em que $T\{\bullet\}$ observa o princípio da superposição, isto é, aceita as propriedades da aditividade e da homogeneidade.

Propriedade 2.1. ***Aditividade****: Uma transformação $T\{\bullet\}$ obedece a propriedade da Aditividade se, para entradas $u_1(t)$ e $u_2(t)$ com as respectivas saídas $y_1(t) = T\{u_1(t)\}$ e $y_2(t) = T\{u_2(t)\}$, para a entrada $u(t) = u_1(t) + u_2(t)$ tem-se a saída:*

$$\begin{aligned} y(t) &= T\{u(t)\} \\ &= T\{u_1(t) + u_2(t)\} \\ &= T\{u_1(t)\} + T\{u_2(t)\} \\ y(t) &= y_1(t) + y_2(t) \end{aligned}$$

$$y(t) = y_1(t) + y_2(t) \tag{3}$$

Propriedade 2.2. ***Homogeneidade****: Uma transformação $T\{\bullet\}$ obedece a propriedade da Homogeneidade se, para a entrada $u_1(t)$ com a respectiva saída $y_1(t) = T\{u_1(t)\}$, multiplicando-*

[2] São de especial interesse em processamento de imagens.

se a entrada por um escalar α, $u(t) = \alpha * u_1(t)$, tem-se a saída:

$$\begin{aligned} y(t) &= T\{u(t)\} \\ &= T\{\alpha * u_1(t)\} \\ &= \alpha * T\{u_1(t)\} \\ y(t) &= \alpha * y_1(t) \end{aligned}$$

$$y(t) = \alpha * y_1(t) \tag{4}$$

Definição 2.27. ***Sistema Não Linear****: Sistema Não Linear é aquele em que o princípio da superposição não é observado, isto é, ou a propriedade da aditividade, ou a da homogeneidade, ou ambas não são atendidas pelo sistema.*

Definição 2.28. ***Sistema Invariante no Tempo****: Sistema invariante no tempo apresenta a mesma saída sempre que a mesma entrada for aplicada.*

São os sistemas cujas características não se alteram com o tempo.

Corolário 2.2. *Em um sistema invariante no tempo, um deslocamento no instante de aplicação da entrada implicará apenas o mesmo deslocamento temporal na saída.*

Definição 2.29. ***Modelo de um Sistema****: O modelo de um sistema é uma representação simplificada da realidade do sistema que tem por objetivo facilitar o seu estudo, a previsão e o controle do seu comportamento.*

Existem vários tipos de modelos para todos os tipos de sistema (sistemas físicos, biológicos, mecânicos,etc): modelos reais em escala, modelos elétricos de sistemas mecânicos, modelos computacionais.

Contudo, considera-se a mais aplicável forma de modelagem de um sistema a obtenção das equações (que em geral são equações diferenciais) que descrevem o comportamento deste sistema.

As classificações dos sistemas apresentadas anteriormente estão intrinsecamente relacionadas às características das equações para o modelo do referido sistema. Esta relação é tão forte, que sistemas diferentes que possuem as mesmas equações diferenciais são ditos sistemas análogos. Solucionar um problema de controle para um sistema significa solucionar para o outro também, visto que são definidos pelas mesmas equações.

Quando um sistema mecânico fica muito complexo, muitas vezes tenta-se simplificar a realidade física buscando um sistema elétrico equivalente (ou aproximado) de equações diferenciais conhecidas. Resolvidas as equações elétricas, volta-se para a solução do sistema mecânico.

Definição 2.30. ***Função de Transferência****: Seja um sistema SISO, linear e invariante no tempo (LIT), cuja entrada temporal é dada por u(t) e a saída é dada por y(t). Tal sistema é de*

especial interesse por possibilitar a aplicação da Transformada de Laplace L[•] na equação diferencial do sistema, resultando na chamada função de transferência, dada por:

$$G(s) = \frac{L[saída]}{L[entrada]} = \frac{L[y(t)]}{L[u(t)]} = \frac{Y(s)}{U(s)} \tag{5}$$

A expressão (5) é apenas definida para sistemas relaxados, ou seja, sistemas em que as condições iniciais são nulas.

Para sistemas SISO, LIT e discretos no tempo, temos condições análogas em que é utilizada a Transformada *Z[•]* para se definir a função de transferência discreta, conforme descrito na equação (10).

2.3 Discussão sobre Técnicas de Controle

2.3.1 Teoria de Controle Convencional

A literatura tradicional sobre a engenharia de controle (OGATA, 1993) apresenta a teoria de controle convencional como aplicável apenas a sistemas SISO e LIT, como definidos anteriormente.

Dentro desta perspectiva e ainda para sistemas contínuos, a abordagem da Teoria Convencional é pelo estudo da função de transferência, saindo do domínio do tempo (t) e trabalhando-se mais no domínio da frequência complexa (s) por intermédio da Transformada de Laplace.

Neste contexto, busca-se conhecer o comportamento do sistema, estudando a sua função de transferência nas situações de interesse, considerando a entrada, o controlador e a realimentação.

A estabilidade é o foco da teoria convencional. Mesmo que algumas técnicas rudimentares de otimização possam ser tentadas, o objetivo primordial da teoria convencional é garantir a operação estável do sistema.

Muitas técnicas consagradas são utilizadas nesta teoria: desde a parametrização do sistema, eventualmente pelo levantamento de uma função de transferência desconhecida pela resposta de uma entrada impulso, até o estudo da equação característica do sistema realimentado[3], que pelo critério de Routh pode-se verificar ou não a estabilidade do sistema.

A condição necessária e suficiente para estabilidade é que as raízes da equação característica[4] estejam no semiplano complexo esquerdo.

[3] A equação característica de um sistema realimentado incorpora não somente a função de transferência do sistema controlado, mas também a própria função de transferência do controlador, considerando-se o tipo de realimentação e o sinal de referência.

[4] As raízes da equação característica do sistema correspondem aos polos da função de transferência de malha

O projeto de controlador baseia-se em "ajustar" a função de transferência do controlador para posicionar "adequadamente" as raízes da equação característica de malha fechada no semiplano complexo esquerdo.

O termo "adequadamente" se refere aos objetivos secundários, mas muitas vezes importantes, que eventualmente são igualmente requisitos de projeto. Não é escopo deste trabalho, mas o posicionamento adequado destas raízes irá determinar características importantes do sistema como o coeficiente de amortecimento e frequência natural não amortecida em sistemas de segunda ordem, constituindo o que se entende como resposta transitória.

Também são utilizadas técnicas de análise do erro em regime estacionário.

Uma técnica muito consagrada é a conhecida como "Root Locus". Esta técnica, quando aplicada a alguns sistemas específicos que possuem ganho variável, estuda o posicionamento das raízes da equação característica do sistema em malha fechada, pela variação do ganho de zero a infinito. De uma forma simplificada, pode-se dizer a técnica "Root Locus" objetiva obter o ganho adequado aos requisitos do projeto pela observação do posicionamento das raízes.

2.3.2 Teoria de Controle Moderno

Esta teoria, diferentemente da convencional, pode ser aplicada a sistemas lineares e não lineares, invariantes no tempo, variantes no tempo e sistemas MIMO.

Outra consideração é que a Teoria de Controle Moderno é essencialmente aplicada no domínio do tempo.

Neste sentido, é importante apresentar algumas definições e conceitos relacionados à teoria de controle moderno.

Definição 2.31. ***Estado de um Sistema****: O estado de um sistema corresponde ao menor número de variáveis, chamadas de variáveis de estado, que tendo seus valores conhecidos num instante t_0, juntamente com os valores da entrada até um instante t_i posterior a t_0, determina por completo o comportamento do sistema para este instante t_i, qualquer que seja t_i posterior a t_0.*

Definição 2.32. ***Variáveis de Estado****: São as variáveis que constituem o menor conjunto de variáveis que determinam o estado do sistema. Se são necessárias no mínimo 'n' variáveis x_1 , x_2 ,..., x_n , para descrever por completo o sistema dinâmico, então este conjunto constitui as variáveis de estado deste sistema.*

Definição 2.33. ***Vetor de Estado****: Se considerarmos as n variáveis de estado, x_1 , x_2 ,..., x_n, definidas anteriormente, como as n componentes de um vetor $\boldsymbol{x}$ de dimensão n, então $\boldsymbol{x}$ é chamado de vetor de estado, e n é conhecido como a ordem do sistema.*

fechada.

Definição 2.34. ***Espaço de Estados***: *Se considerarmos o espaço n-dimensional, cujos eixos coordenados são definidos pelos eixos associados às variáveis de estado x_1, x_2 ,..., x_n, tal espaço n-dimensional constitui o chamado Espaço de Estados. Qualquer estado do sistema que é completamente definido pelos valores numéricos das variáveis de estado x_1, x_2 ,..., x_n, constitui um ponto no espaço de estados.*

Definição 2.35. ***Equações do Espaço de Estados***: *A abordagem da teoria moderna de controle por espaço de estados considera três tipos de variáveis do sistema: variáveis de entrada, variáveis de saída e variáveis de estado.*

Considerando a figura 3 para Sistema Contínuo, definem-se as equações de estado em função do vetor entrada $\boldsymbol{u}$*, de ordem r, do vetor saída* $\boldsymbol{y}$*, de ordem m, por:*

$$\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}, \boldsymbol{u}, t) \tag{6}$$

$$\boldsymbol{y}(t) = g(\boldsymbol{x}, \boldsymbol{u}, t) \tag{7}$$

A equação (6) é conhecida como Equação de Estado e a equação (7) é chamada de Equação de Saída.

Com $f : \mathbb{R}^n \times \mathbb{R}^r \to \mathbb{R}^n$ e $g : \mathbb{R}^n \times \mathbb{R}^r \to \mathbb{R}^m$.

Observação 2.1. *A simbologia $\dot{F}$ denota derivada em relação ao tempo, isto é: $\dot{F} \equiv \frac{dF}{dt}$*

Observação 2.2. *A simbologia F_t denota derivada parcial em relação à variável t, isto é: $F_t \equiv \frac{\partial F}{\partial t}$*

As funções 'f' e 'g', acima, não são necessariamente lineares, e o termo 't' indica que o sistema pode ser variante no tempo.

Se as equações (6) e (7) forem linearizadas nas proximidades do estado de operação do sistema, tem-se:

$$\dot{\boldsymbol{x}}(t) = A(t)\boldsymbol{x}(t) + B(t)\boldsymbol{u}(t) \tag{8}$$

$$\boldsymbol{y}(t) = C(t)\boldsymbol{x}(t) + D(t)\boldsymbol{u}(t) \tag{9}$$

Das equações (6) e (7), seguem as definições das matrizes A(t), B(t), C(t) e D(t):

Definição 2.36. ***Matriz de Estado***: *A matriz A(t) é conhecida como Matriz de Estado.*

Definição 2.37. ***Matriz de Entrada***: *A matriz B(t) é conhecida como Matriz de Entrada.*

Definição 2.38. ***Matriz de Saída****: A matriz C(t) é conhecida como Matriz de Saída.*

Definição 2.39. ***Matriz de Transmissão Direta****: A matriz D(t) é conhecida como Matriz de Transmissão Direta.*

Os comentários anteriores se referem essencialmente a sistemas contínuos. Contudo, guardados os devidos cuidados, muito do que foi desenvolvido e estudado para sistemas de tempo contínuo pode ser adaptado para sistemas de tempo discreto.

Definição 2.40. ***Função de Transferência Discreta****: Seja um sistema SISO, LIT e discreto no tempo em que, para a sequencia de entrada u(n) a correspondente saída é y(n). Nestas condições, com o auxílio da da Transformada Z[•], a função de transferência discreta é dada por:*

$$G(z) = \frac{Z[saída]}{Z[entrada]} = \frac{Z[y(n)]}{Z[u(n)]} = \frac{Y(z)}{U(z)} \tag{10}$$

A utilização de funções de transferência, o estudo das raízes da equação característica, a utilização de espaço de estados, etc, são técnicas que podem ser adequadamente adaptadas aos sistemas de tempo discreto.

Além destas técnicas, outras, como o mapeamento do plano **s** no plano **z**, técnicas de discretização de sistemas contínuos, conversão de sinal análogo-digital e digital-análogo, compensação, etc, são utilizadas no controle de sistemas de tempo discreto.

2.3.3 O Problema de Controle Ótimo de Tempo Contínuo

Existem várias possibilidades para se formular um problema de controle ótimo em tempo contínuo. Dependendo da abordagem, pode-se querer minimizar o tempo para, partindo-se de um estado $\boldsymbol{x}_0$, chegar a um estado $\boldsymbol{x}$.

Outra opção muito comum é o controle necessário para se minimizar uma função escalar que depende, por exemplo, das trajetórias dos estados e dos controles. Esta abordagem é sumarizada por (LEWIS, 1986, pp. 153) da seguinte forma:

Modelo do Sistema:

$$\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}, \boldsymbol{u}, t), \quad t \geq t_0, \quad \text{com } t_0 \text{ fixo.} \tag{11}$$

Onde valem: $\boldsymbol{x}(t) \in \mathbb{R}^n$, $\boldsymbol{u}(t) \in \mathbb{R}^m$ e $t \in [t_0, T]$. Obviamente a função f tem dimensão n.

Índice de desempenho (função objetivo):

$$J(t_0) = \Phi(x(T), T) + \int_{t_0}^{T} L(\boldsymbol{x}, \boldsymbol{u}, t)dt \tag{12}$$

Restrição do Estado Final:

$$\Psi(x(T),T) = 0 \tag{13}$$

É importante ressaltar as naturezas diferentes de Φ e Ψ. A expressão (13) representa uma restrição severa por uma equação identicamente nula, enquanto que Φ significa uma função escalar que é minimizada.

O controle ótimo é dado pelas relações:

Hamiltoniano:

$$H(\boldsymbol{x},\boldsymbol{u},t) \;=\; L(\boldsymbol{x},\boldsymbol{u},t) \;+\; \boldsymbol{\lambda}^T f(\boldsymbol{x},\boldsymbol{u},t) \tag{14}$$

Equação de Estado:

$$\dot{\boldsymbol{x}} = \frac{\partial H}{\partial \lambda} \;=\; f, \quad t \geq t_0 \tag{15}$$

Equação de coestado:

$$-\dot{\boldsymbol{\lambda}} \;=\; \frac{\partial H}{\partial x} \;=\; \frac{\partial f^T}{\partial x}\,\boldsymbol{\lambda} \;+\; \frac{\partial L}{\partial x}, \;\; t \leq T \tag{16}$$

Condição de Otimalidade:

$$0 \;=\; \frac{\partial H}{\partial u} \;=\; \frac{\partial f^T}{\partial u}\,\boldsymbol{\lambda} \;+\; \frac{\partial L}{\partial u} \tag{17}$$

Observação 2.3. *A simbologia* $[\bullet]^T$ *denota a transposição de* $[\bullet]$.

Observação 2.4. *O vetor* $\boldsymbol{\lambda} \in \mathbb{R}^n$ *é conhecido como multiplicador de Lagrange.*

2.3.3.1 Planta Linear Variante no Tempo com Índice de Desempenho Quadrático

O problema de controle ótimo conforme proposto em 2.3.3 é bastante generalizado e, consequentemente, de solução analítica muito difícil. Na prática os problemas reais são resolvidos com uma maior especificação das características da planta e da função objetivo.

Uma abordagem muito conhecida considera um estado final de referência $r(T)$ e o índice de desempenho quadrático em que (LEWIS, 1986, pp. 171):

$$J(t_0) = \frac{1}{2}(x(T)-r(T))^T S(T)(x(T)-r(T)) + \frac{1}{2}\int_{t_0}^{T} (\boldsymbol{x}^T Q\boldsymbol{x} + \boldsymbol{u}^T R\boldsymbol{u})dt \tag{18}$$

No caso tem-se $S(T) \geq 0$, $Q \geq 0$ e $R > 0$.

Isto significa que as matrizes S e Q são simétricas e semidefinidas positivas. Já a matriz R é simetrica e definida positiva.

A solução de (18), dentro da proposição inicial de sistema não linear e variante no tempo, é por métodos numéricos[5].

Uma simplificação do problema anterior considera a situação em que planta definida por (8) é linear, variante no tempo e o estado final é livre, isto é, $r(T) = 0$:

$$\dot{\boldsymbol{x}}(t) = A(t)\boldsymbol{x}(t) + B(t)\boldsymbol{u}(t)$$

Nestas condições, (LEWIS, 1986, pp. 181) fornece:

O controle ótimo é dado por:

$$\boldsymbol{u}(t) = -R^{-1}B^T\boldsymbol{\lambda}(t) \tag{19}$$

As equações de estado e de coestado são apresentadas no Sistema Hamiltoniano:

$$\begin{bmatrix} \dot{\boldsymbol{x}} \\ \dot{\boldsymbol{\lambda}} \end{bmatrix} = \begin{bmatrix} A & -BR^{-1}B^T \\ -Q & -A^T \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{\lambda} \end{bmatrix} \tag{20}$$

O encaminhamento para solução supõe uma relação linear entre o estado e o coestado, dada por:

$$\boldsymbol{\lambda}(T) = S(T)\boldsymbol{x}(T) \tag{21}$$

Tal suposição considera que a matriz $S(T)$ desconhecida será determinada, se existir, pela solução da conhecida Equação de Riccati, a seguir:

$$-\dot{S} = A^T S + SA - SBR^{-1}B^T S + Q, \;\; t \leq T \tag{22}$$

Nestas condições, o controle é dado por:

$$\boldsymbol{u}(t) = -R^{-1}B^T S\boldsymbol{x}(t) \tag{23}$$

Onde $S(t)$ é obtida resolvendo-se a Equação de Riccati reversamente no tempo.

O ganho de Kalman é definido por:

$$\boldsymbol{K}(t) = R^{-1}B^T S \tag{24}$$

[5]Este caso mais genérico é conhecido como Problema de Valor de Contorno de Dois Pontos (LEWIS, 1986, pp. 171).

A expressão final do controle é, portanto:

$$\boldsymbol{u}(t) = -\boldsymbol{K}(t)\boldsymbol{x}(t) \tag{25}$$

Este caso particular, conforme (LEWIS, 1986, pp. 190), é conhecido como Regulador Quadrático Linear Contínuo (*Continous Linear Quadratic Regulator* - LQR[6] [7]), em que o estado final é livre.

O caso mais genérico considera o problema de rastreamento de uma trajetória a ser percorrida pelo estado do sistema: trata-se do Rastreador Quadrático Linear Contínuo (*Continous Linear Quadratic Tracker* - LQT). A formulação e a solução deste caso são apresentadas em (LEWIS, 1986, pp. 223).

Alguns conceitos na Engenharia de Controle são especialmente relevantes que merecem ser destacados. Assim acontece com o Critério de Estabilidade de Lyapunov, o Princípio do Máximo de Pontryagin e o Princípio de Otimalidade de Bellman:

2.3.4 Critério de Estabilidade de Lyapunov

A concepção da estabilidade segundo Lyapunov considera um sistema inicialmente sem a atuação do controle.

Segundo (OGATA, 1993), o primeiro método de Lyapunov necessita das soluções explícitas das equações diferenciais para a análise da estabilidade. O segundo método não necessita das soluções e estabelece:

Teorema 2.1. *Supondo o sistema:*

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x},t), \quad \textit{em que } f(\boldsymbol{0},t) = \boldsymbol{0} \textit{ para todo } t. \tag{26}$$

Se houver uma função escalar $V(\boldsymbol{x},t) \in C^1$, isto é, tendo primeiras derivadas parciais e contínuas e satisfazendo as condições,

- *$V(\boldsymbol{x},t)$ é definida positiva; e*
- *$\dot{V}(\boldsymbol{x},t)$ é definida negativa .*

Então o estado de equilíbrio na origem é estável.

[6] A formulação matemática do LQR é clássica na teoria do controle ótimo. O interesse em apresentar esta formulação é devido ao índice de desempenho do PCO principal escolhido para as simulações, o do problema da transmissão da dengue, se aproximar muito mas sem chegar a ser um LQR. Esta pequena diferença é justamente o que tornou o problema interessante e ensejou maiores discussões teóricas.

[7] Outro aspecto que torna o LQR muito importante é a sua solução totalmente analítica.

2.3.5 Princípio do Mínimo (ou do Máximo) de Pontryagin

Considerando as condições definidas em 2.3.3 para o PCO em tempo contínuo, e também que o controle $\boldsymbol{u}(t)$ fica restrito a uma determinada região limitada, (LEWIS, 1986, pp. 252) aponta o Princípio do Máximo de Pontryagin como a substituição da condição de otimalidade em (17) por:

$$H(\boldsymbol{x}^*, \boldsymbol{u}^*, \boldsymbol{\lambda}^*, t) \leq H(\boldsymbol{x}^*, \boldsymbol{u}^* + \boldsymbol{\delta u}, \boldsymbol{\lambda}^*, t), \quad \text{para todo } \boldsymbol{\delta u}. \tag{27}$$

Observação 2.5. *A simbologia* $[\bullet]^*$ *denota valor ótimo.*

A expressão (27), segundo (LEWIS, 1986, pp. 252), sumariza o Princípio do Mínimo de Pontryagin da seguinte forma: "O Hamiltoniano deve ser minimizado em todos os possíveis $\boldsymbol{u}$ para valores ótimos do estado e do coestado".

2.3.6 Princípio de Otimalidade de Bellman

O Princípio de Otimalidade de Bellman é apresentado em (LEWIS, 1986, pp. 293):

> "Uma política de controle ótima tem a propriedade de, não importando quais sejam as decisões passadas que foram tomadas (em termos de controle), as decisões restantes precisam constituir uma política ótima em relação ao estado resultante das decisões anteriores".

2.3.7 Equação de Hamilton-Jacobi-Bellman

A obtenção da Equação de Hamilton-Jacobi-Bellman é demonstrada em (LEWIS, 1986, pp. 311) e percorre os seguintes passos:

- parte-se das mesmas condições definidas em 2.3.3 para o PCO em tempo contínuo;
- supõe-se o tempo atual t e um tempo futuro próximo de t dado por $t + \Delta t$;
- com base no Princípio de Bellman descrito em 2.3.6 obtém-se $J^*(\boldsymbol{x}, t)$;
- expande-se $J^*(\boldsymbol{x}, t)$ em séries de Taylor e aplica-se condições de contorno, chegando-se à HJB.

$$-\frac{\partial J^*}{\partial t} = min_{\boldsymbol{u}} \left(H(\boldsymbol{x}, \boldsymbol{u}, J_x^*), t\right). \tag{28}$$

2.4 Conceitos Afetos à Otimização

Referências importantes da área, (LUENBERGER, 1984), (BAZARAA; SHETTY, 1979) e (BAZARAA et al., 2009), apresentam os seguintes conceitos:

2.4.1 O Problema Restrito de Otimização Não Linear

Segundo (LUENBERGER, 1984, pp. 4), o problema geral de programação matemática pode ser formulado como:

$$\begin{aligned} &\text{Minimizar:} && f(\boldsymbol{x}) && && (29a)\\ &\text{Sujeita a:} && h_i(\boldsymbol{x}) = 0 && i = 1,\ldots,m && (29b)\\ && && g_j(\boldsymbol{x}) \leq 0 && j = 1,\ldots,r && (29c)\\ && && \boldsymbol{x} \in S && && (29d) \end{aligned}$$

Nesta formulação, $\boldsymbol{x}$ é o vetor n-dimensional $(x_1, x_2, \ldots, x_n)$, e $f : \mathbb{R}^n \to \mathbb{R}$, $h_i : \mathbb{R}^n \to \mathbb{R}$ e $g_j : \mathbb{R}^n \to \mathbb{R}$ são funções reais. O espaço S é um subconjunto de $\mathbb{R}^n$.

As equações de igualdade (29b), de desigualdade (29c) e as limitações ao domínio (29d) constituem as restrições do problema de otimização.

Em geral, os problemas tratam com funções contínuas e que possuem primeiras derivadas contínuas[8]. O conjunto S é do tipo conexo, isto é, não é constituído de pontos ou partes isoladas.

Os problemas de otimização são classificados quando à escala por (LUENBERGER, 1984) da seguinte forma:

Definição 2.41. *problema de otimização de pequena escala: quando possuírem 5 ou menos incógnitas/restrições.*

Definição 2.42. *problema de otimização de média escala: quando possuírem de 6 até 100 incógnitas/restrições.*

Definição 2.43. *problema de otimização de grande escala: quando possuírem mais 100 incógnitas/restrições.*

As soluções para os problemas de otimização são obtidas normalmente com a utilização de métodos numéricos.

Uma heurística para obter a solução ótima é:

1. partir de uma solução inicial factível, isto é, que atende às restrições, $\boldsymbol{x}_0$;

[8] Mesmo que muitos problemas práticos não sejam assim.

2. aplica-se o algoritimo definido pelo método escolhido para se obter uma direção a percorrer e a distância a partir da solução anterior, em direção à próxima solução; e
3. verificam-se as condições de otimalidade, entre elas, se a solução atende à precisão necessária. Se foram atendidas, encontrou-se a solução ótima, se não foram, repete-se a iteração 2 a partir da solução encontrada.

Observação 2.6. *A simbologia $\boldsymbol{x}^*$ denota a solução ótima.*

Definição 2.44. *Direção Factível: Seja o ponto $\boldsymbol{x} \in S$. Diz-se o vetor $\boldsymbol{d}$ representa uma direção factível em $\boldsymbol{x}$ se:*

$$\exists\, \bar{\alpha} > 0 \mid \boldsymbol{x} + \alpha \boldsymbol{d} \in S \;\forall\; \alpha \leq \bar{\alpha} \tag{30}$$

2.4.2 Condições Necessárias de Primeira Ordem

A condição necessária de primeira ordem para que uma solução $\boldsymbol{x}^*$ seja um ponto de mínimo local é apresentada em (LUENBERGER, 1984, pp. 169) como:

Propriedade 2.3. *Seja S um subconjunto do $\mathbb{R}^n$ e seja $f \in C^1$ uma função em S. Se $\boldsymbol{x}^*$ é um ponto de mínimo local de $f \in C^1$ em S, então para qualquer vetor $\boldsymbol{d} \in \mathbb{R}^n$ que é uma direção factível em $\boldsymbol{x}^*$ tem-se:*

$$\nabla f(\boldsymbol{x}^*) \cdot \boldsymbol{d} \geq 0 \tag{31}$$

Observação 2.7. *A simbologia $[\cdot]$ denota o produto escalar.*

Observação 2.8. *A simbologia ∇ denota o operador Gradiente.*

Corolário 2.3. *A consequência da aplicação da propriedade 2.3 a um ponto a $\boldsymbol{x}^*$ interno S num problema irrestrito, isto é, sem equações de igualdade e de desigualdade, é que para este ponto existirão direções factíveis $\boldsymbol{d}$ [9] para todos os lados, de forma que (31) apenas se realiza na nulidade, isto é, quando:*

$$\nabla f(\boldsymbol{x}^*) = \boldsymbol{0} \tag{32}$$

2.4.3 Condições Necessárias de Segunda Ordem

As condições necessárias de segunda ordem para que uma solução $\boldsymbol{x}^*$ de um problema irrestrito seja um ponto de mínimo local é apresentada em (LUENBERGER, 1984, pp. 174) como:

[9] As diversas direções do vetor $\boldsymbol{d}$ em sentidos diferentes forçariam a consequente variação de sinal da expressão (31) em caso de ambos os fatores serem não nulos.

Propriedade 2.4. *Seja S um subconjunto do $\mathbb{R}^n$ e seja $f \in C^2$ uma função em S. Se $\boldsymbol{x}^*$ é um ponto de mínimo local de $f \in C^2$ em S, então para qualquer vetor $\boldsymbol{d} \in \mathbb{R}^n$ que é uma direção factível em $\boldsymbol{x}^*$ tem-se:*

- $\nabla f(\boldsymbol{x}^*) \cdot \boldsymbol{d} \geq 0$ (33)
- *se* $\nabla f(\boldsymbol{x}^*) \cdot \boldsymbol{d} = 0$ *, então* $\boldsymbol{d}^T \nabla^2 f(\boldsymbol{x}^*) \cdot \boldsymbol{d} \geq 0$ (34)

Observação 2.9. *A simbologia ∇^2 denota a matriz Hessiana.*

Corolário 2.4. *A consequência da aplicação da propriedade 2.4 a um ponto a $\boldsymbol{x}^*$ interno S num problema irrestrito, isto é, sem equações de igualdade e de desigualdade, é que para este ponto existirão direções factíveis $\boldsymbol{d}$ [10] para todos os lados, de forma que as expressões na propriedade 2.4 apenas se realizam na nulidade, isto é, forçando:*

- $\nabla f(\boldsymbol{x}^*) = \boldsymbol{0}$ (35)
- $\nabla^2 f(\boldsymbol{x}^*) \geq \boldsymbol{0}$ *, isto é,* $\nabla^2 f(\boldsymbol{x}^*)$ *é definida positiva.* (36)

Nestas condições $\boldsymbol{x}^*$ é um ponto de mínimo local de $f(\boldsymbol{x}^*)$.

2.4.4 Critérios de KKT

Até agora foram apresentadas condições de otimalidade para a situação específica do problema de otimização irrestrito. A situação mais geral descrita em 2.4.1, em que existem as restrições de igualdade e de desigualdade, foi estudada pelos matemáticos Karush, Kuhn e Tucker e as condições de otimalidade sumarizadas nas chamadas condições de KKT.

Definição 2.45. *Considerando o problema de otimização restrito apresentado em 2.4.1, define-se a função Lagrangiana como:*

$$L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f(\boldsymbol{x}) + \sum_{j=1}^{r} \lambda_j g_j(\boldsymbol{x}) + \sum_{i=1}^{m} \mu_i h_i(\boldsymbol{x}) \tag{37}$$

As condições de KKT são apresentadas em (BAZARAA; SHETTY, 1979, pp. 511) como:

Propriedade 2.5. *Considerando o problema de otimização restrito apresentado em 2.4.1, o ponto $\boldsymbol{x}^* \in S$ será um ponto de mínimo local se atender as condições:*

- *KKT1* $\quad \lambda_j \geq 0,\ j = 1, .., r$
- *KKT2* $\quad g_j(\boldsymbol{x}^*) \geq 0,\ j = 1, .., r; \quad h_i(\boldsymbol{x}^*) = 0,\ i = 1, .., m$

[10] As diversas direções do vetor $\boldsymbol{d}$ em sentidos diferentes forçariam variações de sinal das expressões na propriedade 2.4 em caso de ambos os fatores serem não nulos.

- *KKT3* $\lambda_j g_j(\boldsymbol{x}^*) = 0,\ j = 1,..,r$
- *KKT4* $\frac{\partial}{\partial x_k} L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = 0,\ k = 1,..,n$

Observação 2.10. *A condição KKT4 da propriedade 2.5 pode ser escrita como:*

$$\nabla f(\boldsymbol{x}^*) + \sum_{j=1}^{r} \lambda_j \nabla g_j(\boldsymbol{x}^*) + \sum_{i=1}^{m} \mu_i \nabla h_i(\boldsymbol{x}^*) = \mathbf{0} \tag{38}$$

Observação 2.11. *Os vetores $\boldsymbol{\lambda}$ e $\boldsymbol{\mu}$ são incógnitas do problema de otimização que são definidos no processo para se encontrar a solução ótima. São conhecidos como Multiplicadores de Lagrange.*

O próximo capítulo apresenta uma discussão sobre o aplicação das técnicas de otimização na engenharia de controle.

3 TÉCNICAS DE OTIMIZAÇÃO APLICADAS AO CONTROLE

O estudo dos trabalhos em que há aplicação prática dos conceitos de controle ótimo vistos no capítulo anterior evidencia uma forte dependência do modelo da planta e das funções integrantes do índice de desempenho adotado.

Em especial, como veremos a seguir, esta dificuldade é sentida ao aplicar o PMP na solução numérica de um PCO com modelagem do sistema, por exemplo, em linguagem C ou Fortran.

3.1 Aplicação direta do Princípio do Máximo de Pontryagin

A aplicação direta do Princípio do Máximo de Pontryagin para solucionar um PCO apresenta grande dificuldade de implementação, principalmente pela característica dinâmica intrínseca a todo PCO.

O PMP afirma que o valor do Hamiltoniano (14) na condição ótima é mínimo, e esta propriedade, num problema de otimização estático, seria utilizada para se obter as restrições sobre o estado e o coestado que determinarão a solução ótima (LEWIS, 1986, pp. 252).

Acontece que as EDO do modelo representam a variação temporal das incógnitas do Problema de Otimização pelo período considerado. Isto significa que resolver um PCO num período de tempo $t \in [0, T]$ é equivalente a resolver infinitos problemas de otimização estáticos, um para cada instante de tempo (MAGNUSSON; ÅKESSON, 2012).

Vimos em (THOMÉ, 2007) e (THOMÉ et al., 2010) a abordagem pela aplicação continuada do PMP, resolvendo-se sucessivos problemas de otimização espaçados no tempo por um intervalo Δt, por um método numérico para solucionar um problema de valores de contorno de dois pontos (KELLER, 1976).

Uma das maiores dificuldades dessa abordagem se encontra na implementação computacional específica, isto é, o modelo do sistema, bem como as restrições e o índice de desempenho são codificados em linguagem C ou Fortran, diretamente pelo pesquisador. Isto demandará ao pesquisador um trabalho adicional para, por exemplo, realizar variações de parâmetros, alterar consideravelmente o seu modelo ou aplicar diretamente o PMP a um sistema totalmente diferente. Na técnica estudada em nossa pesquisa, o processo da codificação do modelo é extremamente simples e a aplicação da técnica, igualmente, é simplificada, cabendo ao programador se concentrar na definição do modelo, não em sua codificação, e no tratamento dos dados.

A implementação computacional específica, por outro lado, é característica positiva porque

espera-se que um código fonte em linguagem C, C++ ou Fortran, após compilado em um executável, forneça resultados mais rapidamente que ferramentas distintas em um ambiente integrado de modelagem e simulação.

3.2 CONTROLE POR PREDIÇÃO DO MODELO

O Controle por Predição do Modelo é descrito em (ALLGÖWER; FINDEISEN, 2002) e (ALLGÖWER et al., 2004) como a técnica para se levar um sistema de um estado inicial $\boldsymbol{x}_0$ a um estado de referência $\boldsymbol{x}_s$ e um controle final de referência $\boldsymbol{u}_s$.

O princípio de operação de um controlador MPC é baseado em dois horizontes de tempo. Conhecendo-se o estado e o controle num instante t, o controlador objetiva atingir um estado final $\boldsymbol{x}_s$ num intervalo de tempo T_P, chamado de *horizonte de predição*, aplicando um controle $\bar{\boldsymbol{u}}$ por um período T_C, chamado de *horizonte de controle*. Tem-se $T_C \leq T_P$.

A figura 7 mostra a evolução temporal de um MPC.

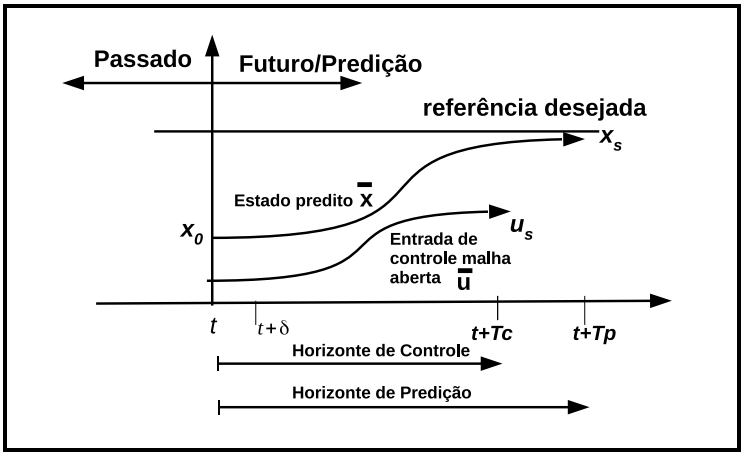

Figura 7: Evolução Temporal de um MPC.(ALLGÖWER; FINDEISEN, 2002)

Na realidade, a partir do instante t e repetidamente a cada instante $t + i * \delta,\ i \in [1,2,...]$, a malha de controle é "fechada", atualizando-se as informações do estado da planta $\boldsymbol{x}$. Com base nestas informações atualizadas, o MPC estima os estados futuros $\bar{\boldsymbol{x}}$ e os controles ótimos necessários $\bar{\boldsymbol{u}}$ para se atingir o estado $\boldsymbol{x}_s$ ao final do horizonte de predição e um controle $\boldsymbol{u}_s$ ao final do horizonte de controle.

Observação 3.1. *A simbologia $\bar{\boldsymbol{x}}$ denota variável estimada pelo MPC, utilizada no processo de otimização e, devido a ruídos e interferências, com um erro em relação aos valores reais da planta.*

O esquema geral do processo de predição do modelo é apresentado na figura 8. As referências citadas tratam em especial de um MPC não linear.

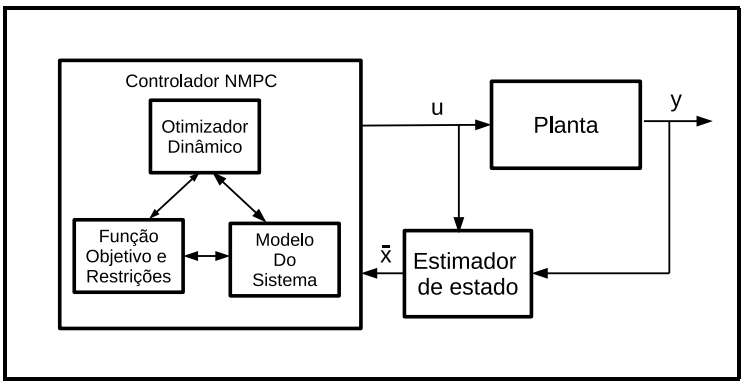

Figura 8: Sistema de Controle NMPC.(ALLGÖWER; FINDEISEN, 2002)

A formulação matemática apresentada em (ALLGÖWER; FINDEISEN, 2002) considera um sistema não linear definido por:

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}(t), \boldsymbol{u}(t)), \quad \boldsymbol{x}(0) = \boldsymbol{x}_0, \tag{39}$$

sujeito a uma entrada limitada do tipo:

$$\boldsymbol{u}(t) \in U, \quad \boldsymbol{x}(t) \in X, \quad \forall t \geq 0 \tag{40}$$

Assume-se $U \subset \mathbb{R}^r$ e $X \subset \mathbb{R}^n$:

$$U \equiv \{\boldsymbol{u} \in \mathbb{R}^r | \boldsymbol{u}_{min} \leq \boldsymbol{u} \leq \boldsymbol{u}_{max}\}, X \equiv \{\boldsymbol{x} \in \mathbb{R}^n | \boldsymbol{x}_{min} \leq \boldsymbol{x} \leq \boldsymbol{x}_{max}\}.$$

Observação 3.2. *Assume-se que o sistema definido em 39 possui, consideradas as condições iniciais, uma única solução para uma entrada específica definida pela função*

$$\boldsymbol{u}(\bullet) : [0, T_p] \rightarrow U.$$

Assim, o MPC é descrito como um PCO:

$$\begin{aligned}
&\text{Min:} \quad J(\boldsymbol{x}(t), \bar{\boldsymbol{u}}(\bullet); T_C, T_P), && \text{(41a)}\\
&\text{em relação a:} \quad \bar{\boldsymbol{u}}(\bullet), && \text{(41b)}\\
&\text{onde} \quad J(\boldsymbol{x}(t), \bar{\boldsymbol{u}}(\bullet); T_C, T_P) \equiv \int_t^{t+T_P} F(\bar{\boldsymbol{x}}(t), \bar{\boldsymbol{u}}(t)) d\tau, && \text{(41c)}\\
&\text{sujeita a:} \quad \dot{\bar{\boldsymbol{x}}}(\tau) = \boldsymbol{f}(\bar{\boldsymbol{x}}(\tau), \bar{\boldsymbol{u}}(\tau)), \quad \bar{\boldsymbol{x}}(\tau) = \boldsymbol{x}(t), && \text{(41d)}\\
&\qquad\qquad \bar{\boldsymbol{u}}(\tau) \in U, \ \forall \tau \in [t, t+T_C], && \text{(41e)}\\
&\qquad\qquad \bar{\boldsymbol{u}}(\tau) = \bar{\boldsymbol{u}}(t+T_C), \ \forall \tau \in [t+T_C, t+T_P], && \text{(41f)}
\end{aligned}$$

$$\bar{\boldsymbol{x}}(\tau) \in X, \ \ \forall t \in [t, t+T_P]. \tag{41g}$$

Observação 3.3. *O sinal ";" entre os argumentos de uma função, como em $F(\boldsymbol{x};\phi)$ denota função F em $\boldsymbol{x}$ com o parâmetro ϕ.*

As implementações de MPC normalmente usam uma função quadrática em (41c), do tipo:

$$F(\bar{\boldsymbol{x}}(t), \bar{\boldsymbol{u}}(t)) = (\boldsymbol{x}-\boldsymbol{x}_S)^T Q(\boldsymbol{x}-\boldsymbol{x}_S) + (\boldsymbol{u}-\boldsymbol{u}_S)^T R(\boldsymbol{u}-\boldsymbol{u}_S) \tag{42}$$

Onde Q e R são matrizes de pesos simétricas, definidas positivas, com x_S e u_S sendo as referências finais para o estado e para o controle.

A solução do problema de otimização é denotada por:

$$\bar{\boldsymbol{u}}^*(\bullet; \boldsymbol{x}(t), T_P, T_C) \colon [t, t+T_P] \to U. \tag{43}$$

Na teoria do MPC, a expressão (43) constitui o chamado controle ótimo de malha aberta e é recalculada a cada instante $t = j\delta, \ j = 0, 1, ...$, assim que nova amostragem é feita.

O controle ótimo em malha fechada para o MPC é definido como o controle ótimo calculado nos instantes das amostragens, isto é:

$$\boldsymbol{u}^*(\tau) \equiv \bar{\boldsymbol{u}}^*(\tau; \boldsymbol{x}(t), T_P, T_C), \ \ \tau \in [t, \delta]. \tag{44}$$

A partir de (41a), a expressão do valor ótimo do problema de otimização com função apenas do estado pode ser definida como:

$$V(\boldsymbol{x}; T_C, T_P) = J(\boldsymbol{x}, \bar{\boldsymbol{u}}^*(\bullet; \boldsymbol{x}(t)); T_C, T_P). \tag{45}$$

A importância de (45) na teoria do NMPC é que, segundo (ALLGÖWER; FINDEISEN, 2002), esta expressão serve como candidata a função de Lyapunov (vide 2.3.4) no que tange à estabilidade do sistema de controle.

Tudo o que foi apresentado sobre MPC tratou de sistemas em tempo contínuo. Segundo (ALLGÖWER; FINDEISEN, 2002) a maioria dos conceitos descritos para tempo contínuo têm contrapartida na teoria para tempo discreto.

3.3 Otimização Dinâmica pelo Método de Colocação Direta

A otimização dinâmica pelo método de *colocação direta* (MAGNUSSON; ÅKESSON, 2012) foi demonstrada em aplicação num ambiente integrado de modelamento e simulação

baseado na linguagem Modelica.

O trabalho em referência citou, como aplicações diretas da otimização dinâmica pelo método de colocação, o *controle ótimo* e a *estimativa de parâmetros* de um modelo.

A aplicação em *controle ótimo* objetiva determinar as trajetórias das variáveis de controle, normalmente denotadas pelo vetor $\boldsymbol{u} \in \mathbb{R}^r$, que minimizam determinados recursos para se executar uma ação específica. Na aplicação em *estimativa de parâmetros*, a intenção é encontrar os valores dos parâmetros desconhecidos do modelo do sistema que possibilitem ao modelo se comportar de acordo com um conjunto de dados fornecidos.

Os problemas de controle ótimo geralmente são solucionados pela abordagem da aplicação direta do Princípio do Mínimo (ou do Máximo) de Pontryagin ou por programação dinâmica baseada no Princípio de Otimalidade de Bellman. O problema dessas abordagens é que, segundo (MAGNUSSON; ÅKESSON, 2012), elas são mal condicionadas para problemas de larga escala e possuem problemas com restrições de desigualdades.

Os métodos diretos[1] atuais usam a técnica de tratar o dinamismo do PCO por aproximação, transformando um problema de otimização de dimensão infinita num programa não linear (PNL) de dimensão finita. Uma das abordagens de método direto é conhecida por *cenários múltiplos* [2], outra, que é a descrita nesta seção, é a *colocação direta*.

A formulação matemática[3] do método de *colocação direta* inicia com a definição do sistema dinâmico alvo da otimização:

$$F(t, \dot{\boldsymbol{x}}(t), \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{w}(t), \boldsymbol{p}) = 0, \tag{46}$$

onde $t \in \mathbb{R}$ corresponde a única variável totalmente independente, o tempo, $\boldsymbol{x} \in \mathbb{R}^{n_x}$ corresponde ao vetor de estado, $\boldsymbol{u} \in \mathbb{R}^{n_u}$ é o vetor de controle a ser otimizado, no caso de aplicação em *controle ótimo*, $\boldsymbol{w} \in \mathbb{R}^{n_w}$ é o vetor que constitui a variável algébrica[4] e $\boldsymbol{p} \in \mathbb{R}^{n_p}$ é o vetor de parâmetros[5] a ser estimado, no caso de aplicação em *estimativa de parâmetros*.

[1] Segundo (MAGNUSSON; ÅKESSON, 2012) os métodos não diretos abordam o PCO por cálculo variacional.

[2] Do inglês *multiple shooting*.

[3] Nesta seção é descrita apenas a formulação matemática básica da Otimização Dinâmica pelo Método de Colocação Direta. As considerações sobre a aplicação com as ferramentas integradas de modelagem e simulação, JModelica (ÅKESSON et al., 2010), (MAGNUSSON; ÅKESSON, 2012), (ANDERSSON et al., 2011), e de otimização, IPOPT (WÄCHTER; BIEGLER, 2006), são apresentadas no próximo capítulo.

[4] Na literatura referenciada, (MAGNUSSON; ÅKESSON, 2012), não encontramos formulação detalhada para o vetor $\boldsymbol{w}$, contudo isto não ofereceu restrições para o entendimento e aplicação da técnica.

[5] O vetor de parâmetros $\boldsymbol{p}$, por constituir um valor fixo durante toda a aplicação da técnica, não constitui alvo de preocupação na formulação matemática. A sua importância está apenas na sua definição inicial, no caso de aplicação em controle ótimo, ou na sua determinação propriamente dita pela aplicação na solução de um problema de estimativa de parâmetros.

As condições iniciais são definidas por:

$$F_0(\dot{\boldsymbol{x}}(t_0), \boldsymbol{x}(t_0), \boldsymbol{u}(t_0), \boldsymbol{w}(t_0), \boldsymbol{p}) = 0, \tag{47}$$

onde t_0 é o instante inicial.

Para facilitar a notação, todas as variáveis dependentes do tempo são agrupadas numa única variável $\boldsymbol{z}$:

$$\boldsymbol{z} \equiv (\dot{\boldsymbol{x}}, \boldsymbol{x}, \boldsymbol{u}, \boldsymbol{w}). \tag{48}$$

A dinâmica do sistema é descrita resumidamente por:

$$F(t, \boldsymbol{z}(t), \boldsymbol{p}) = 0, \quad \forall t \in [t_0, t_f], \tag{49}$$

$$F_0(\boldsymbol{z}(t_0), \boldsymbol{p}) = 0, \tag{50}$$

onde t_f é o instante final e

$$F : \mathbb{R}^{n_z} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_x + n_w}, \tag{51}$$

$$F_0 : \mathbb{R}^{n_z} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_x}, \tag{52}$$

$$n_z = 2n_x + n_u + n_w. \tag{53}$$

De forma que o problema geral de otimização objeto da técnica de *colocação direta* é apresentado por (MAGNUSSON; ÅKESSON, 2012) como:

$$\begin{aligned}
&\text{Minimizar:} && f(t_0, t_f, \boldsymbol{z}, \boldsymbol{p}), && \text{(54a)}\\
&\text{em relação a:} && t_0, t_f, \boldsymbol{z}, \boldsymbol{p}, &&\\
&\text{sujeita a:} && F(t, \boldsymbol{z}(t), \boldsymbol{p}) = \boldsymbol{0}, && \text{(54b)}\\
& && F_0(\boldsymbol{z}(t_0), \boldsymbol{p}) = \boldsymbol{0}, && \text{(54c)}\\
& && \boldsymbol{z}_L \leq \boldsymbol{z}(t) \leq \boldsymbol{z}_U, && \text{(54d)}\\
& && \boldsymbol{p}_L \leq \boldsymbol{p} \leq \boldsymbol{p}_U, && \text{(54e)}\\
& && \boldsymbol{g}_e(t_0, t_f, t, \boldsymbol{z}(t), \boldsymbol{p}) = \boldsymbol{0}, && \text{(54f)}\\
& && \boldsymbol{g}_i(t_0, t_f, t, \boldsymbol{z}(t), \boldsymbol{p}) \leq \boldsymbol{0}, && \text{(54g)}\\
& && \boldsymbol{G}_e(t_0, t_f, \boldsymbol{Z}_e, \boldsymbol{p}) = \boldsymbol{0}, && \text{(54h)}\\
& && \boldsymbol{G}_i(t_0, t_f, \boldsymbol{Z}_i, \boldsymbol{p}) \leq \boldsymbol{0}, && \text{(54i)}\\
& && \forall t \in [t_0, t_f]. &&
\end{aligned}$$

A função a ser minimizada em (54a) é muito similar ao índice de desempenho (12), descrito na página 45. A diferença básica é que incorpora a parametrização, podendo ser, nos problemas de controle ótimo, o funcional de Bolza (MAGNUSSON; ÅKESSON, 2012):

$$f(t_0, t_f, \boldsymbol{z}, \boldsymbol{p}) = \Phi(t_0, \boldsymbol{z}(t_0), t_f, \boldsymbol{z}(t_f), \boldsymbol{p}) + \int_{t_0}^{t_f} L(t, \boldsymbol{z}(t), \boldsymbol{p}) dt, \tag{55}$$

em que $\Phi : \mathbb{R} \times \mathbb{R}^{n_z} \times \mathbb{R} \times \mathbb{R}^{n_z} \times \mathbb{R}^{n_p} \to \mathbb{R}$ é conhecido como termo de Mayer e $L : \mathbb{R} \times \mathbb{R}^{n_z} \times \mathbb{R}^{n_p} \to \mathbb{R}$ é conhecido como integrando de Lagrange ou Lagrangiano.

Para o caso de *estimativa de parâmetros*, a função descrita por (54a) e (55) é formulada usando uma soma quadrática dos erros em relação aos valores medidos, penalizando os desvios em relação aos valores medidos fornecidos. Para o caso de *controle ótimo*, a função pode apresentar certa versatilidade para melhor representar os conceitos que se deseja medir, em geral custos, especificos de cada modelo[6], mas certamente deverá obdecer determinadas restrições de continuidade e de diferenciabilidade que possibilite a solução do PCO.

A dinâmica do sistema e suas condições iniciais são tratadas pelas equações (54b) e (54c) do PCO. As equações (54d) e (54e) representam os limites das variáveis no horizonte de tempo, isto é, $[t_0, t_f]$.

Observação 3.4. *A simbologia $[\bullet]_L$ e $[\bullet]_U$ em (54d) e (54e) denotam respectivamente limite inferior e limite superior.*

Observação 3.5. *As simbologias $[\bullet]_e$ e $[\bullet]_i$ em (54f) a (54i) denotam respectivamente restrições de igualdade e de desigualdade.*

As equações (54f) e (54g) são restrições afetas ao *caminho* seguido pelo sistema. Um exemplo de aplicação destas restrições é quando um braço de robô deve seguir um determinado caminho num tempo mínimo.

Finalmente, as equações (54h) e (54i) representam restrições pontuais que o sistema deve obedecer em determinados instantes específicos de tempo. Os vetores $\boldsymbol{Z}_e$ e $\boldsymbol{Z}_i$ representam os valores das variáveis usadas nas restrições:

$$\boldsymbol{Z}_e = (\boldsymbol{Z}(T_1), \boldsymbol{Z}(T_2), ..., \boldsymbol{Z}(T_m)), \tag{56}$$

em que T_i representa o instante de tempo em que a restrição se aplica.

Outra restrição importante apresentada na formulação por (MAGNUSSON; ÅKESSON, 2012) é que $\boldsymbol{g}_e, \boldsymbol{g}_i, \boldsymbol{G}_e$ e $\boldsymbol{G}_i$ deverão ser continuamente diferenciáveis até a segunda derivada.

[6] No caso deste estudo, que será apresentado no capítulo 5, (THOMÉ, 2007) propôs considerar como valor o "benefício" de uma determinada variável.

3.3.1 Polinômios de Colocação

O método da *colocação direta*[7] aborda o PCO representado por (54a) a (54i) pela discretização do horizonte de tempo em n_e elementos. No interior de cada elemento de tempo i, a variável dependente do tempo $\boldsymbol{z}(t)$ é aproximada por interpolação dos pontos exatos $\boldsymbol{z}_i = (\dot{\boldsymbol{x}}(t), \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{w}(t), \boldsymbol{p})$ pelos chamados polinômios de colocação. Dentro do elemento de tempo i, o tempo é normalizado da seguinte forma:

$$\tilde{t}_i(\tau) = t_{i-1} + h_i.(t_f - t_0).\tau, \quad \tau \in [0,1], \quad \forall i \in [1,..,n_e], \tag{57}$$

onde t_i é o instante de tempo ao final do elemento i e h_i corresponde ao comprimento do elemento i e, como os elementos são iguais, tem-se:

$$h_i = \frac{t_f - t_0}{n_e}. \tag{58}$$

Os polinômios de colocação são formados pela escolha de n_c pontos de colocação, que são os mesmos no interior de cada elemento. Denota-se $\tau_{i,k}$ o "k-ésimo" ponto de colocação dentro do elemento i, e $\boldsymbol{z}_{i,k}$ o valor de $\boldsymbol{z}(\tau_{i,k})$. Como os estados necessitam ser contínuos nos limites dos elementos, é necessário um ponto adicional de interpolação no início de cada elemento, denotado por $\tau_{i,0} \equiv 0$.

Os polinômios de colocação são então definidos por:

$$\boldsymbol{x}_i(\tau) = \sum_{k=0}^{n_c} \boldsymbol{x}_{i,k}.\tilde{\ell}_k(\tau), \tag{59a}$$

$$\boldsymbol{u}_i(\tau) = \sum_{k=1}^{n_c} \boldsymbol{u}_{i,k}.\ell_k(\tau), \tag{59b}$$

$$\boldsymbol{w}_i(\tau) = \sum_{k=1}^{n_c} \boldsymbol{w}_{i,k}.\ell_k(\tau), \tag{59c}$$

onde $\tilde{\ell}_k$ e ℓ_k são polinômios base de Lagrange respectivamente com e sem interpolação no ponto $\tau_{i,0}$.

[7] Podemos destacar como semelhanças entre os métodos de *colocação direta* e o de *cenários múltiplos* o fato de ambos particionarem o intervalo de tempo e a necessidade de continuidade para o estado nas transições entre os intervalos. Contudo, sem sermos exaustivos, destacamos como diferença o fato de que na *colocação direta* as parametrizações e restrições do PCO são definidas para todo o período global de tempo. Isto não acontece no método de *cenários múltiplos*, em que para cada cenário (subintervalo) eventualmente podem ocorrer parametrizações e restrições específicas. Em suma, o particionamente do tempo no método de *colocação direta* serve para o objetivo de constituição de um novo PNL e no método de *cenários múltiplos* serve também para atender a cenários distintos por múltiplas restrições.

Os polinômios base são dados por:

$$\tilde{\ell}_k(\tau) = \prod_{l\in[0,..,n_c]\setminus\{k\}} \frac{\tau-\tau_l}{\tau_k-\tau_l}, \quad \forall k \in [0,..,n_c], \tag{60a}$$

$$\ell_k(\tau) = \prod_{l\in[1,..,n_c]\setminus\{k\}} \frac{\tau-\tau_l}{\tau_k-\tau_l}, \quad \forall k \in [1,..,n_c], \tag{60b}$$

A expressão para a variação do estado pode ser obtida derivando-se (59a):

$$\begin{aligned}\dot{\boldsymbol{x}}_i(\tau) &= \frac{d\boldsymbol{x}_i}{d\tilde{t}_i}(\tau) = \frac{d\tau}{d\tilde{t}_i}.\frac{d\boldsymbol{x}_i}{d\tau}(\tau) \\ &= \frac{1}{h_i(t_f-t_0)}.\sum_{k=0}^{n_c}\boldsymbol{x}_{i,k}.\frac{d\tilde{\ell}_k}{d\tau}(\tau).\end{aligned} \tag{61}$$

Uma representação da distribuição dos elementos e dos pontos de colocação é apresentada na figura 9.

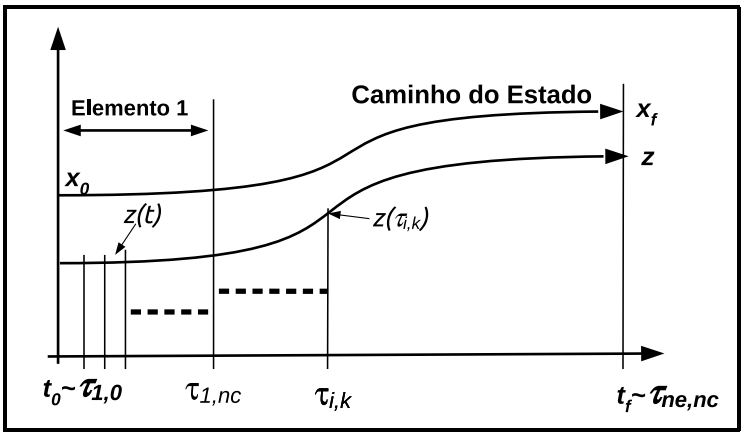

Figura 9: Representação dos Pontos de Colocação. Fonte: o autor.

3.3.2 *Transformação do Problema de Otimização*

A solução apresentada por (MAGNUSSON; ÅKESSON, 2012) para resolver o problema de otimização de dimensão infinita definido pelas equações de (54a) a (54i) é transformá-lo num PNL[8] de dimensão finita com o auxílio dos polinômios apresentados anteriormente. A ideia é que a variavel $\boldsymbol{z}$, contínua no tempo, possa ser representada por um número finito de valores, os valores dos pontos de colocação. Como variáveis para o PNL, (MAGNUSSON; ÅKESSON, 2012) definiu os pontos de colocação $\boldsymbol{z}_{i,k}$, os valores dos estados no início de cada elemento de tempo i, $\boldsymbol{x}_{i,0}$ e o vetor de parâmetor $\boldsymbol{p}$. Além destas, foram inseridas as condições iniciais das variáveis, $\boldsymbol{z}_{1,0}$, o tempo inicial t_0 e o tempo final t_f, de forma a definir-se a variável Z por:

[8] Programa Não Linear

$$\begin{aligned}\boldsymbol{Z} = &(\boldsymbol{z}_{1,0}, \boldsymbol{z}_{1,1}, \boldsymbol{z}_{1,2}, \ldots, \boldsymbol{z}_{1,n_c},\\ &(\boldsymbol{x}_{2,0}, \boldsymbol{z}_{2,1}, \boldsymbol{z}_{2,2}, \ldots, \boldsymbol{z}_{2,n_c},\\ &(\boldsymbol{x}_{3,0}, \boldsymbol{z}_{3,1}, \boldsymbol{z}_{3,2}, \ldots, \boldsymbol{z}_{3,n_c},\\ &\vdots \quad \cdots\\ &(\boldsymbol{x}_{n_e,0}, \boldsymbol{z}_{n_e,1}, \boldsymbol{z}_{n_e,2}, \ldots, \boldsymbol{z}_{n_e,n_c}, \boldsymbol{p}, t_0, t_f).\end{aligned} \tag{62}$$

O tipo de colocação refere-se à maneira como os pontos de colocação são distribuídos dentro de cada elemento de tempo i, conforme definido em 3.3.1.(MAGNUSSON; ÅKESSON, 2012) utilizadou a colocação Radau, que coloca um ponto de colocação ao fim de cada elemento de tempo i e distribui os demais de forma a maximizar a precisão dos cálculos.

Como resultado desta transformação o PNL fica:

$$\begin{aligned}
&\text{Minimizar:} && \tilde{f}(Z), && \text{(63a)}\\
&\text{em relação a:} && Z \in \mathbb{R}^{n_Z}, &&\\
&\text{sujeita a:} && F(t_{i,k}, \boldsymbol{z}_{i,k}, \boldsymbol{p}) = \boldsymbol{0}, && \text{(63b)}\\
& && F_0(\boldsymbol{z}_{1,0}, \boldsymbol{p}) = \boldsymbol{0}, && \text{(63c)}\\
& && \boldsymbol{u}_{1,0} - \sum_{k=1}^{n_c} \boldsymbol{u}_{1,k}.\ell_k(0) = \boldsymbol{0}, && \text{(63d)}\\
& && \boldsymbol{z}_L \leq \boldsymbol{z}_{i,k} \leq \boldsymbol{z}_U, && \text{(63e)}\\
& && \boldsymbol{p}_L \leq \boldsymbol{p} \leq \boldsymbol{p}_U, && \text{(63f)}\\
& && \boldsymbol{g}_e(t_{i,k}, \boldsymbol{z}_{i,k}, \boldsymbol{p}) = \boldsymbol{0}, && \text{(63g)}\\
& && \boldsymbol{g}_i(t_{i,k}, \boldsymbol{z}_{i,k}, \boldsymbol{p}) \leq \boldsymbol{0}, && \text{(63h)}\\
& && \boldsymbol{G}_e(\boldsymbol{Z}_e) = \boldsymbol{0}, && \text{(63i)}\\
& && \boldsymbol{G}_i(\boldsymbol{Z}_i) \leq \boldsymbol{0}, && \text{(63j)}\\
& && \forall (i,k) \in \{(1,0)\} \bigcup ([1..n_e] \times [1..n_c]), &&\\
& && \dot{\boldsymbol{x}}_{j,l} = \frac{1}{h_j(t_f - t_0)}. \sum_{m=0}^{n_c} \boldsymbol{x}_{j,m}.\frac{d\tilde{\ell}_m}{d\tau}(\tau_l), && \text{(63k)}\\
& && \forall (j,l) \in [1..n_e] \times [1..n_c], &&\\
& && \boldsymbol{x}_{n,n_c} = \boldsymbol{x}_{n+1,0}, \quad \forall n \in [1..n_e - 1]. &&
\end{aligned}$$

onde

$$n_Z = (1 + n_e.n_c).n_z + (n_e - 1).n_x + n_p + 2 \tag{64}$$

corresponde ao número de veriáveis do PNL e $t_{i,k} \equiv \tilde{t}_i(\tau_k)$ denota o "k-ésimo" ponto de colocação no elemento i.

Obviamente a função objetivo e o Lagrangiano são igualmente transformados de forma semelhante.

O relevante é que (MAGNUSSON; ÅKESSON, 2012) mostra que a solução do PNL disposto nas equações de (63a) a (63k) resulta na solução aproximada do PCO definido de (54a) a (54i).

Nas aplicações práticas com as ferramentas que serão apresentadas em detalhes no próximo capítulo, esta formulação teórica é tratada internamente pelas ferramentas e o usuário concentra o esforço na definição do modelo e das restrições do PCO de forma consistente.

4 FERRAMENTAS APLICADAS AO CONTROLE ÓTIMO

A seguir, são apresentadas algumas ferramentas importantes para o estudo do controle ótimo e, em particular, as que foram utilizadas neste trabalho.

4.1 MATLAB®

Não se pode falar em ferramentas para estudo da Engenharia de Controle sem citar a aplicação comercial MATLAB®. O *software* possui uma biblioteca específica para atender a área de controle, tanto para modelos contínuos como para modelos discretos.

Outro recurso muito interessante do aplicativo é a interface gráfica para programação e simulação por módulos conhecida como *SIMULINK*®.

Dentro do escopo desta pesquisa, o MATLAB® é interessante por apresentar rotinas específicas para solucionar problemas LQR e MPC.

4.2 JMODELICA.ORG

A ferramenta JModelica.org (http://jmodelica.org/) (ÅKESSON et al., 2010) constitui um ambiente integrado de modelagem, simulação e otimização de sistemas dinâmicos. Trata-se do resultado de uma pesquisa do Departamento de Controle Automático da Universidade de Lund, Suécia, que hoje tem seu desenvolvimento e manutenção a cargo da empresa sueca Modelon AB.

Algumas das principais características da ferramenta são:

- é um software livre distribuído sob a licença GPL v3;
- utiliza a linguagem orientada a objetos Modelica para modelar sistemas;
- soluciona problemas de simulação e de otimização complexos em ambiente integrado com outras ferramentas de diferenciação automática e resolvedores para EDO e EDA, (ANDERSSON et al., 2011; MAGNUSSON; ÅKESSON, 2012), de solucionadores lineares de larga escala, (Science and Technology Facilities Council, 2015), e de otimização, (WÄCHTER; BIEGLER, 2006); e
- permite automatização de tarefas em ambiente iterativo Python, o que facilita a geração e a análise dos dados.

Uma descrição da integração da ferramenta JModelica.org com outras ferramentas, destacando-se os fluxos de dados e de comandos, é mostrada na figura 10.

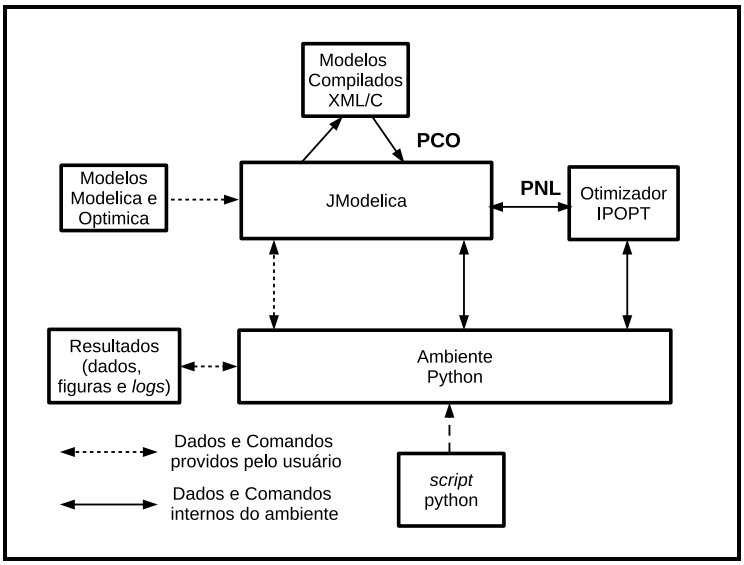

Figura 10: Comunicações do Ambiente Integrado. Fonte: o autor.

A figura 10 apresenta os fluxos de comunicações de dados/comandos definidos pelo usuário e os internos das ferramentas. Os fluxos de comandos originados pelo usuário podem ser pela execução de um programa em linguagem Python, como na figura, ou simplesmente por digitação no console iterativo Python.

Podemos apresentar o seguinte algoritmo para solução de um PCO pela ferramenta JModelica.org:

1. modelar o sistema alvo[1] em linguagem Modelica usando a classe *model*;

2. estender a classe anterior e modelar as restrições do PCO, incluindo e principalmente a função objetivo, com a extensão da linguagem Modelica, Optimica, usando a classe *optimization*;

3. editar programa em linguagem Python que chame as rotinas JModelica.org para ler e compilar os referidos modelos anteriores, otimizar o modelo Optimica, eventualmente simular e, por último, armazenar todos os resultados das compilações[2], otimizações e simulações; e

4. se for o caso, editar programa em linguagem Python para analisar os dados dos resultados.

[1] No caso desta pesquisa, os modelos Modelica e Optimica utilizados estão no apêndice A.

[2] Um importante fator que afeta o processo de compilação é a consistência entre o número de variáveis e de equações do modelo.

Pela observação da figura 10, verificamos que o usuário faz chamadas diretamente às rotinas da ferramenta JModelica.org e esta, por sua vez, por rotinas internas, faz a interface para as outras ferramentas. Por exemplo, o usuário não interage diretamente com o otimizador IPOPT, mas pode, ao chamar a rotina de otimização da ferramenta JModelica.org, definir as tolerâncias e demais parâmetros para o otimizador. De forma análoga, após a otimização, o IPOPT não reporta os resultados diretamente ao usuário, salvando os valores das variáveis, mas sim por intermédio de chamadas a rotinas e objetos da ferramenta JModelica.org

O ambiente JModelica.org permite a configuração de alguns parâmetros de otimização. Por exemplo, os parâmetros para discretização do PCO, como a definição do número de elementos, "n_e", e número de pontos de colocação, "n_c", definidos no capítulo anterior, são configurados para o escopo da ferramenta JModelica.org. Outros parâmetros de otimização do IPOPT, como os abordados na próxima seção, são definidos por rotinas JModelica.org e atuam apenas no escopo da ferramenta IPOPT.

4.3 IPOPT

A ferramenta IPOPT é compilada separadamente da JModelica.org, em procedimento que é descrito no próprio manual da JModelica.org. Uma vez compiladas, cabe ao usuário apenas definir as opções de otimização, caso queira valores diferentes dos padrões, e realizar a otimização. Tudo é feito por chamadas das rotinas do JModelica.org.

É importante salientar os principais mecanismos de parada das iterações de otimização. Existem vários, mas os mais importantes são os seguintes:

- parada por encontrar a solução ótima. Este mecanismo é ativado quando a iteração atinge os requisitos primários de tolerância definidos nas opções de otimização ou, caso não definidos, os padrões. Neste caso, o IPOPT retorna o código "0" e transfere todos os resultados num objeto JModelica ao ambiente Python, podendo ser acessados pelo usuário;
- parada por encontrar a solução aceitável por determinado numero de iterações. Este mecanismo é ativado quando a iteração atinge os requisitos secundários de tolerância (menos rigorosos que os primários) por um determinado número de iterações, definidos nas opções de otimização ou, se não definidos, os padrões. Neste caso, o IPOPT retorna o código "1" e transfere todos os resultados num objeto JModelica ao ambiente Python, podendo ser acessados pelo usuário;
- parada por atingir limite máximo de iterações definido nas opções de otimização. Neste caso, o IPOPT lança uma exceção e retorna o código "-1"; e
- parada por atingir limite máximo de tempo definido nas opções de otimização. Neste caso, o IPOPT lança uma exceção e retorna o código "-4".

Os dois primeiros mecanismos de parada são os únicos que constituem sucesso da otimização. O dois últimos, juntamente com outros 15, constituem insucesso. Todos possuem seus respectivos códigos que identificam paradas, por exemplo, por solicitação do usuário, por erro interno, por atingir ponto infactível, por insucesso em fase de restauração, etc.

O interessante deste ambiente integrado das ferramentas JModelica.org e IPOPT é que todas as opções disponíveis no manual do IPOPT para configuração da otimização podem ser definidas via JModelica.org. Não somente as citadas anteriormente para os mecanismos de parada, mas todas que afetam a otimização, a velocidade e a convergência das iterações, incluindo opções afetas ao método de barreira, condições de KKT e outras. Fica evidenciado que a boa escolha de destas opções resultarão no sucesso[3] e na velocidade otimização.

É importante salientar que o IPOPT realiza uma otimização puramente estática. Cabe às ferramentas integradas JModelica.org, transformar o dinamismo intrínseco ao PCO original num PNL estático como descrito no capítulo anterior.

4.4 ROTINAS HSL

As rotinas HSL, conforme (Science and Technology Facilities Council, 2015), constituem um conjunto de solucionadores que podem resolver problemas lineares de grande escala. Estas rotinas são fornecidas sob a licença "HSL ACADEMIC LICENCE VERSION 1.1 JANUARY 2011", e podem ser compiladas de forma integrada ao IPOPT ou carregadas dinamicamente.

Como são várias rotinas, uma vez compiladas, a utilização é feita por definição da opção "solver" do IPOPT.

4.5 PYTHON E OUTRAS FERRAMENTAS

As ferramentas JModelica.org, IPOPT e HSL constituem o núcleo do ambiente de modelagem, otimização e simulação. Contudo, outras ferramentas são necessárias tanto no processo de instalação do ambiente como na sua operação.

Obviamente, o próprio interpretador para linguagem Python necessita ser instalado. Além disto, é necessário ter um compilador para linguagem C e um Kit de Desenvolvimento Java (JDK) instalados para a compilação e operação das ferramentas JModelica.org, IPOPT e HSL.

Caso o ambiente seja *Microsoft Windows*®, é possivel instalar os binários baixados do *site* da JModelica.org.

[3] Um fator importante que afeta diretamente a otimização é a definição consistente das restrições do PCO. Eventualmente o IPOPT não convergirá pelo simples fato do PCO ter sido formulado de maneira tão restritiva que tornou-se um problema impraticável.

Esta pesquisa, no entanto, utilizou o ambiente Linux[4] com as ferramentas da tabela (1).

Tabela 1: Ferramentas Utilizadas.

Ferramenta	Versão	Licença
JModelica.org	1.15	GPL v3
IPOPT	3.12.3	GPL v3
HSL	2014.01.10	HSL ACADEMIC LICENCE VERSION 1.1 JANUARY 2011
Python	2.7.6	PYTHON SOFTWARE FOUNDATION LICENSE VERSION 2
Java (OpenJDK)	1.7.0_09	GPL v2+
gcc	4.6.4	GPL v3

É importante observar que estas versões das ferramentas foram testadas nos processos de instalação e de operação conjuntas. A eventual diferença em uma versão poderá representar problemas na compilação. Contudo, se houver necessidade de uma versão específica diferente, basta atentar para a compatibilização em relação às dependências de *software* solicitadas nos manuais e procedimentos de instalação.

4.5.1 Automatização com scripts Python

A execução do programa Python automatiza as tarefas e favorece a reprodução e a análise dos dados.

Sugerimos os seguintes passos para a edição do programa de otimização:

1. definir opções (número de elementos para discretização, número de pontos de colocação e outras que julgar necessárias) para compilação dos modelos Modelica e Optimica;
2. chamar rotina JModelica.org para compilar os Modelos;
3. carregar o modelo Optimica;
4. definir opções para otimização (solucionador linear, tolerâncias primárias, secundárias, tempo máximo de otimização, número máximo de iterações e outras que julgar necessárias);
5. realizar otimização; e
6. armazenar resultados.

Na realidade, ocorre um processo de ajuste desde a modelagem, definição das opções tanto de compilação como de otimização, de forma a se obter os resultados. Uma vez obtidos os

[4]Apenas para registro, o sistema operacional utilizado nesta pesquisa foi o Ubuntu Desktop 14.04.3, rodando em um *notebook Samsung*®, modelo RV411, com processador *Intel*® I5 e 4GBytes de memória RAM.

resultados, observamos que estes são relativamente robustos a variações nas tolerâncias, mas podem nem ser alcançados se alguma alteração nas opções tornar a otimização impraticável.

Outra observação importante é que as ferramentas permitem a otimização num processo totalmente determinístico. A menos que se programe para inserção de um sinal ou parâmetro aleatório, definidas as opções de otimização e os parâmetros do modelo Optimica, os resultados de múltiplas execuções de uma mesma otimização, se esta convergir, serão sempre os mesmos[5].

Portanto, alguns ciclos de execução do programa Python poderão ser necessários até que se obtenha um resultado consistente.

Numa segunda fase, pode-se programar um programa de análise para tratar e estudar os resultados. Neste caso, os modelos Modelica e Optimica já estariam compilados.

A vantagem de separar em dois programas é que as otimizações, em geral dependendo tanto do PCO como das opções de otimização adotadas, podem variar muito para convergir, desde poucos segundos até dias. Já as análises são muito rápidas e podem tomar alguns poucos segundos, no máximo alguns minutos. A separação permite distinguir bem a fase de aquisição de dados da fase de tratamento e interpretação dos resultados.

Sugerimos os seguintes passos para a edição do programa de análise:

1. ler os resultados dos sinais de controle e parâmetros de otimização armazenados pelo programa de otimização;
2. definir opções de simulação coerentes com as opções de otimização;
3. simular modelo Modelica ou, se forem necessários dados da otimização como valores da função objetivo no tempo, simular o modelo Optimica; e
4. comparar resultados da simulação com os da otimização.

No caso da presente pesquisa, o programa de otimização, que é apresentado no apêndice A, quando executado realiza 3 contextos de otimizações, tendo em cada contexto uma grande variação de parâmetros em que cada variação representa uma otimização.

Da mesma forma, o programa de análise, que também é apresentado no apêndice A, realiza várias análises para cada um dos contextos de otimização.

4.6 Aferição do Ambiente Instalado

Nesta seção, apresentamos um caso pelo qual podemos aferir os resultados do ambiente instalado.

[5] Esta característica poderá não ser observada se o solucionador linear escolhido não for determinístico, como a rotina HSL_MA86.

4.6.1 Sistema de Segunda Ordem

Sistema de segunda ordem apresentado em (KIRK, 1970, pp. 216).

O PCO dado por:

minimizar:

$$J[u] = \int_0^T \left[x_1^2 + \frac{1}{2}x_2^2 + \frac{1}{4}u^2 \right] dt \tag{65}$$

s.a.:

$$\dot{\boldsymbol{x}}_1(t) = \boldsymbol{x}_2(t) \tag{66}$$

$$\dot{\boldsymbol{x}}_2(t) = 2\boldsymbol{x}_1(t) - \boldsymbol{x}_2(t) + \boldsymbol{u}(t) \tag{67}$$

As condições iniciais[6] são:

- $x_1(0) = 4$;
- $x_2(0) = -4$;
- $u(0) = 9.04$.

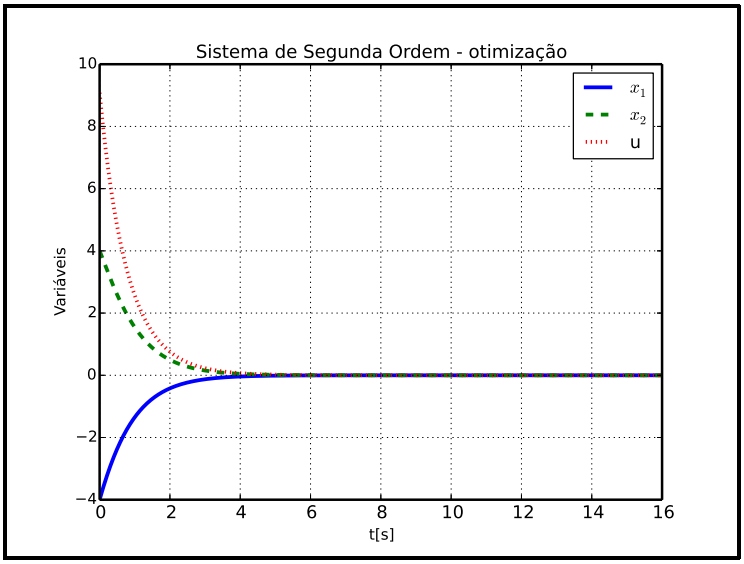

Figura 11: Sistema de Segunda Ordem.(KIRK, 1970, pp. 218)

[6] No exemplo da referência não é apresentada a condição inicial do sinal de controle, contudo o seu valor é facilmente inferido da necessária suposição adotada para se resolver a equação de Riccati, conforme (KIRK, 1970, Equação 5.2-26, pp. 217).

O gráfico da figura 11 é idêntico ao da figura 5-9 b) que se encontra em (KIRK, 1970, pp. 218). Obviamente que os métodos para se obter os resultados foram totalmente diferentes, mas a igualdade da solução permite concluir que o ambiente instalado está coerente com um resultado simples da literatura clássica de controle ótimo.

O próximo capítulo apresenta os resultados e as discussões relativos a nossa aplicação da técnica de otimização dinâmica por pontos de colocação.

5 RESULTADOS E DISCUSSÕES

O problema de controle estudado foi tratado em duas fases. Inicialmente, o modelo do PCO (apresentado na seção - A.1) foi carregado e o problema de controle resolvido numericamente pelas ferramentas JModelica e IPOPT, resultando nos controles (bem como nas demais variáveis) ótimos.

Num segundo passo, foram aplicados ao modelo do sistema os controles otimizados e as variáveis resultantes foram comparadas com os valores calculados na otimização.

Este procedimento foi feito nas condições descritas pelos trabalhos anteriores e, posteriormente, com o ajuste na função objetivo proposto neste trabalho.

5.1 Modelo de Transmissão da Dengue

O modelo de transmissão da dengue foi escolhido para subsidiar esta pesquisa pela importância social do assunto e também por apresentar uma complexidade matemática necessária. Como veremos a seguir, são cinco equações diferenciais, na maioria delas acopladas umas às outras, e dois sinais de controle que constituem um sistema dinâmico não linear. Portanto, do ponto de vista de engenharia de controle, o modelo escolhido é bastante complexo e interessante.

Do ponto de vista social, como veremos nas seções seguintes, a utilização eficiente das técnicas de combate à dengue pode significar melhores resultados econômicos e menos casos da doença, o que é o objetivo primordial do gestor de saúde pública.

Desta forma, apresentamos a seguir os principais conceitos e parâmetros do modelo da transmissão da dengue, conforme constam nas referências básicas, (ESTEVA; YANG, 2005; THOMÉ, 2007):

- A(t) - população na fase aquática;
- I(t) - população de fêmeas imaturas (antes de acasalar);
- F(t) - população de fêmeas fertilizadas (depois de acasalar);
- U(t) - população de fêmeas não fertilizadas (depois de acasalar);
- M(t) - população de machos naturais (férteis); e
- S(t) - população de machos estéreis.

A dinâmica da evolução populacional pode ser sumarizada, (THOMÉ, 2007, veja figura

3.1), em:

- a população da fase aquática 'A' evolui para a fase adulta por uma taxa γ.
- uma fração 'r' da população adulta corresponde a fêmeas imaturas 'I' e, naturalmente, o restante corresponde a machos 'M'.
- após o acasalamento, as fêmeas imaturas evoluem para duas possibilidades: primeiro para fêmeas fertilizadas denotadas por 'F', ou para fêmeas não fertilizadas, 'U'.
- prosseguindo, uma fração das fêmeas fertilizadas inicia uma nova geração de população na fase aquática; e
- por último, existe uma porcentagem de morte natural especifica para todas as populações.

Outros parâmetros, cujos valores nos casos estudados estão definidos em (THOMÉ, 2007; BARSANTE et al., 2013), são:

- μ_X - são as taxas de mortalidade de cada população, em que X corresponde ao índice que representa a respectiva população.
- C - fator que caracteriza o meio quanto a nutrientes, espaço, etc;
- ϕ - taxa de oviposição intrínseca;
- γ - taxa de mosquitos que passam da fase aquática para a adulta;
- r - proporção de fêmeas que passam pra fase adulta;
- β - taxa de acasalamento dos machos naturais;
- β_S - taxa de acasalamento dos machos estéreis; e
- α - taxa de inserção de mosquitos estéreis.

Os resultados das otimizações e simulações consideraram os seguintes valores para os parêmetros, (THOMÉ, 2007; BARSANTE et al., 2013):

Tabela 2: Parâmetros do Modelo de Transmissão da Dengue.

ϕ	C	γ	r	β	β_S	μ_A	μ_I	μ_F	μ_M	μ_S
0,50	13	0,07	0,50	1	0,70	0,05	0,05	0,0.5	0,1	0,1

As equações que definem o modelo são:

$$\frac{dA}{dt} = \phi\left(1 - \frac{A}{C}\right)F - (\gamma + \mu_A)A \tag{68}$$

$$\frac{dI}{dt} = r\gamma A - \frac{\beta MI}{M+S} - \frac{\beta_S SI}{M+S} - \mu_I I \tag{69}$$

$$\frac{dF}{dt} = \frac{\beta MI}{M+S} - \mu_F F \tag{70}$$

$$\frac{dM}{dt} = (1-r)\,\gamma A - \mu_M M \tag{71}$$

$$\frac{dS}{dt} = \alpha - \mu_S S \tag{72}$$

Onde as condições iniciais são dadas por:

$$A_0 = C\left(\frac{R-1}{R}\right) \tag{73}$$

$$I_0 = \frac{r\gamma A_0}{\mu_I + \beta} \tag{74}$$

$$F_0 = \frac{(\gamma + \mu_A)CA_0}{\phi\,(C - A_0)} \tag{75}$$

$$M_0 = \frac{(1-r)\,\gamma A_0}{\mu_M} \tag{76}$$

$$S_0 = 0 \tag{77}$$

$$R = \frac{\phi r \gamma \beta}{(\gamma + \mu_A)\,(\beta + \mu_I)\,\mu_F} \tag{78}$$

5.2 Problema de Controle Ótimo

As expressões de (68) a (78) representam um sistema livre e sem controle externo que, partindo das condições iniciais, prossegue na evolução de cada população.

A ideia do controle é inserir o efeito dos inseticidas e da inserção de machos estéreis. Estes efeitos são modelados por intermédio dos sinais de controle u_1 e u_2:

- u_1 - sinal de controle relacionado aos inseticidas; e
- u_2 - sinal de controle relacionado à inserção de machos estéreis.

De forma que o Problema de Controle Ótimo pode ser descrito por:

minimizar:

$$J[u_1,u_2] = \frac{1}{2}\int_0^T \left[c_1u_1^2 + c_2u_2^2 + c_3F^2 - c_4S^2\right] dt \tag{79}$$

s.a.:

$$\frac{dA}{dt} = \phi\left(1-\frac{A}{C}\right)F - (\gamma+\mu_A)A \tag{80}$$

$$\frac{dI}{dt} = r\gamma A - \frac{\beta MI}{M+S} - \frac{\beta_S SI}{M+S} - (\mu_I+u_1)I \tag{81}$$

$$\frac{dF}{dt} = \frac{\beta MI}{M+S} - (\mu_F+u_1)F \tag{82}$$

$$\frac{dM}{dt} = (1-r)\gamma A - (\mu_M+u_1)M \tag{83}$$

$$\frac{dS}{dt} = u_2 - (\mu_S+u_1)S \tag{84}$$

$$u_1 \geq 0 \tag{85}$$

$$u_2 \geq 0 \tag{86}$$

Na expressão da função objetivo em (79), os coeficientes significam:

- c_1-índice de custo relacionado ao controle por inseticidas;
- c_2-índice de custo relacionado ao controle por mosquitos estéreis;
- c_3-índice de custo social; e
- c_4-índice de benefício pelos machos estéreis.

As condições iniciais foram definidas em (73-78).

5.3 Classificação do Sistema de Controle Estudado

O sistema de controle estudado pode ser classificado com base nos conceitos apresentados na seção 2.2, página 34, como:

- Causal;
- Concentrado;

- Contínuo;
- Controlável[1];
- de Malha Aberta[2];
- Determinístico;
- Dinâmico (com Memória);
- Não-linear;
- Variante no tempo;
- MIMO;

$$\boldsymbol{u} = [u_1, u_2]^T, \quad \boldsymbol{y} = [x_1, x_2, x_3, x_4, x_5]^T = [A, F, I, M, S]^T \tag{87}$$

- Observável;

$$\boldsymbol{y} = \boldsymbol{x} \tag{88}$$

- Restrito quanto à otimização.[3]

As classificações como controlável e de malha aberta merecem maiores detalhes que podem ser discutidos com o auxílio das figuras 10, 1 e 12.

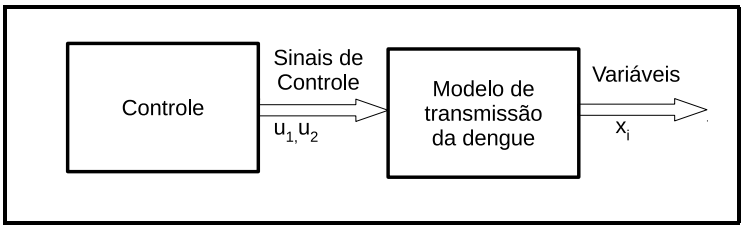

Figura 12: Sistema de Controle da Transmissão da Dengue. Fonte: o autor.

[1]Uma definição de controlabilidade apresentada por (ATHANS, 2007, pp. 200) afirma que o sistema é controlável se puder ser conduzido à origem (estado inicial) num intervalo finito de tempo. Em nosso caso de estudo, se considerarmos a proposta inicial em que não se previu limites superiores para os controles, a controlabilidade sempre seria possível em termos da limitação da variável de interesse, isto é, da população de fêmeas fertilizadas (F).

[2]Há pesquisadores que afirmam que controle em malha aberta não seria a essência da Engenharia de Controle. Certamente não o é, mas acreditamos que, mesmo em malha aberta, a solução de um PCO fornece sinais de controle que seriam, em tese, ideais, isto é, a melhor excitação que o sistema real poderia receber. No escopo da nossa pesquisa não tratamos de um sistema real, mas sim de simulações de um modelo. Somente esta característica da pesquisa, para os mais rigorosos, já poderia dificultar a sua classificação em controle. Contudo entendemos que o estudo inicial representa uma abordagem que é fortemente plausível de aplicação prática, incluindo a possibilidade de realimentação de sinais, desde que todos os sinais e parâmetros sejam rigorosamente processados para melhor espelhar a realidade.

[3]O modelo Optimica permitiu restrições quanto aos valores máximos e mínimos de qualquer variável, mas foram limitados superiormente apenas os sinais de controle.

Na figura 12, o modelo pode ser entendido como um dos dois modelos (códigos Modelica e Optimica) existentes em nosso estudo:

O primeiro código (Modelica) representa o modelo do sistema natural com as variáveis das populações (A,I,F,M e S) e os controles (u_1 e u_2).

Neste modelo, o conceito de estabilidade não é adequado porque ele simplesmente calcula as populações num período em função dos controles neste período. Mesmo o conceito de saturação não seria adequado para este modelo. A saturação seria observada apenas como uma possível característica dos controles que, por exemplo, podem representar limitação de orçamento para o gestor público de saúde.

Entende-se que o principal objetivo a ser analisado neste modelo seria a curva da população "F"(Fêmeas Fertilizadas).

O segundo código (Optimica) estende o primeiro modelo (código Modelica) para inserir as restrições (controles máximos) e a função objetivo.

O segundo modelo serve primordialmente para otimizar os controles para a função objetivo e dentro das restrições estabelecidas.

Obtidos os controles ótimos, eles se aplicam ao código Modelica para calcular as populações. Aplicados ao código Optimica, garantirão o índice de desempenho calculado na otimização. Contudo, ambos os códigos servem para simulação do sistema submetido aos controles ótimos.

A comparação entre as figuras 1 e 12 evidencia que o modelo de sistema de controle da dengue adotado em nosso estudo corresponde a um sistema em malha aberta, sem sinal de referência e sem distúrbios.

A comparação entre as figuras 10 e 12 evidencia que a função de controle nas simulações é exercida pela ferramenta JModelica.org, e o modelo da transmissão da dengue corresponde a um dos códigos (Modelica ou Optimica) compilados.

É evidente que tanto o modelo matemático como o modelo em código que pode ser otimizado ou simulado representam uma simplificação da realidade sobre a qual desejamos atuar.

5.4 Otimização Dinâmica

As expressões de (68) a (78) evidenciam que o Problema de Otimização (PO) em questão possui característica dinâmica, isto é, as variáveis a serem otimizadas variam com o tempo, nos casos estudados, pelo período de 120 dias. Esta complicação representa um enorme acréscimo de dificuldade para solução em comparação ao PO estático, cuja solução seria obtida pelas soluções de infinitos PO estáticos distribuídos no período de 120 dias.

O trabalho original (THOMÉ, 2007) abordou o problema por uma transformada na variável tempo, normalizando para a unidade e, partindo das condições iniciais, resolvendo sequencialmente PO espaçados por Δt, em que a solução do PO anterior era utilizada como ponto inicial para o método de Newton do PO subsequente.

A otimização dinâmica (MAGNUSSON; ÅKESSON, 2012) utiliza a técnica de colocação direta para transformar um PO de dimensão infinita em um programa não linear (PNL) de dimensão finita.

5.5 Resultados

Os resultados das simulações numéricas podem ser agrupados em três contextos diferentes para otimizações do PCO em estudo, cada contexto objetivando investigar aspectos específicos do PCO ou do próprio ambiente integrado de otimização. As otimizações destes contextos foram realizadas sequencialmente pelo *script* Python *otimiza.py*, apresentado na seção A.2.

A análise dos resultados considerando cada contexto foi realizada pelo *script* Python *analisa.py*, apresentado na seção A.2.

5.5.1 *Otimizações do Contexto 1*

Neste contexto, foram estudados os problemas específicos dos casos 1 a 4 dos trabalhos (THOMÉ, 2007; BARSANTE et al., 2013). Nestes casos, ocorrem as limitações dos domínios das variáveis de controle em valores definidos na tabela 3, abaixo, e as otimizações são feitas para controles constantes.

Tabela 3: Resultados da Otimização Dinâmica Aplicada aos Principais Casos (k=1).

Caso	C_i	Domínio dos Controles	(THOMÉ, 2007)		(BARSANTE et al., 2013)		Este Trabalho	
		$[u_1max;u_2max]$	J^*	$[u_1^*;u_2^*]$	J^*	$[u_1^*;u_2^*]$	J^*	$[u_1^*;u_2^*]$
1	[1,10,100,1]	[0,1 ; 0,02]	42155	[0,093 ; 0,0156]	26384	[0,099;0,00015]	6099	[0,1 ; 0,02]
2	[1,10,1,1]	[0,08; 0,01]	429	[0,0786; 0,0055]	313	[0,0787;0,00017]	197	[0,08; 0,01]
3	[1,1,1,1]	[0,07; 0,05]	547	[0,0602; 0,0441]	350	[0,0694;0,00016]	199	[0,07; 0,05]
4	[3,1,1,1]	[0,04; 0,08]	725	[0,0323;0,0738]	608	[0,0399;0,00013]	299	[0,04; 0,08]

Dados de (THOMÉ, 2007) e (BARSANTE et al., 2013) foram obtidos de (BARSANTE et al., 2013).

Os resultados indicam que a otimização dinâmica obtém controles saturados. A função objetivo, como em (79), sofre maior peso das populações dos mosquitos, em detrimento das componentes de controle.

Como tentativa de equalizar as contribuições dos sinais de controle, sugere-se a modificação

da função objetivo com a introdução de um fator de correção 'k', como a seguir[4]:

$$J[u_1,u_2] = \frac{1}{2}\int_0^T \left[k*(c_1u_1^2+c_2u_2^2)+c_3F^2-c_4S^2\right]dt \tag{89}$$

5.5.2 *Análise do Contexto 1*

5.5.2.1 Estudo da Equação da População de Mosquitos Estéreis

Devido às evidências dos resultados das simulações sobre a importância da população de mosquitos estéreis, convém estudar mais detalhadamente a equação diferencial apresentada em (84).

Vemos que a equação envolve apenas os sinais de controle e a própria variável S, de forma que, por ser desacoplada das demais equações, há um indício de possível solução analítica que vamos perseguir. Na realidade, (BOYCE, 1990, pp.10) apresenta a solução na forma genérica:

$$\dot{y} \;=\; f(t)-g(t)y \tag{90}$$

ou

$$\dot{y}+g(t)y \;=\; f(t)$$

A solução de (90) vem multiplicando-se ambos os lados por $m(t) \;=\; e^{\int_0^t g(s)ds}$:

$$y \;=\; \frac{\int_0^t \left[f(s)e^{\int_0^s g(\tau)d\tau}\right]ds+c}{e^{\int_0^t g(s)ds}} \tag{91}$$

Assim, considerando a condição inicial, a expressão para a população de mosquitos estéreis fica[5]:

$$S(t) \;=\; \frac{\int_0^t \left[u_2(s)e^{\int_0^s (u_1(\tau)+\mu_S)d\tau}\right]ds}{e^{\int_0^t (u_1(s)+\mu_S)ds}} \tag{92}$$

[4]Um maior esclarecimento do significado do fator de ajuste 'k' é apresentado em 5.5.2.2 e 5.5.3.

[5]É conveniente ressaltar que não se deve confundir a variável 'S' (população de mosquitos estéreis) com a variável de integração 's'.

5.5.2.2 O PCO Original como um LQR

Inicialmente, observando a função objetivo original como definida na equação (79), poder-se-ia imaginar que o PCO original se enquadraria na classificação de um LQR[6], apresentada em 2.3.3.1.

Poderíamos pensar em matrizes Q e R da forma:

Matriz R:

$$R = \begin{bmatrix} c_1 & 0 \\ 0 & c_2 \end{bmatrix} \tag{93}$$

Matriz Q:

$$Q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & c_3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -c_4 \end{bmatrix} \tag{94}$$

Vetor de Estado:

$$\boldsymbol{x} = [A, I, F, M, S]^T \tag{95}$$

Com as matrizes acima, o vetor de estado e os sinais de controle, facilmente se chega à equação (18) com a condição de estado final livre ($r(T) = 0$).

Mas ainda não é o suficiente para enquadrar o PCO estudado na situação do LQR, pelo detalhe importante de Q não ser semidefinida positiva[7] [8].

De forma que o PCO definido com a função objetivo da equação (79) é uma situação nova que demanda cuidado e atenção. Numa visão inicial, o engenheiro de controle poderia ser levado a "transformar" a função objetivo de forma a se enquadrar no LQR, o que facilitaria em muito a solução do PCO. Bastaria eliminar a influência da população de mosquitos estéreis fazendo $c_4 = 0$. Esta solução facilitaria em muito a vida do engenheiro, mas não resolveria o problema do idealizador do PCO que inseriu o conceito do "benefício" da população de mosquitos estéreis. Este conceito é muito importante e alterá-lo descaracterizaria o PCO original na sua intenção de espelhar a realidade.

Prosseguindo na análise, observa-se na equação (79) que o índice de desempenho é adimensional. No entanto a expressão engloba quatro variáveis que representam significados levemente

[6] Esta classificação seria apenas em relação ao índice de desempenho. As demais características necessárias para um LQR, como linearidade e realimentação, obviamente não estão presentes em nosso estudo.

[7] A definição de matriz semidefinida positiva é dada por (KIRK, 1970, pp. 31) e diz: "Uma matriz $\boldsymbol{H}$ é semidefinida positiva se para todos os vetores $\boldsymbol{z}$ tiver-se $\boldsymbol{z}^T\boldsymbol{H}\boldsymbol{z} \geq 0$ ".

[8] Lembrando que por definição tem-se $c_i > 0$.

diferentes:

1. o sinal de controle u_1 significa a taxa da porcentagem da população de mosquitos que morrem devido aos inseticidas. A dimensão de u_1 seria a dimensão das variáveis de estado[9] pelo tempo, isto é: $[u_1] = [x][\text{dias}]^{-1}$. Por exemplo, o valor constante de $u_1 = 0,01$ significa que cada população decresceria por inseticidas diariamente pela taxa de 1% de sua população; Nos parece que, por ser percentual da população, não tem sentido ter valor de u_1 maior que a unidade.

2. o sinal de controle u_2 significa taxa real de liberação de mosquitos estéreis, não em porcentagem da população, mas na unidade de população adotada pelo modelo. A dimensão de u_2 seria a dimensão das variáveis de estado pelo tempo, isto é: $[u_2] = [x][\text{dias}]^{-1}$; e

3. a variável referente às fêmeas fertilizadas, F, e a relativa aos mosquitos estéreis, S, são os valores das respectivas populações e possuem dimensões $[F] = [S] = [x]$.

Como o índice de desempenho é adimensional, os coeficientes c_i devem ser adequados a uma coerência dimensional, de forma que as dimensões dos coeficientes dos controles devem ser $[c_1] = [c_2] = [x]^{-2}[\text{dias}]$. Igualmente, os coeficientes das variáveis populacionais devem ser $[c_3] = [c_4] = [x]^{-2}[dias]^{-1}$.

Podemos até, com certa facilidade, tentar exprimir pelos coeficientes dos controles os custos operacionais para as respectivas estratégias de combate à dengue. Poderíamos pensar em levar o índice de desempenho a ter a dimensão de unidade monetária. Mas haveria o problema para 'quantificar' os custos relativos às variáveis populacionais F e S.

Mesmo com a difícil possibilidade prática para quantificar monetariamente[10] os prejuízos com uma doença ou os benefícios com a sua prevenção, nos parece que esta não seria uma boa interpretação da equação (79). Nos parece que melhor seria continuar considerando o índice de desempenho adimensional e entender os coeficientes c_i como ponderadores para os conceitos materiais (custos com estratégias de combate à doença) e imateriais (custo social da população de fêmeas fertilizadas e benefício social da população de mosquitos estéreis) que integram cada componente da função objetivo. Neste sentido, o fator de ajuste sugerido 'k' tenta "desacoplar" os coeficientes c_1 e c_2 (correspondentes a valores materiais) dos outros coeficientes (relativos a valores imateriais sociais), como na equação (89). A alteração preserva os conceitos originais e mantém a possibilidade de calcular os custos reais dos controles (como nos itens 1 e 2, acima) frente aos demais conceitos, de forma ponderada.

[9] Podemos definir a dimensão das populações pela notação: $[\text{Unidade de População}] \equiv [x]$

[10] Esta questão certamente tem importância prática na área de seguros e na área jurídica, mas tem também um grande significado filosófico. Uma interessante discussão do *Utilitarismo* do filósofo britânico John Stuart Mill é apresentada em (SANDEL, 2009).

E o problema matemático em tela reflete a questão: não se havendo limites para os gastos em controles, minimizar o índice de desempenho significa priorizar ilimitadamente o coeficiente do "benefício" dos mosquitos estéreis, o que na prática representaria ao gestor público de saúde dispender todos os seus recursos com liberação de mosquitos estéreis. Como consequência, a priorização ilimitada de um benefício específico implica o prejuízo global em relação a todos os demais conceitos, havendo a necessidade de ponderação eficaz.

Prosseguindo, pode-se facilmente provar que para controles limitados teremos a população de machos estéreis limitada[11] por:

$$S(t) \leq S_{max} = \frac{u_{2max} e^{u_{1max} t}}{u_{1max} + \mu_S} \tag{96}$$

Se considerarmos as condições práticas de controles limitados, isto é, $u_1(t) < u_{1max} \leq 1$ e $u_2(t) < u_{2max}$, é facil deduzir por (92) que tanto (79) como (89) possuem mínimos e podem ser encontrados. Nada podemos afirmar quanto aos valores máximos se, na ineficiência dos controles, a população de fêmeas fertilizadas (F) crescer indefinidamente.

Os resultados demostram que para função objetivo original, com k=1, as otimizações ocorrem limitando-se o controle por mosquitos estéreis, e o sistema estabiliza em:

$$u_2(t) \to u_2max\,, \;\; u_1(t) \approx 0 \;\; \text{e} \;\; S(t) = S_{max} = \tfrac{u_{2max}}{\mu_S}, \forall t \geq t_s.$$

O contexto 1 realizou otimizações para k = 1 e k = 100. Fizemos uma comparação entre os dados das otimizações para o caso 3, isto é, usando coeficientes de custo $c_i = [1, 1, 1, 1]$.

As figuras 13 a 16 apresentam os resultados. Seguindo a abordagem dos trabalhos anteriores, a simulação matemática considera sinais de controle e de populações relativos e sem unidades. Certamente um tratamento mais elaborado dos sinais será necessário para aplicação real, que deverá considerar as unidades.

A figura 13 apresenta o sinal de controle u_2 em seu limite superior $u_2max = 10$ e o sinal de controle u_1 tendendo a zero, para k=1. Com a introdução do fator de ajust k=100, estes sinais apresentam valores efetivos[12].

[11] Esta expressão deriva da aplicação dos limites mínimos e máximos, $0 \leq u1 \leq 1$, na equação 92.

[12] Os sinais de controle para k=100 podem ser vistos detalhadamente no apêndice E, especificamente na figura 59.

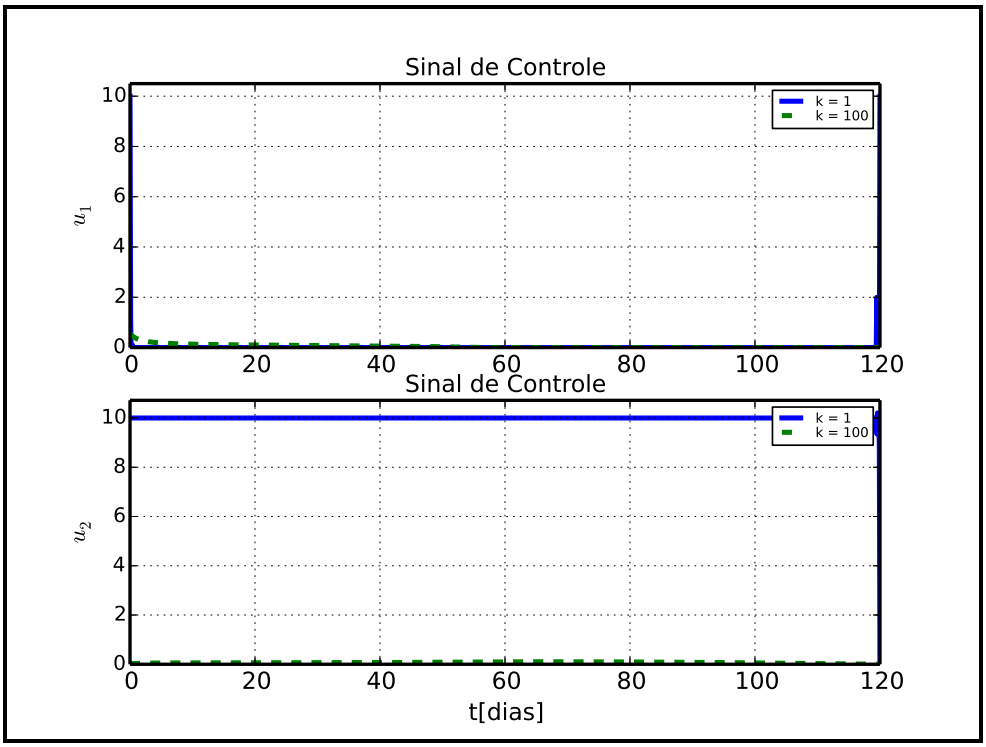

Figura 13: Sinais de Controle para k=1 e k=100. Fonte: o autor.

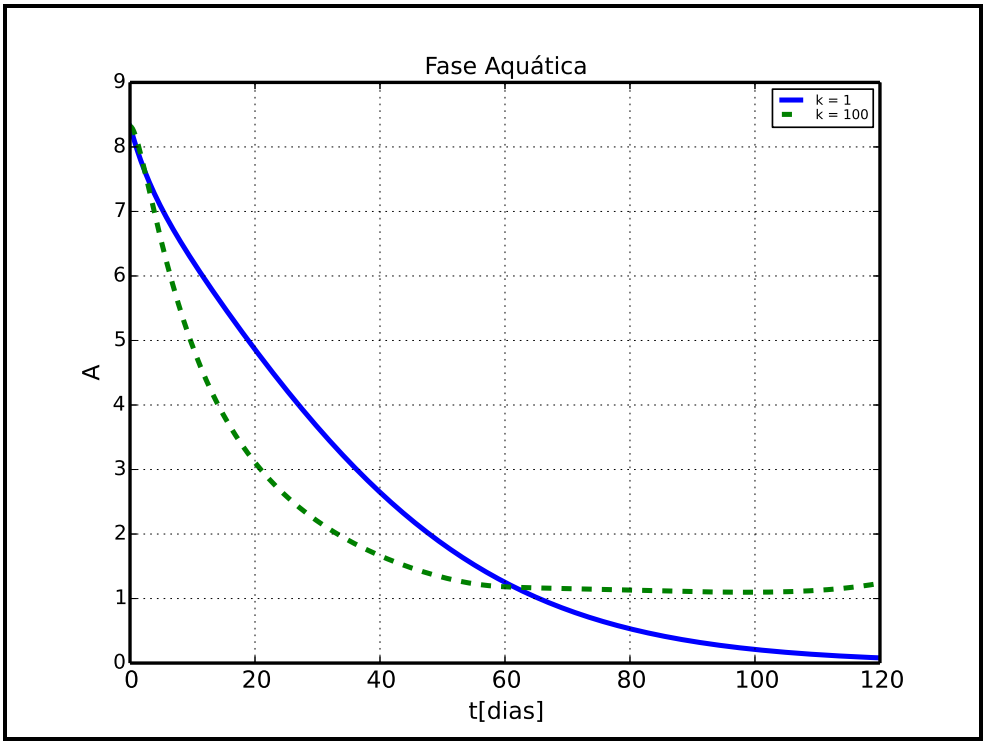

Figura 14: Fase Aquática para k=1 e k=100. Fonte: o autor.

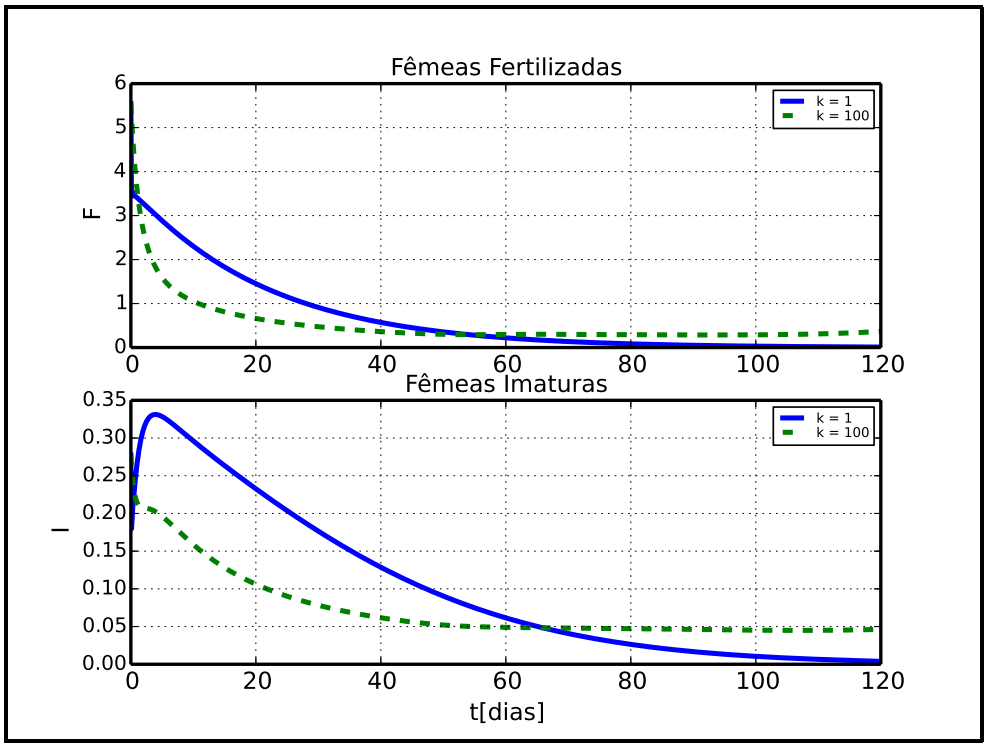

Figura 15: Populações de Fêmeas para k=1 e k=100. Fonte: o autor.

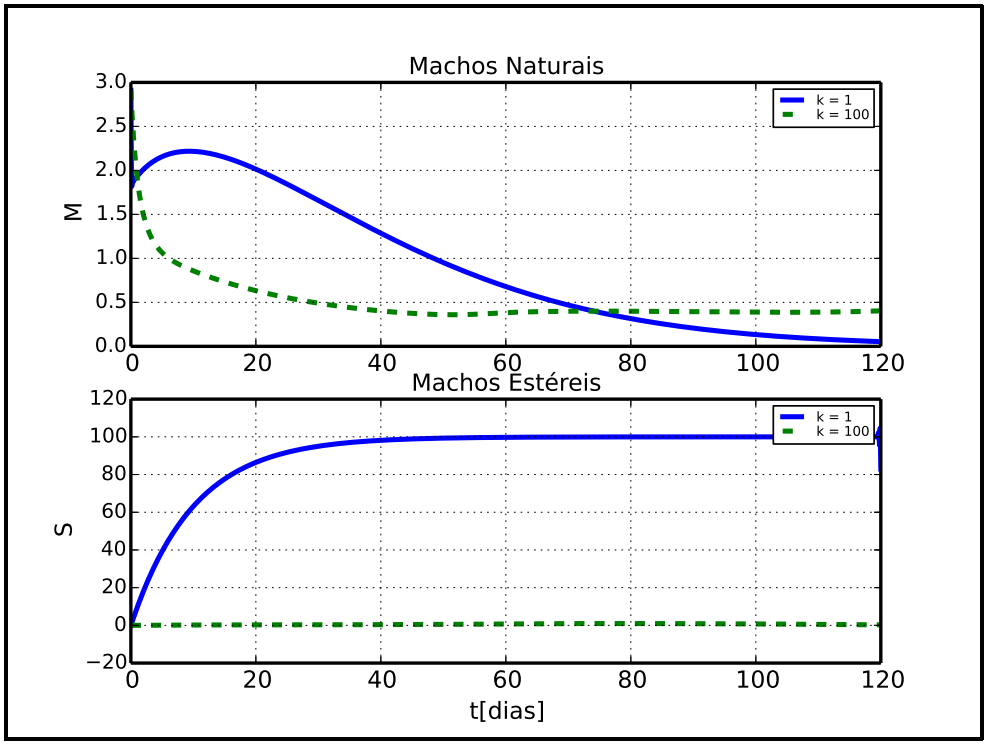

Figura 16: Populações de Machos para k=1 e k=100. Fonte: o autor.

5.5.3 Discussão do Contexto 1

Os resultados evidenciaram que uma abordagem mais precisa e poderosa em termos de recursos computacionais (MAGNUSSON; ÅKESSON, 2012), (WÄCHTER; BIEGLER, 2006), como as rotinas HSL 'ma57' (Science and Technology Facilities Council, 2015) para solucionar sistemas lineares de larga escala, viabilizadas pelas ferramentas utilizadas, permite identificar as componentes populacionais como predominantes na função objetivo.

Há necessidade de se limitar, para o cálculo da otimização, a importância da componente de controle de mosquitos estéreis (u_2). É possível ver em (84) que esta componente influencia direta e unicamente sobre a população de mosquitos machos estéreis (S), levando ao crescimento ilimitado e a consequente infactibilidade do problema.

A correção proposta permite a melhor manipulação dos parâmetros de controle.

Ressaltamos que O PCO estudado, quando descrito pelo modelo genérico apresentado em 2.3.3 com a equação do sistema dada por (11), é tal que o fator de ajuste 'k' não altera a função que define o sistema. O domínio da função de sistema contém o domínio da função que define o índice de desempenho e o fator de ajuste atua apenas sobre o índice de desempenho.

Portanto, o fator de ajuste 'k' não deve ser confundido, pela ideia sugerida com a grafia, com algum tipo de ganho nos controles. O fator 'k' não tem atuação alguma sobre os controles como acontece em algumas formulações, (BRYSON, 1975, pp. 181), e apenas representa uma espécie de transformação na função objetivo que desloca os controles ótimos da borda da região domínio da função $\boldsymbol{f}$ para o seu interior, numa condição em que estes passam a ser mais efetivos.

Tentamos apresentar[13] esta ideia de transformação na figura 17. Acreditamos que talvez tenhamos a oportunidade, pelos recursos da técnica estudada, de perceber uma nova possibilidade de se tratar índices de desempenho com funções não conformes com teoria clássica de controle ótimo.

5.5.4 Otimizações do Contexto 2

O contexto 2 considerou diversas otimizações em que o parâmetro k variou de 1 a 500. O objetivo destas otimizações foi levantar dados sobre os valores da função objetivo.

Neste sentido, a figura 18 apresenta o gráfico dos valores finais da função objetivo em função do parâmetro 'k' de cada otimização.

[13] Apresentamos esta figura não como um gráfico cartesiano formal, mas como uma representação simbólica para visualizar o conceito da transformação sugerida.

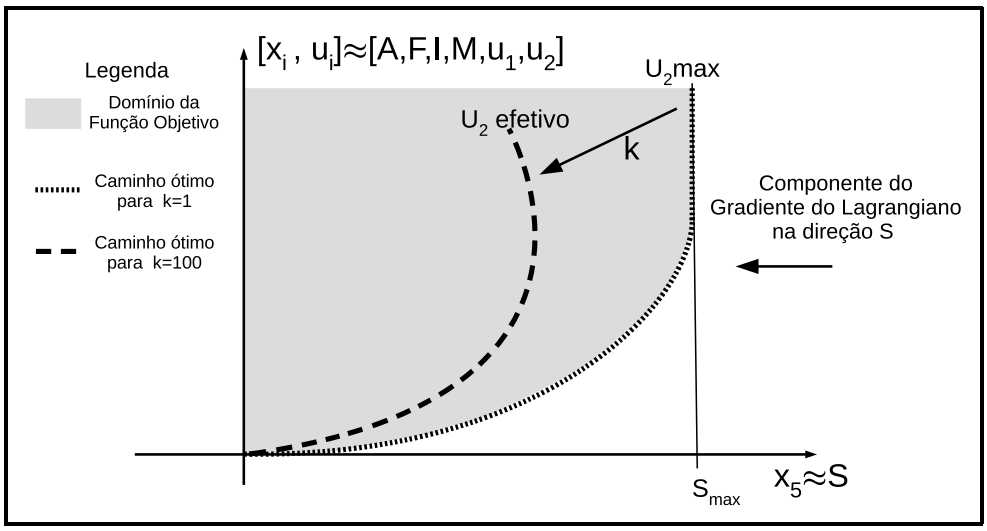

Figura 17: Representação Simbólica do Deslocamento do Mínimo Local da Função Objetivo. Fonte: o autor.

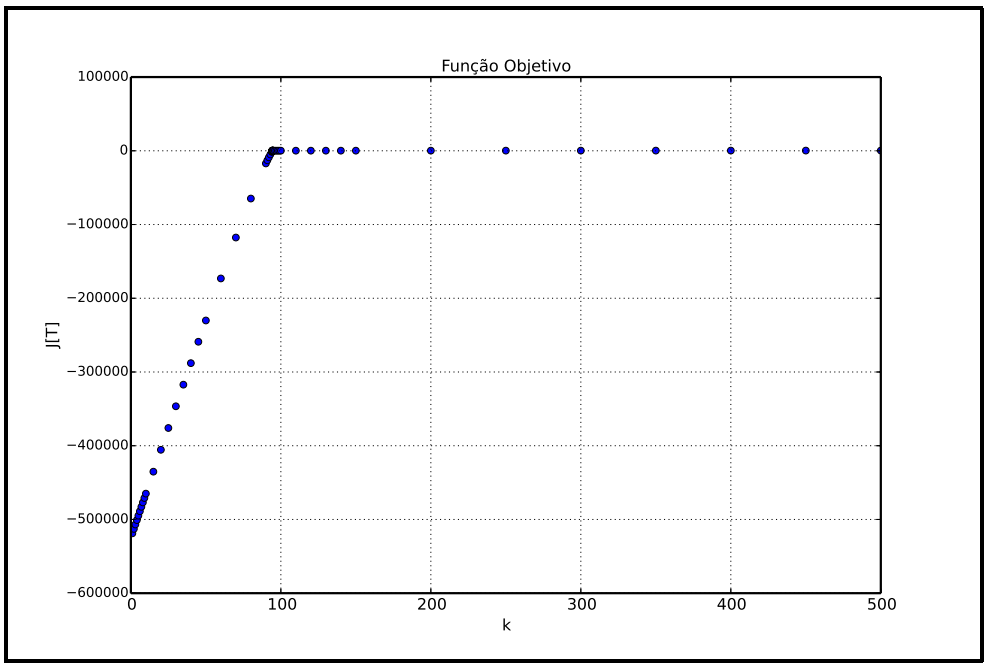

Figura 18: Valores Finais da Função Objetivo. Fonte: o autor.

5.5.5 *Análise do Contexto 2*

Verifica-se que a variação dos valores finais da função objetivo em relação ao parâmetro 'k' aparenta apresentar comportamento aproximadamente linear e claramente distinto em duas regiões separadas pela abscissa $k \approx 94,2438$. Mais ainda, especificamente para este valor, nós não conseguimos convergência na otimização.

Como era de se esperar, os resultados de todo o sistema também aparentam se dividir nestas duas regiões, conforme mostram as figuras de 19 a 23. Observa-se que, para $k > 94,2438$, os sinais de controle com machos estéreis são bem inferiores aos da outra região (em que $k < 94,2438$), e os sinais de controle com inseticidas passam a ter maior expressão no sistema.

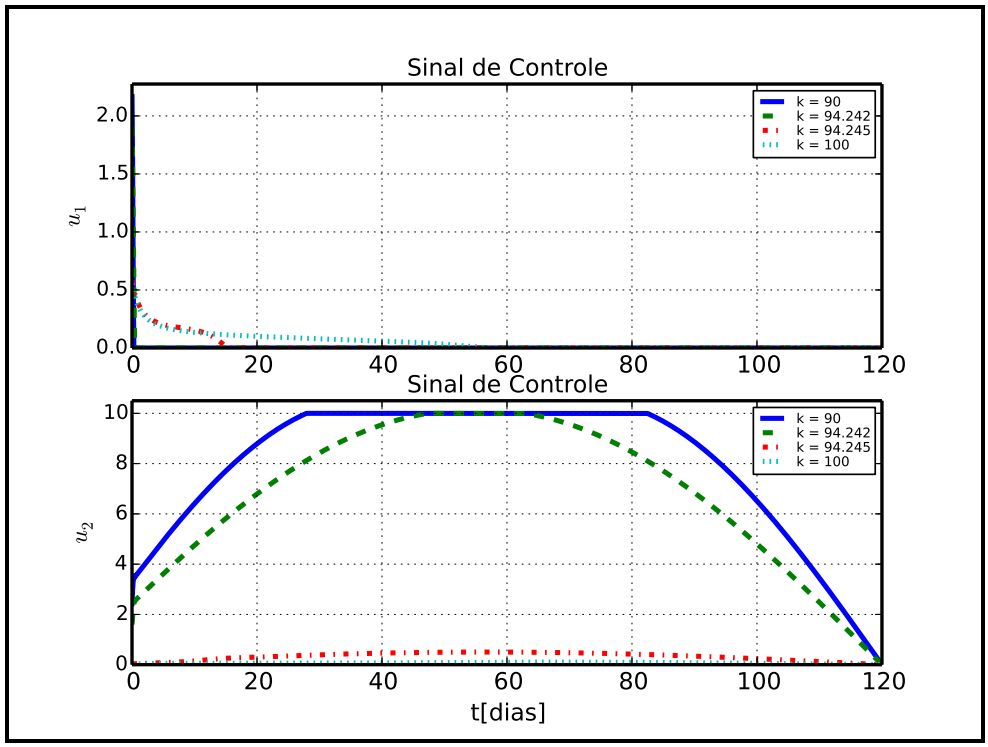

Figura 19: Sinais de Controle: Comparação entre Regiões. Fonte: o autor.

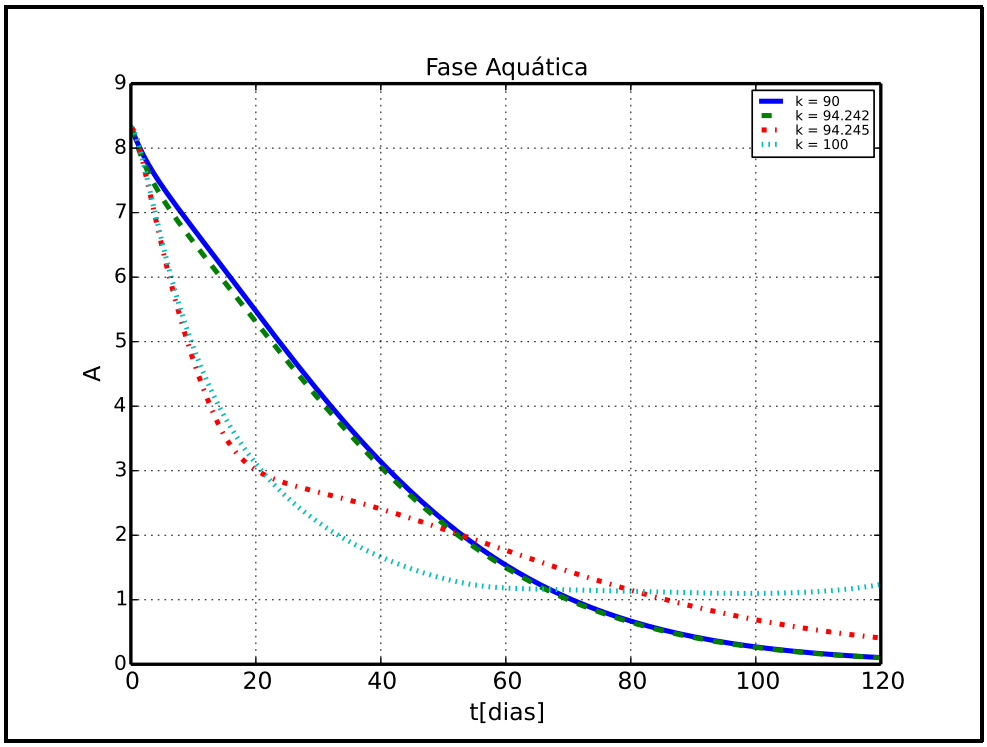

Figura 20: Fase Aquática: Comparação entre Regiões. Fonte: o autor.

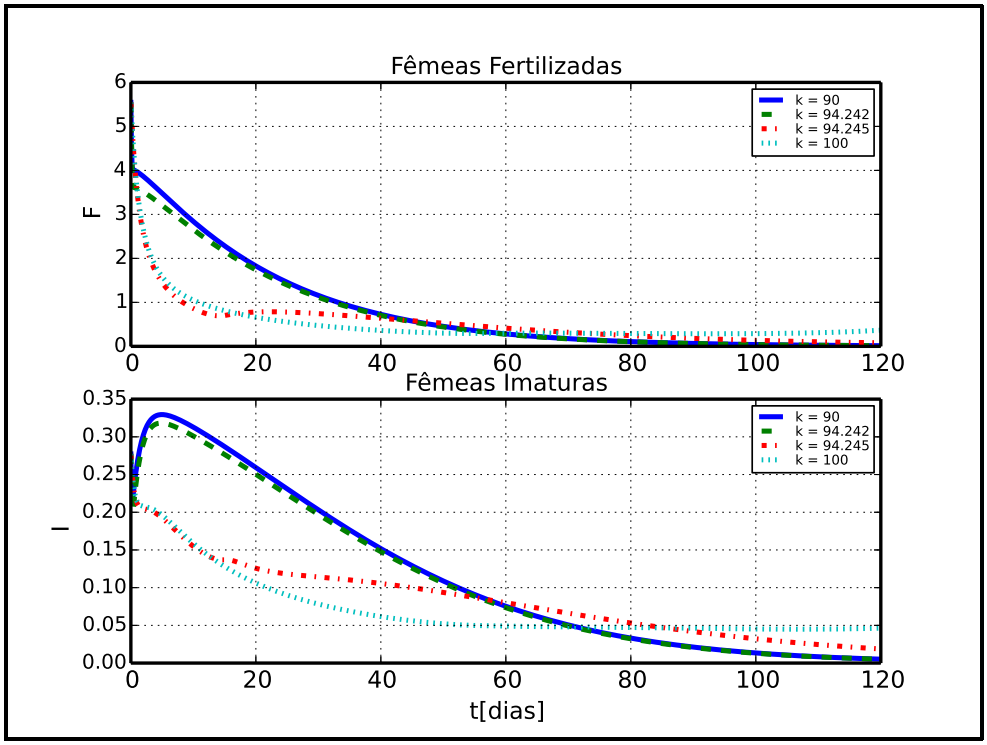

Figura 21: Populações de Fêmeas: Comparação entre Regiões. Fonte: o autor.

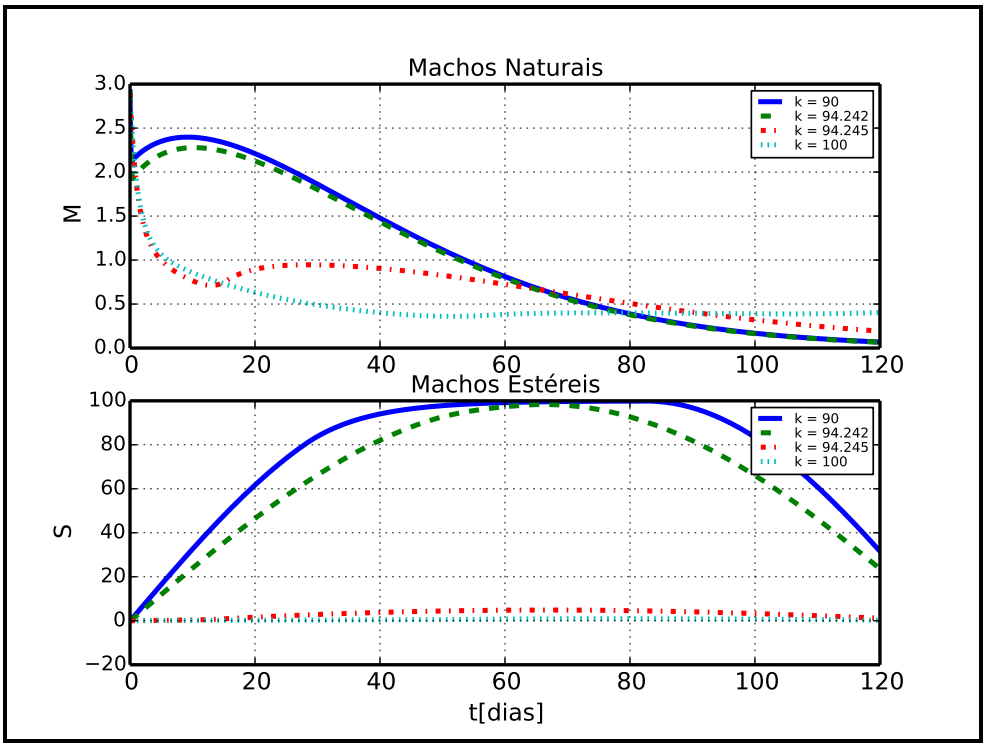

Figura 22: Populações de Machos: Comparação entre Regiões. Fonte: o autor.

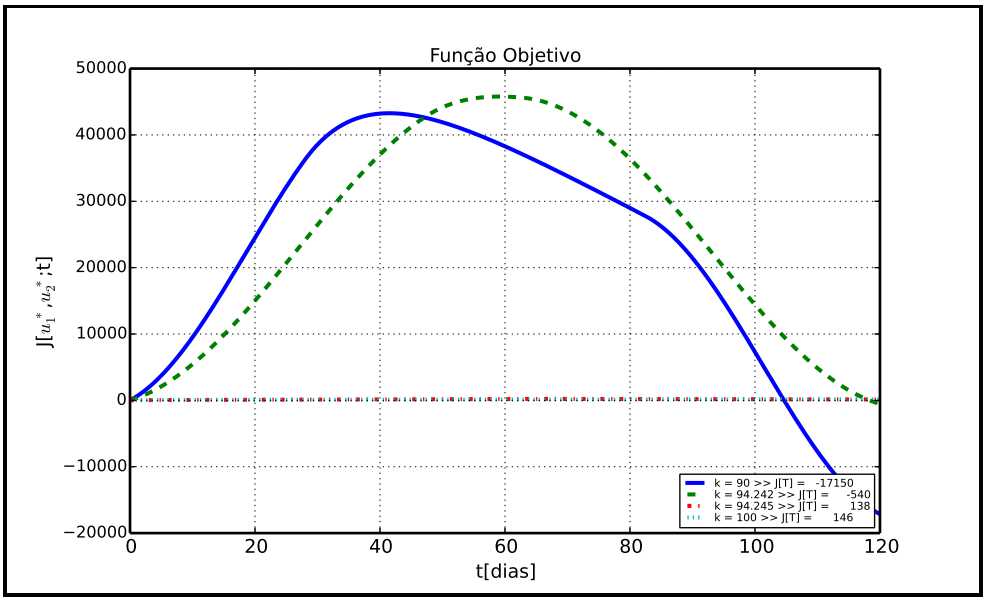

Figura 23: Comportamento da Função Objetivo: Comparação entre Regiões. Fonte: o autor.

5.5.6 Discussão do Contexto 2

Aparentemente, pela análise qualitativa dos gráficos anteriores, seria mais interessante ao gestor público de saúde operar com sinais de controle otimizados para região em que $k > 94,2438$. Os custos com controles serão menores e na maior parte do tempo haverá redução de população de fêmeas fertilizadas.

5.5.7 Otimizações do Contexto 3

Neste contexto, realizamos uma análise variacional baseada em simulações do comportamento do sistema. Escolhemos dois valores de 'k' na região um, $k < 94,2438$, e dois valores de 'k' na região dois, $k > 94,2438$. As otimizações para valores escolhidos já foram realizadas no contexto 1 e possuímos os respectivos sinais de controle otimizados.

Utilizando os valores de controle nominais calculamos controles alternativos pela inserção de dois tipos de variações, a primeira por um valor fixo proporcional[14] ao sinal e a segunda por um valor aleatório.

Simulamos o comportamento do sistema tendo todos estes sinais de controle por entrada e comparamos os resultados com os dos valores nominais.

Os gráficos das comparações para k=90, k=94,242, k=94,245 e k=100 são apresentados nos apêndices B a E.

5.5.8 Análise do Contexto 3

Pela observação doa gráficos, concluímos que o sistema como modelado é robusto às variações por distúrbios aleatórios. Para distúrbios de natureza fixa, haverá maiores variações na população de mosquitos estéreis e, consequentemente, no comportamento da função objetivo. Contudo, mesmo nestas situações as variações sobre a população de fêmeas fertilizadas é mínima.

5.5.9 Discussão do Contexto 3

Aparentemente os resultados são consistentes, demonstrando uma certa robustez do modelo a pequenos distúrbios.

[14] Na realidade o programa de análise usou valores fixos de 10% e 20% do sinal, mas permite também inserir um valor DC que escolhemos ser nulo.

5.5.10 *Otimização para controles constantes em períodos de 10 dias*

Considerando k=100 e o caso 3, isto é, $c_i = [1,1,1,1]$, fizemos a otimização para controles constantes em períodos de 10 dias. Os resultados são apresentados a seguir:

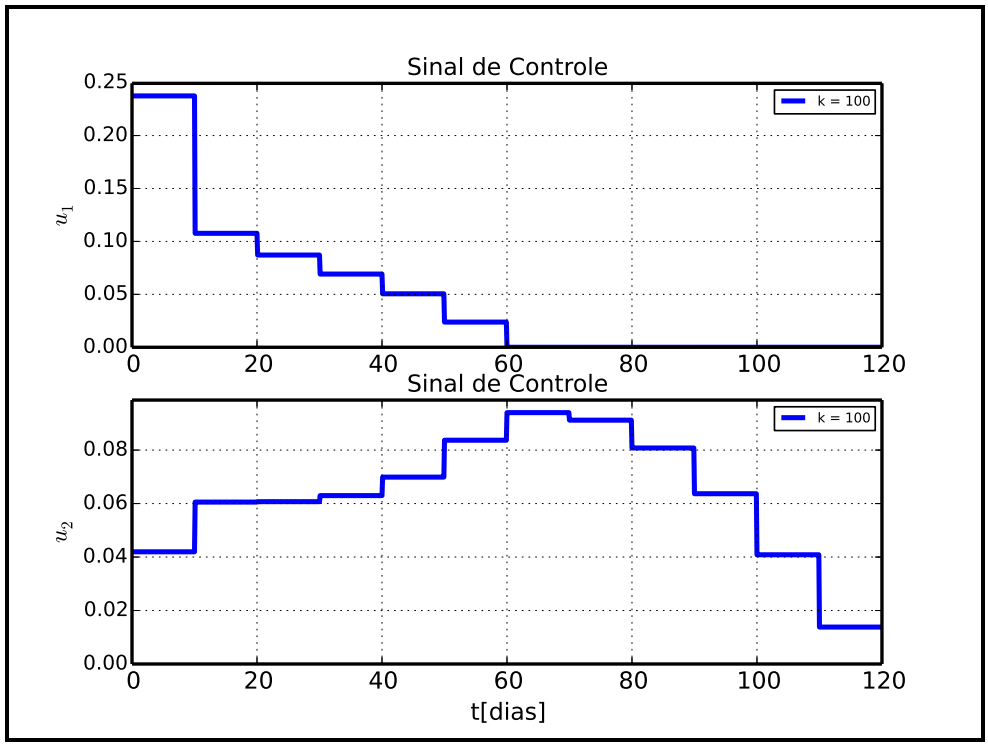

Figura 24: Sinais de Controle: Controles Constantes em Períodos de 10 Dias. Fonte: o autor.

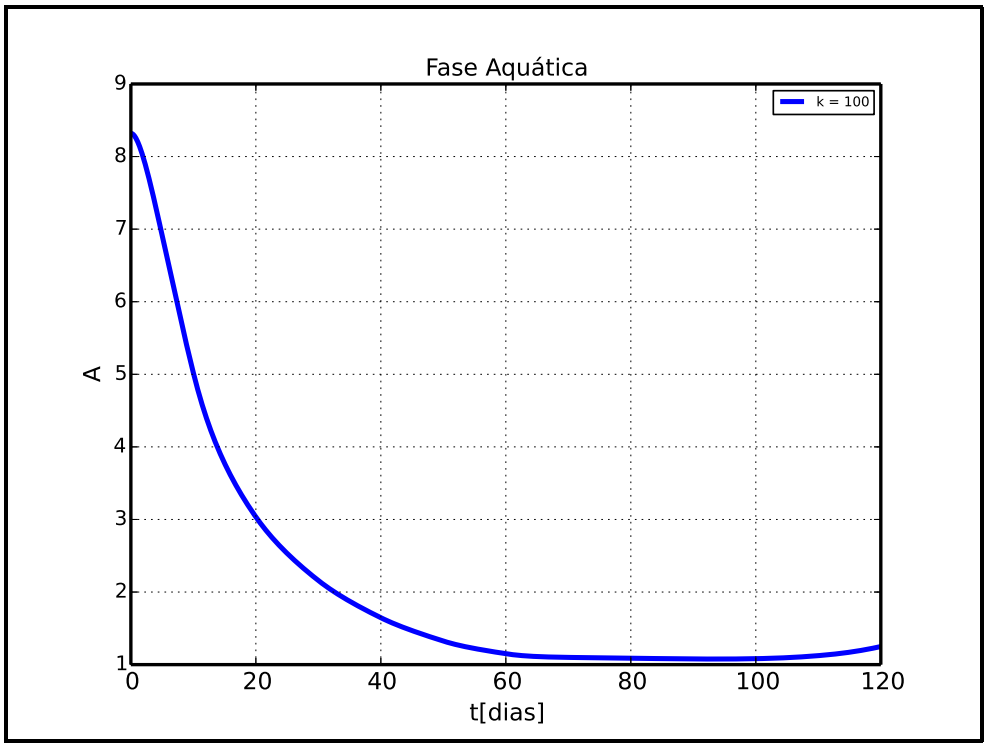

Figura 25: Fase Aquática: Controles Constantes em Períodos de 10 Dias. Fonte: o autor.

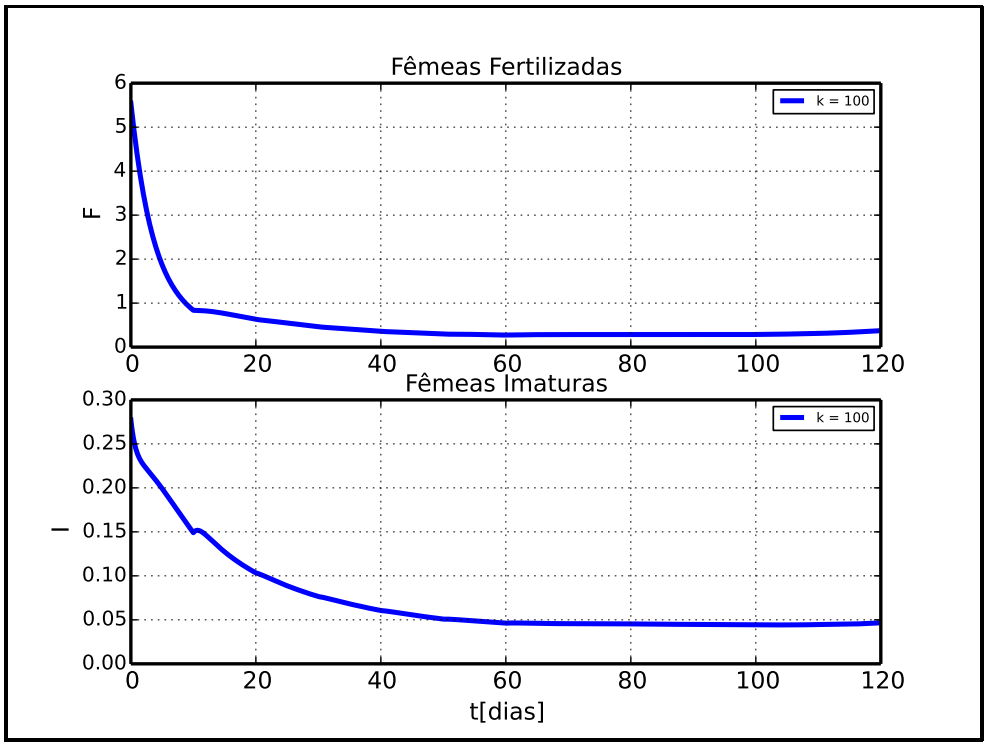

Figura 26: Populações de Fêmeas: Controles Constantes em Períodos de 10 Dias. Fonte: o autor.

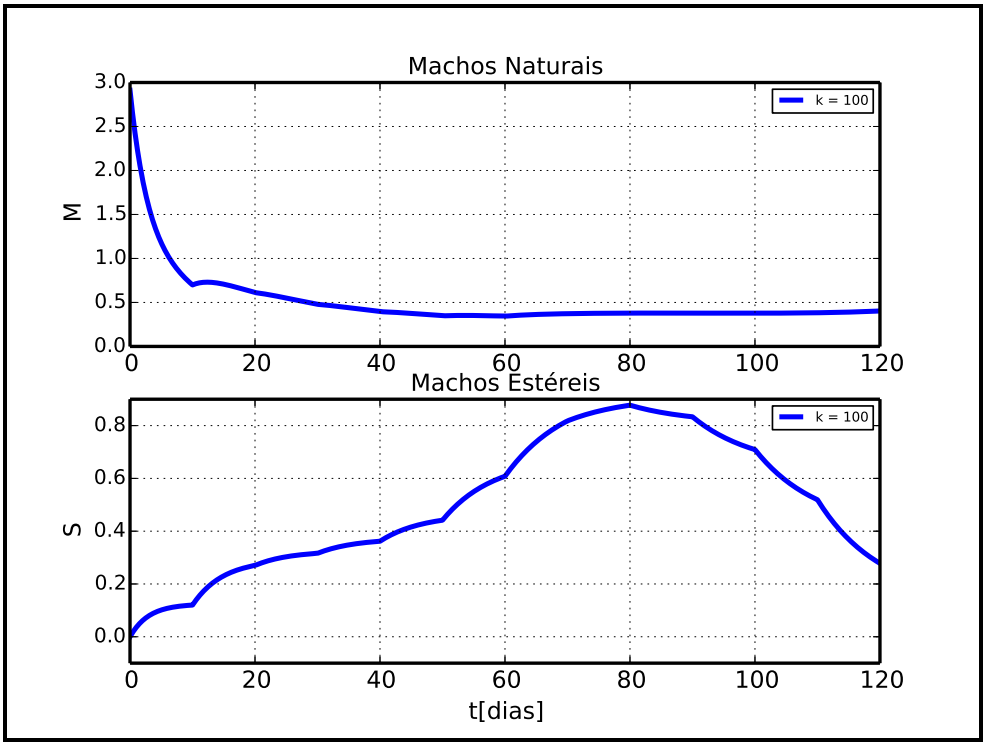

Figura 27: Populações de Machos: Controles Constantes em Períodos de 10 Dias. Fonte: o autor.

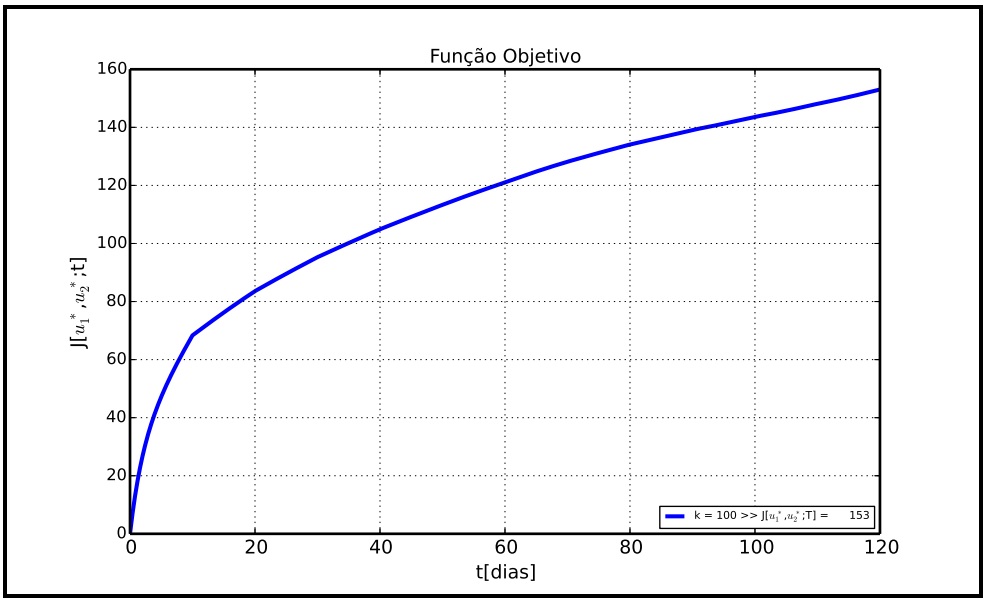

Figura 28: Comportamento da Função Objetivo: Controles Constantes em Períodos de 10 Dias. Fonte: o autor.

6 CONCLUSÕES

Este trabalho apresentou os resultados da pesquisa sobre técnicas de otimização aplicadas à Engenharia de Controle. Todos os objetivos definidos no capítulo 1 foram atingidos, em especial pela implantação de um ambiente integrado de modelagem, otimização e simulação.

A técnica de otimização dinâmica por pontos de colocação se mostrou eficaz, versátil e flexível para ajustar com facilidade parametrizações do modelo e da função objetivo, de forma a poder tratar maior diversidade de PCO. Em particular, foi possível evidenciar, no problema estudado, algumas características da função objetivo que, uma vez ajustadas, permitiram maior eficácia dos sinais de controle.

Trabalhos futuros poderão explorar as potencialidades do ambiente instalado para estudar as possibilidades de otimizações de PCO com funções objetivo não conformes com a teoria clássica de controle ótimo. Certamente é possivel tentar a aplicação da técnica em PCO com índices de desempenho definidos por funções não convexas que, mesmo sendo matematicamente mais difíceis de serem solucionados, venham a representar problemas reais de maior interesse.

Dentro do mesmo problema estudado, o ambiente instalado permite flexibilidade do modelo e do PCO para inserir com facilidade um sinal de controle adicional (controle que atue na fase aquática, por exemplo) ou investigar mais detalhadamente a sensibilidade a variações de parâmetros.

Seria interessante, também, tentar pesquisar os recursos das ferramentas para tratar sistemas discretos.

Por último, uma interessante possibilidade seria tentar aplicar a otimização dinâmica por pontos de colocação, à semelhança do que foi feito com o PMP em (IACOVIELLO; STASIO, 2013), em um modelo mais genérico de transmissão de doenças.

Desta forma, entendemos que este trabalho representou não somente um avanço no estudo do PCO da transmissão da dengue, mas também uma outra referência de aplicação de simulação numérica na solução de Problemas de Controle Ótimo de maior complexidade.

REFERÊNCIAS

ÅKESSON, J.; ÅRZÉN, K.-E.; GÄFVERT, M.; BERGDAHL, T.; TUMMESCHEIT, H. Modeling and optimization with Optimica and JModelica.org—languages and tools for solving large-scale dynamic optimization problems. *Computers and Chemical Engineering*, v. 34, n. 11, p. 1737–1749, nov. 2010. Disponível em: <http://www.control.lth.se/documents/2010-/ake+10cace.pdf>.

ALLGÖWER, F.; FINDEISEN, R. An introduction to nonlinear model predictive control. *21st Benelux Meeting on Systems and Control*, 2002. Disponível em: <http://www.ist.uni-stuttgart-.de/reports/pdf/2002-1.pdf>.

ALLGÖWER, F.; FINDEISEN, R.; NAGY, Z. K. Nonlinear model predictive control:from theory to application. *J. Chin. Inst. Chem. Engrs*, v. 25, n. 3, p. 299–315, 2004. Disponível em: <http://www2.ist.uni-stuttgart.de/˜findeise/papers/chinchemeng04.pdf>.

ANDERSSON, J.; ÅKESSON, J.; CASELLA, F.; DIEHL, M. Integration of CasADi and JModelica.org. In: *8th International Modelica Conference 2011*. Dresden, Germany: [s.n.], 2011. Disponível em: <http://www.control.lth.se/documents/2011/and+11mod11.pdf>.

ATHANS, M. *Optimal control an introduction to the theory and its applications*. New York: Dover Publications, 2007. ISBN 0486453286.

BANNWART, B. F. *Otimização Multiobjetivo no Controle da Dengue*. Dissertação (Mestrado) — Unesp, 2013. Disponível em: <http://repositorio.unesp.br/bitstream/handle/11449/108598-/000757973.pdf?sequence=1>.

BARSANTE, L. S.; CARDOSO, R. T. N.; ACEBAL, J. L. Otimizando custos no combate da dengue através de algoritmos genéticos. In: *X SBAI - Simpósio Brasileiro de Automação e Controle*. [s.n.], 2013. X, p. 27–32. ISSN 2175-8905. Disponível em: <http://fei.edu.br/sbai-/SBAI2011/87344.pdf>.

BAZARAA, M. S.; JARVIS, J. J.; SHERALI, H. D. *Linear Programming and Network Flows*. [S.l.]: Wiley, 2009. ISBN 0470462728.

BAZARAA, M. S.; SHETTY, C. M. *Nonlinear Programming: Theory and Algorithms*. [S.l.]: Wiley, 1979. ISBN 0471786101.

BOYCE, W. *Equacoes diferenciais elementares e problemas de valores de contorno*. Rio de Janeiro: Guanabara Koogan, 1990. ISBN 85-7030-0581.

BRYSON, A. *Applied optimal control*. City: Hemisphere, 1975. ISBN 9780891162285.

ESTEVA, L.; YANG, H. M. Mathematical model to assess the control of aedes aegypt mosquitoes by the sterile insect technique. *Mathematical Biosciences*, 2005. Disponível em: <http:/-

/www.sciencedirect.com/science/article/pii/S0025556405001045>.

FRANKLIN, G. F.; ETC.; POWELL, D.; EMAMI-NAEINI, A. *Feedback and Control of Dynamic Systems (Addison-Wesley series in electrical and computer engineering)*. 2nd. ed. [S.l.]: Addison Wesley Longman Publishing Co, 1991. ISBN 9780201508628.

FRANKLIN, G. F.; POWELL, J. *Digital Control of Dynamic Systems*. [S.l.]: Addison Wesley, 1980. ISBN 9780201028911.

GOLNARAGHI, F.; KUO, B. C. *Automatic Control Systems*. [S.l.]: Wiley, 2009. ISBN 0470048964.

GOLTEN, J.; VERWER, A. *Control System Design and Simulation*. Singapore: McGraw-Hill, 1992. ISBN 0071126309.

HEMERLY, E. M. *Controle por Computador de Sistemas Dinâmicos*. [S.l.]: Editora Edgard Blucher LTDA, 1996.

IACOVIELLO, D.; STASIO, N. Optimal control for sirc epidemic outbreak. *Computer Methods and Programs in Biomedicine*, Elsevier, v. 110, n. 3, p. 333–342, 2013. Disponível em: <http://dx.doi.org/10.1016/j.cmpb.2013.01.006>.

KELLER, H. *Numerical Solution of Two Point Boundary Value Problems*. [S.l.]: Society for Industrial and Applied Mathematics, 1976.

KIRK, D. *Optimal control theory; an introduction*. Englewood Cliffs, N.J: Prentice-Hall, 1970. ISBN 0486434842.

LEWIS, F. L. *Optimal Control*. [S.l.]: Wiley-Interscience, 1986. ISBN 0471812404.

LUENBERGER, D. G. *Linear and Nonlinear Programming*. [S.l.]: Addison-Wesley, 1984. ISBN 0201157942.

MAGNUSSON, F.; ÅKESSON, J. Collocation methods for optimization in a modelica environment. In: *Proceedings of the 9th International Modelica Conference*. [s.n.], 2012. Disponível em: <http://lup.lub.lu.se/record/2972280>.

OGATA, K. *Engenharia de Controle Moderno*. Segunda edição. [S.l.]: Printice Hall do Brasil, 1993. ISBN 8-57054-045-0.

OGATA, K. *Discrete-Time Control Systems (2nd Edition)*. [S.l.]: Prentice Hall, 1995. ISBN 0133286428.

SANDEL, M. *Harvard University's Justice: What's the Right Thing to Do?* 2009. Disponível em: <http://www.justiceharvard.org/2011/02/episode-two/>.

Science and Technology Facilities Council. *HSL: A collection of Fortran codes for large scale scientific computation*. 2015. Disponível em: <http://www.hsl.rl.ac.uk>.

THOMÉ, R. C.; YANG, H. M.; ESTEVA, L. Optimal control of aedes aegypti mosquitoes by the sterile insect technique and insecticide. *Mathematical Biosciences*, 2010. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0025556409001473>.

THOMÉ, R. C. A. *Controle Ótimo Aplicado na Estratégia de Combate ao Aedes Aegypti Utilizando Inseticida e Mosquitos Estéreis*. Tese (Doutorado) — Unicamp, 2007. Disponível em: <http://www.bibliotecadigital.unicamp.br/document/?code=vtls000427930>.

WÄCHTER, A.; BIEGLER, L. T. On the Implementation of a Primal-Dual Interior Point Filter Line Search Algorithm for Large-Scale Nonlinear Programming. *Mathematical Programming*, 2006. Disponível em: <http://cepac.cheme.cmu.edu/pasilectures/biegler/ipopt.pdf>.

APÊNDICE A - CÓDIGOS FONTE UTILIZADOS

Os principais códigos fonte utilizados na pesquisa são apresentados aqui.

O código a seguir corresponde ao arquivo *DenguePack.mop*:

A.1 MODELO OPTIMICA

```
package DenguePack
// Licensed by Ranulfo Acir de Oliveira Resende under the Modelica License 2
// Copyright Â© 2015, Ranulfo Acir de Oliveira Resende.

// This Modelica package is free software and the use is completely at your
// own risk; it can be redistributed and/or modified under the terms of the
// Modelica License 2. For license conditions (including the disclaimer of
// warranty) visit http://www.modelica.org/licenses/ModelicaLicense2.
//
// Ranulfo Acir de Oliveira Resende (acir_r@yahoo.com.br).

model Dengue

//Model Parameters
parameter Real phi = 0.50;
parameter Real C = 13;
parameter Real gama = 0.07;
parameter Real r = 0.50;
parameter Real beta = 1;
parameter Real beta_S = 0.70;
parameter Real mu_A = 0.05;
parameter Real mu_I = 0.05;
parameter Real mu_F = 0.05;
parameter Real mu_M = 0.10;
parameter Real mu_S = 0.10;

// Initial parameters;
parameter Real R = phi*r*gama*beta/((gama+mu_A)*(beta+mu_I)*mu_F); // 2.77777
parameter Real A0 = C*(R-1)/R                    ; //8.3200
parameter Real I0 = r*gama*A0/(mu_I+beta)        ; //0.2773
parameter Real F0 = (gama+mu_A)*C*A0/(phi*(C-A0)) ; //5.5467
parameter Real M0 = (1-r)*gama*A0/mu_M           ; //2.9120
```

```
parameter Real S0 = 0                                    ; //0

//Optimization Parameters
parameter Real s_time = 0.0;
parameter Real f_time = 120.0;
parameter Real t_time = f_time;
parameter Real F_time = 10*F0;

// Problem Variables - Mosquitoes Population;
output Real A (start = A0, fixed = true)  ; // Fase Aquatica
output Real I (start = I0, fixed = true)  ; // Femeas Imaturas
output Real F (start = F0, fixed = true)  ; // Femeas Fertilizadas
output Real M (start = M0, fixed = true)  ; // Machos Naturais
output Real S (start = S0, fixed = true)  ; // Machos Estereis

// Control Variables
input Real u1 (free = true)  ; // control 1
input Real u2 (free = true)  ; // control 2

equation
der(A) = phi*(1-A/C)*F-(gama+mu_A)*A;
der(I) = r*gama*A-(beta*M/(M+S)+beta_S*S/(M+S)+(mu_I+u1))*I;
der(F) = beta*M*I/(M+S)-(mu_F+u1)*F;
der(M) = (1-r)*gama*A-(mu_M+u1)*M;
der(S) = u2-(mu_S+u1)*S;

end Dengue;

optimization Dengue_Opt(objective = fo(finalTime),startTime = s_time,
    finalTime = f_time)

parameter Real c_1 = 1;
parameter Real c_2 = 1;
parameter Real c_3 = 1;
parameter Real c_4 = 1;
parameter Real k=1;
parameter Real u1_min = 0.0;
parameter Real u2_min = 0.0;
parameter Real u1_max = 10.0;
parameter Real u2_max = 10.0;

// Objective function
Real fo(start=0,fixed=true);

//Instance of Dengue model
extends Dengue;

```

```
equation
der(fo) = 0.5*(k*(c_1*u1^2+c_2*u2^2)+c_3+F^2-c_4*S^2);

constraint
// constraints
u1 >= u1_min; u2 >= u2_min;
A>=0; I>=0; F>=0; M>=0; S>=0;
u1 <= u1_max; u2 <= u2_max;
F(t_time) <= F_time;

end Dengue_Opt;
end DenguePack;
```

A.2 *SCRIPTS* PYTHON

O pequeno tutorial para utilizar os *scripts* Python é apresentado a seguir:

A.2.1 LEIAME.txt

```
Copyright (C)  2015, Ranulfo Acir de Oliveira Resende.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.3
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
http://www.gnu.org/licenses/fdl.html.

Antes de rodar os scripts deve-se configurar o ambiente da seguinte forma:
No sistema operacional Linux,

a) Definir variável de ambiente $USERCODEDIR
export USERCODEDIR=<seu caminho para uma pasta de trabalho>

b) Criar as pastas para resultados e para os modelos Modelica:
mkdir   $USERCODEDIR/modelos
mkdir   $USERCODEDIR/resultados

c) Copiar arquivo do modelo para $USERCODEDIR/modelos:
cp  <seu caminho para o modelo>/DenguePack.mop $USERCODEDIR/modelos

d) Executar o script otimiza.py no ambiente iterativo Python com a ferramenta
JModelica instalada:
run otimiza.py
e) Executar o script analiza.py no ambiente iterativo Python com a ferramenta
JModelica instalada:
run analisa.py
```

O arquivo *otimiza.py*, *script* Python para compilar o arquivo modelica e gerar os resultados, é apresentado a seguir.

A.2.2 O script otimiza.py

```
#!/usr/bin/python
# coding=UTF-8

```

```
4  # Copyright © 2015, Ranulfo Acir de Oliveira Resende.
5  #
6  # This program is free software: you can redistribute it and/or modify
7  # it under the terms of the GNU General Public License as published by
8  # the Free Software Foundation, either version 3 of the License, or
9  # (at your option) any later version.
10
11 # This program is distributed in the hope that it will be useful,
12 # but WITHOUT ANY WARRANTY; without even the implied warranty of
13 # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
14 # GNU General Public License for more details.
15
16 # You should have received a copy of the GNU General Public License
17 # along with this program.  If not, see <http://www.gnu.org/licenses/>.
18
19 # To make this software work, it needs to call IPOPT compiled with
20 # The HSL Mathematical Software Library (http://www.hsl.rl.ac.uk/)
     routuines,
21 # that are available under HSL ACADEMIC LICENCE VERSION 1.1 JANUARY 2011.
22
23 # Ranulfo Acir de Oliveira Resende (acir_r@yahoo.com.br)
24
25 # Tradução não oficial:
26 # Este programa é um software livre; você pode redistribuí-lo e/ou
27 # modificá-lo dentro dos termos da Licença Pública Geral GNU como
28 # publicada pela Fundação do Software Livre (FSF); na versão 3 da
29 # Licença, ou (na sua opinião) qualquer versão.
30 #
31 # Este programa é distribuído na esperança de que possa ser útil,
32 # mas SEM NENHUMA GARANTIA; sem uma garantia implícita de ADEQUAÇÃO
33 # a qualquer MERCADO ou APLICAÇÃO EM PARTICULAR. Veja a
34 # Licença Pública Geral GNU para maiores detalhes.
35
36 # Você deve ter recebido uma cópia da Licença Pública Geral GNU junto
37 # com este programa. Se não, veja <http://www.gnu.org/licenses/>.
38 #
39 # Esta tradução é informal, e não aprovada oficialmente pela
40 # Free Software Foundation como válida. Para ter certeza do que é
41 # permitido, refira-se à GPL original (em inglês).
42
43 # Para fazer este software funcionar, ele precisa chamar o IPOPT compilado
44 # com rotinas "The HSL Mathematical Software Library" (http://www.hsl.rl.ac
     .uk/),
45 # que estão disponíveis sob a licença "HSL ACADEMIC LICENCE VERSION 1.1
     JANUARY 2011".
46
47
48 import matplotlib
49 import matplotlib.pyplot as plt
50 import numpy as N
51 from pymodelica import compile_jmu
52 from pyjmi.jmi import JMUModel
53 from datetime import datetime
54 matplotlib.interactive(False)  # para não abrir figura na tela
55 import collections
56 import sys, os
57 import scipy.io as sio
```

```
58
59 sys.path.append(os.environ['PREFIX']+"/Python")
60 sys.path.append(os.environ['PREFIX']+"/lib/python2.7/dist-packages")
61
62 def my_range(start, end, step):
63     while start <= end:
64         yield start
65         start += step
66
67 def aplic_param(k, d, sim):
68     global dir_path
69
70     # model parameters
71     model_param = dict()
72
73     model_param['phi'] = 0.50
74     model_param['C'] = 13
75     model_param['gama'] = 0.07
76     model_param['r'] = 0.50
77     model_param['beta'] = 1.0
78     model_param['beta_S'] = 0.70
79     model_param['mu_A'] = 0.05
80     model_param['mu_I'] = 0.05
81     model_param['mu_F'] = 0.05
82     model_param['mu_M'] = 0.1
83     model_param['mu_S'] = 0.1
84
85     # model_optimization parameters
86     model_opt_param = dict()
87     model_opt_param['c_1'] = 1
88     model_opt_param['c_2'] = 1
89     model_opt_param['c_3'] = 1
90     model_opt_param['c_4'] = 1
91     model_opt_param['k']= k
92
93     # model_optimization parameters
94     # t_time = f_time significa valor final
95     model_opt_param['s_time'] = 0.0 #tempo dias
96     model_opt_param['f_time'] = 120.0
97
98     model_opt_param['u1_min'] = 0.0
99     model_opt_param['u2_min'] = 0.0
100
101    # Definir estes valores juntos
102    model_opt_param['u1_max'] = 10.0
103    model_opt_param['u2_max'] = 10.0
104    u1_max = model_opt_param['u1_max']
105    u2_max = model_opt_param['u2_max']
106
107    # Definir estes valores juntos
108    model_opt_param['t_time'] = 0        # t_time = 0 não tem efeito, vale
   o default do modelo
109    model_opt_param['F_time'] = 0.1
110    t_time = model_opt_param['t_time']
111    F_time = model_opt_param['F_time']
112
113    dias = d    # Dias (periodo) de combate a dengue
114                # (entradas de controle piecewise)
```

```
115             # Deve ser fator de horizonte de tempo em dias (f_time-
    s_time)
116             # só tem significado se dias > 0
117
118     # definiçoes da simulação: de Lu, Lc e LF
119     if sim == 1:
120         simula = "SIM"+str(sim)+"/"
121         n_e = 240 #240  n_e*n_cp multiplo de 120
122         n_cp = 5  #5
123         tol = 1e-8
124         acceptable_tol = 1e-6
125         acceptable_iter = 3
126
127         LuCtrFixo = [[[0, 0.1], [0, 0.02]],
128                      [[0, 0.08], [0, 0.01]],
129                      [[0, 0.07], [0, 0.05]],
130                      [[0, 0.04], [0, 0.08]]]
131
132         LuCtrCont = [[[0, u1_max], [0, u2_max]],
133                      [[0, u1_max], [0, u2_max]],
134                      [[0, u1_max], [0, u2_max]],
135                      [[0, u1_max], [0, u2_max]]]
136
137         LuCtrDisc = [[[0, u1_max], [0, u2_max]],
138                      [[0, u1_max], [0, u2_max]],
139                      [[0, u1_max], [0, u2_max]],
140                      [[0, u1_max], [0, u2_max]]]
141
142
143         LF = [[0, 0.1],  # LF = [t_time, F_time]
144               [0, 0.1],  # Tem que ter muito cuidado aqui
145               [0, 0.1],  # Porque restricoes impossíveis
146               [0, 0.1]]  # de alcancar tornam o problema
147                          # insolúvel ao IPOPT e trava
148            #  [60, 0.1]# o script python dando erro
149                        # Se for t_time = 0 nao tem efeito na otimização
150                        # Se for t_time = 120 representa restrição final
    para F
151
152         Lc = [[1, 10, 100, 1],  # ci para os casos 1 a 4
153               [1, 10, 1, 1],
154               [1, 1, 1, 1],
155               [3, 1, 1, 1]]
156         if dias < 5:
157             Lu = LuCtrCont
158         elif dias > 100:
159             Lu = LuCtrFixo
160         else:
161             Lu = LuCtrDisc
162
163     elif sim == 2:
164         simula = "SIM"+str(sim)+"/"
165         n_e = 240 #240  n_e*n_cp multiplo de 120
166         n_cp = 5  #5
167         tol = 1e-8
168         acceptable_tol = 1e-6
169         acceptable_iter = 3
170         Lc = [[1, 1, 1, 1]]
```

```
171         Lu = [[[0, u1_max], [0, u2_max]]]
172         LF = [[0, 0.1]]
173 
174     elif sim == 3:
175         simula = "SIM"+str(sim)+"/"
176         n_e = 240 #240  n_e*n_cp multiplo de 120
177         n_cp = 5  #5
178         tol = 1e-8
179         acceptable_tol = 1e-6
180         acceptable_iter = 3
181         LuCtrFixo = [[[0, 0.1], [0, 0.02]],
182                      [[0, 0.08], [0, 0.01]],
183                      [[0, 0.07], [0, 0.05]],
184                      [[0, 0.04], [0, 0.08]]]
185 
186         LuCtrCont = [[[0, u1_max], [0, u2_max]],
187                      [[0, u1_max], [0, u2_max]],
188                      [[0, u1_max], [0, u2_max]],
189                      [[0, u1_max], [0, u2_max]]]
190 
191         LuCtrDisc = [[[0, u1_max], [0, u2_max]],
192                      [[0, u1_max], [0, u2_max]],
193                      [[0, u1_max], [0, u2_max]],
194                      [[0, u1_max], [0, u2_max]]]
195 
196         LF = [[0, 0.1],  # LF = [t_time, F_time]
197               [0, 0.1],  # Tem que ter muito cuidado aqui
198               [0, 0.1],  # Porque restricoes impossíveis
199               [0, 0.1]]  # de alcancar tornam o problema
200                          # insolúvel ao IPOPT e trava
201            #  [60, 0.1]# o script python dando erro
202                        # Se for t_time = 0 nao tem efeito na otimização
203                        # Se for t_time = 120 representa restrição final
    para F
204 
205         Lc = [[1, 10, 100, 1],
206               [1, 10, 1, 1],
207               [1, 1, 1, 1],
208               [3, 1, 1, 1]]
209         if dias < 5:
210             Lu = LuCtrCont
211         elif dias > 100:
212             Lu = LuCtrFixo
213         else:
214             Lu = LuCtrDisc
215 
216     elif sim == 4:
217         simula = "SIM"+str(sim)+"/"
218         n_e = 240 #240  n_e*n_cp multiplo de 120
219         n_cp = 5  #5
220         tol = 1e-8
221         acceptable_tol = 1e-6
222         acceptable_iter = 3
223         LuCtrFixo = [[[0, 0.1], [0, 0.02]],
224                      [[0, 0.08], [0, 0.01]],
225                      [[0, 0.07], [0, 0.05]],
226                      [[0, 0.04], [0, 0.08]]]
227 
```

```
228         LuCtrCont = [[[0, u1_max], [0, u2_max]],
229                      [[0, u1_max], [0, u2_max]],
230                      [[0, u1_max], [0, u2_max]],
231                      [[0, u1_max], [0, u2_max]]]
232 
233         LuCtrDisc = [[[0, u1_max], [0, u2_max]],
234                      [[0, u1_max], [0, u2_max]],
235                      [[0, u1_max], [0, u2_max]],
236                      [[0, u1_max], [0, u2_max]]]
237 
238         LF = [[0, 0.1],  # LF = [t_time, F_time]
239               [0, 0.1],  # Tem que ter muito cuidado aqui
240               [0, 0.1],  # Porque restricoes impossíveis
241               [0, 0.1]]  # de alcancar tornam o problema
242                          # insolúvel ao IPOPT e trava
243            #  [60, 0.1]# o script python dando erro
244                        # Se for t_time = 0 nao tem efeito na otimização
245                        # Se for t_time = 120 representa restrição final
    para F
246 
247         Lc = [[1, 10, 100, 1],
248               [1, 10, 1, 1],
249               [1, 1, 1, 1],
250               [3, 1, 1, 1]]
251         if dias < 5:
252             Lu = LuCtrCont
253         elif dias > 100:
254             Lu = LuCtrFixo
255         else:
256             Lu = LuCtrDisc
257 
258     elif sim == 5:
259         simula = "SIM"+str(sim)+"/"
260         n_e = 480 #240  n_e*n_cp multiplo de 120
261         n_cp = 10  #5
262         tol = 1e-9
263         acceptable_tol = 1e-7
264         acceptable_iter = 5
265         Lc = [[1, 1, 1, 1]]
266         Lu = [[[0, u1_max], [0, u2_max]]]
267         LF = [[0, 0.1]]
268 
269     elif sim == 6:
270         simula = "SIM"+str(sim)+"/"
271         n_e = 120 #240  n_e*n_cp multiplo de 120
272         n_cp = 5  #5
273         tol = 1e-8
274         acceptable_tol = 1e-6
275         acceptable_iter = 5
276         # Definir estes valores juntos
277         model_opt_param['u1_max'] = 12.0
278         model_opt_param['u2_max'] = 12.0
279         u1_max = model_opt_param['u1_max']
280         u2_max = model_opt_param['u2_max']
281         model_param['mu_S'] = 0.2
282         Lc = [[1, 1, 1, 1]]
283         Lu = [[[0, u1_max], [0, u2_max]]]
284         LF = [[0, 0.1]]
```

```
285
286     elif sim == 7:
287         simula = "SIM"+str(sim)+"/"
288         n_e = 120 #240  n_e*n_cp multiplo de 120
289         n_cp = 5  #5
290         tol = 1e-6
291         acceptable_tol = 1e-4
292         acceptable_iter = 5
293         # Definir estes valores juntos
294         model_opt_param['u1_max'] = 10.0
295         model_opt_param['u2_max'] = 10.0
296         u1_max = model_opt_param['u1_max']
297         u2_max = model_opt_param['u2_max']
298         model_param['mu_S'] = 0.1
299         Lc = [[1, 1, 1, 1]]
300         Lu = [[[0, u1_max], [0, u2_max]]]
301         LF = [[0, 0.1]]
302
303     elif sim == 8:
304         simula = "SIM"+str(sim)+"/"
305         n_e = 240 #240  n_e*n_cp multiplo de 120
306         n_cp = 10  #5
307         tol = 1e-8
308         acceptable_tol = 1e-6
309         acceptable_iter = 10
310         # Definir estes valores juntos
311         model_opt_param['u1_max'] = 10.0
312         model_opt_param['u2_max'] = 10.0
313         u1_max = model_opt_param['u1_max']
314         u2_max = model_opt_param['u2_max']
315         model_param['mu_S'] = 0.1
316         Lc = [[1, 1, 1, 1]]
317         Lu = [[[0, u1_max], [0, u2_max]]]
318         LF = [[0, 0.1]]
319
320     dir_path0 = dir_path+simula
321     os.system('mkdir '+dir_path0)
322     dir_path1 = dir_path0+"dias_"+str(d)+"/"
323     os.system('mkdir '+dir_path1)
324     dir_destino = dir_path1+"k_"+str(k)+"/"
325     os.system('export DIR_DESTINO = '+dir_destino)
326     os.system('rm -R '+dir_destino)
327     os.system('mkdir '+dir_destino)
328     os.system('mkdir '+dir_destino+"figuras")
329     os.system('mkdir '+dir_destino+"mat")
330     os.system('mkdir '+dir_destino+"ipopt")
331     file_log = "log"
332     arquivo_nome = file_log+'.txt'
333     arquivo_path = dir_destino+arquivo_nome
334
335     #L = ['ma27', 'ma57', 'ma86']
336     Ls = ['ma57']
337     #possiveis resolvedores lineares:
338             # rotinas HSL: 'ma27', 'ma57', 'ma77', 'ma86' e 'ma97'
339             # default 'mumps'
340             # se configurar as libs: 'pardiso', 'wsmp' e custom
341
342     fig_ext = ["eps", "jpg", "png"]
```

```
343
344     param = dict()
345     param['Lc'] = Lc
346     param['LF'] = LF
347     param['Ls'] = Ls
348     param['Lu'] = Lu
349     param['n_e'] = n_e
350     param['n_cp'] = n_cp
351     param['dir_destino'] = dir_destino
352     param['arquivo_path'] = arquivo_path
353     param['fig_ext'] = fig_ext
354     param['tol'] = tol
355     param['acceptable_tol'] = acceptable_tol
356     param['acceptable_iter'] = acceptable_iter
357
358     return model_param, model_opt_param, param
359
360 def language():
361
362     P_Frase_A = u"Fase Aquática"    # a letra u antes da frase permite
363     P_Frase_I = u"Fêmeas Imaturas"  # reconhecimento de caracteres
    português
364     P_Frase_F = u"Fêmeas Fertilizadas"
365     P_Frase_M = "Machos Naturais"
366     P_Frase_S = u"Machos Estéreis"
367     P_Label_Opt = "Otimizado"
368     P_Label_Sim = "Simulado"
369     P_Label_X = "t[dias]"
370     P_Controle = "Sinal de Controle"
371
372     I_Frase_A = "Aquatic Phase"
373     I_Frase_I = "Imature Famales"
374     I_Frase_F = "Fertilized Females"
375     I_Frase_M = "Natural Males"
376     I_Frase_S = "Sterile Males"
377     I_Label_Opt = "Optmized"
378     I_Label_Sim = "Simulated"
379     I_Label_X = "t[days]"
380     I_Controle = "Control Signal"
381
382     I_frases = [I_Frase_A,
383                 I_Frase_I,
384                 I_Frase_F,
385                 I_Frase_M,
386                 I_Frase_S,
387                 I_Label_Opt,
388                 I_Label_Sim,
389                 I_Label_X,
390                 I_Controle]
391     P_frases = [P_Frase_A,
392                 P_Frase_I,
393                 P_Frase_F,
394                 P_Frase_M,
395                 P_Frase_S,
396                 P_Label_Opt,
397                 P_Label_Sim,
398                 P_Label_X,
399                 P_Controle]
```

```
400
401     return I_frases, P_frases
402
403 def media(*args):
404     if not args:
405         return None
406     if len(args) == 1 and isinstance(args[0], collections.Container):
407         args = args[0]
408     total = sum(args)
409     ave = 1.0 * total / len(args)
410     return ave
411 def maximo(*args):
412     if not args:
413         return None
414     if len(args) == 1 and isinstance(args[0], collections.Container):
415         args = args[0]
416     mxm = max(args)
417     return mxm
418
419 def minimo(*args):
420     if not args:
421         return None
422     if len(args) == 1 and isinstance(args[0], collections.Container):
423         args = args[0]
424     mnm = min(args)
425     return mnm
426
427 # Definicao de funcoes
428 def def_parametros(f, solver, n_e, n_cp, s_time, f_time, tol,
    acceptable_tol, acceptable_iter):
429
430     # Optimization (IPOPT) parameters
431     max_cpu_time = float(72*60*60) # ao atingir tempo de max_cpu_time (em
    seg)
432     max_iter = 500000              # ao atingir numero de iteracoes =
    max_ier
433
434
435     #tol = 1e-7  # default 1e-8 converge ao atingir esta tol uma vez em
    todos
436                 # os parametros converge if the (scaled) NLP error becomes
437                 # smaller than this value, and if the (absolute) criteria
438                 # according to 'dual_inf_tol', 'primal_inf_tol',
439                 # and 'compl_inf_tol' are met.
440
441                 # Overall NLP error = equacao complicada
442
443     dual_inf_tol = 1.0         # default 1.0      campo inf_du
444     constr_viol_tol = 1e-4     # default 1e-4     campo inf_pr
445     compl_inf_tol = 1e-4       # default 1e-4     campo lg(mu) corresponde a
    mu
446
447     # Segunda condicao de parada por convergencia (tolerancias relaxadas)
448     # Converge ao atingir estas tolerancias por acceptable_iter iteracoes
449    # acceptable_tol = 1e-5# default 1e-6 converge ao atingir acceptable_tol
450                         # por acceptable_iter vezes em todos os
    parametros
451    # acceptable_iter = 3   # default 15
```

```
452     acceptable_dual_inf_tol = 1e5       # default 1e10
453     acceptable_constr_viol_tol = 0.01    # default 0.01
454     acceptable_compl_inf_tol = 0.01      # default 0.01
455
456     print ""
457     print "
    ************************************************************"
458     print "Parametros gerais:"
459     print "-- Solucionador Linear:"
460     print "solver = ", solver
461     print "-- Tempo inicial:"
462     print "s_time", s_time
463     print "-- Tempo final:"
464     print "f_time", f_time
465     print ""
466     print "
    ************************************************************"
467     print "Parametros de discretização:"
468     print "a) numero de elementos"
469     print "n_e = ", n_e
470     print "b) numero de pontos de colocacao"
471     print "n_cp = ", n_cp
472     print " "
473     print "
    ************************************************************"
474     print 'Primeira condicao de parada - convergencia (tolerancias
    rigorosas)'
475     print 'Converge ao atingir estas tolerancias na na primeira vez'
476     print 'tol = ', tol, ' default 1e-8'
477     print 'converge ao atingir esta tol uma vez em todos os parametros'
478     print 'converge if the (scaled) NLP error becomes smaller than'
479     print 'this value, and if the (absolute) criteria'
480     print 'according to  dual_inf_tol,  primal_inf_tol, '
481     print 'and  compl_inf_tol  are met.'
482     print 'dual_inf_tol = ', dual_inf_tol, ' default 1.0'
483     print 'constr_viol_tol = ', constr_viol_tol, ' default 1e-4'
484     print 'compl_inf_tol = ', compl_inf_tol, ' default 1e-4'
485     print ""
486     print "
    ************************************************************"
487     print 'Segunda condicao de parada - convergencia (tolerancias
    relaxadas)'
488     print 'Converge ao atingir estas tolerancias em acceptable_iter
    iteracoes'
489     print ""
490     print 'acceptable_tol = ', acceptable_tol, \
491     'default 1e-6 converge ao atingir acceptable_tol por'
492     print 'acceptable_iter vezes em todos os parametros'
493     print 'acceptable_iter = ', acceptable_iter, '    default 15'
494     print 'acceptable_dual_inf_tol= ', acceptable_dual_inf_tol, ' default
    1e10'
495     print 'acceptable_constr_viol_tol= ', acceptable_constr_viol_tol ,\
496     ' default 0.01'
497     print 'acceptable_compl_inf_tol= ', acceptable_compl_inf_tol ,\
498     ' default 0.01'
499     print ""
500     print "
    ************************************************************"
```

```
501     print 'Parada por limite de tempo ou do numero máximo de iteracoes'
502     print ""
503     print 'Interrompe o processamento e diverge ao atingir max_cpu_time \
504 ou max_iter:'
505     print 'a) ao atingir tempo de max_cpu_time (em segundos)'
506     print "max_cpu_time = ", max_cpu_time
507     print 'b) ao atingir o numero de iteracoes = max_iter'
508     print "max_iter = ", max_iter
509     print ""
510
511     print >> f, ""
512     print >> f, "
    ************************************************************"
513     print >> f, "Parametros gerais:"
514     print >> f, "-- Solucionador Linear:"
515     print >> f, "solver = ", solver
516     print >> f, "-- Tempo inicial:"
517     print >> f, "s_time", s_time
518     print >> f, "-- Tempo final:"
519     print >> f, "f_time", f_time
520     print >> f, ""
521     print >> f, "
    ************************************************************"
522     print >> f, "Parametros de discretizacao:"
523     print >> f, "a) numero de elementos"
524     print >> f, "n_e = ", n_e
525     print >> f, "b) numero de pontos de colocacao"
526     print >> f, "n_cp = ", n_cp
527     print >> f, " "
528     print >> f, "
    ************************************************************"
529     print >> f, 'Primeira condicao de parada por convergencia \
530 (tolerancias rigorosas)'
531     print >> f, 'Converge ao atingir estas tolerancias na na primeira vez'
532     print >> f, 'tol = ', tol, ' default 1e-8 converge ao atingir esta tol
    uma\
533 vez em todos os parametros'
534     print >> f, 'converge if the (scaled) NLP error becomes smaller than'
535     print >> f, 'this value, and if the (absolute) criteria'
536     print >> f, 'according to  dual_inf_tol,  primal_inf_tol, '
537     print >> f, 'and  compl_inf_tol  are met.'
538     print >> f, 'dual_inf_tol = ', dual_inf_tol, ' default 1.0'
539     print >> f, 'constr_viol_tol = ', constr_viol_tol, '  default 1e-4'
540     print >> f, 'compl_inf_tol = ', compl_inf_tol, '      default 1e-4'
541     print >> f, ""
542     print >> f, "
    ************************************************************"
543     print >> f, 'Segunda condicao de parada por convergencia (tolerancias \
544 relaxadas)'
545     print >> f, 'Converge ao atingir estas tolerancias por acceptable_iter
    \
546 iteracoes'
547     print >> f, ""
548     print >> f, 'acceptable_tol = ', acceptable_tol, 'default 1e-6 converge
    ao\
549 atingir acceptable_tol por'
550     print >> f, 'acceptable_iter vezes em todos os parametros'
551     print >> f, 'acceptable_iter = ', acceptable_iter, ' default 15'
```

```
552     print >> f, 'acceptable_dual_inf_tol = ', acceptable_dual_inf_tol, ' \
553 default 1e10'
554     print >> f, 'acceptable_constr_viol_tol = ', acceptable_constr_viol_tol
        , '\
555 default 0.01'
556     print >> f, 'acceptable_compl_inf_tol = ', acceptable_compl_inf_tol, '
        \
557 default 0.01'
558     print >> f, ""
559     print >> f, "
        ************************************************************"
560     print >> f, 'Parada por limite de tempo ou do numero maximo de
        iteracoes'
561     print >> f, ""
562     print >> f, 'Interrompe o processamento e diverge ao atingir
        max_cpu_time \
563 ou max_iter:'
564     print >> f, 'a) ao atingir tempo de max_cpu_time (em segundos)'
565     print >> f, "max_cpu_time = ", max_cpu_time
566     print >> f, 'b) ao atingir o numero de iteracoes = max_iter'
567     print >> f, "max_iter = ", max_iter
568     print >> f, ""
569
570     lista_opt = dict()
571     lista_opt['n_e'] = n_e
572     lista_opt['n_cp'] = n_cp
573     lista_opt['max_cpu_time'] = max_cpu_time
574     lista_opt['max_iter'] = max_iter
575     lista_opt['tol'] = tol
576     lista_opt['acceptable_tol'] = acceptable_tol
577     lista_opt['acceptable_iter'] = acceptable_iter
578     lista_opt['solver'] = solver
579     lista_opt['dual_inf_tol'] = dual_inf_tol
580     lista_opt['constr_viol_tol'] = constr_viol_tol
581     lista_opt['compl_inf_tol'] = compl_inf_tol
582     lista_opt['acceptable_dual_inf_tol'] = acceptable_dual_inf_tol
583     lista_opt['acceptable_constr_viol_tol'] = acceptable_constr_viol_tol
584     lista_opt['acceptable_compl_inf_tol'] = acceptable_compl_inf_tol
585
586     return lista_opt
587
588 def otimiza(f,
589             model_opt,
590             fig,
591             model_param,
592             model_opt_param,
593             lista_opt,
594             k,
595             dias,
596             ci,
597             ui,
598             t_time,
599             F_time,
600             param):
601     s_time = model_opt_param['s_time']
602     f_time = model_opt_param['f_time']
603
604     u1_min = ui[0][0]
```

```
605     u2_min = ui[1][0]
606     u1_max = ui[0][1]
607     u2_max = ui[1][1]
608 
609     n_e = lista_opt['n_e']
610     n_cp = lista_opt['n_cp']
611     max_cpu_time = lista_opt['max_cpu_time']
612     max_iter = lista_opt['max_iter']
613     tol = lista_opt['tol']
614     acceptable_tol = lista_opt['acceptable_tol']
615     acceptable_iter = lista_opt['acceptable_iter']
616     solver = lista_opt['solver']
617     dual_inf_tol = lista_opt['dual_inf_tol']
618     constr_viol_tol = lista_opt['constr_viol_tol']
619     compl_inf_tol = lista_opt['compl_inf_tol']
620     acceptable_dual_inf_tol = lista_opt['acceptable_dual_inf_tol']
621     acceptable_constr_viol_tol = lista_opt['acceptable_constr_viol_tol']
622     acceptable_compl_inf_tol = lista_opt['acceptable_compl_inf_tol']
623 
624     caso = param['caso']
625     dir_destino = param['dir_destino']
626 
627     opt_opts = model_opt.optimize_options()
628 
629     opt_opts["n_e"] = n_e
630     opt_opts["n_cp"] = n_cp
631     opt_opts["result_mode"] = "mesh_interpolation"
632     n = n_e*n_cp+1
633     t = N.linspace(s_time, f_time, n)
634     if dias > 0:
635         n_periodo = int((f_time-s_time)/dias)
636         bf = [int(n_e/n_periodo)]*n_periodo # bf eh lista de elementos
637                                             # (nao de pontos de colocacao)
638         print "Implementando Controle com Entradas Piecewise:"
639         print "periodo em dias = ", dias
640         print "numero de periodos = ", n_periodo
641         print "blocking_factors = ", bf
642 
643         print >> f, "Implementando Controle com Entradas Piecewise:"
644         print >> f, "período em dias = ", dias
645         print >> f, "numero de períodos = ", n_periodo
646         print >> f, "blocking_factors = ", bf
647 
648         opt_opts["blocking_factors"] = bf
649 
650     opt_opts["result_mesh"] = t
651     opt_opts['IPOPT_options']['tol'] = tol
652     opt_opts["IPOPT_options"]["max_iter"] = max_iter
653     opt_opts['IPOPT_options']['max_cpu_time'] = max_cpu_time
654     opt_opts['IPOPT_options']['dual_inf_tol'] = dual_inf_tol
655     opt_opts['IPOPT_options']['constr_viol_tol'] = constr_viol_tol
656     opt_opts['IPOPT_options']['compl_inf_tol'] = compl_inf_tol
657     opt_opts['IPOPT_options']['acceptable_tol'] = acceptable_tol
658     opt_opts["IPOPT_options"]["acceptable_iter"] =\
659     acceptable_iter
660     opt_opts['IPOPT_options']['acceptable_constr_viol_tol'] =\
661     acceptable_constr_viol_tol
662     opt_opts['IPOPT_options']['acceptable_dual_inf_tol'] =\
```

```
663     acceptable_dual_inf_tol
664     opt_opts['IPOPT_options']['acceptable_compl_inf_tol'] =\
665     acceptable_compl_inf_tol
666     opt_opts['IPOPT_options']['linear_solver'] = solver
667     opt_opts['IPOPT_options']['mu_max'] = 1e5
668     opt_opts['IPOPT_options']['mu_max_fact'] = 1e3
669     opt_opts["IPOPT_options"]["mu_min"] = 1e-200
670     opt_opts['IPOPT_options']['mu_strategy'] = "adaptive"
671     opt_opts['IPOPT_options']['barrier_tol_factor'] = 10.0
672     opt_opts['IPOPT_options']['adaptive_mu_globalization'] =\
673     'obj-constr-filter'
674     opt_opts['IPOPT_options']['alpha_for_y'] = 'safer-min-dual-infeas'
675     opt_opts['IPOPT_options']['print_level'] = 5
676     opt_opts['IPOPT_options']['output_file'] = dir_destino+"ipopt/"+caso+\
677     "_ipopt_output.txt"
678     opt_opts['IPOPT_options']['file_print_level'] = 5
679     print caso
680     print >> f, caso
681
682     model_opt.set('s_time', s_time)
683     print "s_time do modelo = ", model_opt.get('s_time')
684     print  >> f, "s_time do modelo = ", model_opt.get('s_time')
685     model_opt.set('f_time', f_time)
686     print "f_time do modelo = ", model_opt.get('f_time')
687     print >> f, "f_time do modelo = ", model_opt.get('f_time')
688
689     model_opt.set('c_1', ci[0])
690     model_opt.set('c_2', ci[1])
691     model_opt.set('c_3', ci[2])
692     model_opt.set('c_4', ci[3])
693     model_opt.set('k', k)
694
695     c1_model = model_opt.get('c_1')
696     c2_model = model_opt.get('c_2')
697     c3_model = model_opt.get('c_3')
698     c4_model = model_opt.get('c_4')
699     k_model = model_opt.get('k')
700
701
702     print "c_1 do modelo = ", c1_model
703     print  >> f, "c_1 do modelo = ", c1_model
704     print "c_2 do modelo = ", c2_model
705     print  >> f, "c_2 do modelo = ", c2_model
706     print "c_3 do modelo = ", c3_model
707     print  >> f, "c_3 do modelo = ", c3_model
708     print "c_4 do modelo = ", c4_model
709     print  >> f, "c_4 do modelo = ", c4_model
710
711     print "Fator corretor dos sinais de controle 'k' = ", k_model
712     print  >> f, "Fator corretor dos sinais de controle 'k' = ", k_model
713
714     model_opt.set('u1_min', u1_min)
715     model_opt.set('u2_min', u2_min)
716     model_opt.set('u1_max', u1_max)
717     model_opt.set('u2_max', u2_max)
718
719     u1_min_model = model_opt.get('u1_min')
720     u1_max_model = model_opt.get('u1_max')
```

```
721     u2_min_model = model_opt.get('u2_min')
722     u2_max_model = model_opt.get('u2_max')
723 
724     print "u1_min do modelo = ", u1_min_model
725     print  >> f, "u1_min do modelo = ", u1_min_model
726     print "u2_min do modelo = ", u2_min_model
727     print  >> f, "u2_min do modelo = ", u2_min_model
728     print "u1_max do modelo = ", u1_max_model
729     print  >> f, "u1_max do modelo = ", u1_max_model
730     print "u2_max do modelo = ", u2_max_model
731     print  >> f, "u2_max do modelo = ", u2_max_model
732 
733     if t_time > 0:
734         model_opt.set('t_time', t_time)
735         model_opt.set('F_time', F_time)
736         t_time_model = model_opt.get('t_time')
737         F_time_model = model_opt.get('F_time')
738 
739         print "t_time do modelo = ", t_time_model
740         print  >> f, "t_time do modelo = ", t_time_model
741         print "F_time do modelo = ", F_time_model
742         print  >> f, "F_time do modelo = ", F_time_model
743 
744         print "F(", t_time_model, ") = ", F_time_model
745         print  >> f, "F(", t_time_model, ") = ", F_time_model
746     else:
747         print "Este caso não tem em restrição sobre F"
748         print  >> f, "Este caso não tem em restrição sobre F"
749 
750     res_objeto = model_opt.optimize(options=opt_opts)
751 
752     return res_objeto, opt_opts,  model_opt
753 
754 def graficos(f,
755              res,
756              fig,
757              model,
758              model_opt,
759              model_param,
760              model_opt_param,
761              n_e,
762              n_cp,
763              k,
764              dias,
765              detalhe_comp,
766              detalhe_sim,
767              detalhe_opt,
768              param):
769 
770     global grade
771 
772     caso = param['caso']
773 
774     u1_lim_model = [model_opt.get('u1_min'), model_opt.get('u1_max')]
775     u2_lim_model = [model_opt.get('u2_min'), model_opt.get('u2_max')]
776 
777     s_time = model_opt.get('s_time')
778     f_time = model_opt.get('f_time')
```

```
779
780     t_time_model = model_opt.get('t_time')
781     F_time_model = model_opt.get('F_time')
782
783     A_opt = res['A']
784     F_opt = res['F']
785     I_opt = res['I']
786     M_opt = res['M']
787     S_opt = res['S']
788
789     ci_1 = model_opt.get('c_1')
790     ci_2 = model_opt.get('c_2')
791     ci_3 = model_opt.get('c_3')
792     ci_4 = model_opt.get('c_4')
793     #k = model_opt.get('k')
794     ci_model = [ci_1, ci_2, ci_3, ci_4]
795
796     R_model = model_opt.get('R')
797
798     A0_model = model_opt.get('A0')
799     F0_model = model_opt.get('F0')
800     I0_model = model_opt.get('I0')
801     M0_model = model_opt.get('M0')
802     S0_model = model_opt.get('S0')
803
804     I_Frases = param['I_Frases']
805     I_Frase_A = I_Frases[0]
806     I_Frase_I = I_Frases[1]
807     I_Frase_F = I_Frases[2]
808     I_Frase_M = I_Frases[3]
809     I_Frase_S = I_Frases[4]
810     I_Label_Opt = I_Frases[5]
811     I_Label_Sim = I_Frases[6]
812     I_Label_X = I_Frases[7]
813     I_Controle = I_Frases[8]
814
815     P_Frases = param['P_Frases']
816     P_Frase_A = P_Frases[0]
817     P_Frase_I = P_Frases[1]
818     P_Frase_F = P_Frases[2]
819     P_Frase_M = P_Frases[3]
820     P_Frase_S = P_Frases[4]
821     P_Label_Opt = P_Frases[5]
822     P_Label_Sim = P_Frases[6]
823     P_Label_X = P_Frases[7]
824     P_Controle = P_Frases[8]
825
826     dir_destino = param['dir_destino']
827
828     fig_ext = param['fig_ext']
829
830     [return_status,
831      nbr_iter,
832      objetive,
833      total_exec_time] = res.solver.opt_coll_ipopt_get_statistics()
834
835     print "Parâmetros para ", caso, " verificados do modelo:"
836     print "ci = ", ci_model
```

```
837     print "u1 = ", u1_lim_model
838     print "u2 = ", u2_lim_model
839     print "k = ", k
840     print "d = ", dias
841     print "A0 = ", A0_model
842     print "F0 = ", F0_model
843     print "I0 = ", I0_model
844     print "M0 = ", M0_model
845     print "S0 = ", S0_model
846     print "R = ", R_model
847     print "Saida IPOPT = ", return_status
848     print "Numero de iteracoes = ", nbr_iter
849     print "Funcao objetivo = ", objetive
850     print "Tempo para otimização = ", total_exec_time
851 
852     print >> f, "Parâmetros para ", caso, " verificados do modelo:"
853     print >> f, "ci = ", ci_model
854     print >> f, "u1 = ", u1_lim_model
855     print >> f, "u2 = ", u2_lim_model
856     print >> f, "k = ", k
857     print >> f, "d = ", dias
858     print >> f, "A0 = ", A0_model
859     print >> f, "F0 = ", F0_model
860     print >> f, "I0 = ", I0_model
861     print >> f, "M0 = ", M0_model
862     print >> f, "S0 = ", S0_model
863     print >> f, "R = ", R_model
864     print >> f, "Saida IPOPT = ", return_status
865     print >> f, "Numero de iteracoes = ", nbr_iter
866     print >> f, "Funcao objetivo = ", objetive
867     print >> f, "Tempo para otimização = ", total_exec_time
868 
869     if t_time_model > 0 and t_time_model < 120.0:
870         print "Restrição sobre F:"
871         print "t_time = ", t_time_model
872         print "F_(", t_time_model, ") = ", F_time_model
873         print >> f, "Restrição sobre F:"
874         print >> f, "t_time = ", t_time_model
875         print >> f, "F(", t_time_model, ") = ", F_time_model
876     else:
877         print "Este caso não tem em restrição sobre F"
878         print  >> f, "Este caso não tem em restrição sobre F"
879 
880     u1_opt = res['u1']
881     u2_opt = res['u2']
882     t_opt = res['time']
883     fo_opt = res['fo']
884     Frase1 = "Soluções para Controles u1 e u2 Parcialmente Discretos"
885     Frase2 = "Perído de Controle = "+str(dias)+" dias"
886 
887     u1_axis = maximo(u1_opt)
888     if u1_axis < 0:
889         u1_axis = 1e-6
890         for i in range(0, len(u1_opt)):
891             u1_opt[i] = 0
892 
893     u2_axis = maximo(u2_opt)
894     if u2_axis < 0:
```

```
895         u2_axis = 1e-6
896         for i in range(0, len(u2_opt)):
897             u2_opt[i] = 0
898 
899     if dias > 119:
900         u1_axis = media(u1_opt)
901         u2_axis = media(u2_opt)
902 
903         print "Soluções para Controles u1 e u2 Constantes"
904         print "Média do Sinal de Controle u1_med = ", u1_axis
905         print "Média do Sinal de Controle u2_med = ", u2_axis
906         print >> f, "Soluções para Controles u1 e u2 Constantes"
907         print >> f, "Média do Sinal de Controle u1_med = ", u1_axis
908         print >> f, "Média do Sinal de Controle u2_med = ", u2_axis
909         Frase1 = ""
910         Frase2 = ""
911 
912     if dias < 1:
913         print "Soluções para Controles u1 e u2 totalmente Contínuos"
914         print >> f, "Soluções para Controles u1 e u2 totalmente Contínuos"
915         Frase1 = ""
916         Frase2 = ""
917 
918     print Frase1
919     print Frase2
920     print >> f, Frase1
921     print >> f, Frase2
922 
923     legenda = ["k = %g" % (k)]
924     print "legenda = ",legenda
925     print >> f,"legenda = ",legenda
926     legenda_obj = [""]
927 
928     fig = fig+1
929     plt.figure(fig)  # fig 2
930     plt.subplot(2, 1, 1)
931     plt.plot(t_opt, u1_opt, linewidth=3)
932     plt.legend(legenda)
933     if grade > 0:
934         plt.grid()
935     plt.ylim(0, u1_axis*1.05)
936 
937     ax = plt.gca()
938     for axis in ['top', 'bottom', 'left', 'right']:
939         ax.spines[axis].set_linewidth(2)
940     for tick in ax.xaxis.get_major_ticks():
941         tick.label.set_fontsize(10)
942 
943     plt.title(I_Controle, fontsize=10)
944 
945     plt.ylabel('$u_{1}$', fontsize=12)
946     plt.subplot(2, 1, 2)
947     plt.plot(t_opt, u2_opt, linewidth=3)
948     plt.legend(legenda)
949     if grade > 0:
950         plt.grid()
951     plt.ylim(0, u2_axis*1.05)
952     plt.ylabel('$u_{2}$', fontsize=12)
```

```
953     ax = plt.gca()
954     for axis in ['top', 'bottom', 'left', 'right']:
955         ax.spines[axis].set_linewidth(2)
956     for tick in ax.xaxis.get_major_ticks():
957         tick.label.set_fontsize(10)
958 
959     plt.title(I_Controle, fontsize=10)
960     plt.xlabel(I_Label_X, fontsize=10)
961     for i in range(0, len(fig_ext)):
962         plt.savefig(dir_destino+"/figuras/"+caso+"/"+fig_ext[i]+"/"+\
963         caso+"_Controles-opt-ing."+fig_ext[i])
964 
965     plt.title(P_Controle, fontsize=10)
966     plt.xlabel(P_Label_X, fontsize=10)
967 
968     plt.subplot(2, 1, 1)
969     plt.title(P_Controle, fontsize=10)
970     for i in range(0, len(fig_ext)):
971         plt.savefig(dir_destino+"/figuras/"+caso+"/"+fig_ext[i]+"/"+\
972         caso+"_Controles-opt."+fig_ext[i])
973 
974     plt.close()
975 
976     if detalhe_opt:
977         fig = fig+1
978         plt.figure(fig, figsize=(10, 6), dpi=100)  # fig 1
979         plt.plot(t_opt, fo_opt, linewidth=3)
980         if grade > 0:
981             plt.grid()
982         plt.ylabel('J[$u_{1}^{*}$,$u_{2}^{*}$;t]', fontsize=10)
983 
984         ax = plt.gca()
985         for axis in ['top', 'bottom', 'left', 'right']:
986             ax.spines[axis].set_linewidth(2)
987         for tick in ax.xaxis.get_major_ticks():
988             tick.label.set_fontsize(10)
989 
990         plt.title("Cost Function", fontsize=10)
991         plt.xlabel(I_Label_X, fontsize=10)
992         legenda_obj[0]=legenda[0]+" >> J[T] = %8.0f" % (fo_opt[len(fo_opt)
    -1])
993         plt.legend(legenda_obj, loc = 4)
994         print "legenda_obj = ",legenda_obj
995         print >> f,"legenda_obj = ",legenda_obj
996         for i in range(0, len(fig_ext)):
997             plt.savefig(dir_destino+"/figuras/"+caso+"/"+fig_ext[i]+"/"+\
998             caso+"_fo-opt-ing."+fig_ext[i])
999 
1000        plt.title(u"Função Objetivo", fontsize=10)
1001        plt.xlabel(P_Label_X, fontsize=10)
1002        for i in range(0, len(fig_ext)):
1003            plt.savefig(dir_destino+"/figuras/"+caso+"/"+fig_ext[i]+"/"+\
1004            caso+"_fo-opt."+fig_ext[i])
1005 
1006        plt.close()
1007 
1008        fig = fig+1
1009        plt.figure(fig)  # fig 1
```

```
1010        plt.plot(t_opt, A_opt, linewidth=3)
1011        plt.legend(legenda)
1012        if grade > 0:
1013            plt.grid()
1014        plt.ylabel('A', fontsize=10)
1015
1016        ax = plt.gca()
1017        for axis in ['top', 'bottom', 'left', 'right']:
1018            ax.spines[axis].set_linewidth(2)
1019        for tick in ax.xaxis.get_major_ticks():
1020            tick.label.set_fontsize(10)
1021
1022        plt.title(I_Frase_A, fontsize=10)
1023        plt.xlabel(I_Label_X, fontsize=10)
1024        for i in range(0, len(fig_ext)):
1025            plt.savefig(dir_destino+"/figuras/"+caso+"/"+fig_ext[i]+"/"+\
1026            caso+"_Fase_Aquatica-opt-ing."+fig_ext[i])
1027
1028        plt.title(P_Frase_A, fontsize=10)
1029        plt.xlabel(P_Label_X, fontsize=10)
1030        for i in range(0, len(fig_ext)):
1031            plt.savefig(dir_destino+"/figuras/"+caso+"/"+fig_ext[i]+"/"+\
1032            caso+"_Fase_Aquatica-opt."+fig_ext[i])
1033
1034        plt.close()
1035
1036        fig = fig+1
1037        plt.figure(fig)
1038        plt.subplot(2, 1, 1)
1039        plt.plot(t_opt, F_opt, linewidth=3)
1040        plt.legend(legenda)
1041        if grade > 0:
1042            plt.grid()
1043        plt.title(I_Frase_F, fontsize=10)
1044        plt.ylabel('F', fontsize=10)
1045
1046        ax = plt.gca()
1047        for axis in ['top', 'bottom', 'left', 'right']:
1048            ax.spines[axis].set_linewidth(2)
1049        for tick in ax.xaxis.get_major_ticks():
1050            tick.label.set_fontsize(10)
1051
1052        plt.subplot(2, 1, 2)
1053        plt.plot(t_opt, I_opt, linewidth=3)
1054        plt.legend(legenda)
1055        if grade > 0:
1056            plt.grid()
1057        plt.title(I_Frase_I, fontsize=10)
1058        plt.ylabel('I', fontsize=10)
1059        plt.xlabel(I_Label_X, fontsize=10)
1060
1061        ax = plt.gca()
1062        for axis in ['top', 'bottom', 'left', 'right']:
1063            ax.spines[axis].set_linewidth(2)
1064        for tick in ax.xaxis.get_major_ticks():
1065            tick.label.set_fontsize(10)
1066        for i in range(0, len(fig_ext)):
1067            plt.savefig(dir_destino+"/figuras/"+caso+"/"+fig_ext[i]+"/"+\
```

```
1068            caso+"_Femeas-opt-ing."+fig_ext[i])
1069
1070        plt.subplot(2, 1, 1)
1071        plt.title(P_Frase_F, fontsize=10)
1072        plt.subplot(2, 1, 2)
1073        plt.title(P_Frase_I, fontsize=10)
1074        plt.xlabel(P_Label_X, fontsize=10)
1075        for i in range(0, len(fig_ext)):
1076            plt.savefig(dir_destino+"/figuras/"+caso+"/"+fig_ext[i]+"/"+\
1077            caso+"_Femeas-opt."+fig_ext[i])
1078
1079        plt.close()
1080
1081        fig = fig+1
1082        plt.figure(fig)
1083        plt.subplot(2, 1, 1)
1084        plt.plot(t_opt, M_opt, linewidth=3)
1085        plt.legend(legenda)
1086        if grade > 0:
1087            plt.grid()
1088        plt.title(I_Frase_M, fontsize=10)
1089        plt.ylabel('M', fontsize=10)
1090
1091        ax = plt.gca()
1092        for axis in ['top', 'bottom', 'left', 'right']:
1093            ax.spines[axis].set_linewidth(2)
1094        for tick in ax.xaxis.get_major_ticks():
1095            tick.label.set_fontsize(10)
1096
1097        plt.subplot(2, 1, 2)
1098        plt.plot(t_opt, S_opt, linewidth=3)
1099        plt.legend(legenda)
1100        if grade > 0:
1101            plt.grid()
1102        plt.title(I_Frase_S, fontsize=10)
1103        plt.ylabel('S', fontsize=10)
1104        plt.xlabel(I_Label_X, fontsize=10)
1105
1106        ax = plt.gca()
1107        for axis in ['top', 'bottom', 'left', 'right']:
1108            ax.spines[axis].set_linewidth(2)
1109        for tick in ax.xaxis.get_major_ticks():
1110            tick.label.set_fontsize(10)
1111
1112        for i in range(0, len(fig_ext)):
1113            plt.savefig(dir_destino+"/figuras/"+caso+"/"+fig_ext[i]+"/"+\
1114            caso+"_Machos-opt-ing."+fig_ext[i])
1115
1116        plt.subplot(2, 1, 1)
1117        plt.title(P_Frase_M, fontsize=10)
1118        plt.subplot(2, 1, 2)
1119        plt.title(P_Frase_S, fontsize=10)
1120        plt.xlabel(P_Label_X, fontsize=10)
1121        for i in range(0, len(fig_ext)):
1122            plt.savefig(dir_destino+"/figuras/"+caso+"/"+fig_ext[i]+"/"+\
1123            caso+"_Machos-opt."+fig_ext[i])
1124
1125        plt.close()
```

```
1126
1127     # Load model
1128     sim_model = model  # O modelo do sistema sem o problema de controle
        otimizado
1129
1130     sim_model.set('s_time', s_time)
1131     print "s_time do modelo a ser simulado = ", sim_model.get('s_time')
1132     print  >> f, "s_time do modelo a ser simulado = ", sim_model.get('
        s_time')
1133     sim_model.set('f_time', f_time)
1134     print "f_time do modelo a ser simulado = ", sim_model.get('f_time')
1135     print >> f, "f_time do modelo a ser simulado = ", sim_model.get('f_time
        ')
1136
1137     print 'valores otimização'
1138     print 'n_e = ', n_e
1139     print 'n_cp = ', n_cp
1140     print 'start_time = ', s_time
1141     print 'final_time = ', f_time
1142
1143     u_traj = N.transpose(N.vstack((t_opt, u1_opt, u2_opt)))
1144
1145     input_obj = (['u1', 'u2'], u_traj)
1146
1147     sim_opts = sim_model.simulate_options()
1148
1149     sim_opts['ncp'] = n_e*n_cp
1150
1151     sim_opts["initialize"] = True
1152     res = sim_model.simulate(start_time=s_time, final_time=f_time,
1153                              input=input_obj, options=sim_opts)
1154     arq_sim_mat = "mat/"+caso+"_res_sim.mat"
1155     sio.savemat(dir_destino+arq_sim_mat, dict(res))
1156
1157     A_sim = res['A']
1158     F_sim = res['F']
1159     I_sim = res['I']
1160     M_sim = res['M']
1161     S_sim = res['S']
1162
1163     u1_sim = res['u1']
1164     u2_sim = res['u2']
1165
1166     t_sim = res['time']
1167
1168     print 'valores simulação'
1169     print 'n_e = ', n_e
1170     print 'n_cp = ', n_cp
1171     print 'start_time = ', s_time
1172     print 'final_time = ', f_time
1173
1174     if detalhe_sim:
1175         fig = fig+1
1176         plt.figure(fig)  # fig 4
1177         plt.plot(t_sim, A_sim, linewidth=2.0)
1178         plt.legend(legenda)
1179         if grade > 0:
1180             plt.grid()
```

```
1181        plt.title(I_Frase_A, fontsize=10)
1182        plt.ylabel('A', fontsize=10)
1183        plt.xlabel(I_Label_X, fontsize=10)
1184
1185        ax = plt.gca()
1186        for axis in ['top', 'bottom', 'left', 'right']:
1187            ax.spines[axis].set_linewidth(2)
1188        for tick in ax.xaxis.get_major_ticks():
1189            tick.label.set_fontsize(10)
1190        for i in range(0, len(fig_ext)):
1191            plt.savefig(dir_destino+"/figuras/"+caso+"/"+fig_ext[i]+"/"\
1192            +caso+"_Fase_Aquatica-sim-ing."+fig_ext[i])
1193
1194        plt.title(P_Frase_A, fontsize=10)
1195        plt.xlabel(P_Label_X, fontsize=10)
1196        for i in range(0, len(fig_ext)):
1197            plt.savefig(dir_destino+"/figuras/"+caso+"/"+fig_ext[i]+"/"\
1198            +caso+"_Fase_Aquatica-sim."+fig_ext[i])
1199
1200        plt.close()
1201
1202        fig = fig+1
1203        plt.figure(fig)  # fig 4
1204        plt.subplot(2, 1, 1)
1205        plt.plot(t_sim, F_sim, linewidth=2.0)
1206        plt.legend(legenda)
1207        if grade > 0:
1208            plt.grid()
1209        plt.title(I_Frase_F, fontsize=10)
1210        plt.ylabel('F', fontsize=10)
1211
1212        ax = plt.gca()
1213        for axis in ['top', 'bottom', 'left', 'right']:
1214            ax.spines[axis].set_linewidth(2)
1215        for tick in ax.xaxis.get_major_ticks():
1216            tick.label.set_fontsize(10)
1217
1218        plt.subplot(2, 1, 2)
1219        plt.plot(t_sim, I_sim, linewidth=2.0)
1220        plt.legend(legenda)
1221        if grade > 0:
1222            plt.grid()
1223        plt.title(I_Frase_I, fontsize=10)
1224        plt.ylabel('I', fontsize=10)
1225        plt.xlabel(I_Label_X, fontsize=10)
1226
1227        ax = plt.gca()
1228        for axis in ['top', 'bottom', 'left', 'right']:
1229            ax.spines[axis].set_linewidth(2)
1230        for tick in ax.xaxis.get_major_ticks():
1231            tick.label.set_fontsize(10)
1232        for i in range(0, len(fig_ext)):
1233            plt.savefig(dir_destino+"/figuras/"+caso+"/"+fig_ext[i]+"/"\
1234            +caso+"_Femeas-sim-ing."+fig_ext[i])
1235
1236        plt.subplot(2, 1, 1)
1237        plt.title(P_Frase_F, fontsize=10)
1238        plt.subplot(2, 1, 2)
```

```
1239        plt.title(P_Frase_I, fontsize=10)
1240        plt.xlabel(P_Label_X, fontsize=10)
1241        for i in range(0, len(fig_ext)):
1242            plt.savefig(dir_destino+"/figuras/"+caso+"/"+fig_ext[i]+"/"\
1243            +caso+"_Femeas-sim."+fig_ext[i])
1244
1245        plt.close()
1246
1247        fig = fig+1
1248        plt.figure(fig)  # fig 4
1249        plt.subplot(2, 1, 1)
1250        plt.plot(t_sim, M_sim, linewidth=2.0)
1251        plt.legend(legenda)
1252        if grade > 0:
1253            plt.grid()
1254        plt.title(I_Frase_M, fontsize=10)
1255        plt.ylabel('M', fontsize=10)
1256
1257        ax = plt.gca()
1258        for axis in ['top', 'bottom', 'left', 'right']:
1259            ax.spines[axis].set_linewidth(2)
1260        for tick in ax.xaxis.get_major_ticks():
1261            tick.label.set_fontsize(10)
1262
1263        plt.subplot(2, 1, 2)
1264        plt.plot(t_sim, S_sim, linewidth=2.0)
1265        plt.legend(legenda)
1266        if grade > 0:
1267            plt.grid()
1268        plt.title(I_Frase_S, fontsize=10)
1269        plt.ylabel('S', fontsize=10)
1270        plt.xlabel(I_Label_X, fontsize=10)
1271
1272        ax = plt.gca()
1273        for axis in ['top', 'bottom', 'left', 'right']:
1274            ax.spines[axis].set_linewidth(2)
1275        for tick in ax.xaxis.get_major_ticks():
1276            tick.label.set_fontsize(10)
1277        for i in range(0, len(fig_ext)):
1278            plt.savefig(dir_destino+"/figuras/"+caso+"/"+fig_ext[i]+"/"\
1279            +caso+"_Machos-sim-ing."+fig_ext[i])
1280
1281        plt.subplot(2, 1, 1)
1282        plt.title(P_Frase_M, fontsize=10)
1283        plt.subplot(2, 1, 2)
1284        plt.title(P_Frase_S, fontsize=10)
1285        plt.xlabel(P_Label_X, fontsize=10)
1286        for i in range(0, len(fig_ext)):
1287            plt.savefig(dir_destino+"/figuras/"+caso+"/"+fig_ext[i]+"/"\
1288            +caso+"_Machos-sim."+fig_ext[i])
1289
1290        plt.close()
1291
1292        fig = fig+1
1293        plt.figure(fig)  # fig 5
1294        plt.subplot(2, 1, 1)
1295        plt.plot(t_sim, u1_sim, linewidth=2.0)
1296        plt.legend(legenda)
```

```
1297            if grade > 0:
1298                plt.grid()
1299            plt.ylim(0, u1_axis*1.05)
1300            plt.title(I_Controle, fontsize=10)
1301            ax = plt.gca()
1302            for axis in ['top', 'bottom', 'left', 'right']:
1303                ax.spines[axis].set_linewidth(2)
1304            for tick in ax.xaxis.get_major_ticks():
1305                tick.label.set_fontsize(10)
1306
1307            plt.ylabel('$u_{1}$', fontsize=12)
1308            plt.subplot(2, 1, 2)
1309            plt.plot(t_sim, u2_sim, linewidth=2.0)
1310            plt.legend(legenda)
1311            if grade > 0:
1312                plt.grid()
1313            plt.ylim(0, u2_axis*1.05)
1314            plt.title(I_Controle, fontsize=10)
1315            plt.ylabel('$u_{2}$', fontsize=12)
1316            plt.xlabel(I_Label_X, fontsize=10)
1317
1318            ax = plt.gca()
1319            for axis in ['top', 'bottom', 'left', 'right']:
1320                ax.spines[axis].set_linewidth(2)
1321            for tick in ax.xaxis.get_major_ticks():
1322                tick.label.set_fontsize(10)
1323            for i in range(0, len(fig_ext)):
1324                plt.savefig(dir_destino+"/figuras/"+caso+"/"+fig_ext[i]+"/"\
1325                +caso+"_Controles-sim-ing."+fig_ext[i])
1326
1327            plt.subplot(2, 1, 1)
1328            plt.title(P_Controle, fontsize=10)
1329            plt.subplot(2, 1, 2)
1330            plt.title(P_Controle, fontsize=10)
1331            plt.xlabel(P_Label_X, fontsize=10)
1332            for i in range(0, len(fig_ext)):
1333                plt.savefig(dir_destino+"/figuras/"+caso+"/"+fig_ext[i]+"/"\
1334                +caso+"_Controles-sim."+fig_ext[i])
1335            plt.close()
1336
1337        if detalhe_comp:
1338            fig = fig+1
1339            plt.figure(fig)  # fig 7
1340            plt.plot(t_opt, A_opt, '--', linewidth=2.0)
1341            plt.plot(t_sim, A_sim, linewidth=2.0)
1342            plt.legend((I_Label_Opt, I_Label_Sim))
1343            if grade > 0:
1344                plt.grid()
1345            plt.title(I_Frase_A, fontsize=10)
1346            plt.ylabel('A', fontsize=10)
1347            plt.xlabel(I_Label_X, fontsize=10)
1348            ax = plt.gca()
1349            for axis in ['top', 'bottom', 'left', 'right']:
1350                ax.spines[axis].set_linewidth(2)
1351            for tick in ax.xaxis.get_major_ticks():
1352                tick.label.set_fontsize(10)
1353            for i in range(0, len(fig_ext)):
1354                plt.savefig(dir_destino+"/figuras/"+caso+"/"+fig_ext[i]+"/"\
```

```
            +caso+"_Fase_Aquatica-comp-ing."+fig_ext[i])

        plt.legend((P_Label_Opt, P_Label_Sim))
        plt.title(P_Frase_A, fontsize=10)
        plt.xlabel(P_Label_X, fontsize=10)
        for i in range(0, len(fig_ext)):
            plt.savefig(dir_destino+"/figuras/"+caso+"/"+fig_ext[i]+"/"\
            +caso+"_Fase_Aquatica-comp."+fig_ext[i])

        plt.close()

        fig = fig+1
        plt.figure(fig)  # fig 7
        plt.hold(True)
        plt.subplot(2, 1, 1)
        plt.plot(t_opt, F_opt, '--', linewidth=2.0)
        plt.plot(t_sim, F_sim, linewidth=2.0)
        plt.legend((I_Label_Opt, I_Label_Sim))
        if grade > 0:
            plt.grid()
        plt.title(I_Frase_F)
        plt.ylabel('F')
        ax = plt.gca()
        for axis in ['top', 'bottom', 'left', 'right']:
            ax.spines[axis].set_linewidth(2)
        for tick in ax.xaxis.get_major_ticks():
            tick.label.set_fontsize(10)

        plt.subplot(2, 1, 2)
        plt.plot(t_opt, I_opt, '--', linewidth=2.0)
        plt.plot(t_sim, I_sim, linewidth=2.0)
        plt.legend((I_Label_Opt, I_Label_Sim))
        if grade > 0:
            plt.grid()
        plt.title(I_Frase_I, fontsize=10)
        plt.ylabel('I', fontsize=10)
        plt.xlabel(I_Label_X, fontsize=10)

        ax = plt.gca()
        for axis in ['top', 'bottom', 'left', 'right']:
            ax.spines[axis].set_linewidth(2)
        for tick in ax.xaxis.get_major_ticks():
            tick.label.set_fontsize(10)
        for i in range(0, len(fig_ext)):
            plt.savefig(dir_destino+"/figuras/"+caso+"/"+fig_ext[i]+"/"\
            +caso+"_Femeas-comp-ing."+fig_ext[i])

        plt.subplot(2, 1, 1)
        plt.legend((P_Label_Opt, P_Label_Sim))
        plt.title(P_Frase_F)
        plt.subplot(2, 1, 2)
        plt.legend((P_Label_Opt, P_Label_Sim))
        plt.title(P_Frase_I, fontsize=10)
        plt.xlabel(P_Label_X, fontsize=10)
        for i in range(0, len(fig_ext)):
            plt.savefig(dir_destino+"/figuras/"+caso+"/"+fig_ext[i]+"/"\
            +caso+"_Femeas-comp."+fig_ext[i])

```

```
1413            plt.close()
1414
1415            fig = fig+1
1416            plt.figure(fig)  # fig 7
1417            plt.hold(True)
1418            plt.subplot(2, 1, 1)
1419            plt.plot(t_opt, M_opt, '--', linewidth=2.0)
1420            plt.plot(t_sim, M_sim, linewidth=2.0)
1421            plt.legend((I_Label_Opt, I_Label_Sim))
1422            if grade > 0:
1423                plt.grid()
1424            plt.title(I_Frase_M, fontsize=10)
1425            plt.ylabel('M', fontsize=10)
1426            ax = plt.gca()
1427            for axis in ['top', 'bottom', 'left', 'right']:
1428                ax.spines[axis].set_linewidth(2)
1429            for tick in ax.xaxis.get_major_ticks():
1430                tick.label.set_fontsize(10)
1431
1432            plt.subplot(2, 1, 2)
1433            plt.plot(t_opt, S_opt, '--', linewidth=2.0)
1434            plt.plot(t_sim, S_sim, linewidth=2.0)
1435            plt.legend((I_Label_Opt, I_Label_Sim))
1436            if grade > 0:
1437                plt.grid()
1438            plt.title(I_Frase_S, fontsize=10)
1439            plt.ylabel('S', fontsize=10)
1440
1441            plt.xlabel(I_Label_X, fontsize=10)
1442
1443            ax = plt.gca()
1444            for axis in ['top', 'bottom', 'left', 'right']:
1445                ax.spines[axis].set_linewidth(2)
1446            for tick in ax.xaxis.get_major_ticks():
1447                tick.label.set_fontsize(10)
1448            for i in range(0, len(fig_ext)):
1449                plt.savefig(dir_destino+"/figuras/"+caso+"/"+fig_ext[i]+"/"\
1450                +caso+"_Machos-comp-ing."+fig_ext[i])
1451
1452            plt.subplot(2, 1, 1)
1453            plt.legend((P_Label_Opt, P_Label_Sim))
1454            plt.title(P_Frase_M, fontsize=10)
1455            plt.subplot(2, 1, 2)
1456            plt.legend((P_Label_Opt, P_Label_Sim))
1457            plt.title(P_Frase_S, fontsize=10)
1458            plt.xlabel(P_Label_X, fontsize=10)
1459            for i in range(0, len(fig_ext)):
1460                plt.savefig(dir_destino+"/figuras/"+caso+"/"+fig_ext[i]+"/"\
1461                +caso+"_Machos-comp."+fig_ext[i])
1462
1463            plt.close()
1464           # plt.show();
1465            return fig
1466
1467 def compila_modelo(model_mane_str):
1468
1469     comp_options = {"generate_html_diagnostics":True}
1470     compiler_log_level_ = 'w'
```

```
1471     pack_name = model_mane_str+"Pack"
1472     class_name = pack_name+"."+model_mane_str
1473     file_name = os.environ['USERCODEDIR']+"/modelos/"+pack_name+".mop"
1474     compile_to_ = dir_path
1475 
1476     nome_modelo = compile_jmu(class_name
1477                               , file_name
1478                               , compiler='auto'
1479                               #, target = 'me'
1480                               #, version = '2.0'
1481                               , compiler_options=comp_options
1482                               #, compile_to = compile_to_+class_name+".jmu"
1483                               , compiler_log_level=compiler_log_level_
1484                               , separate_process='True'
1485                               )
1486 
1487     print "passou compilação modelo básico"
1488     print "nome_modelo = ", nome_modelo
1489     model = JMUModel(nome_modelo)
1490 
1491     print "carregou modelo básico"
1492 
1493     class_name = pack_name+"."+model_mane_str+"_Opt"
1494     nome_opt = compile_jmu(class_name
1495                     , file_name
1496                     , compiler='auto'
1497                      #, target = 'me'
1498                      #, version = '2.0'
1499                     , compiler_options=comp_options
1500                     #, compile_to = compile_to_+class_name+".jmu"
1501                     , compiler_log_level=compiler_log_level_
1502                     , separate_process='True'
1503                     )
1504 
1505     print "passou compilação Opt"
1506     model_opt = JMUModel(nome_opt)
1507     print "carregou modelo Opt"
1508     os.system('cp '+ file_name +' '+compile_to_+"modelos")
1509     os.system('mv *html_diagnostics '+compile_to_+"html")
1510     os.system('mv *.jmu '+compile_to_+"modelos")
1511 
1512     return model, model_opt
1513 
1514 def carrega_modelos(model_mane_str, model_opt_mane_str):
1515 
1516     pack_name = model_mane_str+"Pack"
1517     dir_path = os.environ['USERCODEDIR']+"/resultados/jm/dengue/modelos/"
1518     class_name = pack_name+"_"+model_mane_str
1519     name = dir_path+class_name+".jmu"
1520     model = JMUModel(name)
1521     print  "Carregou modelo", name
1522 
1523     class_name_opt = pack_name+"_"+model_opt_mane_str
1524     name_opt = dir_path+class_name_opt+".jmu"
1525     model_opt = JMUModel(name_opt)
1526     print  "Carregou modelo", name
1527     print  "Carregou modelo", name_opt
1528 
```

```
1529     return model, model_opt
1530
1531 def processa(k, d, compilar, sim):
1532     global caso
1533     global ci
1534     global f_1
1535     global inicio_otim, inicio_proc, res_estat, tempo_ultima_otim
1536
1537     model_param, model_opt_param, param = aplic_param(k, d, sim)
1538     s_time = model_opt_param['s_time']
1539     f_time = model_opt_param['f_time']
1540     dias = d
1541     Lc = param['Lc']
1542     LF = param['LF']
1543     Ls = param['Ls']
1544     Lu = param['Lu']
1545     n_e = param['n_e']
1546     n_cp = param['n_cp']
1547     dir_destino = param['dir_destino']
1548     arquivo_path = param['arquivo_path']
1549     fig_ext = param['fig_ext']
1550     tol = param['tol']
1551     acceptable_tol = param['acceptable_tol']
1552     acceptable_iter = param['acceptable_iter']
1553
1554     if os.path.isfile(arquivo_path):
1555         os.remove(arquivo_path)
1556
1557     f = open(arquivo_path, 'w+')
1558
1559     inicio_proc = datetime.now()
1560     print "Início do processamento:", str(inicio_proc)
1561     print "Arquivo de log = ", arquivo_path
1562
1563     print >> f, "Início do processamento:", str(inicio_proc)
1564     print >> f, "Arquivo de log = ", arquivo_path
1565
1566     if compilar:
1567         model, model_opt = compila_modelo("Dengue")
1568     else:
1569         model, model_opt = carrega_modelos("Dengue", "Dengue_Opt")
1570         print "Não compilou os modelos"
1571         print >> f, "Não compilou os modelos"
1572
1573     cont = 0
1574
1575     fig = 1
1576     I_Frases, P_Frases = language()
1577
1578     for i in range(0, len(Ls)):
1579         #
     ****************************************************************
1580         # Início da otimização
1581         # Valores realmente utilizados sao definidos na rotina
     def_parametros()
1582         cont = cont+1
1583         fig = fig+1
1584         solver = Ls[i]
```

```
1585        lista_opt = def_parametros(f, solver, n_e, n_cp, s_time, f_time,
    tol, \
1586        acceptable_tol, acceptable_iter )
1587
1588        for i in range(0, len(Lc)):
1589            ci = Lc[i]
1590            ui = Lu[i]
1591            LFI = LF[i]
1592            t_time = LFI[0]
1593            F_time = LFI[1]
1594
1595            caso = solver+"_caso_"+str(i+1)
1596
1597            os.system('mkdir '+dir_destino+"/figuras/"+caso)
1598
1599
1600            for i in range(0, len(fig_ext)):
1601                os.system('mkdir '+dir_destino+"/figuras/"+caso+"/"+\
1602                                    fig_ext[i])
1603
1604            param = dict()
1605            param['caso'] = caso
1606            param['I_Frases'] = I_Frases
1607            param['P_Frases'] = P_Frases
1608            param['dir_destino'] = dir_destino
1609            param['fig_ext'] = fig_ext
1610
1611            inicio_otim = datetime.now()
1612            print " "
1613            print "
    ************************************************************\
1614 ***********"
1615            print "Início da otimização do "+caso+" com solucionador linear
     \
1616 ", solver, " = ", str(inicio_otim)
1617            print "ci = ", ci
1618            print "u1 pertence ao intervalo = ", ui[0]
1619            print "u2 pertence ao intervalo = ", ui[1]
1620
1621            print >> f, " "
1622            print >> f, "
    ************************************************\
1623 *****************"
1624            print >> f, "Início da otimização do "+caso+" com solucionador
    \
1625 linear ", solver, " = ", str(inicio_otim)
1626            print >> f, "ci = ", ci
1627            print >> f, "u1 pertence ao intervalo = ", ui[0]
1628            print >> f, "u2 pertence ao intervalo = ", ui[1]
1629
1630            if t_time>0 and t_time<120.00:
1631                print "Restrição sobre F:"
1632                print "t_time = ", t_time
1633                print "F(", t_time, ") = ", F_time
1634                print  >> f, "Restrição sobre F:"
1635                print  >> f, "t_time = ", t_time
1636                print  >> f, "F(", t_time, ") = ", F_time
1637            else:
```

```
1638                print "Este caso não tem restrição sobre F"
1639                print  >> f, "Este caso não tem restrição sobre F"
1640            otimizou = False
1641
1642            try:
1643                print "Aguarde a saída do IPOPT:"
1644                f.flush()
1645                sys.stdout.flush()
1646                res_objeto, opt_opts, model_opt = otimiza(f,
1647                                                          model_opt,
1648                                                          fig,
1649                                                          model_param,
1650                                                          model_opt_param,
1651                                                          lista_opt,
1652                                                          k,
1653                                                          dias,
1654                                                          ci,
1655                                                          ui,
1656                                                          t_time,
1657                                                          F_time,
1658                                                          param)
1659                [return_status,
1660                 nbr_iter,
1661                 objetive,
1662                 total_exec_time] = res_objeto.solver.
    opt_coll_ipopt_get_statistics()
1663
1664                res_estat = {}
1665                res_estat['return_status'] = return_status
1666                res_estat['nbr_iter'] = nbr_iter
1667                res_estat['objetive'] = objetive
1668                res_estat['total_exec_time'] = total_exec_time
1669                res_estat['d'] = dias
1670                res_estat['k'] = k
1671                res_estat['ci'] = ci
1672                res_estat['u1_intervalo'] = ui[0]
1673                res_estat['u2_intervalo'] = ui[1]
1674                res_estat['caso'] = caso
1675
1676                arq_opt_mat = "mat/"+caso+"_res_obj_opt.mat"
1677                sio.savemat(dir_destino+arq_opt_mat, dict(res_objeto))
1678                otimizou = True
1679
1680                arq_opt_mat = "mat/"+caso+"_res_est_opt.mat"
1681                sio.savemat(dir_destino+arq_opt_mat, dict(res_estat))
1682
1683                arq_opt_mat = "mat/"+caso+"_model_otim_param.mat"
1684                sio.savemat(dir_destino+arq_opt_mat, dict(lista_opt))
1685
1686            except Exception as e:
1687                print "Não foi possível Otimizar:"
1688                print "k = "+str(k)
1689                print "d = "+str(d)
1690                print "Recebeu exception no "+caso
1691                print e.message
1692
1693                print >> f_1, "Não foi possível Otimizar:"
1694                print >> f_1, "k = "+str(k)
```

```
1695                print >> f_1, "d = "+str(d)
1696                print >> f_1, "Recebeu exception no "+caso
1697                print >> f_1, e.message
1698                #break # pra degugar tem que deixar o break
1699                # - depois, pra rodar, tirar o break
1700
1701            if otimizou == True:
1702                try:
1703                    comp = True
1704                    simul = True
1705                    opt = True
1706
1707                    n_e = lista_opt['n_e']
1708                    n_cp = lista_opt['n_cp']
1709                    fig = graficos(f,
1710                                   res_objeto,
1711                                   fig,
1712                                   model,
1713                                   model_opt,
1714                                   model_param,
1715                                   model_opt_param,
1716                                   n_e,
1717                                   n_cp,
1718                                   k,
1719                                   dias,
1720                                   comp,
1721                                   simul,
1722                                   opt,
1723                                   param)
1724
1725                except Exception as e:
1726                    print "Não foi possível gerar gráficos:"
1727                    print "k = "+str(k)
1728                    print "d = "+str(d)
1729                    print "Recebeu exception no "+caso
1730                    print e.message
1731
1732                    print >> f_1, "Não foi possível gerar gráficos:"
1733                    print >> f_1, "k = "+str(k)
1734                    print >> f_1, "d = "+str(d)
1735                    print >> f_1, "Recebeu exception no "+caso
1736                    print >> f_1, e.message
1737                    #break # pra degugar tem que deixar o break
1738                    # - depois, pra rodar, tirar o break
1739
1740            data_opt = datetime.now()
1741            delta = data_opt - inicio_otim
1742
1743            tempo_ultima_otim = delta.total_seconds()
1744            print  "Fim da otimização do "+caso+" com solucionador linear \
1745", solver, " = ", str(data_opt)
1746            print  "Tempo de otimização em segundos = \
1747", tempo_ultima_otim
1748            print  " "
1749            print   "**************************************************\
1750*****************"
1751            print  " "
1752            print >> f, "Fim da otimização do "+caso+" com solucionador \
```

```
1753 linear ", solver, " = ", str(data_opt)
1754             print >> f, "Tempo de otimização em segundos= \
1755 ", tempo_ultima_otim
1756             print >> f, " "
1757             print >> f, "**********************************************\
1758 ********************"
1759             print >> f, " "
1760
1761     data_final = datetime.now()
1762     delta = data_final-inicio_proc
1763
1764     print  "Tempo de processamento = ", str(delta)
1765     print  "Segundos = ", delta.total_seconds()
1766
1767     print >> f, "Tempo de processamento = ", str(delta)
1768     print >> f, "Segundos = ", delta.total_seconds()
1769     f.close()
1770
1771     # Fim da rotina  processa
1772
1773 #
    ********************************************************************************

1774 #Início do programa principal
1775
1776 print "Otimiza Copyright (C) 2015, Ranulfo Acir de Oliveira Resende"
1777 print "This program comes with ABSOLUTELY NO WARRANTY."
1778 print "This is free software, and you are welcome to redistribute it"
1779 print "under certain conditions."
1780 print "Read README.txt for details."
1781 print "Sugestions and corrections to improve the software are welcome:"
1782 print "acir_r@yahoo.com.br"
1783
1784 inicio_progama = datetime.now()
1785 inicio_otim = inicio_progama
1786 inicio_proc = inicio_progama
1787 grade = 1
1788 dir_path = os.environ['USERCODEDIR']+'/resultados/jm/dengue/'
1789 os.system('mkdir '+dir_path)
1790 os.system('mkdir '+dir_path+"html")
1791 os.system('mkdir '+dir_path+"modelos")
1792 res_estat = dict()
1793 tempo_ultima_otim = float()
1794 arquivo_1_path=dir_path+"Log_Geral.txt"
1795
1796 if os.path.isfile(arquivo_1_path):
1797         os.remove(arquivo_1_path)
1798 f_1 = open(arquivo_1_path, 'w+')
1799
1800 print  "Início do programa = ", str(inicio_progama)
1801 print >> f_1, "Início do programa = ", str(inicio_progama)
1802
1803 caso="Início"
1804 ci=[0,0,0,0]
1805
1806 sim = 1  # ulim=definidos nos casos
1807 print  "Início dos casos da Simulação "+str(sim)      #Artigo DINCON
1808 print >> f_1, "Início dos casos da Simulação "+str(sim) #Artigo DINCON
```

```
k_seq = 1,100
d_seq = 0,10,120

compilar = True
for d in d_seq:
    for k in k_seq:
        #processa(k, d, compilar, sim)
        compilar = False

sim = 2
print  "Início dos casos da Simulação "+str(sim)
print >> f_1, "Início dos casos da Simulação "+str(sim)
k_seq = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 25, 30, 35, 40, 45, 50, 60,
    70,\
80, 90, 91, 92, 93, 94, 94.1, 94.2, 94.240, 94.2410, 94.2420, 94.2430,
    94.24365,\
94.2440, 94.2450, 94.250, 94.3, 94.4, 94.5, 94.6, 94.7, 94.8, 94.9, 95, 96,
     97,\
98, 99, 100, 110, 120, 130, 140, 150, 200, 250, 300, 350, 400, 450, 500
d_seq = [0]


sim_seq = [2,]

for d in d_seq:
    for k in k_seq:
        #processa(k, d, compilar, sim)
        compilar = False

compilar = True
d_seq = [0]
k_seq = 1,100
sim = 6
for d in d_seq:
    for k in k_seq:
        #processa(k, d, compilar, sim)
        compilar = False

sim = 7
print  "Início dos casos da Simulação "+str(sim)
print >> f_1, "Início dos casos da Simulação "+str(sim)
k_seq = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 25, 30, 35, 40, 45, 50, 60,
    70,\
80, 90, 91, 92, 93, 94, 94.1, 94.2, 94.240, 94.2410, 94.2420, 94.2430,
    94.24365,\
94.2440, 94.2450, 94.250, 94.3, 94.4, 94.5, 94.6, 94.7, 94.8, 94.9, 95, 96,
     97,\
98, 99, 100, 110, 120, 130, 140, 150, 200, 250, 300, 350, 400, 450, 500

k_seq = 94.240005, 94.24002, 94.24005, 94.24010, 94.24025, 94.24040,
    94.24035, 94.24045, 94.24050

d_seq = [0]
compilar = True

for d in d_seq:
    for k in k_seq:
        processa(k, d, compilar, sim)
```

```
1860         compilar = False
1861
1862 sim_seq = [2, 5, 6, 7, 8]
1863 sim_seq = [5, 6, 7, 8]
1864 # SIM                2     5     6     7     8
1865 # n_e  =           240   480   120   120   240
1866 # n_cp =             5    10     5     5    10
1867 # tol  =          1e-8  1e-9  1e-8  1e-6  1e-8
1868 # acceptable_tol = 1e-6  1e-7  1e-6  1e-4  1e-6
1869 # acceptable_iter =  3     5     5     5    10
1870 # u2_max          =  10    10    12    10    10
1871 #Acir
1872 k_seq = 94.2400, 94.2405, 94.2410, 94.2415,94.2420, 94.2425, 94.2430,
        94.2433, \
1873 94.2435, 94.2438, 94.2440, 94.2445, 94.2450
1874 d_seq = [0]
1875 for sim in sim_seq:
1876     for d in d_seq:
1877         for k in k_seq:
1878             #processa(k, d, compilar, sim)
1879             compilar = False
1880
1881 data_fim = datetime.now()
1882 print  "Fim do programa = ", str(data_fim)
1883 print >> f_1,   "Fim do programa = ", str(data_fim)
1884 tempo = data_fim-inicio_progama
1885 print  "Tempo de processamento total= ", str(tempo)
1886 print  >> f_1,  "Tempo de processamento total= ", str(tempo)
1887
1888 f_1.close()
1889
1890 print "Otimiza Copyright (C) 2015, Ranulfo Acir de Oliveira Resende"
1891 print "This program comes with ABSOLUTELY NO WARRANTY."
1892 print "This is free software, and you are welcome to redistribute it"
1893 print "under certain conditions."
1894 print "Read README.txt for details."
1895 print "Sugestions and corrections to improve the software are welcome:"
1896 print "acir_r@yahoo.com.br"
```

O arquivo *analisa.py*, *script* Python para analisar os resultados, é apresentado a seguir.

A.2.3 O script analisa.py

```
1 #!/usr/bin/python
2 # coding=UTF-8
3
4 # Copyright © 2015, Ranulfo Acir de Oliveira Resende.
5 #
6 # This program is free software: you can redistribute it and/or modify
7 # it under the terms of the GNU General Public License as published by
8 # the Free Software Foundation, either version 3 of the License, or
9 # (at your option) any later version.
10
11 # This program is distributed in the hope that it will be useful,
12 # but WITHOUT ANY WARRANTY; without even the implied warranty of
13 # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
14 # GNU General Public License for more details.
15
```

```
16 # You should have received a copy of the GNU General Public License
17 # along with this program.  If not, see <http://www.gnu.org/licenses/>.
18
19
20 # Ranulfo Acir de Oliveira Resende (acir_r@yahoo.com.br)
21
22 # Tradução não oficial:
23 # Este programa é um software livre; você pode redistribuí-lo e/ou
24 # modificá-lo dentro dos termos da Licença Pública Geral GNU como
25 # publicada pela Fundação do Software Livre (FSF); na versão 3 da
26 # Licença, ou (na sua opinião) qualquer versão.
27 #
28 # Este programa é distribuído na esperança de que possa ser útil,
29 # mas SEM NENHUMA GARANTIA; sem uma garantia implícita de ADEQUAÇÃO
30 # a qualquer MERCADO ou APLICAÇÃO EM PARTICULAR. Veja a
31 # Licença Pública Geral GNU para maiores detalhes.
32
33 # Você deve ter recebido uma cópia da Licença Pública Geral GNU junto
34 # com este programa. Se não, veja <http://www.gnu.org/licenses/>.
35 #
36 # Esta tradução é informal, e não aprovada oficialmente pela
37 # Free Software Foundation como válida. Para ter certeza do que é
38 # permitido, refira-se à GPL original (em inglês).
39
40 import random
41 import matplotlib
42 import matplotlib.pyplot as plt
43 import numpy as N
44 from pymodelica import compile_jmu
45 from pyjmi.jmi import JMUModel
46 from datetime import datetime
47 matplotlib.interactive(False)  # para não abrir figura na tela
48 import collections
49 import sys, os
50 import scipy.io as sio
51 import mat4py
52 import math
53 sys.path.append(os.environ['PREFIX']+"/Python")
54 sys.path.append(os.environ['PREFIX']+"/lib/python2.7/dist-packages")
55
56 def my_range(start, end, step):
57     while start <= end:
58         yield start
59         start += step
60
61 def lim_zero(num):
62     if num < 0:
63         num = 0
64     return num
65
66 def integral(f, t, t0, tf):
67     integra = float()
68     if t0 == tf:
69         integra = 0
70     elif t0 < tf:
71         integra = 0
72         for i in range(t0, tf-1):
```

```
73              deltat = t[i+1] - t[i]
74              deltaf = (f[i+1] + f[i])/2
75              integra = integra + deltat*deltaf
76      elif tf < t0:
77          integra = 0
78          for i in range(t0, tf+1):
79              deltat = t[i] - t[i+1]
80              deltaf = (f[i+1] + f[i])/2
81              integra = integra - deltat*deltaf
82      return integra
83
84  def derivada(f, t):
85      tam_t = len(t)
86      deriva = N.zeros(tam_t-1, float)
87      for i in range(0, tam_t-1):
88          deltat = t[i+1] - t[i]
89          deriva[i] = (f[i+1] - f[i])/deltat
90
91      return deriva
92
93  def aplic_param_analise(d, sim):
94      global dir_path
95
96      # model parameters
97
98      model_param = dict()
99
100     model_param['phi'] = 0.50
101     model_param['C'] = 13
102     model_param['gama'] = 0.07
103     model_param['r'] = 0.50
104     model_param['beta'] = 1.0
105     model_param['beta_S'] = 0.70
106     model_param['mu_A'] = 0.05
107     model_param['mu_I'] = 0.05
108     model_param['mu_F'] = 0.05
109     model_param['mu_M'] = 0.1
110     model_param['mu_S'] = 0.1
111
112
113     # model_optimization parameters
114     model_opt_param = dict()
115     model_opt_param['c_1'] = 1
116     model_opt_param['c_2'] = 1
117     model_opt_param['c_3'] = 1
118     model_opt_param['c_4'] = 1
119 #     model_opt_param['k']= k
120
121
122     # model_optimization parameters
123     # t_time = f_time significa valor final
124     model_opt_param['s_time'] = 0.0 #tempo dias
125     model_opt_param['f_time'] = 120.0
126
127     model_opt_param['u1_min'] = 0.0
128     model_opt_param['u2_min'] = 0.0
129
130     # Definir estes valores juntos
```

```
131     model_opt_param['u1_max'] = 10.0
132     model_opt_param['u2_max'] = 10.0
133     u1_max = model_opt_param['u1_max']
134     u2_max = model_opt_param['u2_max']
135 
136     # Definir estes valores juntos
137     model_opt_param['t_time'] = 0        # t_time = 0 não tem efeito, vale
    o default do modelo
138     model_opt_param['F_time'] = 0.1
139     t_time = model_opt_param['t_time']
140     F_time = model_opt_param['F_time']
141 
142     dias = d    # Dias (periodo) de combate a dengue
143                 # (entradas de controle piecewise)
144                 # Deve ser fator de horizonte de tempo em dias (f_time-
    s_time)
145                 # só tem significado se dias > 0
146 
147     # definiçoes da simulação: de Lu, Lc e LF
148     if sim == 1:
149         simula = "SIM"+str(sim)+"/"
150         n_e = 240 #240  n_e*n_cp multiplo de 120
151         n_cp = 5  #5
152         tol = 1e-8
153         acceptable_tol = 1e-6
154         acceptable_iter = 3
155 
156         LuCtrFixo = [[[0, 0.1], [0, 0.02]],
157                      [[0, 0.08], [0, 0.01]],
158                      [[0, 0.07], [0, 0.05]],
159                      [[0, 0.04], [0, 0.08]]]
160 
161         LuCtrCont = [[[0, u1_max], [0, u2_max]],
162                      [[0, u1_max], [0, u2_max]],
163                      [[0, u1_max], [0, u2_max]],
164                      [[0, u1_max], [0, u2_max]]]
165 
166         LuCtrDisc = [[[0, u1_max], [0, u2_max]],
167                      [[0, u1_max], [0, u2_max]],
168                      [[0, u1_max], [0, u2_max]],
169                      [[0, u1_max], [0, u2_max]]]
170 
171 
172         LF = [[0, 0.1],  # LF = [t_time, F_time]
173               [0, 0.1],  # Tem que ter muito cuidado aqui
174               [0, 0.1],  # Porque restricoes impossíveis
175               [0, 0.1]]  # de alcancar tornam o problema
176                          # insolúvel ao IPOPT e trava
177            #  [60, 0.1]# o script python dando erro
178                        # Se for t_time = 0 nao tem efeito na otimização
179                        # Se for t_time = 120 representa restrição final
    para F
180 
181         Lc = [[1, 10, 100, 1],  # ci para os casos 1 a 4
182               [1, 10, 1, 1],
183               [1, 1, 1, 1],
184               [3, 1, 1, 1]]
185         if dias < 5:
```

```
186             Lu = LuCtrCont
187         elif dias > 100:
188             Lu = LuCtrFixo
189         else:
190             Lu = LuCtrDisc
191 
192     elif sim == 2:
193         simula = "SIM"+str(sim)+"/"
194         n_e = 240 #240  n_e*n_cp multiplo de 120
195         n_cp = 5  #5
196         tol = 1e-8
197         acceptable_tol = 1e-6
198         acceptable_iter = 3
199         Lc = [[1, 1, 1, 1]]
200         Lu = [[[0, u1_max], [0, u2_max]]]
201         LF = [[0, 0.1]]
202 
203     elif sim == 3:
204         simula = "SIM"+str(sim)+"/"
205         n_e = 240 #240  n_e*n_cp multiplo de 120
206         n_cp = 5  #5
207         tol = 1e-8
208         acceptable_tol = 1e-6
209         acceptable_iter = 3
210         LuCtrFixo = [[[0, 0.1], [0, 0.02]],
211                      [[0, 0.08], [0, 0.01]],
212                      [[0, 0.07], [0, 0.05]],
213                      [[0, 0.04], [0, 0.08]]]
214 
215         LuCtrCont = [[[0, u1_max], [0, u2_max]],
216                      [[0, u1_max], [0, u2_max]],
217                      [[0, u1_max], [0, u2_max]],
218                      [[0, u1_max], [0, u2_max]]]
219 
220         LuCtrDisc = [[[0, u1_max], [0, u2_max]],
221                      [[0, u1_max], [0, u2_max]],
222                      [[0, u1_max], [0, u2_max]],
223                      [[0, u1_max], [0, u2_max]]]
224 
225 
226         LF = [[0, 0.1],  # LF = [t_time, F_time]
227               [0, 0.1],  # Tem que ter muito cuidado aqui
228               [0, 0.1],  # Porque restricoes impossíveis
229               [0, 0.1]]  # de alcancar tornam o problema
230                          # insolúvel ao IPOPT e trava
231            #  [60, 0.1]# o script python dando erro
232                        # Se for t_time = 0 nao tem efeito na otimização
233                        # Se for t_time = 120 representa restrição final
    para F
234 
235         Lc = [[1, 10, 100, 1],
236               [1, 10, 1, 1],
237               [1, 1, 1, 1],
238               [3, 1, 1, 1]]
239         if dias < 5:
240             Lu = LuCtrCont
241         elif dias > 100:
242             Lu = LuCtrFixo
```

```
243         else:
244             Lu = LuCtrDisc
245 
246     elif sim == 4:
247         simula = "SIM"+str(sim)+"/"
248         n_e = 240 #240  n_e*n_cp multiplo de 120
249         n_cp = 5  #5
250         tol = 1e-8
251         acceptable_tol = 1e-6
252         acceptable_iter = 3
253         LuCtrFixo = [[[0, 0.1], [0, 0.02]],
254                      [[0, 0.08], [0, 0.01]],
255                      [[0, 0.07], [0, 0.05]],
256                      [[0, 0.04], [0, 0.08]]]
257 
258         LuCtrCont = [[[0, u1_max], [0, u2_max]],
259                      [[0, u1_max], [0, u2_max]],
260                      [[0, u1_max], [0, u2_max]],
261                      [[0, u1_max], [0, u2_max]]]
262 
263         LuCtrDisc = [[[0, u1_max], [0, u2_max]],
264                      [[0, u1_max], [0, u2_max]],
265                      [[0, u1_max], [0, u2_max]],
266                      [[0, u1_max], [0, u2_max]]]
267 
268 
269         LF = [[0, 0.1],  # LF = [t_time, F_time]
270               [0, 0.1],  # Tem que ter muito cuidado aqui
271               [0, 0.1],  # Porque restricoes impossíveis
272               [0, 0.1]]  # de alcancar tornam o problema
273                          # insolúvel ao IPOPT e trava
274            #  [60, 0.1]# o script python dando erro
275                        # Se for t_time = 0 nao tem efeito na otimização
276                        # Se for t_time = 120 representa restrição final
    para F
277 
278         Lc = [[1, 10, 100, 1],
279               [1, 10, 1, 1],
280               [1, 1, 1, 1],
281               [3, 1, 1, 1]]
282         if dias < 5:
283             Lu = LuCtrCont
284         elif dias > 100:
285             Lu = LuCtrFixo
286         else:
287             Lu = LuCtrDisc
288 
289 
290     elif sim == 5:
291         simula = "SIM"+str(sim)+"/"
292         n_e = 480 #240  n_e*n_cp multiplo de 120
293         n_cp = 10  #5
294         tol = 1e-9
295         acceptable_tol = 1e-7
296         acceptable_iter = 3
297         Lc = [[1, 1, 1, 1]]
298         Lu = [[[0, u1_max], [0, u2_max]]]
299         LF = [[0, 0.1]]
```

```
300
301     elif sim == 6:
302         simula = "SIM"+str(sim)+"/"
303         n_e = 120 #240  n_e*n_cp multiplo de 120
304         n_cp = 5  #5
305         tol = 1e-8
306         acceptable_tol = 1e-6
307         acceptable_iter = 3
308         # Definir estes valores juntos
309         model_opt_param['u1_max'] = 12.0
310         model_opt_param['u2_max'] = 12.0
311         u1_max = model_opt_param['u1_max']
312         u2_max = model_opt_param['u2_max']
313         model_param['mu_S'] = 0.2
314         Lc = [[1, 1, 1, 1]]
315         Lu = [[[0, u1_max], [0, u2_max]]]
316         LF = [[0, 0.1]]
317
318     elif sim == 7:
319         simula = "SIM"+str(sim)+"/"
320         n_e = 120 #240  n_e*n_cp multiplo de 120
321         n_cp = 5  #5
322         tol = 1e-6
323         acceptable_tol = 1e-4
324         acceptable_iter = 3
325         # Definir estes valores juntos
326         model_opt_param['u1_max'] = 10.0
327         model_opt_param['u2_max'] = 10.0
328         u1_max = model_opt_param['u1_max']
329         u2_max = model_opt_param['u2_max']
330         model_param['mu_S'] = 0.1
331         Lc = [[1, 1, 1, 1]]
332         Lu = [[[0, u1_max], [0, u2_max]]]
333         LF = [[0, 0.1]]
334
335     elif sim == 8:
336         simula = "SIM"+str(sim)+"/"
337         n_e = 240 #240  n_e*n_cp multiplo de 120
338         n_cp = 10  #5
339         tol = 1e-8
340         acceptable_tol = 1e-6
341         acceptable_iter = 10
342         # Definir estes valores juntos
343         model_opt_param['u1_max'] = 10.0
344         model_opt_param['u2_max'] = 10.0
345         u1_max = model_opt_param['u1_max']
346         u2_max = model_opt_param['u2_max']
347         model_param['mu_S'] = 0.1
348         Lc = [[1, 1, 1, 1]]
349         Lu = [[[0, u1_max], [0, u2_max]]]
350         LF = [[0, 0.1]]
351
352     dir_path0 = dir_path+simula
353     dir_destino = dir_path0
354     file_log = "log"
355     arquivo_nome = file_log+'.txt'
356     arquivo_path = dir_destino+arquivo_nome
357
```

```
358     #L = ['ma27', 'ma57', 'ma86']
359     Ls = ['ma57']
360     #possiveis resolvedores lineares:
361             # rotinas HSL: 'ma27', 'ma57', 'ma77', 'ma86' e 'ma97'
362             # default 'mumps'
363             # se configurar as libs: 'pardiso', 'wsmp' e custom
364 
365     fig_ext = ["eps", "jpg", "png"]
366 
367     param = dict()
368     param['Lc'] = Lc
369     param['LF'] = LF
370     param['Ls'] = Ls
371     param['Lu'] = Lu
372     param['n_e'] = n_e
373     param['n_cp'] = n_cp
374     param['dir_destino'] = dir_destino
375     param['arquivo_path'] = arquivo_path
376     param['fig_ext'] = fig_ext
377     param['tol'] = tol
378     param['acceptable_tol'] = acceptable_tol
379     param['acceptable_iter'] = acceptable_iter
380 
381     return model_param, model_opt_param, param
382 
383 def analise_param(k, d, sim): # Rotina necessária para carregar arquivos .
        mat
384     dir_path = os.environ['USERCODEDIR']+'/resultados/jm/dengue/'
385 
386     simula = "SIM"+str(sim)+"/"
387 
388     dir_path0 = dir_path+simula
389     dir_path1 = dir_path0+"dias_"+str(d)+"/"
390     dir_origem = dir_path1+"k_"+str(k)+"/"
391     dir_destino=dir_path+"/analise/"+simula
392 
393     fig_ext = ["eps", "jpg", "png"]
394 
395     param = [dir_origem,
396              dir_destino,
397              fig_ext]
398 
399     return param
400 
401 def language():
402 
403     P_Frase_A = u"Fase Aquática"    # a letra u antes da frase permite
404     P_Frase_I = u"Fêmeas Imaturas"  # reconhecimento de caracteres
        português
405     P_Frase_F = u"Fêmeas Fertilizadas"
406     P_Frase_M = "Machos Naturais"
407     P_Frase_S = u"Machos Estéreis"
408     P_Label_Opt = "Otimizado"
409     P_Label_Sim = "Simulado"
410     P_Label_X = "t[dias]"
411     P_Controle = "Sinal de Controle"
412 
413     I_Frase_A = "Aquatic Phase"
```

```
414     I_Frase_I = "Imature Famales"
415     I_Frase_F = "Fertilized Females"
416     I_Frase_M = "Natural Males"
417     I_Frase_S = "Sterile Males"
418     I_Label_Opt = "Optmized"
419     I_Label_Sim = "Simulated"
420     I_Label_X = "t[days]"
421     I_Controle = "Control Signal"
422 
423     I_frases = [I_Frase_A,
424                 I_Frase_I,
425                 I_Frase_F,
426                 I_Frase_M,
427                 I_Frase_S,
428                 I_Label_Opt,
429                 I_Label_Sim,
430                 I_Label_X,
431                 I_Controle]
432     P_frases = [P_Frase_A,
433                 P_Frase_I,
434                 P_Frase_F,
435                 P_Frase_M,
436                 P_Frase_S,
437                 P_Label_Opt,
438                 P_Label_Sim,
439                 P_Label_X,
440                 P_Controle]
441 
442     return I_frases, P_frases
443 
444 def media(*args):
445     if not args:
446         return None
447     if len(args) == 1 and isinstance(args[0], collections.Container):
448         args = args[0]
449     total = sum(args)
450     ave = 1.0 * total / len(args)
451     return ave
452 def maximo(*args):
453     if not args:
454         return None
455     if len(args) == 1 and isinstance(args[0], collections.Container):
456         args = args[0]
457     mxm = max(args)
458     return mxm
459 
460 def minimo(*args):
461     if not args:
462         return None
463     if len(args) == 1 and isinstance(args[0], collections.Container):
464         args = args[0]
465     mnm = min(args)
466     return mnm
467 
468 # Definicao de funcoes
469 
470 def carrega_modelos(model_mane_str, model_opt_mane_str):
471 
```

```
472     pack_name = model_mane_str+"Pack"
473     dir_path = os.environ['USERCODEDIR']+"/resultados/jm/dengue/modelos/"
474     class_name = pack_name+"_"+model_mane_str
475     name = dir_path+class_name+".jmu"
476     model = JMUModel(name)
477     print  "Carregou modelo", name
478 
479     class_name_opt = pack_name+"_"+model_opt_mane_str
480     name_opt = dir_path+class_name_opt+".jmu"
481     model_opt = JMUModel(name_opt)
482     print  "Carregou modelo", name
483     print  "Carregou modelo", name_opt
484 
485     return model, model_opt
486 
487 def graficos_analise_k(res,
488                        f,
489                        k_vet,
490                        Ls,
491                        Lc,
492                        Lu,
493                        dados_dict_casos,
494                        dados_validos,
495                        fig,
496                        param):
497     global grid
498 
499 
500     I_Frases = param[0]
501     I_Frase_A = I_Frases[0]
502     I_Frase_I = I_Frases[1]
503     I_Frase_F = I_Frases[2]
504     I_Frase_M = I_Frases[3]
505     I_Frase_S = I_Frases[4]
506     I_Label_Opt = I_Frases[5]
507     I_Label_Sim = I_Frases[6]
508     I_Label_X = I_Frases[7]
509     I_Controle = I_Frases[8]
510 
511     P_Frases = param[1]
512     P_Frase_A = P_Frases[0]
513     P_Frase_I = P_Frases[1]
514     P_Frase_F = P_Frases[2]
515     P_Frase_M = P_Frases[3]
516     P_Frase_S = P_Frases[4]
517     P_Label_Opt = P_Frases[5]
518     P_Label_Sim = P_Frases[6]
519     P_Label_X = P_Frases[7]
520     P_Controle = P_Frases[8]
521 
522     dir_destino = param[2]
523 
524     fig_ext = param[3]
525 
526     s_time = media(res['s_time']) # s_time é vetor no resultado 'res'
527     f_time = media(res['f_time']) # f_time é vetor no resultado 'res'
528     mu_S = media(res['mu_S'])
529     beta = media(res['beta'])
```

```
530
531     params = {'legend.fontsize': 8,
532               'legend.linewidth': 1,
533               'legend.labelspacing':0.1
534               }
535     plt.rcParams.update(params)
536
537     indice_Ls = 0
538     for j in range(0, len(Ls)):
539         indice_Ls = indice_Ls+1
540         solver = Ls[j]
541
542         indice_Lc = 0
543         for i in range(0, len(Lc)):
544             indice_Lc = indice_Lc+1
545
546             ci = Lc[i]
547             ui = Lu[i]
548             caso = solver+"_caso_"+str(i+1)
549
550             validos = False
551             for l in range(0, len(k_vet)):
552                 if dados_validos[0][l] > 0:
553                     validos = True
554                     res = dados_dict_casos[0][l]
555
556             contador = 0
557
558             t = res['time']
559             tam_t = len(t)
560             tam_k = len(k_vet)
561
562             u1max_t = N.zeros(tam_k, float)
563             u2max_t = N.zeros(tam_k, float)
564             Kmax_t = N.zeros(tam_k, float)
565
566             aux = N.zeros(tam_t, float)
567
568             K_t = N.zeros((tam_k, tam_t), float)
569
570             int1_vet = N.zeros((tam_k, tam_t), float)
571             int2_vet = N.zeros((tam_k, tam_t), float)
572             int3_vet = N.zeros((tam_k, tam_t), float)
573             Aux_vet = N.zeros((tam_k, tam_t), float)
574
575             A_vet = N.zeros((tam_k, tam_t), float)
576             F_vet = N.zeros((tam_k, tam_t), float)
577             I_vet = N.zeros((tam_k, tam_t), float)
578             M_vet = N.zeros((tam_k, tam_t), float)
579             S_vet = N.zeros((tam_k, tam_t), float)
580             u1_vet = N.zeros((tam_k, tam_t), float)
581             u2_vet = N.zeros((tam_k, tam_t), float)
582             fo_vet = N.zeros((tam_k, tam_t), float)
583             t_vet = N.zeros((tam_k, tam_t), float)
584             indice_k = 0
585             legenda = [str() for x in range(len(k_vet))]
586             legenda_obj = [str() for x in range(len(k_vet))]
587             legenda_K = [str() for x in range(len(k_vet))]
```

```
588
589             for l in range(0, len(k_vet)):
590                 indice_k = indice_k + 1
591                 validos = False
592                 if dados_validos[indice_Lc-1][indice_k-1] > 0:
593                     validos = True
594                     res = dados_dict_casos[indice_Lc-1][indice_k-1]
595
596                 if validos:
597                     k = k_vet[l]
598                     #print "Entrou res validos k=",k
599
600                     legenda[contador]="k = "+str(k)
601                     A_vet[contador, ] = res['A']
602                     F_vet[contador, ] = res['F']
603                     I_vet[contador, ] = res['I']
604                     M_vet[contador, ] = res['M']
605                     S_vet[contador, ] = res['S']
606                     u1_vet[contador, ] = res['u1']
607                     u2_vet[contador, ] = res['u2']
608                     fo_vet[contador, ] = res['fo']
609                     t_vet[contador, ] = res['time']
610                     contador=contador+1
611
612             u1_axis=-1
613             u2_axis=-1
614             if contador == 4:
615                 estilo_linha = ['-','--','-.',':']
616             if contador == 3:
617                 estilo_linha = ['-','--','-.']
618             if contador == 2:
619                 estilo_linha = ['-','--']
620             if contador == 1:
621                 estilo_linha = ['-']
622
623             for j in range(0, contador):
624                 u1max_geral = maximo(u1_vet[j,])
625                 u1max_t[j] = maximo(u2_vet[j,])
626                 if u1_axis < u1max_geral:
627                     u1_axis = u1max_geral
628                 u2max_geral = maximo(u2_vet[j,])
629                 u2max_t[j] = maximo(u2_vet[j,])
630                 if u2_axis < u2max_geral:
631                     u2_axis = u2max_geral
632
633             if u1_axis < 0:
634                 u1_axis = 1e-6
635                 for j in range(0, contador):
636                     for i in range(0, len(u1_vet[j,])):
637                         u1_vet[j,] = 0
638
639             if u2_axis < 0:
640                 u2_axis = 1e-6
641                 for j in range(0, len(k_vet)):
642                     for i in range(0, len(u2_vet[j,])):
643                         u2_vet[j,] = 0
644
645             fig = fig+1
```

```
646             plt.figure(fig)
647 
648             for j in range(0, contador):
649 
650                 for n in range(0, tam_t-1):
651                     int1_vet[j,n] = u1_vet[j,n] + mu_S
652                     int2_vet[j,n] = u1_vet[j,n] + u2_vet[j,n]
653                     int3_vet[j,n] = u2_vet[j,n] - beta*M_vet[j,n]*I_vet[j,n
    ]/(M_vet[j,n]+S_vet[j,n])
654                 for i in range(0, tam_t-1):
655 
656                     aux[i] = math.exp(integral(int1_vet[j,0:i],t_vet[j,0:i
    ],0,i))
657                       #  print "aux[m]=",aux[m]
658                     Aux_vet[j,i]=int3_vet[j,i]*aux[i]
659                     if int2_vet[j,i] > 0:
660                         K_t[j,i] = integral(Aux_vet[j,0:i],t_vet[j,0:i],0,i
    )/((int2_vet[j,i])*math.exp(integral(int1_vet[j,0:i],t_vet[j,0:i],0,i)))
661                         if K_t[j,i] < k_vet[j]:
662                             K_t[j,i] = k_vet[j]
663                     else:
664                         K_t[j,i] = k_vet[j]
665 
666                 Kmax_t[j] = maximo(K_t[j,])
667                 legenda_K[j] = legenda[j]+" >> K_max = %8.0f" % (Kmax_t[j])
668                 plt.plot(t_vet[j,], K_t[j,], linewidth=3,linestyle=
    estilo_linha[j])
669             plt.legend(legenda_K,fontsize=14)
670             if grid > 0:
671                 plt.grid()
672             ax = plt.gca()
673             for axis in ['top', 'bottom', 'left', 'right']:
674                 ax.spines[axis].set_linewidth(2)
675             for tick in ax.xaxis.get_major_ticks():
676                 tick.label.set_fontsize(14)
677 
678             plt.title(u"Função K(t)", fontsize=14)
679             plt.ylabel("K(t)", fontsize=14)
680             plt.xlabel(P_Label_X, fontsize=14)
681 
682 
683             for i in range(0, len(fig_ext)):
684                plt.savefig(dir_destino+"/figuras/"+caso+"/"+fig_ext[i]+"/"
    +\
685                caso+"_K_t."+fig_ext[i])
686 
687             plt.close()
688 
689             fig = fig+1
690             plt.figure(fig)  # fig 2
691             plt.subplot(2, 1, 1)
692             for j in range(0, contador):
693                 plt.plot(t_vet[j,], u1_vet[j,], linewidth=3,linestyle=
    estilo_linha[j])
694 
695             plt.legend(legenda)
696             if grid > 0:
697                 plt.grid()
```

```
698             plt.ylim(0, u1_axis*1.05)
699             ax = plt.gca()
700             for axis in ['top', 'bottom', 'left', 'right']:
701                 ax.spines[axis].set_linewidth(2)
702             for tick in ax.xaxis.get_major_ticks():
703                 tick.label.set_fontsize(14)
704 
705             plt.title(I_Controle, fontsize=14)
706 
707             plt.ylabel('$u_{1}$', fontsize=14)
708             plt.subplot(2, 1, 2)
709             for j in range(0, contador):
710                 plt.plot(t_vet[j,], u2_vet[j,], linewidth=3,linestyle=
    estilo_linha[j])
711 
712 
713             plt.ylim(0, u2_axis*1.05)
714             if grid > 0:
715                 plt.grid()
716             plt.ylabel('$u_{2}$', fontsize=14)
717             plt.legend(legenda)
718             ax = plt.gca()
719             for axis in ['top', 'bottom', 'left', 'right']:
720                 ax.spines[axis].set_linewidth(2)
721             for tick in ax.xaxis.get_major_ticks():
722                 tick.label.set_fontsize(14)
723 
724             plt.title(I_Controle, fontsize=14)
725             plt.xlabel(I_Label_X, fontsize=14)
726 
727 
728             for i in range(0, len(fig_ext)):
729                 plt.savefig(dir_destino+"/figuras/"+caso+"/"+fig_ext[i]+"/"
    +\
730                 caso+"_Controles-analise-ing."+fig_ext[i])
731 
732             plt.title(P_Controle, fontsize=14)
733             plt.xlabel(P_Label_X, fontsize=14)
734 
735             plt.subplot(2, 1, 1)
736             plt.title(P_Controle, fontsize=14)
737             for i in range(0, len(fig_ext)):
738                 plt.savefig(dir_destino+"/figuras/"+caso+"/"+fig_ext[i]+"/"
    +\
739                 caso+"_Controles-analise."+fig_ext[i])
740 
741             plt.close()
742 
743             fig = fig+1
744             plt.figure(fig, figsize=(10, 6), dpi=100)  # fig 1
745 
746             for j in range(0, contador):
747                 print("Função Objetivo = %8.0f" % (fo_vet[j,len(fo_vet[j,])
    -1]))
748                 print >> f, "Função Objetivo = %8.0f" % (fo_vet[j,len(
    fo_vet[j,])-1])
749                 legenda_obj[j] = legenda[j]+" >> J[T] = %8.0f" % (fo_vet[j,
    len(fo_vet[j,])-1])
```

```
750                 plt.plot(t_vet[j,], fo_vet[j,], linewidth=3,linestyle =
    estilo_linha[j])
751
752             plt.legend(legenda_obj, loc = 4)
753             if grid > 0:
754                 plt.grid()
755
756             plt.ylabel('J[$u_{1}^{*}$,$u_{2}^{*}$;t]', fontsize=14)
757             ax = plt.gca()
758             for axis in ['top', 'bottom', 'left', 'right']:
759                 ax.spines[axis].set_linewidth(2)
760             for tick in ax.xaxis.get_major_ticks():
761                 tick.label.set_fontsize(14)
762             plt.title("Cost Function", fontsize=14)
763             plt.xlabel(I_Label_X, fontsize=14)
764
765             for i in range(0, len(fig_ext)):
766                 plt.savefig(dir_destino+"/figuras/"+caso+"/"+fig_ext[i]+"/"
    +\
767                 caso+"_fo-analise-ing."+fig_ext[i])
768             plt.title(u"Função Objetivo", fontsize=14)
769             plt.xlabel(P_Label_X, fontsize=14)
770             #plt.show()
771
772             for i in range(0, len(fig_ext)):
773                 plt.savefig(dir_destino+"/figuras/"+caso+"/"+fig_ext[i]+"/"
    +\
774                 caso+"_fo-analise."+fig_ext[i])
775             plt.close()
776
777             fig = fig+1
778             plt.figure(fig)  # fig 1
779             for j in range(0, contador):
780                 plt.plot(t_vet[j,], A_vet[j,], linewidth=3,linestyle=
    estilo_linha[j])
781             plt.legend(legenda)
782             if grid > 0:
783                 plt.grid()
784
785             plt.ylabel('A', fontsize=14)
786             ax = plt.gca()
787             for axis in ['top', 'bottom', 'left', 'right']:
788                 ax.spines[axis].set_linewidth(2)
789             for tick in ax.xaxis.get_major_ticks():
790                 tick.label.set_fontsize(14)
791             plt.title(I_Frase_A, fontsize=14)
792             plt.xlabel(I_Label_X, fontsize=14)
793             for i in range(0, len(fig_ext)):
794                 plt.savefig(dir_destino+"/figuras/"+caso+"/"+fig_ext[i]+"/"
    +\
795                 caso+"_Fase_Aquatica-analise-ing."+fig_ext[i])
796             plt.title(P_Frase_A, fontsize=14)
797             plt.xlabel(P_Label_X, fontsize=14)
798             for i in range(0, len(fig_ext)):
799                 plt.savefig(dir_destino+"/figuras/"+caso+"/"+fig_ext[i]+"/"
    +\
800                 caso+"_Fase_Aquatica-analise."+fig_ext[i])
801             plt.close()
```

```
802
803             fig = fig+1
804             plt.figure(fig)
805             plt.subplot(2, 1, 1)
806             for j in range(0, contador):
807                 plt.plot(t_vet[j,], F_vet[j,], linewidth=3,linestyle=
    estilo_linha[j])
808             plt.legend(legenda)
809             if grid > 0:
810                 plt.grid()
811             plt.title(I_Frase_F, fontsize=14)
812             plt.ylabel('F', fontsize=14)
813
814             ax = plt.gca()
815             for axis in ['top', 'bottom', 'left', 'right']:
816                 ax.spines[axis].set_linewidth(2)
817             for tick in ax.xaxis.get_major_ticks():
818                 tick.label.set_fontsize(14)
819
820             plt.subplot(2, 1, 2)
821             for j in range(0, contador):
822                 plt.plot(t_vet[j,], I_vet[j,], linewidth=3,linestyle=
    estilo_linha[j])
823             plt.legend(legenda)
824             if grid > 0:
825                 plt.grid()
826             plt.title(I_Frase_I, fontsize=14)
827             plt.ylabel('I', fontsize=14)
828             plt.xlabel(I_Label_X, fontsize=14)
829
830             ax = plt.gca()
831             for axis in ['top', 'bottom', 'left', 'right']:
832                 ax.spines[axis].set_linewidth(2)
833             for tick in ax.xaxis.get_major_ticks():
834                 tick.label.set_fontsize(14)
835             for i in range(0, len(fig_ext)):
836                 plt.savefig(dir_destino+"/figuras/"+caso+"/"+fig_ext[i]+"/"
    +\
837                 caso+"_Femeas-analise-ing."+fig_ext[i])
838             plt.subplot(2, 1, 1)
839             plt.title(P_Frase_F, fontsize=14)
840             plt.subplot(2, 1, 2)
841             plt.title(P_Frase_I, fontsize=14)
842             plt.xlabel(P_Label_X, fontsize=14)
843             for i in range(0, len(fig_ext)):
844                 plt.savefig(dir_destino+"/figuras/"+caso+"/"+fig_ext[i]+"/"
    +\
845                 caso+"_Femeas-analise."+fig_ext[i])
846
847             plt.close()
848
849             fig = fig+1
850             plt.figure(fig)
851             plt.subplot(2, 1, 1)
852             for j in range(0, contador):
853                 plt.plot(t_vet[j,], M_vet[j,], linewidth=3,linestyle=
    estilo_linha[j])
854             plt.legend(legenda)
```

```
855             if grid > 0:
856                 plt.grid()
857             plt.title(I_Frase_M, fontsize=14)
858             plt.ylabel('M', fontsize=14)
859 
860             ax = plt.gca()
861             for axis in ['top', 'bottom', 'left', 'right']:
862                 ax.spines[axis].set_linewidth(2)
863             for tick in ax.xaxis.get_major_ticks():
864                 tick.label.set_fontsize(14)
865 
866             plt.subplot(2, 1, 2)
867             for j in range(0, contador):
868                 plt.plot(t_vet[j,], S_vet[j,], linewidth=3,linestyle=
    estilo_linha[j])
869             plt.legend(legenda)
870             if grid > 0:
871                 plt.grid()
872             plt.title(I_Frase_S, fontsize=14)
873             plt.ylabel('S', fontsize=14)
874             plt.xlabel(I_Label_X, fontsize=14)
875 
876             ax = plt.gca()
877             for axis in ['top', 'bottom', 'left', 'right']:
878                 ax.spines[axis].set_linewidth(2)
879             for tick in ax.xaxis.get_major_ticks():
880                 tick.label.set_fontsize(14)
881 
882             for i in range(0, len(fig_ext)):
883                 plt.savefig(dir_destino+"/figuras/"+caso+"/"+fig_ext[i]+"/"
    +\
884                 caso+"_Machos-analise-ing."+fig_ext[i])
885 
886             plt.subplot(2, 1, 1)
887             plt.title(P_Frase_M, fontsize=14)
888             plt.subplot(2, 1, 2)
889             plt.title(P_Frase_S, fontsize=14)
890             plt.xlabel(P_Label_X, fontsize=14)
891             for i in range(0, len(fig_ext)):
892                 plt.savefig(dir_destino+"/figuras/"+caso+"/"+fig_ext[i]+"/"
    +\
893                 caso+"_Machos-analise."+fig_ext[i])
894 
895             plt.close()
896 
897         # plt.show();
898     return fig
899 
900 
901 def graficos_analise_IPOPT(res,
902                            f_caso,
903                            k_vet,
904                            Ls,
905                            Lc,
906                            Lu,
907                            dados_dict_casos,
908                            dados_validos,
909                            fig,
```

```
                    param):
    global grid

    dir_destino = param[2]


    params = {'legend.fontsize': 8,
              'legend.linewidth': 1,
              'legend.labelspacing':0.1
              }
    plt.rcParams.update(params)

    indice_Ls=0
    for j in range(0, len(Ls)):
        indice_Ls=indice_Ls+1
        solver = Ls[j]

        indice_Lc=0
        for i in range(0, len(Lc)):
            indice_Lc=indice_Lc+1

            ci = Lc[i]
            ui = Lu[i]
            caso = solver+"_caso_"+str(i+1)

            f = f_caso[i]

            validos = False
            contador=0
            indice_k = 0
            return_status = [int() for x in range(len(k_vet))]
            nbr_iter = [int() for x in range(len(k_vet))]
            objetive = [float() for x in range(len(k_vet))]
            total_exec_time = [float() for x in range(len(k_vet))]
            d_vetor = [int() for x in range(len(k_vet))]
            k_vetor = [float() for x in range(len(k_vet))]

            print >> f, "Arquivo de Análise IPOPT:"
            print >> f, "|   #   |      k      | Funçao Objetivo |  Nº Iter
    | Segundos |"
            print "Arquivo de Análise IPOPT:"
            print "|   #   |      k      | Funçao Objetivo |  Nº Iter |
    Segundos |"

            for l in range(0, len(k_vet)):
                indice_k = indice_k + 1
                validos = False
                if dados_validos[indice_Lc-1][indice_k-1] > 0:
                    validos = True
                    res = dados_dict_casos[indice_Lc-1][indice_k-1]

                if validos:
                    k=k_vet[l]
                    return_status[contador]=res['return_status']
                    nbr_iter[contador]=res['nbr_iter']
                    objetive[contador]=res['objetive']
                    total_exec_time[contador]=res['total_exec_time']
                    d_vetor[contador]=res['d']
```

```
966                     k_vetor[contador]=res['k']
967
968                     print("| %5d | %10.5f | %15.5f | %8d | %8d | %8d |" % (
    contador+1,k, objetive[contador], nbr_iter[contador], total_exec_time[
    contador],return_status[contador] ))
969                     print >> f, ("| %5d | %10.5f | %15.5f | %8d | %8d | %8d
     |" % (contador+1,k, objetive[contador], nbr_iter[contador],
    total_exec_time[contador],return_status[contador] ))
970
971                     contador=contador+1
972
973             fig = fig+1
974             plt.figure(fig,figsize=(12,8))  # 6x8 inches
975             plt.plot(k_vetor[0:contador], objetive[0:contador],'bo')
976             if grid > 0:
977                 plt.grid()
978             ax = plt.gca()
979             for axis in ['top', 'bottom', 'left', 'right']:
980                 ax.spines[axis].set_linewidth(2)
981             for tick in ax.xaxis.get_major_ticks():
982                 tick.label.set_fontsize(14)
983
984             plt.ylabel('J[T]', fontsize=14)
985             plt.xlabel('k', fontsize=14)
986             plt.title(u"Função Objetivo")
987             plt.savefig(dir_destino+caso+"/Analise_IPOPT_linear_por.eps",
    dpi = 150)
988
989             plt.title("Cost Function")
990             plt.savefig(dir_destino+caso+"/Analise_IPOPT_linear_ing.eps",
    dpi = 150)
991
992             #plt.show()
993             plt.close()
994
995             f.close()
996
997     return fig
998
999 def graficos_variacao_continua(f,
1000                                res,
1001                                res_est,
1002                                n_e,
1003                                n_cp,
1004                                Ls,
1005                                Lc,
1006                                Lu,
1007                                fig,
1008                                param):
1009
1010     global grid
1011     global varia1, dc1, varia2, dc2
1012
1013     I_Frases = param[0]
1014     I_Frase_A = I_Frases[0]
1015     I_Frase_I = I_Frases[1]
1016     I_Frase_F = I_Frases[2]
1017     I_Frase_M = I_Frases[3]
```

```
1018    I_Frase_S = I_Frases[4]
1019    I_Label_Opt = I_Frases[5]
1020    I_Label_Sim = I_Frases[6]
1021    I_Label_X = I_Frases[7]
1022    I_Controle = I_Frases[8]
1023
1024    P_Frases = param[1]
1025    P_Frase_A = P_Frases[0]
1026    P_Frase_I = P_Frases[1]
1027    P_Frase_F = P_Frases[2]
1028    P_Frase_M = P_Frases[3]
1029    P_Frase_S = P_Frases[4]
1030    P_Label_Opt = P_Frases[5]
1031    P_Label_Sim = P_Frases[6]
1032    P_Label_X = P_Frases[7]
1033    P_Controle = P_Frases[8]
1034
1035    dir_destino = param[2]
1036
1037    fig_ext = param[3]
1038    k = res_est['k']
1039    ci = res_est['ci']
1040
1041    legenda = ["-2*delta% -2*dc     " ,
1042               "   -delta% -dc       " ,
1043               "     Nominal          ",
1044               "   +delta% +dc     " ,
1045               "+2*delta% +2*dc "]
1046
1047    legenda1 = [" -%2d%% -%4.2f " % (2*varia1 , 2*dc1),
1048                " -%2d%% -%4.2f " % (varia1 ,dc1),
1049                "   Nominal",
1050                " +%2d%% +%4.2f" % (varia1 ,dc1),
1051                " +%2d%% +%4.2f" % (2*varia1 , 2*dc1)]
1052
1053    legenda2 = [" -%2d%% -%4.2f  " % (2*varia2 , 2*dc2),
1054                " -%2d%% -%4.2f " % (varia2 ,dc2),
1055                "   Nominal",
1056                " +%2d%% +%4.2f" % (varia2 ,dc2),
1057                " +%2d%% +%4.2f" % (2*varia2 , 2*dc2)]
1058
1059
1060    legenda_obj = ['','','','','']
1061
1062    params = {'legend.fontsize': 8,
1063              'legend.linewidth': 1,
1064              'legend.labelspacing':0.1
1065              }
1066    plt.rcParams.update(params)
1067
1068    model, model_opt = carrega_modelos("Dengue","Dengue_Opt")
1069
1070    indice_Ls=0
1071    for j in range(0, len(Ls)):
1072        indice_Ls=indice_Ls+1
1073        solver = Ls[j]
1074
1075        indice_Lc=0
```

```
1076        for i in range(0, len(Lc)):
1077            indice_Lc=indice_Lc+1
1078
1079            ci = Lc[i]
1080            ui = Lu[i]
1081            caso = solver+"_caso_"+str(i+1)
1082
1083            contador=0
1084
1085            t = res['time']
1086            tam_t=len(t)
1087
1088            contador = 5   #corresponde ao sinal e 4 variacoes
1089            tam_k = contador
1090
1091            A_vet = N.zeros((tam_k, tam_t), float)
1092            F_vet = N.zeros((tam_k, tam_t), float)
1093            I_vet = N.zeros((tam_k, tam_t), float)
1094            M_vet = N.zeros((tam_k, tam_t), float)
1095            S_vet = N.zeros((tam_k, tam_t), float)
1096            u1_vet = N.zeros((tam_k, tam_t), float)
1097            u2_vet = N.zeros((tam_k, tam_t), float)
1098            fo_vet = N.zeros((tam_k, tam_t), float)
1099
1100            t_vet = N.zeros((tam_k, tam_t), float)
1101
1102            varia1_vet = [1.0, 1.0, 1.0, 1.0, 1.0]
1103            varia1_vet[0] = 1.0-2.0*varia1/100.0
1104            varia1_vet[1] = 1.0-varia1/100.0
1105            varia1_vet[3] = 1.0+varia1/100.0
1106            varia1_vet[4] = 1.0+2.0*varia1/100.0
1107            dc1_vet = [-2.0*dc1, -dc1, 0, dc1, 2.0*dc1]
1108
1109            varia2_vet = [1.0, 1.0, 1.0, 1.0, 1.0]
1110            varia2_vet[0] = 1.0-2.0*varia2/100.0
1111            varia2_vet[1] = 1.0-varia2/100.0
1112            varia2_vet[3] = 1.0+varia2/100.0
1113            varia2_vet[4] = 1.0+2.0*varia2/100.0
1114            dc2_vet = [-2.0*dc2, -dc2, 0, dc2, 2.0*dc2 ]
1115
1116            print "vari1_vet = ", varia1_vet
1117            print "vari2_vet = ", varia2_vet
1118            print "dc1_vet = ", dc1_vet
1119            print "dc2_vet = ", dc2_vet
1120            print >> f,"vari1_vet = ", varia1_vet
1121            print >> f,"vari2_vet = ", varia2_vet
1122            print >> f,"dc1_vet = ", dc1_vet
1123            print >> f,"dc2_vet = ", dc2_vet
1124
1125
1126            u1_vet[2, ] = res['u1']
1127            u2_vet[2, ] = res['u2']
1128            A_vet[2, ] = res['A']
1129            F_vet[2, ] = res['F']
1130            I_vet[2, ] = res['I']
1131            M_vet[2, ] = res['M']
1132            S_vet[2, ] = res['S']
1133            fo_vet[2, ] = res['fo']
```

```
1134
1135            t_vet[2, ] = res['time']
1136
1137            s_time = media(res['s_time']) # s_time é vetor no resultado '
    res'
1138            f_time = media(res['f_time']) # f_time é vetor no resultado '
    res'
1139
1140            # Load model
1141            sim_model = model_opt  # O modelo do sistema sem o problema de
    controle otimizado
1142
1143            print 'valores otimização'
1144            sim_model.set('s_time', s_time)
1145            print "s_time do modelo a ser simulado = ", sim_model.get('
    s_time')
1146
1147            sim_model.set('f_time', f_time)
1148            print "f_time do modelo a ser simulado = ", sim_model.get('
    f_time')
1149
1150            sim_model.set('k', k)
1151            print "k do modelo a ser simulado = ", sim_model.get('k')
1152
1153            sim_model.set('c_1', ci[0])
1154            print "c_1 do modelo a ser simulado = ", sim_model.get('c_1')
1155
1156            sim_model.set('c_3', ci[1])
1157            print "c_3 do modelo a ser simulado = ", sim_model.get('c_2')
1158
1159            sim_model.set('c_3', ci[2])
1160            print "c_3 do modelo a ser simulado = ", sim_model.get('c_3')
1161
1162            sim_model.set('c_4', ci[3])
1163            print "c_4 do modelo a ser simulado = ", sim_model.get('c_4')
1164
1165            print 'n_e = ', n_e
1166            print 'n_cp = ', n_cp
1167            print 'start_time = ', s_time
1168            print 'final_time = ', f_time
1169
1170
1171            print >> f,'valores otimização'
1172            print >> f, "s_time do modelo a ser simulado = ", sim_model.get
    ('s_time')
1173            print >> f, "f_time do modelo a ser simulado = ", sim_model.get
    ('f_time')
1174            print >> f, "k do modelo a ser simulado = ", sim_model.get('k')
1175            print >> f, "c_1 do modelo a ser simulado = ", sim_model.get('
    c_1')
1176            print >> f, "c_3 do modelo a ser simulado = ", sim_model.get('
    c_2')
1177            print >> f, "c_3 do modelo a ser simulado = ", sim_model.get('
    c_3')
1178            print >> f, "c_4 do modelo a ser simulado = ", sim_model.get('
    c_4')
1179            print >> f, 'n_e = ', n_e
1180            print >> f, 'n_cp = ', n_cp
```

```
1181            print >> f, 'start_time = ', s_time
1182            print >> f, 'final_time = ', f_time
1183
1184            for j in range(0, contador):
1185                u1_vet[j,] = u1_vet[2,]*varia1_vet[j] + dc1_vet[j]
1186                u2_vet[j,] = u2_vet[2,]*varia2_vet[j] + dc2_vet[j]
1187
1188                for m in range(0, len(u1_vet[j,])):
1189                    u1_vet[j,m] = lim_zero(u1_vet[j,m])
1190                    u2_vet[j,m] = lim_zero(u2_vet[j,m])
1191
1192                t_vet[j,] = t_vet[2,]
1193                u_traj = N.transpose(N.vstack((t_vet[j,], u1_vet[j,],
    u2_vet[j,])))
1194                input_obj = (['u1', 'u2'], u_traj)
1195                sim_opts = sim_model.simulate_options()
1196                sim_opts['ncp'] = n_e*n_cp
1197                sim_opts["initialize"] = True
1198                res_sim = sim_model.simulate(start_time=s_time, final_time=
    f_time,
1199                                input=input_obj, options=sim_opts)
1200
1201                A_vet[j,] = res_sim['A']
1202                F_vet[j,] = res_sim['F']
1203                I_vet[j,] = res_sim['I']
1204                M_vet[j,] = res_sim['M']
1205                S_vet[j,] = res_sim['S']
1206                fo_vet[j,] = res_sim['fo']
1207
1208            estilo_linha = ['-','--','-','-.',':']
1209            stride = max( int(len(t_vet[j,]) / 20), 1)
1210
1211            fig = fig+1
1212            plt.figure(fig)  # fig 1
1213            for j in range(0, contador):
1214                print("Função Objetivo = %8.0f" % (fo_vet[j,len(fo_vet[j,])
    -1]))
1215                print >> f, "Função Objetivo = %8.0f" % (fo_vet[j,len(
    fo_vet[j,])-1])
1216                legenda_obj[j] = legenda[j]+" >> J[T] = %8.0f" % (fo_vet[j,
    len(fo_vet[j,])-1])
1217                if j==2:
1218                    plt.plot(t_vet[j,], fo_vet[j,], "-o",color='black',
    markevery=stride)
1219                else:
1220                    plt.plot(t_vet[j,], fo_vet[j,], linewidth=3,linestyle =
     estilo_linha[j])
1221
1222            plt.legend(legenda_obj, loc = 4)
1223            if grid > 0:
1224                plt.grid()
1225
1226            plt.ylabel('J[$u_{1}^{*}$,$u_{2}^{*}$;t]', fontsize=14)
1227            ax = plt.gca()
1228            for axis in ['top', 'bottom', 'left', 'right']:
1229                ax.spines[axis].set_linewidth(2)
1230            for tick in ax.xaxis.get_major_ticks():
1231                tick.label.set_fontsize(14)
```

```
1232            plt.title("Cost Function (k="+str(k)+")", fontsize=14)
1233            plt.xlabel(I_Label_X, fontsize=14)
1234            for i in range(0, len(fig_ext)):
1235                plt.savefig(dir_destino+"/figuras/"+caso+"/"+fig_ext[i]+"/"
    +\
1236                caso+"_fo-var_cont-ing."+fig_ext[i])
1237            plt.title(u"Função Objetivo (k="+str(k)+")", fontsize=14)
1238            plt.xlabel(P_Label_X, fontsize=14)
1239            for i in range(0, len(fig_ext)):
1240                plt.savefig(dir_destino+"/figuras/"+caso+"/"+fig_ext[i]+"/"
    +\
1241                caso+"_fo-var_cont."+fig_ext[i])
1242            plt.close()
1243
1244            u1_axis=-1
1245            u2_axis=-1
1246
1247            for j in range(0, contador):
1248                umax_geral = maximo(u1_vet[j,])
1249                if u1_axis < umax_geral:
1250                    u1_axis = umax_geral
1251                umax_geral = maximo(u2_vet[j,])
1252                if u2_axis < umax_geral:
1253                    u2_axis = umax_geral
1254
1255            if u1_axis < 0:
1256                u1_axis = 1e-6
1257                for j in range(0, contador):
1258                    for i in range(0, len(u1_vet[j,])):
1259                        u1_vet[j,] = 0
1260
1261            if u2_axis < 0:
1262                u2_axis = 1e-6
1263                for j in range(0, contador):
1264                    for i in range(0, len(u2_vet[j,])):
1265                        u2_vet[j,] = 0
1266
1267            fig = fig+1
1268            plt.figure(fig)  # fig 2
1269            plt.subplot(2, 1, 1)
1270            for j in range(0, contador):
1271                if j==2:
1272                    plt.plot(t_vet[j,], u1_vet[j,],  "-o",color='black',
    markevery=stride)
1273                else:
1274                    plt.plot(t_vet[j,], u1_vet[j,], linewidth=3,linestyle =
     estilo_linha[j])
1275            plt.legend(legenda1)
1276            if grid > 0:
1277                plt.grid()
1278            plt.ylim(0, u1_axis*1.05)
1279            ax = plt.gca()
1280            for axis in ['top', 'bottom', 'left', 'right']:
1281                ax.spines[axis].set_linewidth(2)
1282            for tick in ax.xaxis.get_major_ticks():
1283                tick.label.set_fontsize(14)
1284
1285            plt.title(I_Controle +" (k="+str(k)+")", fontsize=14)
```

```
1286
1287            plt.ylabel('$u_{1}$', fontsize=14)
1288            plt.subplot(2, 1, 2)
1289            for j in range(0, contador):
1290                if j==2:
1291                    plt.plot(t_vet[j,], u2_vet[j,],  "-o",color='black',
    markevery=stride)
1292                else:
1293                    plt.plot(t_vet[j,], u2_vet[j,], linewidth=3,linestyle =
     estilo_linha[j])
1294
1295            plt.ylim(0, u2_axis*1.05)
1296            if grid > 0:
1297                plt.grid()
1298            plt.ylabel('$u_{2}$', fontsize=14)
1299            plt.legend(legenda2)
1300            ax = plt.gca()
1301            for axis in ['top', 'bottom', 'left', 'right']:
1302                ax.spines[axis].set_linewidth(2)
1303            for tick in ax.xaxis.get_major_ticks():
1304                tick.label.set_fontsize(14)
1305
1306            plt.title(I_Controle +" (k="+str(k)+")", fontsize=14)
1307            plt.xlabel(I_Label_X, fontsize=14)
1308            for i in range(0, len(fig_ext)):
1309                plt.savefig(dir_destino+"/figuras/"+caso+"/"+fig_ext[i]+"/"
    +\
1310                caso+"_Controles-var_cont-ing."+fig_ext[i])
1311
1312            plt.title(P_Controle +" (k="+str(k)+")", fontsize=14)
1313            plt.xlabel(P_Label_X, fontsize=14)
1314
1315            plt.subplot(2, 1, 1)
1316            plt.title(P_Controle +" (k="+str(k)+")", fontsize=14)
1317            for i in range(0, len(fig_ext)):
1318                plt.savefig(dir_destino+"/figuras/"+caso+"/"+fig_ext[i]+"/"
    +\
1319                caso+"_Controles-var_cont."+fig_ext[i])
1320
1321            plt.close()
1322
1323            fig = fig+1
1324            plt.figure(fig)  # fig 1
1325            for j in range(0, contador):
1326                if j==2:
1327                    plt.plot(t_vet[j,], A_vet[j,], "-o",color='black',
    markevery=stride)
1328                else:
1329                    plt.plot(t_vet[j,], A_vet[j,], linewidth=3,linestyle =
    estilo_linha[j])
1330            plt.legend(legenda)
1331            if grid > 0:
1332                plt.grid()
1333
1334            plt.ylabel('A', fontsize=14)
1335            ax = plt.gca()
1336            for axis in ['top', 'bottom', 'left', 'right']:
1337                ax.spines[axis].set_linewidth(2)
```

```
1338            for tick in ax.xaxis.get_major_ticks():
1339                tick.label.set_fontsize(14)
1340            plt.title(I_Frase_A +" (k="+str(k)+")", fontsize=14)
1341            plt.xlabel(I_Label_X, fontsize=14)
1342            for i in range(0, len(fig_ext)):
1343                plt.savefig(dir_destino+"/figuras/"+caso+"/"+fig_ext[i]+"/"
    +\
1344                caso+"_Fase_Aquatica-var_cont-ing."+fig_ext[i])
1345            plt.title(P_Frase_A +" (k="+str(k)+")", fontsize=14)
1346            plt.xlabel(P_Label_X, fontsize=14)
1347            for i in range(0, len(fig_ext)):
1348                plt.savefig(dir_destino+"/figuras/"+caso+"/"+fig_ext[i]+"/"
    +\
1349                caso+"_Fase_Aquatica-var_cont."+fig_ext[i])
1350            plt.close()
1351
1352            fig = fig+1
1353            plt.figure(fig)
1354            plt.subplot(2, 1, 1)
1355            for j in range(0, contador):
1356                if j==2:
1357                    plt.plot(t_vet[j,], F_vet[j,],  "-o",color='black',
    markevery=stride)
1358                else:
1359                    plt.plot(t_vet[j,], F_vet[j,], linewidth=3,linestyle =
    estilo_linha[j])
1360            plt.legend(legenda)
1361            if grid > 0:
1362                plt.grid()
1363            plt.title(I_Frase_F +" (k="+str(k)+")", fontsize=14)
1364            plt.ylabel('F', fontsize=14)
1365
1366            ax = plt.gca()
1367            for axis in ['top', 'bottom', 'left', 'right']:
1368                ax.spines[axis].set_linewidth(2)
1369            for tick in ax.xaxis.get_major_ticks():
1370                tick.label.set_fontsize(14)
1371
1372            plt.subplot(2, 1, 2)
1373            for j in range(0, contador):
1374                if j==2:
1375                    plt.plot(t_vet[j,], I_vet[j,],  "-o",color='black',
    markevery=stride)
1376                else:
1377                    plt.plot(t_vet[j,], I_vet[j,], linewidth=3,linestyle =
    estilo_linha[j])
1378            plt.legend(legenda)
1379            if grid > 0:
1380                plt.grid()
1381            plt.title(I_Frase_I +" (k="+str(k)+")", fontsize=14)
1382            plt.ylabel('I', fontsize=14)
1383            plt.xlabel(I_Label_X, fontsize=14)
1384
1385            ax = plt.gca()
1386            for axis in ['top', 'bottom', 'left', 'right']:
1387                ax.spines[axis].set_linewidth(2)
1388            for tick in ax.xaxis.get_major_ticks():
1389                tick.label.set_fontsize(14)
```

```
            for i in range(0, len(fig_ext)):
                plt.savefig(dir_destino+"/figuras/"+caso+"/"+fig_ext[i]+"/"
+\
                caso+"_Femeas-var_cont-ing."+fig_ext[i])
            plt.subplot(2, 1, 1)
            plt.title(P_Frase_F +" (k="+str(k)+")", fontsize=14)
            plt.subplot(2, 1, 2)
            plt.title(P_Frase_I +" (k="+str(k)+")", fontsize=14)
            plt.xlabel(P_Label_X, fontsize=14)
            for i in range(0, len(fig_ext)):
                plt.savefig(dir_destino+"/figuras/"+caso+"/"+fig_ext[i]+"/"
+\
                caso+"_Femeas-var_cont."+fig_ext[i])

            plt.close()

            fig = fig+1
            plt.figure(fig)
            plt.subplot(2, 1, 1)
            for j in range(0, contador):
                if j==2:
                    plt.plot(t_vet[j,], M_vet[j,],  "-o",color='black',
markevery=stride)
                else:
                    plt.plot(t_vet[j,], M_vet[j,], linewidth=3,linestyle =
estilo_linha[j])
            plt.legend(legenda)
            if grid > 0:
                plt.grid()
            plt.title(I_Frase_M +" (k="+str(k)+")", fontsize=14)
            plt.ylabel('M', fontsize=14)

            ax = plt.gca()
            for axis in ['top', 'bottom', 'left', 'right']:
                ax.spines[axis].set_linewidth(2)
            for tick in ax.xaxis.get_major_ticks():
                tick.label.set_fontsize(14)

            plt.subplot(2, 1, 2)
            for j in range(0, contador):
                if j==2:
                    plt.plot(t_vet[j,], S_vet[j,],  "-o",color='black',
markevery=stride)
                else:
                    plt.plot(t_vet[j,], S_vet[j,], linewidth=3,linestyle =
estilo_linha[j])
            plt.legend(legenda)
            if grid > 0:
                plt.grid()
            plt.title(I_Frase_S +" (k="+str(k)+")", fontsize=14)
            plt.ylabel('S', fontsize=14)
            plt.xlabel(I_Label_X, fontsize=14)

            ax = plt.gca()
            for axis in ['top', 'bottom', 'left', 'right']:
                ax.spines[axis].set_linewidth(2)
            for tick in ax.xaxis.get_major_ticks():
                tick.label.set_fontsize(14)
```

```
1442
1443            for i in range(0, len(fig_ext)):
1444                plt.savefig(dir_destino+"/figuras/"+caso+"/"+fig_ext[i]+"/"
    +\
1445                caso+"_Machos-var_cont-ing."+fig_ext[i])
1446
1447            plt.subplot(2, 1, 1)
1448            plt.title(P_Frase_M +" (k="+str(k)+")", fontsize=14)
1449            plt.subplot(2, 1, 2)
1450            plt.title(P_Frase_S +" (k="+str(k)+")", fontsize=14)
1451            plt.xlabel(P_Label_X, fontsize=14)
1452            for i in range(0, len(fig_ext)):
1453                plt.savefig(dir_destino+"/figuras/"+caso+"/"+fig_ext[i]+"/"
    +\
1454                caso+"_Machos-var_cont."+fig_ext[i])
1455
1456            plt.close()
1457
1458        # plt.show();
1459    return fig
1460
1461 def graficos_variacao_randomica(f,
1462                                 res,
1463                                 res_est,
1464                                 n_e,
1465                                 n_cp,
1466                                 Ls,
1467                                 Lc,
1468                                 Lu,
1469                                 fig,
1470                                 param):
1471
1472     global grid
1473     global varia1, varia2
1474
1475     I_Frases = param[0]
1476     I_Frase_A = I_Frases[0]
1477     I_Frase_I = I_Frases[1]
1478     I_Frase_F = I_Frases[2]
1479     I_Frase_M = I_Frases[3]
1480     I_Frase_S = I_Frases[4]
1481     I_Label_Opt = I_Frases[5]
1482     I_Label_Sim = I_Frases[6]
1483     I_Label_X = I_Frases[7]
1484     I_Controle = I_Frases[8]
1485
1486     P_Frases = param[1]
1487     P_Frase_A = P_Frases[0]
1488     P_Frase_I = P_Frases[1]
1489     P_Frase_F = P_Frases[2]
1490     P_Frase_M = P_Frases[3]
1491     P_Frase_S = P_Frases[4]
1492     P_Label_Opt = P_Frases[5]
1493     P_Label_Sim = P_Frases[6]
1494     P_Label_X = P_Frases[7]
1495     P_Controle = P_Frases[8]
1496
1497     dir_destino = param[2]
```

```
1498
1499    fig_ext = param[3]
1500    k = res_est['k']
1501    ci = res_est['ci']
1502
1503
1504
1505
1506    legenda = ["    Nominal                ",
1507               "+random(delta%)     ",
1508               "+random(2*delta%) "]
1509    legenda1 = ["    Nominal       ",
1510               "+random(%2d" % (varia1)+"%)",
1511               "+random(%2d" % (2*varia1)+"%)"]
1512    legenda2 = ["    Nominal       ",
1513               "+random(%2d" % (varia2)+"%)",
1514               "+random(%2d" % (2*varia2)+"%)"]
1515
1516    legenda_obj = ['','','']
1517
1518    params = {'legend.fontsize': 8,
1519              'legend.linewidth': 1,
1520              'legend.labelspacing':0.1
1521              }
1522    plt.rcParams.update(params)
1523
1524    model, model_opt = carrega_modelos("Dengue","Dengue_Opt")
1525
1526    indice_Ls=0
1527    for j in range(0, len(Ls)):
1528        indice_Ls=indice_Ls+1
1529        solver = Ls[j]
1530
1531        indice_Lc=0
1532        for i in range(0, len(Lc)):
1533            indice_Lc=indice_Lc+1
1534
1535            ci = Lc[i]
1536            ui = Lu[i]
1537            caso = solver+"_caso_"+str(i+1)
1538
1539            contador=0
1540
1541            t = res['time']
1542            tam_t=len(t)
1543
1544            contador = 3   #corresponde ao sinal e 4 variacoes
1545            tam_k = contador
1546
1547            A_vet = N.zeros((tam_k, tam_t), float)
1548            F_vet = N.zeros((tam_k, tam_t), float)
1549            I_vet = N.zeros((tam_k, tam_t), float)
1550            M_vet = N.zeros((tam_k, tam_t), float)
1551            S_vet = N.zeros((tam_k, tam_t), float)
1552            u1_vet = N.zeros((tam_k, tam_t), float)
1553            u2_vet = N.zeros((tam_k, tam_t), float)
1554            fo_vet = N.zeros((tam_k, tam_t), float)
1555
```

```
1556            t_vet = N.zeros((tam_k, tam_t), float)
1557
1558            u1_vet[0, ] = res['u1']
1559            u2_vet[0, ] = res['u2']
1560            A_vet[0, ] = res['A']
1561            F_vet[0, ] = res['F']
1562            I_vet[0, ] = res['I']
1563            M_vet[0, ] = res['M']
1564            S_vet[0, ] = res['S']
1565            fo_vet[0, ] = res['fo']
1566
1567            t_vet[0, ] = res['time']
1568
1569            s_time = media(res['s_time']) # s_time é vetor no resultado '
    res'
1570            f_time = media(res['f_time']) # f_time é vetor no resultado '
    res'
1571
1572            # Load model
1573            sim_model = model_opt  # O modelo do sistema sem o problema de
    controle otimizado
1574
1575            print 'valores otimização'
1576            sim_model.set('s_time', s_time)
1577            print "s_time do modelo a ser simulado = ", sim_model.get('
    s_time')
1578
1579            sim_model.set('f_time', f_time)
1580            print "f_time do modelo a ser simulado = ", sim_model.get('
    f_time')
1581
1582            sim_model.set('k', k)
1583            print "k do modelo a ser simulado = ", sim_model.get('k')
1584
1585            sim_model.set('c_1', ci[0])
1586            print "c_1 do modelo a ser simulado = ", sim_model.get('c_1')
1587
1588            sim_model.set('c_3', ci[1])
1589            print "c_3 do modelo a ser simulado = ", sim_model.get('c_2')
1590
1591            sim_model.set('c_3', ci[2])
1592            print "c_3 do modelo a ser simulado = ", sim_model.get('c_3')
1593
1594            sim_model.set('c_4', ci[3])
1595            print "c_4 do modelo a ser simulado = ", sim_model.get('c_4')
1596
1597            print 'n_e = ', n_e
1598            print 'n_cp = ', n_cp
1599            print 'start_time = ', s_time
1600            print 'final_time = ', f_time
1601
1602
1603
1604            print >> f,'valores otimização'
1605            print >> f, "s_time do modelo a ser simulado = ", sim_model.get
    ('s_time')
1606            print >> f, "f_time do modelo a ser simulado = ", sim_model.get
    ('f_time')
```

```
1607            print >> f, "k do modelo a ser simulado = ", sim_model.get('k')
1608            print >> f, "c_1 do modelo a ser simulado = ", sim_model.get('
    c_1')
1609            print >> f, "c_3 do modelo a ser simulado = ", sim_model.get('
    c_2')
1610            print >> f, "c_3 do modelo a ser simulado = ", sim_model.get('
    c_3')
1611            print >> f, "c_4 do modelo a ser simulado = ", sim_model.get('
    c_4')
1612            print >> f, 'n_e = ', n_e
1613            print >> f, 'n_cp = ', n_cp
1614            print >> f, 'start_time = ', s_time
1615            print >> f, 'final_time = ', f_time
1616
1617
1618            for x in range(1, contador):
1619                t_vet[x,] = t_vet[0,]
1620                if x == 1:
1621                    for l in range(0, len(t_vet[j,])):
1622                        u1_vet[x,l] = u1_vet[0,l]*(1+varia1*random.uniform
    (-1.0, 1.0)/100.0)
1623                        u2_vet[x,l] = u2_vet[0,l]*(1+varia2*random.uniform
    (-1.0, 1.0)/100.0)
1624                if x ==2 :
1625                    for l in range(0, len(t_vet[j,])):
1626                        u1_vet[x,l] = u1_vet[0,l]*(1+2*varia1*random.
    uniform(-1.0, 1.0)/100.0)
1627                        u2_vet[x,l] = u2_vet[0,l]*(1+2*varia2*random.
    uniform(-1.0, 1.0)/100.0)
1628
1629                u_traj = N.transpose(N.vstack((t_vet[x,], u1_vet[x,],
    u2_vet[x,])))
1630                input_obj = (['u1', 'u2'], u_traj)
1631                sim_opts = sim_model.simulate_options()
1632                sim_opts['ncp'] = n_e*n_cp
1633                sim_opts["initialize"] = True
1634                res_sim = sim_model.simulate(start_time=s_time, final_time=
    f_time,
1635                                             input=input_obj, options=sim_opts)
1636
1637                A_vet[x,] = res_sim['A']
1638                F_vet[x,] = res_sim['F']
1639                I_vet[x,] = res_sim['I']
1640                M_vet[x,] = res_sim['M']
1641                S_vet[x,] = res_sim['S']
1642                fo_vet[x,] = res_sim['fo']
1643
1644            estilo_linha = ['-','--',':']
1645            stride = max( int(len(t_vet[j,]) / 20), 1)
1646
1647            fig = fig+1
1648            plt.figure(fig)  # fig 1
1649            for j in range(0, contador):
1650                print("Função Objetivo = %8.0f" % (fo_vet[j,len(fo_vet[j,])
    -1]))
1651                print >> f, "Função Objetivo = %8.0f" % (fo_vet[j,len(
    fo_vet[j,])-1])
1652                legenda_obj[j] = legenda[j]+" >> J[$u_{1}^{*}$,$u_{2}^{*}$;
```

```
    T] = %8.0f" % (fo_vet[j,len(fo_vet[j,])-1])
1653                if j==0:
1654                    plt.plot(t_vet[j,], fo_vet[j,], "-o",color='black',
    markevery=stride)
1655                else:
1656                    plt.plot(t_vet[j,], fo_vet[j,], linewidth=3,linestyle =
     estilo_linha[j])
1657
1658            plt.legend(legenda_obj, loc = 4)
1659            if grid > 0:
1660                plt.grid()
1661
1662            plt.ylabel('J[$u_{1}^{*}$,$u_{2}^{*}$;t]', fontsize=14)
1663            ax = plt.gca()
1664            for axis in ['top', 'bottom', 'left', 'right']:
1665                ax.spines[axis].set_linewidth(2)
1666            for tick in ax.xaxis.get_major_ticks():
1667                tick.label.set_fontsize(14)
1668
1669
1670            plt.title("Cost Function" +" (k="+str(k)+")", fontsize=14)
1671            plt.xlabel(I_Label_X, fontsize=14)
1672
1673            for i in range(0, len(fig_ext)):
1674                plt.savefig(dir_destino+"/figuras/"+caso+"/"+fig_ext[i]+"/"
    +\
1675                caso+"_fo-var_rand-ing."+fig_ext[i])
1676
1677            plt.title(u"Função Objetivo" +" (k="+str(k)+")", fontsize=14)
1678            plt.xlabel(P_Label_X, fontsize=14)
1679
1680            for i in range(0, len(fig_ext)):
1681                plt.savefig(dir_destino+"/figuras/"+caso+"/"+fig_ext[i]+"/"
    +\
1682                caso+"_fo-var_rand."+fig_ext[i])
1683
1684            plt.close()
1685
1686            u1_axis=-1
1687            u2_axis=-1
1688
1689            for m in range(0, contador):
1690                umax_geral = maximo(u1_vet[m,])
1691                if u1_axis < umax_geral:
1692                    u1_axis = umax_geral
1693                umax_geral = maximo(u2_vet[m,])
1694                if u2_axis < umax_geral:
1695                    u2_axis = umax_geral
1696
1697            if u1_axis < 0:
1698                u1_axis = 1e-6
1699                for j in range(0, contador):
1700                    for i in range(0, len(u1_vet[j,])):
1701                        u1_vet[j,] = 0
1702
1703            if u2_axis < 0:
1704                u2_axis = 1e-6
1705                for j in range(0, contador):
```

```
1706                    for i in range(0, len(u2_vet[j,])):
1707                        u2_vet[j,] = 0
1708
1709            fig = fig+1
1710            plt.figure(fig)  # fig 2
1711            plt.subplot(2, 1, 1)
1712            for j in range(0, contador):
1713                if j==0:
1714                    plt.plot(t_vet[j,], u1_vet[j,],  "-o",color='black',
    markevery=stride)
1715                else:
1716                    plt.plot(t_vet[j,], u1_vet[j,], linewidth=3,linestyle =
     estilo_linha[j])
1717            plt.legend(legenda1)
1718            if grid > 0:
1719                plt.grid()
1720            plt.ylim(0, u1_axis*1.05)
1721            ax = plt.gca()
1722            for axis in ['top', 'bottom', 'left', 'right']:
1723                ax.spines[axis].set_linewidth(2)
1724            for tick in ax.xaxis.get_major_ticks():
1725                tick.label.set_fontsize(14)
1726
1727            plt.title(I_Controle +" (k="+str(k)+")", fontsize=14)
1728
1729            plt.ylabel('$u_{1}$', fontsize=14)
1730            plt.subplot(2, 1, 2)
1731            for j in range(0, contador):
1732                if j==0:
1733                    plt.plot(t_vet[j,], u2_vet[j,],  "-o",color='black',
    markevery=stride)
1734                else:
1735                    plt.plot(t_vet[j,], u2_vet[j,], linewidth=3,linestyle =
     estilo_linha[j])
1736
1737            plt.ylim(0, u2_axis*1.05)
1738            if grid > 0:
1739                plt.grid()
1740            plt.ylabel('$u_{2}$', fontsize=14)
1741            plt.legend(legenda2)
1742            ax = plt.gca()
1743            for axis in ['top', 'bottom', 'left', 'right']:
1744                ax.spines[axis].set_linewidth(2)
1745            for tick in ax.xaxis.get_major_ticks():
1746                tick.label.set_fontsize(14)
1747
1748            plt.title(I_Controle +" (k="+str(k)+")", fontsize=14)
1749            plt.xlabel(I_Label_X, fontsize=14)
1750            for i in range(0, len(fig_ext)):
1751                plt.savefig(dir_destino+"/figuras/"+caso+"/"+fig_ext[i]+"/"
    +\
1752                caso+"_Controles-var_rand-ing."+fig_ext[i])
1753
1754            plt.title(P_Controle +" (k="+str(k)+")", fontsize=14)
1755            plt.xlabel(P_Label_X, fontsize=14)
1756
1757            plt.subplot(2, 1, 1)
1758            plt.title(P_Controle +" (k="+str(k)+")", fontsize=14)
```

```
1759            for i in range(0, len(fig_ext)):
1760                plt.savefig(dir_destino+"/figuras/"+caso+"/"+fig_ext[i]+"/"
    +\
1761                caso+"_Controles-var_rand."+fig_ext[i])
1762
1763            plt.close()
1764
1765            fig = fig+1
1766            plt.figure(fig)  # fig 1
1767            for j in range(0, contador):
1768                if j==0:
1769                    plt.plot(t_vet[j,], A_vet[j,], "-o",color='black',
    markevery=stride)
1770                else:
1771                    plt.plot(t_vet[j,], A_vet[j,], linewidth=3,linestyle =
    estilo_linha[j])
1772            plt.legend(legenda)
1773            if grid > 0:
1774                plt.grid()
1775
1776            plt.ylabel('A', fontsize=14)
1777            ax = plt.gca()
1778            for axis in ['top', 'bottom', 'left', 'right']:
1779                ax.spines[axis].set_linewidth(2)
1780            for tick in ax.xaxis.get_major_ticks():
1781                tick.label.set_fontsize(14)
1782            plt.title(I_Frase_A +" (k="+str(k)+")", fontsize=14)
1783            plt.xlabel(I_Label_X, fontsize=14)
1784            for i in range(0, len(fig_ext)):
1785                plt.savefig(dir_destino+"/figuras/"+caso+"/"+fig_ext[i]+"/"
    +\
1786                caso+"_Fase_Aquatica-var_rand-ing."+fig_ext[i])
1787            plt.title(P_Frase_A +" (k="+str(k)+")", fontsize=14)
1788            plt.xlabel(P_Label_X, fontsize=14)
1789            for i in range(0, len(fig_ext)):
1790                plt.savefig(dir_destino+"/figuras/"+caso+"/"+fig_ext[i]+"/"
    +\
1791                caso+"_Fase_Aquatica-var_rand."+fig_ext[i])
1792            plt.close()
1793
1794            fig = fig+1
1795            plt.figure(fig)
1796            plt.subplot(2, 1, 1)
1797            for j in range(0, contador):
1798                if j==0:
1799                    plt.plot(t_vet[j,], F_vet[j,],  "-o",color='black',
    markevery=stride)
1800                else:
1801                    plt.plot(t_vet[j,], F_vet[j,], linewidth=3,linestyle =
    estilo_linha[j])
1802            plt.legend(legenda)
1803            if grid > 0:
1804                plt.grid()
1805            plt.title(I_Frase_F +" (k="+str(k)+")", fontsize=14)
1806            plt.ylabel('F', fontsize=14)
1807
1808            ax = plt.gca()
1809            for axis in ['top', 'bottom', 'left', 'right']:
```

```
1810                ax.spines[axis].set_linewidth(2)
1811            for tick in ax.xaxis.get_major_ticks():
1812                tick.label.set_fontsize(14)
1813
1814            plt.subplot(2, 1, 2)
1815            for j in range(0, contador):
1816                if j==0:
1817                    plt.plot(t_vet[j,], I_vet[j,],  "-o",color='black',
    markevery=stride)
1818                else:
1819                    plt.plot(t_vet[j,], I_vet[j,], linewidth=3,linestyle =
    estilo_linha[j])
1820            plt.legend(legenda)
1821            if grid > 0:
1822                plt.grid()
1823            plt.title(I_Frase_I +" (k="+str(k)+")", fontsize=14)
1824            plt.ylabel('I', fontsize=14)
1825            plt.xlabel(I_Label_X, fontsize=14)
1826
1827            ax = plt.gca()
1828            for axis in ['top', 'bottom', 'left', 'right']:
1829                ax.spines[axis].set_linewidth(2)
1830            for tick in ax.xaxis.get_major_ticks():
1831                tick.label.set_fontsize(14)
1832            for i in range(0, len(fig_ext)):
1833                plt.savefig(dir_destino+"/figuras/"+caso+"/"+fig_ext[i]+"/"
    +\
1834                caso+"_Femeas-var_rand-ing."+fig_ext[i])
1835            plt.subplot(2, 1, 1)
1836            plt.title(P_Frase_F +" (k="+str(k)+")", fontsize=14)
1837            plt.subplot(2, 1, 2)
1838            plt.title(P_Frase_I +" (k="+str(k)+")", fontsize=14)
1839            plt.xlabel(P_Label_X, fontsize=14)
1840            for i in range(0, len(fig_ext)):
1841                plt.savefig(dir_destino+"/figuras/"+caso+"/"+fig_ext[i]+"/"
    +\
1842                caso+"_Femeas-var_rand."+fig_ext[i])
1843
1844            plt.close()
1845
1846            fig = fig+1
1847            plt.figure(fig)
1848            plt.subplot(2, 1, 1)
1849            for j in range(0, contador):
1850                if j==0:
1851                    plt.plot(t_vet[j,], M_vet[j,],  "-o",color='black',
    markevery=stride)
1852                else:
1853                    plt.plot(t_vet[j,], M_vet[j,], linewidth=3,linestyle =
    estilo_linha[j])
1854            plt.legend(legenda)
1855            if grid > 0:
1856                plt.grid()
1857            plt.title(I_Frase_M +" (k="+str(k)+")", fontsize=14)
1858            plt.ylabel('M', fontsize=14)
1859
1860            ax = plt.gca()
1861            for axis in ['top', 'bottom', 'left', 'right']:
```

```
1862                 ax.spines[axis].set_linewidth(2)
1863             for tick in ax.xaxis.get_major_ticks():
1864                 tick.label.set_fontsize(14)
1865 
1866             plt.subplot(2, 1, 2)
1867             for j in range(0, contador):
1868                 if j==0:
1869                     plt.plot(t_vet[j,], S_vet[j,],  "-o",color='black',
    markevery=stride)
1870                 else:
1871                     plt.plot(t_vet[j,], S_vet[j,], linewidth=3,linestyle =
    estilo_linha[j])
1872             plt.legend(legenda)
1873             if grid > 0:
1874                 plt.grid()
1875             plt.title(I_Frase_S +" (k="+str(k)+")", fontsize=14)
1876             plt.ylabel('S', fontsize=14)
1877             plt.xlabel(I_Label_X, fontsize=14)
1878 
1879             ax = plt.gca()
1880             for axis in ['top', 'bottom', 'left', 'right']:
1881                 ax.spines[axis].set_linewidth(2)
1882             for tick in ax.xaxis.get_major_ticks():
1883                 tick.label.set_fontsize(14)
1884 
1885             for i in range(0, len(fig_ext)):
1886                 plt.savefig(dir_destino+"/figuras/"+caso+"/"+fig_ext[i]+"/"
    +\
1887                 caso+"_Machos-var_rand-ing."+fig_ext[i])
1888 
1889             plt.subplot(2, 1, 1)
1890             plt.title(P_Frase_M +" (k="+str(k)+")", fontsize=14)
1891             plt.subplot(2, 1, 2)
1892             plt.title(P_Frase_S +" (k="+str(k)+")", fontsize=14)
1893             plt.xlabel(P_Label_X, fontsize=14)
1894             for i in range(0, len(fig_ext)):
1895                 plt.savefig(dir_destino+"/figuras/"+caso+"/"+fig_ext[i]+"/"
    +\
1896                 caso+"_Machos-var_rand."+fig_ext[i])
1897 
1898             plt.close()
1899 
1900         # plt.show();
1901     return fig
1902 
1903 def analisa_k(k_vet, d, sim, analise):
1904     dir_path = os.environ['USERCODEDIR']+'/resultados/jm/dengue/analise/'
1905     os.system('mkdir '+dir_path)
1906     dir_path = os.environ['USERCODEDIR']+'/resultados/jm/dengue/analise/
    param_k/'
1907     os.system('mkdir '+dir_path)
1908 
1909     global caso
1910     global ci
1911     global inicio_analise
1912 
1913     model_param, model_opt_param, param = aplic_param_analise(d, sim)
1914 
```

```
1915     Lc = param['Lc']
1916     Ls = param['Ls']
1917     Lu = param['Lu']
1918     LF = param['LF']
1919
1920     simula = "SIM"+str(sim)+"/"
1921     dir_path=dir_path+simula
1922     os.system('mkdir '+dir_path)
1923
1924     dir_path = dir_path+"dias_"+str(d)+"/"
1925     os.system('mkdir '+dir_path)
1926
1927     dir_path=dir_path+"/analise_"+str(analise)+"/"
1928     os.system('rm -R '+dir_path)
1929     os.system('mkdir '+dir_path)
1930     os.system('mkdir '+dir_path+"/figuras/")
1931
1932     dados_dict_casos = [[dict() for x in range(len(k_vet))] for y in range(
         len(Lc))]
1933     dados_validos = [[int() for x in range(len(k_vet))] for y in range(len(
         Lc))]
1934
1935     file_log = "log_analise_k"
1936     arquivo_nome = file_log+'.txt'
1937     arquivo_path = dir_path+arquivo_nome
1938     if os.path.isfile(arquivo_path):
1939         os.remove(arquivo_path)
1940
1941     f = open(arquivo_path, 'w+')
1942
1943     inicio_analise = datetime.now()
1944     print "Início da analise", str(inicio_analise)
1945     print "Arquivo de log = ", arquivo_path
1946
1947     print >> f, "Início do analise:", str(inicio_analise)
1948     print >> f, "Arquivo de log = ", arquivo_path
1949
1950     print >> f, "Análise da Simulação =",simula
1951     print "Análise da Simulação =",simula
1952
1953     cont=0
1954     indice_k = 0
1955     diretorio = True
1956     for n in range(0, len(k_vet)):
1957         k = k_vet[n]
1958         indice_k = indice_k + 1
1959         param1 = analise_param(k, d, sim)
1960         dir_origem = param1[0]
1961         fig_ext = param1[2]
1962
1963         print "k=", str(k)
1964         print "d=", str(d)
1965
1966         print >> f, "k=", str(k)
1967         print >> f, "d=", str(d)
1968
1969         cont = 0
1970
```

```
1971        fig = 1
1972        I_Frases, P_Frases = language()
1973
1974        indice_Ls = 0
1975        for j in range(0, len(Ls)):
1976        #
    ***********************************************************************
1977        # Início da otimização
1978        # Valores realmente utilizados sao definidos na rotina
    def_parametros()
1979            indice_Ls = indice_Ls + 1
1980            solver = Ls[j]
1981            indice_Lc = 0
1982
1983            for i in range(0, len(Lc)):
1984                indice_Lc = indice_Lc + 1
1985                ci = Lc[i]
1986                ui = Lu[i]
1987                LFI = LF[i]
1988                t_time = LFI[0]
1989                F_time = LFI[1]
1990
1991                caso = solver+"_caso_"+str(i+1)
1992
1993                print  "caso=", caso
1994                print  >> f, "caso=", caso
1995
1996                if diretorio == True:
1997
1998
1999                    os.system('mkdir '+dir_path+"/figuras/"+caso)
2000
2001                    for p in range(0, len(fig_ext)):
2002                        os.system('mkdir '+dir_path+"/figuras/"+caso+"/"+\
2003                            fig_ext[p])
2004
2005                param = [I_Frases,
2006                         P_Frases,
2007                         dir_path,
2008                         fig_ext,
2009                         d]
2010
2011                arq_opt_mat = "mat/"+caso+"_res_obj_opt.mat"
2012                try:
2013                    res = mat4py.loadmat(dir_origem+arq_opt_mat)
2014                    print "Carregou arquivo .mat"
2015                    dados_dict_casos[indice_Lc-1][indice_k-1] = res
2016                    dados_validos[indice_Lc-1][indice_k-1] = 1
2017
2018                    cont=cont+1
2019
2020                except Exception as e:
2021                    dados_dict_casos[indice_Lc-1][indice_k-1] = None
2022                    dados_validos[indice_Lc-1][indice_k-1]=0
2023                    print "Rotina de Análise"
2024                    print "Recebeu exception no "+caso
2025                    print "Não encontrou arquivo .mat com variáveis para:"
2026                    print "k=", str(k)
```

```
2027                    print "d=", str(d)
2028                    print "O IPOPT não convergiu para estas condições."
2029                    print e.__doc__
2030                    print e.message
2031
2032                    print >> f, "Rotina de Análise"
2033                    print >> f, "Recebeu exception no "+caso
2034                    print >> f, "Não encontrou arquivo .mat com variáveis
    para:"
2035                    print >> f, "k=", str(k)
2036                    print >> f, "d=", str(d)
2037                    print >> f, "O IPOPT não convergiu para estas condições
    ."
2038                    print >> f, e.__doc__
2039                    print >> f, e.message
2040
2041        diretorio = False
2042
2043    print "k_vet =  "+str(k_vet)
2044    print >> f, "k_vet =  "+str(k_vet)
2045    try:
2046        graficos_analise_k(res,
2047                    f,
2048                    k_vet,
2049                    Ls,
2050                    Lc,
2051                    Lu,
2052                    dados_dict_casos,
2053                    dados_validos,
2054                    fig,
2055                    param)
2056
2057    except Exception as e:
2058        print "Rotina de Análise"
2059        print "Recebeu exception no "+caso
2060        print "Não gerou gráficos:"
2061        print e.__doc__
2062        print e.message
2063
2064        print >> f, "Rotina de Análise"
2065        print >> f, "Recebeu exception no "+caso
2066        print >> f, "Não gerou gráficos:"
2067
2068        print >> f, e.__doc__
2069        print >> f, e.message
2070
2071
2072    data2 = datetime.now()
2073    delta= data2-inicio_analise
2074    print  "Tempo de analise = ", str(delta)
2075    print  "Segundos = ", delta.total_seconds()
2076
2077    print >> f, "Tempo de analise = ", str(delta)
2078    print >> f, "Segundos = ", delta.total_seconds()
2079    f.close()
2080
2081    # Fim da rotina  analisa_k
2082
```

```
2083 def analisa_IPOPT(k_vet, d, sim, analise):
2084     dir_path = os.environ['USERCODEDIR']+'/resultados/jm/dengue/analise/'
2085     os.system('mkdir '+dir_path)
2086     dir_path = os.environ['USERCODEDIR']+'/resultados/jm/dengue/analise/
         saidas_IPOPT/'
2087     os.system('mkdir '+dir_path)
2088 
2089     global caso
2090     global ci
2091     global inicio_analise
2092 
2093     model_param, model_opt_param, param = aplic_param_analise(d, sim)
2094 
2095     Lc = param['Lc']
2096     Ls = param['Ls']
2097     Lu = param['Lu']
2098 
2099     simula = "SIM"+str(sim)+"/"
2100     dir_path=dir_path+simula
2101     os.system('mkdir '+dir_path)
2102 
2103     dir_path = dir_path+"dias_"+str(d)+"/"
2104     os.system('mkdir '+dir_path)
2105 
2106     dir_path=dir_path+"/analise_"+str(analise)+"/"
2107     os.system('rm -R '+dir_path)
2108     os.system('mkdir '+dir_path)
2109 
2110     #possiveis resolvedores lineares:
2111            # rotinas HSL: 'ma27', 'ma57', 'ma77', 'ma86' e 'ma97'
2112            # default 'mumps'
2113            # se configurar as libs: 'pardiso', 'wsmp' e custom
2114     dados_dict_casos = [[dict() for x in range(len(k_vet))] for y in range(
         len(Lc))]
2115     dados_validos = [[int() for x in range(len(k_vet))] for y in range(len(
         Lc))]
2116 
2117     file_log = "log_analise_IPOPT"
2118     arquivo_nome = file_log+'.txt'
2119     arquivo_path = dir_path+arquivo_nome
2120     if os.path.isfile(arquivo_path):
2121         os.remove(arquivo_path)
2122 
2123     f = open(arquivo_path, 'w+')
2124 
2125     inicio_analise = datetime.now()
2126     print  "Início da analise IPOPT:", str(inicio_analise)
2127     print  "Arquivo de log = ", arquivo_path
2128 
2129     print  >> f, "Início do analise IPOPT:", str(inicio_analise)
2130     print  >> f, "Arquivo de log = ", arquivo_path
2131 
2132     print >> f, "Análise da Simulação =",simula
2133     print "Análise da Simulação =",simula
2134 
2135 
2136     cont=0
2137     indice_k = 0
```

```
2138     diretorio = True
2139     for n in range(0, len(k_vet)):
2140         k = k_vet[n]
2141         indice_k = indice_k + 1
2142         param1 = analise_param(k, d, sim)
2143         dir_origem = param1[0]
2144 
2145         fig_ext = param1[2]
2146 
2147         print  "k=", str(k)
2148         print  "d=", str(d)
2149 
2150         print  >> f, "k=", str(k)
2151         print  >> f, "d=", str(d)
2152 
2153         cont = 0
2154 
2155         fig = 1
2156         I_Frases, P_Frases = language()
2157 
2158         indice_Ls = 0
2159         for j in range(0, len(Ls)):
2160         #
     ********************************************************************
2161         # Início da otimização
2162         # Valores realmente utilizados sao definidos na rotina
     def_parametros()
2163             indice_Ls = indice_Ls + 1
2164             solver = Ls[j]
2165             indice_Lc = 0
2166 
2167             f_caso = []
2168 
2169             for i in range(0, len(Lc)):
2170                 indice_Lc = indice_Lc + 1
2171                 ci = Lc[i]
2172 
2173                 caso = solver+"_caso_"+str(i+1)
2174 
2175                 print  "caso=", caso
2176                 print  >> f, "caso=", caso
2177                 if diretorio == True:
2178                     os.system('mkdir '+dir_path+"/"+caso)
2179 
2180                 file_log = "log_analise_IPOPT_"+caso
2181                 arquivo_nome = file_log+'.txt'
2182                 arquivo_path = dir_path+"/"+caso+"/"+arquivo_nome
2183                 if os.path.isfile(arquivo_path):
2184                     os.remove(arquivo_path)
2185 
2186                 f_out = open(arquivo_path, 'w+')
2187                 f_caso.append(f_out)
2188 
2189                 param = [I_Frases,
2190                          P_Frases,
2191                          dir_path,
2192                          fig_ext,
2193                          d]
```

```
2194
2195                arq_est_mat = "mat/"+caso+"_res_est_opt.mat"
2196                try:
2197                    res = mat4py.loadmat(dir_origem+arq_est_mat)
2198                    print "Carregou arquivo .mat"
2199                    dados_dict_casos[indice_Lc-1][indice_k-1] = res
2200                    dados_validos[indice_Lc-1][indice_k-1] = 1
2201
2202                    cont=cont+1
2203
2204                except Exception as e:
2205                    dados_dict_casos[indice_Lc-1][indice_k-1] = None
2206                    dados_validos[indice_Lc-1][indice_k-1]=0
2207                    print "Rotina de Análise IPOPT"
2208                    print "Recebeu exception no "+caso
2209                    print "Não encontrou arquivo .mat com variáveis para:"
2210                    print "k=", str(k)
2211                    print "d=", str(d)
2212                    print "O IPOPT não convergiu para estas condições."
2213                    print e.__doc__
2214                    print e.message
2215
2216                    print >> f, "Rotina de Análise IPOPT"
2217                    print >> f, "Recebeu exception no "+caso
2218                    print >> f, "Não encontrou arquivo .mat com variáveis
    para:"
2219                    print >> f, "k=", str(k)
2220                    print >> f, "d=", str(d)
2221                    print >> f, "O IPOPT não convergiu para estas condições
    ."
2222                    print >> f, e.__doc__
2223                    print >> f, e.message
2224
2225        diretorio = False
2226
2227    print "k_vet =  "+str(k_vet)
2228    print >> f, "k_vet =  "+str(k_vet)
2229    try:
2230        graficos_analise_IPOPT(res,
2231                    f_caso,
2232                    k_vet,
2233                    Ls,
2234                    Lc,
2235                    Lu,
2236                    dados_dict_casos,
2237                    dados_validos,
2238                    fig,
2239                    param)
2240
2241    except Exception as e:
2242        print "Rotina de Análise IPOPT - gráficos"
2243        print "Recebeu exception no "+caso
2244        print "Não gerou gráficos:"
2245        print e.__doc__
2246        print e.message
2247
2248        print >> f, "Rotina de Análise - gráficos"
2249        print >> f, "Recebeu exception no "+caso
```

```
2250            print >> f, "Não gerou gráficos:"
2251
2252            print >> f, e.__doc__
2253            print >> f, e.message
2254
2255
2256        data2 = datetime.now()
2257        delta= data2-inicio_analise
2258        print  "Tempo de analise = ", str(delta)
2259        print  "Segundos = ", delta.total_seconds()
2260
2261        print >> f, "Tempo de analise = ", str(delta)
2262        print >> f, "Segundos = ", delta.total_seconds()
2263        f.close()
2264
2265        # Fim da rotina  analisa_IPOPT
2266
2267    def variacional(k, d, sim, variacao):
2268        dir_path = os.environ['USERCODEDIR']+'/resultados/jm/dengue/analise/'
2269        os.system('mkdir '+dir_path)
2270        dir_path = os.environ['USERCODEDIR']+'/resultados/jm/dengue/analise/
        variacional/'
2271        os.system('mkdir '+dir_path)
2272
2273        global caso
2274        global ci
2275        global inicio_variacao
2276        u1_m=10
2277        u2_m=10
2278
2279        dias=d
2280
2281        # os#próximos parâmetros devem ser os mesmos da otimização
2282        if sim == 1:
2283            simula = "SIM1/"
2284            LuCtrFixo = [[[0, 0.1], [0, 0.02]],
2285                         [[0, 0.08], [0, 0.01]],
2286                         [[0, 0.07], [0, 0.05]],
2287                         [[0, 0.04], [0, 0.08]]]
2288
2289            LuCtrCont = [[[0, u1_m], [0, u2_m]],
2290                         [[0, u1_m], [0, u2_m]],
2291                         [[0, u1_m], [0, u2_m]],
2292                         [[0, u1_m], [0, u2_m]]]
2293
2294            LuCtrDisc = [[[0, u1_m], [0, u2_m]],
2295                         [[0, u1_m], [0, u2_m]],
2296                         [[0, u1_m], [0, u2_m]],
2297                         [[0, u1_m], [0, u2_m]]]
2298
2299
2300            LF = [[0, 0.1],  # LF = [t_time, F_time]
2301                  [0, 0.1],  # Tem que ter muito cuidado aqui
2302                  [0, 0.1],  # Porque restricoes impossíveis
2303                  [0, 0.1]]  # de alcancar tornam o problema
2304                             # insolúvel ao IPOPT e trava
2305               #  [60, 0.1]# o script python dando erro
2306                           # Se for t_time = 0 nao tem efeito na otimização
```

```
                    # Se for t_time = 120 representa restrição final para F

        Lc = [[1, 10, 100, 1],
              [1, 10, 1, 1],
              [1, 1, 1, 1],
              [3, 1, 1, 1]]
        if dias < 5:
            Lu = LuCtrCont
        elif dias > 100:
            Lu = LuCtrFixo
        else:
            Lu = LuCtrDisc

    elif sim == 2:
        simula = "SIM2/"
        Lc = [[1, 1, 1, 1]]
        Lu = [[[0, u1_m], [0, u2_m]]]
        LF = [[0, 0.1]]

    elif sim == 3:
        simula = "SIM3/"
        LuCtrFixo = [[[0, 0.1], [0, 0.02]],
                     [[0, 0.08], [0, 0.01]],
                     [[0, 0.07], [0, 0.05]],
                     [[0, 0.04], [0, 0.08]]]

        LuCtrCont = [[[0, u1_m], [0, u2_m]],
                     [[0, u1_m], [0, u2_m]],
                     [[0, u1_m], [0, u2_m]],
                     [[0, u1_m], [0, u2_m]]]

        LuCtrDisc = [[[0, u1_m], [0, u2_m]],
                     [[0, u1_m], [0, u2_m]],
                     [[0, u1_m], [0, u2_m]],
                     [[0, u1_m], [0, u2_m]]]


        LF = [[0, 0.1],  # LF = [t_time, F_time]
              [0, 0.1],  # Tem que ter muito cuidado aqui
              [0, 0.1],  # Porque restricoes impossíveis
              [0, 0.1]]  # de alcancar tornam o problema
                         # insolúvel ao IPOPT e trava
           #  [60, 0.1]# o script python dando erro
                       # Se for t_time = 0 nao tem efeito na otimização
                       # Se for t_time = 120 representa restrição final para F

        Lc = [[1, 10, 100, 1],
              [1, 10, 1, 1],
              [1, 1, 1, 1],
              [3, 1, 1, 1]]
        if dias < 5:
            Lu = LuCtrCont
        elif dias > 100:
            Lu = LuCtrFixo
        else:
            Lu = LuCtrDisc
```

```
2363
2364    elif sim == 4:
2365        simula = "SIM4/"
2366        Lc = [[1, 1, 1, 1]]
2367        Lu = [[[0, u1_m], [0, u2_m]]]
2368        LF = [[0, 0.1]]
2369
2370    dir_path=dir_path+simula
2371    os.system('mkdir '+dir_path)
2372
2373    dir_path = dir_path+"dias_"+str(d)+"/"
2374    os.system('mkdir '+dir_path)
2375
2376    dir_path=dir_path+"/variacao_"+str(variacao)+"/"
2377    os.system('rm -R '+dir_path)
2378    os.system('mkdir '+dir_path)
2379    os.system('mkdir '+dir_path+"/figuras/")
2380    Ls = ['ma57']
2381    #possiveis resolvedores lineares:
2382           # rotinas HSL: 'ma27', 'ma57', 'ma77', 'ma86' e 'ma97'
2383           # default 'mumps'
2384           # se configurar as libs: 'pardiso', 'wsmp' e custom
2385
2386    file_log = "log_variacao"
2387    arquivo_nome = file_log+'.txt'
2388    arquivo_path = dir_path+arquivo_nome
2389    if os.path.isfile(arquivo_path):
2390        os.remove(arquivo_path)
2391
2392    f1 = open(arquivo_path, 'w+')
2393
2394    inicio_variacao = datetime.now()
2395    print "Início da estudo de variação", str(inicio_variacao)
2396    print "Arquivo de log = ", arquivo_path
2397
2398    print >> f1, "Início da estudo de variação", str(inicio_variacao)
2399    print >> f1, "Arquivo de log = ", arquivo_path
2400
2401    print >> f1, "Estudo de variação da Simulação =",simula
2402    print "Estudo de variação da Simulação =",simula
2403
2404    cont=0
2405    indice_k = 0
2406    diretorio = True
2407    param1 = analise_param(k, d, sim)
2408    dir_origem = param1[0]
2409    dir_destino = param1[1]
2410    fig_ext = param1[2]
2411    print "k=", str(k)
2412    print "d=", str(d)
2413
2414    print >> f1, "k=", str(k)
2415    print >> f1, "d=", str(d)
2416
2417    cont = 0
2418
2419    fig = 1
2420    I_Frases, P_Frases = language()
```

```
2421     param = [I_Frases,
2422              P_Frases,
2423              dir_path,
2424              fig_ext,
2425              k,
2426              d]
2427     indice_Ls = 0
2428     for j in range(0, len(Ls)):
2429         indice_Ls = indice_Ls + 1
2430         solver = Ls[j]
2431         indice_Lc = 0
2432 
2433         for i in range(0, len(Lc)):
2434             indice_Lc = indice_Lc + 1
2435             ci = Lc[i]
2436             ui = Lu[i]
2437             LFI = LF[i]
2438             t_time = LFI[0]
2439             F_time = LFI[1]
2440 
2441             caso = solver+"_caso_"+str(i+1)
2442 
2443             print  "caso=", caso
2444             print  >> f1, "caso=", caso
2445 
2446             os.system('mkdir '+dir_path+"/figuras/"+caso)
2447 
2448             for p in range(0, len(fig_ext)):
2449                 os.system('mkdir '+dir_path+"/figuras/"+caso+"/"+\
2450                     fig_ext[p])
2451 
2452             arq_obj_opt_mat = "mat/"+caso+"_res_obj_opt.mat"
2453             arq_est_opt_mat = "mat/"+caso+"_res_est_opt.mat"
2454             arq_model_otim_param_mat = "mat/"+caso+"_model_otim_param.mat"
2455             try:
2456                 res = mat4py.loadmat(dir_origem+arq_obj_opt_mat)
2457                 print "Carregou arquivo "+arq_obj_opt_mat
2458                 res_est = mat4py.loadmat(dir_origem+arq_est_opt_mat)
2459                 print "Carregou arquivo "+arq_est_opt_mat
2460                 res_model_param = mat4py.loadmat(dir_origem+
    arq_model_otim_param_mat)
2461                 print "Carregou arquivo "+arq_model_otim_param_mat
2462                 n_e = res_model_param['n_e']
2463                 n_cp = res_model_param['n_cp']
2464                 dados_validos = 1
2465 
2466             except Exception as e:
2467                 dados_validos = 0
2468                 print "Recebeu exception no "+caso
2469                 print "Não encontrou arquivo .mat com variáveis para:"
2470                 print "k=", str(k)
2471                 print "d=", str(d)
2472                 print "O IPOPT não convergiu para estas condições."
2473                 print e.__doc__
2474                 print e.message
2475 
2476                 print >> f1, "Recebeu exception no "+caso
2477                 print >> f1, "Não encontrou arquivo .mat com variáveis para
```

```
    :"
                print >> f1, "k=", str(k)
                print >> f1, "d=", str(d)
                print >> f1, "O IPOPT não convergiu para estas condições."
                print >> f1, e.__doc__
                print >> f1, e.message
            if dados_validos == 1:
                try:
                    fig = graficos_variacao_continua(f1,
                                                     res,
                                                     res_est,
                                                     n_e,
                                                     n_cp,
                                                     Ls,
                                                     Lc,
                                                     Lu,
                                                     fig,
                                                     param)

                    fig = graficos_variacao_randomica(f1,
                                                      res,
                                                      res_est,
                                                      n_e,
                                                      n_cp,
                                                      Ls,
                                                      Lc,
                                                      Lu,
                                                      fig,
                                                      param)

                except Exception as e:
                    print "Rotina do Estudo Variacional"
                    print "Recebeu exception no "+caso
                    print "Não gerou gráficos:"
                    print e.__doc__
                    print e.message

                    print >> f1,"Rotina do Estudo Variacional"
                    print >> f1, "Recebeu exception no "+caso
                    print >> f1, "Não gerou gráficos:"

                    print >> f1, e.__doc__
                    print >> f1, e.message


    data2 = datetime.now()
    delta= data2-inicio_variacao
    print  "Tempo do estudo de variação = ", str(delta)
    print  "Segundos = ", delta.total_seconds()

    print >> f1, "Tempo do estudo de variação = ", str(delta)
    print >> f1, "Segundos = ", delta.total_seconds()
    f1.close()

    # Fim da rotina  variacional

def processa_analise(k_seq, d_seq, sim):
    n = len(k_seq)/4
```

```
2535     cont = 1
2536     for d in d_seq:
2537         analisa_IPOPT(k_seq, d, sim,cont)
2538         cont = cont + 1
2539         indice=0
2540         for i in range(1, n+1):
2541             indice=(i-1)*4
2542             vetor=[k_seq[(indice)],k_seq[(indice+1)],k_seq[(indice+2)],
    k_seq[(indice+3)] ]
2543             analisa_k(vetor, d, sim, i)
2544             indice=i
2545 
2546         m=n*4
2547         diferenca=len(k_seq)-m
2548         indice=indice+1
2549         if diferenca == 3:
2550             vetor3=[k_seq[(m)],k_seq[(m+1)],k_seq[(m+2)]]
2551             analisa_k(vetor3, d, sim, indice)
2552         if diferenca == 2:
2553             vetor2=[k_seq[(m)],k_seq[(m+1)]]
2554             analisa_k(vetor2, d, sim, indice)
2555         if diferenca == 1:
2556             vetor1=[k_seq[(m)]]
2557             analisa_k(vetor1, d, sim, indice)
2558 
2559 #
    ********************************************************************************

2560 #Início do programa principal
2561 print "Analisa Copyright (C) 2015, Ranulfo Acir de Oliveira Resende"
2562 print "This program comes with ABSOLUTELY NO WARRANTY."
2563 print "This is free software, and you are welcome to redistribute it"
2564 print "under certain conditions."
2565 print "Read README.txt for details."
2566 print "Sugestions and corrections to improve the software are welcome:"
2567 print "acir_r@yahoo.com.br"
2568 
2569 inicio_progama = datetime.now()
2570 
2571 dir_path = os.environ['USERCODEDIR']+'/resultados/jm/dengue/analise/'
2572 os.system('mkdir '+dir_path)
2573 
2574 arquivo_1_path=dir_path+"Log_Geral.txt"
2575 
2576 if os.path.isfile(arquivo_1_path):
2577         os.remove(arquivo_1_path)
2578 f_1 = open(arquivo_1_path, 'w+')
2579 
2580 print  "Início do programa = ", str(inicio_progama)
2581 print >> f_1, "Início do programa = ", str(inicio_progama)
2582 
2583 caso="Início"
2584 grid = 1 # gráficos com grade
2585 # Variações para controle 1
2586 varia1 = 10 # variacao = +- 10%
2587 dc1 = 0.0   # variacao DC = +- 0.1.
2588 # Variações para controle 2
2589 varia2 = 10 # variacao = +- 10%
```

```
dc2 = 0.0   # variacao DC = +- 0.1.

sim = 1  # ulim=definidos nos casos
print  "Início dos casos da Análise da Simulação "+str(sim)     #Artigo DINCON
print >> f_1, "Início dos casos da Análise da Simulação "+str(sim) #Artigo DINCON
k_seq = 1,100
d_seq = 0,10,120

processa_analise(k_seq, d_seq, sim)

analise = 11
analisa_IPOPT(k_seq, 0, sim, analise)    #k_seq sem limite para esta funcao
                                         #d_seq max 1 para esta funcao
analise = 11
analisa_k(k_seq, 0, sim,  analise)   #k_seq max 4 para esta funcao
                                     #d_seq max 1 para esta funcao
analise = 100
k_seq=[100]
analisa_k(k_seq, 10, sim,  analise)   #k_seq max 4 para esta funcao
                                     #d_seq max 1 para esta funcao

sim = 2  #  ulim=10
print  "Início dos casos da Análise da Simulação "+str(sim)
print >> f_1, "Início dos casos da Análise da Simulação "+str(sim)
k_seq = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 25, 30, 35, 40, 45, 50, 60, 70,\
80, 90, 91, 92, 93, 94, 94.1, 94.2, 94.240,94.24002, 94.24005, 94.24010, 94.24025, \
94.24040, 94.24035, 94.24038, 94.24045, 94.24050, 94.2410, 94.2420, 94.2430, 94.2433,\
 94.2435,  94.2438, 94.2440, 94.2445,  94.2450, 94.250, 94.3, 94.4, 94.5, 94.6, 94.7, 94.8, 94.9, 95, 96, 97,\
98, 99, 100, 110, 120, 130, 140, 150, 200, 250, 300, 350, 400, 450, 500
d_seq = [0]
processa_analise(k_seq, d_seq, sim)

analise = 500
sim = 7
processa_analise(k_seq, d_seq, sim)
analisa_IPOPT(k_seq, 0, sim, analise)


sim = 8
k_seq = 94.2433, 94.2435, 94.2438
d_seq = [0]



d_seq = [0]
k_seq = 1,100
sim = 6
processa_analise(k_seq, d_seq, sim)

sim = 2
k_seq = 90, 94.242, 94.245, 100
analisa_k(k_seq, 0, sim, 500)
```

```
2642 k_seq1 = 90,94.2410, 94.2415, 94.2420
2643 analisa_k(k_seq1, 0, sim, 501)
2644 k_seq2 = 94.2440, 94.2450, 94.25, 100
2645 analisa_k(k_seq2, 0, sim, 502)
2646
2647 sim = 7
2648 k_seq = 90, 94.242, 94.245, 100
2649 analisa_k(k_seq, 0, sim, 500)
2650 k_seq1 = 90,94.2410, 94.2415, 94.2420
2651 analisa_k(k_seq1, 0, sim, 501)
2652 k_seq2 = 94.2440, 94.2450, 94.25, 100
2653 analisa_k(k_seq2, 0, sim, 502)
2654
2655 k = k_seq[0]
2656 d = 0
2657 sim = 2
2658 variacao = 1
2659 variacional(k, d, sim, variacao)
2660
2661 k = k_seq[1]
2662 d = 0
2663 sim = 2
2664 variacao = 2
2665 variacional(k, d, sim, variacao)
2666
2667 k = k_seq[2]
2668 d = 0
2669 sim = 2
2670 variacao = 3
2671 variacional(k, d, sim, variacao)
2672
2673 k = k_seq[3]
2674 d = 0
2675 sim = 2
2676 variacao = 4
2677 variacional(k, d, sim, variacao)
2678
2679
2680 data_fim = datetime.now()
2681 print  "Fim do programa = ", str(data_fim)
2682 print >> f_1,   "Fim do programa = ", str(data_fim)
2683 tempo = data_fim-inicio_progama
2684 print  "Tempo de processamento total= ", str(tempo)
2685 print  >> f_1,  "Tempo de processamento total= ", str(tempo)
2686
2687 f_1.close()
2688
2689 print "Analisa Copyright (C) 2015, Ranulfo Acir de Oliveira Resende"
2690 print "This program comes with ABSOLUTELY NO WARRANTY."
2691 print "This is free software, and you are welcome to redistribute it"
2692 print "under certain conditions."
2693 print "Read README.txt for details."
2694 print "Sugestions and corrections to improve the software are welcome:"
2695 print "acir_r@yahoo.com.br"
```

APÊNDICE B - ESTUDO VARIACIONAL - SIMULAÇÕES PARA K = 90

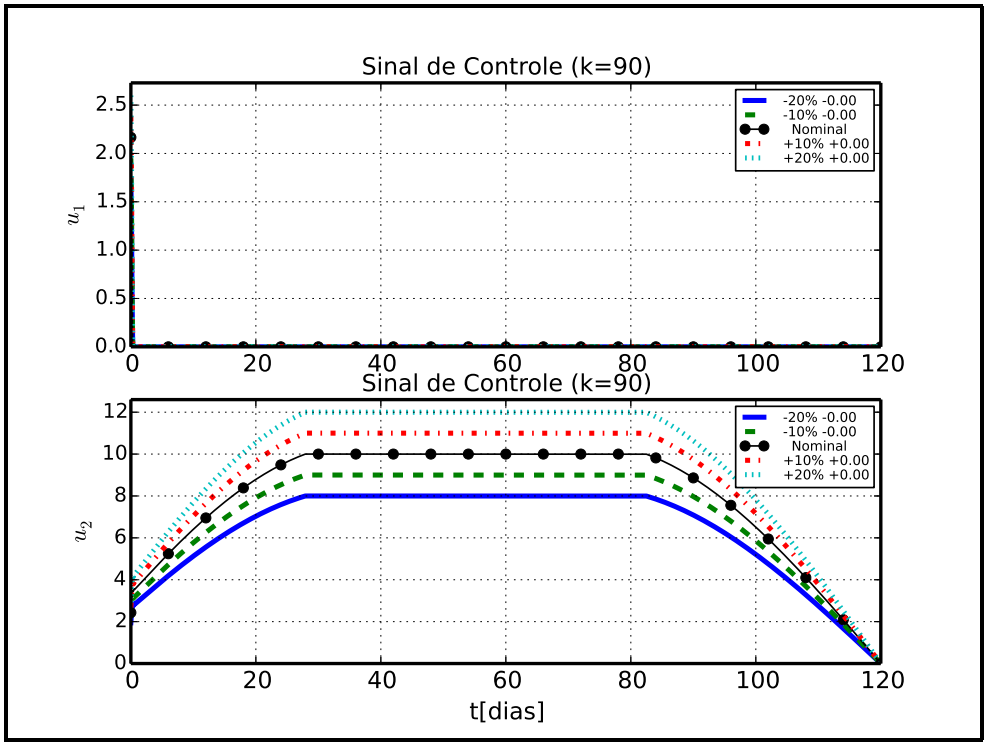

Figura 29: Controles - Variações Proporcionais e Fixas para k = 90. Fonte: o autor.

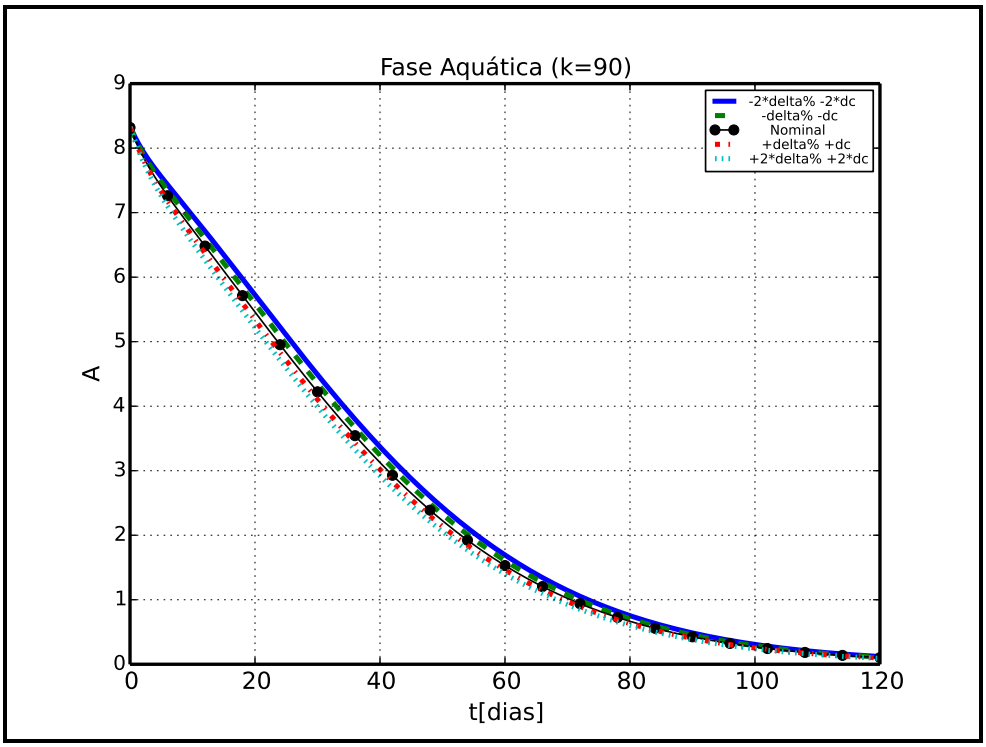

Figura 30: Fase Aquática - Variações Proporcionais e Fixas para k = 90. Fonte: o autor.

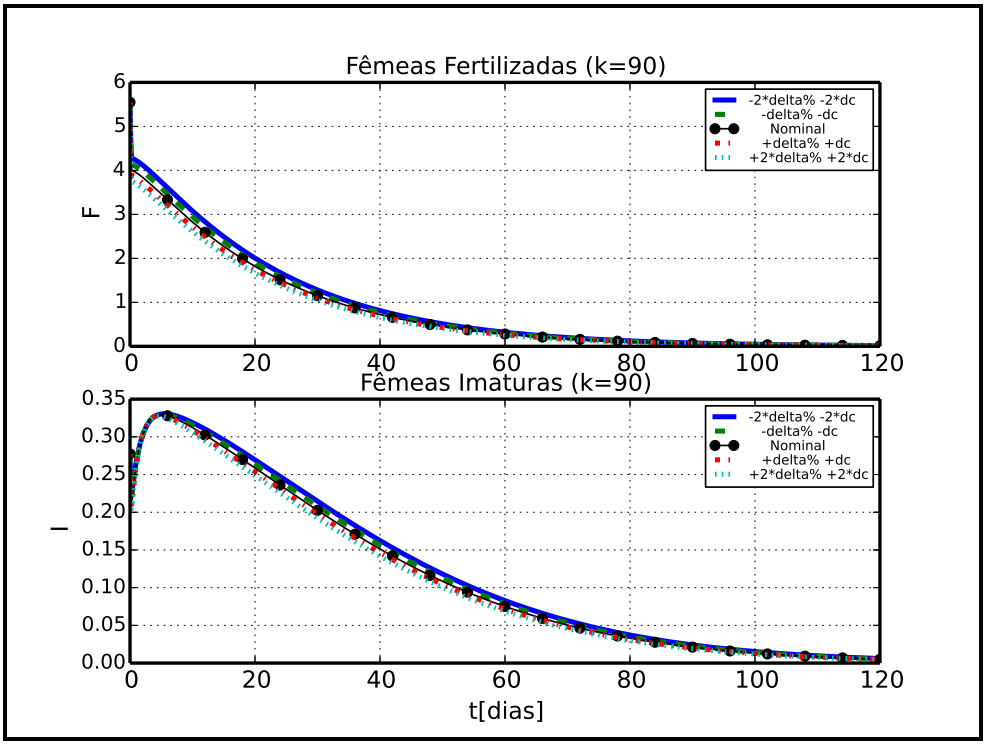

Figura 31: Fêmeas - Variações Proporcionais e Fixas para k = 90. Fonte: o autor.

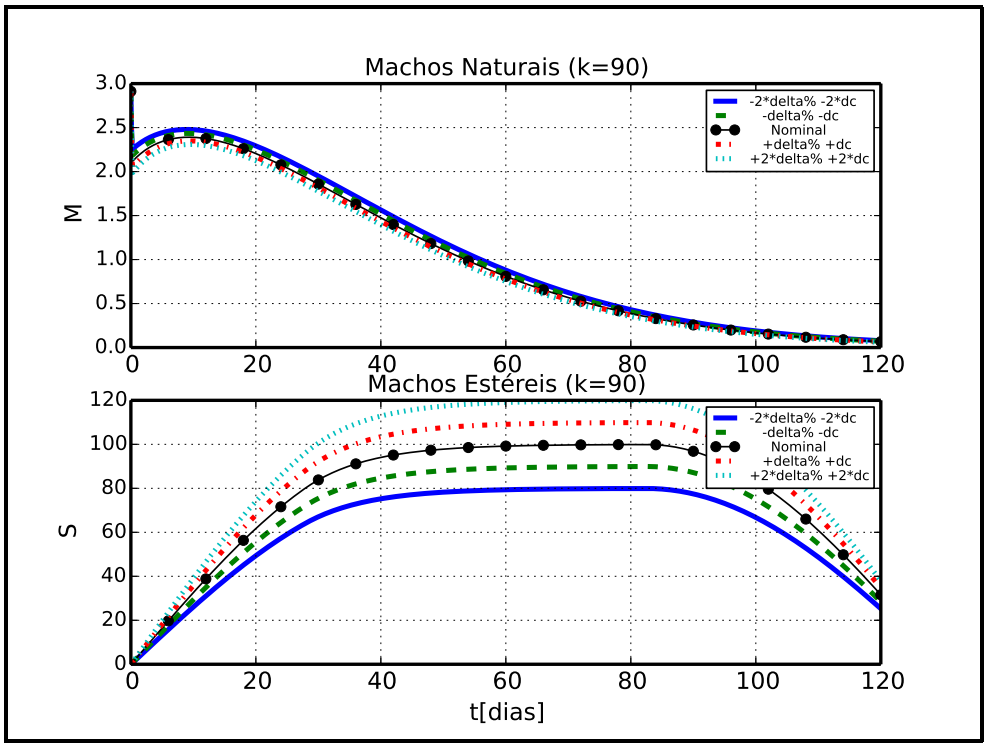

Figura 32: Machos - Variações Proporcionais e Fixas para k = 90. Fonte: o autor.

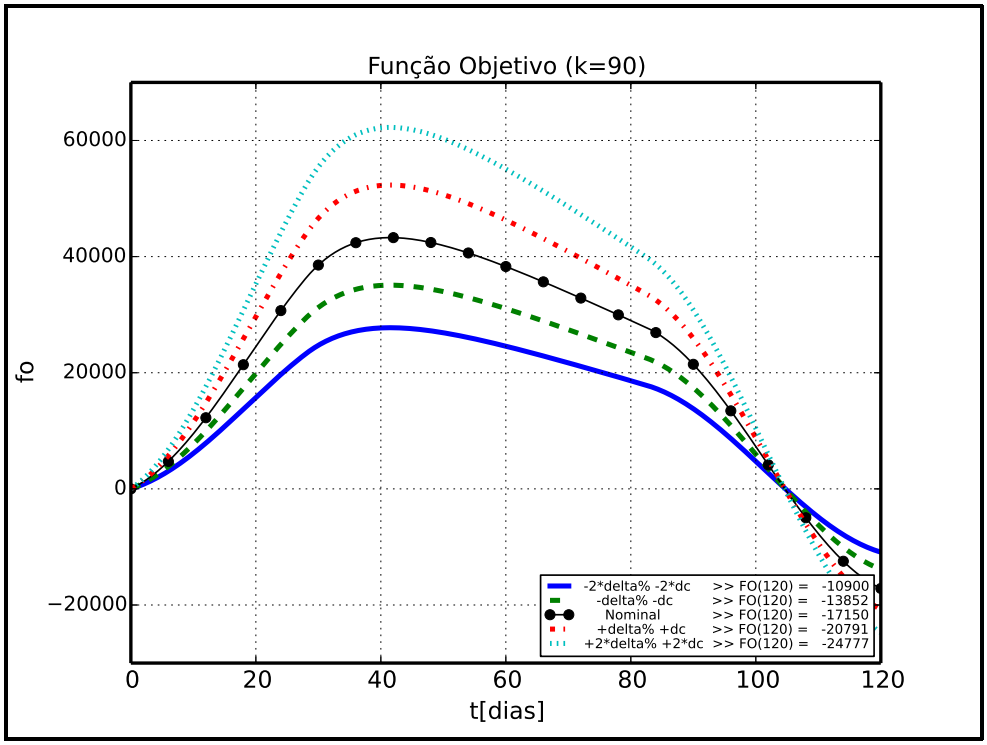

Figura 33: Função Objetivo - Variações Proporcionais e Fixas para k = 90. Fonte: o autor.

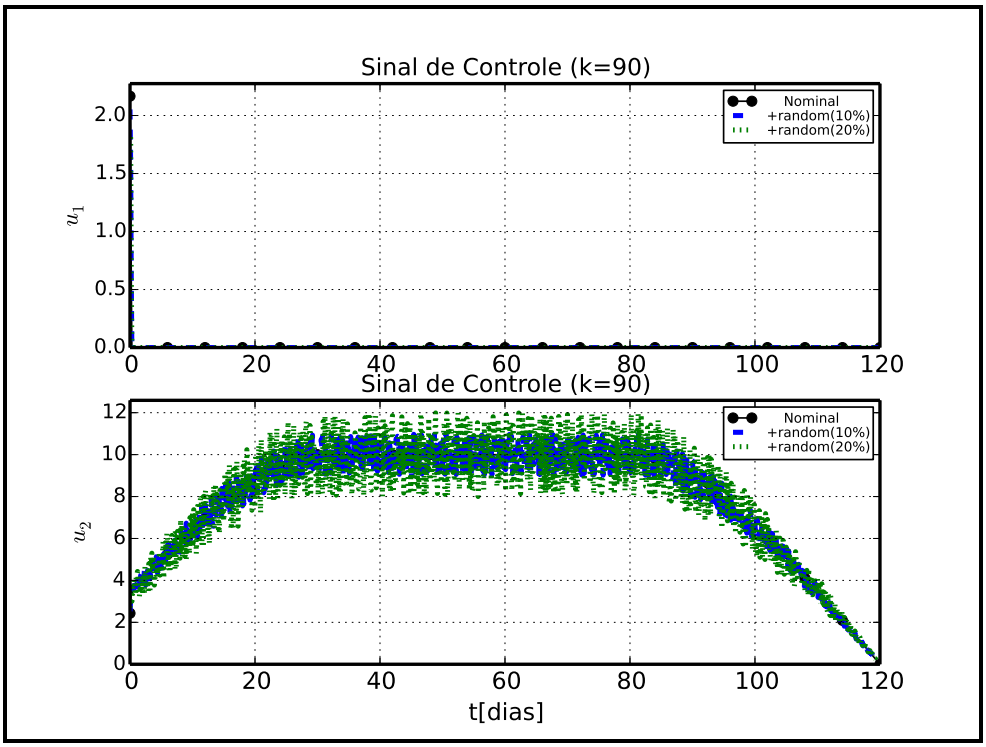

Figura 34: Controles - Variações Aleatórias para k = 90. Fonte: o autor.

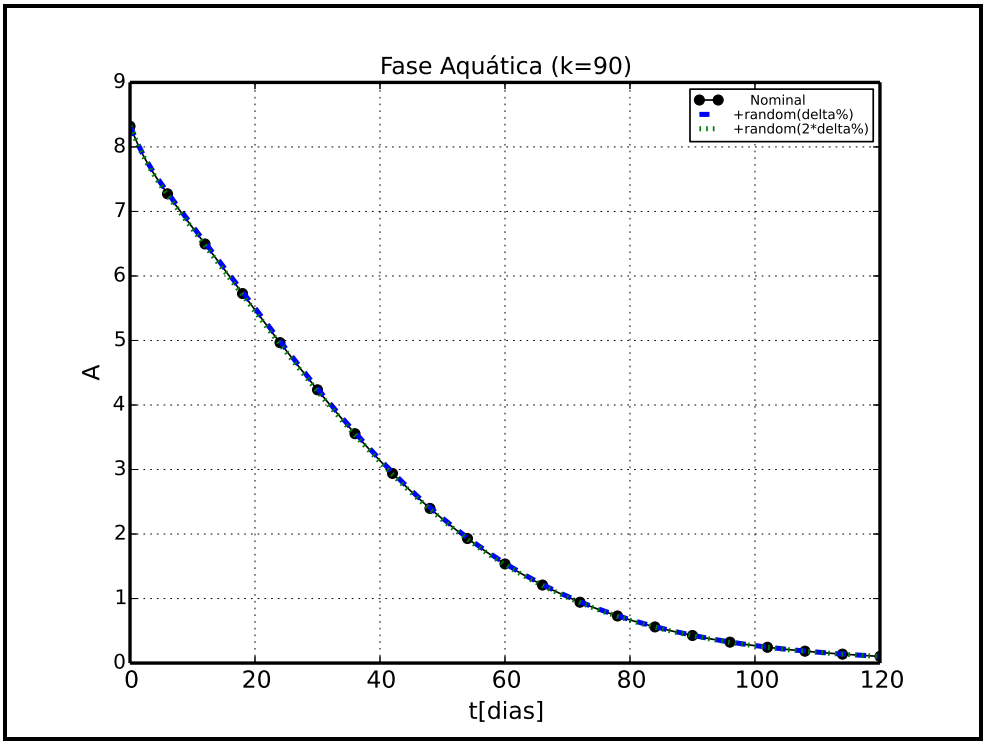

Figura 35: Fase Aquática - Variações Aleatórias para k = 90. Fonte: o autor.

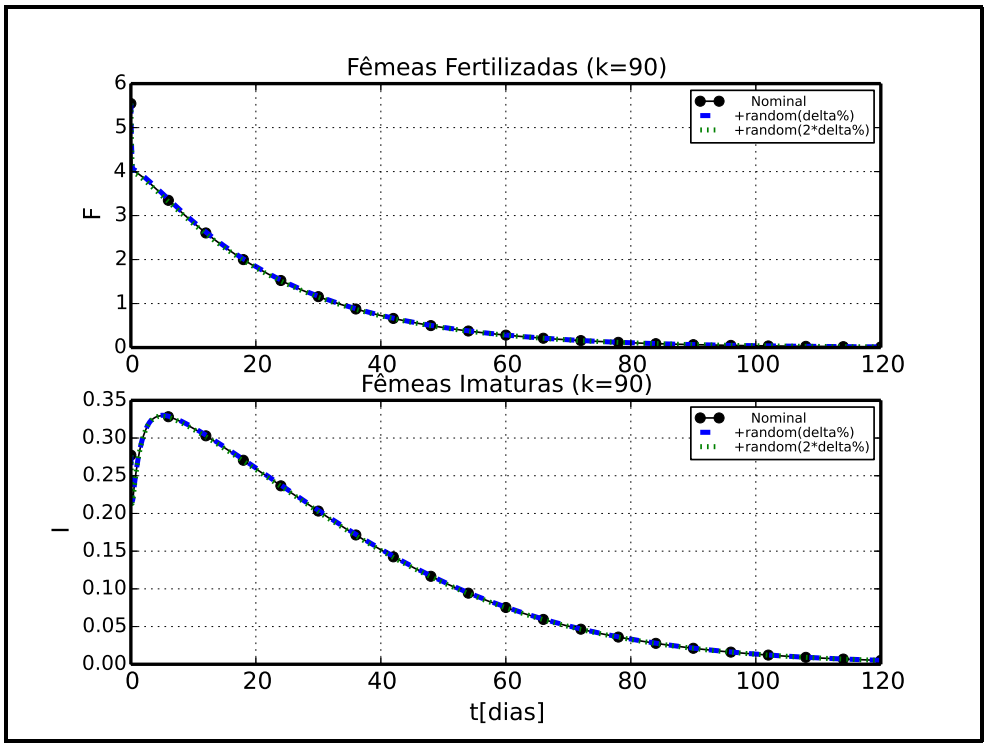

Figura 36: Fêmeas - Variações Aleatórias para k = 90. Fonte: o autor.

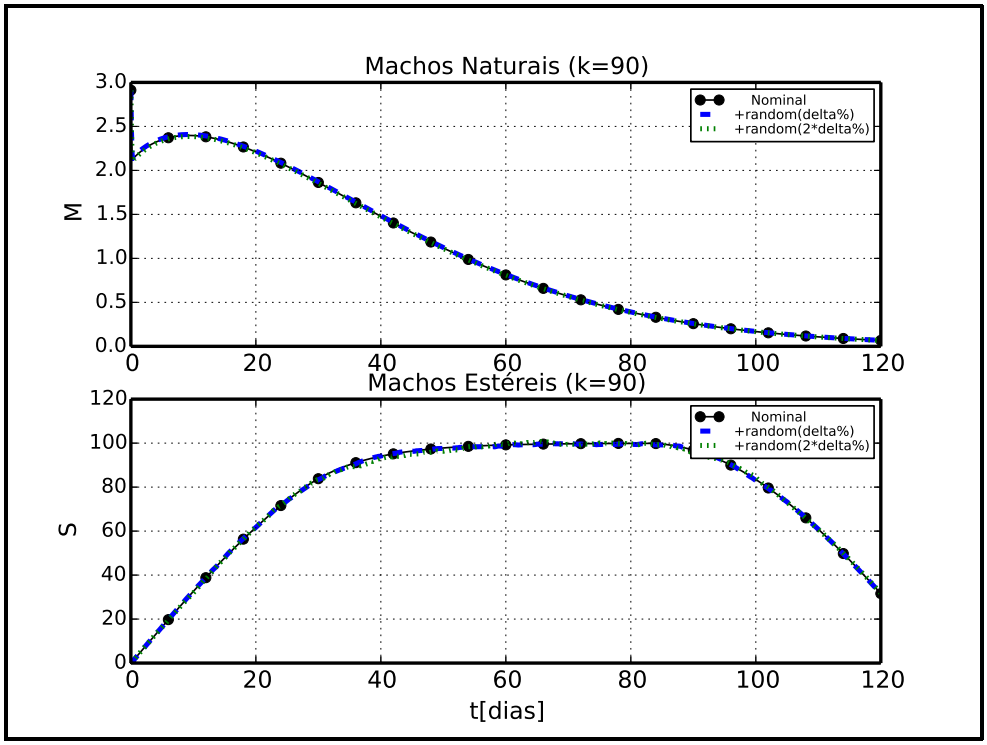

Figura 37: Machos - Variações Aleatórias para k = 90. Fonte: o autor.

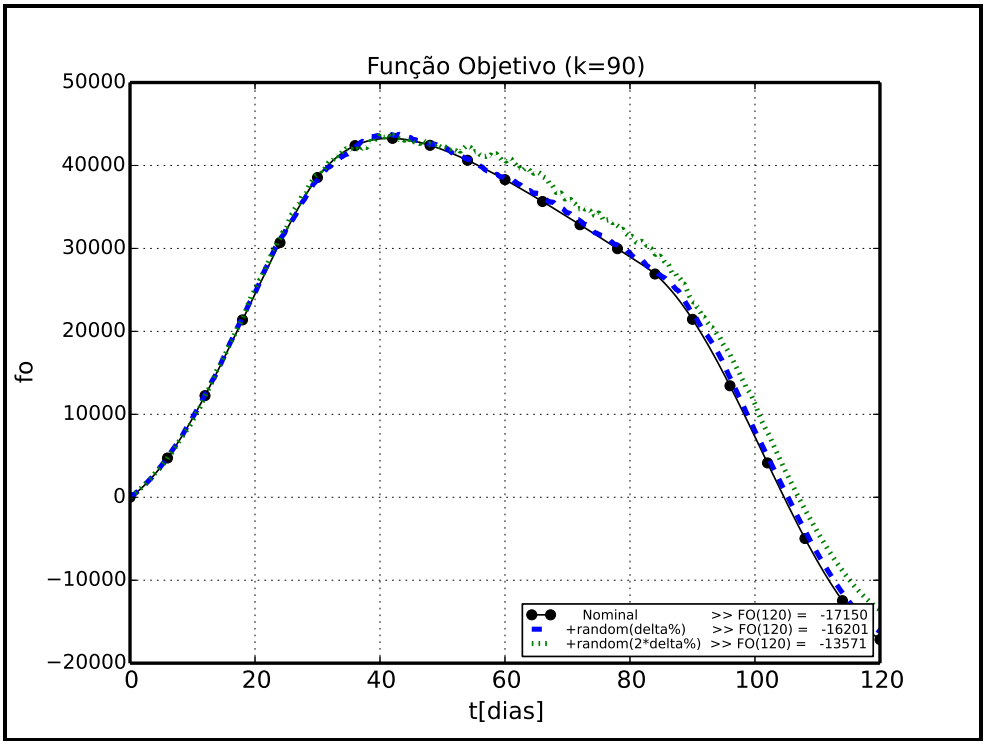

Figura 38: Função Objetivo - Variações Aleatórias para k = 90. Fonte: o autor.

APÊNDICE C - ESTUDO VARIACIONAL - SIMULAÇÕES PARA K = 94,242

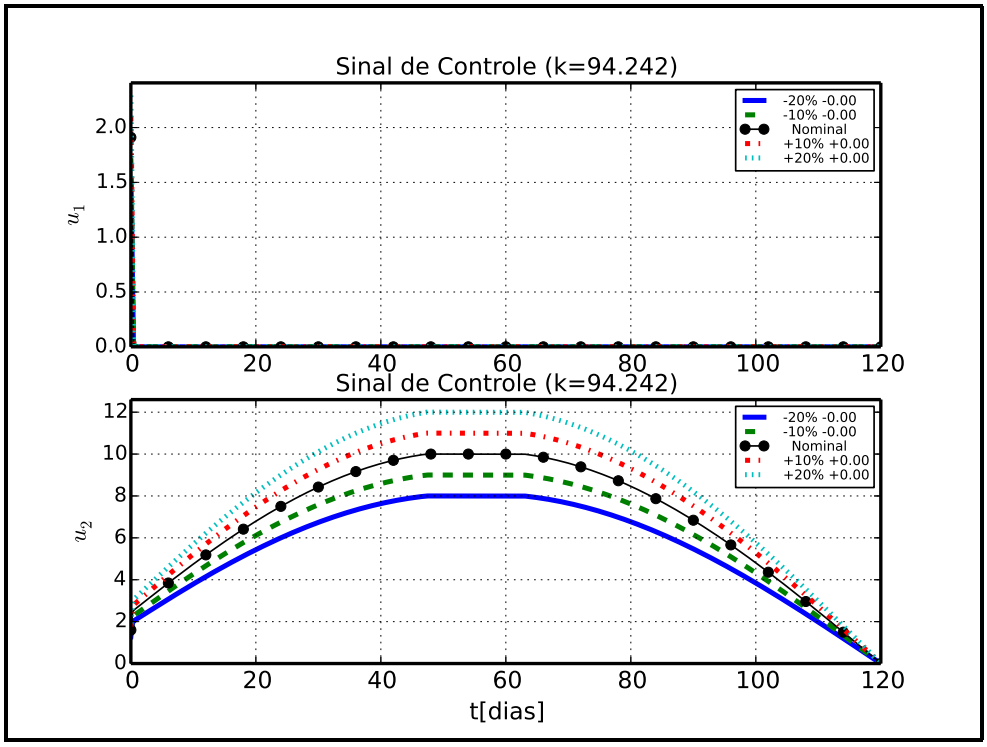

Figura 39: Controles - Variações Proporcionais e Fixas para k = 94,242. Fonte: o autor.

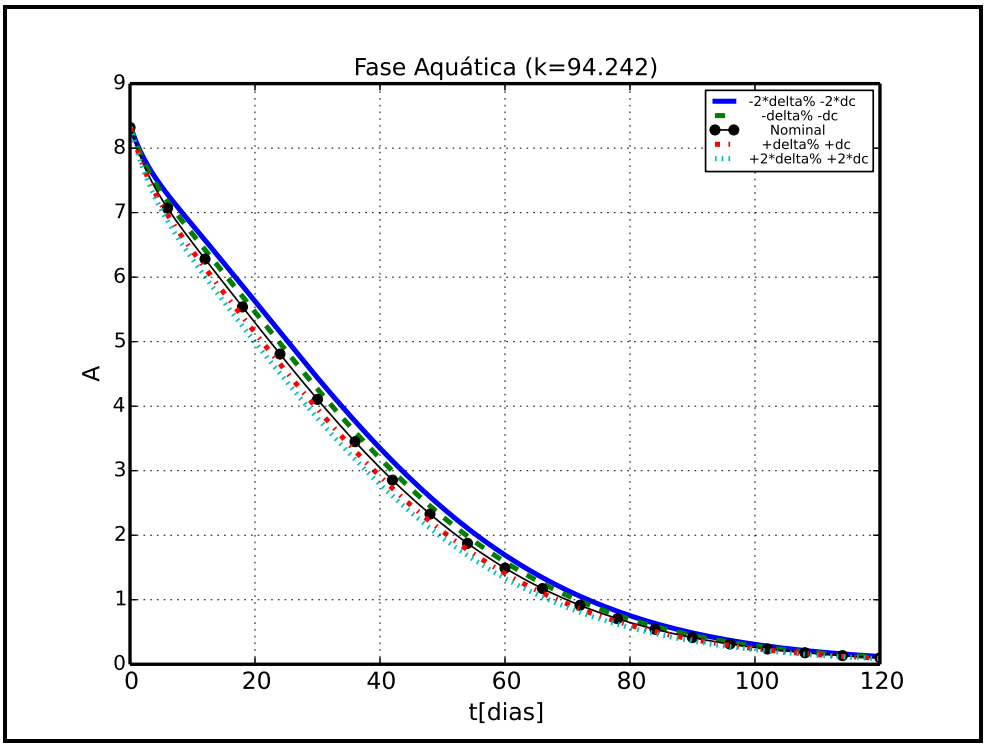

Figura 40: Fase Aquática - Variações Proporcionais e Fixas para k = 94,242. Fonte: o autor.

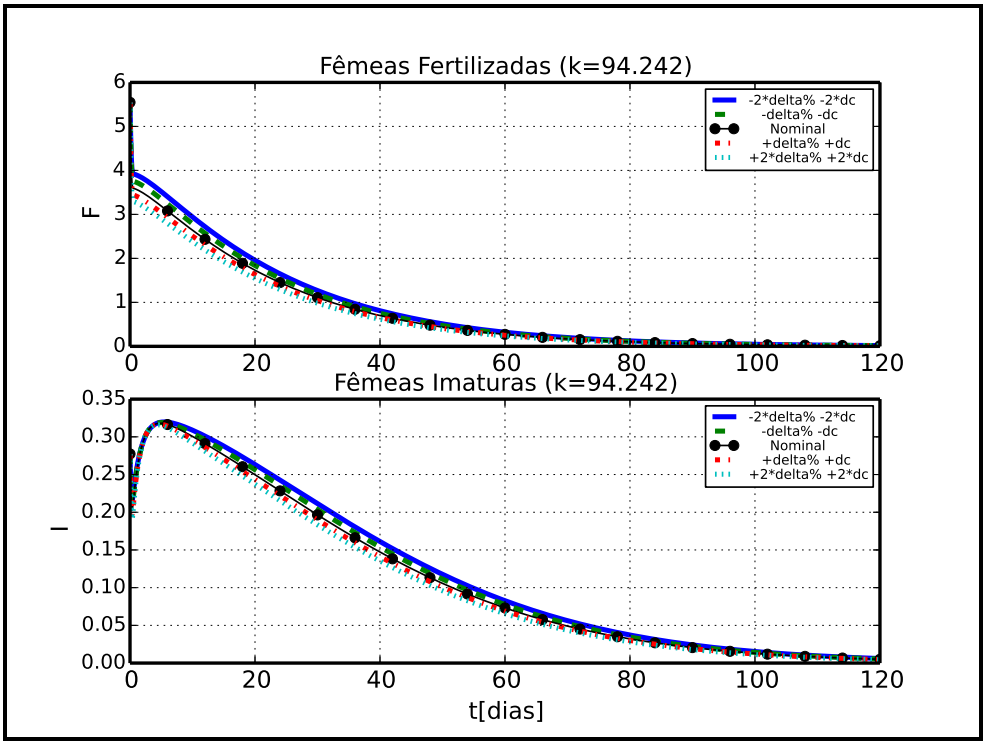

Figura 41: Fêmeas - Variações Proporcionais e Fixas para k = 94,242. Fonte: o autor.

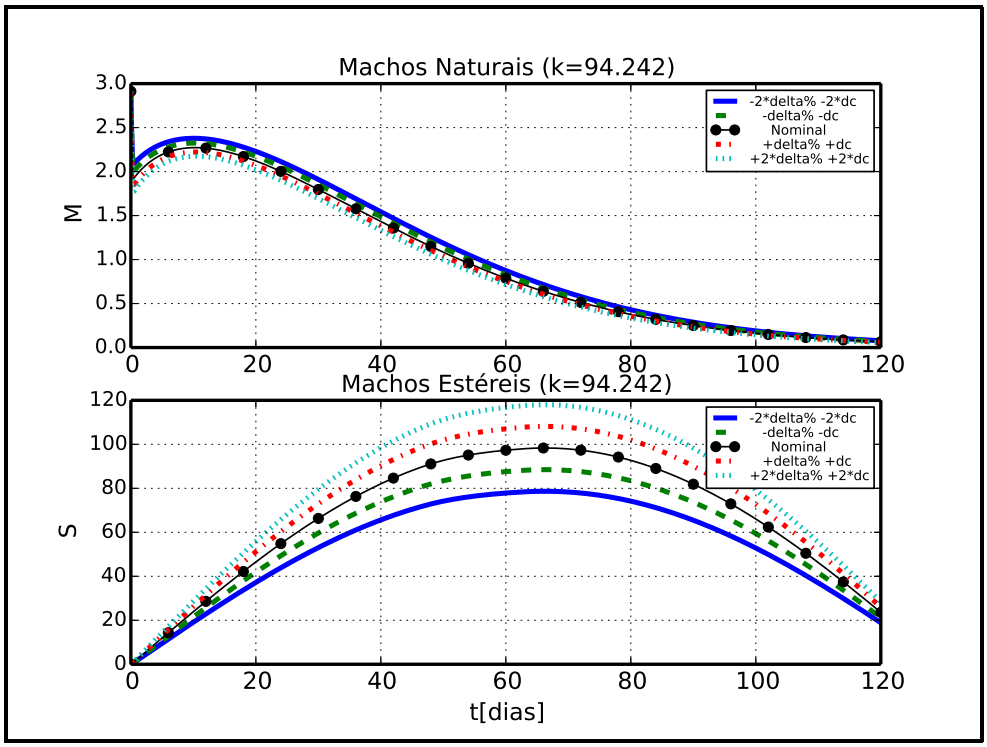

Figura 42: Machos - Variações Proporcionais e Fixas para k = 94,242. Fonte: o autor.

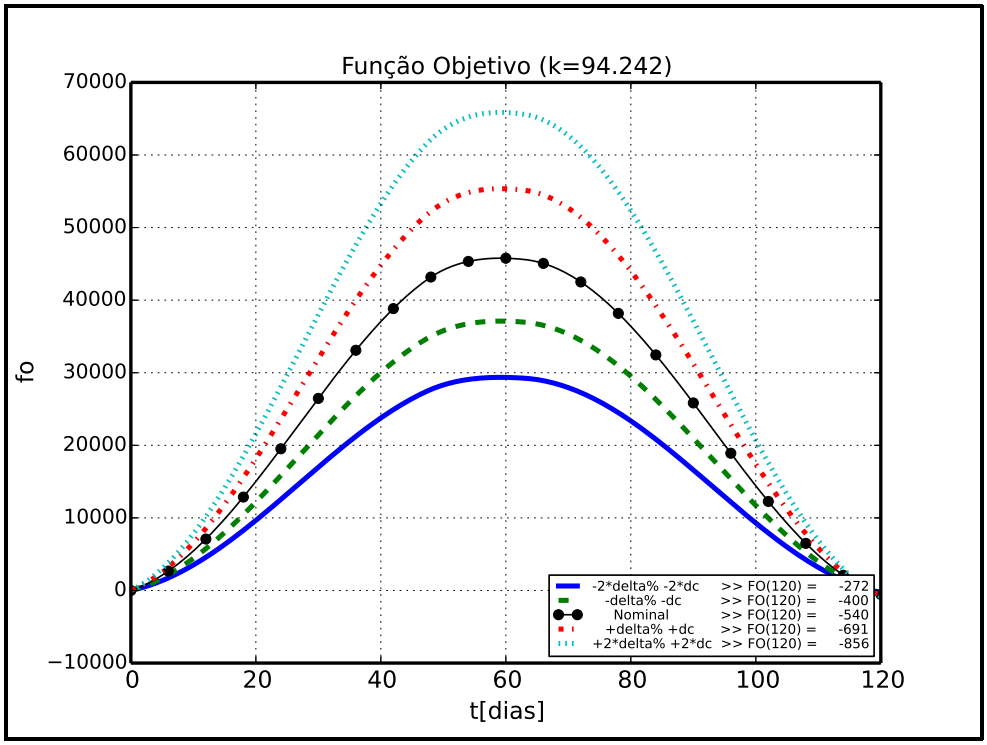

Figura 43: Função Objetivo - Variações Proporcionais e Fixas para k = 94,242. Fonte: o autor.

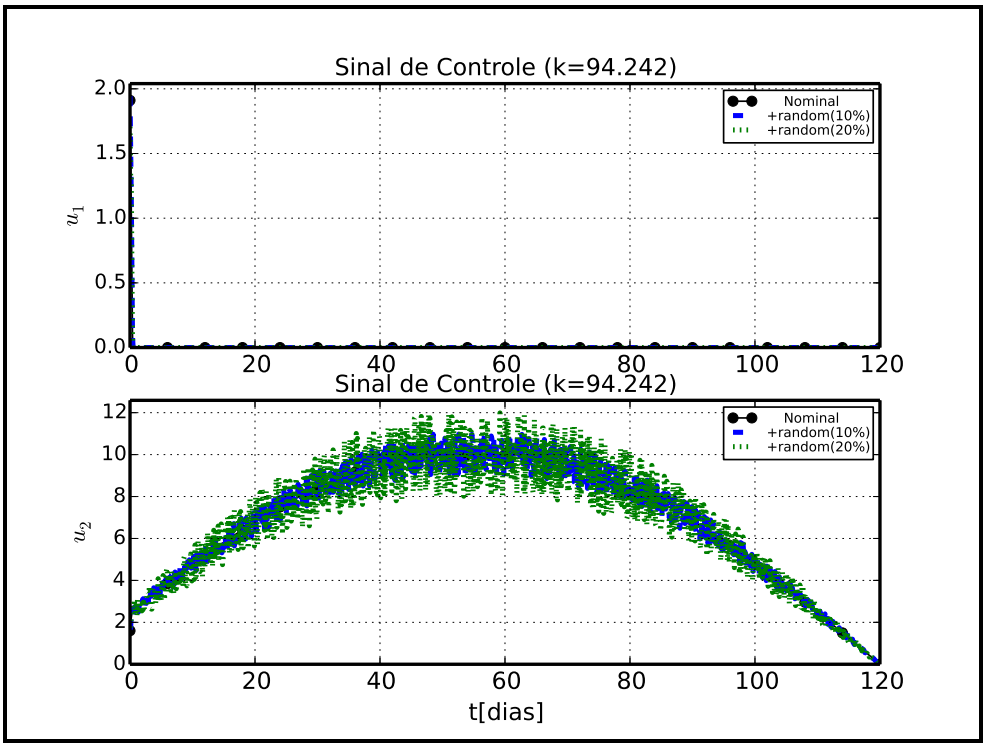

Figura 44: Controles - Variações Aleatórias para k = 94,242. Fonte: o autor.

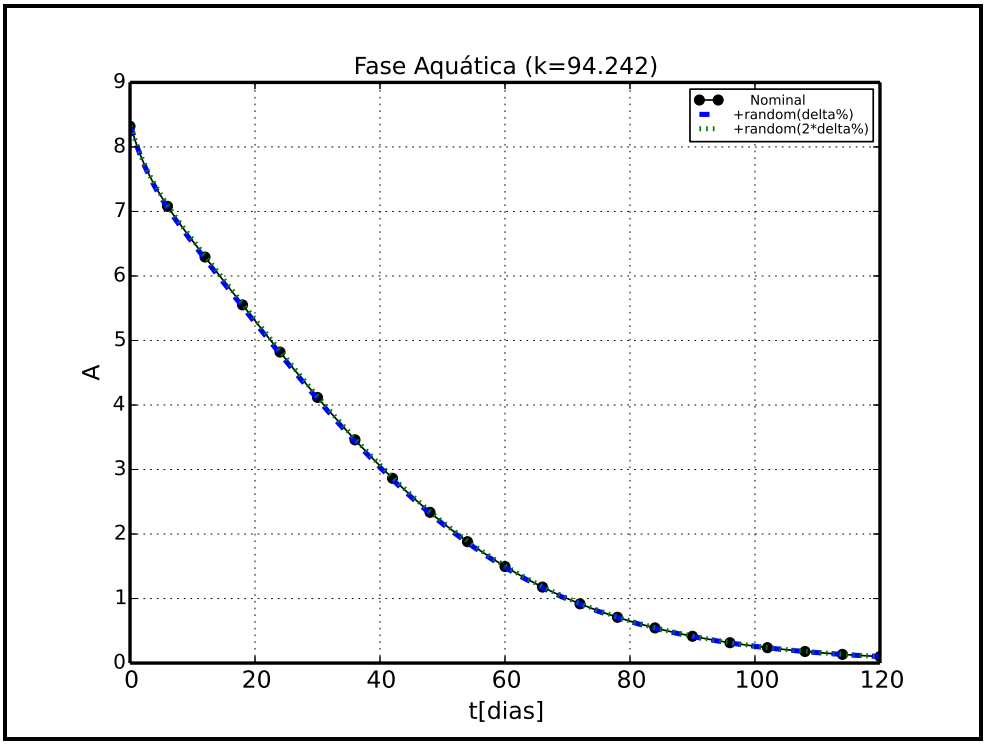

Figura 45: Fase Aquática - Variações Aleatórias para k = 94,242. Fonte: o autor.

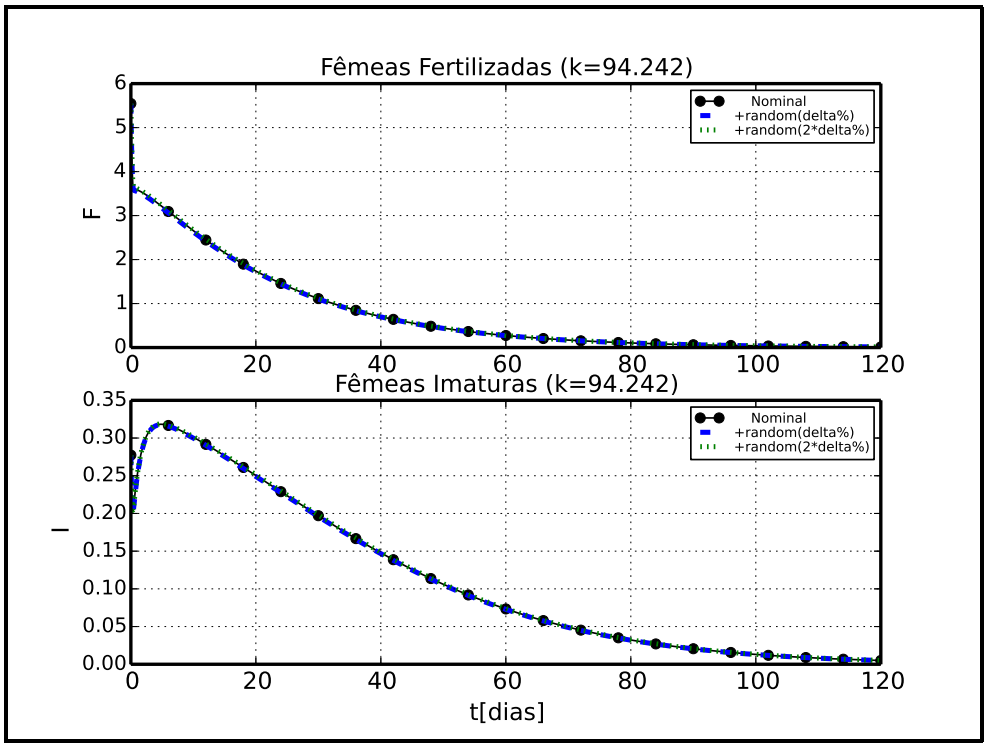

Figura 46: Fêmeas - Variações Aleatórias para k = 94,242. Fonte: o autor.

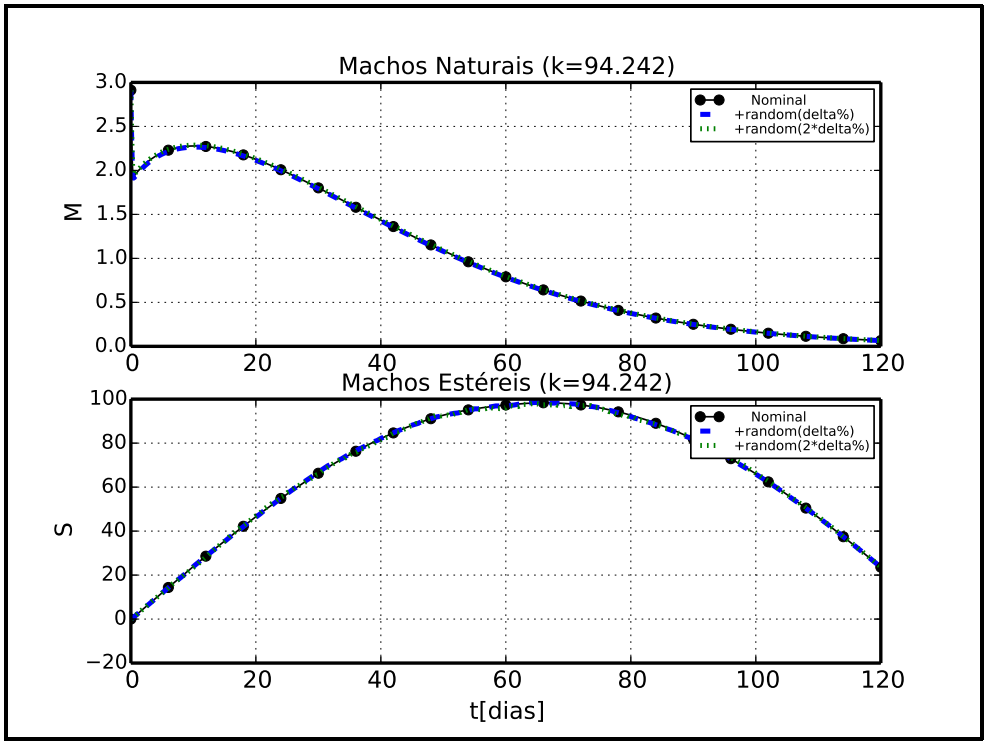

Figura 47: Machos - Variações Aleatórias para k = 94,242. Fonte: o autor.

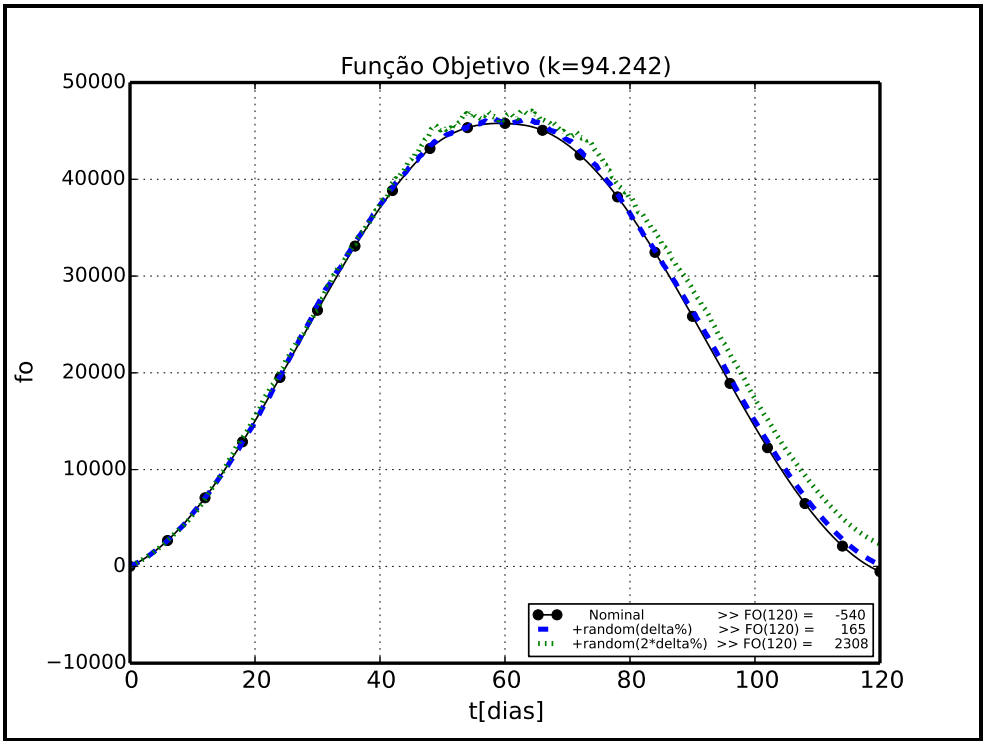

Figura 48: Função Objetivo - Variações Aleatórias para k = 94,242. Fonte: o autor.

APÊNDICE D - ESTUDO VARIACIONAL - SIMULAÇÕES PARA K = 94,245

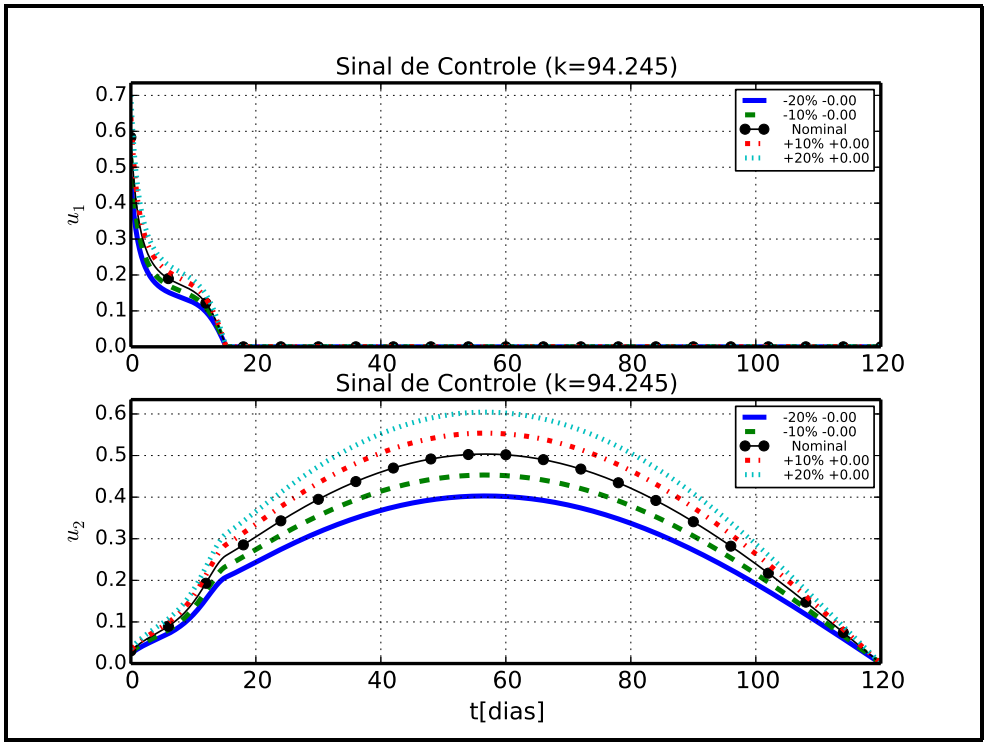

Figura 49: Controles - Variações Proporcionais e Fixas para k = 94,245. Fonte: o autor.

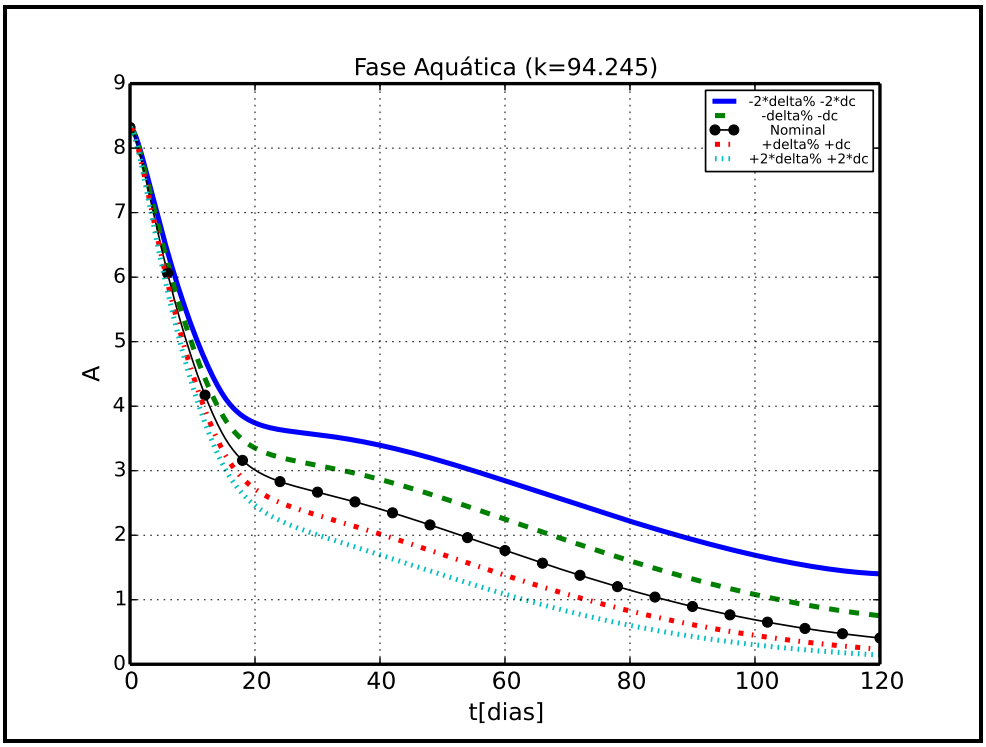

Figura 50: Fase Aquática - Variações Proporcionais e Fixas para k = 94,245. Fonte: o autor.

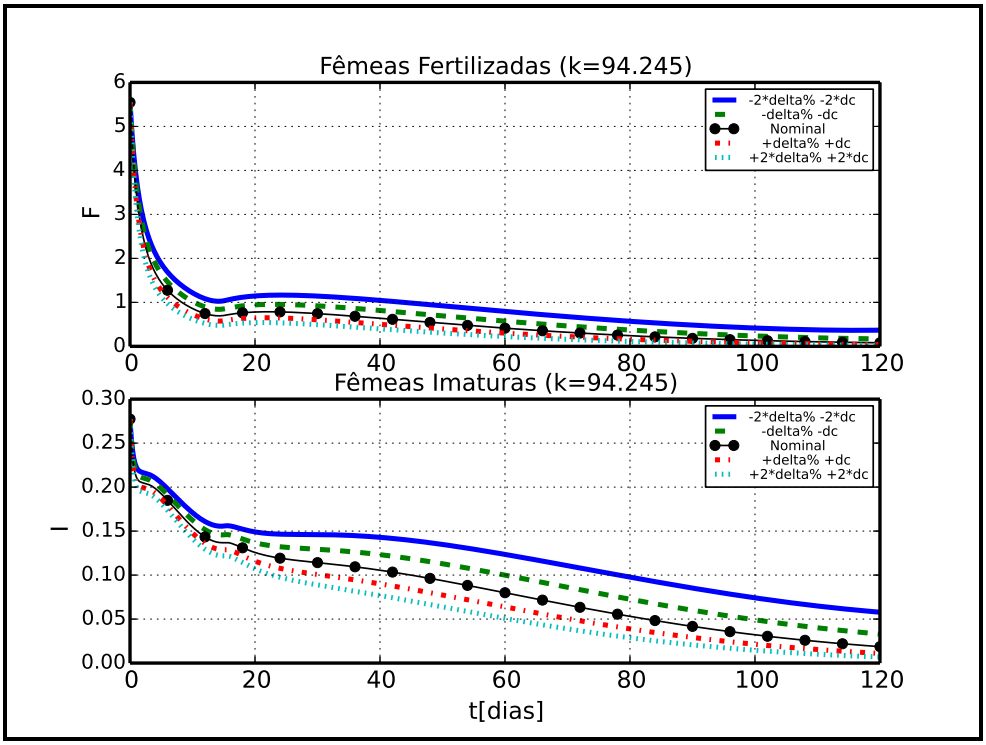

Figura 51: Fêmeas - Variações Proporcionais e Fixas para k = 94,245. Fonte: o autor.

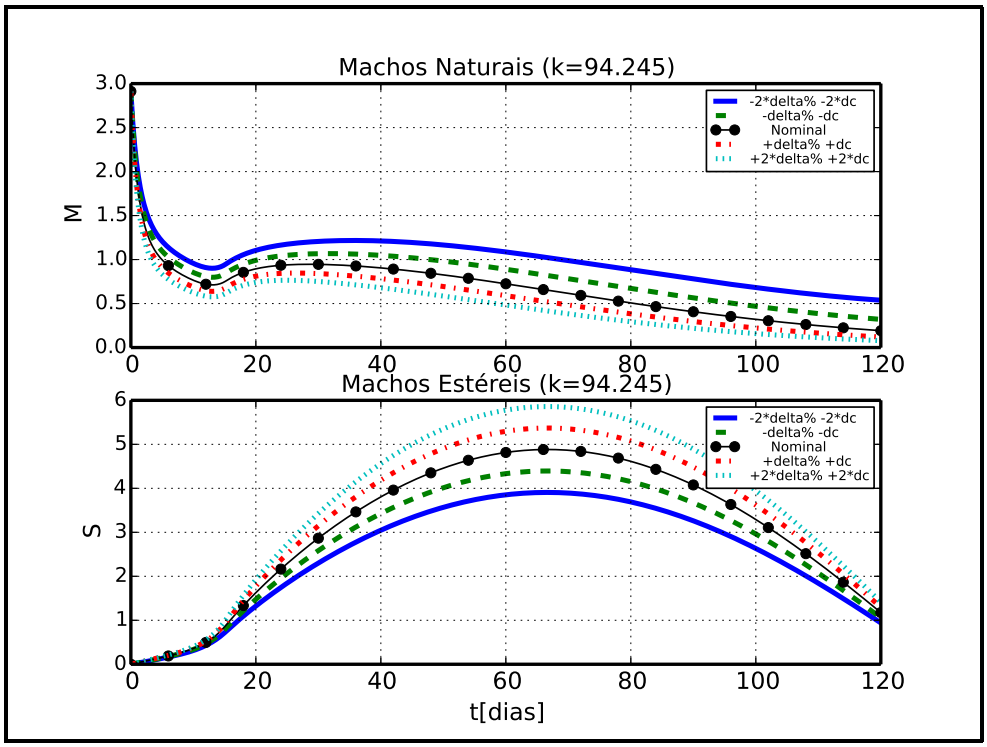

Figura 52: Machos - Variações Proporcionais e Fixas para k = 94,245. Fonte: o autor.

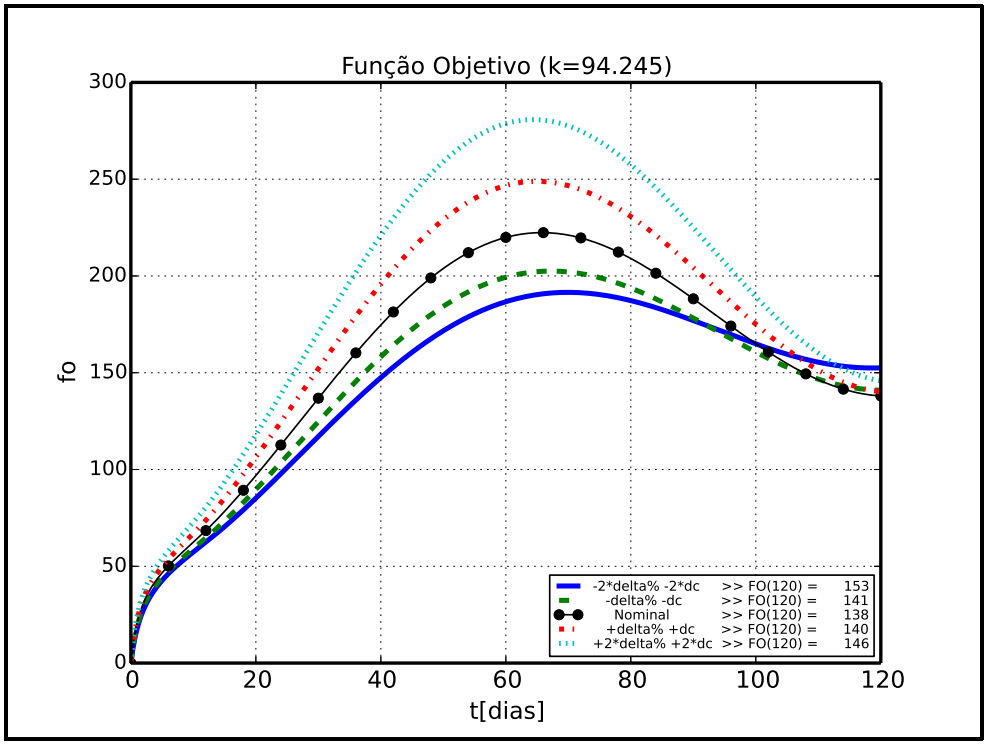

Figura 53: Função Objetivo - Variações Proporcionais e Fixas para k = 94,245. Fonte: o autor.

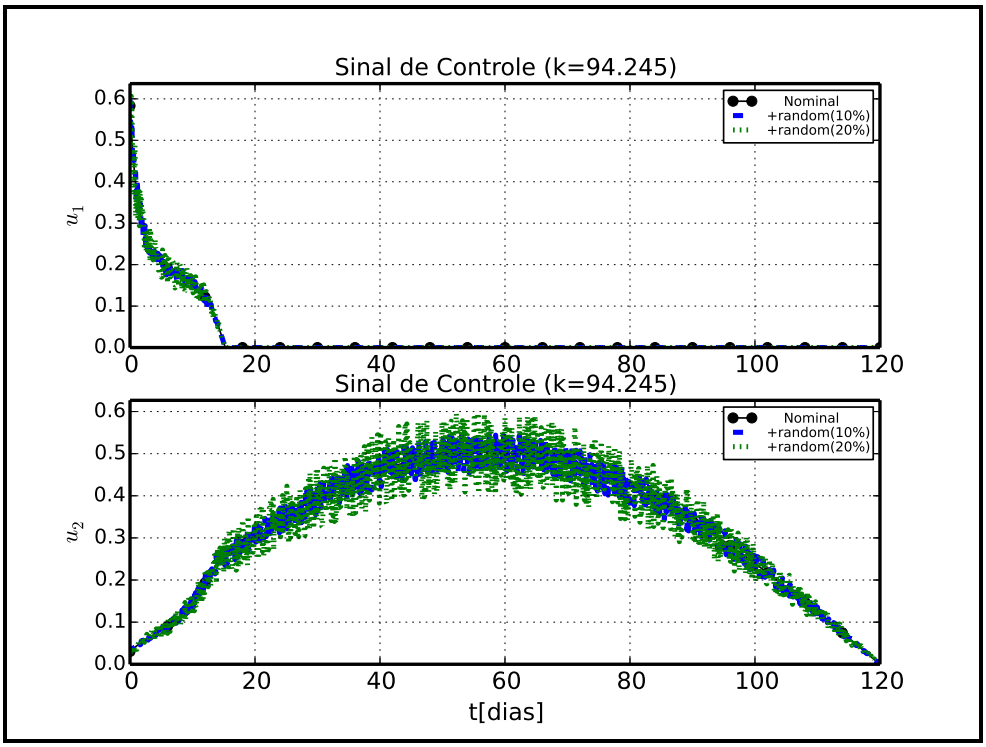

Figura 54: Controles - Variações Aleatórias para k = 94,245. Fonte: o autor.

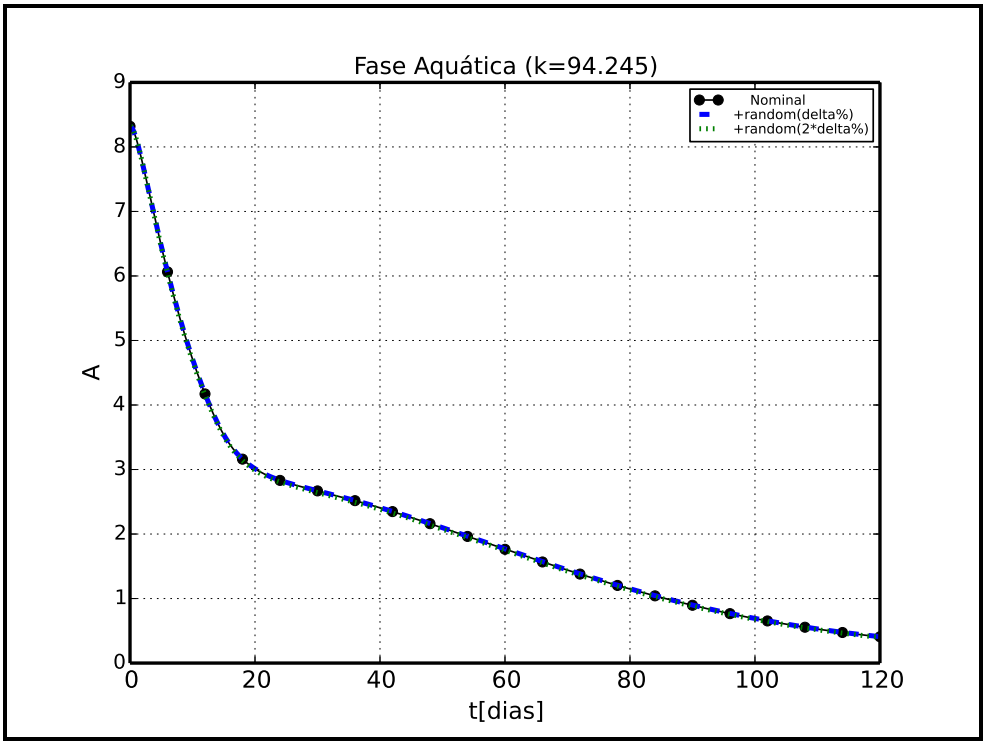

Figura 55: Fase Aquática - Variações Aleatórias para k = 94,245. Fonte: o autor.

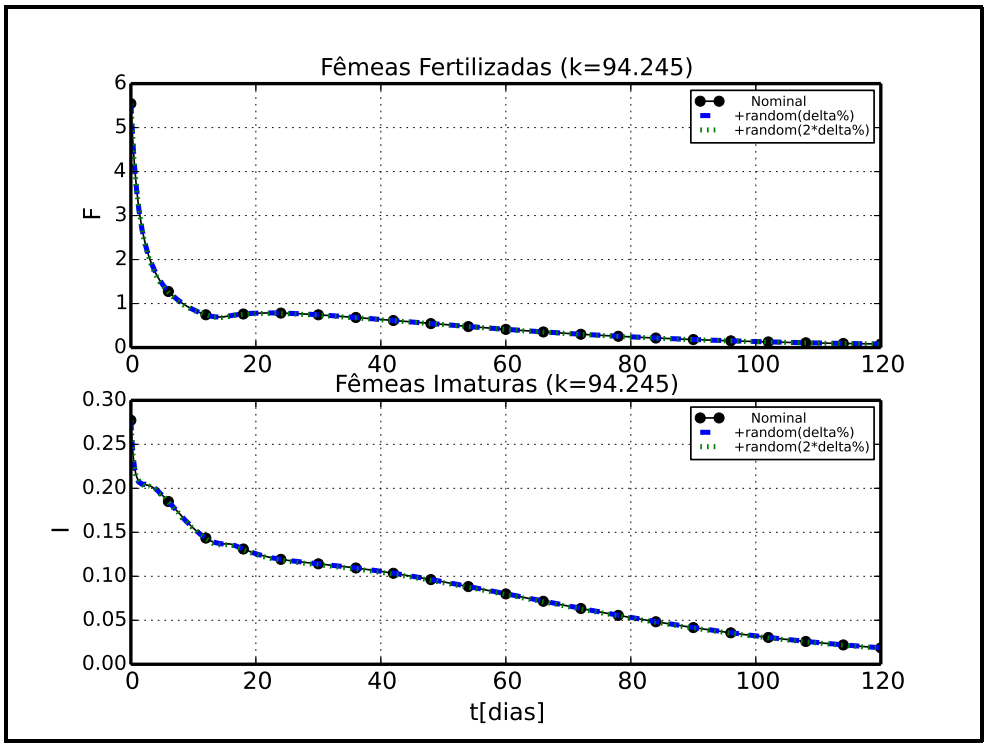

Figura 56: Fêmeas - Variações Aleatórias para k = 94,245. Fonte: o autor.

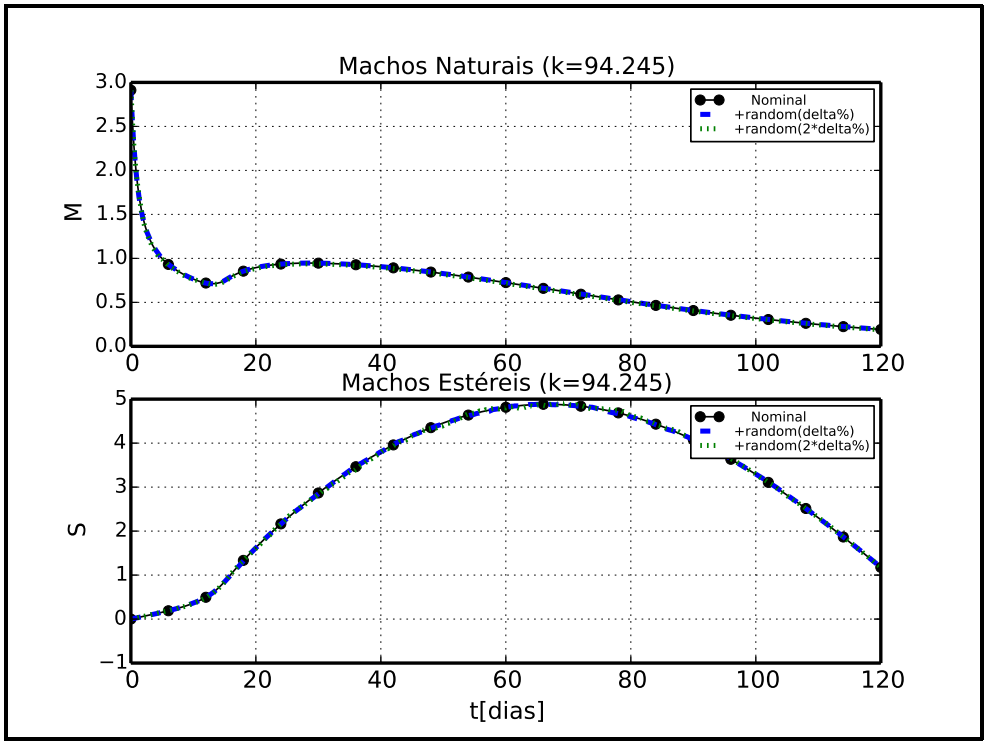

Figura 57: Machos - Variações Aleatórias para k = 94,245. Fonte: o autor.

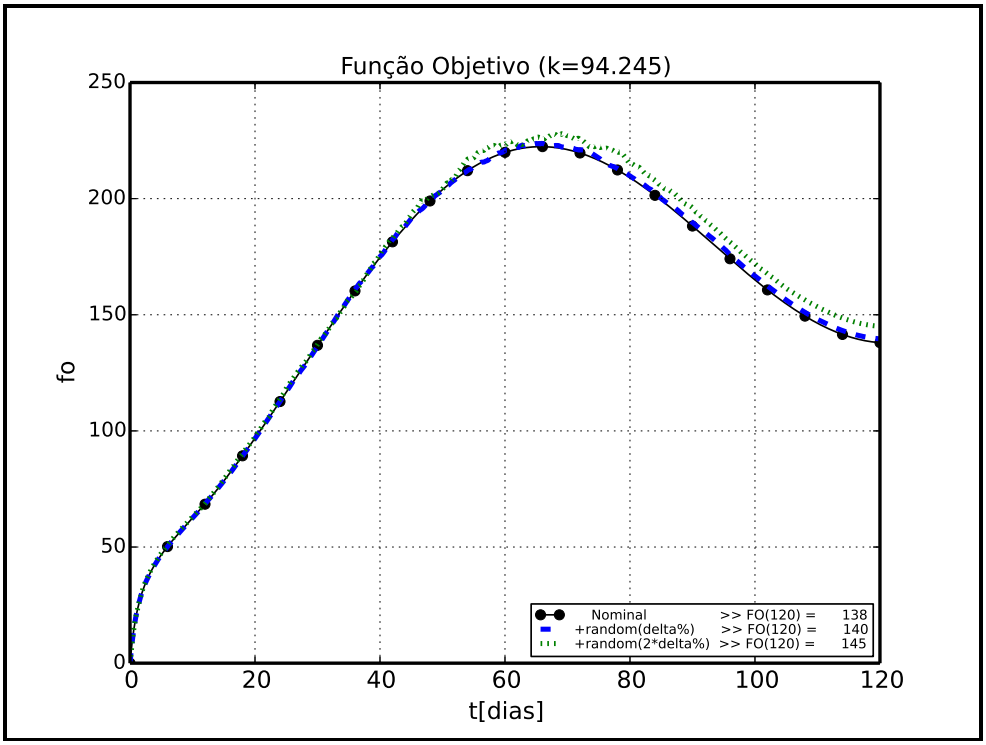

Figura 58: Função Objetivo - Variações Aleatórias para k = 94,245. Fonte: o autor.

APÊNDICE E - ESTUDO VARIACIONAL - SIMULAÇÕES PARA K = 100

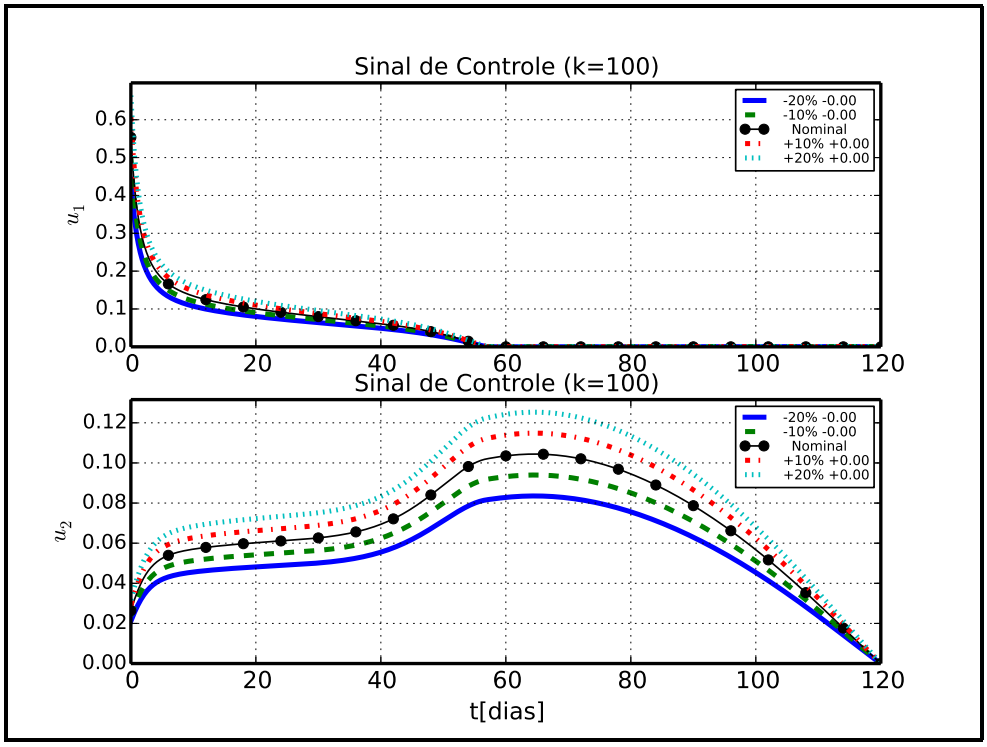

Figura 59: Controles - Variações Proporcionais e Fixas para k = 100. Fonte: o autor.

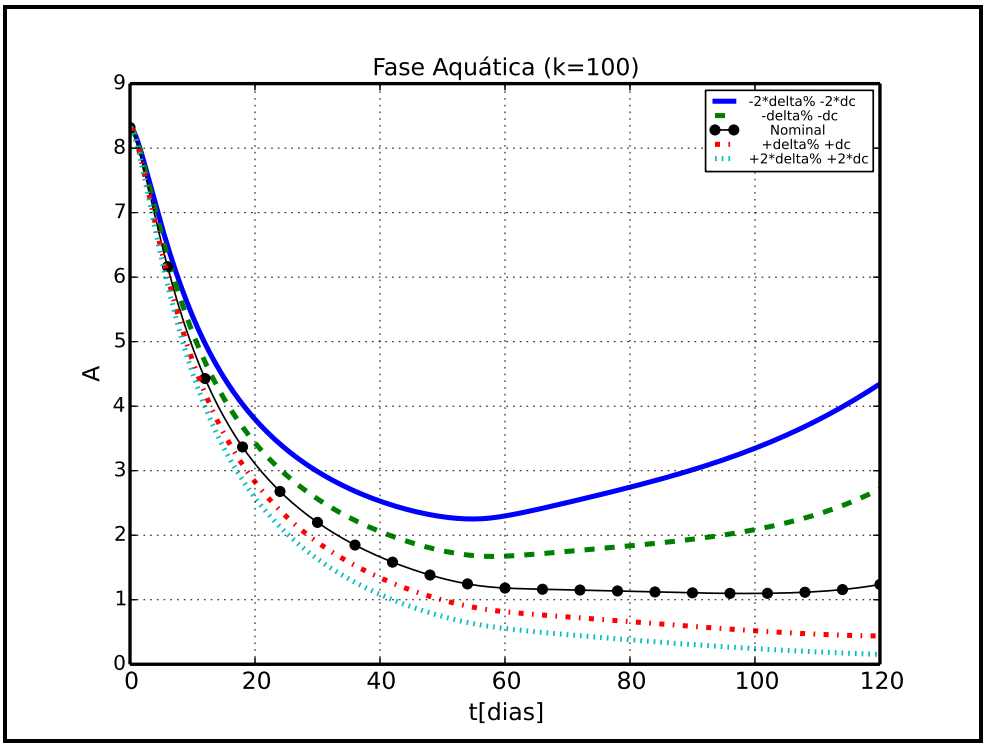

Figura 60: Fase Aquática - Variações Proporcionais e Fixas para k = 100. Fonte: o autor.

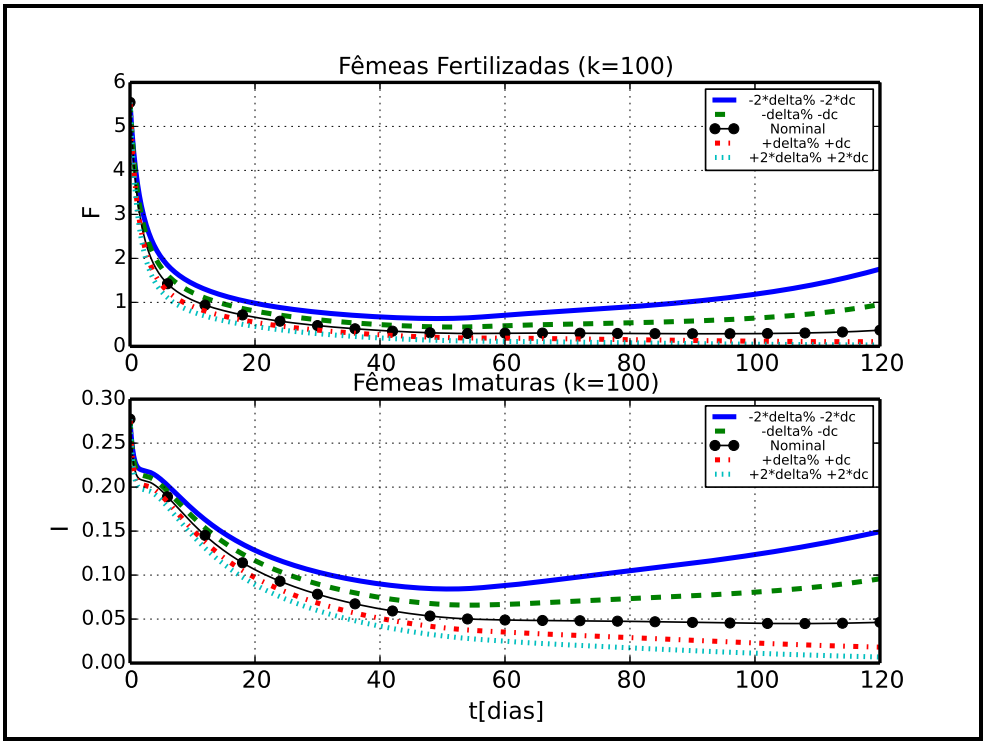

Figura 61: Fêmeas - Variações Proporcionais e Fixas para k = 100. Fonte: o autor.

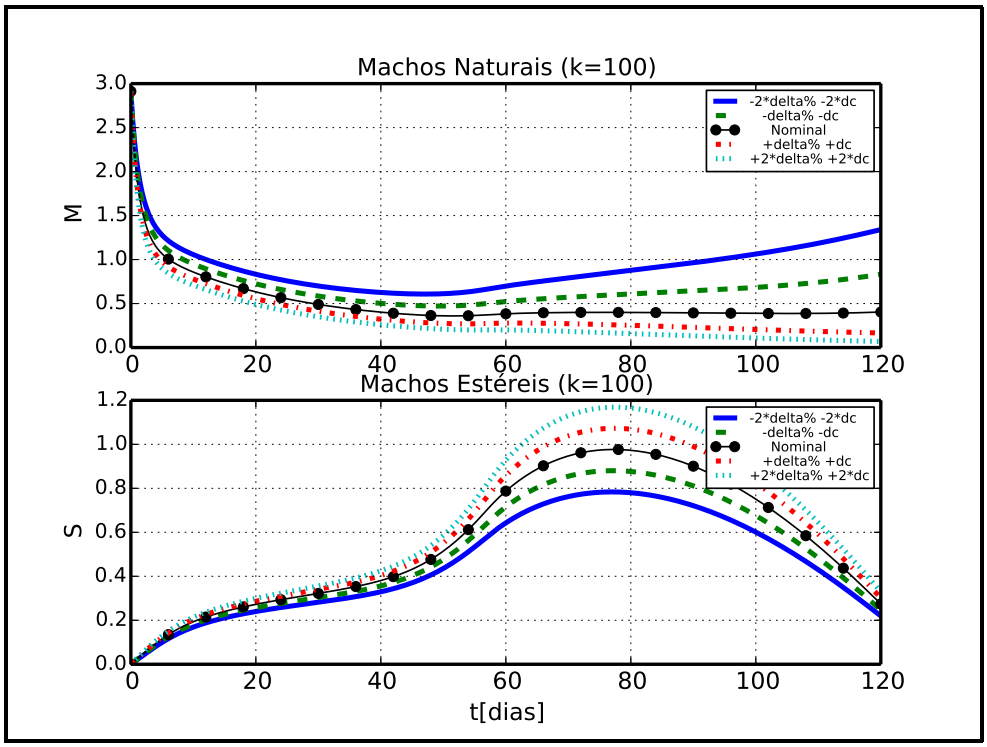

Figura 62: Machos - Variações Proporcionais e Fixas para k = 100. Fonte: o autor.

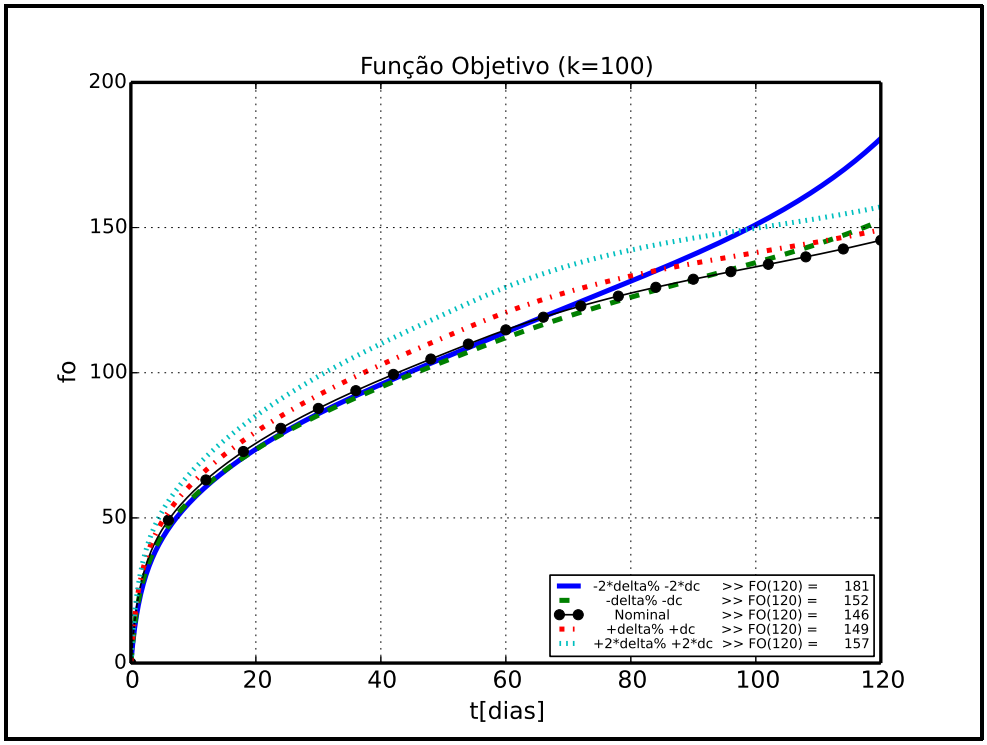

Figura 63: Função Objetivo - Variações Proporcionais e Fixas para k = 100. Fonte: o autor.

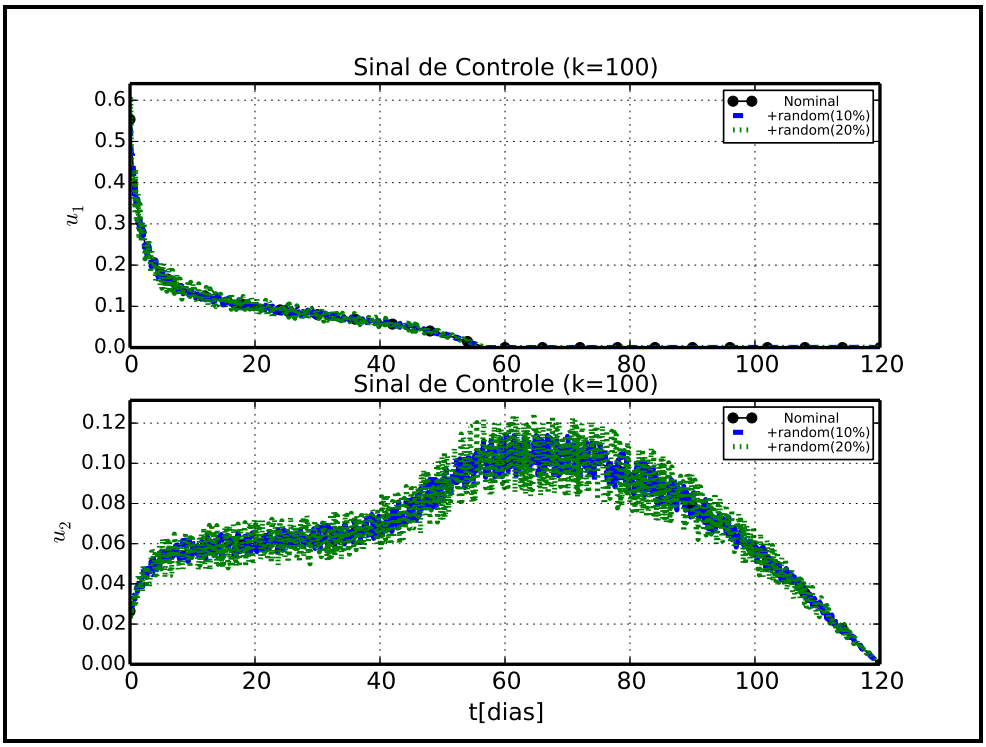

Figura 64: Controles - Variações Aleatórias para k = 100. Fonte: o autor.

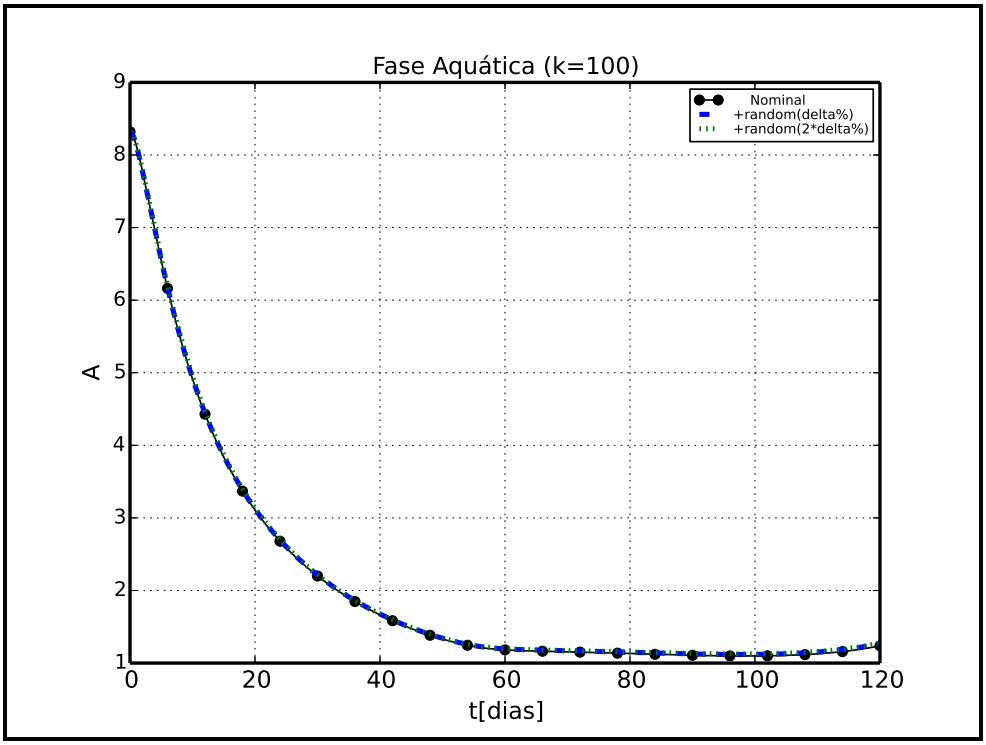

Figura 65: Fase Aquática - Variações Aleatórias para k = 100. Fonte: o autor.

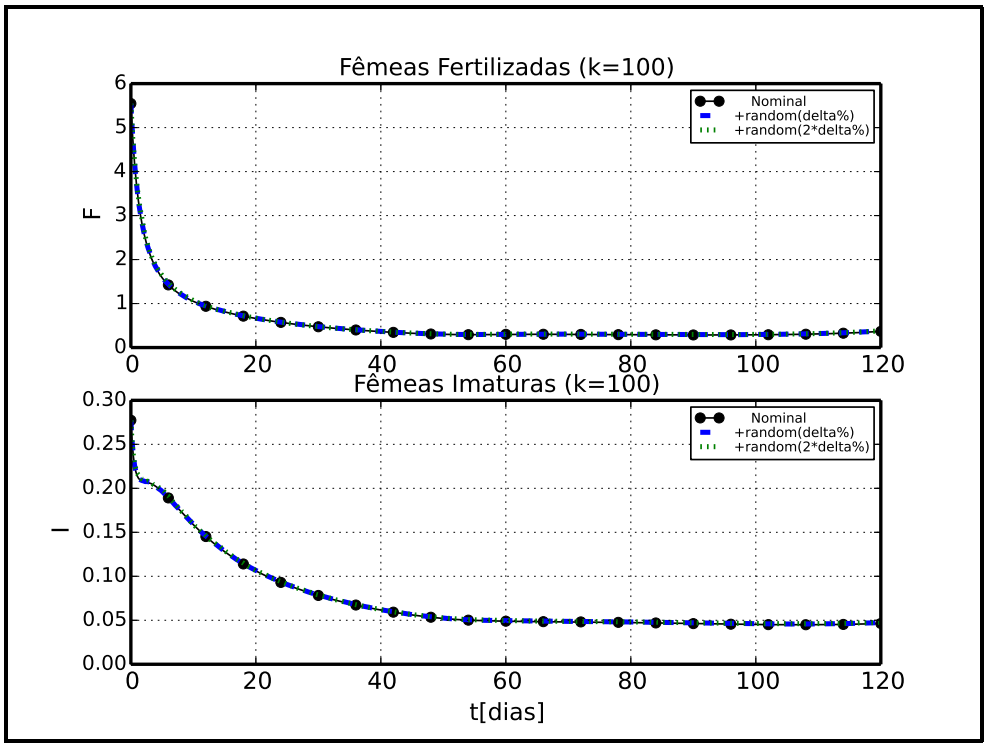

Figura 66: Fêmeas - Variações Aleatórias para k = 100. Fonte: o autor.

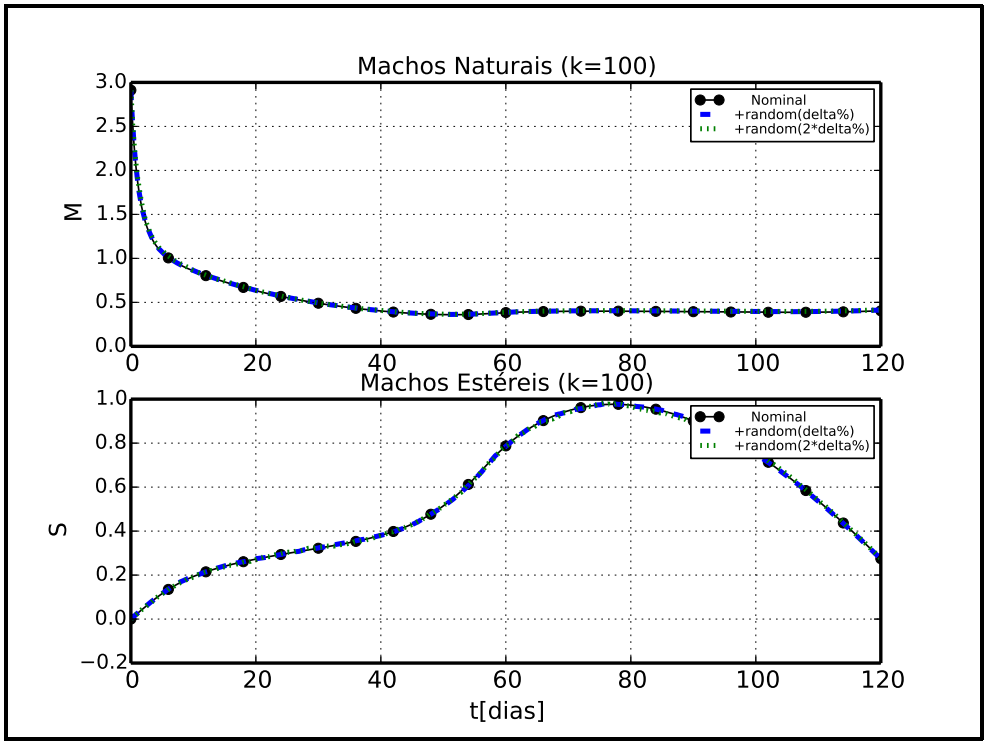

Figura 67: Machos - Variações Aleatórias para k = 100. Fonte: o autor.

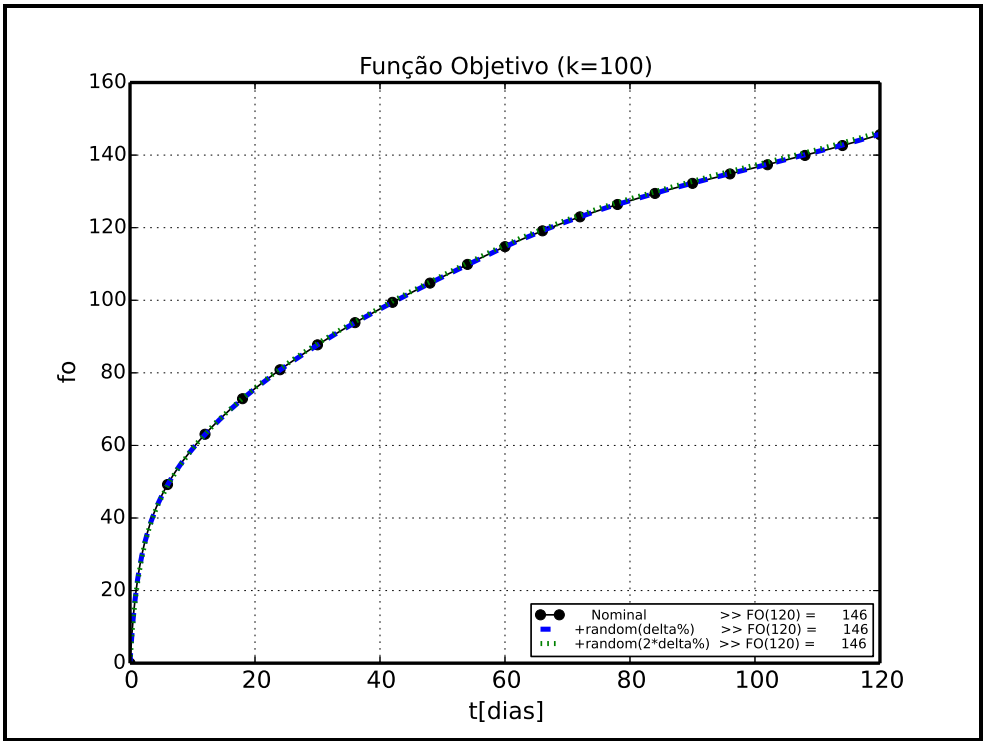

Figura 68: Função Objetivo - Variações Aleatórias para k = 100. Fonte: o autor.

ÍNDICE REMISSIVO