FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Computer Aid for Dieting and Weight Control using Model Predictive Control

Ajuda Computacional no Controlo de Peso usando Controlo Predictivo

Diogo Marques Bastos

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Orientador: Maria do Rosário Marques Fernandes Teixeira de Pinho

29 de Julho de 2015

U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

MIEEC - MESTRADO INTEGRADO EM ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES **2014/2015**

A Dissertação intitulada

"Computer Aid for Dieting and Weight Control Using Model Predictive Control"

foi aprovada em provas realizadas em 22-07-2015

o júri

Presidente Professor Doutor António Pedro Rodrigues Aguiar
Professor Associado do Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

Professora Doutora Teresa Paula Coelho Azevedo Perdicoúlis
Professora Assistente do Departamento de Matemática da Universidade Trás-os-Montes e Alto Douro

Professora Doutora Maria do Rosário Marques Fernandes Teixeira de Pinho
Professora Associada do Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

O autor declara que a presente dissertação (ou relatório de projeto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extratos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são corretamente citados.

Diogo Bastos

Autor - Diogo Marques Bastos

Faculdade de Engenharia da Universidade do Porto

Resumo

Este projeto visa modelar as equações da Dinâmica da Perda de Peso, propor um Dimensionamento da Ingestão Calórica adequado (saudável e eficaz) e por fim utilizar Modelos de Controlo Predictivo na recomendação de um plano dietético que admite perturbações inesperadas nos resultados.

Primeiramente, foi modelado o funcionamento do corpo humano, construindo um sistema cujas entradas são a Actividade Física e a Ingestão Alimentar e a saída é o Peso. Para isso, utilizou-se o Simulink, uma ferramenta gráfica que possibilita uma maior percepção das partes constituintes do problema, decompondo este em imagens (blocos). Após o estudo e implementação bem sucedida da Dinâmica, são apresentados dois métodos de dimensionamento da Ingestão Alimentar. O primeiro é um algoritmo (script MATLAB) totalmente original, que regula manualmente a Ingestão Alimentar. Através deste algoritmo, validou-se a Dinâmica desenvolvida anteriormente. O segundo algoritmo advém de um código adaptado do ICLOCS. O código ICLOCS está escrito em linguagem MATLAB e o seu propósito é definir e resolver problemas de controlo óptimo. Com a passagem de variáveis entre programas e utilizando de novo ICLOCS, concebeu-se outro programa que lê a trajectória referência do peso gerada anteriormente (segundo algoritmo), compara esses dados com os dados inseridos pelo utilizador (actualizações de peso real) e devolve uma nova trajectória e respectivo plano alimentar. Este novo plano possui os mesmos objectivos que o inicial.

Abstract

This project aims to model the Weight Loss dynamics equations, propose an appropriate energy intake scheduling (healthy and effective) and finally use Model Predictive Control in the recomendation of a dietary plan that admits unexpected disturbances in the results.

First, it was built a model of how the human body works, designing a system with Physical Activity and Energy Intake as inputs and Body Weight as output. In order to do that, it was used Simulink, a graphical tool which allows a greater perception of the parts of the problem decomposing the previous in images (blocs). After the studying and successful implementation of the dynamics, two sizing methods of Food Intake are presented. The first is an algorithm (MATLAB script) completely unique that manually regulates the Energy Intake. Using that algorithm, the previously developed dynamic was validated. The second algorithm came from an adaptation of ICLOCS code. ICLOCS is written in MATLAB language and its purpose is to define and solve optimal control problems. With the passing of variables between programs and again, using ICLOCS, it was conceived another program that reads the weight of the reference trajectory generated earlier (second algorithm), compares that value with the user-entered data (actual weight update) and returns a new trajectory and its food plan. This new eating plan has the same goals of the first one.

Agradecimentos

Nunca pensei que esta parte fosse tão difícil de escrever, por isso fico-me por aqui.

Diogo Marques Bastos

No entanto, seria um tanto ingrato da minha parte ignorar todos aqueles que me deram apoio ao longo deste bonito percurso académico. Nesse sentido, congratulo os meus pais pelo maravilhoso filho que educaram, por conseguirem pacientemente, e sem exercerem qualquer pressão, suportar com graça todos estes anos, anos estes com um pouco de tudo. Como prova esta dissertação, a criação é realmente das coisas mais majestosas que a vida nos dá.

Quero agradecer também à minha Maria (Guimarães), tão boa a estabilizar como a destabilizar, factor essencial para uma vida feliz que me ajudaste a viver. Uma nota especial nesta fase final: os pouquíssimos convites que me obrigaste a recusar, como uma ida à praia ou outro sítio paradisíaco, ou seja, qualquer outro destino bem mais glamoroso que a FEUP. Fica então registado neste documento oficial que gosto muito de ti (já não precisamos de casar!).

Abraço também com palavras todos os meus amigos, os que nunca deixei quando fui para a Faculdade, os que trouxe para a Faculdade e os que fiz na Faculdade - englobando todos os mencionados (espero que não me "castiguem"por não os nomear): os que levo quando deixar a Faculdade.

Seguindo a ordem cronológica, chega a vez da minha orientadora Professora Maria Rosário de Pinho e co-orientadora Professora Amélia Caldeira. No contexto desta dissertação, com as suas capacidades de motivação, e sobretudo, reconhecimento, foram de facto as pessoas mais importantes. Reconheço-as agora eu, como as grandes orientadoras que são, e mais do que isso, como verdadeiras professoras.

Devia mesmo ter-me ficado por ali, resumindo: Obrigado!

Diogo Marques Bastos

"Successful engineering
is all about understanding how things break or fail."

Henry Petroski

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

AMPL	A Mathematical Programming Language	
BEE	Basal Energy Expenditure	Gastos Energéticos Basais
BF	Body Fat	Gordura Comporal
BM	Body Mass	Massa Corporal
BMR	Basal Metabolic Rate	Taxa Metabólica Basal
BW	Body Weight	Peso Corporal
CI	Carbohydrate Intake	Ingestão de Hidratos de Carbono
CUTEr	Constrained and Unconstrained Testing Environment, revisited	
EB	Energy Balance	Balanço Energético
EC	Energy Cost	Custo Energético
EE	Energy Expenditure	Gastos Energéticos
EI	Energy Intake	Ingestão Alimentar
EER	Energy Expenditure Reduction	Redução de Gastos Energéticos
FCNAUP	Faculdade de Nutrição da Universidade do Porto	
FFM	Fat Free Mass	Massa Magra
FI	Fat Intake	Ingestão de Gorduras
FM	Fat Mass	Massa Gorda
GPS	Global Positioning System	
ICLOCS	Imperial College London Optimal Control Software	
IPOPT	Interior Point Optimizer	
LBM	Lean Body Mass	Massa Corporal Magra
MET	Metabolic Equivalent of Task	Equivalente Metabólico de Tarefa
MPC	Model Predictive Control	Controlo Predictivo em Modelos
NLP	Non Linear Problems	Problemas não lineares
ODE	Ordinary Differential Equation	Equações ordinárias diferenciais
PA	Physical Activity	Actividade Física
PAL	Physical Activity Level	Nível de Actividade Física
PI	Protein Intake	Ingestão de Proteínas
RMR	Resting Metabolic Rate	Taxa Metabólica de Repouso
TEF	Thermic Effect of Feeding	Efeito Térmico da Alimentação

Nota: Algumas siglas representam as mesmas grandezas. Foram mantidas as diferentes notações para facilitar a consulta das referências, entre as quais:

FM significa o mesmo que LBM;
BEE significa o mesmo que BMR;
BM significa o mesmo que BW.

Capítulo 1

Introdução

Quando se tem uma fonte de tensão a alimentar um circuito resistivo, há duas maneiras de variar a potência dissipada: através da intensidade da fonte ou da resistência utilizada. Numa dieta tem-se exactamente a mesma situação: pode-se regular a quantidade de alimentos ingeridos (fonte) e/ou ajustar o exercício físico (resistência) aos objectivos que se ambiciona atingir. Torna-se então oportuna a modelação da perda de peso, de modo a facultar informação com valor acrescentado, visando uma futura aplicação. O modelo da dinâmica apresentada está capacitado a responder a um cenário tanto de perda como de ganho de peso. No entanto, devido à maior procura, o dimensionamento da Ingestão Alimentar foi desenvolvido apenas para o cenário de perda de peso.

Existem dezenas de estudos, *websites* e aplicações que abordam a temática das Dietas e baseiam-se em diferentes equações estáticas e por vezes, desactualizadas.Nos últimos anos tem sido feito um enorme esforço para modelizar a evolução do peso de uma pessoa como um sistema dinâmico (ver [9], [10] e respectivas referências). Este trabalho é baseado na mais recente literatura a que se obteve acesso. Nela tais modelos são propostos, discutidos e confrontados com informação de pessoas submetidas a dietas alimentares. Pretende-se simular as variações de peso em individuos adultos e saudáveis. Não é usada qualquer informação real, pois tal implicaria vários passos incluindo autorizações oficiais da protecção de dados. No entanto, este trabalho foi executado em estrita colaboração com a Faculdade de Nutrição da Universidade do Porto (em colaboração com o professor Pedro Moreira e a professora Patrícia Padrão).

Fazer dieta tem dois focos principais: saúde e aparência. Concentrando-nos na saúde, é de notar a incessante preocupação dos consumidores pelos produtos que levam para casa: de onde vêm, a qualidade, o conteúdo nutricional. Uma alimentação com rácios nutricionais ou quantidades calóricas inadequadas pode significar uma maior susceptibilidade a um grande leque de doenças. Utilizando o exemplo da Obesidade, esta doença é considerada como a Epidemia do século XXI. De acordo com um estudo da autoria do McKinsey Global Institute [11] , o combate à Obesidade tem um custo de 2 mil biliões de dólares por ano em custos de Cuidados de Saúde e Investimentos para mitigar o seu impacto.

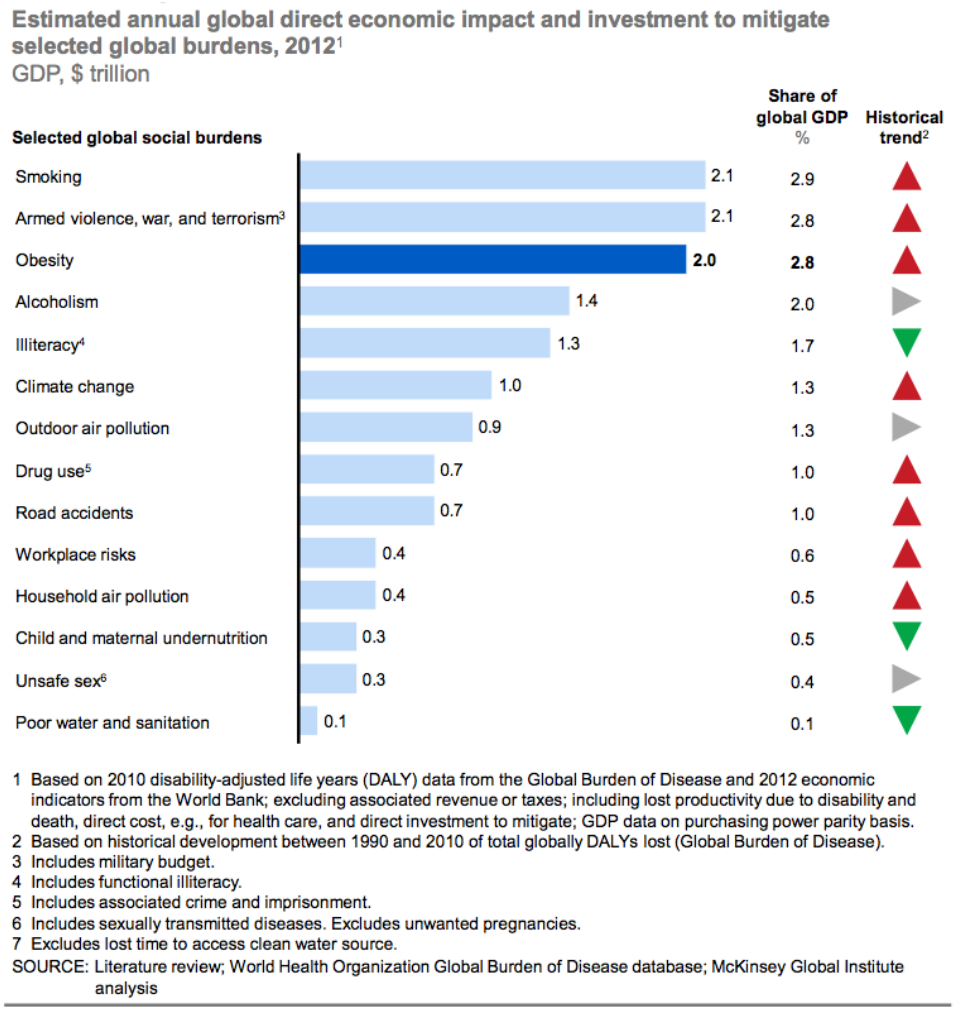


Figura 1.1: Investimentos em Epidemias Mundiais

Com o proliferar da Obesidade e o aumento da preocupação com a própria imagem, as dietas motivadas pela aparência também ganham relevo. Fenómenos como "*Fat Phobia*"[12], o medo de se tornar obeso, indesejável e não aceite são grandes impulsionadores de dietas integrando o vasto "mercado da beleza".

Independentemente da razão que leva o utilizador a fazer dieta, a ferramenta de controlo proposta neste trabalho potencia o cumprimento dos seus objectivos desde que sejam realistas e não comprometam a sua saúde.

1.1 Motivação

A principal motivação deste projecto é utilizar ferramentas de controlo num problema tão dinâmico como a perda de peso. Esta nova abordagem difere das existentes no mercado. Perante uma incoerência nos resultados esperados com os resultados reais, uma ferramenta com *Model*

Predictive Control não se limita a recalcular o plano dietético com a nova informação mas tem em conta o progresso passado, obtendo resultados mais precisos e personalizados.

1.2 Objectivos e Fases do Trabalho

O principal objectivo deste trabalho é o teste e simulação de ferramentas computacionais que ajudem um individuo submetido a uma dieta alimentar a controlar o seu peso. Esse individuo confronta periodicamente o peso que deveria ter com o seu peso actual e a ferramenta redefine o plano dietético para alcançar os seus objectivos.

As fases deste projecto foram as seguintes:

* Obtenção, a partir da literatura especializada, das equações dinâmicas da variação de peso necessárias à simulação de programas de dietas alimentares;
* Dimensionamento da Ingestão Alimentar de uma forma plausível, saudável e eficaz na perda de peso;
* Desenvolvimento de um modelo matemático adaptável à utilização de software numérico;
* Simulação de várias dietas para teste;
* Formulação do problema de controlo óptimo a ser usado. Tal implica a escolha do controlo, da função custo e a determinação das restrições de estado e controlo;
* Cálculo de uma trajectória de referência (peso versus tempo) a ser usado no custo do Model Predictive Control.
* Simulações com MPC.

Sabe-se que os dois factores que controlam o peso são a actividade física e a energia ingerida (alimentação). É importante referir que nas simulações computacionais apresentadas nesta dissertação a actividade física é considerada constante ao longo da dieta. Assim, o controlo é apenas um e esse é a variação da Energia Ingerida, $\Delta EI(t) = u(t)$.

1.3 Estrutura do Documento

Este documento está dividido em 6 capítulos:

- No capítulo 2, intitulado de Preliminares, são apresentados os Conceitos Teóricos essenciais à aprendizagem da temática, o Estado de Arte relativo às aplicações dietéticas e respectivos modelos (se facultados), as Ferramentas de Controlo e Software utilizados no projecto e uma Breve Descrição do MPC;

- No capítulo 3 é descrito o Modelo Dinâmico da Mudança de Peso, acompanhado com a sua representação gráfica em *Simulink*, onde foi cruzada informação de vários artigos de modo a obter uma trajectória realista da massa corporal;
- No capítulo 4 sugere-se um exemplo de algoritmo de escalonamento de Ingestão Alimentar como suporte a um plano dietético;
- No capítulo 5 é apresentado o algoritmo de resolução de problemas com controlo óptimo no software *ICLOCS*;
- No capítulo 6 são apresentadas conclusões e propostas para Trabalhos Futuros. Apresentam-se ideias que não foram executadas, propostas de resolução de alguns problemas encontrados e soluções para optimizar os productos do trabalho;
- Em anexo estão presentes na integra os programas desenvolvidos e material de suporte utilizado ao longo do projeto.

Capítulo 2

Preliminares

Neste capítulo são apresentadas as bases deste trabalho. Apresentam-se conceitos básicos no âmbito das dietas, uma abordagem sucinta do software usado e é feito um levantamento das tecnologias computacionais existentes para apoio a dietas. Por fim, acrescenta-se uma breve descrição do Model Predictive Control.

2.1 Conceitos Fisiológicos

Devido ao grande número de variáveis e definições na área de Fisiologia, é necessário fazer uma contextualização e um estudo aprofundado relativamente aos conceitos mais prementes na variação de peso de um individuo. Estes conceitos são aprofundados e completados no Capítulo 3.

2.1.1 Massa Corporal Magra

Massa Corporal Magra (*Lean Body Mass* ou *Fat Free Mass*) é um constituinte da composição do corpo que equivale à parte não gordurosa do Ser Humano (ossos, pele, músculo, órgãos, etc). Matematicamente é calculada através da subtracção da gordura corporal (lípidos armazenados) ao peso total do corpo:

$$FFM = BW - FM, \tag{2.1}$$

onde:
* FFM – Massa Magra;
* BW – Peso Corporal;
* FM – Massa Gorda.

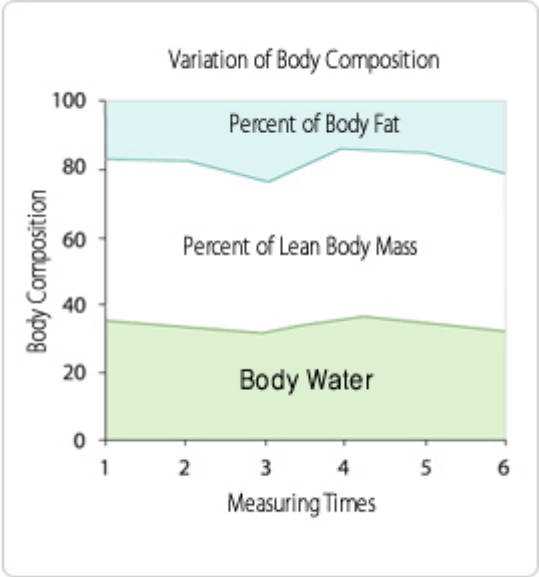


Figura 2.1: Exemplo da Constituição do Corpo Humano [1]

Como determinar a Massa Gorda:

Há uma grande variedade de métodos para determinar a FM. Alguns desses métodos utilizam equipamento tecnológico especializado, por exemplo:

* Pesagem através de Imersão (figura 2.2);

* BOD POD (compartimento computorizado) (figura 2.3);

* DEXA (*Dual Energy X-Ray Absorptiometry*) (figura 2.4).

Existem outras soluções mais simples e de menor dimensão, por exemplo:

* Lipocalibrador (figura 2.5);

* BIA (*Bioelectric Impedance Analysis*) (Figura 2.6) .

Figura 2.2: Pesagem através de Imersão [2]

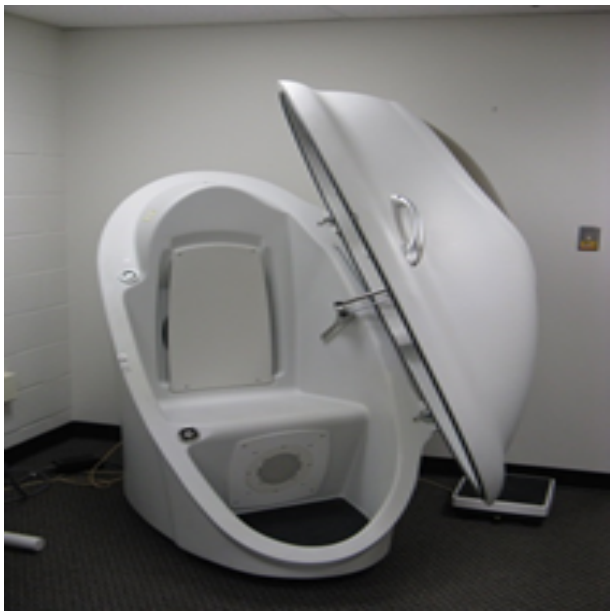


Figura 2.3: BOD POD [3]

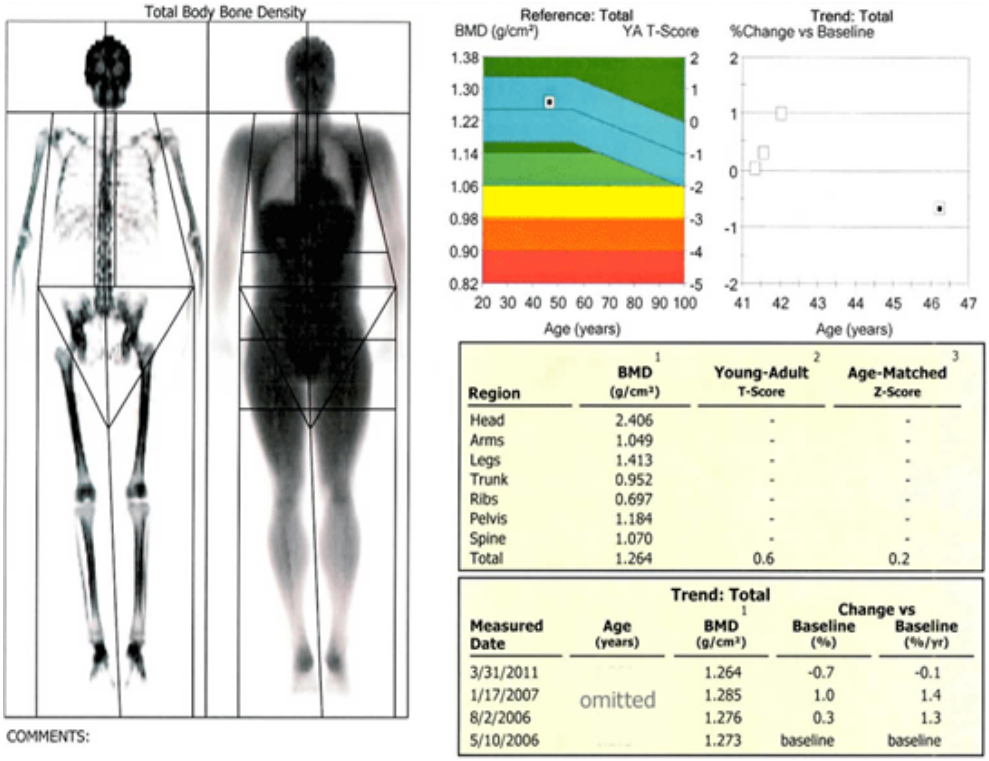


Figura 2.4: Dual Energy X-Ray Absorptiometry [4]

Figura 2.5: LipoCalibrador [5]

Figura 2.6: Bioelectric Impedance Analysis [6]

2.1.2 Taxa Metabólica Basal

A Taxa Metabólica Basal (*Basal Metabolic Rate* - BMR) é uma equação que calcula o mínimo de energia despendida por unidade de tempo em repouso [13]. Manteve-se a notação de [13] onde BMR está designado por P. Reserva-se a designação de BMR para as fórmulas adoptadas ao longo deste documento. O primeiro cientista a propor uma equação para este índice foi Harris-Benedict em 1919. Essa equação foi revista em 1984 e depois aperfeiçoada por Mifflin St Jeor em 1990 [13], sendo essa:

$$P = 10m + 6.25h - 5a + s, \tag{2.2}$$

onde:
* P - produção de energia total (kcal);
* m - massa (kg);
* h - altura (cm);
* a - idade (anos);
* s - factor correctivo (+5 para homens e -161 para mulheres).

Esta equação não toma em consideração um dado crucial: que quantidade da massa apresentada é gordura corporal. Um indivíduo com 1,80 metros e 100 kg pode ao mesmo tempo ser obeso com alta percentagem de massa gorda como ser atlético e alta percentagem de massa muscular. Deste facto adveio a necessidade de formular outra equação, surgindo a equação Katch-McArdle:

$$P = 370 + (21.6 * FFM). \tag{2.3}$$

Existem diversas fórmulas para o BMR que foram obtidas empiricamente em amostras com tamanho diferente e cuja população apresentava características distintas. No desenvolvimento do projeto justificar-se-á a escolha entre a variedade disponível.

2.1.3 Fases da Variação de Peso

Existem três fases relevantes à Variação de Peso: Perda de Peso Inicial, Perda de Peso Estável e Manutenção de Peso.

2.1.3.1 Perda de Peso

A Perda de Peso é o fenómeno que traduz uma diminuição na soma total da massa corporal de um sujeito. Existem vários catalisadores dessa perda de peso: voluntários e involuntários. A perda de peso pode ocorrer devido a doença, estado psicológico ou pode ser forçada através de treino e/ou dieta. Essa perda de peso acontece dado o decréscimo de Gordura Corporal, de Massa Corporal Magra, perdas de fluídos ou tecido adiposo. Biologicamente, a perda de peso sucede quando a ingestão de calorias (alimentação) é inferior ao consumo interno das mesmas (libertação de calor, transpiração, fluxo sanguíneo, etc).

Quando é iniciada a dieta (indução de balanço energético negativo) começa um período de perda de peso acentuada que dura geralmente um mês com desvio de 5 a 26 dias. Nesse período registam-se diminuições significativas da concentração de proteínas [7] e desidratação molecular [14], levando a uma diminuição rápida da massa corporal magra e consequentemente da massa corporal total.

Após este período inicial, entra a segunda fase da perda de peso, uma fase mais estável (variações menos acentuadas do que a primeira). Nesta fase as perdas de água e perdas nutricionais são normalizadas e intensifica-se a proporção de perda de Massa Gorda.

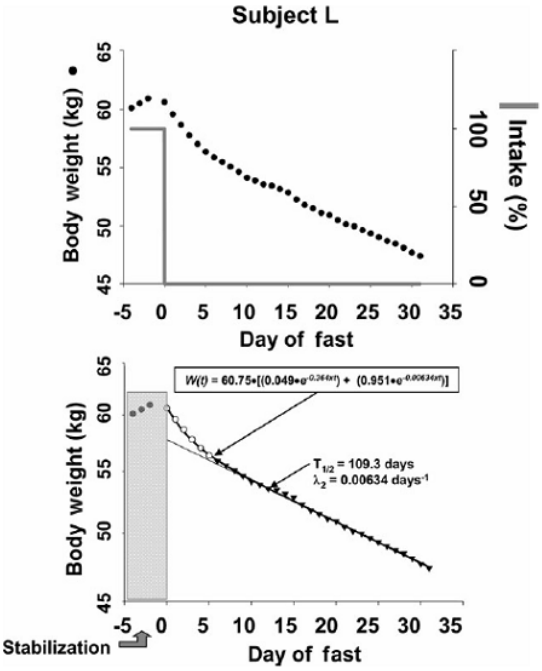


Figura 2.7: Peso em relação à Ingestão (cima) e Fase I e II da Perda de Peso (baixo) [7]

2.1.3.2 Manutenção de Peso

O que define uma Manutenção de Peso bem sucedida? Wing and Hill [15] tentaram formalizar esta questão e delimitá-la em magnitude e duração. Explicam então que uma Manutenção de Peso pode ser eficaz mesmo que o indivíduo volte a ganhar algum do peso que perdeu no inicio do programa. Um indivíduo que perca 10% do seu peso original e se mantenha constante durante um ano é considerado um caso de perda de peso bem sucedida. Para que haja uma Manutenção de Peso, o balanço energético tem que ser nulo, isto é, a ingestão calórica e os gastos energéticos têm de ser iguais para o intervalo de tempo em questão.

2.1.4 Medida da Intensidade da Actividade Física

MET é uma unidade de medida que representa a intensidade da actividade física. Por definição, um MET é equivalente à Taxa Metabólica Basal obtida (secção 2.1.2) quando se está tranquilamente sentado (1.0 MET $\rightarrow$ 4.184 kJ/ kg/h). Utilizando essa medida, é possível escalar todas as actividades físicas, desde dormir (0.9 MET's) a correr 17.5 km/h (18 MET's). Pode-se encontrar todos os valores MET's para cada actividade física no Compêndio das Actividades Físicas [16]. Eis um exemplo das tabelas desse documento:

Tabela 2.1: Excerto das Tabelas de MET's

CÓDIGO	METS	ACTIVIDADE ESPECIFICA	EXEMPLOS
16010	2.0	transportes	conduzir automóvel ou carrinha pequena
16015	1.0	transportes	andar de carro ou carrinha
16016	1.0	transportes	andar de autocarro
16020	2.0	transportes	pilotar avião
16030	2.5	transportes	conduzir motociclo
17010	7.0	caminhar	andar de mochila às costas
17020	3.5	caminhar	carregar criança ou 30 kilos de carga

2.2 Estado de Arte das Aplicações Dietéticas

Esta secção pretende fazer o levantamento das tecnologias e abordagens actuais a dietas permitindo viabilizar o ajuizamento do melhor estratégia dietética a seguir. São comentadas funcionalidades pertinentes de ser incorporadas nesta dissertação.

A preocupação com o Peso/Saúde e Figura levou que ao longo dos anos fossem criadas diversas "calculadoras" e aplicações baseadas em diferentes pressupostos e parâmetros, cujos focos são analisados de modo crítico posteriormente. A selecção destas aplicações teve como critério principal a identificação de características a implementar numa ferramenta resultante da continuação deste trabalho.

2.2.1 Katch-McArdle

Este *website* [17] baseia a sua calculadora nas fórmulas (2.3) da dupla Katch-McArdle.

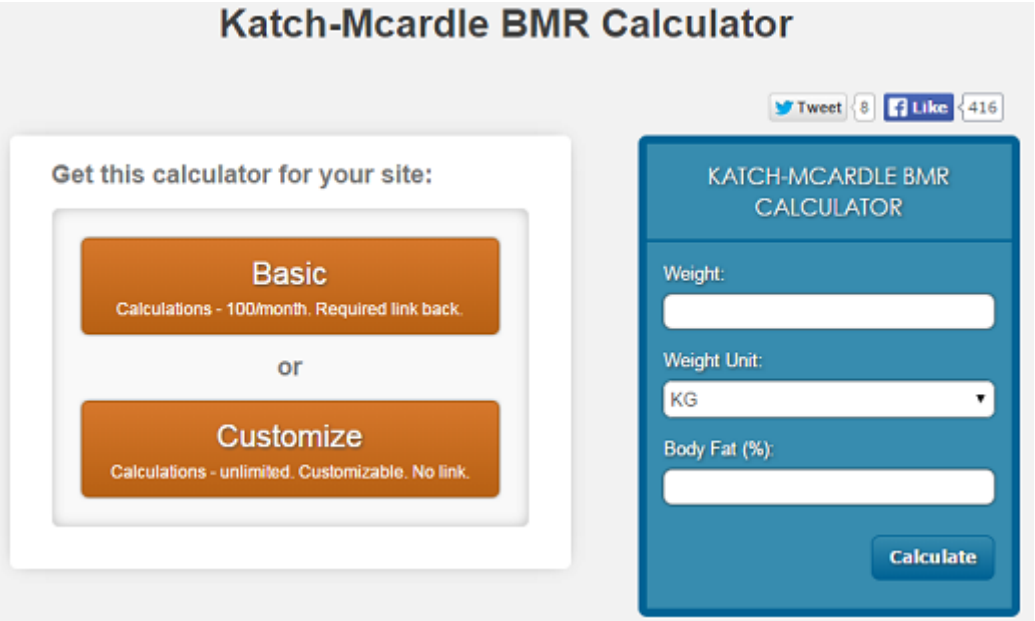


Figura 2.8: Calculadora Katch-McArdle

• **Relevância:**

As fórmulas de Kath-McArdle têm em conta apenas a *Lean Body Mass* (FFM) nos seus cálculos, ignorando factores como altura, idade e género que podem produzir um erro absoluto de até 4% na massa gorda livre dentro de um grupo de estudo homogéneo [18].

Salienta-se a possibilidade de incorporação da calculadora em qualquer *website* desenhado pelo utilizador, com limite diário de utilizações e possibilidade de evoluir a conta de *free* para *premium*.

2.2.2 Model My Diet

O *website* funciona como um simulador do corpo humano gráfico que depende da altura, tipo de corpo e peso introduzidos pelo utilizador. Tem como comparação outra figura que representa o peso desejado do utilizador [19].

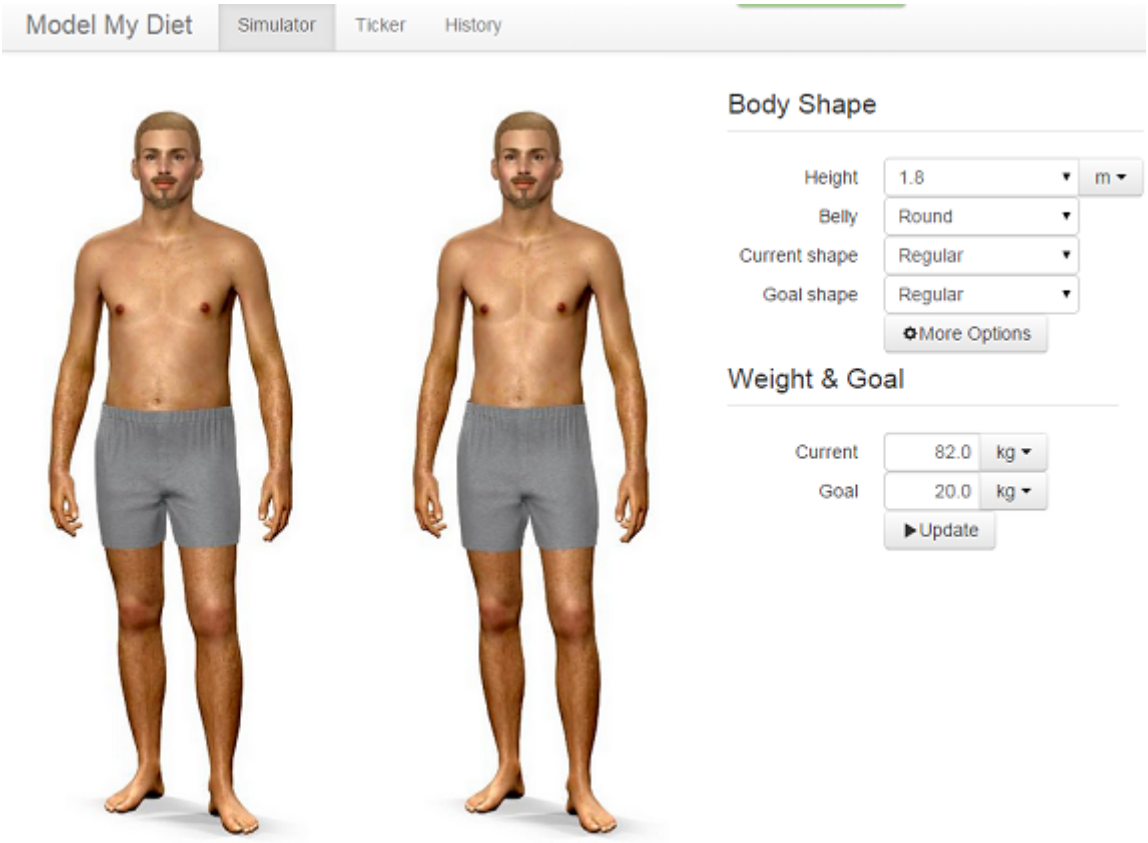


Figura 2.9: Interface Model My Diet

• **Relevância:**

A ideia de visualizar o corpo com o peso desejado num avatar com características semelhantes ao utilizador pode trazer vantagens uma vez que facilita o reajuste das espectativas do mesmo. Neste *website* é apresentada uma relação linear entre a morfologia do avatar e o peso. Visando uma futura aplicação, existe espaço de progressão para a interligação da transformação presente neste website aliada a outros parâmetros mais avançados, ou seja, junto a equações mais complexas associadas à perda de peso. Por exemplo, a introdução da variável percentagem de gordura corporal actual como parâmetro permitiria aproximar com mais precisão a morfologia do avatar ao corpo do utilizador.

2.2.3 Calories Per Day

Este *website* é um exemplo de muitos que calculam as calorias despendidas por dia dando automaticamente o número de calorias necessário para manutenção de peso (têm o mesmo valor). [20]

Figura 2.10: Calculadora Calories Per Day

• **Relevância:**

A calculadora apresentada introduz o nível de actividade praticado no cálculo das calorias despendidas, o que levanta o problema da mensuração desse mesmo nível, dado que é susceptível a um grande grau de subjectividade.

2.2.4 Calories Burned - Activity Calculator

Outra calculadora, *Activity Calculator*, correlaciona o peso, altura e género com o tipo de actividade física e a duração [21].

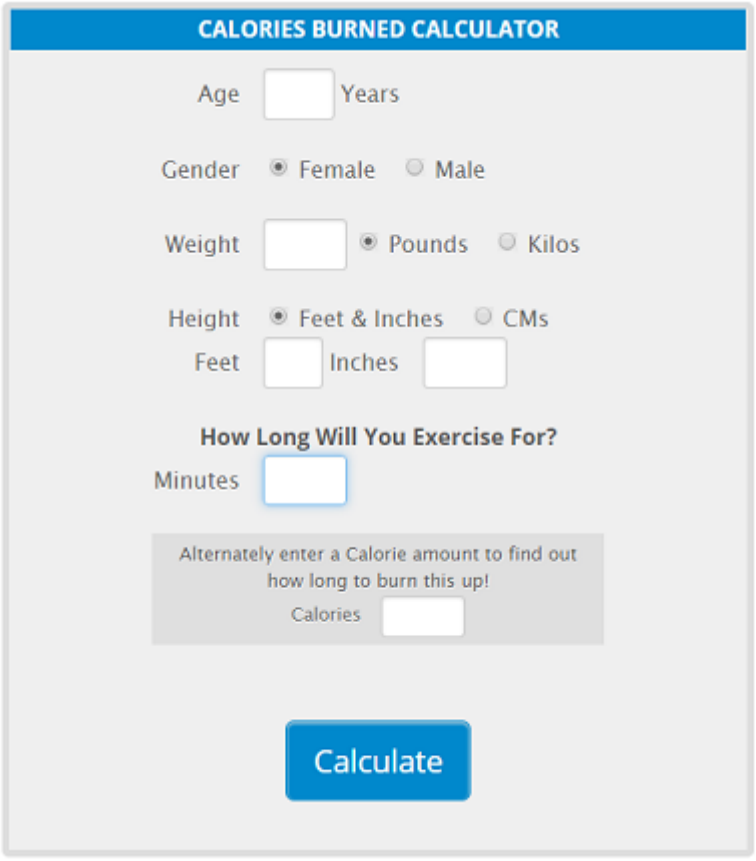


Figura 2.11: Calculadora Calories Burned - Activity

• **Relevância:**

Esta calculadora não só calcula as calorias consumidas sobre determinadas condições, como também devolve o seu inverso, isto é, permite ao utilizador saber quantos minutos tem de exercitar em várias actividades diferentes para despender as calorias pretendidas. Interligando essa informação com um plano de dieta, aproxima o tempo de actividade física necessária para perder o peso indesejado.

2.2.5 Lose It!

No âmbito das aplicações móveis, pode-se mencionar a aplicação Lose It!, um programa disponível para todos os sistemas operativos dos *smart phones* que incorpora na sua versão *premium* vários campos de interesse [22] ilustrados pelas figuras 2.12 - 2.15:

Objectivos Saúde & Fitness	
Saúde	**Medicinal**
Peso	Glucose no Sangue
Gordura Corpural	Pressão Arterial
Hidratação	**Nutrição**
Sono	Hidratos de Carbono
Fitness	Gorduras
Passos	Proteínas
Calorias dos Exercicios	Fibras
Minutos dos Exercicios	Sódio
NikeFuel	**Medidas Corporais**
	Tamanho Pescoço
	Tamanho da Anca
	Tamanho da Cintura
	Tamanho do Bíceps
	Tamanho da Coxa
	Tamanho do Peito

Figura 2.12: Lose It! - Objectivos Saúde e Fitness

Tracking Tools
Telemóvel
Scanner código de barras
Lembretes iOS
Lembretes Android
Mobile + Web
Planeamento de Refeições
Planeamento de Exercício

Figura 2.13: Lose It! - Tracking Tools

Conexões entre Apps&Dispositivos	
App Perda de Peso	**Fitness Apps**
iPhone, iPad & iPod Touch	Nike+ Running
Android	RunKeeper
Kindle & Nook	MapMyFitness
Balanças Wireless	**Apps Sociais**
Withings Scale	Facebook
Fitbit Aria	Twitter
Detectores de Actividade	**Aparelhos Medicinais**
Nike FuelBand	Withings Blood Pressure Monitor
Fitbit One, Force, Flex, Zip & Classic	
Jawbone UP & UP24	

Figura 2.14: Lose It! - Conexões entre Apps e Dispositivos

<table>
<tr><th colspan="2">Motivação & Suporte</th></tr>
<tr><th>Amigos</th><th>Desafios</th></tr>
<tr><td>Encontrar e Juntar-se e Amigos</td><td>Juntar-se a Desafios Públicos</td></tr>
<tr><td>Grupos</td><td>Juntar-se a Desafios Privados e Escondidos</td></tr>
<tr><td>Juntar-se a Grupos Públicos</td><td>Criar Desafios Públicos, Privados e Escondidos</td></tr>
<tr><td>Juntar-se a Grupos Privados e Escondidos</td><td rowspan="2">Participar em Desafios sugeridos pela Aplicação</td></tr>
<tr><td>Criar Grupos Públicos, Privados e Escondidos</td></tr>
<tr><td>Relatórios</td><td></td></tr>
<tr><td>Relatório My Plate</td><td></td></tr>
<tr><td>Email</td><td></td></tr>
</table>

Figura 2.15: Lose It! - Motivação e Suporte

• **Relevância:**

A aplicação Lose It! é um bom exemplo das multifaces da perda de peso. Começando pelas características de Saúde e *Fitness* (figura 2.12) , estas abordam dois parâmetros: a hidratação e a monitorização do sono. Visto que a média de percentagem de água no corpo humano ronda os 50%-65%, a hidratação do corpo tem um papel importante na pesagem. E não só, apenas uma desidratação de 1% já incapacita a performance física e mental do ser humano [23]. A privação do sono além de afectar a performance influencia a tomada de decisões e desinibe os impulsos instintivos, levando a más resoluções e exageros alimentares [24].

Quanto às características de *Tracking* (figura 2.13) , a aplicação tem uma funcionalidade de leitura de códigos de barras dos alimentos, que, aliado à sua base de dados nutricional, completa a informação relevante a cada refeição. Um problema ainda por resolver é a imprecisão nas quantidades ingeridas.

A nível de conectividade (figura 2.14) , é uma mais valia ser possível interligar-se com outras *apps* líderes do mercado (RunKeeper por exemplo) e dispositivos mais precisos nas medições biométricas (como a balança FitBit Aria).

Considerando o efeito de Hawthorne [25], dá-se significado à última tabela de características: Motivação e Suporte (figura 2.15) . O facto de o utilizador saber que publicou as suas metas a

amigos ou familiares ou mesmo desconhecidos é considerada uma motivação extra para cumprir os objectivos.

2.2.6 The Body Weight Simulator

O seguinte simulador é uma aplicação Java que permite a variação de cenários, entre os quais diferentes planos dietéticos e físicos sequenciais [26].

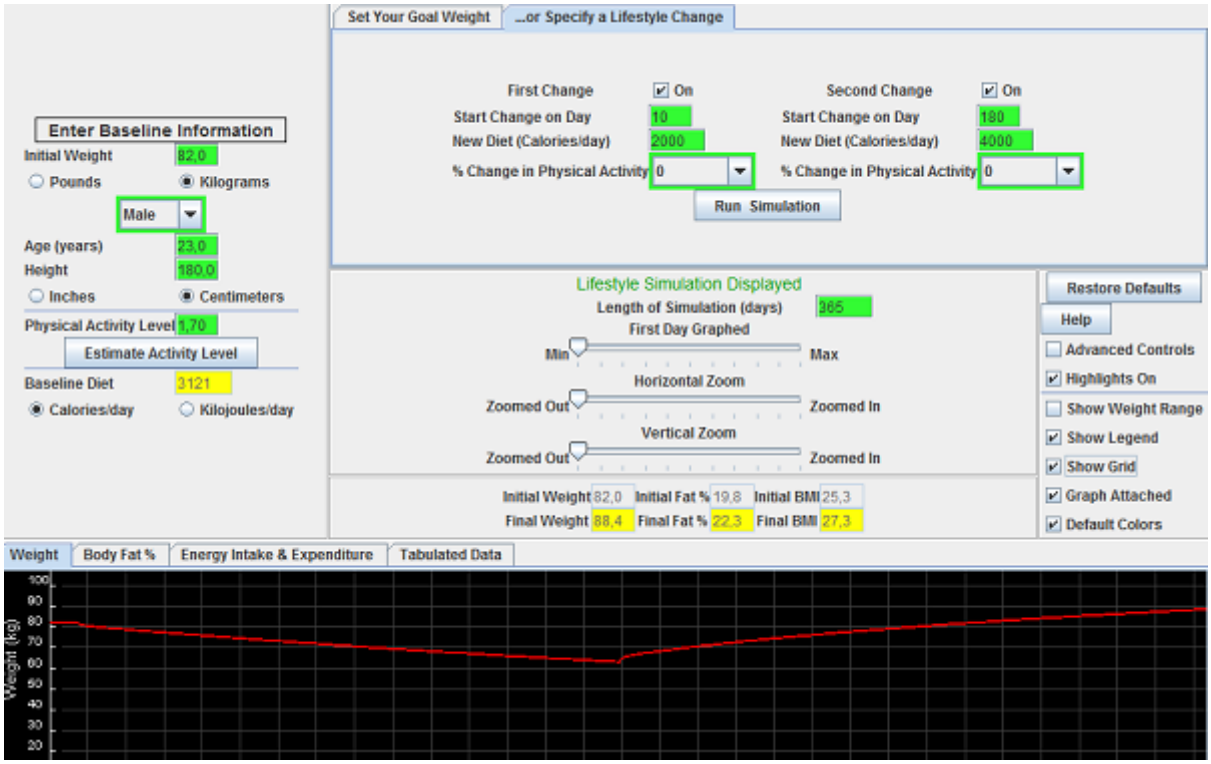


Figura 2.16: Interface Body Weight Simulator

• **Relevância:**

Este simulador foi desenvolvido por um grupo de Investigação no National Institute of Diabetes and Digestive and Kidney Diseases vindo contrariar as recomendações do Serviço Nacional da Saúde do Reino Unido e Estados Unidos da América e a Associação Americana Dietética, recomendações essas que integram o senso comum no que toca à perda de peso. Nessas recomendações foi anunciada uma noção de que se deve ter um deficit de 3500 kcal de modo a perder 1 *pound* (aproximadamente 0.4536 kg). Ora essa regra ignora adaptações fisiológicas dinâmicas para a mudança de peso (emagrecimento, ganho de massa muscular, etc.) que provocam alterações tanto no metabolismo em repouso como nos gastos energéticos da actividade física [27], inviabilizando regras tão generalistas como as mencionadas. De seguida, é apresentado o modelo que deu origem a este simulador, especificamente à primeira fase da mudança de peso e consequente repartição energética.

2.2.6.1 Primeira Fase na Mudança de Peso

Em [14] foi proposto um modelo matemático para descrever com precisão as mudanças iniciais nas primeiras semanas de uma dieta de redução energética (secção 2.1.3.1), onde se calculam os efeitos no glicogénio e fluidos corporais.

No glicogénio é onde estão armazenados os hidratos de carbono, responsáveis pelo balanço energético do corpo. Podem ser de dois tipos: rápida absorção e absorção lenta, sendo que o primeiro está associado ao aumento de peso. Na seguinte equação é demonstrada a dinâmica do Glicogénio:

$$\rho_G \times \frac{dG}{dt} = CI - k_G \times G^2, \tag{2.4}$$

$$k_G = \frac{CI_b}{G_{init}^2}, \tag{2.5}$$

onde:

* G - Glicogénio (Kg);
* ρ_G - Densidade energética do Glicogénio (17.6 MJ/kg);
* CI - Ingestão de Hidratos de Carbono (Kg);
* k_G - Parâmetro Correctivo;
* CI_b - *Baseline* da Ingestão de Hidratos de Carbono (Kg);
* G_{init} - Glicogénio Inicial (Kg).

2.2.6.2 Repartição Energética entre Gordura Corporal e Massa Corporal Magra

Torna-se então necessário saber, quanta da energia disponível (2.4) é utilizada nas mudanças da Gordura Corporal (FM) e Massa Corporal Magra (FFM):

$$\rho_{FM} \times \frac{dFM}{dt} = (1-p)(EI - EE - \rho_G \times \frac{dG}{dt}), \tag{2.6}$$

$$\rho_{FFM} \times \frac{dFFM}{dt} = p(EI - EE - \rho_G \times \frac{dG}{dt}), \tag{2.7}$$

$$p = \frac{C}{C + FM}, \tag{2.8}$$

$$C = 10.4 \times \frac{\rho_{FFM}}{\rho_{FM}}, \tag{2.9}$$

onde:

* ρ_{FM} - Energia por unidade de mudança de massa em gordura corporal (39,5 MJ/kg);
* ρ_{FFM} - Energia por unidade de mudança de massa corporal magra (7,6 MJ/kg);
* p - Proporção de Repartição de Energia (adimensional);
* EI - Entrada de Energia (kcal);

* EE - Consumo de Energia (kcal).

2.2.6.3 Variação do Consumo de Energia com a Actividade Física

Introduzindo a actividade física na modelação, pode-se verificar os efeitos directos no peso corporal, calculando primeiramente EE:

$$EE = K + \gamma_{FM} \times FM + \gamma_{FFM} \times FFM + \delta BW + TEF + AT + \eta_{FFM}\frac{dFFM}{dt} + \eta_{FM}\frac{dFM}{dt}, \quad (2.10)$$

$$TEF = \beta_{TEF}\Delta EI, \quad (2.11)$$

$$\tau_{AT}\frac{dAT}{dt} = \beta_{AT}\Delta EI - AT, \quad (2.12)$$

$$\delta = ((1-\beta_{TEF}) \times PAL - 1) \times \frac{RMR}{BW}, \quad (2.13)$$

onde:

* K - Constante determinada pela condição de balanço energético inicial;
* γ_{FM} - Coeficiente de regressão relativo ao metabolismo em repouso versus gordura corporal (13 kj/kg/dia);
* γ_{FFM} - Coeficiente de regressão relativo ao metabolismo em repouso versus massa magra (92 kj/kg/dia);
* δ - Actividade física (30 kj/kg/dia é a média para uma pessoa sedentária);
* η_{FFM} - Rendimento bioquímico relativo à síntese de massa magra (960 kj/kg);
* η_{FM} - Rendimento bioquímico relativo à síntese de massa gorda (750 kj/kg);
* TEF - Efeito térmico da ingestão (kcal);
* AT - Termogénese adaptativa;
* β_{TEF} - 10%TEF;
* Δ EI - Diferença entre entradas de energia face a novo peso (kcal);
* τ_{AT} - escala temporal para AT (14 dias);
* β_{AT} - 0.14;
* PAL - Nível de Actividade Física (1,5 é a média para sedentários);
* RMR - Taxa Metabólica em Repouso (equações Mifflin-St. Jeor (2.2)).

Caso a Massa de Gordura Corporal inicial (FM) seja desconhecida, utilizam-se as equações de Jackson et al. [28] para a calcular:

- Para Homem:

$$FM(0) = \frac{BWinit}{100} \times (0.14 \times age + 37.31 \times ln(\frac{BWinit \times 10000}{H^2}) - 103.94), \tag{2.14}$$

- Para Mulher:

$$FM(0) = \frac{BWinit}{100} \times (0.14 \times age + 39.96 \times ln(\frac{BWinit \times 10000}{H^2}) - 102.01), \tag{2.15}$$

onde:
* BWinit - Peso Inicial (kg);
* age - idade (anos);
* H - altura (cm).

Sabendo que EE é uma função do grau de mudança de FM e FFM e estes dependem de EE, segue-se a substituição das equações (2.6) e (2.7) na equação (2.10) :

$$EE = \frac{K + \gamma_{FM} FM + \gamma_{FFM} FFM + \delta BW + TEF + AT + (EI - \rho_G \frac{dG}{dt})(p\frac{\eta_{FFM}}{\rho_{FFM}} + (1-p)\frac{\eta_{FM}}{\rho_{FM}})}{1 + p\frac{\eta_{FFM}}{\rho_{FFM}} + (1-p)\frac{\eta_{FM}}{\rho_{FM}}}. \tag{2.16}$$

Com este conjunto de equações o simulador da figura 2.17 é capaz de desenhar a curva que vai do instante zero (onde começa a dieta) até o momento onde acaba a primeira fase da perda de peso, que dura neste caso em particular de 1 a 2 dias.

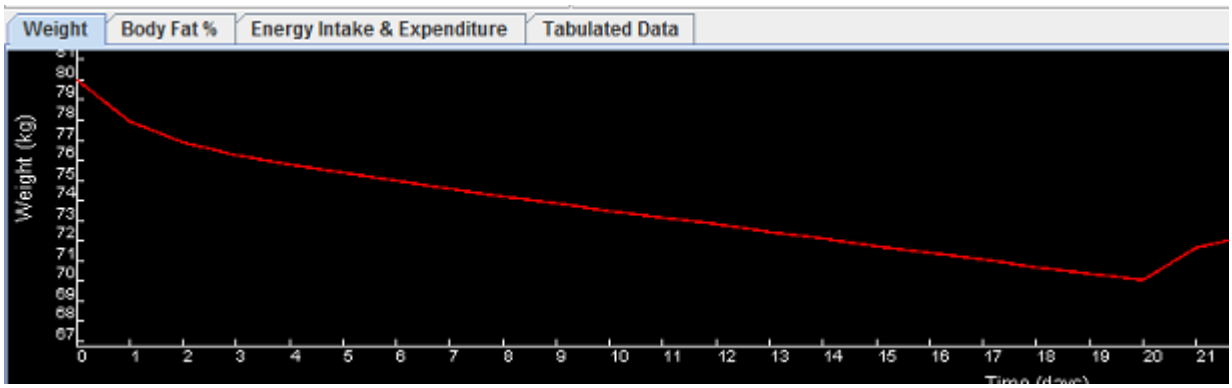


Figura 2.17: Excerto da Interface Body Weight Simulator - 80kg para 70kg em 20 dias

2.3 Ferramentas de Controlo e Software

Nesta secção é apresentada a definição, exemplo e formulação de Modelos de Controlo Predictivo. Posteriormente, é realizada uma descrição conceptual dos ambientes de programação utilizados neste projecto e as suas aplicações.

2.3.1 MATLAB

MATLAB (*Matrix Laboratory*) é uma linguagem de alto nível que foi originalmente concebida (finais de 1970) para facilitar a comunicação com os primeiros software de computação de matrizes. Com a sua disseminação em ambiente universitário, foi sendo desenvolvido através dos inputs dos utilizadores e adaptado mais tarde para uma linguagem mais semelhante a C. [29]

É utilizado nas mais diversas áreas e tarefas, tais como:

- Matemática e Computação;
- Desenvolvimento de Algoritmos;
- Modelação, Simulação e Prototipagem;
- Análise de Dados, exploração e visualização;
- Gráficos Científicos (incluindo engenharia);
- Desenvolvimento de Aplicações, incluindo criação de Interfaces Gráficas para o Utilizador.

O MATLAB tornou-se numa ferramenta bastante versátil devido às suas aplicações especificas a certos problemas, denominadas *toolboxes*. As *toolboxes* permitem aprender e aplicar tecnologia especializada através da sua colecção de funções MATLAB (M-files) em problemas direccionados para áreas como processamento de sinal, sistemas de controlo, redes neurais, lógica difusa, transformadas de Wavelets e muitas outras.

2.3.1.1 SIMULINK

Simulink é um dos add-ons mais populares do *Matlab*, pois permite que o utilizador programe através de blocos pré-existentes ou blocos criados pelo próprio. Esses blocos são uma representação gráfica de: variáveis, operadores, funções, fontes, subsistemas, conversores, roteamento, etc, divididos pelas seguintes categorias [30]:

- Blocos de Funções Contínuas como Derivador e Integrador;
- Blocos de Funções Descontínuas como Saturação;
- Blocos de Funções de Tempo Discreto como Atraso Unitário;
- Blocos Lógicos ou de Operações de Bits como Operador Lógico e Operador Relacional;

- Blocos de Tabelas de Valores como Coseno e Seno;
- Blocos de Funções Matemáticas como Ganho, Produto e Soma;
- Blocos para Modelos Auto Verificáveis como Verificador da Resolução de Entradas;
- Blocos de Operações *Model-wide* como *Model Info* e Bloco de Tabelas de dados Suportados;
- Blocos relacionados com Subsistemas como *Inport*, *Outport*, *Subsystem*, e *Model*;
- Blocos de Modificação de Atributos de Sinal como Conversores de Tipo de Dados;
- Blocos de Roteamento de Sinais como *Bus Creator* e Interruptores;
- Blocos de *Display* ou Exportação de Sinais como *Scope* e *To Workspace*;
- Blocos para Gerar ou Importar Sinais como Onda de Seno e From Workspace;
- Blocos de Funções Costumizáveis como *MATLAB Function* e Simulink Function;
- Blocos de Funções Matemáticas e Discretas como Decrement *Stored Integer*;
- Blocos de Optimização HDL.

O facto de se poder visualizar os modelos durante a sua criação facilita o desenvolvimento e compreensão dos mesmos (especialmente em modelos maiores e mais complexos) pois pode-se dividir o modelo em subsistemas que incluem informação que não requer análise contínua e detalhada.

2.3.1.2 ICLOCS

ICLOCS (*Imperial College London Optimal Control Software*) é um código implementado em MATLAB que permite aos utilizadores definir e resolver problemas de controlo óptimo dada uma trajectória, limites restringentes e tempo final livre ou fixo. É também possível incluir parâmetros de design contínuos como parâmetros desconhecidos [31]. Esta ferramenta transforma problemas de controlo óptimo em problemas de optimização que são resolvidos pelo IPOPT (consultar 2.3.1.2.1).

Os problemas de controlo óptimo obedecem à seguinte formulação [32]:

$$min_{u(t),t_f,p,x_0} \int_{t_0}^{t_f} L(x(t),u(t),p,t)dt + E(x_0,x_f,u_0,u_f,p,t_f)dt, \tag{2.17}$$

$$\text{sujeito a: } \dot{x} = f(x(t),u(t),p,t) \qquad \forall t \in [t_0,t_f], \tag{2.18}$$

$$g_L \leq g(x(t),u(t),p,t) \leq g_U \qquad \forall t \in [t_0,t_f], \tag{2.19}$$

$$\phi_L \leq \phi(x_0,x_f,u_0,u_f,p,t_f) \leq \phi_U \qquad \forall t \in [t_0,t_f], \tag{2.20}$$

$$x_L \leq x(t) \leq x_U \qquad \forall t \in [t_0,t_f], \tag{2.21}$$

$$u_L \leq u(t) \leq u_U \qquad \forall t \in [t_0,t_f], \tag{2.22}$$

$$p_L \leq p \leq p_U \qquad . \tag{2.23}$$

onde:

* L(·) - função custo integral;
* E(·) - função custo associada aos estados finais e iniciais;
* x - variáveis de estado;
* $\dot{x} = f(\cdot)$ - dinâmica dos estados;
* g - *general path constraints*;
* g_L - limite inferior das *general path constraints*;
* g_U - limite superior das *general path constraints*;
* ϕ - *boundary conditions*;
* ϕ_L - limite inferior das *boundary conditions*;
* ϕ_U - limite superior das *boundary conditions*;
* x_L - limite inferior das variáveis de estado;
* x_U - limite superior das variáveis de estado;
* u - variáveis de controlo;

* u_L - limite inferior das variáveis de controlo;

* u_U - limite superior das variáveis de controlo;

* p - conjunto de parâmetros;

* p_L - limite inferior do conjunto de parâmetros;

* p_U - limite superior do conjunto de parâmetros.

O problema (2.17) - (2.23) diz-se de tempo fixo quando t_f e t_0 estão fixos. A função custo (2.17) pode ser minimizada em função do controlo, tempo final, conjunto de parâmetros e estado inicial.

As funções L e E podem ser lineares ou não lineares, convexas ou não convexas mas devem ser pelo menos uma vez (idealmente duas vezes) diferenciáveis.

2.3.1.2.1 IPOPT

IPOPT (*Interior Point Optimizer*) apresenta-se como um pacote de *software open source* para optimizações não lineares de problemas de grande escala. IPOPT pode ser utilizado como biblioteca para códigos da linguagem C, C++, Fortran, Java e também como um *solver* para a modelação AMPL (*A Mathematical Programming Language*), O pacote inclui interfaces com CUTEr(*Constrained and Unconstrained Testing Environment, revisited*) e MATLAB, podendo ser utilizado nos sistemas operativos Linux/UNIX, Mac OS X e Windows.

O método de filtragem *interior-point line-search* que o IPOPT utiliza é o que o torna apto para resolver problemas de grande escala, assumindo que a matriz Jacobiana da função restrição é dispersa (mais zeros que não zeros). No entanto, também pode resolver eficientemente problemas mais pequenos e densos. Como o algoritmo tenta encontrar o mínimo local do problema, se este for não convexo, poderão existir vários pontos estacionários com diferentes valores da função objectivo. Esse resultado dependerá do ponto de partida e em como é que o método converge.

2.4 Breve descrição do MPC

Este método consiste em utilizar o modelo de um determinado processo de modo a prever a sua evolução futura e a partir daí, optimizar as suas variáveis de controlo. Na figura seguinte demonstra-se um exemplo simplista de um sistema deste tipo:

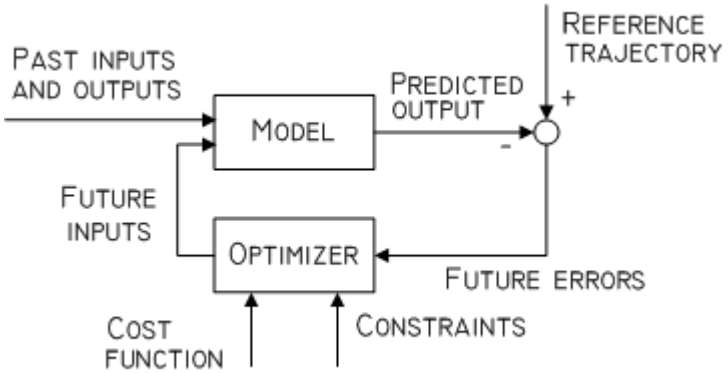


Figura 2.18: Diagrama Model Predictive Control [8]

Observando a figura 2.18 e completando-a com o exemplo descrito nos slides da Faculdade de Penn Engineering [33] sobre *Model Predictive Control*:

- Descrição do problema Dispositivo GPS:

- *prediction model*: como o veículo se movimenta no mapa;

- *reference trajectory*: mapa;

- *constraints*: conduzir em estradas, evitar portagens, etc;

- *disturbances*: desatenção do condutor, etc;

- *set point*: ponto de chegada;

- *cost function*: tempo mínimo, distância mínima, etc;

- *receding horizon mechanism*: o caminho óptimo é recalculado quando o percurso anterior é perdido, ou seja, quando os *outputs* previstos não coincidem com a trajectória referência;

- *future inputs*: ajustes à trajectória no mapa.

O *Model Predictive Control* utiliza processos transformados em expressões matemáticas para antecipar comportamentos. Essas previsões são usadas para optimizar o processo durante um determinado tempo. A aplicação de um controlador MPC obedece aos seguintes passos [34]:

1. Transformação de processos em dinâmica matemática;
2. A qualquer instante t, as entradas e saídas passadas são usadas juntamente com a dinâmica, prevendo os valores futuros do controlo u durante um horizonte de previsão;
3. O sinal de controlo que provoca um comportamento mais desejável é seleccionado após as iteracções necessárias a encontrá-lo;
4. O sinal de controlo é implementado desde t até ao restante horizonte de previsão;
5. O tempo avança até ao próximo intervalo de previsão e os procedimentos repetem-se desde o passo 2.

A implementação de MPC exige assim a resolução numérica de grandes problemas de controlo óptimo . Usualmente os problemas de controlo óptimo são resolvidos pelo método directo, ou seja, são discretizados obtendo-se depois um problema de optimização de larga escala. Problemas de optimização podem ser resolvidos numéricamente. No contexto das dietas, os instantes t_τ, com τ=0,1,...,k são predefinidos e t_0=0, $t_\tau < t_{\tau+1}$, $t_k <$ T.

Para τ=0,1,...,k o problema é o seguinte:

$$\text{Min} \int_{t_\tau}^{T} (y(s) - ref(s))^2 ds, \tag{2.24}$$

$$\begin{aligned} &\text{sujeito a:} \\ &\dot{x}(s) = f(x(s), u(s)), \\ &y(s) = g(x(s)) \qquad s \in [t_\tau, T], \\ &u(s) \in \mathscr{U}, \\ &x(s) \in \mathscr{X}, \\ &x(t_\tau) = \bar{x}_{t_\tau}, \end{aligned} \tag{2.25}$$

onde:

* $\bar{y}(t_\tau)$=g($\bar{x}(\tau)$) - peso real do individuo no instante τ da sua dieta.

Capítulo 3

Modelo Dinâmico da Mudança de Peso

Neste capítulo é descrito o modelo dinâmico de variação de peso adaptado das referências bibliográficas inerentes à dissertação em conjunto com as adequações necessárias provenientes de pesquisas independentes. Em simultâneo, é feita uma contextualização técnica de alguns conceitos de Biologia e de Nutrição de modo a explicar o enquadramento matemático. As equações usadas neste capítulo foram retiradas do artigo [9] , salvo as excepções devidamente referenciadas.

Para desenhar o modelo de variação de peso foi utilizada a ferramenta Simulink do Matlab 2012b. Numa fase inicial, verificou-se benéfico utilizar o Simulink e a sua representação gráfica para compreender melhor a complexa dinâmica do modelo em causa, pois permite modelizar as diferentes partes do mesmo em subsistemas. A outra razão que levou a esta escolha prende-se com o facto de o Simulink possuir blocos de controlo já implementados (por exemplo de lógica *Fuzzy* e *Model Predictive Control*) que permitiriam a sua experimentação. Todas as simulações presentes neste capítulo e no capítulo 4 e 5 utilizam o mesmo ficheiro de inicialização de variáveis que apresenta um sujeito do sexo masculino de 20 anos de idade e 1.80 metros de altura como se pode verificar na figura 3.5. Esta inicialização é totalmente configurável.

3.1 Energia/Calorias Ingerida(s)

A caloria (kcal ou CAL) é a unidade que mede a energia utilizada pelo corpo humano para todas as suas funções voluntárias ou involuntárias do dia a dia.

Todos os alimentos são constituídos por uma combinação de gorduras e/ou hidratos de carbono e/ou proteínas. Através de um controlo alimentar é possível extrair essa informação nutricional e calcular as Calorias Ingeridas (*Energy Intake* - EI) correspondentes:

$$EI(t) = CI(t) \times a1 + FI(t) \times a2 + PI(t) \times a3, \tag{3.1}$$

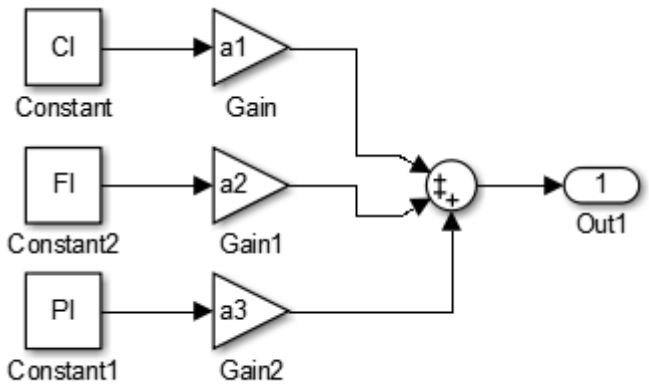


Figura 3.1: Subsistema Simulink de EI

onde:
* CI - (*carbohydrate intake*) ingestão de hidratos de carbono (g/dia);
* a1 - coeficiente energético de CI (4 kcal/g);
* FI - (*fat intake*) - ingestão de gorduras (g/dia);
* a2 - coeficiente energético de FI (9 kcal/g);
* PI - (*protein intake*) - ingestão de proteínas (g/dia);
* a3 - coeficiente energético de PI (4 kcal/g).

Nota de desenvolvimento: Inicialmente, para efeitos de simulação, utilizou-se EI constante através de um gerador de sinais personalizável (bloco *signal builder*). Este foi substituído, em versões posteriores, pela fórmula descrita em (3.1) que está em função da alimentação. As quantidades CI, FI e PI estão representados como constantes na figura 3.1. O coeficiente energético de CI, a1, assume o valor 4 kcal/g, no entanto as tabelas alimentares britânicas utilizam o valor 3,75 kcal/g. Apesar de serem representados por blocos constantes, estes podem ser substituídos por *Data Stores* de modo a obter uma dinâmica contínua. Como para efeitos de simulação se pretendeu um EB constante, EI também aparece representado como constante apesar da sua inicialização ser configurável.

3.2 Gastos Energéticos

Cada actividade a que o Corpo Humano é sujeito tem intensidade e duração variável, tendo cada uma dessas actividades um custo energético associado.

Para o cálculo dos gastos energéticos (*Energy Expenditure* - EE) é necessário considerar três fenómenos distintos:

$$EE(t) = TEF(t) + PA(t) + BMR(t), \tag{3.2}$$

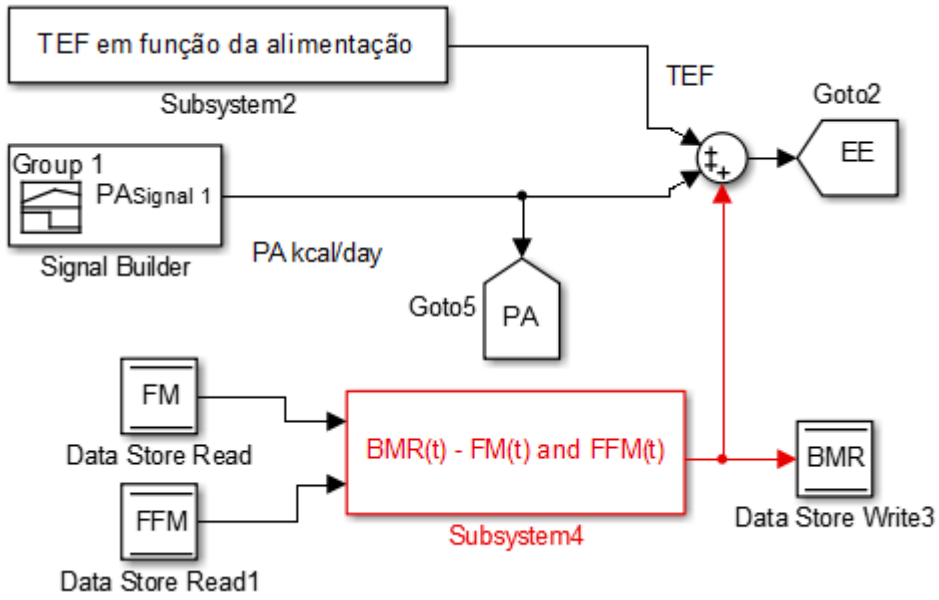


Figura 3.2: Subsistema em Simulink de EE

onde:

* TEF - (*Thermic Effect of Feeding*) - Efeito Térmico da Alimentação, ou seja, a energia despendida por todo o sistema digestivo em função dos alimentos ingeridos (kcal/dia);
* PA - (*Physical Activity*) - Actividade Física (kcal/dia);
* BMR - (*Basal Metabolic Rate*) - Taxa Metabólica Basal, ou seja, energia base de manutenção do corpo humano (kcal/dia).

3.2.1 Efeito Térmico da Alimentação

Dado que o Efeito Térmico da Alimentação (TEF) é proporcional à Energia Ingerida (EI), o cálculo de ambos é semelhante, dado por CI, FI e PI com coeficientes específicos [10] :

$$TEF(t) = ac * CI(t) + af * FI(t) + ap * PI(t), \tag{3.3}$$

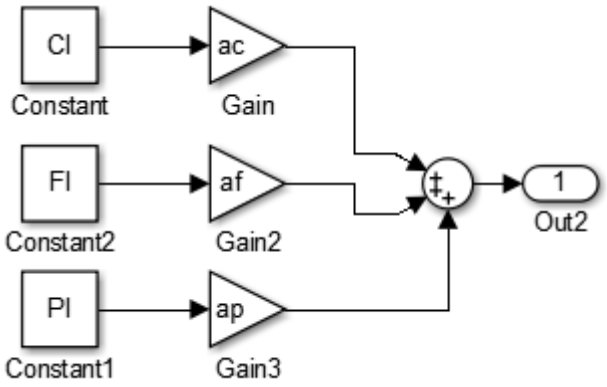


Figura 3.3: Subsistema em Simulink de TEF

onde:

* ac - coeficiente energético de CI (0.075 kcal/g);

* af - coeficiente energético de FI (0.025 kcal/g);

* ap - coeficiente energético de PI (0.25 kcal/g).

Nota de Desenvolvimento: nos primeiros modelos deste trabalho utilizou-se TEF como uma aproximação de cerca de 1/10 de EI, como indicado em [9]. Depois de analisada esta aproximação, TEF foi adaptado para um valor mais preciso em 3.3, estando de momento em função da alimentação (tal como EI).

3.2.2 Taxa Metabólica Basal

A Taxa Metabólica Basal - BMR(t) - é directamente influenciada pela quantidade de massa de cada tipo, FM(t) e FFM(t).

$$BMR(t) = 0.024FM(t) + 0.102FFM(t) + 0.85. \tag{3.4}$$

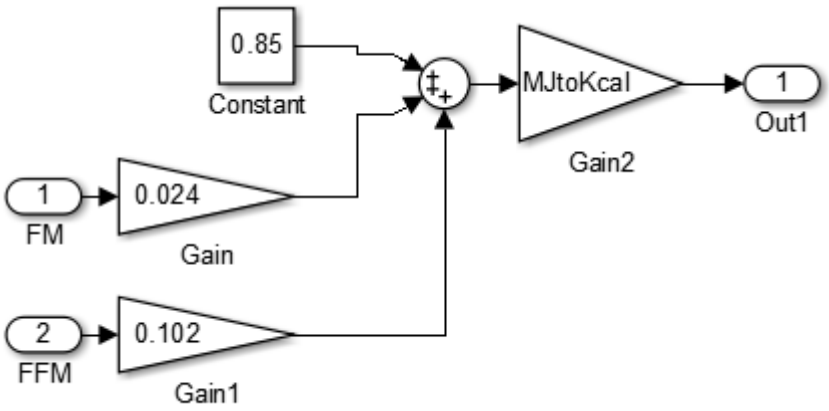


Figura 3.4: Subsistema em Simulink de BMR

Nota de desenvolvimento: A fórmula escolhida para BMR(t) em (3.4) está em MegaJoule (MJ). Como se pode observar na figura 3.4 após o cálculo exemplificado, o resultado final é submetido a um ganho denominado "MJtoKcal"de modo a converter esse resultado para kcal. O valor de conversão "MJtoKcal"utilizado é 238.845896627 e é inicializado juntamente com outras constantes no painel inicial do programa em Simulink. Os valores de FM(t) e FFM(t) advêm das equações (3.17) e (3.18).

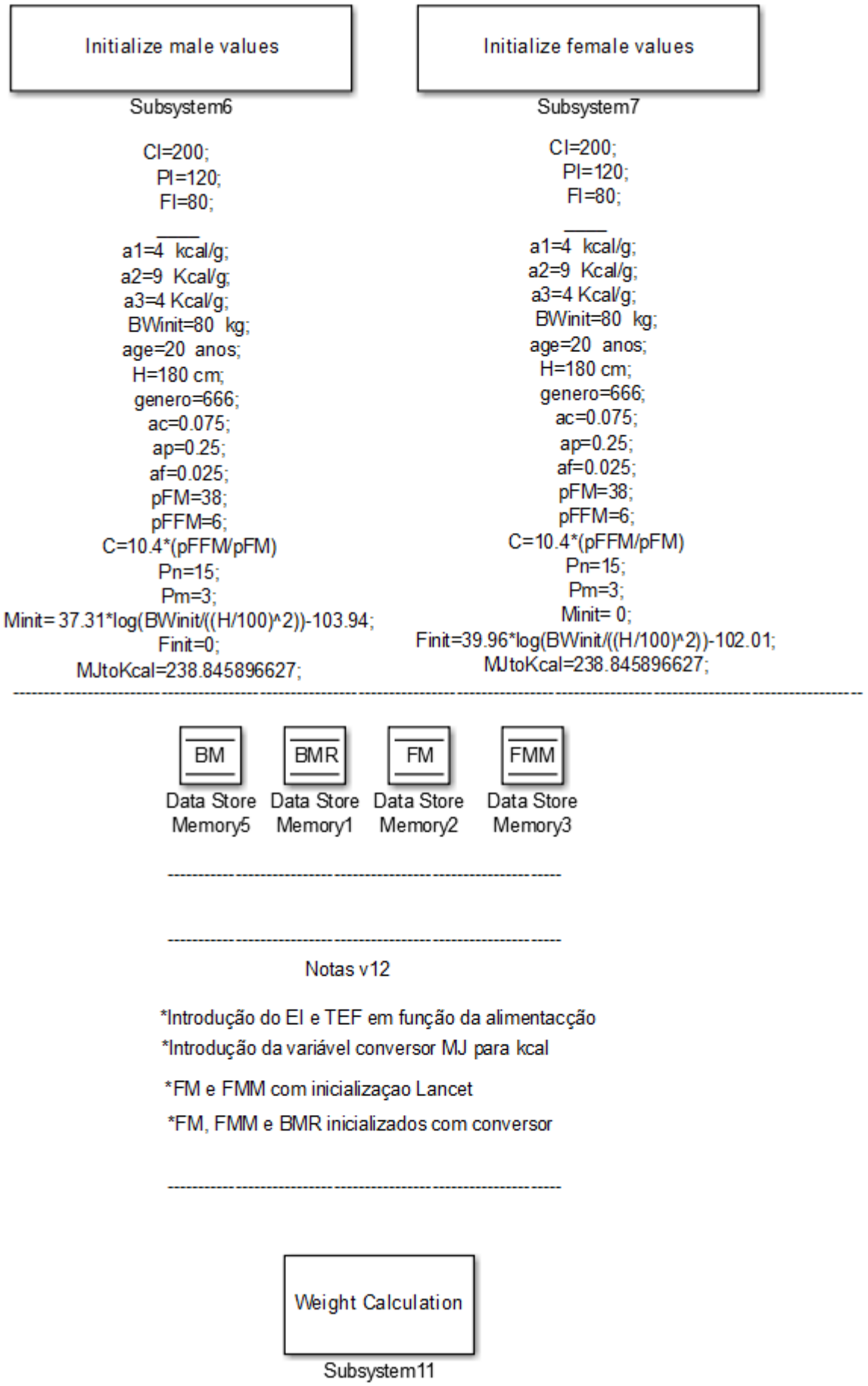


Figura 3.5: Painel Inicial do programa em Simulink

3.2.3 Actividade Física

De modo a mensurar correctamente a Actividade Física, é necessário contabilizar todas as actividades executadas ao longo do dia, a duração e intensidade das mesmas. Recorrendo ao artigo [16], pode-se extrair a escala MET (*Metabolic Equivalent of Task*) de modo a completar a fórmula do Nível de Actividade Física (*Physical Activity Level* - PAL) [35] :

$$\Delta PAL = (MET's - 1) \times \frac{1.15}{0.9} \times \frac{Duration(min)}{1440} \times \frac{0.0175 \times 1440 \times weight(kg)}{BEE}. \tag{3.5}$$

A equação (3.5) deve ser aplicada para cada actividade física que o utilizador introduza. A variável BEE (*Basal Energy Expenditure*) é calculada consoante o género e no artigo [35] aparece descrita como:

- Para Homem:

$$BEE = 2933.8 \times age(years) + 456.4 \times height(meters) + 10.12 \times weight(kg), \tag{3.6}$$

- Para Mulher:

$$BEE = 2472.67 \times age(years) + 401.5 \times height(meters) + 8.6 \times weight(kg). \tag{3.7}$$

Utilizando a variável ΔPAL e adicionando-lhe um factor correctivo, obtém-se:

$$PA = \Delta PAL + 1.1. \tag{3.8}$$

Nota de Desenvolvimento: BEE tem o mesmo significado físico que BMR, a diferenciação é feita apenas por referência aos distintos artigos que utilizam diferentes fórmulas e por consequência, diferentes coeficientes multiplicativos. Nas simulações desenvolvidas, PA é um input proveniente de um gerador de sinais, visto que os cálculos necessários estão dependentes dos valores tabulados MET.

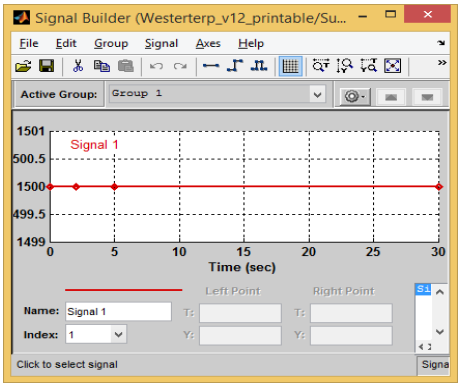


Figura 3.6: Gerador de PA

3.3 Balanço Energético

Escalonando o tempo em dias e numa perspectiva de sistema, torna-se percetível que se deve calcular a diferença entre as calorias que entram e as calorias que saem de modo a verificar rendimento energético. Essa diferença resulta no Balanço Energético *(Energy Balance* - EB) diário do indivíduo (kcal/dia).

$$EB(t) = EI(t) - EE(t). \tag{3.9}$$

No entanto, as variações de peso têm efeito directo na maneira como o Balanço Energético é calculado:

$$EBw(t) = \begin{cases} EB(t) + EER(t) & \text{se } EB < 0, \\ EB(t) + EC(t) & \text{se } EB > 0. \end{cases} \tag{3.10}$$

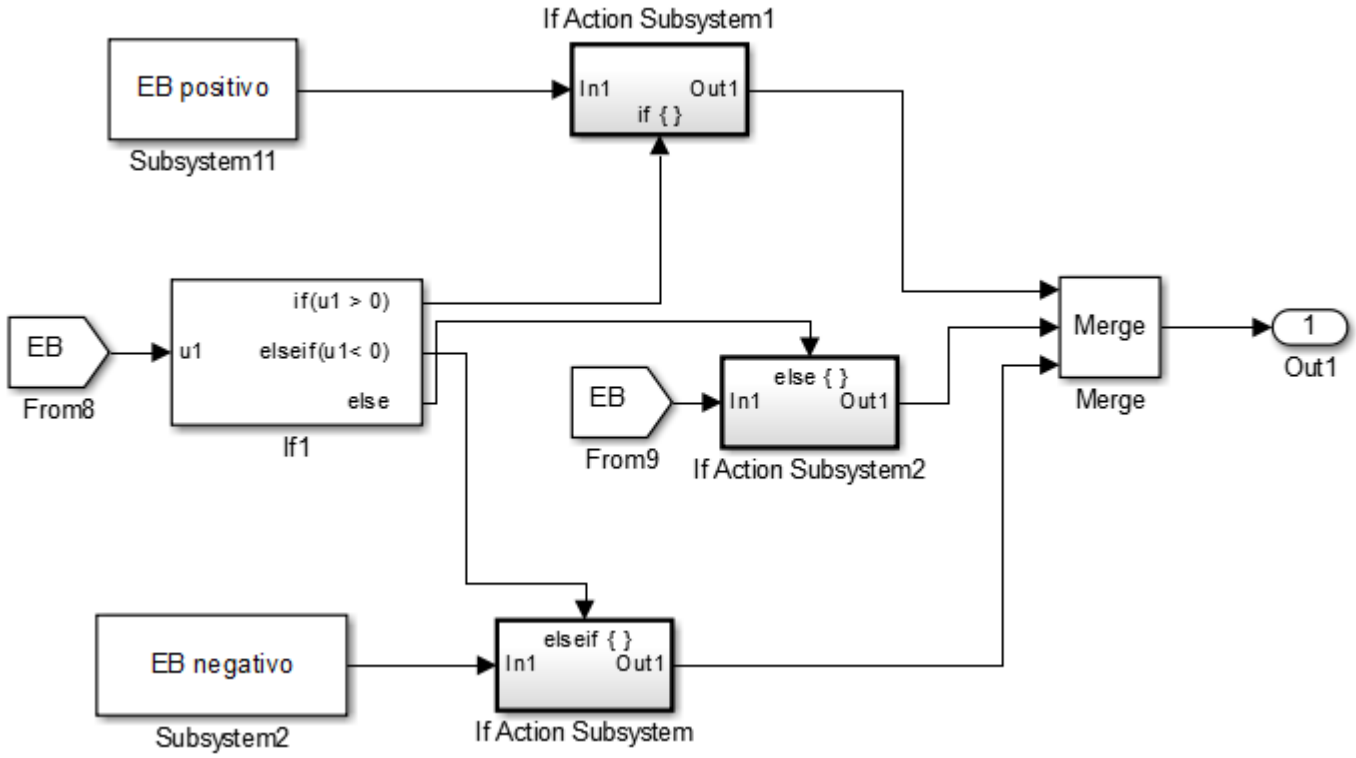


Figura 3.7: Subsistema em Simulink de EBw

onde:

* EER(t) - (*Energy Expenditure Reduction*) - Redução nos Gastos Energéticos (kcal);

* EC(t) - (*Energy Cost*) - Custos Energéticos (kcal).

No caso de perda de peso, isto é, de massa corporal (Body Mass - BM), existe simultâneamente uma Redução nos Gastos Energéticos associados a uma menor massa. Logo, para uma massa menor os gastos energéticos de manutenção dessa mesma massa são inferiores aos gastos

energéticos da massa anterior à perda de peso. Essa relação é descrita pela equação:

$$EER(t) = e(t)BMR(t), \tag{3.11}$$

onde e(t) é o coeficiente de gastos energéticos que depende da relação EI/EE:

$$e(t) = \begin{cases} 0 & \text{se } EI(t)/EE(t) \geq 1, \\ 0.1 & \text{se } EI(t)/EE(t) \leq 0.5, \\ 0.2 \times (1 - EI(t)/EE(t)) & \text{se } 0.5 < EI(t)/EE(t) < 1. \end{cases} \tag{3.12}$$

Verificando-se perda de peso, o balanço energético é negativo (primeiro caso do sistema (3.10)) e pode ser representado pela seguinte figura:

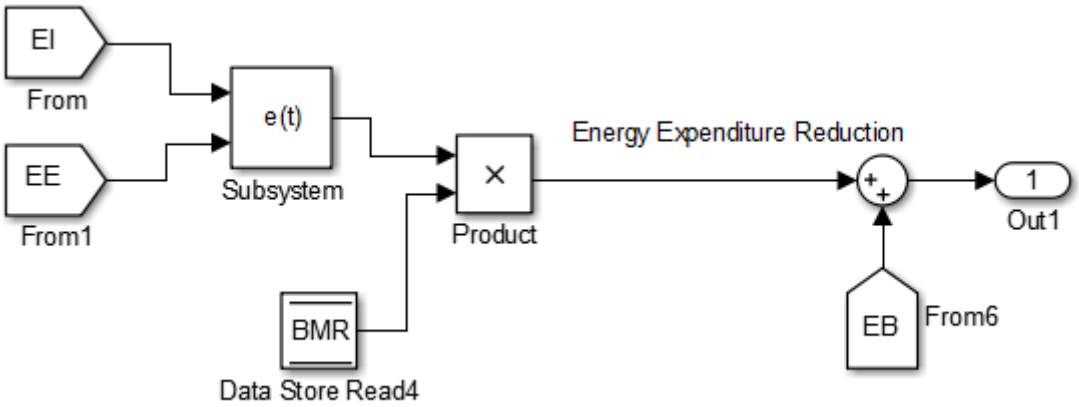


Figura 3.8: Subsistema em Simulink de EB negativo

Simetricamente, no caso de ganho de peso, têm de ser equacionados os Custos Energéticos extraordinários direccionados ao armazenamento da massa extraordinária.

$$EC(t) = h(t) \times (EI(t) - EE(t)), \tag{3.13}$$

onde h(t) é o coeficiente de conversão energética:

$$h(t) = \begin{cases} 0 & \text{se } EI(t) \leq \text{EE(t)}, \\ 0.1 & \text{se } EI(t) > \text{EE(t)}. \end{cases} \tag{3.14}$$

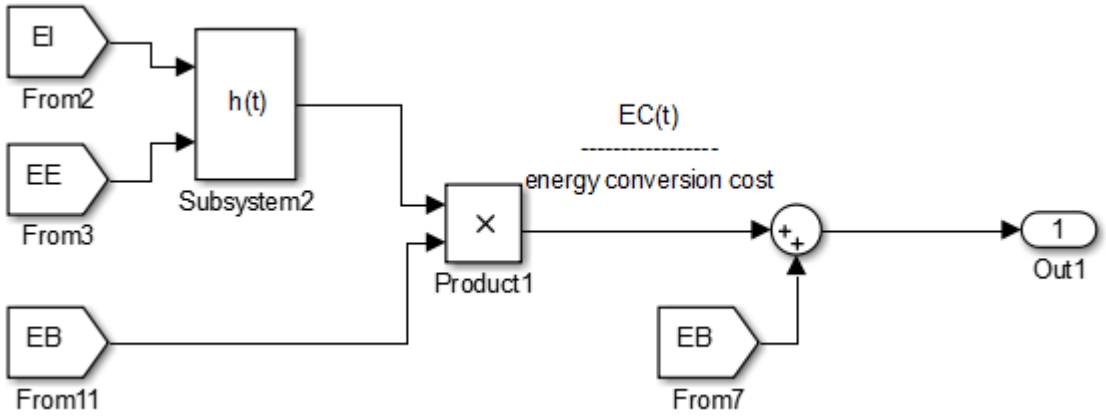


Figura 3.9: Subsistema em Simulink de EB positivo

Nota de Desenvolvimento: apesar de h(t) ser dado por (3.14) proveniente do artigo [9], verifica-se que h(t) só é utilizado para o cálculo de EC(t) e este presume um EB positivo. Logo, EI(t) é sempre maior que EE(t), sendo que o bloco de h(t) pode ser restringido ao valor do segundo caso, a constante 0.1 do sistema referenciado.

3.4 Repartição de Massa

Consoante o balanço energético diário, o Corpo Humano distribui a energia para todas as actividades necessárias ao seu funcionamento, entre as quais a reestruturação e equilíbro do seu peso. Essa divisão depende de o balanço energético ser positivo ou negativo, significando em cada caso a produção de massa ou perda da mesma.

3.4.1 Variação de Ganhos

O que determina a existência de casos de produção ou perda de massa são os os Ganhos (positivos ou negativos) de FM ou FMM:

$$Ganho_{FM} = \frac{(1-p(t)) \times EBw(t)}{\rho_{FM}}, \tag{3.15}$$

$$Ganho_{FFM} = \frac{p(t) \times EBw(t)}{\rho_{FFM}}, \tag{3.16}$$

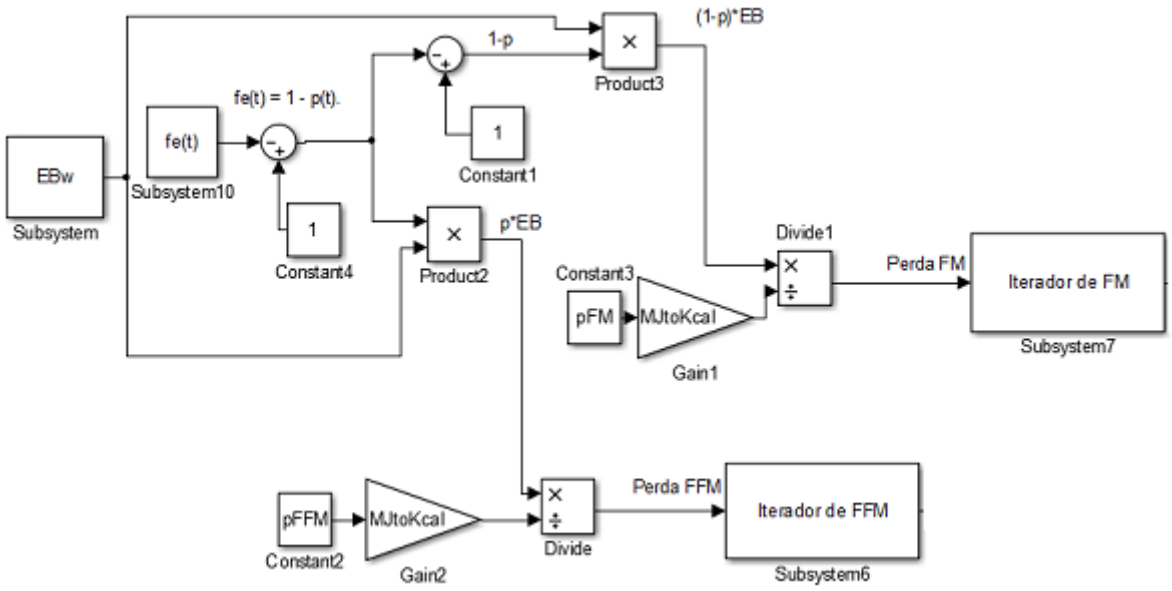


Figura 3.10: Subsistema em Simulink de Ganhos de FM(t) e FFM(t)

onde:

* p(t) - proporção da energia do Balanço Energético direccionada para transformações de FFM (equações (3.19) , (3.20) e (3.21));
* ρ_{FM} - densidade energética de FM (38 MJ/kg) ;
* ρ_{FFM} - densidade energética de FFM (6 MJ/kg) .

O cálculo do estado actual (dia t) de uma determinada massa é passível de ser extraído das equações (3.15) e (3.16) , adicionando o ganho desse instante ao seu estado anterior.

$$FM(t) = \frac{(1-p(t)) \times EBw(t)}{\rho_{FM}} + FM(t-1), \tag{3.17}$$

$$FFM(t) = \frac{p(t) \times EBw(t)}{\rho_{FFM}} + FFM(t-1). \tag{3.18}$$

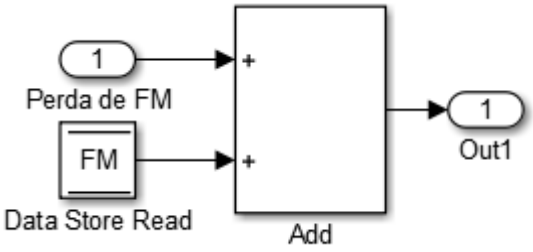


Figura 3.11: Iterador de FM(t)

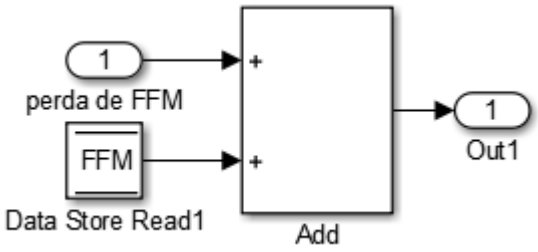


Figura 3.12: Iterador de FFM(t)

Por analogia, verifica-se que 1-p(t) é a proporção de EBw que actuará na massa que será transformada em Massa Gorda. A proporção 1-p(t) é calculada no artigo [9] na condição de uma nova variável, o factor energético de gordura (*fat energy factor* - fe):

· Para $P_f(t) < P_m$

$$fe(t) = 0, \tag{3.19}$$

· Para $P_f(t) > P_n$

$$fe(t) = \begin{cases} 0.95 & \text{se } g(t) \geq 0, \\ g(t) + 0.9 & \text{se } g(t) < 0, \end{cases} \tag{3.20}$$

· Para $P_m \leq P_f(t) \leq P_n$

$$fe(t) = \begin{cases} -0.2375 + 0.08 \times P_f(t) & \text{se } g(t) \geq 0, \\ -0.2375 + (g(t) + 0.9) \times (0.0833 \times P_f(t) - 0.25) & \text{se } g(t) < 0, \end{cases} \tag{3.21}$$

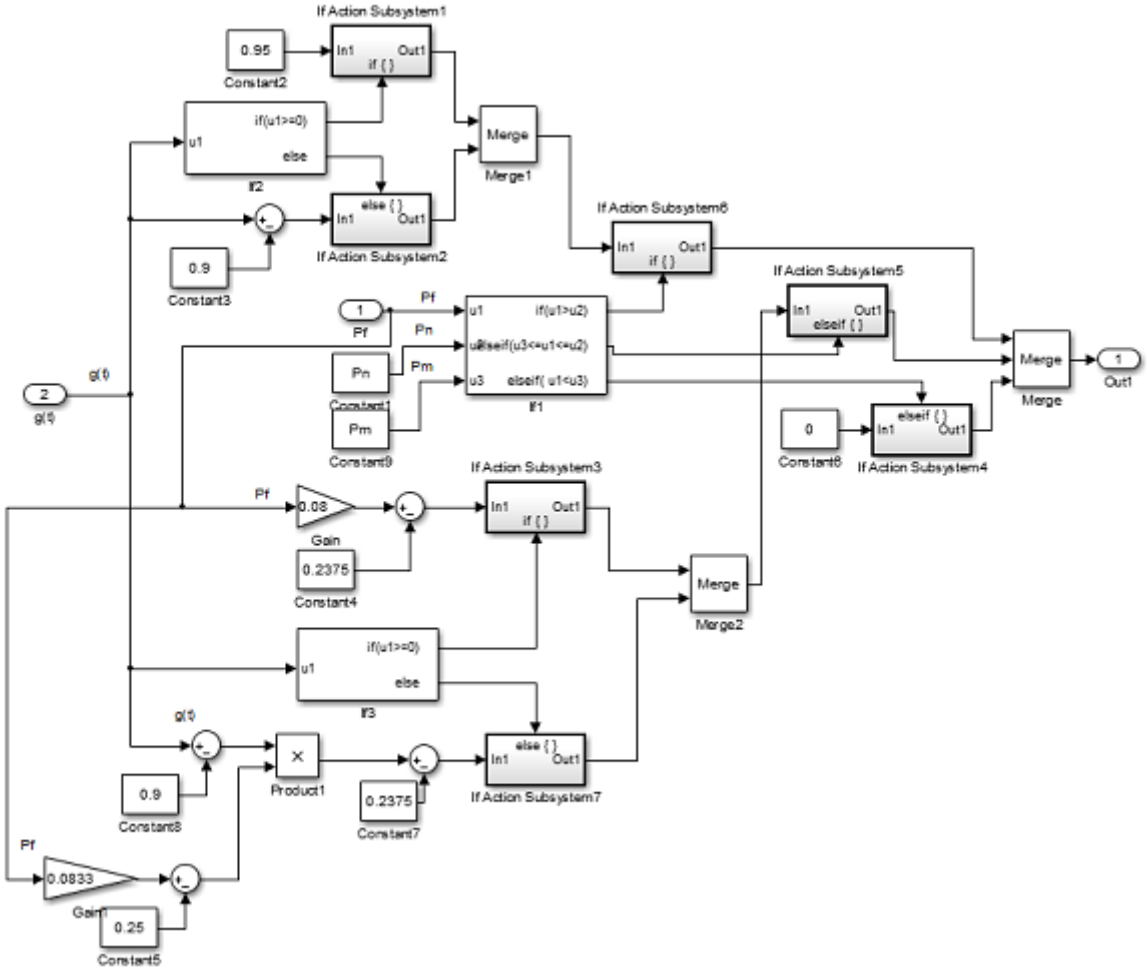


Figura 3.13: Subsistema em Simulink de fe(t)

onde:

* $P_f(t)$ - percentagem de massa gorda actual (equação (3.22));
* P_n - percentagem de massa gorda normal (15 %);
* P_m - percentagem de massa gorda mínima (3 %);
* g(t) - ganho alimentar actual (consultar subsecção 3.4.2).

Se $P_f(t)$ não for dado à partida, pode ser calculado através da seguinte fórmula:

$$P_f(t) = 100 \times \frac{FM(t)}{BM(t)}. \tag{3.22}$$

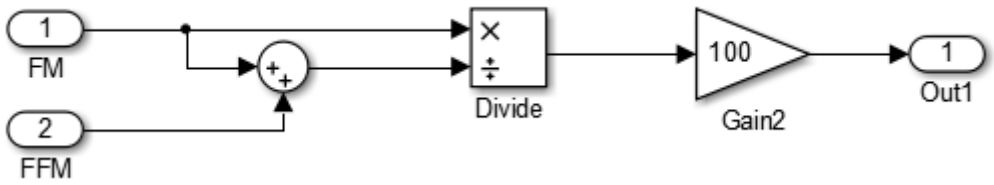


Figura 3.14: Subsistema em Simulink de P_f

3.4.2 Ingestão Calórica para Manutenção

A variável g(t) representa o ganho alimentar actual, que relaciona EI com EI_0, a variável que traduz a quantidade de energia necessária para se manter determinado peso objectivo. Essa manutenção implica EB=0 (consultar subsecção 2.1.3.2), logo implica que EI seja igual a EE.

Substituindo EE por EI_0 em (3.2) e TEF por $0.1 \times EI_0$ (estimativa), obtém-se:

$$EI_0(t) = \frac{PA(t) + BMR(t)}{0.9}. \tag{3.23}$$

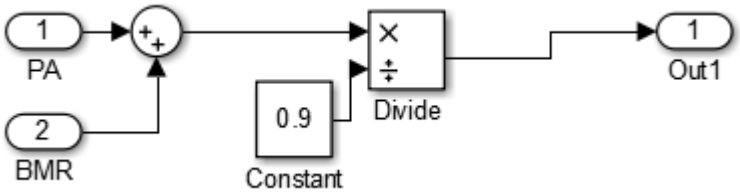


Figura 3.15: Subsistema em Simulink de EI_0

$$g(t) = \frac{EI(t)}{EI_0(t)}. \tag{3.24}$$

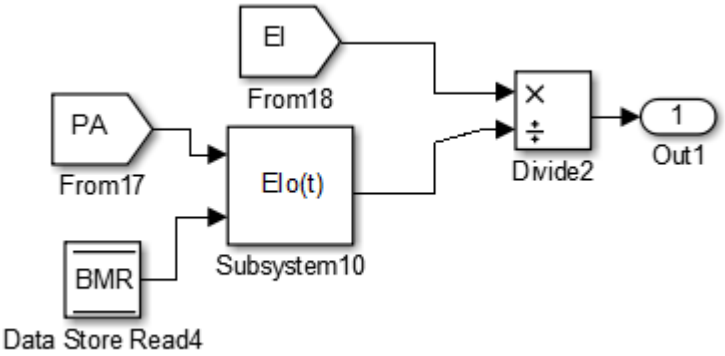


Figura 3.16: Subsistema em Simulink de g(t)

Nota de Desenvolvimento: Na fórmula (3.23) é utilizado o valor aproximado de TEF, mas para resultados mais precisos deve-se utilizar TEF em relação à alimentação dado por (3.3). No entanto, para o valor inicial de EI_0 é necessário obrigatoriamente uma estimativa de TEF (visto que depende de EI e este no momento inicial é igual a EI_0).

A fórmula (3.23) foi ajustada empiricamente para o calculo do instante inicial de EI_0:

$$EI_0(1) = \frac{PA(1) + BMR(1)}{0.9677}. \tag{3.25}$$

Nas fórmulas de FM(t) e FFM(t) em (3.17) e (3.18) substituiu-se p(t) pela relação fe(t)=1-p(t) para evitar redundâncias desnecessárias.

No instante t=1 não existem dados simulados para t-1, logo é necessário adaptar uma aproximação para o seu estado inicial. Após a comparação de fórmulas com resultados tabelados e baseado em discussões com os colaboradores da FCNAUP, optou-se por utilizar as fórmulas do artigo [14], equação (2.14) para Homem e equação (2.15) para Mulher de modo a calcular BMR dado por (3.4).

3.4.3 Variação de Peso

Neste Modelo de Balanço Energético adaptado o peso actual diário (BM ou BW) é calculado através da adição dos dois tipos de massa mencionados, FM(t) e FFM(t):

$$BW(t) = FFM(t) + FM(t). \tag{3.26}$$

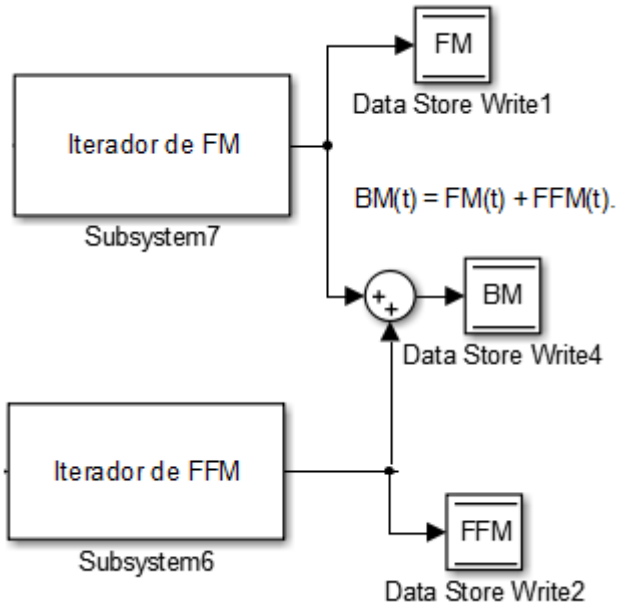


Figura 3.17: Subsistema em Simulink de BM

3.5 Conclusão

Neste capítulo foi demonstrado passo a passo o sistema dinâmico não linear que é o peso do corpo humano. Após a comparação dos resultados obtidos em Simulink com as tabelas do anexo B, verificou-se que a implementação dos blocos foi bem sucedida. Quanto aos valores obtidos nesta fase do projecto, estes só podem ser validados dada uma Ingestão Alimentar adequada (nesta fase, de modo a facilitar a simulação, seguiu-se o cenário não real de Ingestão Alimentar constante). Ou seja, ainda não se consegue prever qual o peso que será atingido dado um intervalo de tempo para a dieta e um balanço energético constante. A resolução deste problema é abordada no capítulo seguinte.

Capítulo 4

Dimensionamento da Ingestão Alimentar

Comparando perspectivas, do ponto de vista nutricional, o eficiente dimensionamento da Ingestão Alimentar é tão ou mais importante do que o eficaz dimensionamento da Ingestão Alimentar, sendo este último mais premente do ponto de vista matemático. Este capítulo propõe uma formulação exemplo desse dimensionamento que tenta o entrosamento dos dois pontos de vista, ou seja, perder peso de uma maneira saudável (eficiência nutricional) atingindo os objectivos previstos (rigor matemático).

4.1 Variação de Peso com EI Constante

Como se pode verificar na figura 3.1, no sentido de validar a construção da fase 2 (perda de peso) usou-se um EI constante. A cada EI corresponde um peso estável, isto é, cada valor que EI pode assumir equivale ao EIo (Ingestão Alimentar de Manutenção) de uma certa massa.

Neste contexto, é de esperar que o horizonte temporal (t_{final}) e peso final ($BW(t_{final})$) da figura 4.1 estejam com valores desadequados, visto que EI não foi dimensionado para nenhum plano dietético especifico.

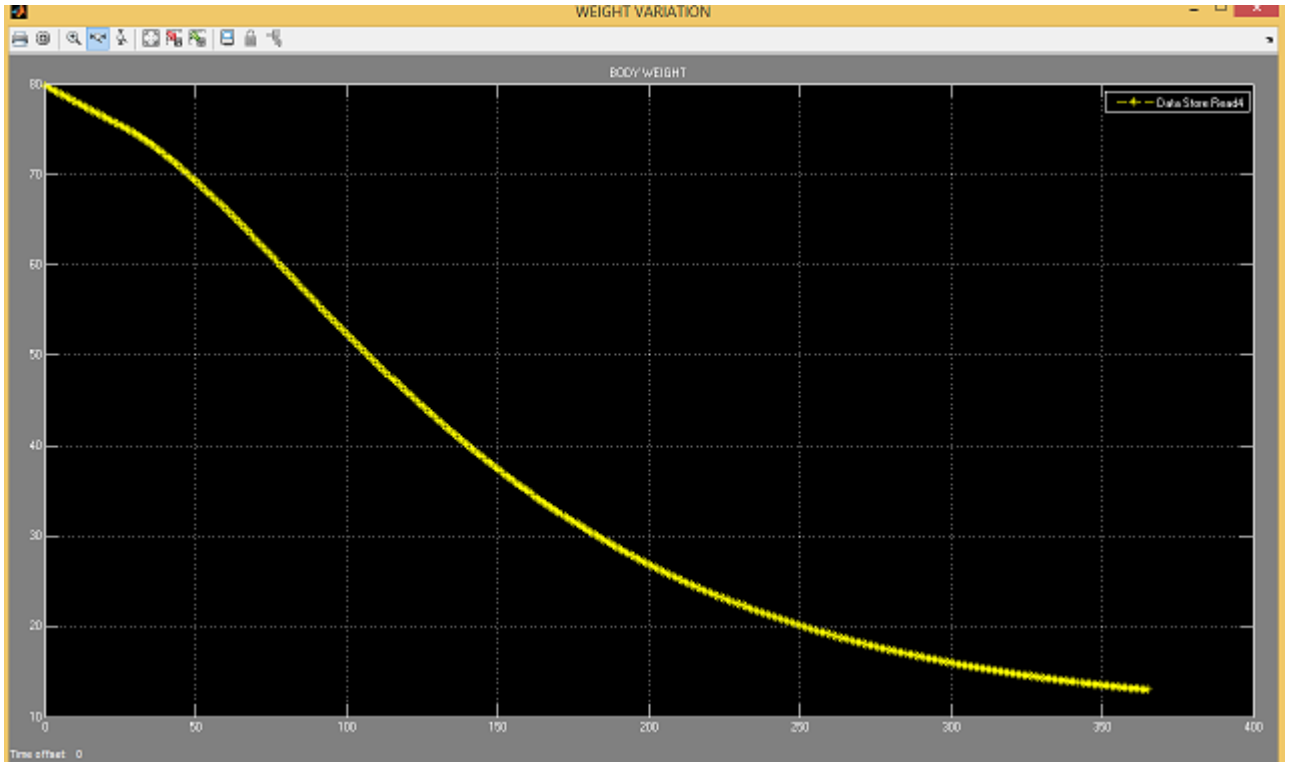


Figura 4.1: Gráfico de BW diário

Este gráfico representa um output do modelo Simulink para 365 dias com Ingestão Alimentar e Actividade Física constante, cujo Balanço Energético correspondente é negativo. Como explicado na Conclusão da secção 3.5, o gráfico na prática representa apenas o comportamento do corpo humano face a esse balanço negativo mas os seus valores finais não têm um significado realista. Este facto deve-se à ausência de um plano de Ingestão Alimentar dimensionado para um peso objectivo realista (problema abordado na secção seguinte). Os pesos resultantes do modelo em Simulink foram comparados dia a dia (através da figura em cima e blocos "*Display*") com um ficheiro excel que continha os dados dos primeiros 5 dias da fase 2 (consultar anexo B) permitindo validar a utilização da ferramenta e ajustar os parâmetros da modelação.

4.2 Variação de Peso com EI Regulado

De forma a dar utilidade ao modelo, é necessário limitar o tempo de dieta e ter conhecimento do peso inicial do utilizador e do peso que pretende alcançar.

Foi arquitectado um script que inclui a recolha dessa informação:

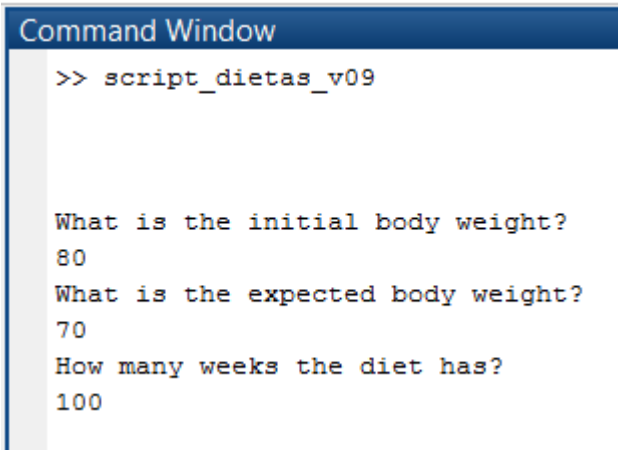


Figura 4.2: Perguntas na Janela de Comandos do Matlab

Perante essa recolha, o algoritmo sugerido nesta secção pretende escalonar valores de EI de modo a que o peso objectivo seja atingido no tempo pretendido. De acordo com um estudo direccionado para o Departamento de Defesa Nacional da Austrália [36] , uma perda gradual de 5% do Peso Total em três semanas não provoca perdas de desempenho físico nos militares. Essa percentagem para o intervalo de tempo mencionado foi escolhida como referência para toda a população, dada a falta de qualquer outra informação ou estudo mais adequados.

Posto isso, o algoritmo segue os seguintes passos:

- Percentagem de Peso a Perder:

$$BW\,goal\,percentage = (1 - \frac{BW\,goal}{BW(1)}) \times 100, \tag{4.1}$$

  onde:

  * BWgoal - Peso objectivo;

  * BW(1) - Peso Inicial.

- Percentagem de Peso Médio a perder por semana:

$$averageWeekLoss = \frac{BW\,goal\,percentage}{weeks}, \tag{4.2}$$

onde:

* weeks - semanas.

- Valor Referência de Percentagem de Perda de Peso (5 % em três semanas) ajustado para uma semana:

$$standardWeekpercent = 5/3. \tag{4.3}$$

- Verificações de Saúde e Segurança:

$$BWgoalpercentage >= 50. \tag{4.4}$$

```
What is the initial body weight?
80
What is the expected body weight?
35
How many weeks the diet has?
40
*************
 Advise:
The goal weight is below 50% of your current weight
```

Figura 4.3: Verificação de Saúde - Perdas superiores a 50%

$$averageWeekLoss > standardWeekpercent. \tag{4.5}$$

```
What is the initial body weight?
80
What is the expected body weight?
60
How many weeks the diet has?
5
*************
 Advise:
 The weight to lose in the given time span is not wealthy.
 Please increase the weeks of diet
```

Figura 4.4: Verificação de Saúde - Perda Demasiado Acentuada

4.2.1 Déficit Energético

O Déficit Energético pode ser visto como a aceleração (neste caso, negativa) que se aplica à Ingestão Alimentar de modo a regular a velocidade da perda de peso. Para definir essa aceleração, é necessário ter em conta a percentagem de peso total a perder e as semanas para o fazer.

Voltando à perspectiva nutricional (que inclui psicologia comportamental), dada a informação partilhada pelos colaboradores de nutrição, quando se inicia uma dieta é a altura de maior motivação. Considerando esse facto, a percentagem de Peso a perder nas primeiras semanas de dieta deve ser maior que a Percentagem de Peso Médio a perder por semana (4.2). Seguindo a mesma lógica de compensação, a percentagem de Peso a perder nas últimas semanas de dieta deve ser menor que a Percentagem de Peso Médio a perder por semana.

A primeira parte do algoritmo identifica em que intervalo se situa o número de semanas de dieta, atribuindo coeficientes diferentes consoante o intervalo. Para uma dieta de, por exemplo, 76 semanas, o algoritmo divide 76 em 3 partes, sendo a primeira parte igual à última e o excesso da divisão recai para a parte intermédia. Neste caso, o número de primeiras e últimas semanas é 25 e o número de semanas intermédias é 26.

- Percentagem de Peso a Perder por Semana enquanto j <= 25 :

$$weeklyBWlosspercentage(j) = averageWeekloss \times coeficient, \tag{4.6}$$

  onde:

  * j - Iterador de Semanas,

  * coeficient - coeficiente motivacional (regulável de 1.1 a 1.9).

- Percentagem de Peso a Perder por Semana enquanto 25 < j <= 25 + 26 :

$$weeklyBWlosspercentage(j) = averageWeekloss, \tag{4.7}$$

- Percentagem de Peso a Perder por Semana enquanto 25 + 26 < j <= 76 :

$$weeklyBWlosspercentage(j) = averageWeekloss \times (2 - coeficient). \tag{4.8}$$

Visto que o Valor Referência de Percentagem de Perda de Peso (4.3) corresponde a um déficit de EI máximo (maxEIdeficit - valor ajustado dependendo dos dados do individuo, nomeadamente o seu peso actual e o seu peso objectivo), usando a regra de três simples pode-se calcular o déficit energético correspondente à Percentagem de Peso a perder em cada semana.

$$EIdeficit(t) = \frac{weeklyBWlosspercentage(j) \times maxEIdeficit}{standardWeekpercent}. \tag{4.9}$$

Nota de Desenvolvimento: Verificou-se que o modelo descrito no capítulo 3 manifesta comportamentos distintos para uma Percentagem de Peso a Perder (4.1) superior a 13,75 % (anexo A.3), levando a que maxEIdeficit tivesse que ser reajustado para esses casos. Os valores de maxEIdeficit foram empiricamente experimentados para o peso inicial referência 80 kg. Consequentemente, EIdeficit foi recalculado de modo a contemplar qualquer peso inicial, dando origem aos valores de EI diários recomendados:

- Para BWinit <= 80

$$EI(t) = EE(t-1) - EIdeficit(t-1) \times \left(\frac{BWinit}{80}\right)^{2.6}, \tag{4.10}$$

- Para BWinit > 80

$$EI(t) = EE(t-1) - EIdeficit(t-1) \times \left(\frac{BWinit}{80}\right)^{2}. \tag{4.11}$$

A partir dos valores de EI(t), considerando a dieta popular "*The Zone Diet*"ou Dieta 40-30-30 [37] como referência, sem qualquer critério em especial (outros rácios poderão ser mais proveitosos), pode-se calcular os respectivos valores de CI(t), PI(t) e FI(t):

$$\begin{cases} CI(t) = \frac{0.4 \times EI(t)}{a1}, \\ FI(t) = \frac{0.3 \times EI(t)}{a2}, \\ PI(t) = \frac{0.3 \times EI(t)}{a3}. \end{cases} \tag{4.12}$$

Actualizados os valores mencionados para cada instante, volta-se a aplicar a Dinâmica da Mudança de Peso relatada no capítulo 3, mais especificamente desde o cálculo de EE(t) (3.2) até BW(t) (3.26):

```
What is the initial body weight?
70
What is the expected body weight?
68
How many weeks the diet has?
8

************************
Phase 2 Results:

Final BW is = 68.13
error is = 0.19 %

************************
Phase 3 Results:

Final BW is = 68.01
error is = 0.01 %

******************
Do you wish to know your Energy Intake for phase2? Type:
[2 for YES]
[Other for NO]
2

******************
EI of week 1 is 3340 kcal |EI of week 2 is 3065 kcal |
EI of week 3 is 3058 kcal |EI of week 4 is 3057 kcal |EI of week 5 is 3056 kcal |
EI of week 6 is 3055 kcal |EI of week 7 is 3055 kcal |EI of week 8 is 3054 kcal |
******************
```

Figura 4.5: Exemplo de Execução do Programa - Fase 2

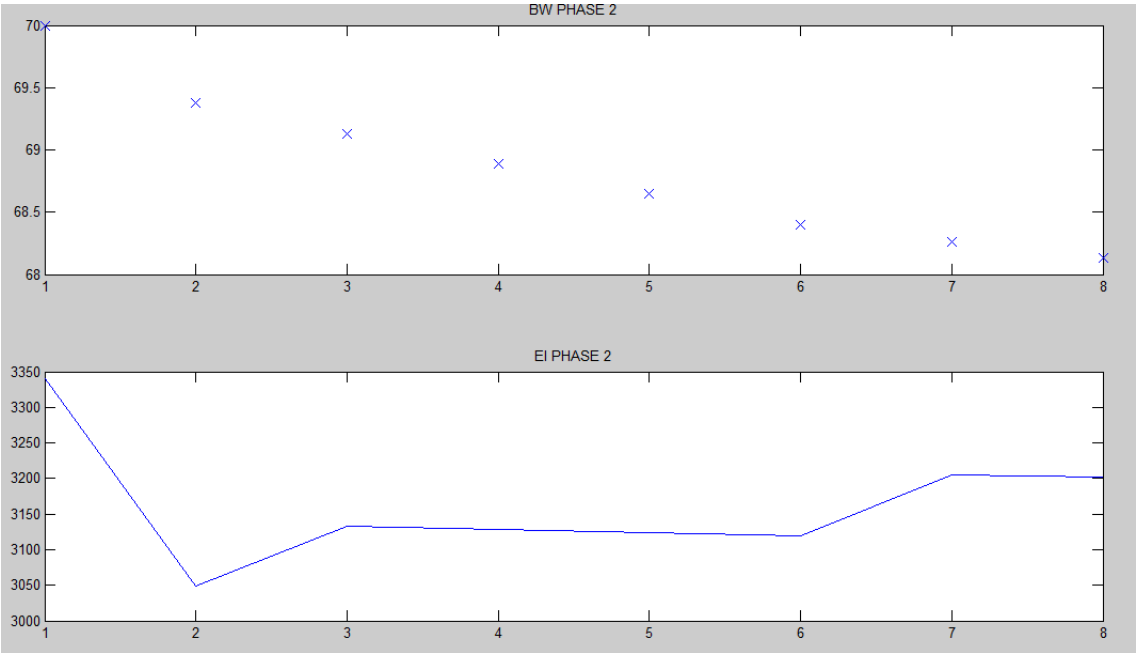


Figura 4.6: Gráfico de BW em relação a semanas (em cima) e EI em relação a semanas (em baixo) - Fase 2

Nota de Desenvolvimento: Apesar de EI ser calculado diariamente, por motivos de ocupação de Interface do Matlab, apenas é apresentado o valor de EI semanal caso o utilizador o deseje. O valor de PA é constante (pode no entanto ser alterado, inclusive por um vector de diferentes valores) e assume o valor de 1500 kcal/dia.

4.2.2 Manutenção Energética

Entrando na terceira fase da Mudança de Peso, como já foi descrito na secção 3.4.2, a Manutenção de um certo peso acontece quando EI é igual a EE desse determinado peso. Como se pode verificar na execução do Programa na figura 4.5, o peso final pode ultrapassar o peso objectivo. Quando isso acontece, significa que a posição final ou posições anteriores do vector Peso (BW) contêm o peso mais aproximado ao peso objectivo e consequentemente, ao EE correspondente.

Ou seja,

- quando BW(t_{phase2}) <= BWgoal

  EI(t_{phase3}) corresponde ao valor de EE($t_{phase2} - k$) quando BW($t_{phase2} - k$) é o valor mais aproximado de BWgoal em todo o vector BW.

  onde:

  * k - iterador de dias;

* $t_{phase3} = t_{phase2}$ + 90 dias (90 é um valor aleatório que pode tender para o infinito idealmente).

- quando BW(t_{phase2}) > BWgoal

  EI(t_{phase3}) corresponde ao valor teórico $EI_0(t_{phase2})$ (3.23).

```
What is the initial body weight?
70
What is the expected body weight?
68
How many weeks the diet has?
8

************************
Phase 2 Results:

Final BW is = 68.13
error is = 0.19 %

************************
Phase 3 Results:

Final BW is = 68.01
error is = 0.01 %

*****************
Do you wish to know your Energy Intake for phase2? Type:
[2 for YES]
[Other for NO]
2

*****************
EI of week 1 is 3340 kcal |EI of week 2 is 3065 kcal |
EI of week 3 is 3058 kcal |EI of week 4 is 3057 kcal |EI of week 5 is 3056 kcal |
EI of week 6 is 3055 kcal |EI of week 7 is 3055 kcal |EI of week 8 is 3054 kcal |
*****************
*****************
Do you wish to know your Energy Intake for phase3? Type:
[3 for YES]
[Other for NO]
3

*****************
EI to maintain BWgoal of 68 kg is 3132 kcal |
*****************
```

Figura 4.7: Exemplo de Execução do Programa - Fase 2 e 3

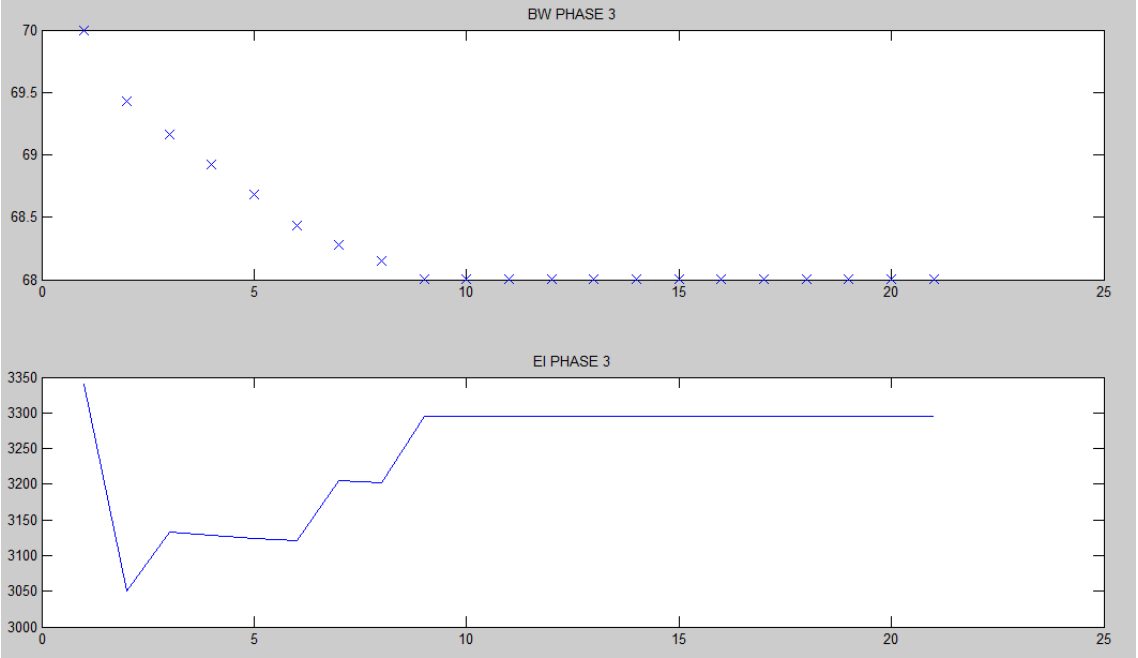


Figura 4.8: Gráfico de BW em relação a semanas (em cima) e EI em relação a semanas (em baixo) - Fase 2 e 3

Nota de Desenvolvimento: Como se pode observar na figura 4.5 e 4.7 existe um erro entre o resultado obtido e o resultado esperado. Esse erro é calculado através da fórmula:

$$error = (1 - \frac{BW\,goal}{BW(t_{phase2})}) \times 100. \tag{4.13}$$

4.3 Conclusão

Após o dimensionamento da Ingestão Alimentar regulada manualmente conseguiu-se fazer uma aproximação ao peso objectivo no tempo de dieta pretendido. Para isso, foi adquirido conhecimento empírico sobre quantidades de Ingestão Alimentar e seus efeitos através de várias tentativas e erros no ajuste das fórmulas criadas. Essa sensibilidade e informação valiosa permitiu o desenvolvimento do capítulo seguinte, onde se utiliza o poder computacional do IPOPTS para resolver o mesmo problema, visando um erro (4.13) nulo.

Capítulo 5

Controlo com ICLOCS

Neste capítulo são apresentadas várias abordagens ao cálculo da trajectória referência e demonstra-se a definição e resolução do problema de controlo óptimo abordado. O solver de problemas não lineares (NLP) escolhido para esse efeito foi o ICLOCS (mais informação na secção 2.3.1.2) devido ao enquadramento com o trabalho desenvolvido previamente em MATLAB. As variáveis pertinentes à modelação foram exportadas do Simulink e importadas para os ficheiros ICLOCS, que por sua vez chama o IPOPTS.

5.1 Dimensionamento de EI - Trajectória Referência - Fase 2

Nesta secção é demonstrado como se utilizou o ICLOCS para o dimensionamento da Ingestão Alimentar na segunda fase da perda de peso. O objectivo é reduzir o erro (4.13) obtido pelo algoritmo desenvolvido no capítulo 4 de modo a facilitar a futura implementação de MPC. A justificação das escolhas de modelação encontram-se a seguir à formula (5.2).

5.1.1 Definição do Problema

Função Objectivo:

$$\min \mathrm{L} = \sum_{t=1}^{t_{phase2}} 0 \times t, \tag{5.1}$$

sujeito a:

$$\begin{cases} x_1(t+1) = \frac{fe(t+1)\times EBw(t+1)}{pFM\times MJtoKcal} + x_1(t), \\ x_2(t+1) = \frac{(1-fe(t+1))\times EBw(t+1)}{pFFM\times MJtoKcal} + x_2(t), \\ x_3(t+1) = x_3(t) + f(u(t)), \end{cases}$$

$$x(0) = [FM(1), FFM(1), EI(1)],$$

$$-inf \leq u(t) \leq inf,$$

$$\begin{bmatrix} 0.5 \\ 29 \\ 1200 \end{bmatrix} \leq \begin{bmatrix} x_1(t+1) \\ x_2(t+1) \\ x_3(t+1) \end{bmatrix} \leq \begin{bmatrix} 445 \\ 200 \\ 20000 \end{bmatrix} \forall t \in [1, t_{phase3}],$$

$$x1(t_f) + x2(t_f) = BWgoal, \tag{5.2}$$

$$f(u(t)) = \begin{cases} u(t) & \text{se t = 1, t = (tphase2)/3 ou t = (2*tphase2)/3,} \\ 0 & \text{outros,} \end{cases}$$

$$f(u_0)/f(u_f) = -1.5,$$

$$-inf \leq u0 \leq 0,$$

$$0 \leq uf \leq inf,$$

$$29 \leq x_1(t) + x_2(t) \leq 635,$$

$$0 \leq PA(t) - 0.2 \times (0.024 \times x_1(t) + 0.102 \times x_2(t) + 0.85) \times MJtoKcal + ac \times \frac{0.4\times x_3(t)}{a1} + af \times \frac{0.3\times x_3(t)}{a2} + ap \times \frac{0.3\times x_3(t)}{a3} \leq 10000,$$

$$0 \leq PA(t) + (0.024 \times x_1(t) + 0.102 \times x_2(t) + 0.85) \times MJtoKcal + ac \times \frac{0.4\times x_3(t)}{a1} + af \times \frac{0.3\times x_3(t)}{a2} + ap \times \frac{0.3\times x_3(t)}{a3} \leq 10000.$$

De modo a implementar o modelo apresentado, foram concebidos três estados:

* x1 - Massa Gorda (FM);
* x2 - Massa Magra (FFM);
* x3 - Ingestão Alimentar (EI).

Explicação/Justificação das escolhas na modelação:

- Nesta fase não se está a resolver um problema de controlo óptimo, apenas se está a utilizar as vantagens computacionais do ICLOCS para obter uma trajectória referência com menos erro do que a apresentada no capítulo 4. Desse modo, a função objectivo (5.1) não se aplica e é anulada através da sintaxe aconselhada pelo *User Guide* [38] do ICLOCS, multiplicando 0 por t.

- Como se pode verificar no cálculo dos estados, a variável de controlo u(t) obtém-se através de ΔEI , ou seja, impõe o valor a aumentar ou diminuir à Ingestão Alimentar do dia seguinte.

- O que determina o número de mudanças de valor de u(t) é o número de acções de controlo. Decidiu-se dividir o plano alimentar em três partes, isto é, o problema tem três acções de controlo (consultar figura 5.2 e a Nota de Desenvolvimento explicativa em baixo dessa mesma figura para mais informações).

- Os limites escolhidos para os estados advêm dos registos extremos verificados do humanamente possível. Para o limite inferior de x_1 e x_2 utilizou-se como referência Lizzie Velasquez, a mulher com menor percentagem de massa gorda no mundo [39]. Para o limite superior de x_1 e x_2 utilizou-se como referência Jon Brower Minnoch, o homem mais pesado do mundo de acordo com o *Guinness World Records* [40]. Os limites para x_3 foram baseados em informação disponível no *website* da Harvard Medical School [41].

- Foi implementada uma *boundary condition* para $x1(t_f) + x2(t_f)$ que surgiu da necessidade de atingir o Peso Final Objectivo introduzido pelo utilizador, BWgoal.

- A segunda *boundary condition* define o rácio entre o primeiro e o último controlo, u_0/u_f. O valor foi fixado em -1.5 de modo a garantir (juntamente com as duas seguintes *boundary conditions*) que o decréscimo de EI é mais acentuado no primeiro controlo do que no último. Esta intenção torna-se relevante tendo em conta a motivação acrescida no início da dieta face ao fim da mesma.

- Ainda numa perspectiva motivadora, forçaram-se as *boundary conditions* u_0 e u_f a tomar valores negativos e positivos, respectivamente. Deste modo garantiu-se que o controlo no último instante é positivo, deixando o utilizador finalizar o último terço da sua dieta com um aumento da sua Ingestão Alimentar.

- Pretendeu-se estabelecer uma *general path constraint* para EE(t) cujo limite inferior é 1.2*BMR(t) kcal/dia e o seu limite superior é 10000 kcal/dia. De modo a implementar essa restrição, substituiu-se CI, FI e PI (4.12) na equação de TEF (3.3). De seguida, substituiu-se a equação obtida anteriormente na equação de EE (3.2).

Nota de Desenvolvimento: De forma a introduzir os dados no ficheiro do problema ICLOCS foi necessário efectuar o desdobramento da equação de EE obtida anteriormente, pois o programa

não aceita variáveis como limites, permitindo apenas valores numéricos. Esse desdobramento consistiu em passar 1.2*BMR para a parte central da inequação, ficando zero como limite inferior. Para obter resultados equivalentes à restrição pretendida, foi necessária outra inequação onde EE tem como limites 0 e 10000. Este desdobramento é representado nas duas últimas equações do conjunto (5.2) .

5.1.2 Resolução do Problema

Depois de inserida a modulação mencionada no programa ICLOCS, foi simulada a perda de peso de um indivíduo do sexo masculino com 77 kg de peso inicial, 1.80 metros de altura, 20 anos de idade e cujo objectivo é atingir 73 kg em 10 semanas:

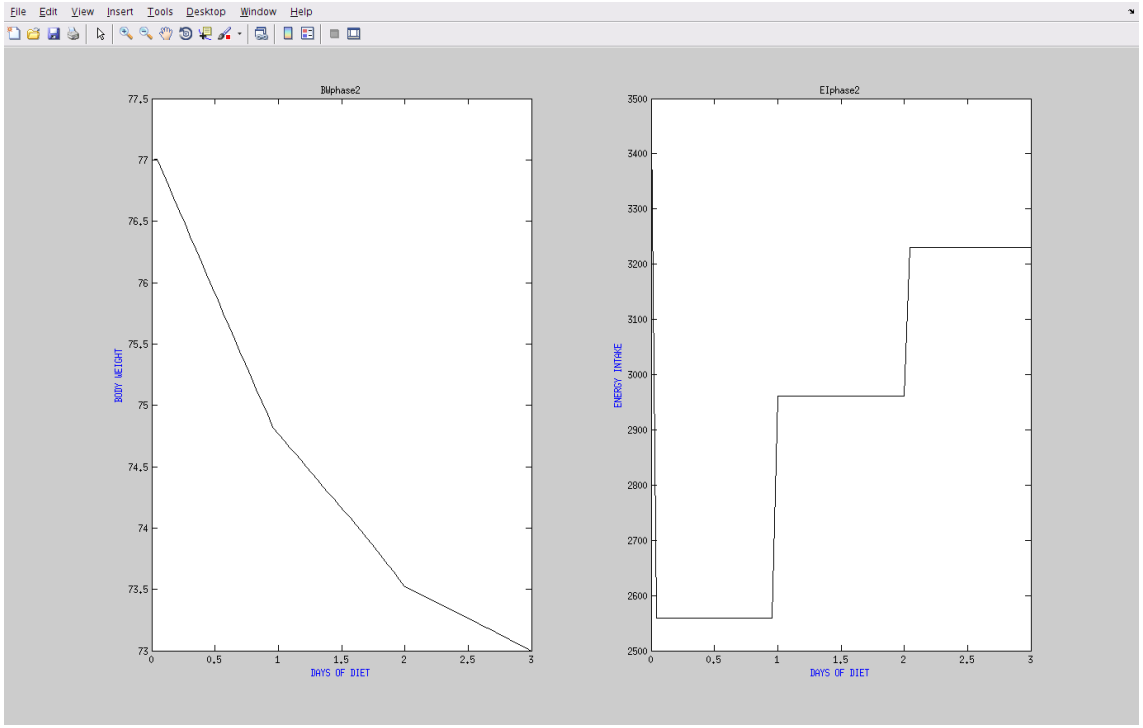


Figura 5.1: Relação entre Peso (esquerda) e EI (direita)

Nota de Desenvolvimento: Em alguns gráficos o eixo temporal representa as acções de controlo e não os dias de dieta facultados pelo utilizador de forma a facilitar a visualização dos momentos de mudança de controlo.

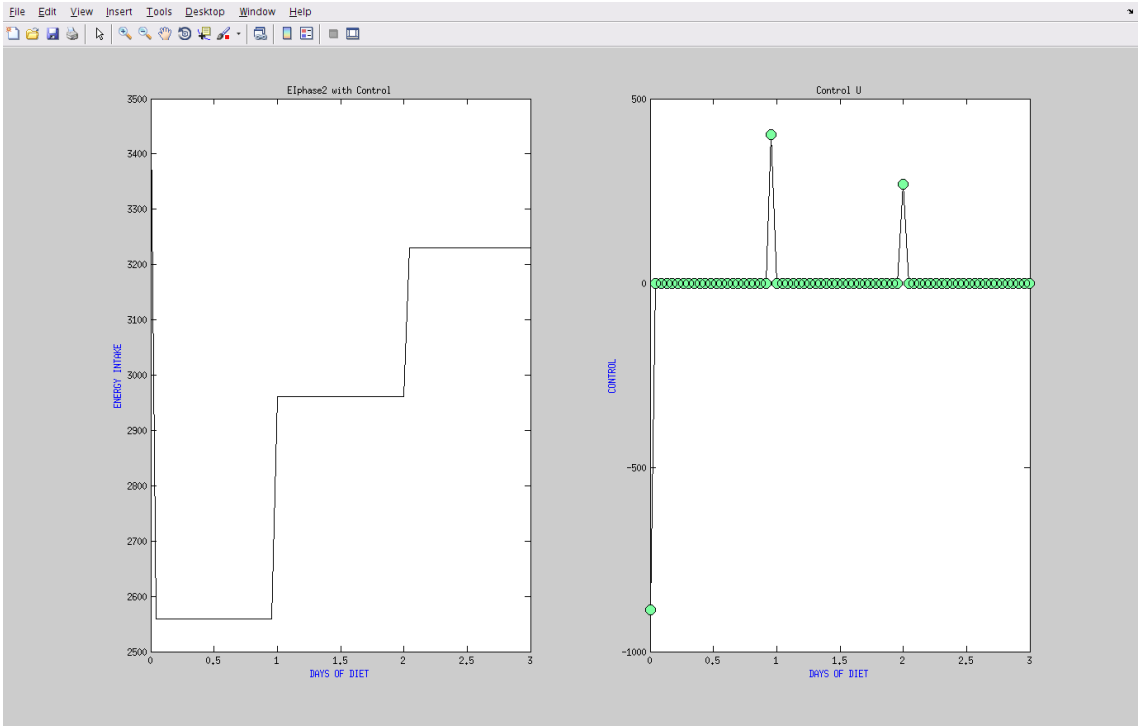


Figura 5.2: EI capítulo 4 (preto) vs Peso ICLOCS (vermelho)

Nota de Desenvolvimento: Foi introduzida uma variável Utimming (consultar código na integra no anexo C para mais informação) de modo a variar f(u(t)) apenas em três momentos: u(1), u($\frac{t_{phase2}}{3}$) e u(2 $\frac{t_{phase2}}{3}$). Em todos os outros pontos u(t) é zero como se pode verificar na figura acima. O propósito desta alteração é tornar a dieta mais realista, pois não era exequível ter um plano alimentar onde cada dia retirava-se um valor de kcal à Ingestão do dia anterior. Deste modo, o utilizador tem apenas três valores fixos de Ingestão alimentar (calculados pelo IPOPTS que verifica as restrições dadas) no seu plano dietético.

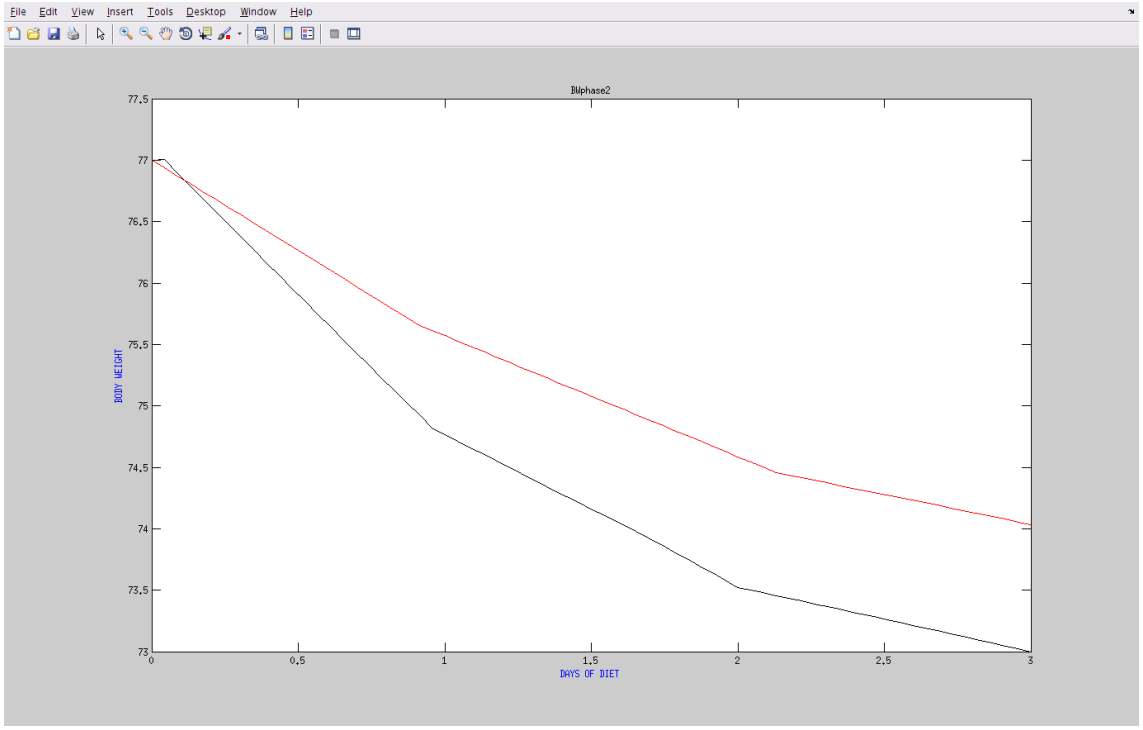


Figura 5.3: Peso capítulo 4 (preto) vs Peso ICLOCS (vermelho)

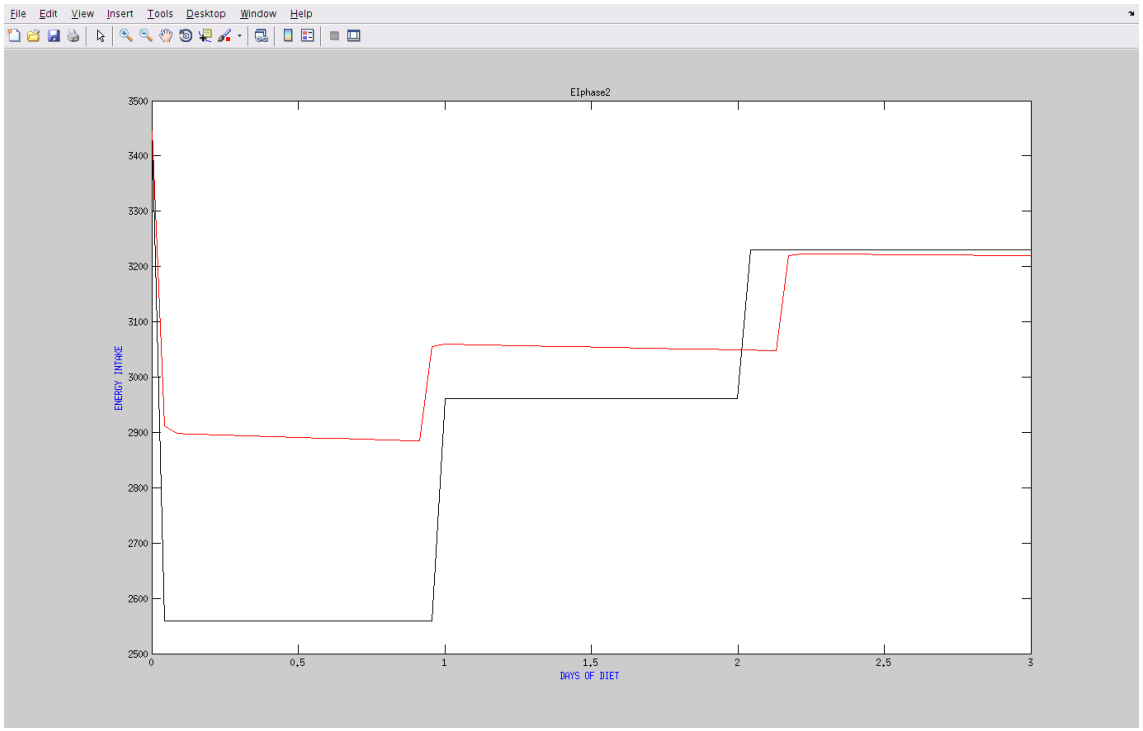


Figura 5.4: EI capítulo 4 (preto) vs Peso ICLOCS (vermelho)

Nota de Desenvolvimento: Como se pode verificar pela figura 5.3, o erro obtido nesta fase é aproximadamente 0%, representando uma melhoria significativa face aos resultados anteriores. Os valores finais dos estados (x_f) foram gravados após a conclusão do programa da fase 2 da perda de peso de modo a serem utilizados como valores iniciais (x_0) no programa da fase 3.

5.2 Dimensionamento de EI - Trajectória Referência - Fase 3

Nesta secção explicam-se as alterações ao modelo da secção anterior de modo a calcular a Ingestão Alimentar ideal para a terceira fase da perda de peso.

5.2.1 Definição do Problema

Função Objectivo:

$$\min \mathrm{L} = \sum_{t_{phase2}}^{t_{phase3}} 0 \times t, \tag{5.3}$$

sujeito a:

$$\begin{cases} x_1(t+1) = \frac{fe(t+1)\times EBw(t+1)}{pFM\times MJtoKcal} + x_1(t), \\ x_2(t+1) = \frac{(1-fe(t+1))\times EBw(t+1)}{pFFM\times MJtoKcal} + x_2(t), \\ x_3(t+1) = x_3(t) + f(u(t)), \end{cases}$$

$$x(0) = [x1phase2(t_{phase2}), x2phase2(t_{phase2}), x3phase2(t_{phase2})],$$

$$f(u(t)) = \begin{cases} u(t) & \text{se t = 1,} \\ 0 & \text{outros,} \end{cases}$$

$$-inf \leq u(t) \leq inf, \tag{5.4}$$

$$\begin{bmatrix} 0.5 \\ 29 \\ 1200 \end{bmatrix} \leq \begin{bmatrix} x_1(t+1) \\ x_2(t+1) \\ x_3(t+1) \end{bmatrix} \leq \begin{bmatrix} 445 \\ 200 \\ 20000 \end{bmatrix} \forall t \in [1, t_{phase3}],$$

$$x1(t_{phase3}) + x2(t_{phase3}) = BWgoal,$$

$$29 \leq x_1(t) + x_2(t) \leq 635,$$

$$0 \leq PA(t) - 0.2 \times (0.024 \times x_1(t) + 0.102 \times x_2(t) + 0.85) \times MJtoKcal + ac \times \frac{0.4\times x_3(t)}{a1} + af \times \frac{0.3\times x_3(t)}{a2} + ap \times \frac{0.3\times x_3(t)}{a3} \leq 10000,$$

$$0 \leq PA(t) + (0.024 \times x_1(t) + 0.102 \times x_2(t) + 0.85) \times MJtoKcal + ac \times \frac{0.4\times x_3(t)}{a1} + af \times \frac{0.3\times x_3(t)}{a2} + ap \times \frac{0.3\times x_3(t)}{a3} \leq 10000.$$

Explicação/Justificação das escolhas na modelação:

(Escolhas em comum com a secção anterior não serão comentadas. Código na integra no anexo D)

- Neste caso, o plano alimentar deve ser ajustado apenas uma vez. Uma acção de controlo no primeiro dia da fase de manutenção garante que o valor da Ingestão Alimentar é o EI_0 equivalente ao peso objectivo.

5.2.2 Resolução do Problema

Depois de inserida a modulação da secção anterior no programa ICLOCS foi simulada a manutenção de peso do mesmo indivíduo (secção 5.1.2), sendo que o horizonte temporal simulado é de 90 dias:

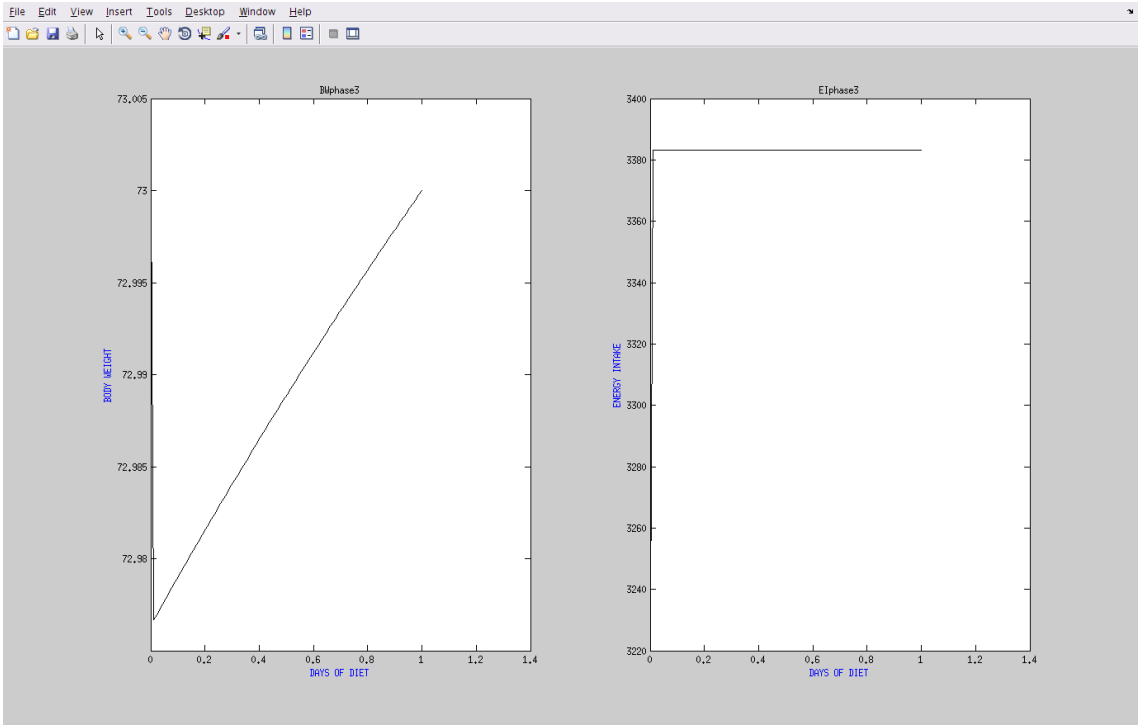


Figura 5.5: Relação entre Peso (esquerda) e EI (direita)

Nota de Desenvolvimento: Apesar do gráfico do peso ser uma recta crescente, é de mencionar que a precisão é $\pm$ 0.005, logo o erro é residual, sendo aproximadamente 0.03% neste caso em concreto.

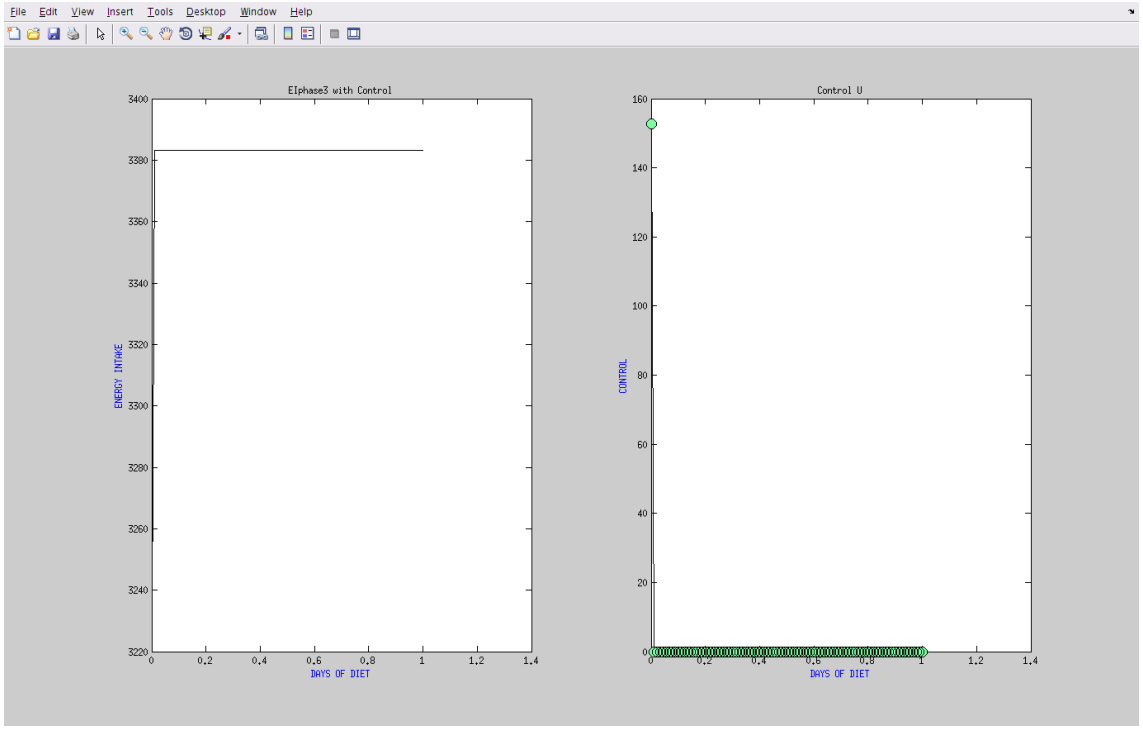


Figura 5.6: Relação entre EI (esquerda) vs Controlo(direita)

Nota de Desenvolvimento: Após a execução do ICLOCS e observando a figura acima, pode-se concluir que, dado um acréscimo de 153 kcal à alimentação no fim da segunda fase, a Ingestão Alimentar assume o valor constante de 3383 kcal, sendo esse o valor diário recomendado para manter o peso objectivo.

5.3 Aplicação de MPC - Fase 2

O Controlo Preditivo baseado em Modelo, MPC, tem demonstrado ser extremamente útil em inúmeros processos. É capaz de calcular acções de controlo que garantem o seguimento de uma dada trajectória de referência. No caso de uma dieta equilibrada, o que se pretende é que o peso de um individuo vá diminuindo segundo uma regra pré-estabelecida até atingir um determinado peso. A forma como o peso varia deverá ser acordado previamente com o nutricionista. O modelo matemático da dinâmica do peso humano é uma aproximação ainda rudimentar, logo, tem muita incerteza associada. Sabe-se também que o plano de variação de peso de uma pessoa pode sofrer ajustes. Tomando em consideração todas estas particularidades das dietas considera-se que o MPC pode ser um instrumento de grande valor na monitorização das mesmas.

Revendo as equações (2.24) e (2.25) da Breve Descrição de MPC (secção 2.4), pode-se afirmar que o utilizador começa a dieta com o peso $\bar{y}(0)$. De seguida, o programa indica-lhe qual o EI que deve ingerir em cada dia. No dia t_τ, o utilizador pesa-se e recalcula a solução de P_τ, inserindo no programa o seu peso no instante t_τ, $g(t_\tau)$. Uma nova estratégia alimentar é calculada e o utilizador

deverá segui-la. Tudo se repete novamente. De seguida pode-se observar o diagrama equivalente à aplicação de MPC para o problema em causa:

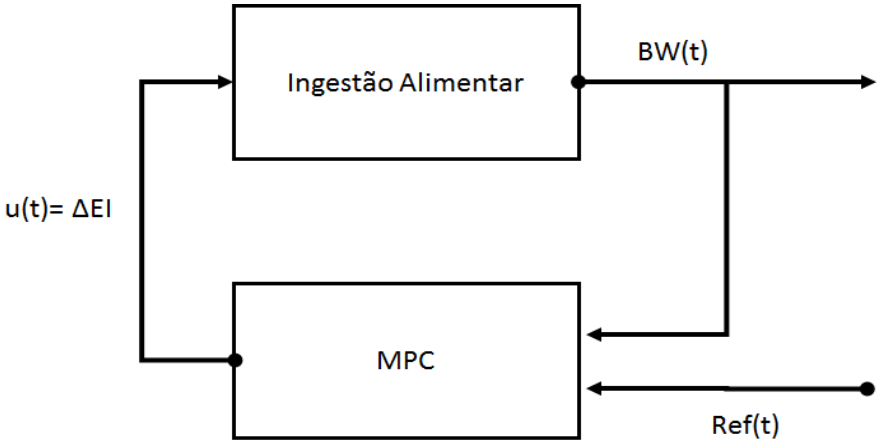


Figura 5.7: Diagrama MPC para Dietas

Partindo dos resultados da secção anterior, torna-se possível aplicar MPC a uma planta com a dinâmica perturbada face à referência. Essa planta simula a introdução de dados por parte de um utilizador da aplicação que actualiza os seus resultados dia a dia. De seguida encontra-se um exemplo de perturbação da Dinâmica:

$$\begin{aligned} x_1(t+1) &= \frac{fe(t+1)\times EBw(t+1)}{pFM\times MJtoKcal} + 0.002 \times sin(\tfrac{t}{5}) + x_1(t), \\ x_2(t+1) &= \frac{(1-fe(t+1))\times EBw(t+1)}{pFFM\times MJtoKcal} + 0.001 \times sin(\tfrac{t}{5}) + x_2(t), \\ x_3(t+1) &= x_3(t) + u(t). \end{aligned} \tag{5.5}$$

A função objectivo foi actualizada de modo a representar a necessidade de minimizar o erro entre o Peso que era suposto ter a determinado momento e o peso que realmente foi atingido:

$$\min \mathrm{L} = \sum_{t=1}^{t_{phase2}} (xref_1(t) + xref_2(t) - x_1(t) - x_2(t))^2. \tag{5.6}$$

Nota de Desenvolvimento: A planta descrita em (5.5) iria provocar uma variação diária do peso referência. Para efeitos de simulação, optou-se por aplicar uma perturbação em quatro pontos aleatórios no período de perda de peso escolhido pelo utilizador. Os dados do utilizador são os mesmos que os utilizados anteriormente (secção 5.1.2).

Explicação/Justificação das escolhas na modelação:

- Ao contrário do que foi feito aquando da obtenção da trajectória de referência, o controlo u é a variável a ser determinada e a sua estrutura não está pré-definida.

- Todas as *boundary constraints* referentes ao controlo foram retiradas de modo a que o programa decida o melhor controlo sem influências directas a agir sobre o mesmo. Assim sendo, a indicação principal que terá que seguir (além das *general path constraints*) é o *Stage Cost*.

Passos da Aplicação de MPC:

1. O programa gera uma trajectória referência para a perda de peso consoante os dados inicias do utilizador;

2. É introduzida uma perturbação num ponto dessa trajectória (por exemplo, através das equações em (5.5));

3. O algoritmo procura a solução óptima para que, a partir desse ponto, possa voltar a aproximar-se da trajectória, mantendo em vigor os objectivos iniciais.

4. Faz-se a actualização da trajectória referência. Inicialmente é dada uma trajectória referência calculada na secção 5.1. Quando o utilizador insere a primeira leitura que difere dessa referência, uma nova referência é gerada (com base na antiga) desse ponto para a frente de modo a minimizar o *stageCost* em (5.6).

5. Quando/Se existir uma nova perturbação, isto é, é inserida outra leitura que difere da nova referência, recomeça o ciclo desde o passo 3.

Nesta fase é explicado como é que o programa lida com apenas uma perturbação de 1.0025 × BW no instante t=23 dias (perturbação na ordem das centenas de gramas). O algoritmo comporta-se da seguinte forma:

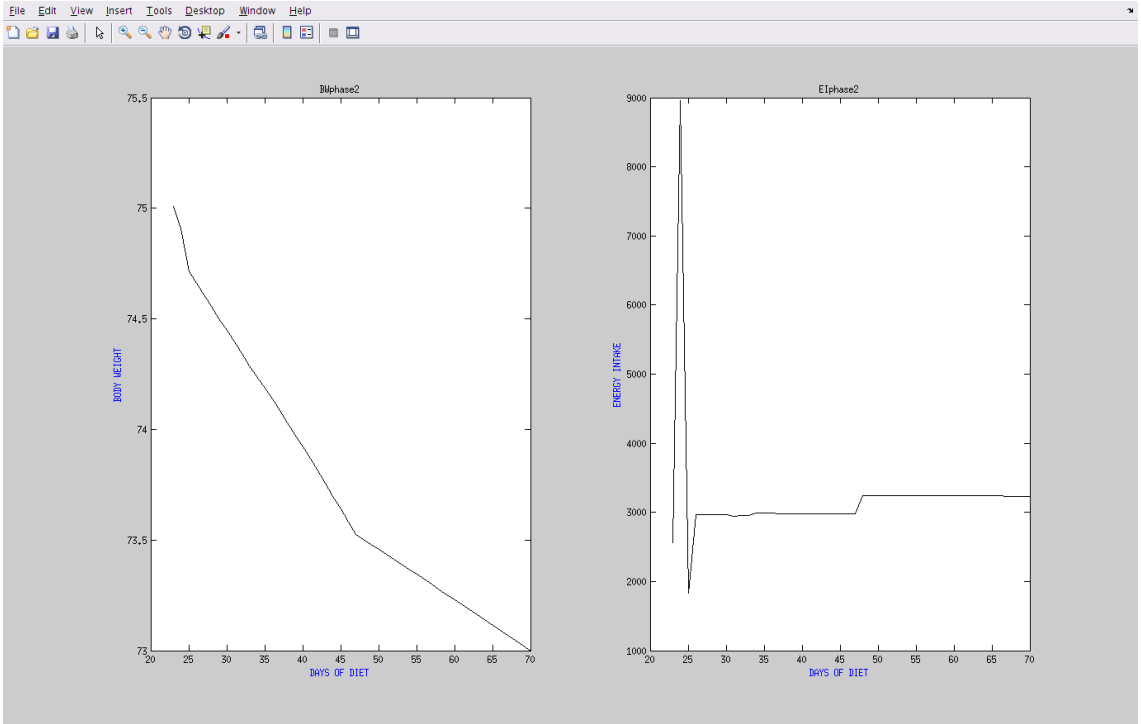


Figura 5.8: Relação entre BW (esquerda) vs EI (direita)

Após o ajuste inicial na Ingestão Calórica, o algoritmo ICLOCS consegue aproximar os valores calculados dia a dia à referência fornecida. Na prática, isto significa que o utilizador seguiu o plano alimentar inicial e mesmo assim não atingiu o peso expectável para o dia 23. Para o manter na linha de perda de peso prevista, é dado um novo plano alimentar.

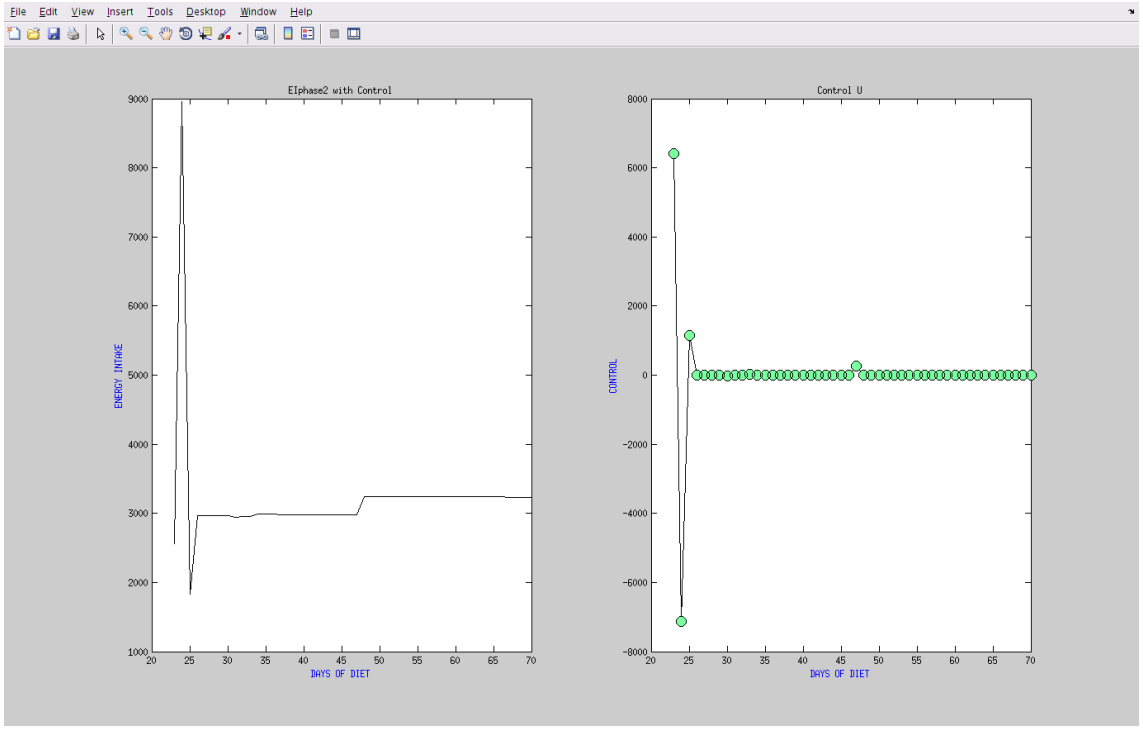


Figura 5.9: Relação entre EI (esquerda) vs Controlo (direita)

O pico inicial de EI observável nas figuras 5.8 e 5.9 corresponde ao valor que levou à introdução dessa perturbação, isto é, a um acréscimo de peso corresponde um acréscimo de ingestão alimentar. Na figura seguinte pode-se comparar o controlo original com o controlo ajustado de forma a atingir as metas iniciais:

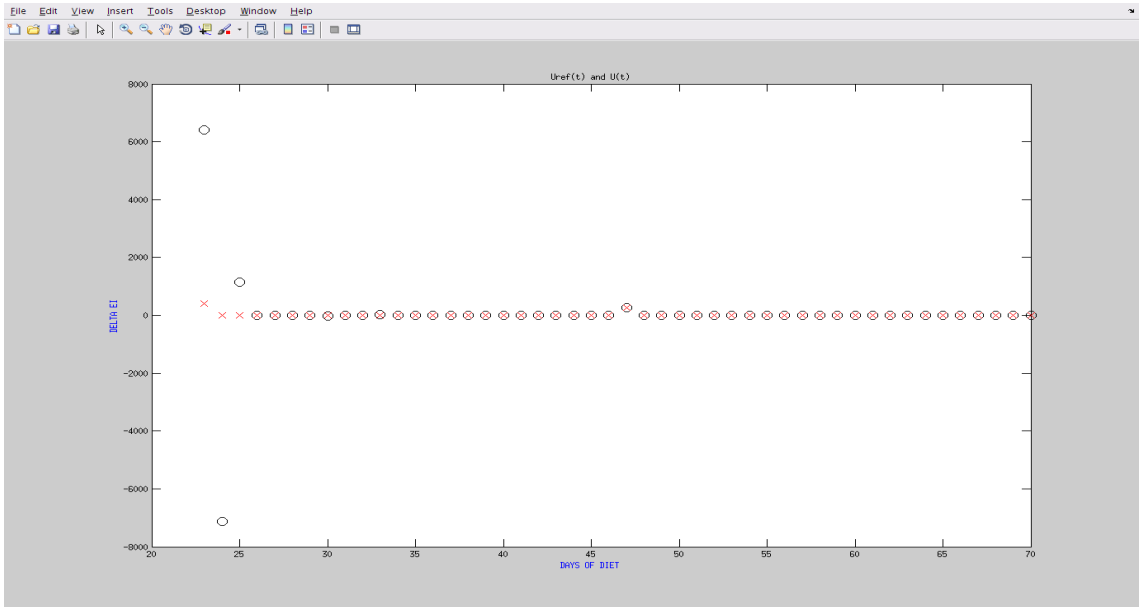


Figura 5.10: Relação entre Controlo referência (cruzes) vs Controlo (círculos)

Utilizando a função objectivo proposta (5.6), o programa ICLOCS tende a minimizar o *stage cost* o mais rápido possível. Demonstra-se a consequente evolução do peso na seguinte figura:

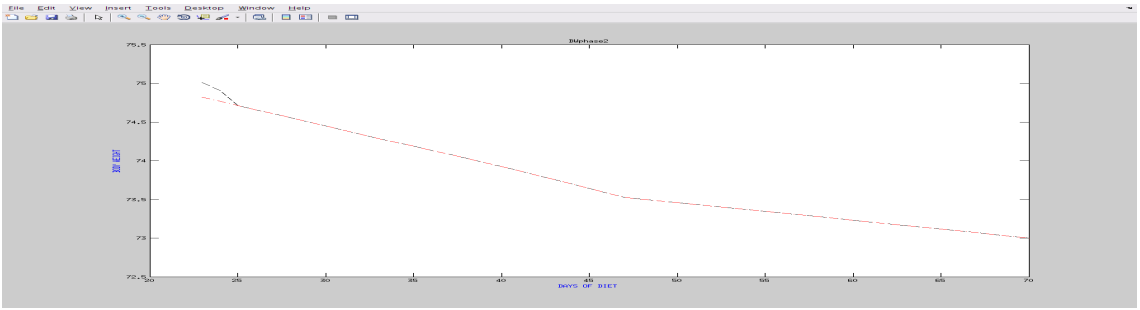
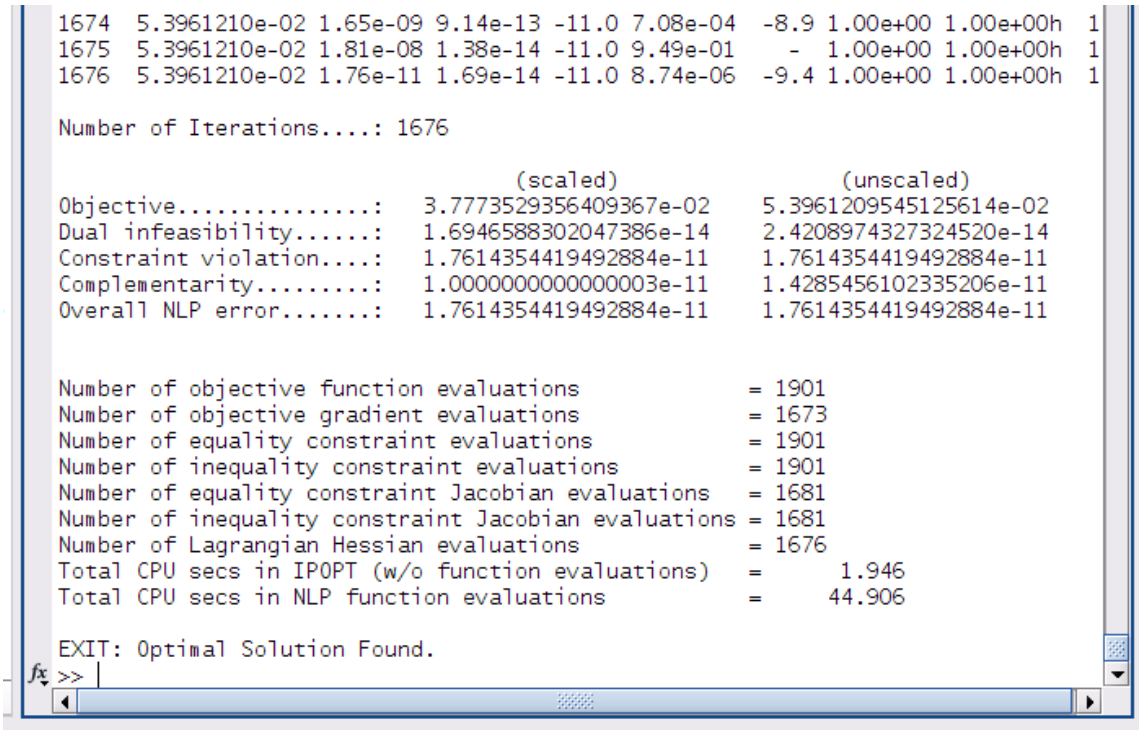


Figura 5.11: Relação entre Peso referência (vermelho) vs Peso Real (preto)

Estes resultados foram obtidos após 1676 iterações para as condições exempladas. Os erros associados a essa execução podem ser observados na figura em baixo:



```
1674  5.3961210e-02 1.65e-09 9.14e-13 -11.0 7.08e-04  -8.9 1.00e+00 1.00e+00h  1
1675  5.3961210e-02 1.81e-08 1.38e-14 -11.0 9.49e-01    -  1.00e+00 1.00e+00h  1
1676  5.3961210e-02 1.76e-11 1.69e-14 -11.0 8.74e-06  -9.4 1.00e+00 1.00e+00h  1

Number of Iterations....: 1676

                                   (scaled)                 (unscaled)
Objective...............:   3.7773529356409367e-02    5.3961209545125614e-02
Dual infeasibility......:   1.6946588302047386e-14    2.4208974327324520e-14
Constraint violation....:   1.7614354419492884e-11    1.7614354419492884e-11
Complementarity.........:   1.0000000000000003e-11    1.4285456102335206e-11
Overall NLP error.......:   1.7614354419492884e-11    1.7614354419492884e-11


Number of objective function evaluations             = 1901
Number of objective gradient evaluations             = 1673
Number of equality constraint evaluations            = 1901
Number of inequality constraint evaluations          = 1901
Number of equality constraint Jacobian evaluations   = 1681
Number of inequality constraint Jacobian evaluations = 1681
Number of Lagrangian Hessian evaluations             = 1676
Total CPU secs in IPOPT (w/o function evaluations)   =      1.946
Total CPU secs in NLP function evaluations           =     44.906

EXIT: Optimal Solution Found.
>>
```

Figura 5.12: Interface Matlab após execução de ICLOCS

Nota de desenvolvimento: De modo a aplicar MPC em mais do que um ponto e em não tantos pontos como na planta (5.5), foi elaborado um programa que replica as condições iniciais para o ponto que se pretende perturbar, ou seja, $x1(1) = x1_{ref}(ponto) \times 1.0025$ e $x2(1) = x2_{ref}(ponto) \times 1.0025$. Quando recalculadas as trajetórias após a primeira perturbação, essas são

gravadas, lidas como entradas e passadas como nova trajectória referência até a perturbação seguinte.

Introduzidas perturbações $BW_{ref}(ponto) \times 1.0025$ nos dias aleatórios 23 (P1), 24 (P2), 59 (P4) e $BW_{ref}(ponto) \times 0.9975$ no dia 43 (P3), o programa comporta-se da seguinte maneira:

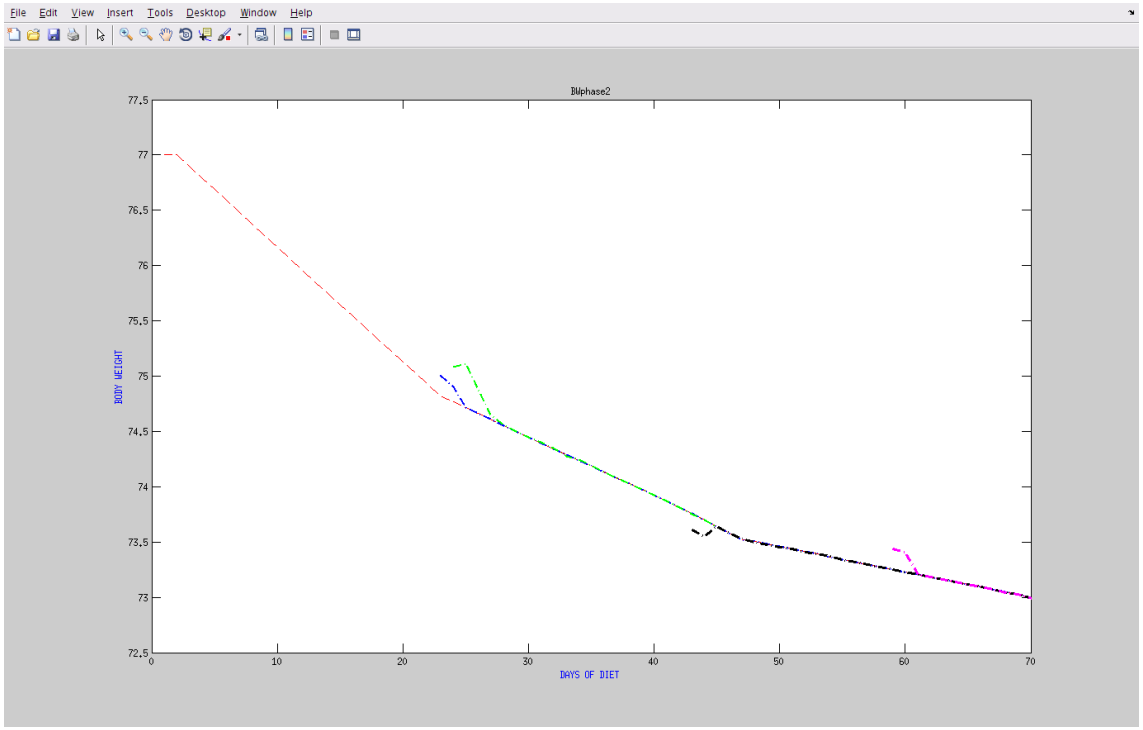


Figura 5.13: Relação entre BWref (vermelho), BW após P1 (azul), BW após P2 (verde), BW após P3 (preto) e BW após P4 (magenta)

As sucessivas correcções ao peso são efectuadas através dos controlos em 5.14 que originam os novos planos de ingestão alimentar em 5.15:

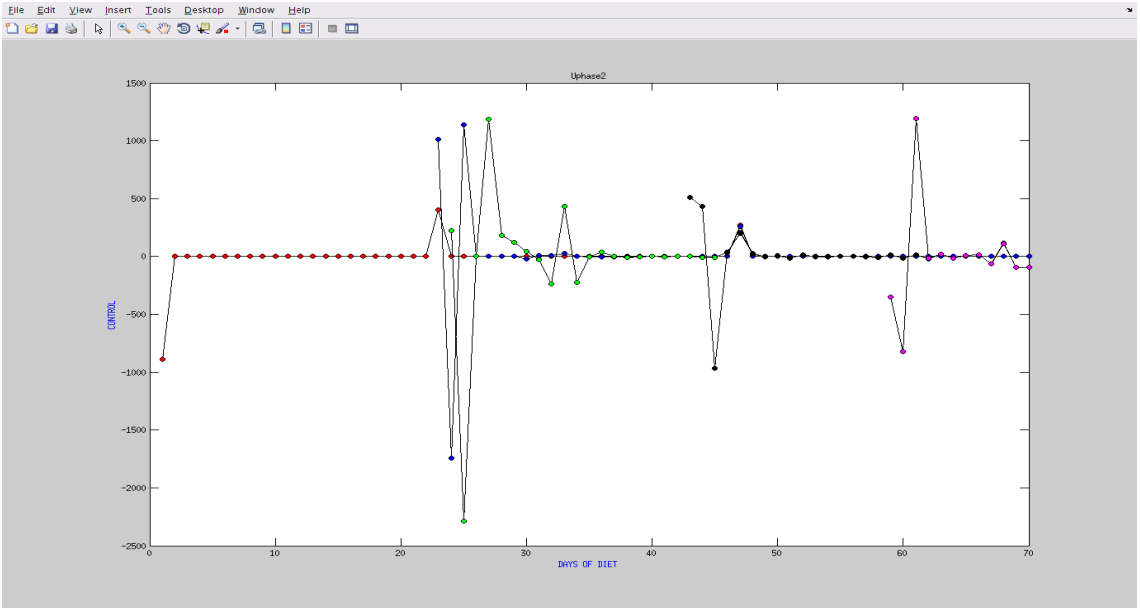


Figura 5.14: Relação entre Uref (vermelho), U após P1 (azul), U após P2 (verde), U após P3 (preto) e U após P4 (magenta)

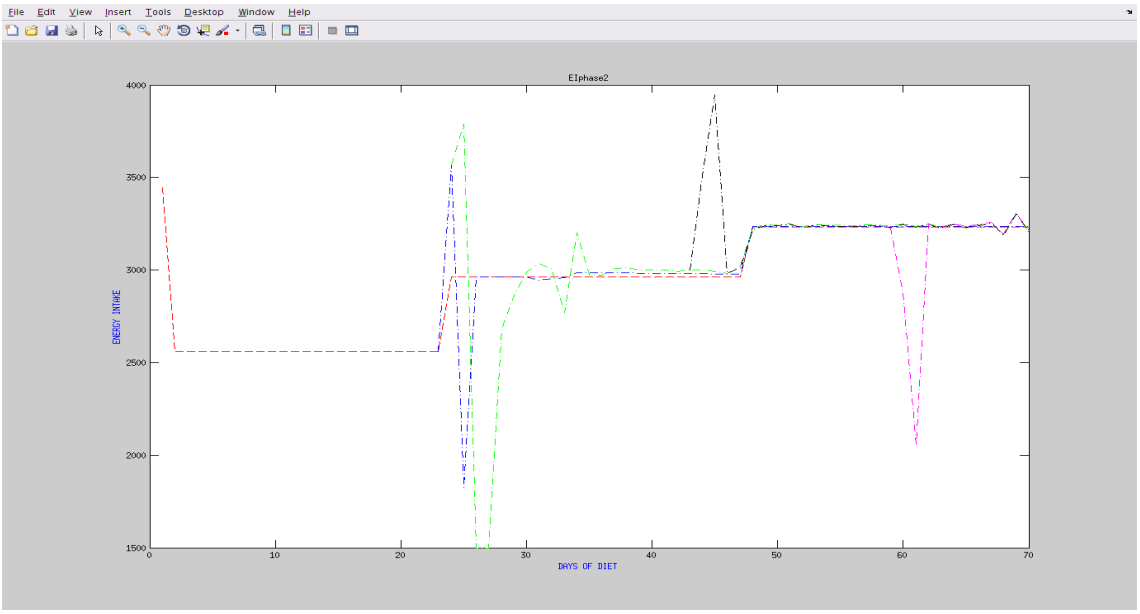


Figura 5.15: Relação entre EIref (vermelho), EI após P1 (azul), EI após P2 (verde), EI após P3 (preto) e EI após P4 (magenta)

Nota de Desenvolvimento: Os algoritmos desenvolvidos para esta secção são actualmente específicos a cada dia, no entanto, a sua lógica é semelhante para todos os dias. Pode ser consultado no anexo E um exemplo do algoritmo para um dia de perturbação.

5.4 Conclusão

O poder computacional de IPOPTS foi utilizado com sucesso na resolução das equações da dinâmica face às restrições impostas (5.2), no sentido em que nulificou o erro (4.13) do dimensionamento da Ingestão Alimentar do capítulo 4. Ou seja, a trajectória referência atinge o peso objectivo no tempo de dieta previsto. Utilizou-se ICLOCS (que por sua vez recorre a IPOPTS) para aplicar Model Predictive Control ao problema de controlo óptimo. Face a uma perturbação, o programa minimiza a diferença entre a trajectória referência anterior e a nova trajectória gerada. A trajectória referência é então actualizada para os valores gerados, sendo de novo utilizada dada uma nova perturbação. O programa reajusta a Ingestão Alimentar (plano dietético) sem modificar o horizonte temporal de modo a que o peso final seja o inicialmente ambicionado.

Capítulo 6

Conclusões e Trabalho Futuro

6.1 Satisfação dos Objectivos / Dificuldades Encontradas

Nesta secção são explorados os objectivos, o seu grau de conclusão, as dificuldades que surgiram durante o trabalho e a qualidade dos resultados obtidos.

Partindo dos objectivos iniciais da Dissertação:

1. Estudo do Estado da Arte:

   * A Revisão Bibliográfica inclui alguns conceitos teóricos fulcrais ao desenvolvimento do trabalho e à compreensão das abordagens adoptadas no documento e no projecto.

   Após o inteiramento desses conceitos, foi feita uma pequena introdução à ferramenta mais utilizada neste projecto, o MATLAB. Apesar de ser um software largamente utilizado, os seus *add ons* e *toolboxes* têm um nível de especificidade que requereu um enquadramento teórico das suas aplicações e os seus princípios de utilização.

   No Estado de Arte procurou-se que as pesquisas, apesar de terem um foco preciso na temática, fossem altamente abrangentes tanto na perspectiva do utilizador comum e do utilizador técnico como na perspectiva de aplicações completas já comercializadas (as suas características, funcionalidades e, se possível, a sua modelação).

2. Aquisição do conhecimento para atingir os objectivos propostos:

   a. Análise e Implementação de algoritmos desenvolvidos na literatura:

      * O Simulink foi crucial nesta fase do projeto, pois permitiu uma modelação gráfica que minimizou a resistência inicial do desenvolvimento, resistência essa essencialmente presente devido ao cruzamento e intra dependência de variáveis. O facto de algumas variáveis se apresentarem com unidades diferentes (no mesmo artigo existiam variáveis em kcal e outras em MJ sem estar referenciada essa mudança) dificultou a celeridade na obtenção de resultados coerentes, pois julgava-se ser um erro de programação.

b. Implementação numérica de Esquemas de Modelos Predictivos:

* O Problema de dimensionamento de EI foi resolvido através dos algoritmos apresentados no capítulo 4. A metodologia utilizada para essa resolução representa apenas uma possibilidade de solução (baseada em premissas adquiridas pelo estudo da temática). No entanto, comparando os erros (4.13) do simulador sugerido em [26] e os erros obtidos, denota-se uma melhoria significativa nos resultados.

* O conhecimento adquirido nesse dimensionamento facilitou a modelação necessário ao dimensionamento de EI em ICLOCS. Através de testes exaustivos foi possível verificar que restrições eram pertinentes e que resultados eram expectáveis.

* Foram aplicados Controlos Predictivos baseados no Modelo (MPC) introduzindo perturbações atrás de perturbações na dinâmica desse modelo referência gerado pelo ICLOCS e nas seguintes trajectórias obtidas, originando assim uma planta de valores que simularam as entradas de dados pelo utilizador. Nas simulações efectuadas perturbaram-se quatro momentos distintos e os resultados estão em conformidade com os objectivos, ou seja, novos planos alimentares foram fornecidos a cada inserção errónea (comparada com a referência de EI nesse instante). Esses planos alimentares aproximam rapidamente a nova trajectória do Peso à sua referência nesse momento, levando a que o utilizador cumpra as suas metas iniciais (tempo final e peso final).

3. Comparação de várias abordagens de maneira a desenvolver e validar a solução desejada:

* A Revisão Bibliográfica foi uma fase muito extensa pois existiu uma preocupação em perseguir cada referência de todos os artigos estudados, entre outros. Essa análise crítica conduziu a algumas adaptações de ambos os modelos, obtendo assim um modelo "hibrido"melhorado e justificado. A falta de informação quanto ao dimensionamento de EI (capítulo 4) levou à arquitectação de um algoritmo 100% original de modo a permitir a validação do modelo final.

4. Desenvolvimento e Teste da Solução Final:

a. Implementação da metodologia desenvolvida:

* O Modelo Dinâmico da Mudança de Peso (capítulo 3) foi implementado com sucesso. Quando conjugado com o dimensionamento da Ingestão Alimentar (capítulo 4), obtiveram-se resultados credíveis e realistas, com um erro médio (4.13) de cálculo associado ao dimensionamento de EI inferior a 0.8% para a fase 2 e 0.2% para a fase 3 (dada uma amostra de 50 testes de simulação aleatórios para ambas as fases).

* A utilização estratégica do *template* ICLOCS facilitou a transição de dados entre o ficheiro Simulink, os *scripts* da trajéctoria real e esse mesmo MPC usado.

b. Validação da ferramenta com dados da FCNAUP:

* A FCNAUP não possui uma base de dados relativa a grupos sujeitos a perda de peso, impossibilitando uma verificação de resultados imediata. No entanto, como mencionado no ponto f4 da Secção 6.2 deste capítulo, um dos futuros trabalhos associados

a este projeto será modificar os rácios da equação de EI (4.12) de acordo com valores aprovados e validados por nutricionistas. Tornando a trajectória referência mais adequada, também os valores após o Controlo serão mais realistas.

6.2 Trabalho Futuro

O projeto que envolve esta dissertação tem objectivos próprios bem definidos. Dentro desses objectivos e também fora deles (perspectiva mais geral, puramente de desenvolvimento de potencialidades da aplicação) foram apurados alguns pontos que acrescentam valor ao resultado final:

f1 - Actualizar Simulink:

* Integrar a leitura do vector EI em vez do conjunto de blocos actual (figura 3.1) de modo a que receba valores Dimensionados de EI de acordo com uma dieta especifica (actualmente, *The Zone Diet*) provenientes do *script* ICLOCS produzido.

* Introduzir o *Model Predictive Controller Block* [42] ao modelo actual após a integração do dimensionamento de EI acima mencionado, obtendo um ficheiro Simulink independente e totalmente funcional.

f2 - Programa de Cálculo de Actividade Física:

* De forma a rentabilizar uma futura aplicação, é necessário conjugar com os algoritmos desenvolvidos um programa de Cálculo de Actividade Física o mais preciso possível. Pode ser feito de dois modos:

- A actividade física é introduzida manualmente pelo utilizador através da escolha de uma série de actividades [16] sendo posteriormente calculadas as kcal equivalentes. Este método peca pela falta de detalhe e/ou equacionamento subjectivo das actividades introduzidas, a sua duração e intensidade.

- Sugere-se a utilização de Sistemas de biomonitorização que incluem (até durante o sono) medição de ritmo cardíaco, acelerómetro de 3 eixos, temperatura da pele e resposta condutora da pele.

f3 - Programa de Cálculo de Ingestão Alimentar:

* De novo visando uma futura aplicação, a introdução de um programa de Cálculo de Ingestão Alimentar pode ser feita de duas maneiras:

- Introdução Manual das quantidades de cada alimento e líquido ingerido e posterior conversão em quantidades ingeridas de hidratos de carbono, proteínas e gorduras (CI, PI e FI).

- Métodos mais automatizados como o utilizado na aplicação na secção 2.2.5, desde a leitura de códigos de barras dos alimentos que são cruzados com uma base de dados interna até balanças dedicadas à conversão automática em kcal após pesagem e identificação do alimento/líquido.

f4 - Mudar a dieta do algoritmo de dimensionamento de Ingestão Alimentar:

* Substituir "*The Zone Diet*"por uma dieta com rácios adequados aos vários tipos de utilizador (dependendo do género, morfologia de corpo, objectivos de treino) aprovada por nutricionistas.

f5 - Aumentar o número de previsões do algoritmo de dimensionamento de Ingestão Alimentar do capítulo 4:

* Aumentar o número de intervalos de acção, por exemplo, o algoritmo actual age de um modo para uma percentagem de perda de peso total superior a 13,75% e de outro modo para valores inferiores a essa percentagem. Actualmente existem dois intervalos diferentes para essa variável. Caso se aumente esse número de intervalos, aumentar-se-á a precisão do algoritmo.

f6 - Introduzir Fase 1 no modelo:

* Desenvolver e Integrar um modelo fidedigno da primeira fase da perda de peso, uma fase mais irregular mas importante para a plenitude do plano dietético.

f7 - Controlar variável PA e EI:

* Estudar as combinações possíveis e saudáveis de variação de Actividade Física e Ingestão Calórica junto a nutricionistas, isto é, verificar qual a melhor relação entre subida de PA e/ou descida de EI para uma perda de peso eficiente e eficaz. Após esse estudo, implementar um dimensionamento de Actividade Física e Ingestão Calórica no Modelo Dinâmico da Mudança de Peso. Partindo dessa nossa trajectória, controlar as duas variáveis em conformidade com o estudo efectuado.

f8 - Automatizar MPC:

* Devido à escassez de tempo, os algoritmos elaborados para a aplicação do MPC são singulares, isto é, cada dia de perturbação tem o seu próprio *Main*. Após a execução individual dos algoritmos respectivos a cada dia (onde cada um lê os valores do anterior para os tomar como referência), outro *Main* global aglomera toda a informação e imprime os resultados finais (consultar figuras 5.13, 5.15 e 5.14). O que se pretende é um programa onde se possam inserir os dias e valores da perturbação e, de seguida, calcule com *Model Predictive Control* o plano de Ingestão Alimentar equivalente.

f9 - Utilizar Software Open Source:

* A transposição do código MATLAB para um software *Open Source*, como Python por exemplo, pode ser uma mais valia para o projecto. Facilitaria a continuação do desenvolvimento das aplicações fora do contexto académico, contexto esse cujas licenças são suportadas pela faculdade. No âmbito da dissertação efectuada, o MATLAB é a ferramenta mais aconselhável devido à sua curva de aprendizagem rapidamente ascendente. No entanto, visando actualizações futuras mais complexas executadas fora desse contexto, é de ponderar

a utilização do Python por esse motivo e outros de nível técnico, como os mencionados em [43], nomeadamente bons rácios linhas de código/rapidez de execução, interoperabilidade entre linguagens (incluindo MATLAB), plataforma de testes embutida, entre outros.

Anexo A

Script Dietas - Sem Controlo

Neste anexo apresenta-se o código elaborado para o projeto relativamente à Dinâmica da Mudança de Peso.

A.1 Script Dietas com Integração das funções de dimensionamento de EI

```
%-------------------------------------------------------------------------------------------
%--------------------------   MODEL INITIALIZATION  ...
    ---------------------------------------
%-------------------------------------------------------------------------------------------

clear all;

%{
%Variable initialization
question1 = 'What is the gender of the subject? (m or f)\n';
gender = input(question1);
if gender == 'm'
    run('male_variable_initialization.m');
else_if gender == 'f'
    run('female_variable_initialization.m');
else
    fprintf('Invalid value\n');
    break;
end;
%}

fprintf('\n\n\n');

%Initial Weight
question1   = 'What is the initial body weight?\n';
```

```
BWinit      = input(question1);

run('variable_initialization.m');

%Physical Activity
PA(1) = 1500;

%{
KEYBORD INPUT
question1 = 'What is the Physical Activity value in kcal?\n';
PA = input(question1);
%}

BW(1) = BWinit;

%Fat Free Mass and Fat Mass Initial Values
FM(1) = (BW(1)./100) .* (0.14*age+37.31 .* log((BW(1).*10000)./(H^2)) - ...
    103.94);
FFM(1)= BW(1) - FM(1);

%Basal Metabolic Rate Inital Value
BMR(1) = (0.024.*FM(1)+0.102.*FFM(1)+0.85).*MJtoKcal;

%Energy Intake for Maintenance Inital Value
EIo(1)  = (PA + BMR(1))./(1-0.0323);
EI(1)   = EIo(1);
CI(1)   =(0.4*EI(1))/a1;
FI(1)   = (0.3*EI(1))/a2;
PI(1)   = (0.3*EI(1))/a3;

%Thermic Effect of Feeding Inital Value
TEF(1) = ac.*CI(1)+af.*FI(1)+ap.*PI(1);

%Energy Expenditure Inital Value
EE(1) = BMR(1)+PA(1)+TEF(1);

%Energy Balance Inital Value
EB(1) = EI(1)-EE(1);

%Energy Balance Rectified
if EB(1) > 0
    EBw(1) = 0.1+EB(1);
elseif EB(1) <= 0
    if EI(1)./EE(1) >=1
        e(1) = 0;
    elseif EI(1)./EE(1) <= 0.5
        e(1) = 0.1;
    else
        e(1) = 0.2*(1-EI(1)./EE(1));
```

```
    end
    EBw(1) = (BMR(1) .* e(1))+EB(1);
end

%Pf present fat percentage Inital Value
Pf(1) = 100 .* (FM(1)./BW(1));

%g(t) Intake Gain Inital Value
g(1) = EI(1)./EIo(1);

%fe Fat Energy Factor Inital Value
if Pf(1) < Pm
    fe(1) = 0;
elseif Pf(1) > Pn
    if g(1) >= 0
        fe(1) = 0.95;
    else
        fe(1) = g(1) + 0.9;
    end
elseif (Pm <= Pf(1) & Pf(1) <= Pn)
    if g(1) >= 0
        fe(1) = -0.2375 + 0.08 .*Pf(1);
    else
        fe(1) = -0.2375 + (g(1)+0.9) .* (0.0833.*Pf(1)-0.25);
    end
end;


%-------------------------------------------------------------------------------------------
%-------------------------------- GOALS - WEIGHT AND TIME ...
    --------------------------------
%-------------------------------------------------------------------------------------------

%Ideal Weight
question2   = 'What is the expected body weight?\n';
BWgoal      = input(question2);
%Weeks of Diet
question3   = 'How many weeks the diet has?\n';
weeks       = input (question3);
%Corresponding Days
DaysofDiet = weeks*7;

%Goal Body Weight loss Percentage
BWgoal_percentage = (1-(BWgoal./BWinit)) .*100;

%Average Percentage to lose each week
average_week_loss = BWgoal_percentage/weeks;
```

```
%Weekly percentage comparing to 5 percent in 3 weeks (1.6667 per week -> ...
    5/3 -> standard value)
standard_week_percent = 5/3 +0.0001;

%Verification of Wealth Standards
if BWgoal_percentage >= 50
   fprintf('*************\nAdvise:\nThe goal weight is below 50%% of your ...
       current weight\n\n');
   break;
end

if average_week_loss > standard_week_percent
    fprintf('*************\nAdvise:\n The weight to lose in the given time ...
        span is not wealthy.\n Please increase the weeks of diet\n\n');
    break;
end

current_BWloss_percentage   = 0;
weekly_BWloss_percentage    = zeros;
EI_deficit                  = zeros;

%-------------------------------------------------------------------------------------------
%--------------------------- WEEKLY ENERGY DEFICIT ...
    ---------------------------------------
%-------------------------------------------------------------------------------------------

%Weekly EI Distribuition

    if BWgoal_percentage < 13.74

      [current_BWloss_percentage , EI_deficit , weekly_BWloss_percentage ] ...
          = weekly_energy_deficit_low_percent( 1450 , weeks, ...
          average_week_loss, standard_week_percent, BWgoal, ...
          BWgoal_percentage );

    else

      [current_BWloss_percentage , EI_deficit , weekly_BWloss_percentage ] ...
          = weekly_energy_deficit_high_percent( 1700 , weeks, ...
          average_week_loss, standard_week_percent, BWgoal, ...
          BWgoal_percentage );

    end;


%-------------------------------------------------------------------------------------------
%-------------------------       PHASE 2       ...
    ----------------------------------------
%-------------------------------------------------------------------------------------------
```

```
%Time/Day Iterator
t=2;
%Time/Week Iterator
j=1;

%Phase 2 - Weight Loss
while t <= DaysofDiet

    %Constant Physical Activity
    PA(t)=1500;

    if BWinit <= 80

        EI(t) = EE(t-1) - ((EI_deficit(t-1))*((BWinit/80)^2.6)); % EI ...
            aproximation

    else

        EI(t) = EE(t-1) - ((EI_deficit(t-1))*((BWinit/80)^2)); % EI ...
            aproximation

    end;

    CI(t) =(0.4*EI(t))/a1; %Based on Diet 40-30-30
    FI(t) = (0.3*EI(t))/a2;
    PI(t) = (0.3*EI(t))/a3;

    TEF(t) = ac.*CI(t)+af.*FI(t)+ap.*PI(t);

    BMR(t) =(0.024.*FM(t-1)+0.102.*FFM(t-1)+0.85).*MJtoKcal;

    EE(t)  = PA(t) + BMR(t) + TEF(t);

    EIo(t) = EE(t);

    %Energy Balance
    EB(t) = EI(t)-EE(t);

    %Energy Balance - Corrected
    if EB(t) > 0
        EBw(t) = 0.1+EB(t);
    elseif EB(t) <= 0
        if EI(t)./EE(t) >=1
            e(t) = 0;
        elseif EI(t)./EE(t) <= 0.5
            e(t) = 0.1;
        else
            e(t) = 0.2*(1-EI(t)./EE(t));
```

```
        end
        EBw(t) = (BMR(t) .* e(t))+EB(t);
    end

    %Pf present fat percentage
    Pf(t) = 100 .* (FM(t-1)./BW(t-1));

    %g(t) Intake Gain Inital Value
    g(t) = EI(t)./EIo(t);

    %fe Fat Energy Factor Inital Value
    if Pf(t) < Pm
        fe(t) = 0;
    elseif Pf(t) > Pn
        if g(t) >= 0
            fe(t) = 0.95;
        else
            fe(t) = g(t) + 0.9;
        end
    elseif (Pm <= Pf(t) & Pf(t) <= Pn)
        if g(t) >= 0
            fe(t) = -0.2375 + 0.08 .*Pf(t);
        else
            fe(t) = -0.2375 + (g(t)+0.9) .* (0.0833.*Pf(t)-0.25);
        end
    end;

    %Fat Free Mass and Fat Mass
    FM(t)  = (fe(t) .* EBw(t))./(pFM*MJtoKcal) + FM(t-1);
    FFM(t) = ((1-fe(t)) .* EBw(t))./(pFFM*MJtoKcal) + FFM(t-1);

    %Body Weight
    BW(t) = FM(t) + FFM(t);

    t = t+1;

end;

%Print of Phase 2 Results
error = (1 - BWgoal/BW(t-1))*100;
fprintf('\n**********************\n');
fprintf('Phase 2 Results:\n');
fprintf('\nFinal BW is = %4.2f \n', BW(t-1));
fprintf('error is = %4.2f %%\n', error);

%-------------------------------------------------------------------------------------------
%-------------------------       PHASE 3       ...
    ----------------------------------------
%-------------------------------------------------------------------------------------------
```

```
%Time/Day Iterator
t_phase2 = (t-1);
t_phase3 = (t-1) + 90; % closer to infinity

%Phase 3 - Weight Maintenance
while t <= t_phase3

    %Constant Physical Activity
    PA(t)=1500;

    if BW(t_phase2) <= BWgoal
        %Iterator
        y = 0;

        while BW(t_phase2 - y) < BWgoal

            EI(t) = EE(t_phase2 - y);
            y = y +1;

        end;

    else

        %Fat Free Mass and Fat Mass Initial Values
        auxFM = (BWgoal./100) .* (0.14*age+37.31 .* ...
            log((BWgoal.*10000)./(H^2)) - 103.94);
        auxFFM = BWgoal - auxFM;

        %Basal Metabolic Rate Inital Value
        auxBMR= (0.024.*auxFM+0.102.*auxFFM+0.85).*MJtoKcal;

        %Energy Intake for Maintenance Inital Value
        EI(t)  = (PA(DaysofDiet+1) + auxBMR)./(1-0.028);

    end;



    CI(t) =(0.4*EI(t))/a1; %Based on Diet 40-30-30
    FI(t) = (0.3*EI(t))/a2;
    PI(t) = (0.3*EI(t))/a3;

    TEF(t) = ac.*CI(t)+af.*FI(t)+ap.*PI(t);

    BMR(t) =(0.024.*FM(t-1)+0.102.*FFM(t-1)+0.85).*MJtoKcal;

    EE(t)  = PA(t) + BMR(t) + TEF(t);
```

```
EIo(t) = EE(t);

%Energy Balance
EB(t) = EI(t)-EE(t);

%Energy Balance - Corrected
if EB(t) > 0
    EBw(t) = 0.1+EB(t);
elseif EB(t) <= 0
    if EI(t)./EE(t) >=1
        e(t) = 0;
    elseif EI(t)./EE(t) <= 0.5
        e(t) = 0.1;
    else
        e(t) = 0.2*(1-EI(t)./EE(t));
    end
    EBw(t) = (BMR(t) .* e(t))+EB(t);
end

%Pf present fat percentage
Pf(t) = 100 .* (FM(t-1)./BW(t-1));

%g(t) Intake Gain Inital Value
g(t) = EI(t)./EIo(t);

%fe Fat Energy Factor Inital Value
if Pf(t) < Pm
    fe(t) = 0;
elseif Pf(t) > Pn
    if g(t) >= 0
        fe(t) = 0.95;
    else
        fe(t) = g(t) + 0.9;
    end
elseif (Pm <= Pf(t) & Pf(t) <= Pn)
    if g(t) >= 0
        fe(t) = -0.2375 + 0.08 .*Pf(t);
    else
        fe(t) = -0.2375 + (g(t)+0.9) .* (0.0833.*Pf(t)-0.25);
    end
end;

%Fat Free Mass and Fat Mass
FM(t)  = (fe(t) .* EBw(t))./(pFM*MJtoKcal) + FM(t-1);
FFM(t) = ((1-fe(t)) .* EBw(t))./(pFFM*MJtoKcal) + FFM(t-1);

%Body Weight
BW(t) = FM(t) + FFM(t);
```

```
    t = t+1;

end;

%Print of Phase 3 Results
error = (1 - BWgoal/BW(t-1))*100;
fprintf('\n**********************\n');
fprintf('Phase 3 Results:\n');
fprintf('\nFinal BW is = %4.2f\n', BW(t-1));
fprintf('error is = %4.2f %%\n', error);

%-------------------------------------------------------------------------------------------
%--------------------------       RESULTS PLOTS       ...
    ------------------------------------
%-------------------------------------------------------------------------------------------

%Vector Time in Weeks

k = 2;
weeks_phase2 = zeros;
weeks_phase2(1) = 1;

while weeks_phase2 < weeks

    weeks_phase2(k)= weeks_phase2(k-1)+1;
    k = k +1;

end;

countweeks_phase2 = k;

k = 2;
weeks_phase3(1) = 1;

while weeks_phase3 < round(t_phase3/7)

    weeks_phase3(k)= weeks_phase3(k-1)+1;
    k = k +1;

end;

countweeks_phase3 = k - 1;

%Vector BW in Weeks

k = 2;
BW_phase2 = zeros;
BW_phase2(1) = BW(1);
```

```
while k < countweeks_phase2

    BW_phase2(k)= BW(k*7);
    k = k +1;

end;

k = 2;
BW_phase3 = zeros;
BW_phase3(1) = BW(1);

while k < countweeks_phase3 + 1

    BW_phase3(k)= BW(k*7-1);
    k = k +1;

end;

%Vector EI in weeks

k = 2;
EI_phase2 = zeros;
EI_phase2(1) = EI(1);


while k < countweeks_phase2

    EI_phase2(k)= EI(k*7);
    k = k +1;

end;

k = 2;
EI_phase3 = zeros;
EI_phase3(1) = EI(1);

while k < countweeks_phase3 +1

    EI_phase3(k)= EI(k*7-1);
    k = k +1;

end;

%Vector BW and EI in days - phase2

j=1;
BW_phase2_days = zeros;
EI_phase2_days = zeros;
```

```
while j <= t_phase2

    BW_phase2_days(j)= BW(j);
    j = j +1;

end;

j=1;
while j <= t_phase2

    EI_phase2_days(j)= EI(j);
    j = j +1;

end;


%Fake Initialization
knowEIphase2 = 0;
knowEIphase3 = 0;

fprintf('\n*****************\n');
question4 = 'Do you wish to know your Energy Intake for phase2? Type:\n[2 ...
    for YES]\n[Other for NO]\n';
knowEIphase2 = input(question4);
fprintf('\n*****************\n');

if knowEIphase2 == 2

    %Week Iterator
    j = 1;
    while j <= weeks

        if  3 * round(double(j)/3) == j

            fprintf('\n');

        end;

        fprintf('EI of week %d is %4.0f kcal |', j , EI(j));
        j = j+1;
    end;

    fprintf('\n*****************\n');

end;

fprintf('\n*****************\n');
```

```
question5 = 'Do you wish to know your Energy Intake for phase3? Type:\n[3 ...
    for YES]\n[Other for NO]\n';
knowEIphase3 = input(question5);

if knowEIphase3 == 3

        fprintf('\n*****************\n');
        fprintf('EI to maintain BWgoal of %d kg is %4.0f kcal |', BWgoal , ...
            EI(countweeks_phase3));
        fprintf('\n*****************\n');

end;

%Extra ICLOCS xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

PAc = [];
CIc = [];
FIc = [];
PIc = [];
TEFc = [];
BMRc = [];
EEc  = [];
EIoc = [];
EBc = [];
EBwc = [];
ec = [];
Pfc = [];
gc = [];
fec = [];

save('datadiet', 'BW','BW_phase2_days','EI_phase2_days', 'BWgoal', ...
    'PA','BMR','MJtoKcal', 'pFM','pFFM', 'a1','a2','a3','ac','ap','af', ...
    'Pn','Pm','t_phase3','t_phase2','FM','FFM','EI', ...
    'PAc','CIc','PIc','FIc','TEFc','BMRc','EEc','EIoc','EBc','EBwc','fec','ec','Pfc','gc');

%xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

%RESULTS

%Figure 1
subplot(2,2,1) % first subplot
xlabel('WEEKS OF DIET')
ylabel('BODY WEIGHT')
plot(weeks_phase2,BW_phase2,'x','MarkerSize',10)
title('BW PHASE 2')

subplot(2,2,2) % second subplot
xlabel('WEEKS OF DIET')
ylabel('BODY WEIGHT')
```

```
plot(weeks_phase3,BW_phase3, 'x','MarkerSize',10)
title('BW PHASE 2 and 3')

subplot(2,2,3) % third subplot
xlabel('WEEKS OF DIET')
ylabel('ENERGY INTAKE')
plot(weeks_phase2,EI_phase2)
title('EI PHASE 2')

subplot(2,2,4) % fourth subplot
xlabel('WEEKS OF DIET')
ylabel('ENERGY INTAKE')
plot(weeks_phase3,EI_phase3)
title('EI PHASE 2 and 3')

%Figure 2
figure
subplot(2,1,1) % first subplot
xlabel('WEEKS OF DIET')
ylabel('BODY WEIGHT')
plot(weeks_phase2,BW_phase2,'x','MarkerSize',10)
title('BW PHASE 2')

subplot(2,1,2) % second subplot
xlabel('WEEKS OF DIET')
ylabel('ENERGY INTAKE')
plot(weeks_phase2,EI_phase2)
title('EI PHASE 2')

%Figure 3
figure
subplot(2,1,1) % first subplot
xlabel('WEEKS OF DIET')
ylabel('BODY WEIGHT')
plot(weeks_phase3,BW_phase3,'x','MarkerSize',10)
title('BW PHASE 3')

subplot(2,1,2) % second subplot
xlabel('WEEKS OF DIET')
ylabel('ENERGY INTAKE')
plot(weeks_phase3,EI_phase3)
title('EI PHASE 3')



%-------------------------------------------------------------------------------------------
%-------------------------------------------------------------------------------------------
%-------------------------------------------------------------------------------------------
```

A.2 Função de dimensionamento de EI para Percentagens de Perda inferiores a 13.75 %

```
function [ current_BWloss_percentage , EI_deficit , ...
    weekly_BWloss_percentage ] = weekly_energy_deficit_low_percent( ...
    maxEI_deficit, weeks, average_week_loss, standard_week_percent, BWgoal, ...
    BWgoal_percentage )

    %Auxiliar Iterator
    j = 1; %week iterator
    k = 1; %day iterator

    %Initializing Variables
    current_BWloss_percentage = 0;
    weekly_BWloss_percentage = zeros;
    EI_deficit = zeros;

    if weeks <= 3

        maxEI_deficit = maxEI_deficit * (BWgoal/78);

        while j <= 3
            weekly_BWloss_percentage(j) = BWgoal_percentage/weeks;
            current_BWloss_percentage = BWgoal_percentage;
            while k <= (7*weeks)
                EI_deficit(k) = ...
                    (weekly_BWloss_percentage(j).*maxEI_deficit)/(standard_week_percent); ...
                    %If standard_week_percent loss in 3 weeks is equivalent ...
                    to maxEI_deficit, weekly_BWloss_percentage is ...
                    equivalente to...
                k = k +1;
            end;
            j = j + 1;
        end;

    elseif (weeks > 3 & weeks <= 6)

        maxEI_deficit = maxEI_deficit * (BWgoal/78);

        if average_week_loss <= 4/3 % Verifies if it can elevate the ...
            current value nearer to 5% in 3 weeks (1.6667 per week -> 5/3 ...
            -> standard)

            while j <= weeks/2
                weekly_BWloss_percentage(j) = average_week_loss + 1/3;
```

```
            current_BWloss_percentage = current_BWloss_percentage + ...
                weekly_BWloss_percentage (j);
            while k <= (weeks/2)*7
                EI_deficit(k) = ...
                    (weekly_BWloss_percentage(j).*maxEI_deficit)/(standard_week_percent);
                k = k +1;
            end;
            j = j + 1;
        end;

        while j <= weeks
            weekly_BWloss_percentage(j) = average_week_loss - 1/3;
            current_BWloss_percentage = current_BWloss_percentage + ...
                weekly_BWloss_percentage (j);
            while k <= (7*weeks)
                EI_deficit(k) = ...
                    (weekly_BWloss_percentage(j).*maxEI_deficit)/(standard_week_percent);
                k = k +1;
            end;
            j = j + 1;
        end;

    else
        while j <= weeks/2
            weekly_BWloss_percentage(j) = standard_week_percent;
            current_BWloss_percentage = current_BWloss_percentage + ...
                weekly_BWloss_percentage (j);
            while k <= (weeks/2)
                EI_deficit(k) = maxEI_deficit;
                k = k +1;
            end;
            j = j + 1;
        end;

        while j <= weeks
            weekly_BWloss_percentage(j) = average_week_loss - ...
                (standard_week_percent-average_week_loss); %Substracts ...
                the excess added above
            current_BWloss_percentage = current_BWloss_percentage + ...
                weekly_BWloss_percentage (j);
            while k <= (7*weeks)
                EI_deficit(k) = ...
                    (weekly_BWloss_percentage(j).*maxEI_deficit)/(standard_week_percent);
                k = k +1;
            end;
            j = j + 1;
        end;
    end;
```

```
elseif weeks > 6 & weeks <= 40

    maxEI_deficit = 1300 * (BWgoal/78);

        while j <= fix(weeks/3) %Round towards zero

            weekly_BWloss_percentage(j) = average_week_loss*1.5;
            current_BWloss_percentage = current_BWloss_percentage + ...
                weekly_BWloss_percentage (j);

            while k <= fix(weeks/3)*7

                EI_deficit(k) = ...
                    (weekly_BWloss_percentage(j).*maxEI_deficit)/(standard_week_percent);
                k = k +1;

            end;
            j = j + 1;
        end;

        while j <= fix(weeks/3)*2 + rem(weeks,3) %rem is Remanining of ...
            division

            weekly_BWloss_percentage(j) = average_week_loss;
            current_BWloss_percentage = current_BWloss_percentage + ...
                weekly_BWloss_percentage (j);

            while k <= (fix(weeks/3)*2 + rem(weeks,3))*7

                EI_deficit(k) = ...
                    (weekly_BWloss_percentage(j).*maxEI_deficit)/(standard_week_percent);
                k = k +1;

            end;

            j = j + 1;

        end;

        while j <= weeks

            weekly_BWloss_percentage(j) = average_week_loss*0.5;
            current_BWloss_percentage = current_BWloss_percentage + ...
                weekly_BWloss_percentage (j);

            while k <= (7*weeks)

                EI_deficit(k) = ...
                    (weekly_BWloss_percentage(j).*maxEI_deficit)/(standard_week_percent);
```

```
                k = k +1;

            end;

            j = j + 1;

        end;

        current_BWloss_percentage = current_BWloss_percentage+0.0001; ...
            %Matlab isn't recognising < comparison if it has exactly ...
            the same value


else

    maxEI_deficit = 1650 * (BWgoal/78);
    ratio = 0.8;

        while j <= fix(weeks/3) %Round towards zero

            weekly_BWloss_percentage(j) = average_week_loss*1.2;
            current_BWloss_percentage = current_BWloss_percentage + ...
                weekly_BWloss_percentage (j);

            while k <= fix(weeks/3)*7

                EI_deficit(k) = ...
                    (weekly_BWloss_percentage(j).*maxEI_deficit)/(standard_week_percent);
                k = k +1;

            end;
            j = j + 1;
        end;

        while j <= fix(weeks/3)*2 + rem(weeks,3) %rem is Remanining of ...
            division

            weekly_BWloss_percentage(j) = average_week_loss;
            current_BWloss_percentage = current_BWloss_percentage + ...
                weekly_BWloss_percentage (j);

            while k <= (fix(weeks/3)*2 + rem(weeks,3))*7

                EI_deficit(k) = ...
                    (weekly_BWloss_percentage(j).*maxEI_deficit)/(standard_week_percent);
                k = k +1;

            end;
```

```
            j = j + 1;

        end;

        while j <= weeks

            weekly_BWloss_percentage(j) = average_week_loss*(1-ratio);
            current_BWloss_percentage = current_BWloss_percentage + ...
                weekly_BWloss_percentage (j);

            while k <= (7*weeks)

                EI_deficit(k) = ...
                    (weekly_BWloss_percentage(j).*maxEI_deficit)/(standard_week_percent);
                k = k +1;

            end;

            j = j + 1;

        end;

        current_BWloss_percentage = current_BWloss_percentage+0.0001; ...
            %Matlab isn't recognising < comparison if it has exactly ...
            the same value

    end;
```

A.3 Função de dimensionamento de EI para Percentagens de Perda superiores a 13.75 %

```
function [ current_BWloss_percentage , EI_deficit , ...
    weekly_BWloss_percentage ] = weekly_energy_deficit_high_loss( ...
    maxEI_deficit, weeks, average_week_loss, standard_week_percent, BWgoal, ...
    BWgoal_percentage )

    %Auxiliar Iterator
    j = 1; %week iterator
    k = 1; %day iterator

    %Initializing Variables
    current_BWloss_percentage = 0;
    weekly_BWloss_percentage = zeros;
    EI_deficit = zeros;

    if weeks <= 3
        while j <= 3
```

```
            weekly_BWloss_percentage(j) = BWgoal_percentage/weeks;
            current_BWloss_percentage = BWgoal_percentage;
            while k <= (7*weeks)
                EI_deficit(k) = ...
                    (weekly_BWloss_percentage(j).*maxEI_deficit)/(standard_week_percent); ...
                    %If standard_week_percent loss in 3 weeks is equivalent ...
                    to maxEI_deficitkcal deficit, weekly_BWloss_percentage ...
                    is equivalente to...
                k = k +1;
            end;
            j = j + 1;
        end;

    elseif (weeks > 3 & weeks <= 6)

        if average_week_loss <= 4/3 % Verifies if it can elevate the ...
            current value nearer to 5% in 3 weeks (1.6667 per week -> 5/3 ...
            -> standard)

            while j <= weeks/2
                weekly_BWloss_percentage(j) = average_week_loss + 1/3;
                current_BWloss_percentage = current_BWloss_percentage + ...
                    weekly_BWloss_percentage (j);
                while k <= (weeks/2)*7
                    EI_deficit(k) = ...
                        (weekly_BWloss_percentage(j).*maxEI_deficit)/(standard_week_percent);
                    k = k +1;
                end;
                j = j + 1;
            end;

            while j <= weeks
                weekly_BWloss_percentage(j) = average_week_loss - 1/3;
                current_BWloss_percentage = current_BWloss_percentage + ...
                    weekly_BWloss_percentage (j);
                while k <= (7*weeks)
                    EI_deficit(k) = ...
                        (weekly_BWloss_percentage(j).*maxEI_deficit)/(standard_week_percent);
                    k = k +1;
                end;
                j = j + 1;
            end;

        else
            while j <= weeks/2
                weekly_BWloss_percentage(j) = standard_week_percent;
                current_BWloss_percentage = current_BWloss_percentage + ...
                    weekly_BWloss_percentage (j);
                while k <= (weeks/2)
```

```
                EI_deficit(k) = maxEI_deficit;
                k = k +1;
            end;
            j = j + 1;
        end;

        while j <= weeks
            weekly_BWloss_percentage(j) = average_week_loss - ...
                (standard_week_percent-average_week_loss); %Substracts ...
                the excess added above
            current_BWloss_percentage = current_BWloss_percentage + ...
                weekly_BWloss_percentage (j);
            while k <= (7*weeks)
                EI_deficit(k) = ...
                    (weekly_BWloss_percentage(j).*maxEI_deficit)/(standard_week_percent);
                k = k +1;
            end;
            j = j + 1;
        end;
    end;

elseif weeks > 6 & weeks <= 40

    maxEI_deficit = 1010 * (BWgoal/60);

        while j <= fix(weeks/3) %Round towards zero

            weekly_BWloss_percentage(j) = average_week_loss*1.5;
            current_BWloss_percentage = current_BWloss_percentage + ...
                weekly_BWloss_percentage (j);

            while k <= fix(weeks/3)*7

                EI_deficit(k) = ...
                    (weekly_BWloss_percentage(j).*maxEI_deficit)/(standard_week_percent);
                k = k +1;

            end;
            j = j + 1;
        end;

        while j <= fix(weeks/3)*2 + rem(weeks,3) %rem is Remanining of ...
            division

            weekly_BWloss_percentage(j) = average_week_loss;
            current_BWloss_percentage = current_BWloss_percentage + ...
                weekly_BWloss_percentage (j);

            while k <= (fix(weeks/3)*2 + rem(weeks,3))*7
```

```
                EI_deficit(k) = ...
                    (weekly_BWloss_percentage(j).*maxEI_deficit)/(standard_week_percent);
                k = k +1;

            end;

            j = j + 1;

        end;

        while j <= weeks

            weekly_BWloss_percentage(j) = average_week_loss*0.5;
            current_BWloss_percentage = current_BWloss_percentage + ...
                weekly_BWloss_percentage (j);

            while k <= (7*weeks)

                EI_deficit(k) = ...
                    (weekly_BWloss_percentage(j).*maxEI_deficit)/(standard_week_percent);
                k = k +1;

            end;

            j = j + 1;

        end;

        current_BWloss_percentage = current_BWloss_percentage+0.0001; ...
            %Matlab isn't recognising < comparison if it has exactly ...
            the same value

    elseif weeks > 41 & weeks <= 99

      maxEI_deficit = 1000 * (BWgoal/60);

        while j <= fix(weeks/3) %Round towards zero

            weekly_BWloss_percentage(j) = average_week_loss*1.5;
            current_BWloss_percentage = current_BWloss_percentage + ...
                weekly_BWloss_percentage (j);

            while k <= fix(weeks/3)*7

                EI_deficit(k) = ...
                    (weekly_BWloss_percentage(j).*maxEI_deficit)/(standard_week_percent);
                k = k +1;
```

```
            end;
            j = j + 1;
        end;

        while j <= fix(weeks/3)*2 + rem(weeks,3) %rem is Remanining of ...
            division

            weekly_BWloss_percentage(j) = average_week_loss;
            current_BWloss_percentage = current_BWloss_percentage + ...
                weekly_BWloss_percentage (j);

            while k <= (fix(weeks/3)*2 + rem(weeks,3))*7

                EI_deficit(k) = ...
                    (weekly_BWloss_percentage(j).*maxEI_deficit)/(standard_week_percent);
                k = k +1;

            end;

            j = j + 1;

        end;

        while j <= weeks

            weekly_BWloss_percentage(j) = average_week_loss*0.5;
            current_BWloss_percentage = current_BWloss_percentage + ...
                weekly_BWloss_percentage (j);

            while k <= (7*weeks)

                EI_deficit(k) = ...
                    (weekly_BWloss_percentage(j).*maxEI_deficit)/(standard_week_percent);
                k = k +1;

            end;

            j = j + 1;

        end;

        current_BWloss_percentage = current_BWloss_percentage+0.0001; ...
            %Matlab isn't recognising < comparison if it has exactly ...
            the same value

    else

        maxEI_deficit = 1250 * (BWgoal/60);
        ratio = 0.8;
```

```
while j <= fix(weeks/3) %Round towards zero

    weekly_BWloss_percentage(j) = average_week_loss*1.2;
    current_BWloss_percentage = current_BWloss_percentage + ...
        weekly_BWloss_percentage (j);

    while k <= fix(weeks/3)*7

        EI_deficit(k) = ...
            (weekly_BWloss_percentage(j).*maxEI_deficit)/(standard_week_percent);
        k = k +1;

    end;
    j = j + 1;
end;

while j <= fix(weeks/3)*2 + rem(weeks,3) %rem is Remanining of ...
    division

    weekly_BWloss_percentage(j) = average_week_loss;
    current_BWloss_percentage = current_BWloss_percentage + ...
        weekly_BWloss_percentage (j);

    while k <= (fix(weeks/3)*2 + rem(weeks,3))*7

        EI_deficit(k) = ...
            (weekly_BWloss_percentage(j).*maxEI_deficit)/(standard_week_percent);
        k = k +1;

    end;

    j = j + 1;

end;

while j <= weeks

    weekly_BWloss_percentage(j) = average_week_loss*(1-ratio);
    current_BWloss_percentage = current_BWloss_percentage + ...
        weekly_BWloss_percentage (j);

    while k <= (7*weeks)

        EI_deficit(k) = ...
            (weekly_BWloss_percentage(j).*maxEI_deficit)/(standard_week_percent);
        k = k +1;

    end;
```

```
            j = j + 1;

        end;

        current_BWloss_percentage = current_BWloss_percentage+0.0001; ...
            %Matlab isn't recognising < comparison if it has exactly ...
            the same value

end;
```

Anexo B

Simulação em Excel do Modelo da Dinâmica da Mudança de Peso

Tabela B.1: Excel - Primeiro Dia de Dieta

1st Cycle				
Variable	Units	Value	EI/ee	0,58422
age	years	20	EBW	-1267,35
BW	kg	80	e(t)	0,08316
H	m	1,8		
EI	kcal/d	2000	FFM(init)	65,2058
EE	kcal/d	3423,39	FM(init)	14,7942
EB	kcal/d	-1423,39	BMR(t)	1876,39
PA	kcal/d	1500	EER	156,034
Pn		15	PF	18,4928
Pm		3	Eio	3423,39
pFM	MJ/kg	38	g(t)	0,58422
pFFM	MJ/kg	6	PF	18,4928
MJ to kcal		238,846	fe(t)	0,95
CI	g	200	p(t)	0,05
FI	g	80	p*EB	-63,3676
PI	g	120	(1-p)*EB	-1203,98
TEF	kcal/d	47		
			FM(gain)	-0,13265
			FFM(gain)	-0,04422
fat %	0,18493		FM(final)	14,6616
			FFM(final)	65,1616
			BW(final)	79,8231

Tabela B.2: Excel - Segundo Dia de Dieta

2nd Cycle				
Variable	Units	Value	EI/ee	0,58453
age	years	20	EBW	-1265,78
BW	kg	79,8231	e(t)	0,08309
H	m	1,8		
EI	kcal/d	2000	FFM(init)	65,1616
EE	kcal/d	3421,55	FM(init)	14,6616
EB	kcal/d	-1421,55	BMR(t)	1874,55
PA	kcal/d	1500	EER	155,763
Pn		15	PF	18,3676
Pm		3	Eio	3421,55
pFM	MJ/kg	38	g(t)	0,58453
pFFM	MJ/kg	6	PF	18,3676
MJ to kcal		238,846	fe(t)	0,95
CI	g	200	p(t)	0,05
FI	g	80	p*EB	-63,2892
PI	g	120	(1-p)*EB	-1202,5
TEF	kcal/d	47		
			FM(gain)	-0,13249
			FFM(gain)	-0,04416
fat %	0,18368		FM(final)	14,5291
			FFM(final)	65,1174
			BW(final)	79,6465

Tabela B.3: Excel - Terceiro Dia de Dieta

3rd Cycle				
Variable	Units	Value	EI/ee	0,58484
age	years	20	EBW	-1264,22
BW	kg	79,6465	e(t)	0,08303
H	m	2,8		
EI	kcal/d	2000	FFM(init)	65,1174
EE	kcal/d	3419,71	FM(init)	14,5291
EB	kcal/d	-1419,71	BMR(t)	1872,71
PA	kcal/d	1500	EER	155,493
Pn		15	PF	18,242
Pm		3	Eio	3419,71
pFM	MJ/kg	38	g(t)	0,58484
pFFM	MJ/kg	6	PF	18,242
MJ to kcal		238,846	fe(t)	0,95
CI	g	200	p(t)	0,05
FI	g	80	p*EB	-63,211
PI	g	120	(1-p)*EB	-1201,01
TEF	kcal/d	47		
			FM(gain)	-0,13233
			FFM(gain)	-0,04411
fat %	0,18242		FM(final)	14,3967
			FFM(final)	65,0733
			BW(final)	79,47

Tabela B.4: Excel - Quarto Dia de Dieta

4th Cycle				
Variable	Units	Value	EI/ee	0,58516
age	years	20	EBW	-1262,66
BW	kg	79,47	e(t)	0,08297
H	m	2,8		
EI	kcal/d	2000	FFM(init)	65,0733
EE	kcal/d	3417,88	FM(init)	14,3967
EB	kcal/d	-1417,88	BMR(t)	1870,88
PA	kcal/d	1500	EER	155,224
Pn		15	PF	18,1159
Pm		3	Eio	3417,88
pFM	MJ/kg	38	g(t)	0,58516
pFFM	MJ/kg	6	PF	18,1159
MJ to kcal		238,846	fe(t)	0,95
CI	g	200	p(t)	0,05
FI	g	80	p*EB	-63,1328
PI	g	120	(1-p)*EB	-1199,52
TEF	kcal/d	47		
			FM(gain)	-0,13216
			FFM(gain)	-0,04405
fat %	0,18116		FM(final)	14,2646
			FFM(final)	65,0292
			BW(final)	79,2938

Tabela B.5: Excel - Quinto dia de Dieta

5th Cycle				
Variable	Units	Value	EI/ee	0,58547
age	years	20	EBW	-1261,09
BW	kg	79,2938	e(t)	0,08291
H	m	2,8		
EI	kcal/d	2000	FFM(init)	65,0292
EE	kcal/d	3416,05	FM(init)	14,2646
EB	kcal/d	-1416,05	BMR(t)	1869,05
PA	kcal/d	1500	EER	154,955
Pn		15	PF	17,9895
Pm		3	Eio	3416,05
pFM	MJ/kg	38	g(t)	0,58547
pFFM	MJ/kg	6	PF	17,9895
MJ to kcal		238,846	fe(t)	0,95
CI	g	200	p(t)	0,05
FI	g	80	p*EB	-63,0547
PI	g	120	(1-p)*EB	-1198,04
TEF	kcal/d	47		
			FM(gain)	-0,132
			FFM(gain)	-0,044
fat %	0,1799		FM(final)	14,1326
			FFM(final)	64,9852
			BW(final)	79,1178

Anexo C

Dimensionamento de EI com ICLOCS - Fase 2

Neste anexo apresenta-se o código ICLOCS alterado para o problema em causa.

C.1 Main

```
%MAINMPC - Main script to solve the Model Predictive Control Problem
%
% Copyright (C) 2010 Paola Falugi, Eric Kerrigan and Eugene van Wyk. All ...
    Rights Reserved.
% This code is published under the BSD License.
% Department of Electrical and Electronic Engineering,
% Imperial College London London  England, UK
% ICLOCS (Imperial College London Optimal Control) 5 May 2010
% iclocs@imperial.ac.uk


%-----------------------------------------------------------------------


[problem,guess]= Discrete_Sys;           % Fetch the problem definition
options= settings_Dis;                      % Get options and solver settings

[infoNLP,data]=transcribeOCP(problem,guess,options); % Format for NLP solver
[nt,np,n,m,ng,nb,M,N,ns]=deal(data.sizes{:});
time=[];states=[];inputs=[];



solution = solveNLP(infoNLP,data);               % Solve the NLP
```

```
load('datadiet');
load('controlvar');

solution.U(:,1)=Utimming(:,1);

%-----------------------------------------
%EI and BW with Control comparing with EI and BW from reference
figure(1);
plot(solution.T,solution.X(:,1)+solution.X(:,2), 'k-')
hold on
plot (solution.T, BW_phase2_days, 'r-')
title('BWphase2','FontSize',40,'Color','k')
xlabel('DAYS OF DIET','FontSize',20, 'Color','b')
ylabel('BODY WEIGHT','FontSize',20,'Color', 'b')

figure(2);
plot(solution.T,solution.X(:,3),'k')
hold on
plot (solution.T, EI_phase2_days, 'r-')
title('EIphase2','FontSize',40,'Color','k')
xlabel('DAYS OF DIET','FontSize',20, 'Color','b')
ylabel('ENERGY INTAKE','FontSize',20, 'Color','b')

%-----------------------------------------
%EI and BW with Control
figure(3)
subplot(1,2,1) % first subplot
plot(solution.T,solution.X(:,1)+solution.X(:,2), 'k-')
title('BWphase2','FontSize',40,'Color','k')
xlabel('DAYS OF DIET','FontSize',20, 'Color','b')
ylabel('BODY WEIGHT','FontSize',20,'Color', 'b')

subplot(1,2,2) % second subplot
plot (solution.T,solution.X(:,3),'k')
title('EIphase2','FontSize',40,'Color','k')
xlabel('DAYS OF DIET','FontSize',20, 'Color','b')
ylabel('ENERGY INTAKE','FontSize',20, 'Color','b')

%-----------------------------------------
%EI with Control and U(t)
figure(4);
subplot(1,2,1) % first subplot
plot(solution.T,solution.X(:,3),'k')
title('EIphase2 with Control','FontSize',40,'Color','k')
xlabel('DAYS OF DIET','FontSize',20, 'Color','b')
ylabel('ENERGY INTAKE','FontSize',20, 'Color','b')

subplot(1,2,2) % second subplot
plot(solution.T,Utimming(:,1),'-ko',...
```

```
                'MarkerEdgeColor','k',...
                'MarkerFaceColor',[.49 1 .63],...
                'MarkerSize',10)
title('Control U','FontSize',40,'Color','k')
xlabel('DAYS OF DIET','FontSize',20, 'Color','b')
ylabel('CONTROL','FontSize',20, 'Color','b')
```

C.2 Problem Phase 2

```
function [problem,guess] = Discrete_Sys

%Discrete time problem - Define the optimal control problem for a discrete
%                        time system
%
% Syntax:  [problem,guess] = Discrete_Sys
%
% Outputs:
%    problem - Structure with information on the optimal control problem
%    guess   - Guess for state, control and multipliers.
%
% Other m-files required: none
% Subfunctions: L (stageCost),
%       E (boundaryCost),
%       f (ODE right-hand side),
%       g (path constraints),
%       b (boundary constraints)
% MAT-files required: termset.mat
%
% Copyright (C) 2010 Paola Falugi, Eric Kerrigan and Eugene van Wyk. All ...
    Rights Reserved.
% This code is published under the BSD License.
% Department of Electrical and Electronic Engineering,
% Imperial College London London  England, UK
% ICLOCS (Imperial College London Optimal Control) 5 May 2010
% iclocs@imperial.ac.uk

%------------- BEGIN CODE --------------

load('datadiet');


% Initial time. t0<tf.  For discrete time systems is the initial index

problem.time.t0=0;

% Final time. Let tf_min=tf_max if tf is fixed. tf_min=tf_max=[] for
% discrete time systems
```

```
problem.time.tf_min=[];
problem.time.tf_max=[];
guess.tf=[];

% Parameter bounds. pl=< p <=pu
problem.parameters.pl=[];
problem.parameters.pu=[];
guess.parameters=[];


% Initial conditions for system. Bounds if x0 is free s.t. x0l=< x0 <=x0u

problem.states.x0=[FM(1) FFM(1) EI(1)];

problem.states.x0l=problem.states.x0;
problem.states.x0u=problem.states.x0l;

% State bounds. xl=< x <=xu
problem.states.xl=[0.5 29 1500];
problem.states.xu=[445 200 20000];

% Terminal state bounds. xfl=< xf <=xfu
problem.states.xfl=[0.5 29 2000];
problem.states.xfu=[445 200 20000];

% Guess the state trajectories with [x0 xf]

guess.states(:,1)=[problem.states.x0(1) 0];
guess.states(:,2)=[problem.states.x0(2) 0];
guess.states(:,3)=[problem.states.x0(3) 0];

% Number of control actions N
% Set problem.inputs.N=0 if N is equal to the number of integration steps.
% Note that the number of integration steps defined in settings.m has to be ...
    divisible
% by the  number of control actions N whenever it is not zero.

problem.inputs.N=3;

% Input bounds
problem.inputs.ul=-inf;
problem.inputs.uu=inf;

% Guess the input sequences with [u0 uf]
guess.inputs(:,1)=[-500 300];


%g1=BW g2=EE g3= EE-1.2*BMR
```

```
% Bounds for path constraint function gl =< g(x,u,p,t) =< gu
problem.constraints.gl=[29 0 0];
problem.constraints.gu=[635 10000 10000];

% Bounds for boundary constraints bl =< b(x0,xf,u0,uf,p,t0,tf) =< bu

problem.constraints.bl=[BWgoal -1.5 -inf 0];
problem.constraints.bu=[BWgoal -1.5 0 inf];

% Choose the set-points if required
problem.setpoints.states=[FM(1) FFM(1) EI(1)];
problem.setpoints.inputs=[0];

% store the necessary problem parameters used in the functions

problem.data.BW=BW';
problem.data.a1=a1;
problem.data.a2=a2;
problem.data.a3=a3;
problem.data.ac=ac;
problem.data.ap=ap;
problem.data.af=af;
problem.data.MJtoKcal=MJtoKcal;
problem.data.Pn=Pn;
problem.data.Pm=Pm;
problem.data.pFM=pFM;
problem.data.pFFM=pFFM;
problem.data.t_phase2=t_phase2;
problem.data.Utimming=zeros(1,t_phase2);

% Get function handles and return to Main.m
problem.functions={@L,@E,@f,@g,@b};

%------------- END OF CODE --------------

function stageCost=L(x,xr,u,ur,p,t,data)

% L - Returns the stage cost.
% The function must be vectorized and
% xi, ui are column vectors taken as x(:,i) and u(:,i) (i denotes the i-th
% variable)
%
% Syntax:  stageCost = L(x,xr,u,ur,p,t,data)
%
% Inputs:
%    x  - state vector
%    xr - state reference
%    u  - input
%    ur - input reference
```

```
%    p  - parameter
%    t  - time
%    data- structured variable containing the values of additional data ...
    used inside
%          the function
%
% Output:
%    stageCost - Scalar or vectorized stage cost
%
%  Remark: If the stagecost does not depend on variables it is necessary to ...
    multiply
%          the assigned value by t in order to have right vector dimesion ...
    when called for the optimization.
%          Example: stageCost = 0*t;

%------------- BEGIN CODE --------------

stageCost = 0*t;



%------------- END OF CODE --------------


function boundaryCost=E(x0,xf,u0,uf,p,tf,data)

% E - Returns the boundary value cost
%
% Syntax:  boundaryCost=E(x0,xf,u0,uf,p,tf,data)
%
% Inputs:
%    x0  - state at t=0
%    xf  - state at t=tf
%    u0  - input at t=0
%    uf  - input at t=tf
%    p   - parameter
%    tf  - final time
%    data- structured variable containing the values of additional data ...
    used inside
%          the function
%
% Output:
%    boundaryCost - Scalar boundary cost
%
%------------- BEGIN CODE --------------

boundaryCost=0;
```

```
%------------- END OF CODE --------------


function dx = f(x,u,p,t,data)

% f - Returns the ODE right hand side where x'= f(x,u,p,t)
% The function must be vectorized and
% xi, ui, pi are column vectors taken as x(:,i), u(:,i) and p(:,i). Each
% state corresponds to one column of dx.
%
%
% Syntax:  dx = f(x,u,p,t,data)
%
% Inputs:
%    x  - state vector
%    u  - input
%    p  - parameter
%    t  - time
%    data-structured variable containing the values of additional data used ...
    inside
%          the function
%
% Output:
%    dx - time derivative of x
%
%  Remark: If the i-th ODE right hand side does not depend on variables it ...
    is necessary to multiply
%          the assigned value by a vector of ones with the same length  of ...
    t  in order
%          to have  a vector with the right dimesion  when called for the ...
    optimization.
%          Example: dx(:,i)= 0*ones(size(t,1));
%
%------------- BEGIN CODE --------------

x1=x(:,1); x2=x(:,2); x3=x(:,3);

a1=data.a1;
a2=data.a2;
a3=data.a3;
ac=data.ac;
ap=data.ap;
af=data.af;
MJtoKcal=data.MJtoKcal;
Pn=data.Pn;
Pm=data.Pm;
pFM=data.pFM;
pFFM=data.pFFM;
```

```
t_phase2=data.t_phase2;
EBwc=0;
ec=0;
fec=0;
%XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PAc = 1500;


%Initial Values

%Basal Metabolic Rate Inital Value
BMRc(1) = (0.024.*x1(1)+0.102.*x2(1)+0.85).*MJtoKcal;

%Energy Intake for Maintenance Inital Value
EIoc(1)  = (PAc + BMRc(1))./(1-0.0323);
CIc(1)   =(0.4*x3(1))/a1;
FIc(1)   = (0.3*x3(1))/a2;
PIc(1)   = (0.3*x3(1))/a3;

%Thermic Effect of Feeding Inital Value
TEFc(1) = ac.*CIc(1)+af.*FIc(1)+ap.*PIc(1);

%Energy Expenditure Inital Value
EEc(1) = BMRc(1)+PAc+TEFc(1);

%Energy Balance Inital Value
EBc(1) = x3(1)-EEc(1);

%Energy Balance Rectified
if EBc(1) > 0
    EBwc(1) = 0.1+EBc(1);
elseif EBc(1) <= 0
    if x3(1)/EEc(1) >=1
        ec(1) = 0;
    elseif x3(1)/EEc(1) <= 0.5
        ec(1) = 0.1;
    else
        ec(1) = 0.2*(1-x3(1)./EEc(1));
    end
    EBwc(1) = (BMRc(1) .* ec(1))+EBc(1);
end

%Pf present fat percentage Inital Value
Pfc(1) = 100 .* (x1(1)./(x1(1)+x2(1)));

%g(t) Intake Gain Inital Value
gc(1) = x3(1)./EIoc(1);

%fe Fat Energy Factor Inital Value
```

```
if Pfc(1) < Pm
    fec(1) = 0;
elseif Pfc(1) > Pn
    if gc(1) >= 0
        fec(1) = 0.95;
    else
        fec(1) = gc(1) + 0.9;
    end
elseif (Pm <= Pfc(1) & Pfc(1) <= Pn)
    if gc(1) >= 0
        fec(1) = -0.2375 + 0.08 .*Pfc(1);
    else
        fec(1) = -0.2375 + (gc(1)+0.9) .* (0.0833.*Pfc(1)-0.25);
    end
end;

%XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

for ite = 1:length(t)

    if ite==1
        timming(ite)=1.1;
    elseif ite == fix(t_phase2/3)
        timming(ite)=-0.5;
    elseif ite == fix(t_phase2/3)*2 + rem(t_phase2,3)
        timming(ite)=0.5;
    else
        timming(ite)=0;
    end
end

    timming=timming';
    Utimming=u(:,1).*timming;

    save('controlvar','Utimming');

    dx(:,3)=x3+Utimming;
    x3= dx(:,3);

%XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

 for ite = 2:length(t)

    CIc = (0.4*x3)/a1; %Based on Diet 40-30-30
    FIc = (0.3*x3)/a2;
    PIc = (0.3*x3)/a3;

    TEFc = ac.*CIc+af.*FIc+ap.*PIc;
```

```
    BMRc =(0.024.*x1+0.102.*x2+0.85).*MJtoKcal;

    EEc  = PAc + BMRc + TEFc;

    EIoc = EEc;

    %Energy Balance
    EBc = x3-EEc;

%Energy Balance - Corrected
    if EBc(ite) > 0
        EBwc(ite) = 0.1+EBc(ite);
    elseif EBc(ite) <= 0
        if x3(ite)./EEc(ite) >=1
            ec(ite) = 0;
        elseif x3(ite)./EEc(ite) <= 0.5
            ec(ite) = 0.1;
        else
            ec(ite) = 0.2*(1-x3(ite)./EEc(ite));
        end
        EBwc(ite) = (BMRc(ite) .* ec(ite))+EBc(ite);
    end

    %Pf present fat percentage
    Pfc(ite) = 100 .* (x1(ite-1)./(x1(ite-1)+x2(ite-1)));

    %g(t) Intake Gain Inital Value
    gc(ite) = x3(ite)./EIoc(ite);

    %fe Fat Energy Factor Inital Value
    if Pfc(ite) < Pm
        fec(ite) = 0;
    elseif Pfc(ite) > Pn
        if gc(ite) >= 0
            fec(ite) = 0.95;
        else
            fec(ite) = gc(ite) + 0.9;
        end
    elseif (Pm <= Pfc(ite) & Pfc(ite) <= Pn)
        if gc(ite) >= 0
            fec(ite) = -0.2375 + 0.08 .*Pfc(ite);
        else
            fec(ite) = -0.2375 + (gc(ite)+0.9) .* (0.0833.*Pfc(ite)-0.25);
        end
    end;
 end;

%XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

```
num1=(fec.*EBwc)';
num2=((1-fec).*EBwc)';
den1=pFM*MJtoKcal;
den2=pFFM*MJtoKcal;

%num2/den2

dx(:,1) = (num1 /den1) + x1;
dx(:,2) = (num2 /den2) + x2;



%------------- END OF CODE --------------


function c=g(x,u,p,t,data)

% g - Returns the path constraint function where gl =< g(x,u,p,t) =< gu
% The function must be vectorized and
% xi, ui, pi are column vectors taken as x(:,i), u(:,i) and p(:,i). Each
% constraint corresponds to one column of c
%
% Syntax:  c=g(x,u,p,t,data)
%
% Inputs:
%    x  - state vector
%    u  - input
%    p  - parameter
%    t  - time
%   data- structured variable containing the values of additional data used ...
    inside
%         the function
%
% Output:
%    c - constraint function
%
%------------- BEGIN CODE --------------


PA1=1500;
a1=data.a1;
a2=data.a2;
a3=data.a3;
ac=data.ac;
ap=data.ap;
af=data.af;
MJtoKcal=data.MJtoKcal;
```

```
x1=x(:,1);
x2=x(:,2);
x3=x(:,3);

c=[x1+x2,...
   PA1-0.2.*(0.024.*x1+0.102.*x2+0.85).*MJtoKcal+ac.*((0.4/a1)*x3)+af.*((0.3/a2)*x3)+ap.*((0.3/a3)
   PA1+(0.024.*x1+0.102.*x2+0.85).*MJtoKcal+ac.*((0.4/a1)*x3)+af.*((0.3/a2)*x3)+ap.*((0.3/a3)*x3)]


%------------- END OF CODE --------------

function bc=b(x0,xf,u0,uf,p,tf,data)

% b - Returns a column vector containing the evaluation of the boundary ...
    constraints: bl =< bf(x0,xf,u0,uf,p,t0,tf) =< bu
%
% Syntax:  bc=b(x0,xf,u0,uf,p,tf,data)
%
% Inputs:
%    x0  - state at t=0
%    xf  - state at t=tf
%    u0  - input at t=0
%    uf  - input at t=tf
%    p   - parameter
%    tf  - final time
%    data- structured variable containing the values of additional data ...
    used inside
%          the function
%
%
% Output:
%    bc - column vector containing the evaluation of the boundary function
%
%------------- BEGIN CODE --------------

x1  = xf(1,:);
x2  = xf(2,:);

bc = [x1+x2;u0/uf;u0;uf];

%------------- END OF CODE --------------
```

C.3 Settings

```
function options = settings_Dis

load('datadiet');
```

```
%SETTINGS - General and solver-specific settings are selected here
% Unless specified otherwise the options are set using 0 => no and 1 => yes
%
% Syntax:  options = settings_Dis
%
% Output:
%    options - Structure containing the settings
%
% Other m-files required: none
% Subfunctions: none
% MAT-files required: none
%
% Copyright (C) 2010 Paola Falugi, Eric Kerrigan and Eugene van Wyk. All ...
    Rights Reserved.
% This code is published under the BSD License.
% Department of Electrical and Electronic Engineering,
% Imperial College London London  England, UK
% ICLOCS (Imperial College London Optimal Control) 5 May 2010
% iclocs@imperial.ac.uk

%------------- BEGIN CODE --------------


% Transcription Method:
%---------------------------------------
% Discrete-time model       ('discrete')
% Multiple shooting method  ('multiple_shooting') WARNING: The
%                                                 'quasi-newton' option for ...
    the hessian
%                                                  computation has to be ...
    selected
%                                                 ...
    (options.ipopt.hessian_approximation='limited-memory').
% Euler method              ('euler')
% Trapezoidal method        ('trapezoidal')
% Hermite-Simpson method    ('hermite')


options.transcription='discrete';

% Derivative generation :
%---------------------------------------
% Whenever the analytic differentiation is enabled it is necessary to
% specify the available analytic forms for the cost function, the dynamic ...
    equations
% and  the constraints in the appropriate files .m

% Numerical differentiation: finite differences  ('numeric')
```

```
% Analytic differentiation: analytic gradients   ('analytic')

options.derivatives='numeric';


% Numeric generation of the Hessian:
%-----------------------------------------------------------------

% Whenever the numeric differentiation is enabled it is necessary to
% specify which kind of finite difference approximation to use  between
% the following ones:
%
% Central difference ('central')
% forward difference ('forward')

options.hessianFD='central';


%  The perturbation size for numerical second derivatives
%  can be set in options.perturbation.H. The perturbation size for ...
    numerical first derivatives
%  can be set in  options.perturbation.J.
%  It is possible to select default values for the perturbations by setting ...
     options.perturbation.H and
%  options.perturbation.J to the empty matrix.
%  The default values for the gradient approximation is (eps/2)^(1/3)
%  while for the  second derivative is (8*eps)^(1/3).

options.perturbation.H=[];  % Perturbation size for the second derivatives
options.perturbation.J=[];  % Perturbation size for the first derivatives



% NLP solver
%---------------------------------------
% IPOPT: recommended but needs ipopt.mex        ('ipopt')
% fmincon                                       ('fmincon')

options.NLPsolver='ipopt';

% IPOPT settings (if required)
options.ipopt.tol=1e-9;                         % Desired convergence ...
    tolerance (relative). The default value is  1e-8.
options.ipopt.print_level=5;                    % Print level. The valid ...
    range for this integer option is [0,12] and its default value is 5.
options.ipopt.max_iter=3000;                    % Maximum number of ...
    iterations. The default value is 3000.
```

```
options.ipopt.mu_strategy ='adaptive';          % Determines which barrier ...
    parameter update strategy is to be used.
                                                % The default value for this ...
                                                    string option is "monotone".
                                                % Possible values:
                                                %   'monotone': use the ...
                                                    monotone ...
                                                    (Fiacco-McCormick) strategy
                                                %   'adaptive': use the ...
                                                    adaptive update strategy

options.ipopt.hessian_approximation='exact';    %  Indicates what ...
    information for the Hessian of the Lagrangian function is
                                                %  used by the algorithm. ...
                                                    The default value is ...
                                                    'exact'.
                                                %  Possible values:
                                                %   'exact': Use second ...
                                                    derivatives provided by ...
                                                    ICLOCS.
                                                %   'limited-memory': ...
                                                    Perform a limited-memory ...
                                                    quasi-Newton approximation
                                                %    implemented inside IPOPT

%options.ipopt.limited_memory_max_history=6;   % Maximum size of the ...
    history for the limited quasi-Newton Hessian approximation. The valid ...
    range for this integer option is [0, +inf)
                                                % and its default value is 6.
%options.ipopt.limited_memory_max_skipping=1;  % Threshold for successive ...
    iterations where update is skipped for the quasi-Newton approximation.
                                                % The valid range for this ...
                                                    integer option is ...
                                                    [1,+inf) and its default ...
                                                    value is 2.


% fmincon settings (if required)
 options.fmincon=optimset;


% Automatic scaling (recommended)
%-----------------------------------------
options.scaling=1;


% Output settings
%-----------------------------------------
```

```
% Display computation time
options.print.time=1;

% Display relative local discretization error (recommended for direct ...
    transcription)
options.print.relative_local_error=1;


% Display cost
options.print.cost=1;


% Plot states
options.plot.states=1;

% Plot inputs
options.plot.inputs=1;

% Plot Lagrange multipliers
options.plot.multipliers=1;


% Direct transcription settings
%----------------------------------------

% Number of   integration nodes in the interval t=[0,tf]; nodes=steps+1.
% The quantity steps/N (N number of control actions) must be a positive
% integer.
options.nodes=t_phase2;

% Distribution of integration steps. Set tau=0 for equispaced steps.
% Otherwise: tau is a vector of length M-1 with 0<tau(i)<1 and sum(tau)=1.
% For discrete time system  set  tau=0.

options.tau=0;


% Multiple shooting settings
%----------------------------------------
% N/S=normal/stiff. H/M/L=high/medium/low accuracy
%
% 'cvodes'   N/S    H    High accuracy, difficult problems(slow)
%
%
%  Note: 'cvodes' requires the sundialsTB.
%


options.ODEsolver='cvodes';
```

```
% CVODES settings (if required)
Method='Adams';  % Method: Adams, BDF
Solver='Newton'; % Solver: Newton, Functional (requires dfdx)




% % Options for forward integration when cvodes is enabled
% %
% options.cvodes = CVodeSetOptions('RelTol',1.e-4,...
%                            'AbsTol',1.e-6,...
%                            'LinearSolver','Dense',...
%                            'MaxNumSteps',10000,...
%                            'LMM',Method,...
%                            'NonlinearSolver',Solver);
%
% % Forward sensitivity options when cvodes is enabled
%
%
%
% ...
    options.cvodesf=CVodeSensSetOptions('ErrControl',true,'method','Staggered'); ...
    % FSA initialization
%
%
%
```

Anexo D

Dimensionamento de EI com ICLOCS - Fase 3

Neste anexo apresenta-se o código ICLOCS alterado para o problema em causa.

D.1 Main

```
%MAINMPC - Main script to solve the Model Predictive Control Problem
%
% Copyright (C) 2010 Paola Falugi, Eric Kerrigan and Eugene van Wyk. All ...
    Rights Reserved.
% This code is published under the BSD License.
% Department of Electrical and Electronic Engineering,
% Imperial College London London  England, UK
% ICLOCS (Imperial College London Optimal Control) 5 May 2010
% iclocs@imperial.ac.uk


%-----------------------------------------------------------------------


[problem,guess]= Discrete_Sys_phase3;           % Fetch the problem definition
options= settings_Dis_phase3;                     % Get options and solver ...
    settings

[infoNLP,data]=transcribeOCP(problem,guess,options); % Format for NLP solver
[nt,np,n,m,ng,nb,M,N,ns]=deal(data.sizes{:});
time=[];states=[];inputs=[];



solution = solveNLP(infoNLP,data);               % Solve the NLP
```

```
load('datadiet');
load('controlvar2');

solution.U(:,1)=Utimming2(:,1);
%-----------------------------------------
%EI and BW with Control comparing with EI and BW from reference
figure(1);
plot(solution.T,solution.X(:,1)+solution.X(:,2), 'k-')
title('BWphase3','FontSize',40,'Color','k')
xlabel('DAYS OF DIET','FontSize',20, 'Color','b')
ylabel('BODY WEIGHT','FontSize',20,'Color', 'b')

figure(2);
plot(solution.T,solution.X(:,3),'k')
title('EIphase3','FontSize',40,'Color','k')
xlabel('DAYS OF DIET','FontSize',20, 'Color','b')
ylabel('ENERGY INTAKE','FontSize',20, 'Color','b')

%-----------------------------------------
%EI and BW with Control
figure(3)
subplot(1,2,1) % first subplot
plot(solution.T,solution.X(:,1)+solution.X(:,2), 'k-')
title('BWphase3','FontSize',40,'Color','k')
xlabel('DAYS OF DIET','FontSize',20, 'Color','b')
ylabel('BODY WEIGHT','FontSize',20,'Color', 'b')

subplot(1,2,2) % second subplot
plot (solution.T,solution.X(:,3),'k')
title('EIphase3','FontSize',40,'Color','k')
xlabel('DAYS OF DIET','FontSize',20, 'Color','b')
ylabel('ENERGY INTAKE','FontSize',20, 'Color','b')

%-----------------------------------------
%EI with Control and U(t)
figure(4);
subplot(1,2,1) % first subplot
plot(solution.T,solution.X(:,3),'k')
title('EIphase3 with Control','FontSize',40,'Color','k')
xlabel('DAYS OF DIET','FontSize',20, 'Color','b')
ylabel('ENERGY INTAKE','FontSize',20, 'Color','b')

subplot(1,2,2) % second subplot
plot(solution.T,solution.U(:,1),'-ko',...
                'MarkerEdgeColor','k',...
                'MarkerFaceColor',[.49 1 .63],...
                'MarkerSize',10)
title('Control U','FontSize',40,'Color','k')
```

```
xlabel('DAYS OF DIET','FontSize',20, 'Color','b')
ylabel('CONTROL','FontSize',20, 'Color','b')
```

D.2 Problem Phase 3

```
function [problem,guess] = Discrete_Sys_phase3

%Discrete time problem - Define the optimal control problem for a discrete
%                        time system
%
% Syntax:  [problem,guess] = Discrete_Sys
%
% Outputs:
%    problem - Structure with information on the optimal control problem
%    guess   - Guess for state, control and multipliers.
%
% Other m-files required: none
% Subfunctions: L (stageCost),
%       E (boundaryCost),
%       f (ODE right-hand side),
%       g (path constraints),
%       b (boundary constraints)
% MAT-files required: termset.mat
%
% Copyright (C) 2010 Paola Falugi, Eric Kerrigan and Eugene van Wyk. All ...
    Rights Reserved.
% This code is published under the BSD License.
% Department of Electrical and Electronic Engineering,
% Imperial College London London  England, UK
% ICLOCS (Imperial College London Optimal Control) 5 May 2010
% iclocs@imperial.ac.uk

%------------- BEGIN CODE --------------

load('datadiet');
load('data_ref');

% Initial time. t0<tf.  For discrete time systems is the initial index

problem.time.t0=0;

% Final time. Let tf_min=tf_max if tf is fixed. tf_min=tf_max=[] for
% discrete time systems

problem.time.tf_min=[];
problem.time.tf_max=[];
guess.tf=[];
```

```
% Parameter bounds. pl=< p <=pu
problem.parameters.pl=[];
problem.parameters.pu=[];
guess.parameters=[];


% Initial conditions for system. Bounds if x0 is free s.t. x0l=< x0 <=x0u

problem.states.x0=[x1phase2(t_phase2) x2phase2(t_phase2) x3phase2(t_phase2)];

problem.states.x0l=problem.states.x0;
problem.states.x0u=problem.states.x0l;

% State bounds. xl=< x <=xu
problem.states.xl=[0.5 29 1500];
problem.states.xu=[445 200 20000];

% Terminal state bounds. xfl=< xf <=xfu
problem.states.xfl=[0.5 29 2000];
problem.states.xfu=[445 200 20000];

% Guess the state trajectories with [x0 xf]

guess.states(:,1)=[problem.states.x0(1) problem.states.x0(1)];
guess.states(:,2)=[problem.states.x0(2) problem.states.x0(2)];
guess.states(:,3)=[problem.states.x0(3) problem.states.x0(3)];

% Number of control actions N
% Set problem.inputs.N=0 if N is equal to the number of integration steps.
% Note that the number of integration steps defined in settings.m has to be ...
    divisible
% by the  number of control actions N whenever it is not zero.

problem.inputs.N=1;

% Input bounds
problem.inputs.ul=-inf;
problem.inputs.uu=inf;

% Guess the input sequences with [u0 uf]
guess.inputs(:,1)=[0 0];


%g1=BW g2=EE g3= EE-1.2*BMR
% Bounds for path constraint function gl =< g(x,u,p,t) =< gu
problem.constraints.gl=[29 0 0];
problem.constraints.gu=[635 10000 10000];
```

```
% Bounds for boundary constraints bl =< b(x0,xf,u0,uf,p,t0,tf) =< bu

problem.constraints.bl=BWgoal;
problem.constraints.bu=BWgoal;

% Choose the set-points if required
problem.setpoints.states=[];
problem.setpoints.inputs=[];

% store the necessary problem parameters used in the functions

problem.data.BW=BW';
problem.data.a1=a1;
problem.data.a2=a2;
problem.data.a3=a3;
problem.data.ac=ac;
problem.data.ap=ap;
problem.data.af=af;
problem.data.MJtoKcal=MJtoKcal;
problem.data.Pn=Pn;
problem.data.Pm=Pm;
problem.data.pFM=pFM;
problem.data.pFFM=pFFM;
problem.data.t_phase2=t_phase2;
problem.data.BWgoal=BWgoal;

% Get function handles and return to Main.m
problem.functions={@L,@E,@f,@g,@b};

%------------- END OF CODE --------------

function stageCost=L(x,xr,u,ur,p,t,data)

% L - Returns the stage cost.
% The function must be vectorized and
% xi, ui are column vectors taken as x(:,i) and u(:,i) (i denotes the i-th
% variable)
%
% Syntax:  stageCost = L(x,xr,u,ur,p,t,data)
%
% Inputs:
%    x  - state vector
%    xr - state reference
%    u  - input
%    ur - input reference
%    p  - parameter
%    t  - time
%    data- structured variable containing the values of additional data ...
    used inside
```

```
%          the function
%
% Output:
%    stageCost - Scalar or vectorized stage cost
%
%  Remark: If the stagecost does not depend on variables it is necessary to ...
    multiply
%          the assigned value by t in order to have right vector dimesion ...
    when called for the optimization.
%          Example: stageCost = 0*t;

%------------- BEGIN CODE --------------

stageCost =0*t;



%------------- END OF CODE --------------


function boundaryCost=E(x0,xf,u0,uf,p,tf,data)

% E - Returns the boundary value cost
%
% Syntax:  boundaryCost=E(x0,xf,u0,uf,p,tf,data)
%
% Inputs:
%    x0  - state at t=0
%    xf  - state at t=tf
%    u0  - input at t=0
%    uf  - input at t=tf
%    p   - parameter
%    tf  - final time
%    data- structured variable containing the values of additional data ...
    used inside
%          the function
%
% Output:
%    boundaryCost - Scalar boundary cost
%
%------------- BEGIN CODE --------------

boundaryCost=0;



%------------- END OF CODE --------------
```

```
function dx = f(x,u,p,t,data)

% f - Returns the ODE right hand side where x'= f(x,u,p,t)
% The function must be vectorized and
% xi, ui, pi are column vectors taken as x(:,i), u(:,i) and p(:,i). Each
% state corresponds to one column of dx.
%
%
% Syntax:  dx = f(x,u,p,t,data)
%
% Inputs:
%    x  - state vector
%    u  - input
%    p  - parameter
%    t  - time
%    data-structured variable containing the values of additional data used ...
    inside
%          the function
%
% Output:
%    dx - time derivative of x
%
%  Remark: If the i-th ODE right hand side does not depend on variables it ...
    is necessary to multiply
%          the assigned value by a vector of ones with the same length  of ...
    t  in order
%          to have  a vector with the right dimesion  when called for the ...
    optimization.
%          Example: dx(:,i)= 0*ones(size(t,1));
%
%------------- BEGIN CODE --------------

x1=x(:,1); x2=x(:,2); x3=x(:,3);

a1=data.a1;
a2=data.a2;
a3=data.a3;
ac=data.ac;
ap=data.ap;
af=data.af;
MJtoKcal=data.MJtoKcal;
Pn=data.Pn;
Pm=data.Pm;
pFM=data.pFM;
pFFM=data.pFFM;
t_phase2=data.t_phase2;
EBwc=0;
ec=0;
fec=0;
```

```
%XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PAc = 1500;


%Initial Values

%Basal Metabolic Rate Inital Value
BMRc(1) = (0.024.*x1(1)+0.102.*x2(1)+0.85).*MJtoKcal;

%Energy Intake for Maintenance Inital Value
EIoc(1)  = (PAc + BMRc(1))./(1-0.0323);
CIc(1)   =(0.4*x3(1))/a1;
FIc(1)   = (0.3*x3(1))/a2;
PIc(1)   = (0.3*x3(1))/a3;

%Thermic Effect of Feeding Inital Value
TEFc(1) = ac.*CIc(1)+af.*FIc(1)+ap.*PIc(1);

%Energy Expenditure Inital Value
EEc(1) = BMRc(1)+PAc+TEFc(1);

%Energy Balance Inital Value
EBc(1) = x3(1)-EEc(1);

%Energy Balance Rectified
if EBc(1) > 0
    EBwc(1) = 0.1+EBc(1);
elseif EBc(1) <= 0
    if x3(1)/EEc(1) >=1
        ec(1) = 0;
    elseif x3(1)/EEc(1) <= 0.5
        ec(1) = 0.1;
    else
        ec(1) = 0.2*(1-x3(1)./EEc(1));
    end
    EBwc(1) = (BMRc(1) .* ec(1))+EBc(1);
end

%Pf present fat percentage Inital Value
Pfc(1) = 100 .* (x1(1)./(x1(1)+x2(1)));

%g(t) Intake Gain Inital Value
gc(1) = x3(1)./EIoc(1);

%fe Fat Energy Factor Inital Value
if Pfc(1) < Pm
    fec(1) = 0;
elseif Pfc(1) > Pn
    if gc(1) >= 0
```

```
        fec(1) = 0.95;
    else
        fec(1) = gc(1) + 0.9;
    end
elseif (Pm <= Pfc(1) & Pfc(1) <= Pn)
    if gc(1) >= 0
        fec(1) = -0.2375 + 0.08 .*Pfc(1);
    else
        fec(1) = -0.2375 + (gc(1)+0.9) .* (0.0833.*Pfc(1)-0.25);
    end
end;

%XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  for ite = 1:length(t)

    if ite==1
        timming(ite)=1;
    else
        timming(ite)=0;
    end
end

    timming=timming';
    Utimming2=u(:,1).*timming;

    save('controlvar2','Utimming2');

    dx(:,3)=x3+Utimming2;
    x3= dx(:,3);

%XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

 for ite = 2:length(t)

    CIc = (0.4*x3)/a1; %Based on Diet 40-30-30
    FIc = (0.3*x3)/a2;
    PIc = (0.3*x3)/a3;

    TEFc = ac.*CIc+af.*FIc+ap.*PIc;

    BMRc =(0.024.*x1+0.102.*x2+0.85).*MJtoKcal;

    EEc  = PAc + BMRc + TEFc;

    EIoc = EEc;

    %Energy Balance
    EBc = x3-EEc;
```

```
%Energy Balance - Corrected
    if EBc(ite) > 0
        EBwc(ite) = 0.1+EBc(ite);
    elseif EBc(ite) <= 0
        if x3(ite)./EEc(ite) >=1
            ec(ite) = 0;
        elseif x3(ite)./EEc(ite) <= 0.5
            ec(ite) = 0.1;
        else
            ec(ite) = 0.2*(1-x3(ite)./EEc(ite));
        end
        EBwc(ite) = (BMRc(ite) .* ec(ite))+EBc(ite);
    end

    %Pf present fat percentage
    Pfc(ite) = 100 .* (x1(ite-1)./(x1(ite-1)+x2(ite-1)));

    %g(t) Intake Gain Inital Value
    gc(ite) = x3(ite)./EIoc(ite);

    %fe Fat Energy Factor Inital Value
    if Pfc(ite) < Pm
        fec(ite) = 0;
    elseif Pfc(ite) > Pn
        if gc(ite) >= 0
            fec(ite) = 0.95;
        else
            fec(ite) = gc(ite) + 0.9;
        end
    elseif (Pm <= Pfc(ite) & Pfc(ite) <= Pn)
        if gc(ite) >= 0
            fec(ite) = -0.2375 + 0.08 .*Pfc(ite);
        else
            fec(ite) = -0.2375 + (gc(ite)+0.9) .* (0.0833.*Pfc(ite)-0.25);
        end
    end;
 end;

%XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX


num1=(fec.*EBwc)';
num2=((1-fec).*EBwc)';
den1=pFM*MJtoKcal;
den2=pFFM*MJtoKcal;

%num2/den2

dx(:,1) = (num1 /den1) + x1;
```

```
dx(:,2) = (num2 /den2) + x2;



%------------- END OF CODE --------------


function c=g(x,u,p,t,data)

% g - Returns the path constraint function where gl =< g(x,u,p,t) =< gu
% The function must be vectorized and
% xi, ui, pi are column vectors taken as x(:,i), u(:,i) and p(:,i). Each
% constraint corresponds to one column of c
%
% Syntax:  c=g(x,u,p,t,data)
%
% Inputs:
%    x  - state vector
%    u  - input
%    p  - parameter
%    t  - time
%   data- structured variable containing the values of additional data used ...
    inside
%          the function
%
% Output:
%    c - constraint function
%
%------------- BEGIN CODE --------------


PA1=1500;
a1=data.a1;
a2=data.a2;
a3=data.a3;
ac=data.ac;
ap=data.ap;
af=data.af;
MJtoKcal=data.MJtoKcal;

x1=x(:,1);
x2=x(:,2);
x3=x(:,3);

c=[x1+x2,...
   PA1-0.2.*(0.024.*x1+0.102.*x2+0.85).*MJtoKcal+ac.*((0.4/a1)*x3)+af.*((0.3/a2)*x3)+ap.*((0
   PA1+(0.024.*x1+0.102.*x2+0.85).*MJtoKcal+ac.*((0.4/a1)*x3)+af.*((0.3/a2)*x3)+ap.*((0.3/a3
```

```
%------------- END OF CODE --------------

function bc=b(x0,xf,u0,uf,p,tf,data)

% b - Returns a column vector containing the evaluation of the boundary ...
    constraints: bl =< bf(x0,xf,u0,uf,p,t0,tf) =< bu
%
% Syntax:  bc=b(x0,xf,u0,uf,p,tf,data)
%
% Inputs:
%    x0  - state at t=0
%    xf  - state at t=tf
%    u0  - input at t=0
%    uf  - input at t=tf
%    p   - parameter
%    tf  - final time
%    data- structured variable containing the values of additional data ...
    used inside
%          the function
%
%
% Output:
%    bc - column vector containing the evaluation of the boundary function
%
%------------- BEGIN CODE --------------

x1  = xf(1,:);
x2  = xf(2,:);

bc = x1+x2;

%------------- END OF CODE --------------
```

D.3 Settings

```
function options = settings_Dis_phase3

load('datadiet');

%SETTINGS - General and solver-specific settings are selected here
% Unless specified otherwise the options are set using 0 => no and 1 => yes
%
% Syntax:  options = settings_Dis
%
% Output:
%    options - Structure containing the settings
%
```

```
% Other m-files required: none
% Subfunctions: none
% MAT-files required: none
%
% Copyright (C) 2010 Paola Falugi, Eric Kerrigan and Eugene van Wyk. All ...
    Rights Reserved.
% This code is published under the BSD License.
% Department of Electrical and Electronic Engineering,
% Imperial College London London  England, UK
% ICLOCS (Imperial College London Optimal Control) 5 May 2010
% iclocs@imperial.ac.uk

%------------- BEGIN CODE --------------


% Transcription Method:
%--------------------------------------
% Discrete-time model       ('discrete')
% Multiple shooting method  ('multiple_shooting') WARNING: The
%                                                 'quasi-newton' option for ...
    the hessian
%                                                  computation has to be ...
    selected
%                                                 ...
    (options.ipopt.hessian_approximation='limited-memory').
% Euler method              ('euler')
% Trapezoidal method        ('trapezoidal')
% Hermite-Simpson method    ('hermite')


options.transcription='discrete';

% Derivative generation :
%--------------------------------------
% Whenever the analytic differentiation is enabled it is necessary to
% specify the available analytic forms for the cost function, the dynamic ...
    equations
% and  the constraints in the appropriate files .m

% Numerical differentiation: finite differences  ('numeric')
% Analytic differentiation: analytic gradients   ('analytic')

options.derivatives='numeric';


% Numeric generation of the Hessian:
%-----------------------------------------------------------------

% Whenever the numeric differentiation is enabled it is necessary to
```

```
% specify which kind of finite difference approximation to use  between
% the following ones:
%
% Central difference ('central')
% forward difference ('forward')

options.hessianFD='central';


%  The perturbation size for numerical second derivatives
%  can be set in options.perturbation.H. The perturbation size for ...
    numerical first derivatives
%  can be set in  options.perturbation.J.
%  It is possible to select default values for the perturbations by setting ...
     options.perturbation.H and
%  options.perturbation.J to the empty matrix.
%  The default values for the gradient approximation is (eps/2)^(1/3)
%  while for the  second derivative is (8*eps)^(1/3).

options.perturbation.H=[];  % Perturbation size for the second derivatives
options.perturbation.J=[];  % Perturbation size for the first derivatives



% NLP solver
%---------------------------------------
% IPOPT: recommended but needs ipopt.mex        ('ipopt')
% fmincon                                       ('fmincon')

options.NLPsolver='ipopt';

% IPOPT settings (if required)
options.ipopt.tol=1e-9;                         % Desired convergence ...
    tolerance (relative). The default value is  1e-8.
options.ipopt.print_level=5;                    % Print level. The valid ...
    range for this integer option is [0,12] and its default value is 5.
options.ipopt.max_iter=3000;                    % Maximum number of ...
    iterations. The default value is 3000.

options.ipopt.mu_strategy ='adaptive';          % Determines which barrier ...
    parameter update strategy is to be used.
                                                % The default value for this ...
                                                    string option is "monotone".
                                                % Possible values:
                                                %   'monotone': use the ...
                                                    monotone ...
                                                    (Fiacco-McCormick) strategy
                                                %   'adaptive': use the ...
                                                    adaptive update strategy
```

```
options.ipopt.hessian_approximation='exact';   %  Indicates what ...
    information for the Hessian of the Lagrangian function is
                                                %  used by the algorithm. ...
                                                    The default value is ...
                                                    'exact'.
                                                %  Possible values:
                                                %   'exact': Use second ...
                                                    derivatives provided by ...
                                                    ICLOCS.
                                                %   'limited-memory': ...
                                                    Perform a limited-memory ...
                                                    quasi-Newton approximation
                                                %    implemented inside IPOPT

%options.ipopt.limited_memory_max_history=6;   % Maximum size of the ...
    history for the limited quasi-Newton Hessian approximation. The valid ...
    range for this integer option is [0, +inf)
                                                % and its default value is 6.
%options.ipopt.limited_memory_max_skipping=1;  % Threshold for successive ...
    iterations where update is skipped for the quasi-Newton approximation.
                                                % The valid range for this ...
                                                    integer option is ...
                                                    [1,+inf) and its default ...
                                                    value is 2.


% fmincon settings (if required)
 options.fmincon=optimset;


% Automatic scaling (recommended)
%----------------------------------------
options.scaling=1;


% Output settings
%----------------------------------------

% Display computation time
options.print.time=1;

% Display relative local discretization error (recommended for direct ...
    transcription)
options.print.relative_local_error=1;


% Display cost
options.print.cost=1;
```

```
% Plot states
options.plot.states=1;

% Plot inputs
options.plot.inputs=1;

% Plot Lagrange multipliers
options.plot.multipliers=1;


% Direct transcription settings
%--------------------------------------

% Number of   integration nodes in the interval t=[0,tf]; nodes=steps+1.
% The quantity steps/N (N number of control actions) must be a positive
% integer.
options.nodes=t_phase3-t_phase2;

% Distribution of integration steps. Set tau=0 for equispaced steps.
% Otherwise: tau is a vector of length M-1 with 0<tau(i)<1 and sum(tau)=1.
% For discrete time system  set  tau=0.

options.tau=0;


% Multiple shooting settings
%--------------------------------------
% N/S=normal/stiff. H/M/L=high/medium/low accuracy
%
% 'cvodes'   N/S    H    High accuracy, difficult problems(slow)
%
%
%  Note: 'cvodes' requires the sundialsTB.
%


options.ODEsolver='cvodes';



% CVODES settings (if required)
Method='Adams';  % Method: Adams, BDF
Solver='Newton'; % Solver: Newton, Functional (requires dfdx)
```

```
% % Options for forward integration when cvodes is enabled
% %
% options.cvodes = CVodeSetOptions('RelTol',1.e-4,...
%                           'AbsTol',1.e-6,...
%                           'LinearSolver','Dense',...
%                           'MaxNumSteps',10000,...
%                           'LMM',Method,...
%                           'NonlinearSolver',Solver);
%
% % Forward sensitivity options when cvodes is enabled
%
%
%
% ...
   options.cvodesf=CVodeSensSetOptions('ErrControl',true,'method','Staggered'); ...
   % FSA initialization
%
%
%
```

Anexo E

MPC com ICLOCS - Fase 2

Neste anexo apresenta-se o código ICLOCS com MPC direccionado à perturbação de um dia específico (timeBWfailure). De dia para dia, apenas o main é alterado, sendo que o ficheiro Problem e Settings mantém-se independentemente do dia da perturbação.

E.1 MainDay2

```
% MAIN - Main script to solve a Standard MPC Problem for discrete time systems
%
% Copyright (C) 2010 Paola Falugi, Eric Kerrigan and Eugene van Wyk. All ...
    Rights Reserved.
% This code is published under the BSD License.
% Department of Electrical and Electronic Engineering,
% Imperial College London London  England, UK
% ICLOCS (Imperial College London Optimal Control) 5 May 2010
% iclocs@imperial.ac.uk

%-----------------------------------------------------------------------

clear functions;clear all
format compact

load('datadiet');
load('day1');

%Day with different Body Weight comparing to ref
timeBWfailure = 28;

x1ref_fail= zeros(t_phase2-timeBWfailure,1);
x2ref_fail= zeros(t_phase2-timeBWfailure,1);
x3ref_fail= zeros(t_phase2-timeBWfailure,1);

for ite = 1:(t_phase2-timeBWfailure+1)
```

```
    x1ref_fail(ite)=x1day1(timeBWfailure-timefailday1+ite);
    x2ref_fail(ite)=x2day1(timeBWfailure-timefailday1+ite);
    x3ref_fail(ite)=x3day1(timeBWfailure-timefailday1+ite);

end

[pb,guess]=Discrete_SysWithRef(x1ref_fail,x2ref_fail,x3ref_fail);         % ...
    Fetch the problem definition
opts=settings_Dis2(timeBWfailure); ...
                                                          % Get options and ...
    solver settings

[infoNLP,data_mpc]=transcribeOCP(pb,guess,opts);% Format for NLP solver


% ------------------------------

[sol_mpc,status] = solveNLP(infoNLP,data_mpc);  % Solve the NLP for reference

x1=sol_mpc.X(:,1);
x2=sol_mpc.X(:,2);
time=zeros(1,t_phase2);
timefail=zeros(1,t_phase2-timeBWfailure);

for ite = 1:t_phase2

    time(ite)=ite-1;

end

for ite = 1:(t_phase2-timeBWfailure+1)

    timefail(ite)=timeBWfailure+ite-1;

end

x1day2=sol_mpc.X(:,1);
x2day2=sol_mpc.X(:,2);
x3day2=sol_mpc.X(:,3);
uday2=sol_mpc.U(:,1);
timefailday2=timeBWfailure;
timeplot2=timefail(1,:);

save('day2','x1day2','x2day2','x3day2','uday2','timefailday2','timeplot2');

%-----------------------------------------
%BWfail comparing to BWref
figure(1);
```

```
plot(timefail,sol_mpc.X(:,1)+sol_mpc.X(:,2), 'k--')
hold on
plot (timefail, ...
    x1day1(timeBWfailure-timefailday1+1:t_phase2-timefailday1+1,1)+x2day1(timeBWfailure-timef
    'r-.')
title('BWphase2','FontSize',40,'Color','k')
xlabel('DAYS OF DIET','FontSize',20, 'Color','b')
ylabel('BODY WEIGHT','FontSize',20,'Color', 'b')

%EIfail comparing to EIref
figure(2);
plot(timefail,sol_mpc.X(:,3),'k--')
hold on
plot (timefail, ...
    x3day1(timeBWfailure-timefailday1+1:t_phase2-timefailday1+1,1), 'r-.')
title('EIphase2','FontSize',40,'Color','k')
xlabel('DAYS OF DIET','FontSize',20, 'Color','b')
ylabel('ENERGY INTAKE','FontSize',20, 'Color','b')

%-----------------------------------------
%EI and BW with Control
figure(3)
subplot(1,2,1) % first subplot
plot(timefail,sol_mpc.X(:,1)+sol_mpc.X(:,2), 'k-')
title('BWphase2','FontSize',40,'Color','k')
xlabel('DAYS OF DIET','FontSize',20, 'Color','b')
ylabel('BODY WEIGHT','FontSize',20,'Color', 'b')

subplot(1,2,2) % second subplot
plot (timefail,sol_mpc.X(:,3),'k')
title('EIphase2','FontSize',40,'Color','k')
xlabel('DAYS OF DIET','FontSize',20, 'Color','b')
ylabel('ENERGY INTAKE','FontSize',20, 'Color','b')

%-----------------------------------------
%EI with Control and U(t)
figure(4);
subplot(1,2,1) % first subplot
plot(timefail,sol_mpc.X(:,3),'k')
title('EIphase2 with Control','FontSize',40,'Color','k')
xlabel('DAYS OF DIET','FontSize',20, 'Color','b')
ylabel('ENERGY INTAKE','FontSize',20, 'Color','b')

subplot(1,2,2) % second subplot
plot(timefail,sol_mpc.U(:,1),'-ko',...
                'MarkerEdgeColor','k',...
                'MarkerFaceColor',[.49 1 .63],...
                'MarkerSize',10)
title('Control U','FontSize',40,'Color','k')
```

```
xlabel('DAYS OF DIET','FontSize',20, 'Color','b')
ylabel('CONTROL','FontSize',20, 'Color','b')

%Uref(t) and U(t)
figure(5);
plot(timefail,sol_mpc.U(:,1),'ko','MarkerSize',10)
hold on
plot (timefail, ...
    uday1(timeBWfailure-timefailday1+1:t_phase2-timefailday1+1,1), ...
    'rx','MarkerSize',10)
title('Uref(t) and U(t)','FontSize',40,'Color','k')
xlabel('DAYS OF DIET','FontSize',20, 'Color','b')
ylabel('DELTA EI','FontSize',20, 'Color','b')
```

E.2 Problem

```
function [problem,guess] = ...
    Discrete_SysWithRef(x1ref_fail,x2ref_fail,x3ref_fail)

%Discrete time problem - Define the optimal control problem for a discrete
%                        time system
%
% Syntax:  [problem,guess] = Discrete_Sys
%
% Outputs:
%    problem - Structure with information on the optimal control problem
%    guess   - Guess for state, control and multipliers.
%
% Other m-files required: none
% Subfunctions: L (stageCost),
%       E (boundaryCost),
%       f (ODE right-hand side),
%       g (path constraints),
%       b (boundary constraints)
% MAT-files required: termset.mat
%
% Copyright (C) 2010 Paola Falugi, Eric Kerrigan and Eugene van Wyk. All ...
    Rights Reserved.
% This code is published under the BSD License.
% Department of Electrical and Electronic Engineering,
% Imperial College London London  England, UK
% ICLOCS (Imperial College London Optimal Control) 5 May 2010
% iclocs@imperial.ac.uk

%------------- BEGIN CODE --------------

load('datadiet');
```

```
% Initial time. t0<tf.  For discrete time systems is the initial index

problem.time.t0=0;

% Final time. Let tf_min=tf_max if tf is fixed. tf_min=tf_max=[] for
% discrete time systems

problem.time.tf_min=[];
problem.time.tf_max=[];
guess.tf=[];

% Parameter bounds. pl=< p <=pu
problem.parameters.pl=[];
problem.parameters.pu=[];
guess.parameters=[];


% Initial conditions for system. Bounds if x0 is free s.t. x0l=< x0 <=x0u

problem.states.x0=[x1ref_fail(1)*1.0025 x2ref_fail(1)*1.0025 x3ref_fail(1)];

problem.states.x0l=problem.states.x0;
problem.states.x0u=problem.states.x0l;

% State bounds. xl=< x <=xu
problem.states.xl=[0.5 29 1500];
problem.states.xu=[445 200 20000];

% Terminal state bounds. xfl=< xf <=xfu
problem.states.xfl=[0.5 29 2000];
problem.states.xfu=[445 200 20000];

% Guess the state trajectories with [x0 xf]

guess.states(:,1)=[problem.states.x0(1) 0];
guess.states(:,2)=[problem.states.x0(2) 0];
guess.states(:,3)=[problem.states.x0(3) 0];

% Number of control actions N
% Set problem.inputs.N=0 if N is equal to the number of integration steps.
% Note that the number of integration steps defined in settings.m has to be ...
    divisible
% by the  number of control actions N whenever it is not zero.

problem.inputs.N=0;

% Input bounds
```

```
problem.inputs.ul=-inf;
problem.inputs.uu=inf;

% Guess the input sequences with [u0 uf]
guess.inputs(:,1)=[-500 300];


%g1=BW g2=EE g3= EE-1.2*BMR
% Bounds for path constraint function gl =< g(x,u,p,t) =< gu
problem.constraints.gl=[29 0 0];
problem.constraints.gu=[635 10000 10000];

% Bounds for boundary constraints bl =< b(x0,xf,u0,uf,p,t0,tf) =< bu

problem.constraints.bl=[BWgoal];
problem.constraints.bu=[BWgoal];

% Choose the set-points if required
problem.setpoints.states=[];
problem.setpoints.inputs=[];

% store the necessary problem parameters used in the functions

problem.data.BW=BW';
problem.data.a1=a1;
problem.data.a2=a2;
problem.data.a3=a3;
problem.data.ac=ac;
problem.data.ap=ap;
problem.data.af=af;
problem.data.MJtoKcal=MJtoKcal;
problem.data.Pn=Pn;
problem.data.Pm=Pm;
problem.data.pFM=pFM;
problem.data.pFFM=pFFM;
problem.data.t_phase2=t_phase2;
problem.data.Utimming=zeros(1,t_phase2);
problem.data.x1ref_fail=x1ref_fail;
problem.data.x2ref_fail=x2ref_fail;
problem.data.x3ref_fail=x3ref_fail;


% Get function handles and return to Main.m
problem.functions={@L,@E,@f,@g,@b};

%------------- END OF CODE --------------

function stageCost=L(x,xr,u,ur,p,t,data)
```

```
% L - Returns the stage cost.
% The function must be vectorized and
% xi, ui are column vectors taken as x(:,i) and u(:,i) (i denotes the i-th
% variable)
%
% Syntax:  stageCost = L(x,xr,u,ur,p,t,data)
%
% Inputs:
%    x  - state vector
%    xr - state reference
%    u  - input
%    ur - input reference
%    p  - parameter
%    t  - time
%    data- structured variable containing the values of additional data ...
    used inside
%          the function
%
% Output:
%    stageCost - Scalar or vectorized stage cost
%
%  Remark: If the stagecost does not depend on variables it is necessary to ...
    multiply
%          the assigned value by t in order to have right vector dimesion ...
    when called for the optimization.
%          Example: stageCost = 0*t;

%------------- BEGIN CODE --------------
x1=x(:,1);
x2=x(:,2);
x1ref_fail=data.x1ref_fail(:,1);
x2ref_fail=data.x2ref_fail(:,1);


stageCost = sum(((x1ref_fail+x2ref_fail)-(x1+x2)).^2 ,2);
%------------- END OF CODE --------------


function boundaryCost=E(x0,xf,u0,uf,p,tf,data)

% E - Returns the boundary value cost
%
% Syntax:  boundaryCost=E(x0,xf,u0,uf,p,tf,data)
%
% Inputs:
%    x0  - state at t=0
%    xf  - state at t=tf
%    u0  - input at t=0
%    uf  - input at t=tf
```

```
%    p   - parameter
%    tf  - final time
%    data- structured variable containing the values of additional data ...
    used inside
%          the function
%
% Output:
%    boundaryCost - Scalar boundary cost
%
%------------- BEGIN CODE --------------

boundaryCost=0;



%------------- END OF CODE --------------


function dx = f(x,u,p,t,data)

% f - Returns the ODE right hand side where x'= f(x,u,p,t)
% The function must be vectorized and
% xi, ui, pi are column vectors taken as x(:,i), u(:,i) and p(:,i). Each
% state corresponds to one column of dx.
%
%
% Syntax:  dx = f(x,u,p,t,data)
%
% Inputs:
%    x  - state vector
%    u  - input
%    p  - parameter
%    t  - time
%    data-structured variable containing the values of additional data used ...
    inside
%          the function
%
% Output:
%    dx - time derivative of x
%
%  Remark: If the i-th ODE right hand side does not depend on variables it ...
    is necessary to multiply
%          the assigned value by a vector of ones with the same length  of ...
    t  in order
%          to have  a vector with the right dimesion  when called for the ...
    optimization.
%          Example: dx(:,i)= 0*ones(size(t,1));
%
%------------- BEGIN CODE --------------
```

```
x1=x(:,1); x2=x(:,2); x3=x(:,3);

a1=data.a1;
a2=data.a2;
a3=data.a3;
ac=data.ac;
ap=data.ap;
af=data.af;
MJtoKcal=data.MJtoKcal;
Pn=data.Pn;
Pm=data.Pm;
pFM=data.pFM;
pFFM=data.pFFM;
t_phase2=data.t_phase2;
EBwc=0;
ec=0;
fec=0;
%XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PAc = 1500;


%Initial Values

%Basal Metabolic Rate Inital Value
BMRc(1) = (0.024.*x1(1)+0.102.*x2(1)+0.85).*MJtoKcal;

%Energy Intake for Maintenance Inital Value
EIoc(1)  = (PAc + BMRc(1))./(1-0.0323);
CIc(1)   =(0.4*x3(1))/a1;
FIc(1)   = (0.3*x3(1))/a2;
PIc(1)   = (0.3*x3(1))/a3;

%Thermic Effect of Feeding Inital Value
TEFc(1) = ac.*CIc(1)+af.*FIc(1)+ap.*PIc(1);

%Energy Expenditure Inital Value
EEc(1) = BMRc(1)+PAc+TEFc(1);

%Energy Balance Inital Value
EBc(1) = x3(1)-EEc(1);

%Energy Balance Rectified
if EBc(1) > 0
    EBwc(1) = 0.1+EBc(1);
elseif EBc(1) <= 0
    if x3(1)/EEc(1) >=1
        ec(1) = 0;
    elseif x3(1)/EEc(1) <= 0.5
```

```
        ec(1) = 0.1;
    else
        ec(1) = 0.2*(1-x3(1)./EEc(1));
    end
    EBwc(1) = (BMRc(1) .* ec(1))+EBc(1);
end

%Pf present fat percentage Inital Value
Pfc(1) = 100 .* (x1(1)./(x1(1)+x2(1)));

%g(t) Intake Gain Inital Value
gc(1) = x3(1)./EIoc(1);

%fe Fat Energy Factor Inital Value
if Pfc(1) < Pm
    fec(1) = 0;
elseif Pfc(1) > Pn
    if gc(1) >= 0
        fec(1) = 0.95;
    else
        fec(1) = gc(1) + 0.9;
    end
elseif (Pm <= Pfc(1) & Pfc(1) <= Pn)
    if gc(1) >= 0
        fec(1) = -0.2375 + 0.08 .*Pfc(1);
    else
        fec(1) = -0.2375 + (gc(1)+0.9) .* (0.0833.*Pfc(1)-0.25);
    end
end;

%XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

    dx(:,3)=x3+u(:,1);
    x3= dx(:,3);

%XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

 for ite = 2:length(t)

    CIc = (0.4*x3)/a1; %Based on Diet 40-30-30
    FIc = (0.3*x3)/a2;
    PIc = (0.3*x3)/a3;

    TEFc = ac.*CIc+af.*FIc+ap.*PIc;

    BMRc =(0.024.*x1+0.102.*x2+0.85).*MJtoKcal;

    EEc  = PAc + BMRc + TEFc;
```

```
    EIoc = EEc;

    %Energy Balance
    EBc = x3-EEc;

%Energy Balance - Corrected
    if EBc(ite) > 0
        EBwc(ite) = 0.1+EBc(ite);
    elseif EBc(ite) <= 0
        if x3(ite)./EEc(ite) >=1
            ec(ite) = 0;
        elseif x3(ite)./EEc(ite) <= 0.5
            ec(ite) = 0.1;
        else
            ec(ite) = 0.2*(1-x3(ite)./EEc(ite));
        end
        EBwc(ite) = (BMRc(ite) .* ec(ite))+EBc(ite);
    end

    %Pf present fat percentage
    Pfc(ite) = 100 .* (x1(ite-1)./(x1(ite-1)+x2(ite-1)));

    %g(t) Intake Gain Inital Value
    gc(ite) = x3(ite)./EIoc(ite);

    %fe Fat Energy Factor Inital Value
    if Pfc(ite) < Pm
        fec(ite) = 0;
    elseif Pfc(ite) > Pn
        if gc(ite) >= 0
            fec(ite) = 0.95;
        else
            fec(ite) = gc(ite) + 0.9;
        end
    elseif (Pm <= Pfc(ite) & Pfc(ite) <= Pn)
        if gc(ite) >= 0
            fec(ite) = -0.2375 + 0.08 .*Pfc(ite);
        else
            fec(ite) = -0.2375 + (gc(ite)+0.9) .* (0.0833.*Pfc(ite)-0.25);
        end
    end;
 end;

%XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX


num1=(fec.*EBwc)';
num2=((1-fec).*EBwc)';
den1=pFM*MJtoKcal;
```

```
den2=pFFM*MJtoKcal;

dx(:,1) = (num1 /den1) + x1;
dx(:,2) = (num2 /den2) + x2;



%------------- END OF CODE --------------


function c=g(x,u,p,t,data)

% g - Returns the path constraint function where gl =< g(x,u,p,t) =< gu
% The function must be vectorized and
% xi, ui, pi are column vectors taken as x(:,i), u(:,i) and p(:,i). Each
% constraint corresponds to one column of c
%
% Syntax:  c=g(x,u,p,t,data)
%
% Inputs:
%    x  - state vector
%    u  - input
%    p  - parameter
%    t  - time
%   data- structured variable containing the values of additional data used ...
    inside
%          the function
%
% Output:
%    c - constraint function
%
%------------- BEGIN CODE --------------


PA1=1500;
a1=data.a1;
a2=data.a2;
a3=data.a3;
ac=data.ac;
ap=data.ap;
af=data.af;
MJtoKcal=data.MJtoKcal;

x1=x(:,1);
x2=x(:,2);
x3=x(:,3);

c=[x1+x2,...
   PA1-0.2.*(0.024.*x1+0.102.*x2+0.85).*MJtoKcal+ac.*((0.4/a1)*x3)+af.*((0.3/a2)*x3)+ap.*((0.3/a3)
```

```
   PA1+(0.024.*x1+0.102.*x2+0.85).*MJtoKcal+ac.*((0.4/a1)*x3)+af.*((0.3/a2)*x3)+ap.*((0.3/a3
```

```
%------------- END OF CODE --------------

function bc=b(x0,xf,u0,uf,p,tf,data)

% b - Returns a column vector containing the evaluation of the boundary ...
    constraints: bl =< bf(x0,xf,u0,uf,p,t0,tf) =< bu
%
% Syntax:  bc=b(x0,xf,u0,uf,p,tf,data)
%
% Inputs:
%    x0  - state at t=0
%    xf  - state at t=tf
%    u0  - input at t=0
%    uf  - input at t=tf
%    p   - parameter
%    tf  - final time
%    data- structured variable containing the values of additional data ...
    used inside
%          the function
%
%
% Output:
%    bc - column vector containing the evaluation of the boundary function
%
%------------- BEGIN CODE --------------

xf1=xf(1,:);
xf2=xf(2,:);
bc = [xf1+xf2];

%------------- END OF CODE --------------
```

E.3 Settings

```
function options = settings_Dis2(timeBWfail)

load('datadiet');

%SETTINGS - General and solver-specific settings are selected here
% Unless specified otherwise the options are set using 0 => no and 1 => yes
%
% Syntax:  options = settings_Dis
%
% Output:
```

```
%    options - Structure containing the settings
%
% Other m-files required: none
% Subfunctions: none
% MAT-files required: none
%
% Copyright (C) 2010 Paola Falugi, Eric Kerrigan and Eugene van Wyk. All ...
    Rights Reserved.
% This code is published under the BSD License.
% Department of Electrical and Electronic Engineering,
% Imperial College London London  England, UK
% ICLOCS (Imperial College London Optimal Control) 5 May 2010
% iclocs@imperial.ac.uk

%------------- BEGIN CODE --------------


% Transcription Method:
%---------------------------------------
% Discrete-time model       ('discrete')
% Multiple shooting method  ('multiple_shooting') WARNING: The
%                                                 'quasi-newton' option for ...
    the hessian
%                                                  computation has to be ...
    selected
%                                                 ...
    (options.ipopt.hessian_approximation='limited-memory').
% Euler method              ('euler')
% Trapezoidal method        ('trapezoidal')
% Hermite-Simpson method    ('hermite')


options.transcription='discrete';

% Derivative generation :
%---------------------------------------
% Whenever the analytic differentiation is enabled it is necessary to
% specify the available analytic forms for the cost function, the dynamic ...
    equations
% and  the constraints in the appropriate files .m

% Numerical differentiation: finite differences  ('numeric')
% Analytic differentiation: analytic gradients   ('analytic')

options.derivatives='numeric';


% Numeric generation of the Hessian:
%-----------------------------------------------------------------
```

```
% Whenever the numeric differentiation is enabled it is necessary to
% specify which kind of finite difference approximation to use  between
% the following ones:
%
% Central difference ('central')
% forward difference ('forward')

options.hessianFD='central';


%  The perturbation size for numerical second derivatives
%  can be set in options.perturbation.H. The perturbation size for ...
    numerical first derivatives
%  can be set in  options.perturbation.J.
%  It is possible to select default values for the perturbations by setting ...
     options.perturbation.H and
%  options.perturbation.J to the empty matrix.
%  The default values for the gradient approximation is (eps/2)^(1/3)
%  while for the  second derivative is (8*eps)^(1/3).

options.perturbation.H=[];  % Perturbation size for the second derivatives
options.perturbation.J=[];  % Perturbation size for the first derivatives



% NLP solver
%---------------------------------------
% IPOPT: recommended but needs ipopt.mex        ('ipopt')
% fmincon                                       ('fmincon')

options.NLPsolver='ipopt';

% IPOPT settings (if required)
options.ipopt.tol=1e-9;                         % Desired convergence ...
    tolerance (relative). The default value is  1e-8.
options.ipopt.print_level=5;                    % Print level. The valid ...
    range for this integer option is [0,12] and its default value is 5.
options.ipopt.max_iter=3000;                    % Maximum number of ...
    iterations. The default value is 3000.

options.ipopt.mu_strategy ='adaptive';          % Determines which barrier ...
    parameter update strategy is to be used.
                                                % The default value for this ...
                                                    string option is "monotone".
                                                % Possible values:
                                                %   'monotone': use the ...
                                                    monotone ...
                                                    (Fiacco-McCormick) strategy
```

```
                                                %   'adaptive': use the ...
                                                    adaptive update strategy

options.ipopt.hessian_approximation='exact';    %  Indicates what ...
    information for the Hessian of the Lagrangian function is
                                                %  used by the algorithm. ...
                                                    The default value is ...
                                                    'exact'.
                                                %  Possible values:
                                                %   'exact': Use second ...
                                                    derivatives provided by ...
                                                    ICLOCS.
                                                %   'limited-memory': ...
                                                    Perform a limited-memory ...
                                                    quasi-Newton approximation
                                                %    implemented inside IPOPT

%options.ipopt.limited_memory_max_history=6;   % Maximum size of the ...
    history for the limited quasi-Newton Hessian approximation. The valid ...
    range for this integer option is [0, +inf)
                                                % and its default value is 6.
%options.ipopt.limited_memory_max_skipping=1;  % Threshold for successive ...
    iterations where update is skipped for the quasi-Newton approximation.
                                                % The valid range for this ...
                                                    integer option is ...
                                                    [1,+inf) and its default ...
                                                    value is 2.


% fmincon settings (if required)
 options.fmincon=optimset;


% Automatic scaling (recommended)
%-----------------------------------------
options.scaling=1;


% Output settings
%-----------------------------------------

% Display computation time
options.print.time=1;

% Display relative local discretization error (recommended for direct ...
    transcription)
options.print.relative_local_error=1;
```

```
% Display cost
options.print.cost=1;


% Plot states
options.plot.states=1;

% Plot inputs
options.plot.inputs=1;

% Plot Lagrange multipliers
options.plot.multipliers=1;


% Direct transcription settings
%--------------------------------------

% Number of   integration nodes in the interval t=[0,tf]; nodes=steps+1.
% The quantity steps/N (N number of control actions) must be a positive
% integer.
options.nodes=t_phase2-timeBWfail+1;

% Distribution of integration steps. Set tau=0 for equispaced steps.
% Otherwise: tau is a vector of length M-1 with 0<tau(i)<1 and sum(tau)=1.
% For discrete time system  set  tau=0.

options.tau=0;


% Multiple shooting settings
%--------------------------------------
% N/S=normal/stiff. H/M/L=high/medium/low accuracy
%
% 'cvodes'   N/S    H    High accuracy, difficult problems(slow)
%
%
%  Note: 'cvodes' requires the sundialsTB.
%


options.ODEsolver='cvodes';



% CVODES settings (if required)
Method='Adams';  % Method: Adams, BDF
Solver='Newton'; % Solver: Newton, Functional (requires dfdx)
```

```
% % Options for forward integration when cvodes is enabled
% %
% options.cvodes = CVodeSetOptions('RelTol',1.e-4,...
%                            'AbsTol',1.e-6,...
%                            'LinearSolver','Dense',...
%                            'MaxNumSteps',10000,...
%                            'LMM',Method,...
%                            'NonlinearSolver',Solver);
%
% % Forward sensitivity options when cvodes is enabled
%
%
%
% ...
    options.cvodesf=CVodeSensSetOptions('ErrControl',true,'method','Staggered'); ...
    % FSA initialization
%
%
%
```

Referências

[1] Fitness Trainer Archives - Power Yoga Studio in Islamabad. URL: http://www.yogaislamabad.com/tag/fitness-trainer/.

[2] Fitness testing and training. URL: http://lalvesbtg.weebly.com/about.html.

[3] Let the Bod Pod tell you what you're made of - University News. URL: http://info.umkc.edu/unews/let-the-bod-pod-tell-you-what-youre-made-of/.

[4] The Wind In My Face - DEXA Body Scan for Fat, Muscle, Bone Measurement - Bone density. URL: http://windinmyface.com/dexa-bone.html.

[5] FEUP inova na medição de gordura corporal - JPN - JornalismoPortoNet. URL: http://jpn.up.pt/2008/05/19/feup-inova-na-medicao-de-gordura-corporal/.

[6] Medidor grasa corporal Omron. URL: http://itmadrid.net/page/news/medidor-grasa-corporal-omron.

[7] S B Heymsfield, D Thomas, a M Nguyen, J Z Peng, C Martin, W Shen, B Strauss, a Bosy-Westphal, e M J Muller. Voluntary weight loss: systematic review of early phase body composition changes. *Obesity reviews : an official journal of the International Association for the Study of Obesity*, 2011.

[8] SoftControl Oy - Tuotteet. URL: http://www.kolumbus.fi/softcontrol/tuotteet.htm.

[9] Daniel E Rivera e J.-Emeterio Navarro. A Dynamical Systems Model for Weight Change Behavioral Interventions. *CSEL Technical Progress Report*, 2010.

[10] Kevin D. Hall. Predicting metabolic adaptation, body weight change, and energy intake in humans. *American journal of physiology. Endocrinology and metabolism*, 298:E449–E466, 2010.

[11] Richard Dobbs, Corinne Sawers, Fraser Thompson, James Manyika, Jonathan Woetzel, Peter Child, Sorcha McKenna, e Angela Spatharou. Overcoming obesity : An initial economic analysis Discussion paper. Relatório técnico November, McKinsey Global Institute, 2014.

[12] Nonna Viernes, Ziad A J Zaidan, Atsu S S Dorvlo, Mami Kayano, Kazuhiro Yoishiuchi, Hiroaki Kumano, Tomifusa Kuboki, e Samir Al-Adawi. Tendency toward deliberate food restriction, fear of fatness and somatic attribution in cross-cultural samples. *Eating behaviors*, 8(3):407–17, Agosto 2007. URL: http://www.sciencedirect.com/science/article/pii/S1471015306001115, doi:10.1016/j.eatbeh.2006.12.003.

[13] Basal Metabolic Rate., 1957. URL: http://en.wikipedia.org/wiki/Basal_metabolic_rate.

[14] Hall KD, Sacks G, Chandramohan D, et al. Appendix - Hall Lancet - Web Appendix. URL: http://www.niddk.nih.gov/research-funding/at-niddk/labs-branches/LBM/integrative-physiology-section/body-weight-simulator/Documents/Hall_Lancet_Web_Appendix.pdf.

[15] R R Wing e J O Hill. Successful weight loss maintenance. *Annual review of nutrition*, 2001.

[16] B E Ainsworth, W L Haskell, M C Whitt, M L Irwin, a M Swartz, S J Strath, W L O'Brien, D R Bassett, K H Schmitz, P O Emplaincourt, D R Jacobs, e a S Leon. Compendium of physical activities: an update of activity codes and MET intensities., 2000. URL: http://www.juststand.org/Portals/3/literature/compendium-of-physical-activities.pdf.

[17] Katch-Mcardle BMR Calculator. URL: http://www.calculatorpro.com/calculator/katch-mcardle-bmr-calculator/.

[18] University of Texas. Body Composition. URL: http://www.uta.edu/faculty/beckham/BodyComposition.pdf.

[19] Model My Diet | Virtual Weight Loss Simulator and Motivation Tool. URL: http://modelmydiet.com/women_blog.html?utm_expid=62054907-4.oRDvKT9aTDSmTaAeXHHPTg.1.

[20] Calories Per Day Calculator - How Many Calories Do You Need? URL: http://www.shapefit.com/calculators/calories-per-day-calculator.html.

[21] Calories Burned During Exercise. URL: http://www.nutristrategy.com/activitylist4.htm.

[22] Lose It! - Challenge yourself. URL: https://www.loseit.com/how-it-works/.

[23] Water Chemistry Facts and Information. URL: http://chemistry.about.com/od/waterchemistry/f/How-Much-Of-Your-Body-Is-Water.htm.

[24] Sleep | Obesity Prevention Source | Harvard T.H. Chan School of Public Health. URL: http://www.hsph.harvard.edu/obesity-prevention-source/obesity-causes/sleep-and-obesity/.

[25] Hawthorne effect. 2015. URL: http://en.wikipedia.org/wiki/Hawthorne_effect.

[26] Body Weight Simulator | National Institute of Diabetes and Digestive and Kidney Diseases (NIDDK). URL: http://www.niddk.nih.gov/research-funding/at-niddk/labs-branches/LBM/integrative-physiology-section/body-weight-simulator/Pages/body-weight-simulator.aspx.

[27] Kevin D Hall, Gary Sacks, Dhruva Chandramohan, Carson C Chow, Claire Wang, Steven L Gortmaker, e Boyd a Swinburn. Quantification of the effect of energy imbalance on bodyweight Kevin. 378(9793), 2014.

[28] A S Jackson, P R Stanforth, J Gagnon, T Rankinen, A S Leon, D C Rao, J S Skinner, C Bouchard, e J H Wilmore. The effect of sex, age and race on estimating percentage body fat from body mass index: The Heritage Family Study. *International journal of obesity and related metabolic disorders : journal of the International Association for the Study of Obesity*, 26(6):789–96, 2002. URL: http://www.nature.com/ijo/journal/v26/n6/full/0802006a.html, doi:10.1038/sj.ijo.0802006.

[29] Matlab. URL: http://pt.wikipedia.org/wiki/MATLAB.

[30] Block Libraries - MATLAB & Simulink. URL: http://www.mathworks.com/help/simulink/block-libraries.html.

[31] P Falugi, E C Kerrigan, e E Van˜Wyk. {Imperial College London Optimal Control Software} ({ICLOCS}) {@ONLINE}, 2010. URL: http://www.ee.ic.ac.uk/ICLOCS/.

[32] Andreas Wächter. Short tutorial: Getting started with ipopt in 90 minutes. Em Uwe Naumann, Olaf Schenk, Horst D. Simon, e Sivan Toledo, editores, *Combinatorial Scientific Computing*, número 09061 em Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2009. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany. URL: http://drops.dagstuhl.de/opus/volltexte/2009/2089.

[33] A. Bemporad. Model Predictive Control : Basic Concepts Model Predictive Control (MPC). *Control Systems Engineering, Imperial College London*, 2009.

[34] MPC - ControlsWiki. URL: https://controls.engin.umich.edu/wiki/index.php/MPC.

[35] Shirley Gerrior, Wenyen Juan, e Peter Basiotis. An easy approach to calculating estimated energy requirements. *Preventing chronic disease*, 3, 2006.

[36] C Forbes-Ewan. Australian Defence Force Nutritional Requirements in the 21st Century (Version 1). *Defence Science and Technology Organisation Victoria (Australia) Human Protection and Performance Div*, 2009. URL: papers2://publication/uuid/A161E096-50E1-482D-AFBF-300D430AC9B2.

[37] Zone Diet. *The Cross Fit Journal*, 2004. URL: http://library.crossfit.com/free/pdf/cfjissue21_May04.pdf.

[38] Paola Falugi, Eric Kerrigan, e Eugene Van Wyk. User Guide - Matlab Toolbox IPOPT. 2010.

[39] Lizzie Velasquez Disease. URL: http://www.telegraph.co.uk/news/health/news/7858664/The-girl-who-must-eat-every-15-minutes-to-stay-alive.html.

[40] *Human Body/Extreme Bodies/Heaviest Man*. Guinness World Records. URL: http://www.guinnessworldrecords.com/content_pages/record.asp?recordid=48383.

[41] Health Information and Medical Information. URL: http://www.health.harvard.edu/.

[42] MathWorks. Designing a Model Predictive Controller in Simulink® - MATLAB & Simulink Example. URL: http://www.mathworks.com/help/mpc/examples/designing-a-model-predictive-controller-in-simulink.html.

[43] Washington.edu. 10 Reasons Python Rocks for Research (And a Few Reasons it Doesn't) — Hoyt Koepke, 2010. URL: http://www.stat.washington.edu/~hoytak/blog/whypython.html.