Modelagem e Otimização de Marcha Dinâmica para Robôs Humanoides

Relatório Final

Candidato: Feliphe Gonçalves Galiza
Orientador: Marko Ackermann
Departamento: Engenharia Mecânica
N° FEI: 11.210.357-7
Data: 27/10/2015

1) Resumo

O presente trabalho consiste em um projeto de geração de trajetórias de marcha dinâmica para robôs humanoides. A metodologia adotada fundamenta-se em desenvolver a modelagem das equações de movimento do sistema, aplicar conceitos como o ZMP (Zero Moment Point) [9] e CoP (Center of Pressure) [10] para determinação de critérios de estabilidade dinâmica, e implementar a otimização de trajetórias através de algoritmos provenientes da Teoria de Controle Ótimo. Robôs utilizados pela equipe RoboFEI [12] serão utilizados como plataforma de validação do método.

Palavras Chave: Robôs Humanóides; Zero Moment Point; Locomoção Bípede; Controle Ótimo; Robocup; Humanoid League.

1.2) Objetivo

Nesta iniciação científica, tem-se como objetivo o desenvolvimento de trajetórias para locomoção bípede através de técnicas que utilizam critérios de estabilidade e otimização de parâmetros, dando continuidade a trabalhos prévios e permitindo a participação do Centro Universitário da FEI em campeonatos mundiais de futebol de robôs, como a *RoboCup* [15].

2) Revisão Bibliográfica

2.1) O Robô Humanoide do Centro Universitário da FEI

O projeto de um robô humanoide capaz de competir na Humanoid-League KidSize [14] pelo Centro Universitário da FEI, iniciou-se no ano de 2010 com a Iniciação Científica “Projeto Mecânico de um Robô Humanoide” desenvolvida pelo aluno Milton Cortez [11]. Naquela ocasião, foi desenvolvido um robô humanoide com 22 graus de liberdade, distribuídos da seguinte forma: seis em cada uma das pernas, três em cada braço, dois no tronco e dois no pescoço, conforme representado esquematicamente na Fig.1 [7].

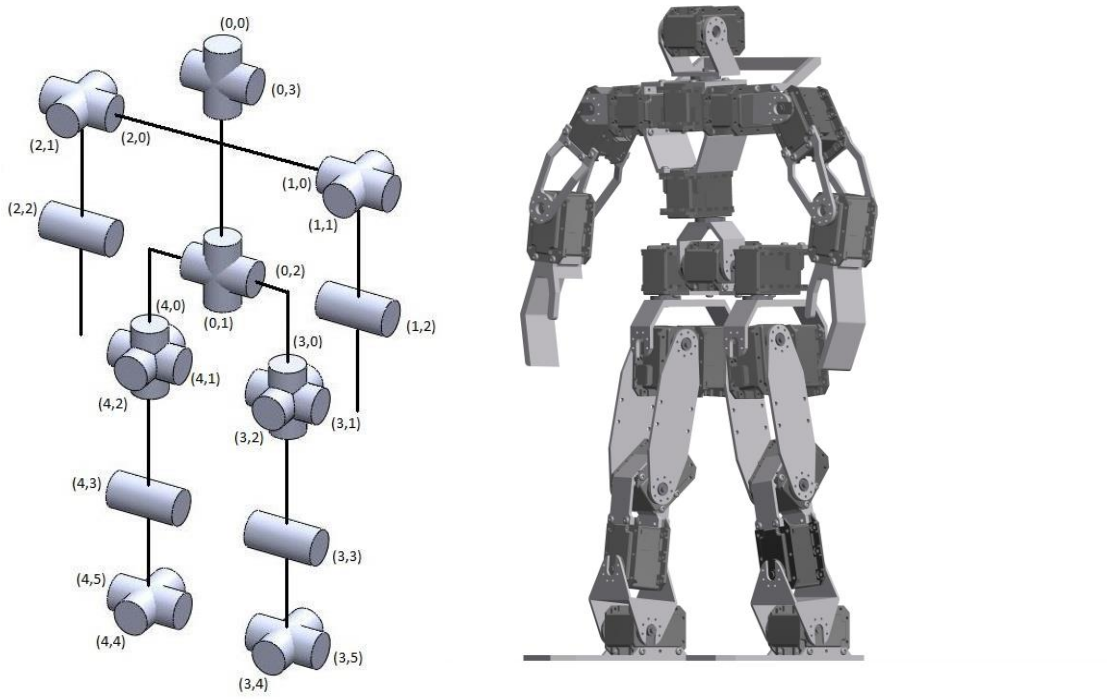


Fig. 1 – Representação esquemática dos graus de liberdade e numeração das juntas.

A estrutura mecânica do robô humanoide é composta por peças de alumínio aeronáutico de série (7XXX) e por 22 servo motores. A união dos componentes dá origem ao robô humanoide e determina a posição do centro de massa do robô (Fig. 2).

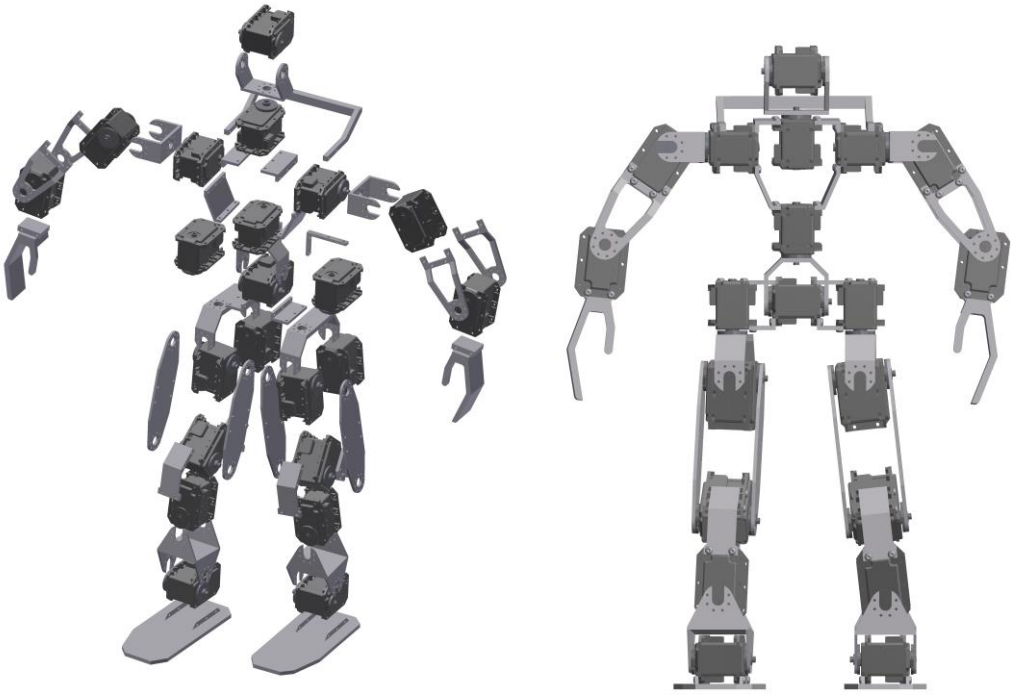


Fig. 2 – Estrutura Mecânica do Robô Humanoide

2.2) Darwin-OP

O Darwin-OP (Figs. 3 e 4) é um robô humanoide desenvolvido e fabricado pela empresa Coreana ROBOTIS [16] em colaboração com as universidades Virginia Tech, Purdue University, e a University of Pennsylvania, com o propósito principal de servir pesquisadores e programadores interessados em robôs humanoides, inteligência artificial, algoritmos de marcha, visão computacional, etc.. O robô possui vinte graus de liberdade.

Fig. 3 – Robô Darwin-OP

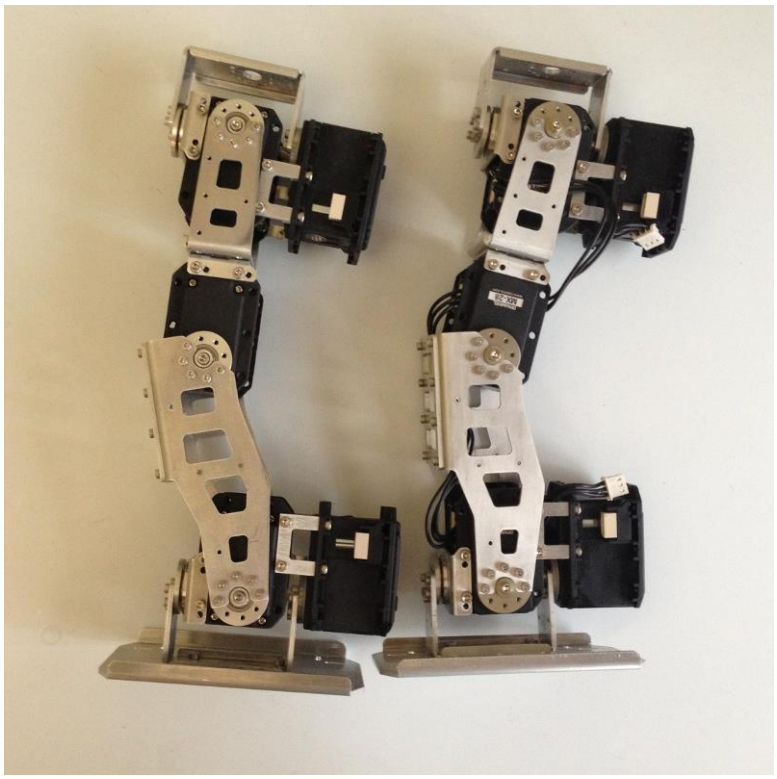


Fig. 4 – Estrutura mecânica da perna do robô Darwin-OP

2.3) Marcha Estática

A locomoção de um robô humanoide é definida como marcha estática quando a projeção do centro de massa do robô no solo está contida dentro do polígono de suporte dos pés do robô. O Polígono de Suporte é delimitado pela posição das arestas dos pés apoiados no solo. Podemos verificar este conceito através da Fig. 5. Esta condição proporciona apenas equilíbrio estático, de forma que a estabilidade só pode ser garantida a baixa velocidades de movimentação [8].

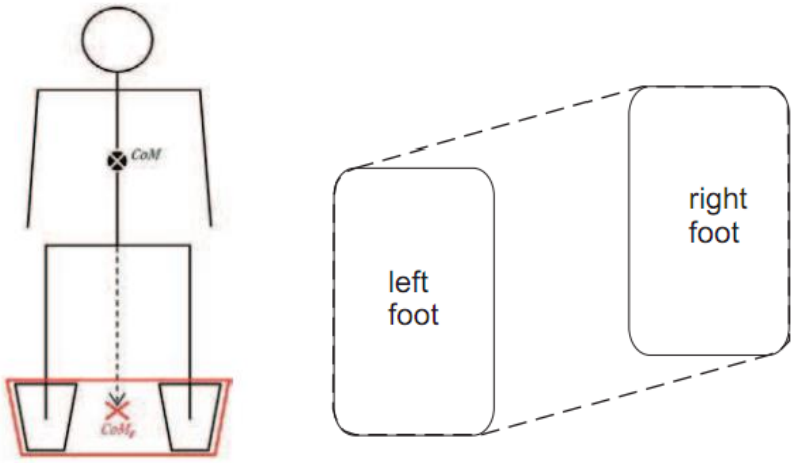


Fig. 5 – Projeção do Centro de Massa e Polígono de Suporte dos pés.

2.4) Marcha Dinâmica

Existem diversas abordagens e métodos acerca do equilíbrio dinâmico em movimentação bípede, contudo a grande maioria é baseada nos conceitos do ZMP (Zero Moment Point) [9] e CoP (Center of Pressure) [10]. Estas condições garantem estabilidade do robô bípede também em altas velocidades. A seguir descreve-se cada um desses conceitos.

2.4.1) Zero Moment Point (ZMP)

Em robôs humanoides todas as juntas do mecanismo são motorizadas e podem ser diretamente controladas, com exceção do contato entre o chão e o pé do robô, que se pode considerar como um grau de liberdade adicional não atuado a partir do momento em que existe movimento relativo entre as duas superfícies, ou a planta do pé do robô não está totalmente em contato com o solo. O contato do pé com o solo tem extrema

importância na movimentação pois exerce influência sobre o tipo de estabilidade da caminhada (estabilidade estática ou dinâmica) [9]. Nas condições em que apenas uma das arestas da planta do pé está em contato com o solo, emerge um grau de liberdade adicional não atuado. Pelo fato de não ser motorizado, esse grau de liberdade não pode ser controlado diretamente como os outros. Em decorrência disso, surgem dificuldades no controle e estabilidade do robô. Dessa forma, o indicador geral do comportamento do mecanismo em termos de estabilidade é o contato completo da base do pé com o solo.

O ponto no qual age uma única força equivalente a (ou que substitui) todas as forças atuantes sobre o sistema é denominado Zero Moment Point, que doravante trataremos por ZMP. A teoria relativa a este método foi proposta por Vukobratovic [9] há aproximadamente quarenta anos. A permanência do ZMP no polígono de suporte dos pés (Fig. 5) garante que as bases dos pés permanecerão em contato com o solo e o sistema estará em equilíbrio dinâmico. Por muito tempo esta abordagem foi utilizada como o único procedimento na síntese da movimentação bípede e ainda hoje é largamente utilizada em conjunto com outras teorias.

2.4.2) Equacionamento do ZMP

O ZMP é definido como o ponto sobre o solo onde a resultante dos momentos de todas as forças externas, inerciais e gravitacionais não possui componente ao longo do eixo horizontal. Em outras palavras, é o ponto onde $\mathbf{M_x = 0}$ e $\mathbf{M_y = 0}$ representam os momentos em torno dos eixos ortogonais x e y, respectivamente, que definem o plano do solo. Neste ponto atuam apenas a força de reação $\mathbf{F_p}$ e momento de reação $\mathbf{T_p}$ na direção normal ao solo. O ZMP está localizado em:

$$\mathbf{P_{ZMP}} = (x_{ZMP}, y_{ZMP}, 0) \in S \qquad (1)$$

onde P_{ZMP} é o ponto dado pelas coordenadas x_{ZMP} e y_{ZMP} e S é a região que delimita a superfície de suporte sob os pés. Enquanto o P_{ZMP} encontra-se dentro da região S, o contato entre o solo e o pé é estável [9]. Em outras palavras, como podemos ver na Figura 6, as forças e momentos que atuam no sistema podem ser substituídas pelas resultantes de Força Fa e Momento Ma, ambos num ponto qualquer A, onde em geral há componente horizontal do momento Ma. No ponto particular P, que representa a posição do ZMP na região de contato do pé com o solo temos ação da força de reação

Fp e do Momento de reação Mp com componente apenas normal ao plano do solo, direção z [8].

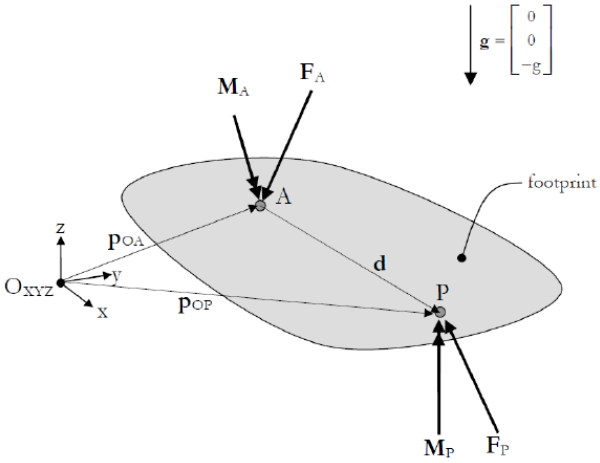


Fig. 6 – Zero Moment Point [8].

2.4.3) Centro de Pressão

Considerando-se o robô apoiado apenas em uma das pernas, existe um carregamento distribuído (de reação do solo) agindo sob o pé. Esse carregamento pode ser representado por uma única força resultante aplicada em um ponto em relação ao qual a resultante dos momentos na direção tangente ao solo é nula [10], Fig. 7. Este ponto denomina-se Centro de Pressão, abreviadamente tratado por CoP (devido ao termo em inglês Center of Pressure). Esse conceito é bastante parecido com o de ZMP, entretanto, este último está relacionado às forças transmitidas sem contato (gravidade, inércia), enquanto que o conceito de CoP está atrelado às forças transmitidas por contato e, por isso, ao contrário do ZMP, não pode existir fora da área do polígono de suporte S (Fig. 8) [9].

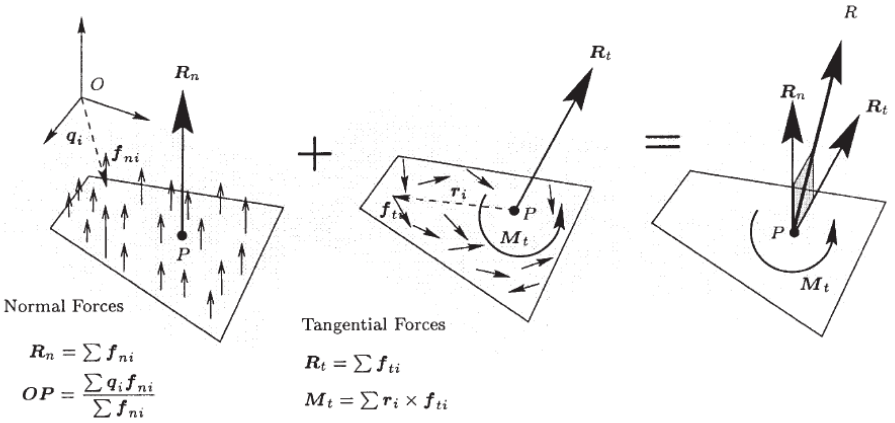


Fig. 7 – CoP (Center of Pressure)

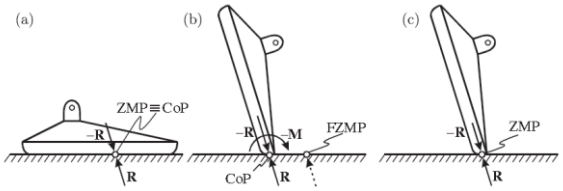


Fig. 8 – Vínculo entre pé e solo / posição do ZMP [9].

(a) Pé totalmente apoiado, (b) pé inclinado e ZMP fora do vértice, (c) pé inclinado e ZMP no vértice.

3) Complexidade associada à modelagem de uma Marcha Dinâmica

A locomoção bípede é reconhecida como um dos problemas fundamentais não-resolvidos em Robótica. Os desafios para a conclusão desta tarefa vão desde o projeto mecânico do robô até outros fatores como a formação de cadeias cinemáticas fechadas e o aparecimento de sistemas subatuados que fazem com que exista uma variação do número de graus de liberdade em função da configuração do sistema ao longo do tempo, implicando em limitações de modelagem e controle do sistema [5].

3.1) Definição de Sistemas Subatuados

A dinâmica de um sistema multicorpo formado por corpos rígidos é descrita pelas equações de movimento, através da formulação de Euler-Lagrange**, pela equação**

$$L(q, \dot{q}) = T(q, \dot{q}) - U(q) \tag{2}$$

Onde o Lagrangiano é a diferença entre a energia cinética e potencial. Para um sistema multicorpo, as equações de Euler-Lagrange são descritas como:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_k}\right) - \frac{\partial L}{\partial q_k} = (B(q) \cdot u)_k \tag{3}$$

Para $k = \{1,2, \ldots., n\}$

onde q é o vetor de coordenadas generalizadas, $\dot{q}$ é o vetor de velocidades generalizadas e u é o vetor de controles (forças e torques). $B(q)$ é a matriz das forças de entrada, com $B(q) \cdot \tau$ descrevendo as forças generalizadas resultantes das entradas de controle τ.

3.1.1) Sistema Completamente Atuado

Um sistema de controle descrito pela equação (3) é classificado como completamente atuado na configuração $(q, \dot{q}, t)$, se for possível comandar uma aceleração instantânea em uma direção arbitrária de q, i.e. se

$$rank[B(q)] = \dim(q). \tag{4}$$

3.1.2) Sistemas Subatuados

Um sistema de controle descrito pela equação 3 é classificado como subatuado na configuração $(q, \dot{q}, t)$, se não for possível comandar uma aceleração instantânea em uma direção arbitrária de q, i.e., se:

$$rank[B(q)] < \dim(q). \tag{5}$$

Em resumo, sistemas subatuados são aqueles para os quais as entradas de controle não conseguem acelerar o estado do sistema em direções arbitrárias. Como consequência, diferente de sistemas completamente atuados, sistemas subatuados não possuem a propriedade de seguirem trajetórias arbitrárias apenas com a ação das entradas de controle [5].

Contudo, para robôs com pés compostos por bases de apoio, dentro de certos limites, como o ZMP dentro da base de apoio, é possível acelerar o estado do sistema em direções arbitrárias.

3.2) Cadeias Cinemáticas e Sistemas Sobre-Atuados

Uma cadeia cinemática é constituída por um conjunto de elos conectados em série por juntas. A maneira como esses elos são conectados, implica em restrições cinemáticas que influenciam a modelagem dinâmica do sistema [7].

3.2.1) Cadeia Cinemática Aberta

Uma cadeia cinemática aberta ocorre quando o sistema se encontra em uma configuração onde em um de seus últimos elos não existe nenhum tipo de restrição cinemática aplicada, por exemplo uma garra vazia, mão, cabeça, etc. [7].

3.2.2) Cadeia Cinemática Fechada

Uma cadeia cinemática fechada ocorre quando uma cadeia cinemática “fecha” nela mesma, formando assim um circuito fechado. Neste caso, o número de graus de liberdade é reduzido, esse fenômeno pode ser observado na Fig. 10 onde um sistema com três graus de liberdade em cadeia cinemática aberta transforma-se em um sistema de apenas um grau de liberdade em cadeia cinemática fechada [7].

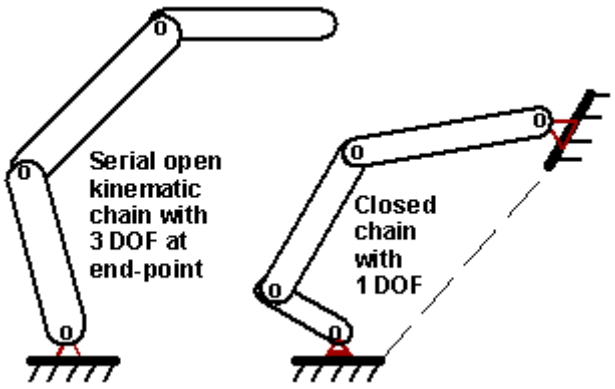


Fig. 9 - a) Cadeia Cinemática Aberta. b) Cadeia Cinemática Fechada.

4) Ciclo de Marcha para Robôs Bípedes

A configuração cinemática de robôs bípedes altera-se entre fase de apoio simples e fase de apoio duplo durante o contato do pé com o solo

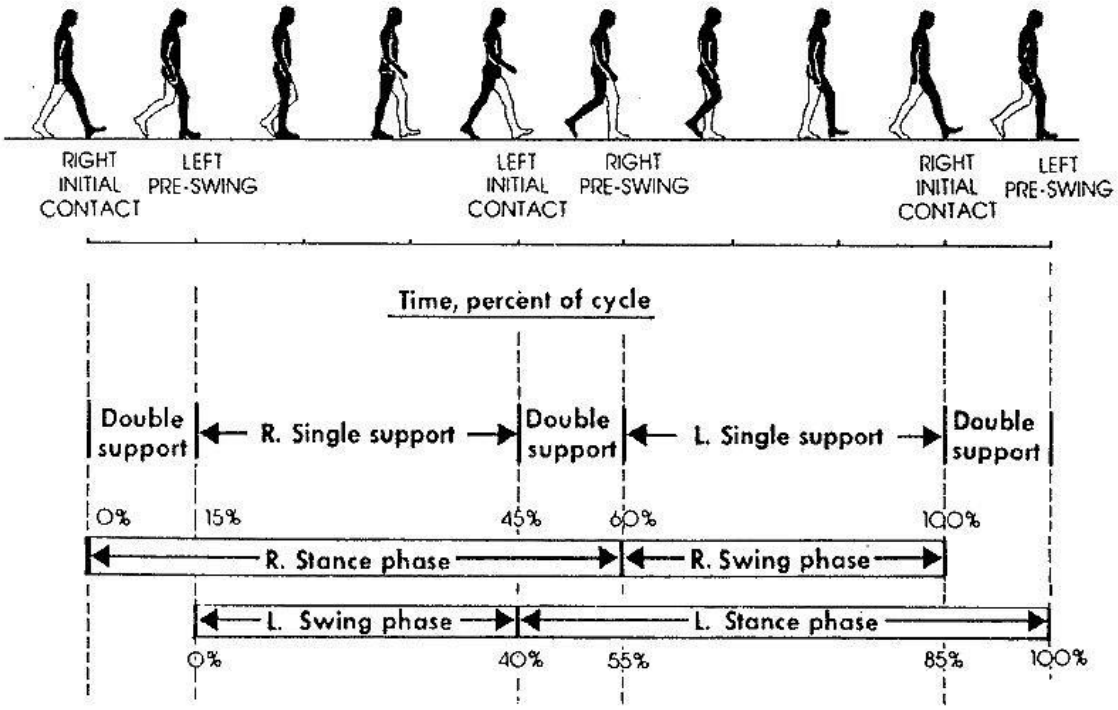


Fig. 10 – Transição entre a fase de apoio simples e apoio duplo.[7]

4.1) Fase de apoio duplo (Double Support Phase)

A fase de apoio duplo ocorre quando os dois pés estão em contato com o solo. Nesta fase, o sistema movimenta-se com cadeia cinemática fechada, caracterizando assim um sistema sobre-atuado, ou seja, cujo número de torque atuantes é maior que o número de graus de liberdade, isso implica em problemas de redundância na solução de problemas de Dinâmica Inversa e aplicações de Controle.

4.2) Fase de apoio simples (Single Support Phase)

A fase de apoio simples ocorre quando apenas um dos pés está em contato com o solo. Nesta fase o sistema movimenta-se com cadeia cinemática aberta. Caso fosse adicionado uma reação de engastamento entre o pé do robô bípede e o solo, o sistema tornar-se-ia completamente atuado.

5) Formulação de um problema de Controle Ótimo

Um problema de controle ótimo consiste em determinar a série temporal dos controles e dos estados de um sistema dinâmico de forma a minimizar um determinado critério de otimização[17].

A formulação de um problema de controle ótimo requer os seguintes parâmetros:

- Modelo matemático do sistema e das equações de movimento correspondentes
- Especificação do critério de otimização
- Especificação de todas as condições de contorno a serem satisfeitas relacionadas com os estados e entradas de controle do sistema
- Especificação de parâmetros livres.

O objetivo de um problema de controle ótimo não-lineares é minimizar o seguinte funcional:

$$J = \int_{t}^{t_f} L(x,u) \cdot dt \tag{6}$$

onde t é o tempo corrente e t_f é o tempo final,sujeito a restrições dadas pelas equações dinâmicas

$$\dot{x}(t) = f(x(t), u(t), t) \tag{7}$$

aqui escritas na forma de espaço de estados, inequações algébricas na forma

$$\mathbf{b}[\mathbf{x}(t), \mathbf{u}(t), t] \leq \mathbf{0}, \tag{8}$$

e condições de contorno:

$$\boldsymbol{\phi}[\mathbf{x}(t_0), t_0, \mathbf{x}(t_f), t_f] = 0 \tag{9}$$

5.1) Problema de Erro de Rastreamento Mínimo

O objetivo neste tipo de problema é levar a trajetória das variáveis de estado $x(t)$ para uma trajetória desejáda x_d [17].

Exemplo:

$$\boldsymbol{L(x,u) = |x - x_d|} \text{ ou} \qquad (10)$$

$$\boldsymbol{L(x,u) = (x - x_d)^T \cdot (x - x_d)}$$

5.2) Problema de Energia Mínima ou Esforço Mínimo

Neste tipo de problema, o objetivo é ir do estado inicial ao estado final utilizando o mínimo de energia ou esforço [17].

Exemplo:

$$\boldsymbol{L(x,u) = |u|} \qquad (11)$$

$$L(\boldsymbol{x,u}) = u^T \cdot u$$

5.3) Problema de Minimização Combinada ou Multiobjetivo

Neste tipo de problema, existe mais de um objetivo a ser atingido pelo sistema de controle ótimo [17].

Exemplo:

$$L(\boldsymbol{x,u}) = (x - x_d)^T \cdot Q \cdot (x - x_d) + u^T \cdot R \cdot u \qquad (12)$$

onde Q e R são pesos para os estados e entrada, respectivamente.

$Q \uparrow \rightarrow$ chegar no estado x_d com maior precisão

$R \uparrow \rightarrow$ economizar energia

6) Metodologia

6.1) Equações de Movimento

Devido ao grau de complexidade do sistema, simplificações foram aplicadas à representação matemática do modelo referente a robôs humanoides presentes no mundo real, de modo a facilitar a derivação das equações de movimento e a validação dos métodos propostos por esse projeto de iniciação científica. São adotadas as seguintes simplificações:

- O modelo representará apenas o movimento realizado no plano (x,y), sagital.
- A massa dos pés do robô é considerada desprezível.

- Os braços, cabeça e outros membros ligados ao tronco do robô serão reduzidos a um único corpo rígido, representado por um ponto de massa e momento de inércia com relação ao centro de gravidade.
- Apenas um pé de cada vez estará em contato com o solo durante todo o ciclo da marcha. Para garantir isso, é necessário que exista a troca instantânea dos pés exatamente no instante em que o robô estiver prestes a assumir a configuração de apoio duplo. Dessa forma, o robô permanecerá sempre na fase de apoio simples.
- Na resolução do problema de Controle Ótimo, o robô dará um passo do tamanho de seu pé, para que não existam descontinuidade na trajetória do ZMP no momento de transição dos pés.

O modelo é representado pela Fig. 11, onde o seguimento:

- L_1 representa o comprimento da tíbia da perna esquerda do robô;
- L_2 representa o comprimento da coxa da perna esquerda do robô;
- L_3 representa o comprimento do conjunto tronco, braços e cabeça do robô reduzidos a um único corpo rígido;
- L_4 representa o comprimento da coxa da perna direita do robô;
- L_5 representa o comprimento da tíbia da perna direita do robô.

Os seguimentos do robô são conectados através de juntas de revolução, onde:

- θ_1 representa a posição angular do tornozelo da perna esquerda do robô;
- θ_2 representa a posição angular do joelho da perna esquerda do robô;
- θ_3 representa a posição angular do quadril lateral esquerdo do robô;
- θ_4 representa a posição angular do quadril lateral direito do robô;
- θ_4 representa a posição angular do joelho da perna direita do robô;

Por fim podemos descrever os parâmetros dinâmicos, onde:

- m_i é a massa do seguimento i;
- r_i é a distância entre uma junta e o centro de massa do seguimento i;

- I_i e ω_i são o momento de inércia e a rotação do seguimento i com relação à posição do seu centro de massa;
- τ_i é o torque externo aplicado à junta θ_i;
- g é a aceleração da gravidade.

O contato entre o pé e o solo é modelado como uma junta de engastamento, dada a restrição imposta pela condição do ZMP. O problema de controle ótimo garantirá que a solução será tal que o ZMP ficará sob a área de apoio do pé ao longo de todo o movimento. As equações de movimento obtidas através do método de Euler-Lagrange.

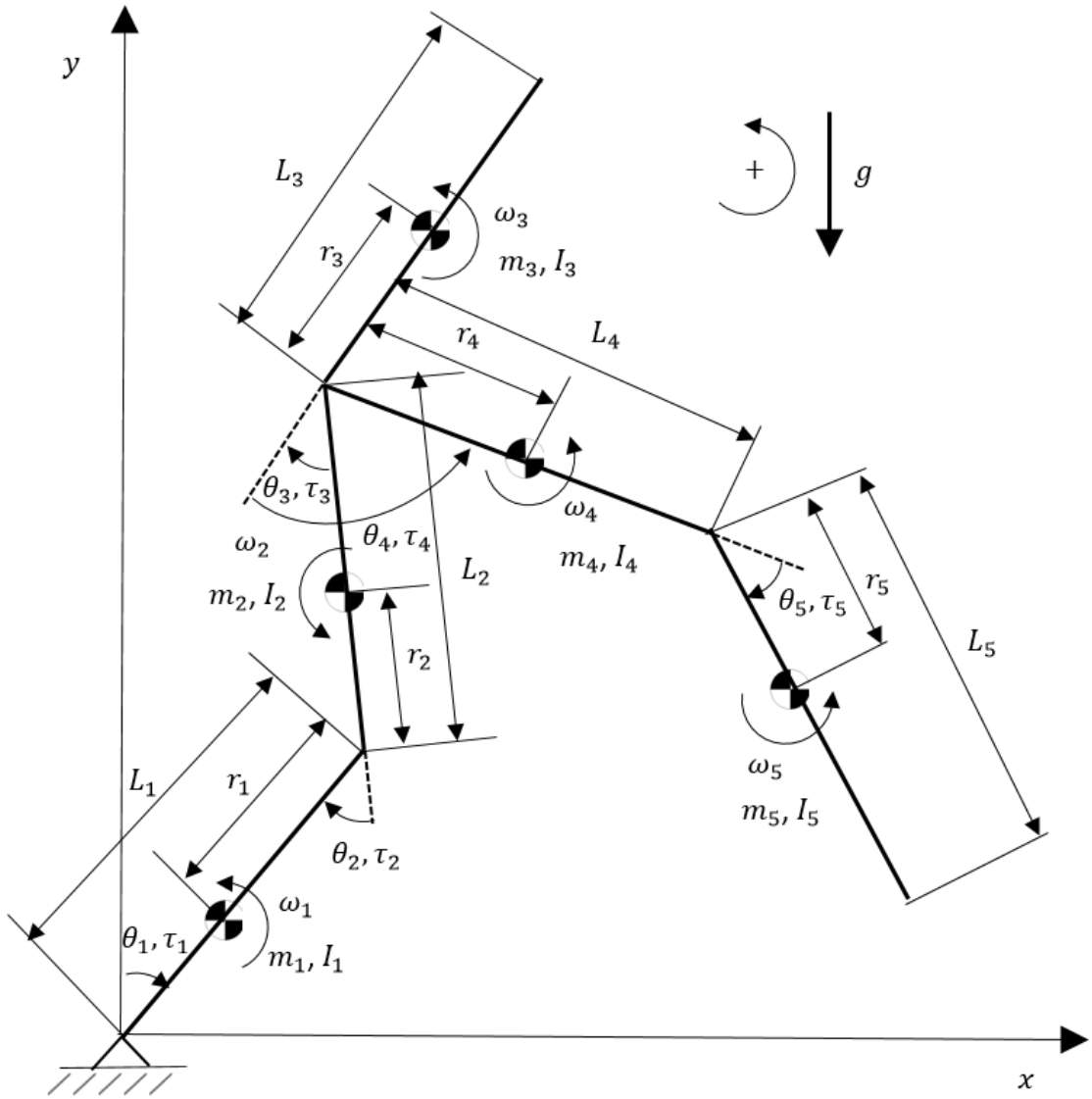


Fig. 11 – Modelagem dinâmica de um robô humanoide

O sistema possui 5 graus de liberdade descritos pelas coordenadas generalizadas no vetor:

$$q = \begin{pmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \end{pmatrix} = \begin{pmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \end{pmatrix} \tag{13}$$

6.1.1) Cinemática Direta

Com o eixo de referência z saindo do plano (x, y), a orientação do centro de massa de cada corpo é definida como:

$$\begin{pmatrix} \varphi_1 \\ \varphi_2 \\ \varphi_3 \\ \varphi_4 \\ \varphi_5 \end{pmatrix} = \begin{pmatrix} -\theta_1 \\ -\theta_1 - \theta_2 \\ -\theta_1 - \theta_2 - \theta_3 \\ -\theta_1 - \theta_2 - \theta_3 + \theta_4 \\ -\theta_1 - \theta_2 - \theta_3 + \theta_4 - \theta_5 \end{pmatrix} \tag{14}$$

A posição do centro de massa de cada corpo é definida como:

$$p_{cm_1} = \begin{pmatrix} x_{cm_1} \\ y_{cm_1} \\ z_{cm_1} \end{pmatrix} = \begin{pmatrix} (r_1 - L_1) \cdot \sin(\varphi_1) \\ (L_1 - r_1) \cdot \cos(\varphi_1) \end{pmatrix} \tag{15}$$

$$p_{cm_2} = \begin{pmatrix} x_{cm_2} \\ y_{cm_2} \\ z_{cm_2} \end{pmatrix} = \begin{pmatrix} -L_1 \cdot \sin(\varphi_1) + (r_2 - L_2) \cdot \sin(\varphi_2) \\ L_1 \cdot \cos(\varphi_1) + (L_2 - r_2) \cdot \cos(\varphi_2) \\ 0 \end{pmatrix} \tag{15}$$

$$p_{cm_3} = \begin{pmatrix} x_{cm_3} \\ y_{cm_3} \\ z_{cm_3} \end{pmatrix} = \begin{pmatrix} -L_1 \cdot \sin(\varphi_1) - L_2 \cdot \sin(\varphi_2) - r_3 \cdot \sin(\varphi_3) \\ L_1 \cdot \sin(\varphi_1) + L_2 \cdot \cos(\varphi_2) + r_3 \cdot \cos(\varphi_3) \end{pmatrix} \tag{16}$$

$$p_{cm_4} = \begin{pmatrix} x_{cm_4} \\ y_{cm_4} \\ z_{cm_4} \end{pmatrix} = \begin{pmatrix} -L_1 \cdot \sin(\varphi_1) - L_2 \cdot \sin(\varphi_2) + r_4 \cdot \sin(\varphi_4) \\ L_1 \cdot \cos(\varphi_1) + L_2 \cdot \cos(\varphi_2) - r_4 \cdot \cos(\varphi_4) \end{pmatrix} \tag{17}$$

(18)

$$p_{cm_5} = \begin{pmatrix} x_{cm_5} \\ y_{cm_5} \\ z_{cm_5} \end{pmatrix} = \begin{pmatrix} -L_1 \cdot \sin(\varphi_1) - L_2 \cdot sin(\varphi_2) + L_4 \cdot \sin(\varphi_4) + r_5 \cdot \sin(\varphi_5) \\ L_1 \cdot \cos(\varphi_1) + L_2 \cdot cos(\varphi_2) - L_4 \cdot \cos(\varphi_4) - r_5 \cdot \cos(\varphi_5) \end{pmatrix}$$

A velocidade de cada corpo é definida como:

$$v_{cm_i} = \begin{pmatrix} \dot{x_{cm_i}} \\ \dot{y_{cm_i}} \\ \dot{z_{cm_i}} \end{pmatrix} \tag{20}$$

O vetor rotação de cada segmento é:

$$\omega_1 = \begin{pmatrix} 0 \\ 0 \\ -\dot{\theta}_1 \end{pmatrix} \tag{21}$$

$$\omega_2 = \begin{pmatrix} 0 \\ 0 \\ -\dot{\theta}_1 - \dot{\theta}_2 \end{pmatrix} \tag{22}$$

$$\omega_3 = \begin{pmatrix} 0 \\ 0 \\ -\dot{\theta}_1 - \dot{\theta}_2 - \dot{\theta}_3 \end{pmatrix} \tag{23}$$

$$\omega_4 = \begin{pmatrix} 0 \\ 0 \\ -\dot{\theta}_1 - \dot{\theta}_2 - \dot{\theta}_3 + \dot{\theta}_4 \end{pmatrix} \tag{24}$$

$$\omega_5 = \begin{pmatrix} 0 \\ 0 \\ -\dot{\theta}_1 - \dot{\theta}_2 - \dot{\theta}_3 + \dot{\theta}_4 - \dot{\theta}_5 \end{pmatrix} \tag{25}$$

6.1.2) Energia Cinética Total

No sistema de corpos rígidos em que o corpo i de massa m_i e matriz de inércia em relação ao centro de massa I_{cm_i}, se move com velocidade do centro de massa v_{cm_i} e vetor rotação ω_i, a energia cinética total do sistema é dada por:

(26)

$$T = \sum_{i=1}^{5} \left(\frac{1}{2} \cdot m_i {v_{cm_i}}^T v_{cm_i} + \frac{1}{2} {\omega_i}^T I_{cm_i} \omega_i \right)$$

onde:

$$I_{cm_i} = \begin{pmatrix} \int (y^2 + z^2) \cdot dm & -\int x \cdot y \cdot dm & -\int x \cdot z \cdot dm \\ -\int x \cdot y \cdot dm & \int (x^2 + z^2) \cdot dm & -\int y \cdot z \cdot dm \\ -\int x \cdot z \cdot dm & -\int y \cdot z \cdot dm & \int (x^2 + y^2) \cdot dm \end{pmatrix} \quad (27)$$

I_{cm_i} e ω_i precisam ser expressos na mesma base de referência, mas o produto ${\omega_i}^T \cdot I_{cm_i} \cdot \omega_i$ é invariante com relação à qualquer referência escolhida.

6.1.3) Cálculo de velocidades através do Jacobiano

A matriz Jacobiana para o corpo *i* de um sistema multicorpo esclerônomos é definida pelas equações

$$v_{cm_i} = J_{p_i}(q) \cdot \dot{q} \quad (28)$$

$$\omega_i = J_{o_i}(q) \cdot \dot{q} \quad (29)$$

onde

$$J_{p_i} = \begin{pmatrix} \frac{\partial x_{cm_i}}{\partial q_1} & \frac{\partial x_{cm_i}}{\partial q_2} & \frac{\partial x_{cm_i}}{\partial q_3} & \frac{\partial x_{cm_i}}{\partial q_4} & \frac{\partial x_{cm_i}}{\partial q_5} \\ \frac{\partial y_{cm_i}}{\partial q_1} & \frac{\partial y_{cm_i}}{\partial q_2} & \frac{\partial y_{cm_i}}{\partial q_3} & \frac{\partial y_{cm_i}}{\partial q_4} & \frac{\partial y_{cm_i}}{\partial q_5} \\ \frac{\partial z_{cm_i}}{\partial q_1} & \frac{\partial z_{cm_i}}{\partial q_2} & \frac{\partial z_{cm_i}}{\partial q_3} & \frac{\partial z_{cm_i}}{\partial q_4} & \frac{\partial z_{cm_i}}{\partial q_5} \end{pmatrix} \quad (30)$$

$$J_{o_i} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \frac{\partial \varphi_1}{\partial q_1} & \frac{\partial \varphi_2}{\partial q_2} & \frac{\partial \varphi_3}{\partial q_3} & \frac{\partial \varphi_4}{\partial q_4} & \frac{\partial \varphi_5}{\partial q_5} \end{pmatrix} \quad (31)$$

6.1.4) Matriz de Massa de um Sistema Multicorpo

Substituindo os jacobianos na equação da energia cinética:

$$T = \frac{1}{2} \cdot \sum_{i=1}^{5} \left(m_i \cdot \dot{q}^T \cdot {J_{p_i}}^T \cdot J_{p_i} \cdot \dot{q} + \dot{q}^T \cdot {J_{o_i}}^T \cdot I_{cm_i} \cdot J_{o_i} \cdot \dot{q} \right) = \frac{1}{2} \cdot \dot{q}^T \cdot H(q) \cdot \dot{q} \quad (26)$$

Onde

$$H(q) = \sum_{i=1}^{5} \left(m_i \cdot {J_{p_i}}^T \cdot J_{p_i} + {J_{o_i}}^T \cdot I_{cm_i} \cdot J_{o_i} \right) \tag{32}$$

A matriz $H(q)$ contém todas as propriedades de massa presentes no sistema dinâmico. Essa matriz é conhecida como matriz de massa de um sistema multicorpo.

6.1.5) Energia Potencial Total

A energia potencial total é dada por:

$$U(q) = \sum_{i=1}^{5} m_i \cdot g \cdot y_{cm_i} \tag{39}$$

6.1.6) Equação de Euler-Lagrange

A função de Lagrange L é definida por :

$$L(q, \dot{q}) = T(q, \dot{q}) - U(q) \tag{40}$$

Usando a função de Lagrange, as equações de movimento do sistema dinâmico são dadas por

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_k}\right) - \frac{\partial L}{\partial q_k} = Q_k \tag{41}$$
$$k = 1,2,3,4,5.$$

onde Q são as forças generalizadas correspondentes às coordenadas generalizadas q_k.

A equação (41) pode ser reescrita como [1] :

$$H(q) \cdot \ddot{q} + C(q, \dot{q}) + G(q) = Q \tag{41}$$

onde *H* é a matriz de massa, *C* é o vetor de forças generalizadas centrífuga e de Coriolis e *G* é o vetor de forças generalizadas gravitacionais [1].

6.1.8) Forças Generalizadas

Considerando o princípio do trabalho virtual realizado pelas forças não conservativas podemos identificar as forças generalizadas agindo no sistema

$$\delta W = \sum_{i=1}^{5} Q_i \cdot \delta q_i \tag{44}$$

Se o trabalho virtual é dado pelo produto interno dos torques das juntas e os deslocamentos virtuais das articulações, e se os destocamentos virtuais das articulações coincidirem com os deslocamentos virtuais das coordenadas generalizadas, os torques nas articulares correspondem às próprias forças generalizadas [1].

$$\boldsymbol{Q_i = \tau_i} \tag{45}$$

6.2) Representação no espaço de estados

$$\ddot{q} = H(q)^{-1} \cdot (\tau - C(q, \dot{q}) - G(q)) \tag{46}$$

$$\dot{x} = \begin{pmatrix} \dot{q} \\ \ddot{q} \end{pmatrix} = \begin{pmatrix} \dot{q} \\ H(q)^{-1} \cdot (Q - C(q, \dot{q}) - G(q)) \end{pmatrix} \tag{47}$$

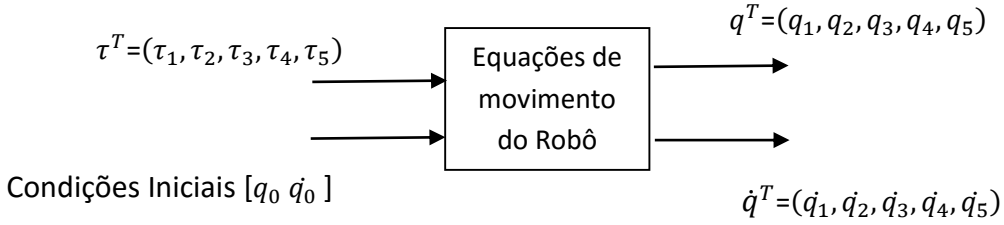


Fig. 12 – Representação de entradas e saídas do sistema

6.3) Cálculo do ZMP para um Sistema Multicorpo

O cálculo do ZMP para um sistema multicorpo como o robô humanoide de cinco graus de liberdade resulta na seguinte fórmula [19]:

$$\boldsymbol{x_{ZMP_{CARTPOLE}}} = \frac{\sum_{i=1}^{5} m_i \cdot (\boldsymbol{\ddot{y}_{cm_i}} + g) \cdot x_{cm_i} - \sum_{i=1}^{5} m_i \cdot \boldsymbol{\ddot{x}_{cm_i}} \cdot y_{cm_i} - \sum_{i=1}^{5} I_i \cdot \boldsymbol{\omega_i}}{\sum_{i=1}^{5} \boldsymbol{m_i} \cdot (\boldsymbol{\ddot{y}_{cm_i}} + \boldsymbol{g_y})} \tag{48}$$

6.3) Cálculo do ZMP para o sistema Cart Table

O ZMP pode ser calculado de uma maneira simplificada se realizarmos a aproximação do modelo multicorpo para um sistema onde um carro sem massa corre em uma mesa também sem massa, este sistema é bastante conhecido pelo nome Cart Table. O modelo simplificado é mostrado nas Fig. 13 e 14.

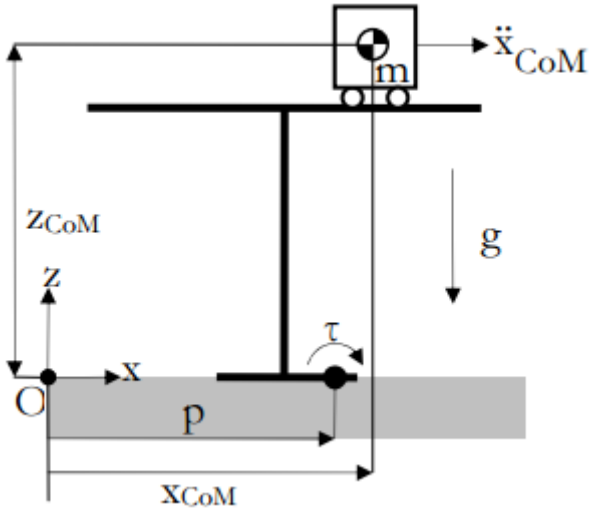


Fig. 13 – Cart Table

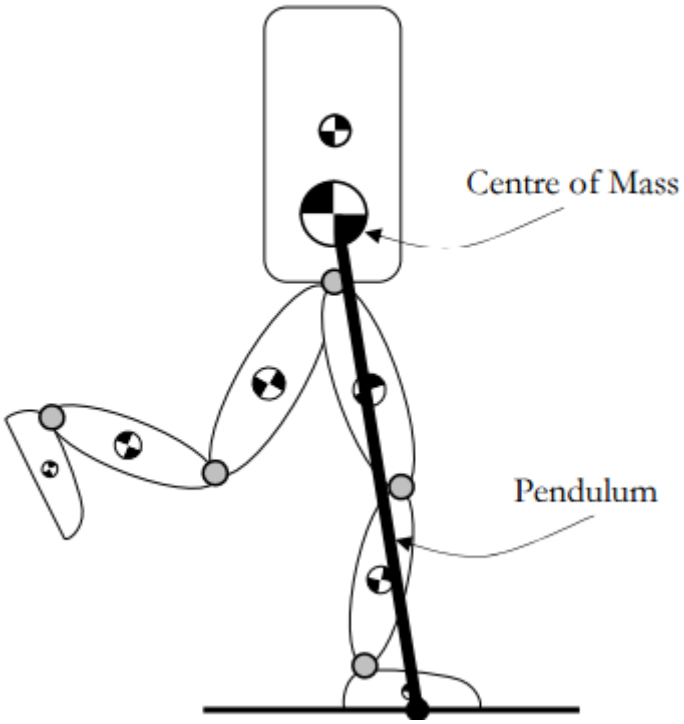


Fig. 14 – Modelo Simplicado baseado no sistema Cart Pole

O cálculo do ZMP para um sistema cart table resulta na seguinte fórmula:

$$x_{ZMP} = x_{COM} - \frac{\ddot{x_{cm_i}}}{g} z_{COM} \tag{49}$$

6.4) Formulação do Problema de Controle Ótimo

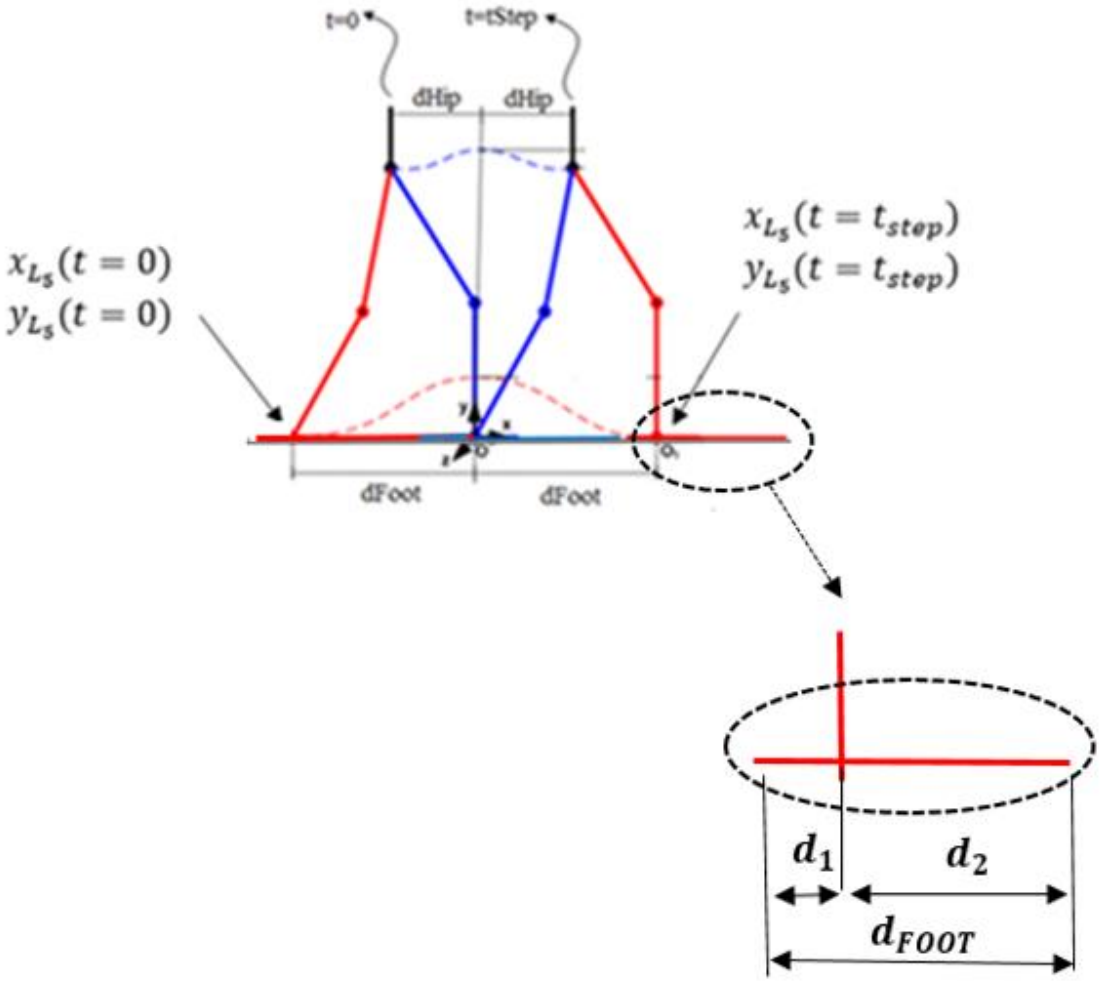


Fig. 12 – Modelo fundamentado para a formulação do problema de Controle Ótimo

Onde

- $\boldsymbol{x_{L_5}}$ e $\boldsymbol{y_{L_5}}$ são definidos como a posição da extremidade distal do elo $\boldsymbol{L_5}$ no plano $(\boldsymbol{x}, \boldsymbol{y})$;
- $\boldsymbol{d_1}$ é definido como o comprimento da parte anterior do pé;

- $\boldsymbol{d_2}$ é definido como o comprimento da parte posterior do pé;
- $\boldsymbol{d_{FOOT}}$ é definido como o comprimento do pé, e também o tamanho do passo do robô.

6.4.1) Restrições do modelo

a) Equações do Movimento

$$\dot{\boldsymbol{x}} = \begin{pmatrix} \dot{\boldsymbol{q}} \\ \ddot{\boldsymbol{q}} \end{pmatrix} = \begin{pmatrix} \dot{\boldsymbol{q}} \\ \boldsymbol{H(q)^{-1}} \cdot (\boldsymbol{Q} - \boldsymbol{C(q,\dot{q})} - \boldsymbol{G(q)}) \end{pmatrix} \tag{47}$$

b) ZMP na base de suporte

$$-\boldsymbol{d_2} \leq \boldsymbol{x_{ZMP}(t)} \leq \boldsymbol{d_1}$$

c) Condições Iniciais e Finais do Estados

$$\boldsymbol{x_{L_5}(t=0) = -d_1 - d_2 = -d_{FOOT}}$$

$$\boldsymbol{y_{L_5}(t=0) = 0}$$

$$\boldsymbol{x_{L_5}(t=t_f) = d_1 + d_2 = d_{FOOT}}$$

$$\boldsymbol{y_{L_5}(t=t_f) = 0}$$

d) Periodicidade

$$\boldsymbol{x(t=tf) = x^*(t=0)}$$

Onde $\boldsymbol{x}^*$ são os valores espelhados do estado $\boldsymbol{x}$. A imposição desta condição faz com que o movimento seja periódico, sendo assim a solução completa do problema de controle ótimo pode ser obtida apenas realizando cálculos referentes à metade do ciclo

de marcha e utilizando os valores espelhados para realizar a simulação dinâmica do ciclo de marcha completo.

e) Restringir torque nos motores, ou hipertensão do joelho

$$\tau_{MÍN_i} < \tau_i < \tau_{MAX_i}$$

6.4.2) Critérios de Estabilidade

a) Projeção do Centro de Gravidade

Projeção do Centro de Gravidade do robô na direção x deve ser permanecer dentro do polígono de apoio durante toda a trajetória.

$$-0,5 \leq x_{COM} \leq 0.25\ [m]$$

b) Zero Moment Point

O ponto onde o momento atuante é nulo deve ser permanecer dentro do polígono de apoio durante toda a trajetória.

$$-0,5 \leq x_{ZMP} \leq 0.25\ [m]$$

c) Cart Pole

O ponto onde o momento atuante é nulo deve ser permanecer dentro do polígono de apoio durante toda a trajetória.

$$-0,5 \leq x_{ZMP_{CARTPOLE}} \leq 0.25\ [m]$$

6.4.3) Função Objetivo

a) Mínima Energia

Função Objetivo: Realizar o movimento utilizando torque mínimo

Para alcançar o torque mínimo, a função objetivo a ser minimizada pelo algoritmo de Controle Ótimo é escrita como:

$$J_{objetivo} = \int_{t=0}^{t=tf} (\tau_1(t)^2 + \tau_2(t)^2 + \tau_3(t)^2 + \tau_4(t)^2 + \tau_5(t)^2)$$

b) Máxima Velocidade

Para obter máxima velocidade é necessário que o movimento seja realizado no menor tempo de simulação possível, a função objetivo a ser minimizada pelo algoritmo de Controle Ótimo é escrita como:

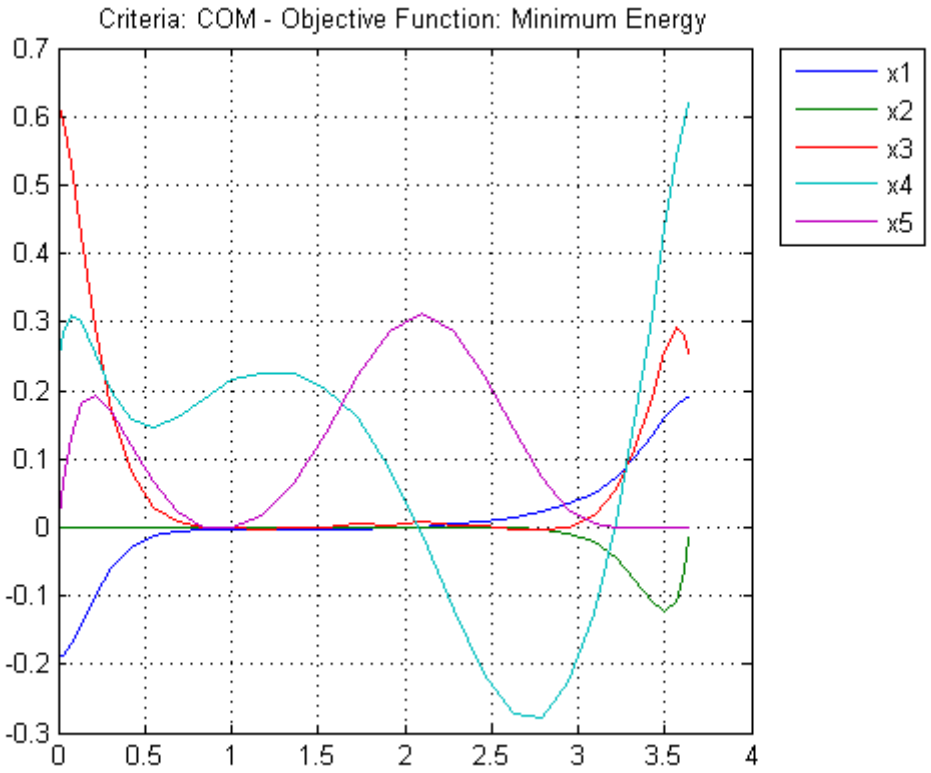
$$J_{objetivo} = tf$$

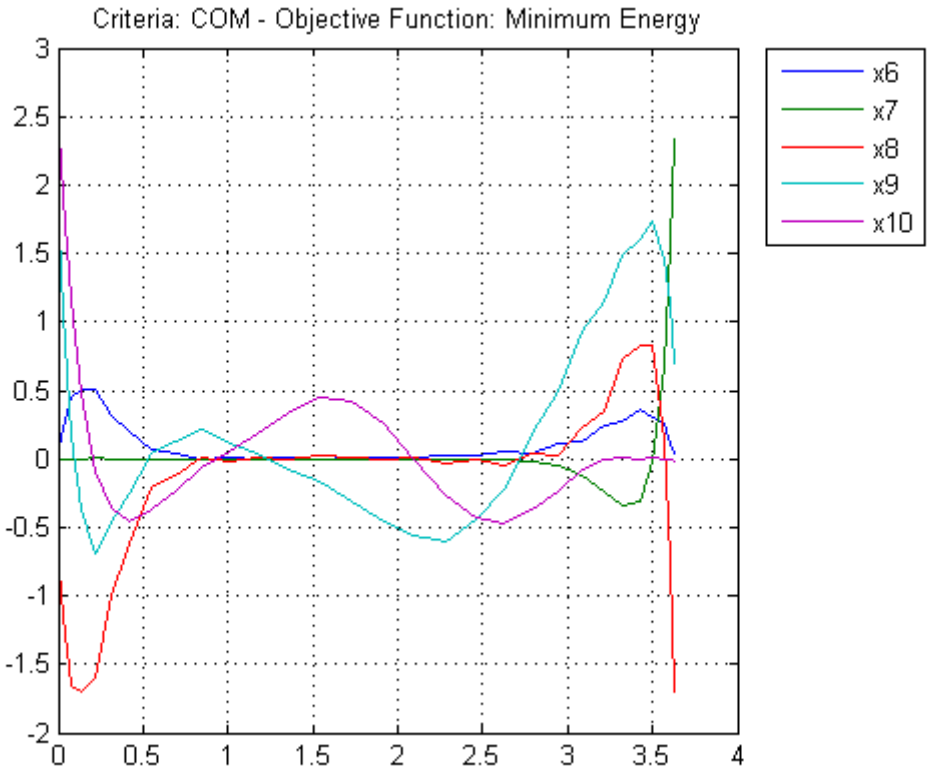
7) Experimentos e Simulações

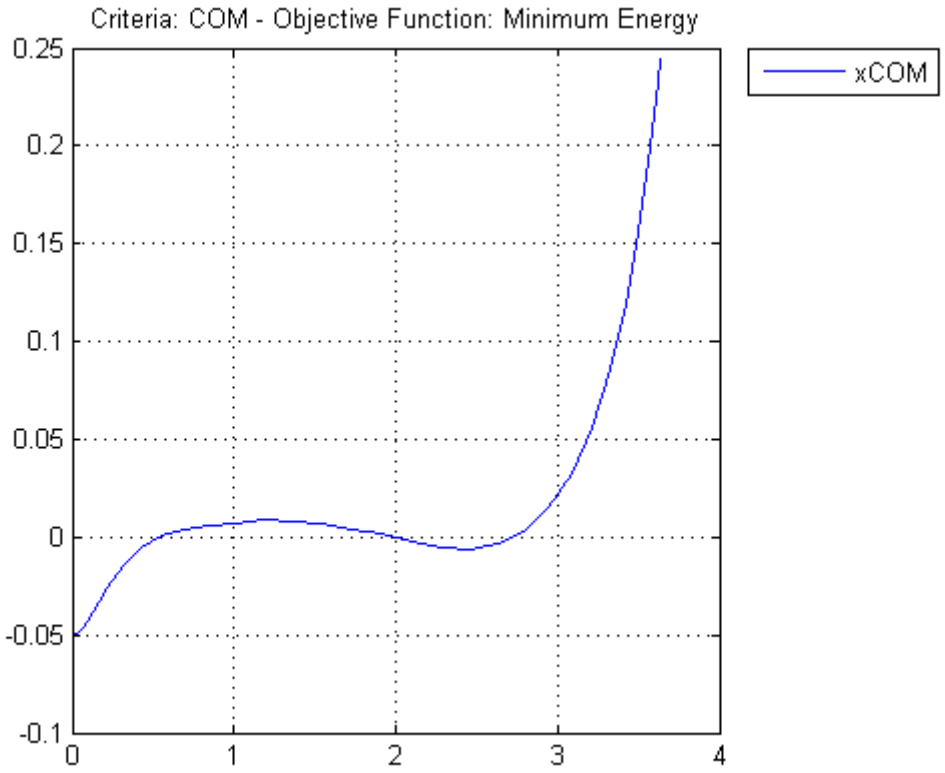
Todo o desenvolvimento e as simulações das equações de movimento foram realizadas utilizando o software Mathworks MATLAB, em conjunto com o TOMLAB toolbox PROPT para aplicações avançadas em aplicações de controle ótimo.

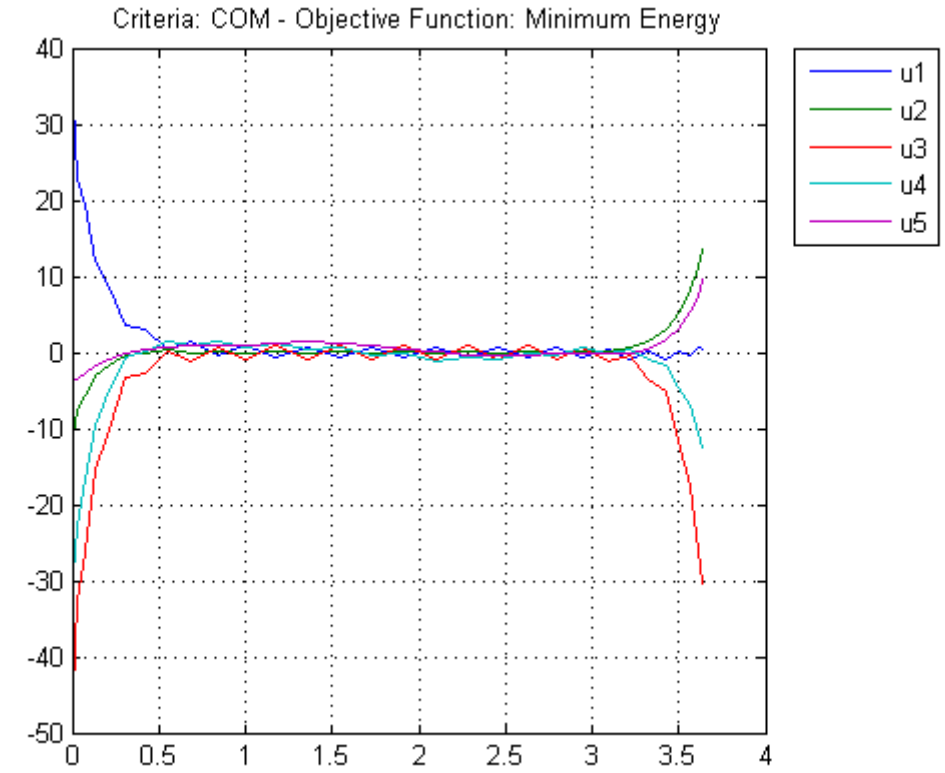
7.1) Experimento 1 – Função Objetivo: Mínima Energia

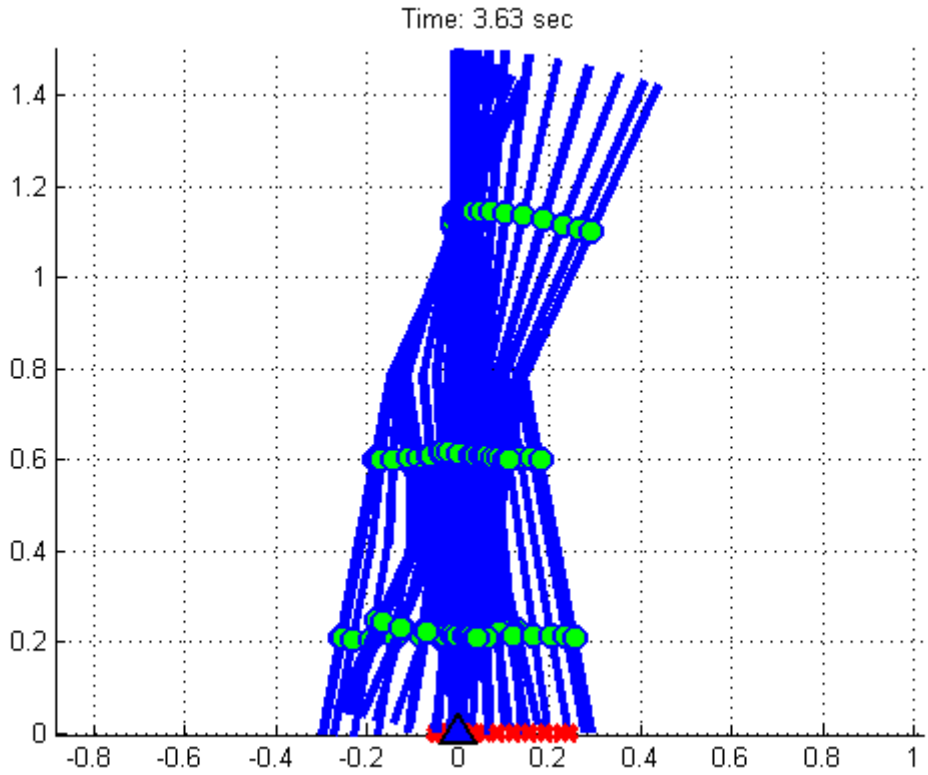
a) Critério: COM


Criteria: COM - Objective Function: Minimum Energy
x1
x2
x3
x4
x5

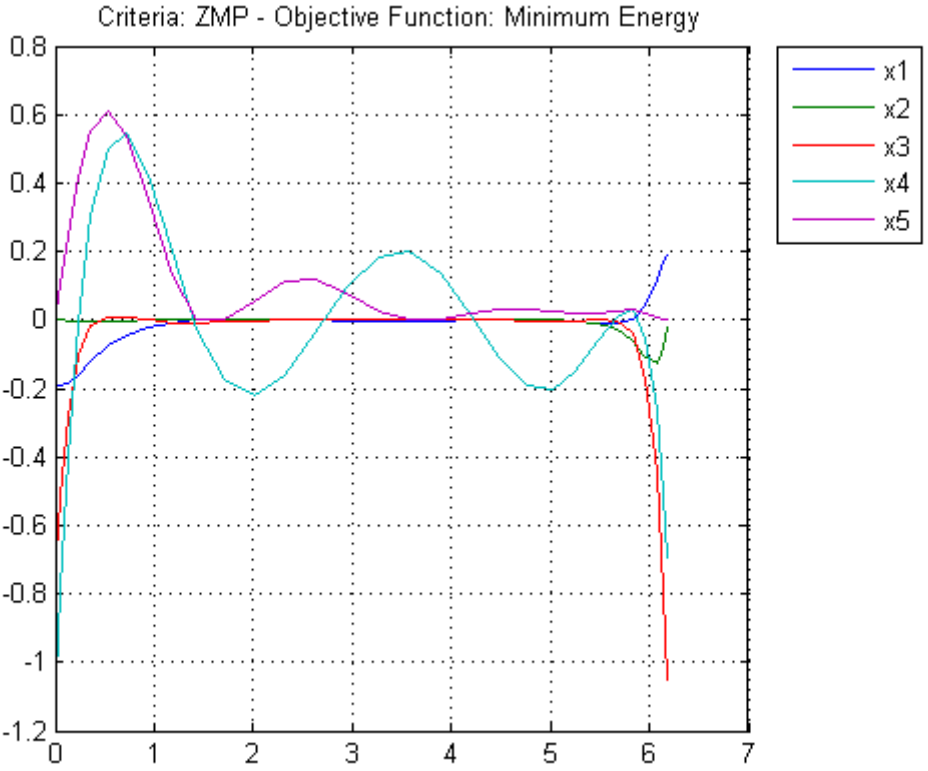

Criteria: COM - Objective Function: Minimum Energy
x6
x7
x8
x9
x10

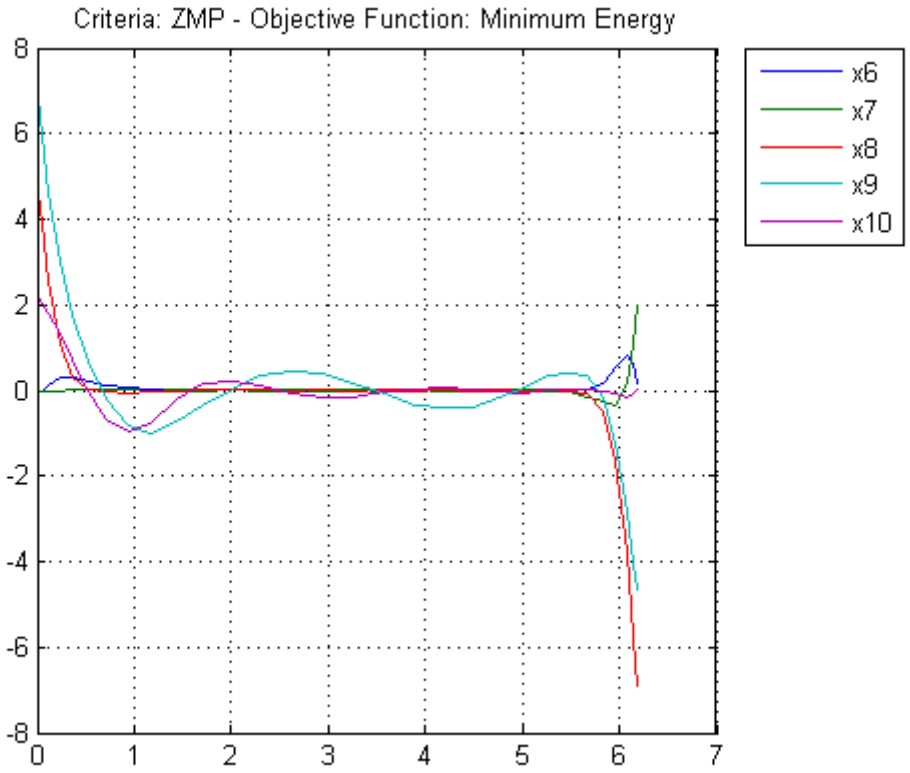

Criteria: COM - Objective Function: Minimum Energy
xCOM
0.25
0.2
0.15
0.1
0.05
0
-0.05
-0.1
0
1
2
3
4

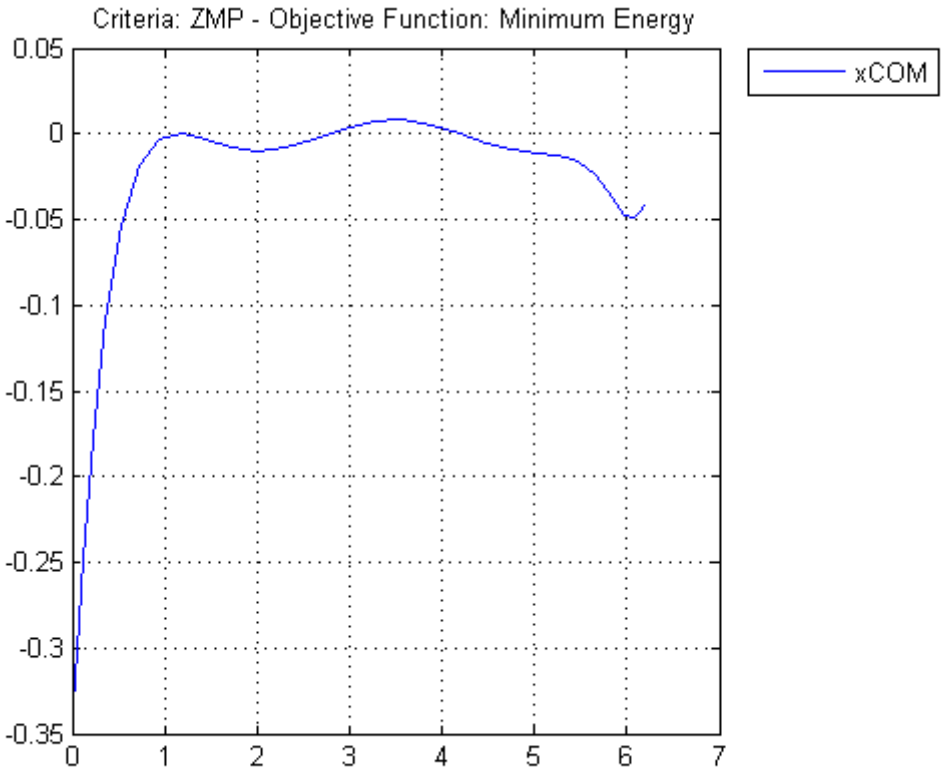

Criteria: COM - Objective Function: Minimum Energy
u1
u2
u3
u4
u5
40
30
20
10
0
-10
-20
-30
-40
-50
0
0.5
1
1.5
2
2.5
3
3.5
4

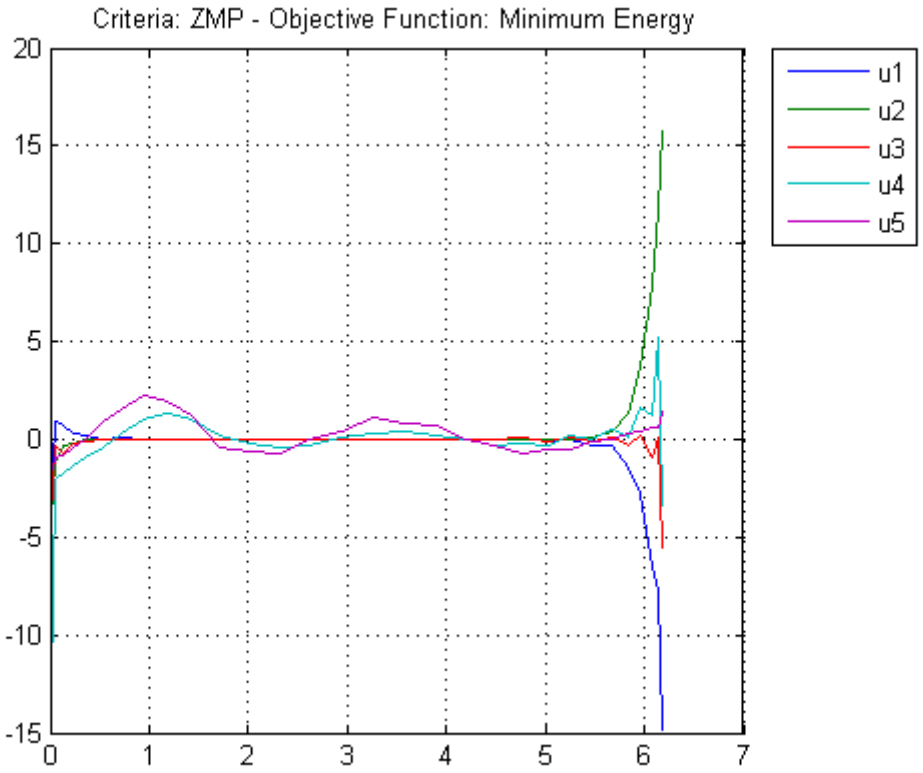


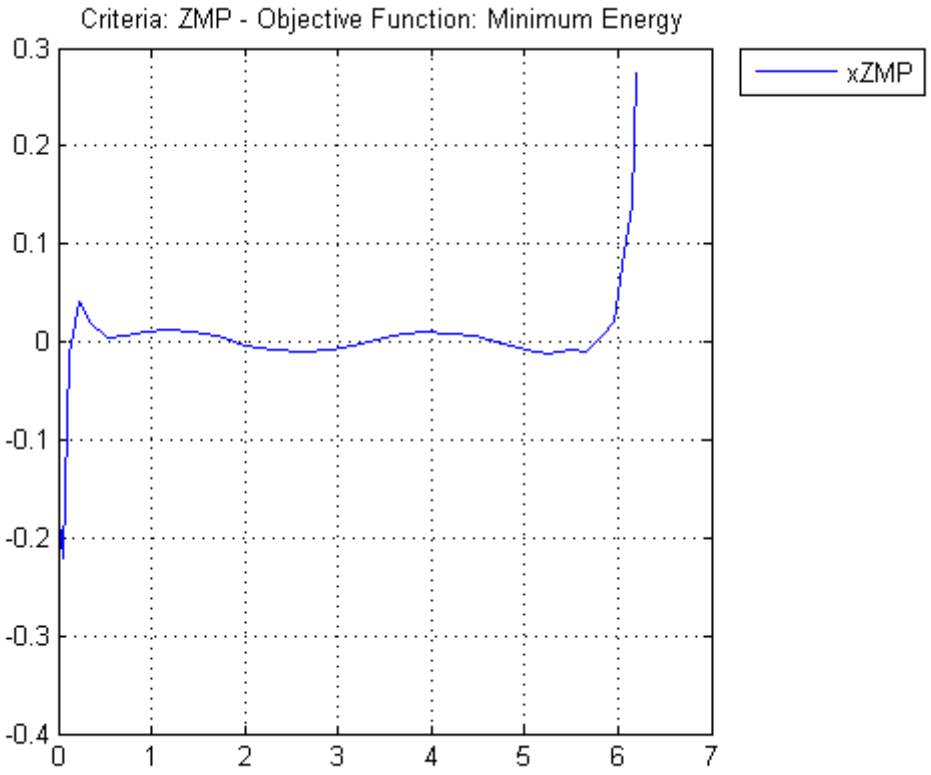
b) Critério: ZMP

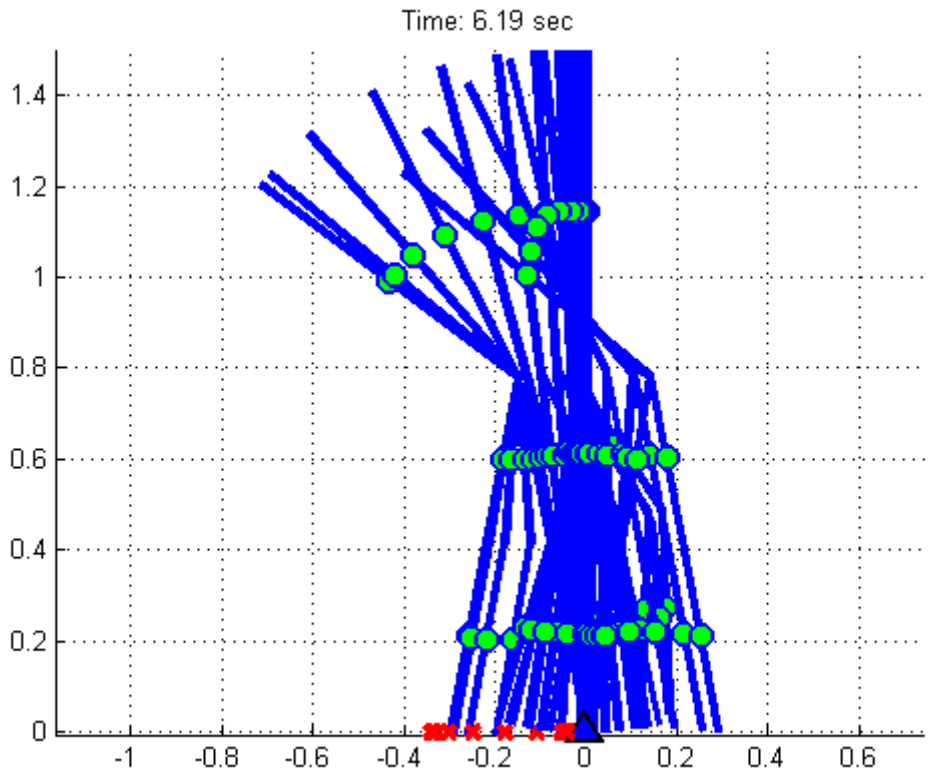



Criteria: ZMP - Objective Function: Minimum Energy
x6
x7
x8
x9
x10
8
6
4
2
0
-2
-4
-6
-8
0
1
2
3
4
5
6
7

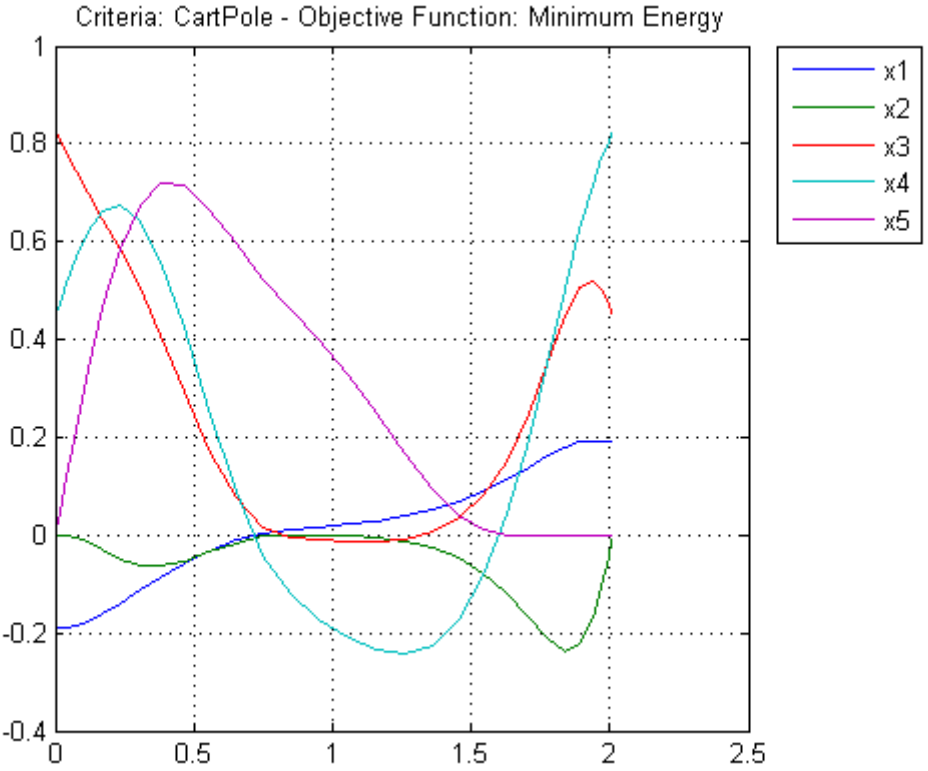

Criteria: ZMP - Objective Function: Minimum Energy
xCOM
0.05
0
-0.05
-0.1
-0.15
-0.2
-0.25
-0.3
-0.35
0
1
2
3
4
5
6
7

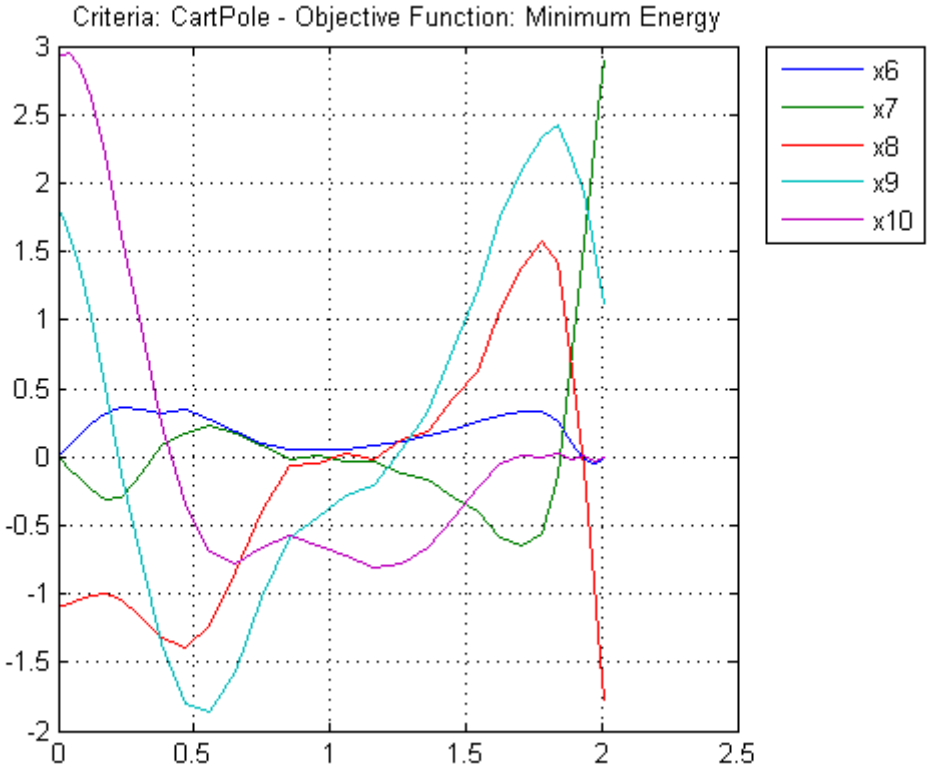

Criteria: ZMP - Objective Function: Minimum Energy
20
15
10
5
0
-5
-10
-15
0
1
2
3
4
5
6
7
u1
u2
u3
u4
u5

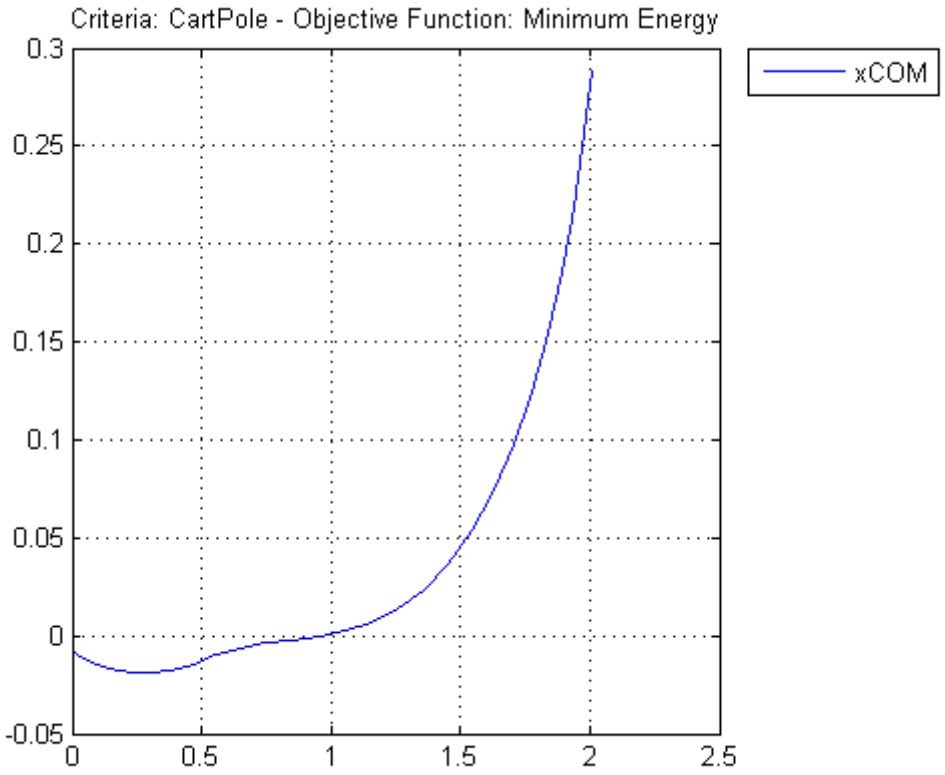

Criteria: ZMP - Objective Function: Minimum Energy
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
0
1
2
3
4
5
6
7
xZMP

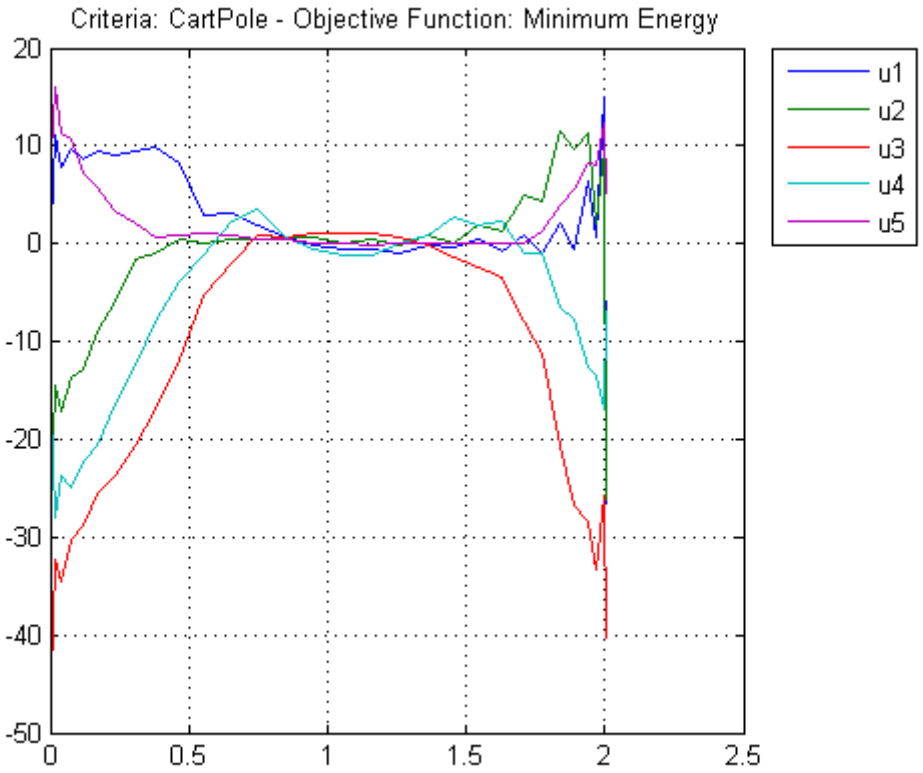


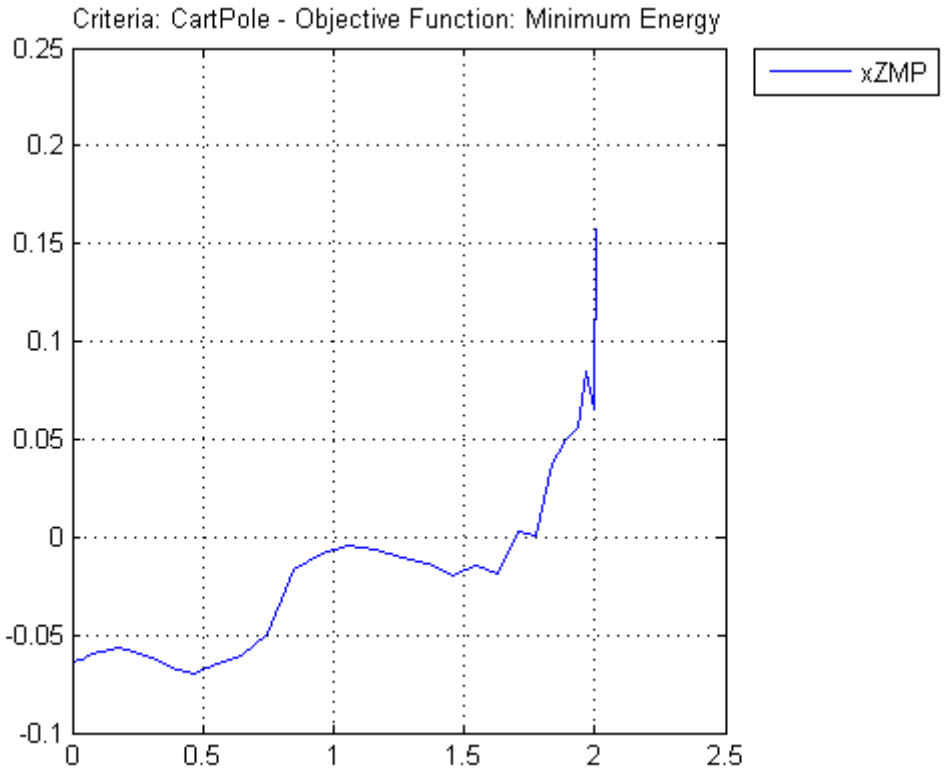
c) Critério: Cart Pole

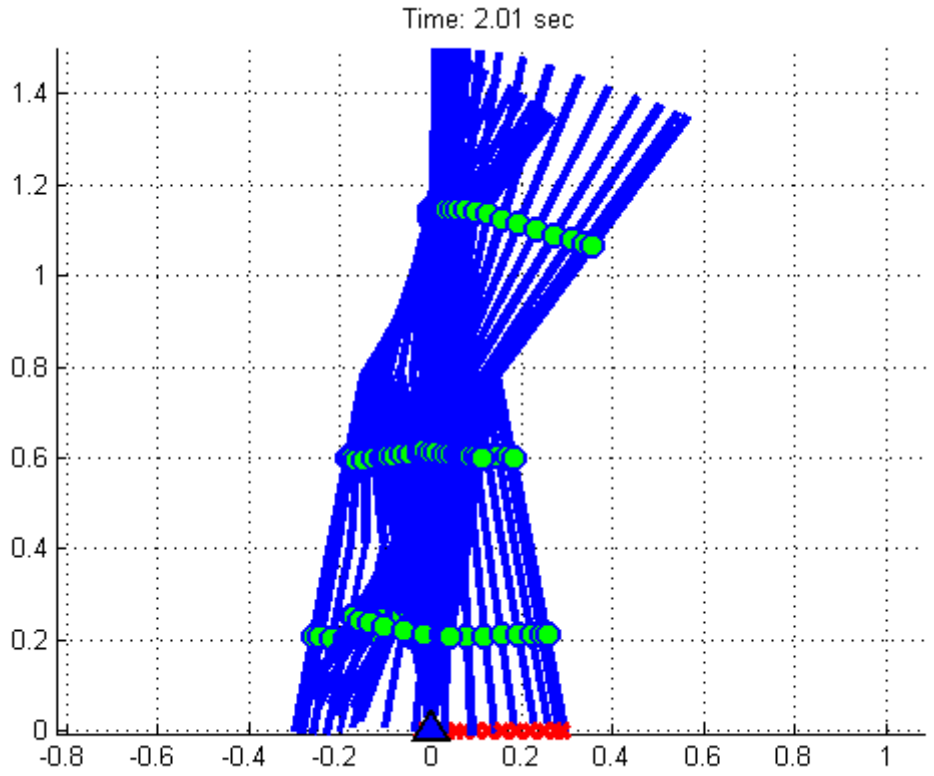



Criteria: CartPole - Objective Function: Minimum Energy
x6
x7
x8
x9
x10
3
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0
0.5
1
1.5
2
2.5


Criteria: CartPole - Objective Function: Minimum Energy
xCOM
0.3
0.25
0.2
0.15
0.1
0.05
0
-0.05
0
0.5
1
1.5
2
2.5

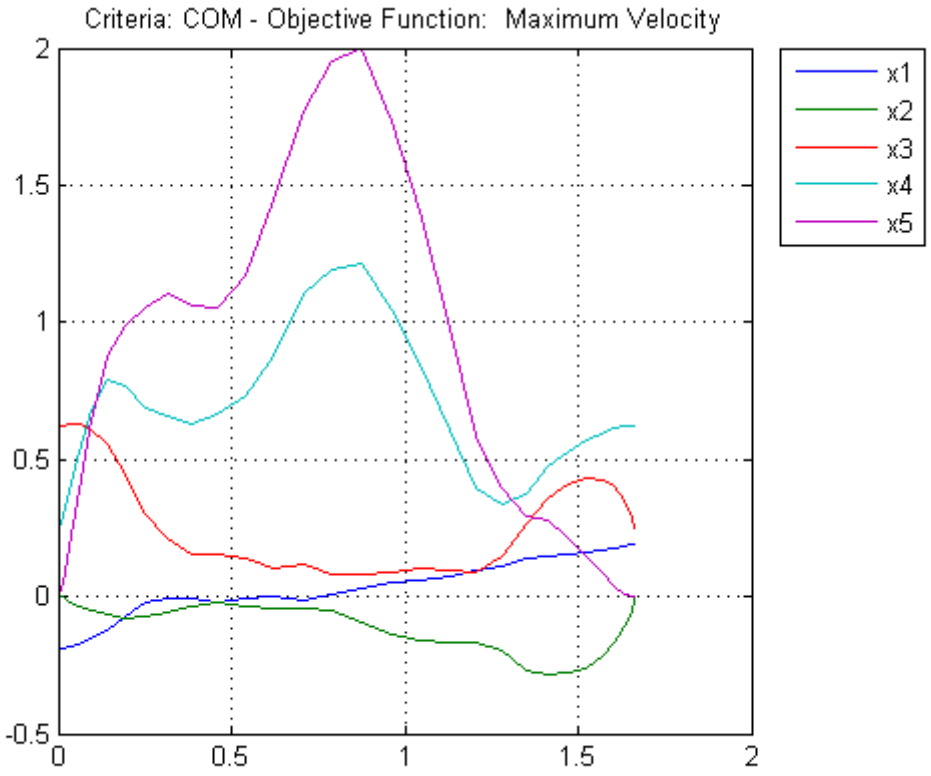

Criteria: CartPole - Objective Function: Minimum Energy
20
10
0
-10
-20
-30
-40
-50
0
0.5
1
1.5
2
2.5
u1
u2
u3
u4
u5

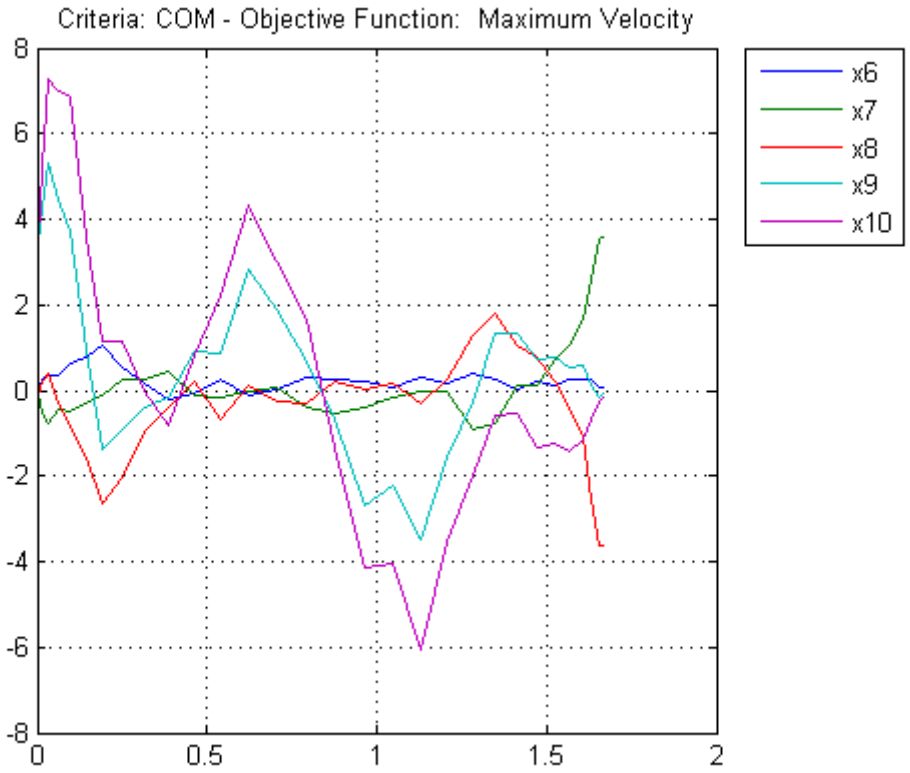


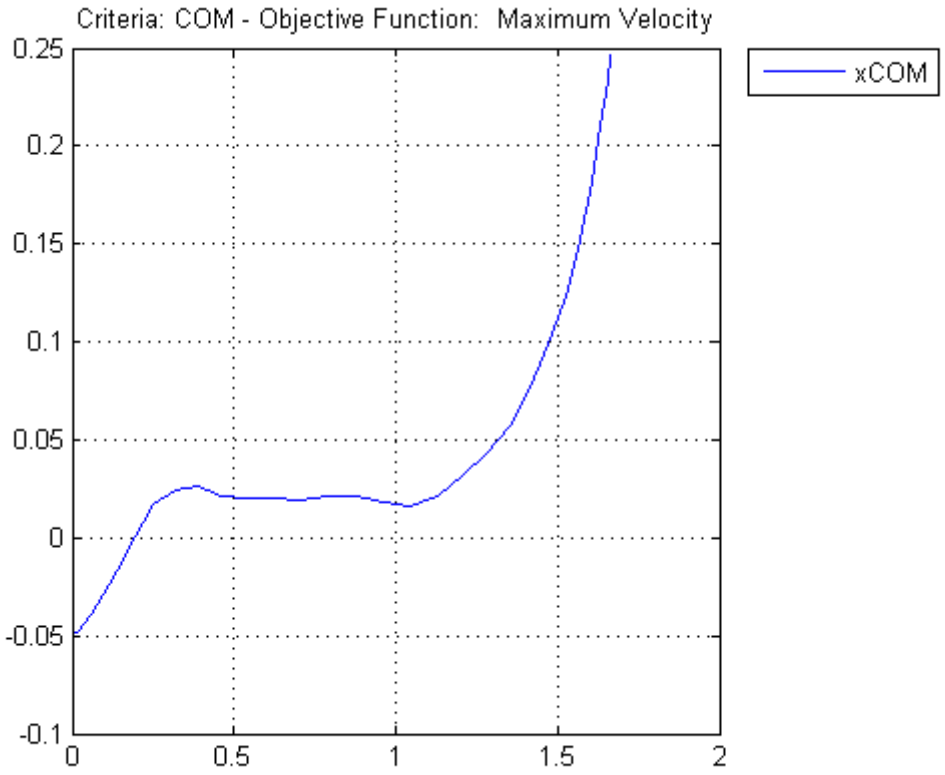


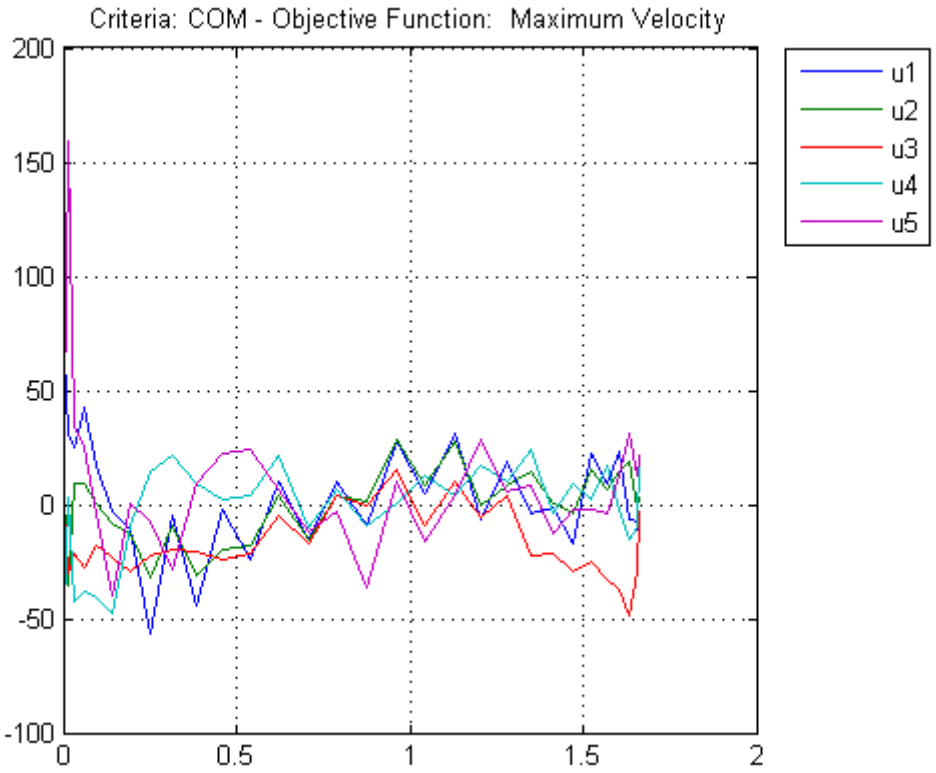
7.2) Experimento 2 – Função Objetivo: Velocidade Máxima

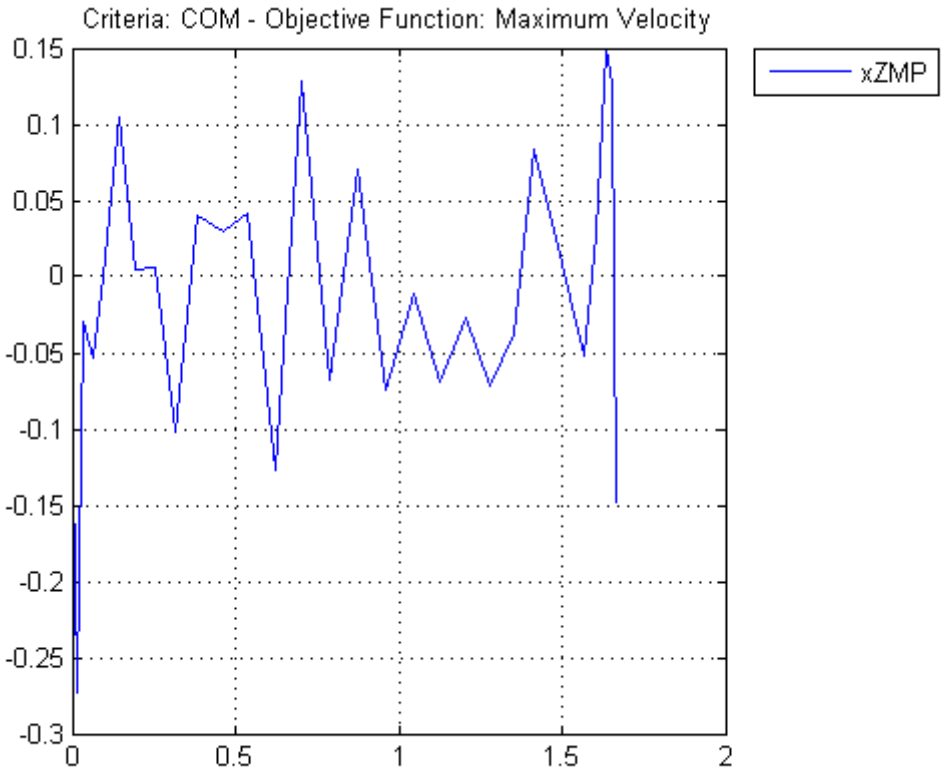
a) Critério: COM

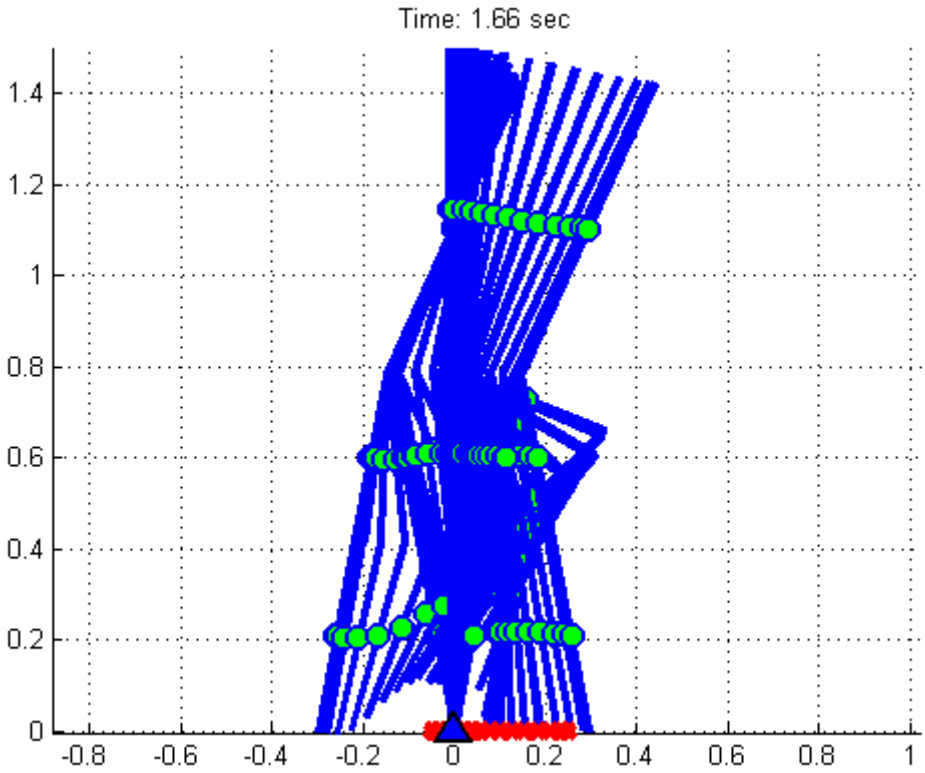



Criteria: COM - Objective Function: Maximum Velocity
8
6
4
2
0
-2
-4
-6
-8
0
0.5
1
1.5
2
x6
x7
x8
x9
x10

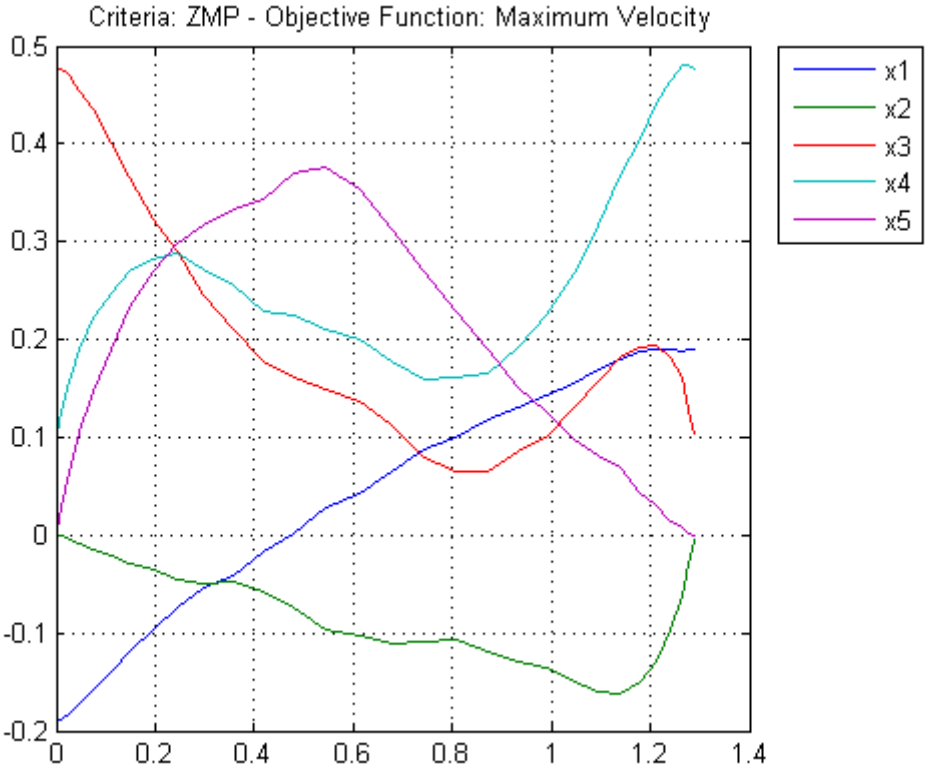

Criteria: COM - Objective Function: Maximum Velocity
0.25
0.2
0.15
0.1
0.05
0
-0.05
-0.1
0
0.5
1
1.5
2
xCOM

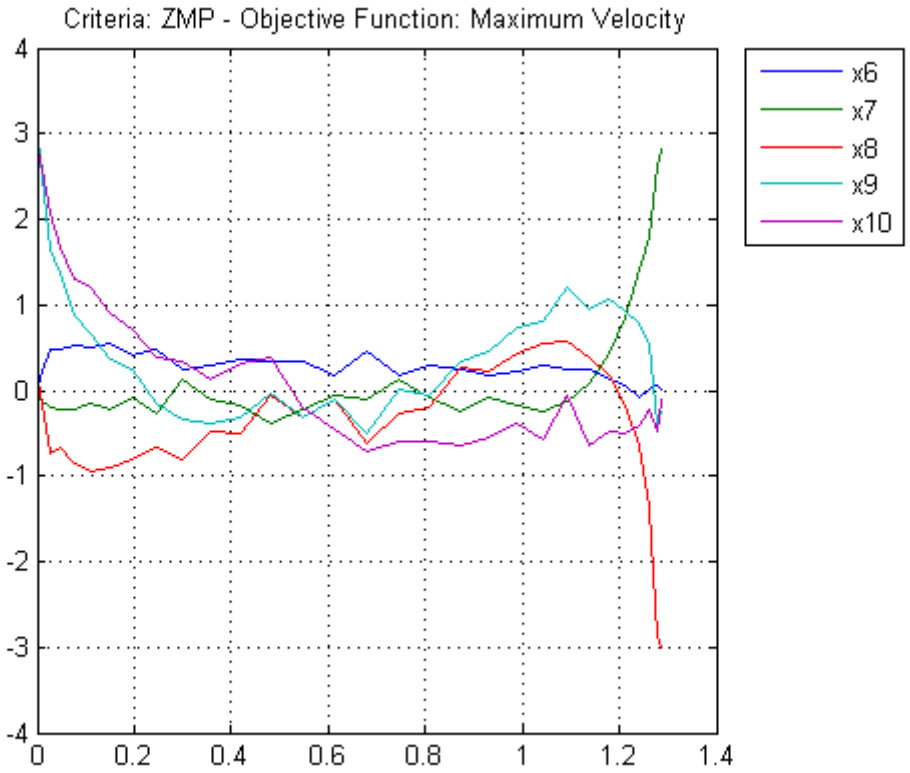

Criteria: COM - Objective Function: Maximum Velocity
200
150
100
50
0
-50
-100
0
0.5
1
1.5
2
u1
u2
u3
u4
u5

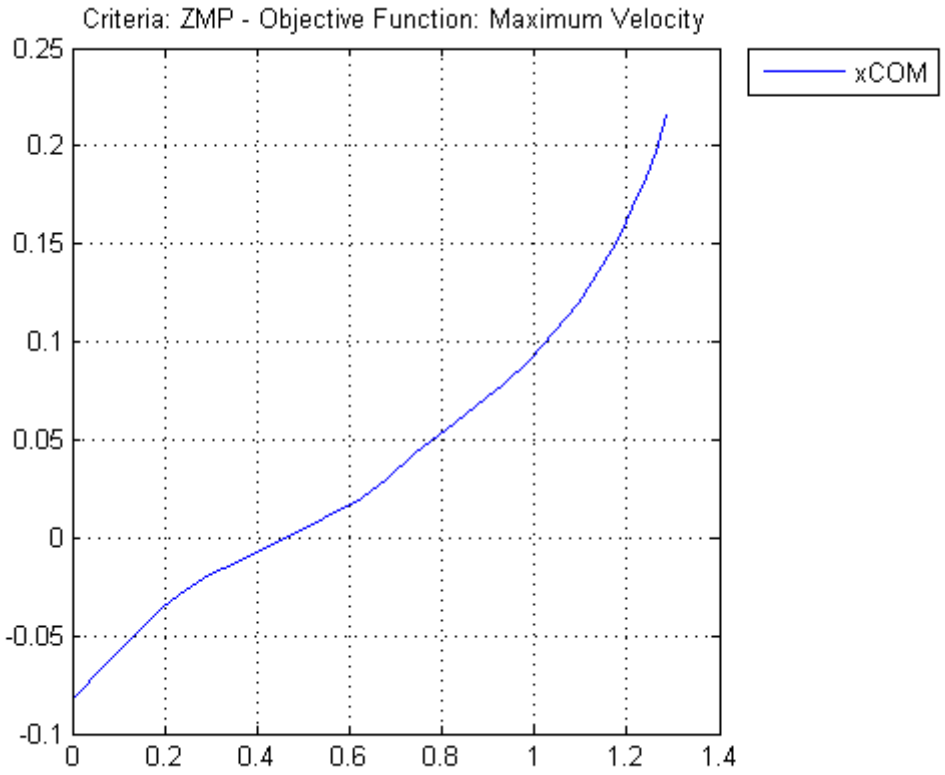

Criteria: COM - Objective Function: Maximum Velocity
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
-0.25
-0.3
0
0.5
1
1.5
2
xZMP

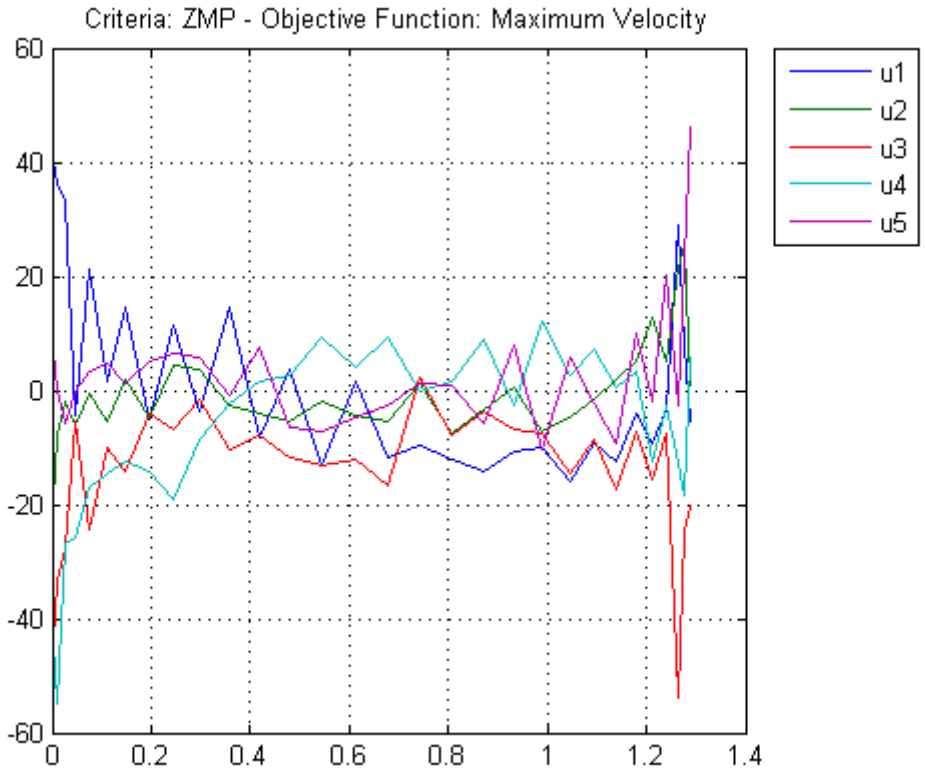


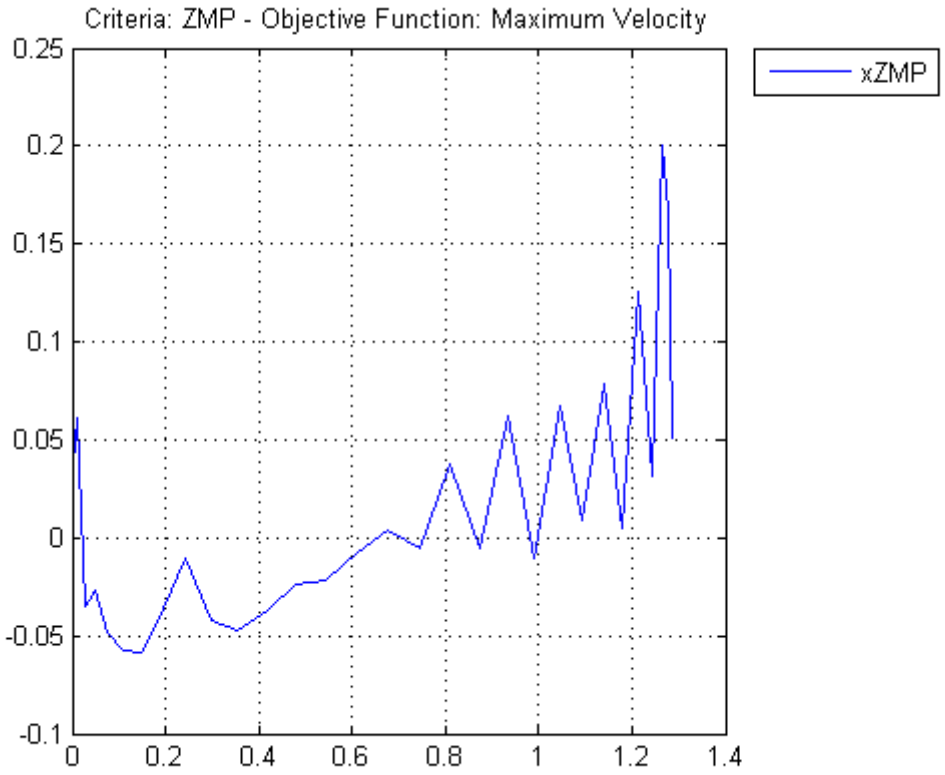
b) Critério: ZMP

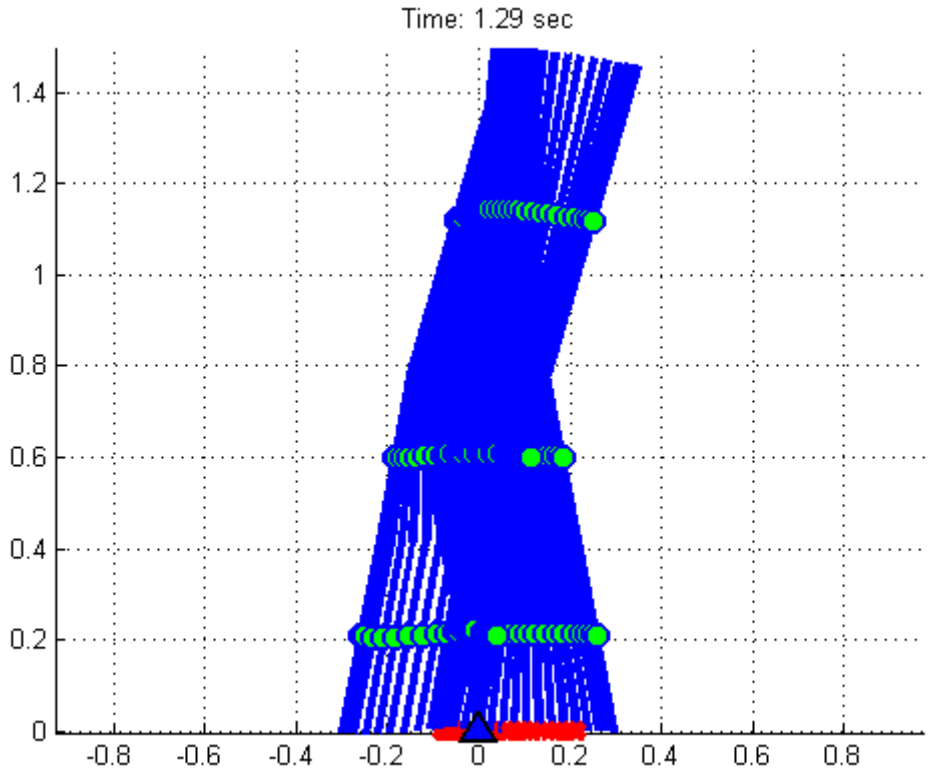



Criteria: ZMP - Objective Function: Maximum Velocity
x6
x7
x8
x9
x10
4
3
2
1
0
-1
-2
-3
-4
0
0.2
0.4
0.6
0.8
1
1.2
1.4

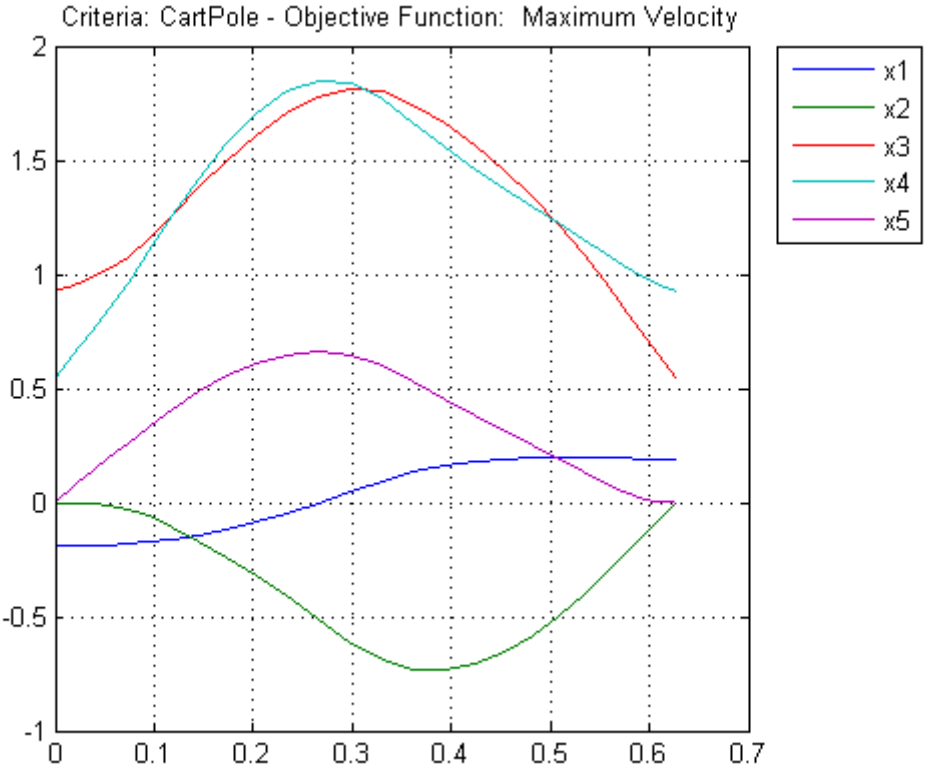

Criteria: ZMP - Objective Function: Maximum Velocity
xCOM
0.25
0.2
0.15
0.1
0.05
0
-0.05
-0.1
0
0.2
0.4
0.6
0.8
1
1.2
1.4

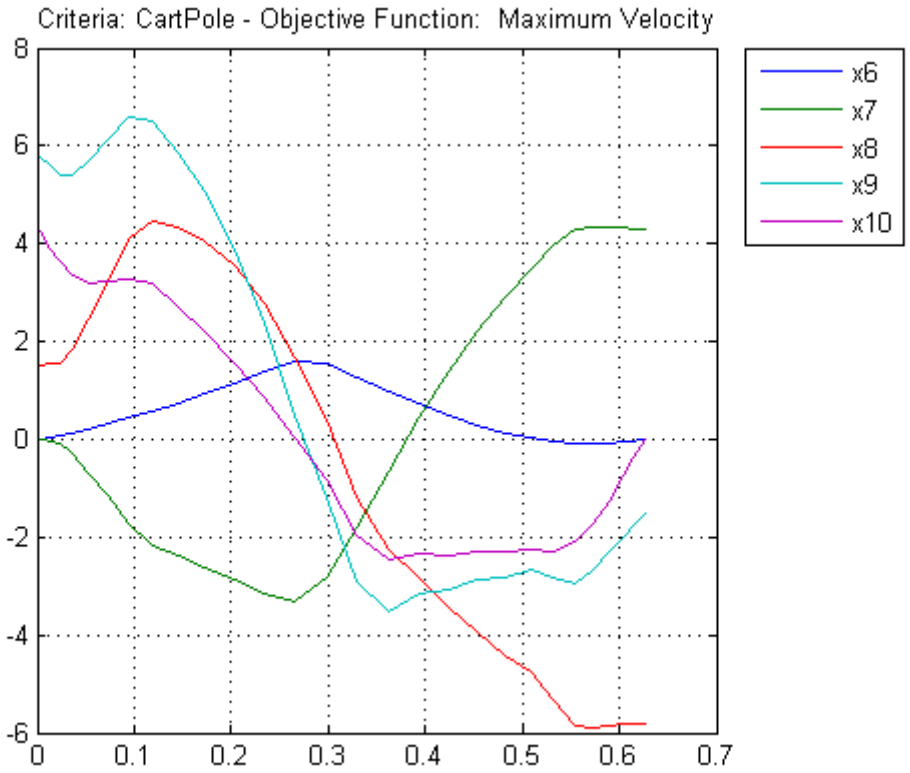

Criteria: ZMP - Objective Function: Maximum Velocity
60
40
20
0
-20
-40
-60
0
0.2
0.4
0.6
0.8
1
1.2
1.4
u1
u2
u3
u4
u5

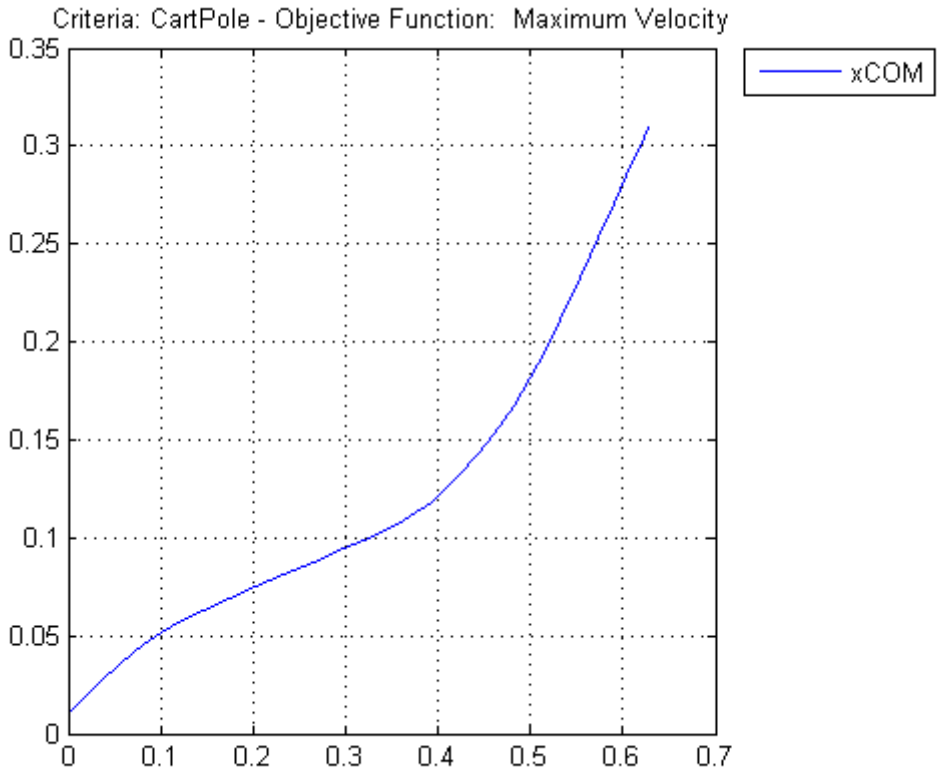

Criteria: ZMP - Objective Function: Maximum Velocity
0.25
0.2
0.15
0.1
0.05
0
-0.05
-0.1
0
0.2
0.4
0.6
0.8
1
1.2
1.4
xZMP

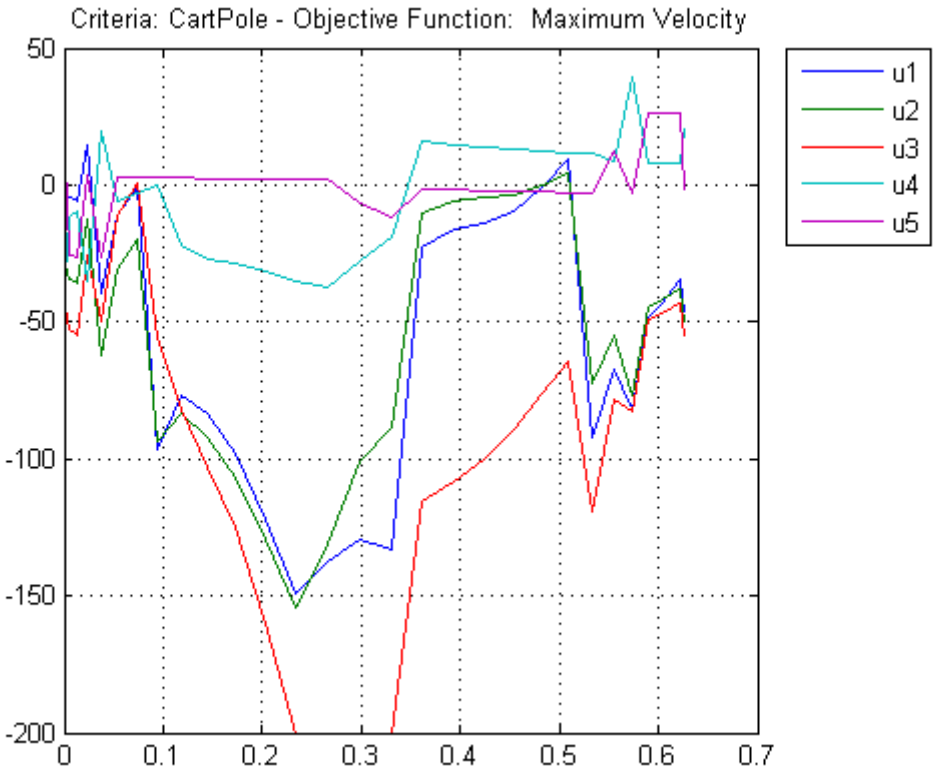


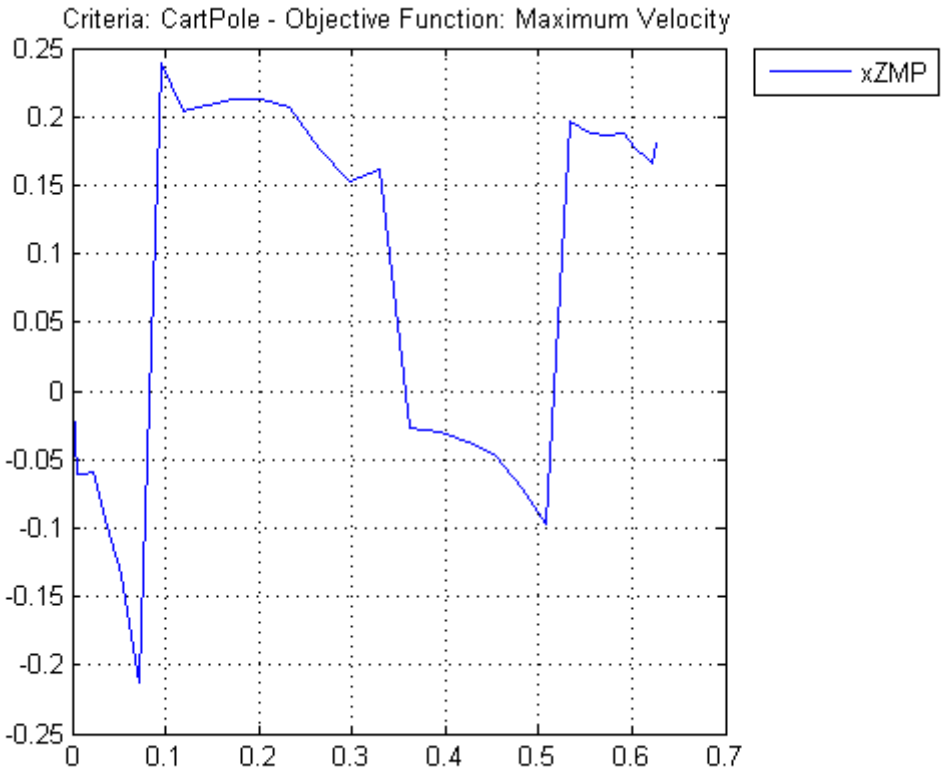
c) Critério: Cart Pole

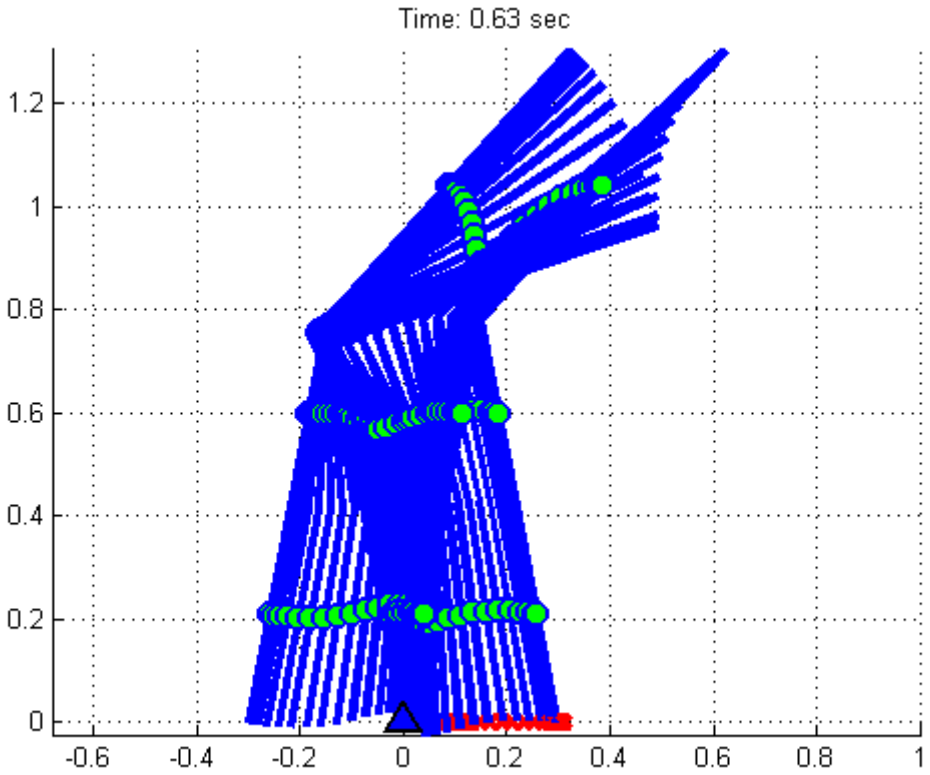



Criteria: CartPole - Objective Function: Maximum Velocity
8
6
4
2
0
-2
-4
-6
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
x6
x7
x8
x9
x10


Criteria: CartPole - Objective Function: Maximum Velocity
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
xCOM


Criteria: CartPole - Objective Function: Maximum Velocity
50
0
-50
-100
-150
-200
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
u1
u2
u3
u4
u5


Criteria: CartPole - Objective Function: Maximum Velocity
0.25
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
-0.25
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
xZMP



8) Conclusão

Através desta iniciação científica foi possível desenvolver habilidades e conhecimento de qualidade nas áreas de Dinâmica de Sistemas Multicorpos e Otimização. Os resultados obtidos satisfazem todas as condições de contorno e minimizam as funções objetivo impostas. Como trabalho futuro, é desejável implementar novos critérios de estabilidade como por exemplo estabilidade através da análise de Ciclos Limite.

Apêndice

Nome do arquivo: runPROPT.m

```
% Computes optimal solution for half of a periodical cycle for a 5-link humanoid robot
% model: five revolute joints liked by five rigid bodies that represent
% legs ( tibia + thigh) and upper body.
% Obs: It is necessary to have the TOMLAB toolbox PROPT for advanced optimal control
applications installed in MATLAB.
% Last modification: 30/03/2014 (by Feliphe G. Galiza)
```

```
close all
clear
clc

% parameters according to human body
height = 1.5; % [m]
L1 = 0.285*height; % [m]
L2 = (0.53 - 0.285)*height; % [m]
L3 = height - (L1 + L2); % [m]
L4 = L2; % [m]
L5 =L1; % [m]

r1 = L1/2; % [m]
r2 = L2/2; % [m]
r3 = L3/2; % [m]
r4 = L4/2; % [m]
r5 = L5/2; % [m]

mass = 50; % [kg]
m1 = 0.061 * mass; % [kg]
m2 = 0.1 * mass; % [kg]
m3 = 0.678 * mass; % [kg]
m4 = m2; % [kg]
m5 = m1; % [kg]
m = [m1 m2 m3 m4 m5] ;
Izz_1 = m1 * 0.735^2; % [kg*m^2]
Izz_2 = m2 * 0.540^2; % [kg*m^2]
Izz_3 = m3 * 0.0798^2; % [kg*m^2]
Izz_4 = Izz_2; % [kg*m^2]
Izz_5 = Izz_1; % [kg*m^2]

Izz = [ Izz_1 Izz_2 Izz_3 Izz_4 Izz_5];

g =9.81; %[m/s^2]

%PROPT

toms t tf
nn = 30; % # nodes
p = tomPhase('p',t,0,tf,nn);
setPhase(p)

tomStates x1 x2 x3 x4 x5 x6 x7 x8 x9 x10
tomControls u1 u2 u3 u4 u5

% State Space Representantion
% where:
% x1_dot = theta1_dot = x6
% x2_dot = theta2_dot = x7
% x3_dot = theta3_dot = x8
% x4_dot = theta4_dot = x9
% x5_dot = theta5_dot = x10
% x6_dot = theta_acc1
% x7_dot = theta_acc2
```

```
% x8_dot = theta_acc3
% x9_dot = theta_acc4
% x10_dot = theta_acc5

q = [x1; x2; x3; x4; x5]; % joint angles
qd = [x6; x7; x8; x9; x10]; % joint velocities
u = [u1; u2; u3; u4; u5]; % joint controls a.k.a. joint torques

% Initial guess for states
x0 = {tf == 2
    icollocate({
     x1 == -0.1894
     x2 == 0
     x3 == 0
     x4 == -0.7227
     x5 == 0
    x6 == 0
    x7 == 0
    x8 == 0
    x9 == 0
    x10 == 0})};

% Initial guess for states
x0 = {x0
    collocate({
    u1 == 0
    u2 == 0
    u3 == 0
    u4 == 0
    u5 == 0})};


% Box constraints
% 1 ft = 0.3 metros
d1=0.05; % forefoot size in [m]
d2=0.25; % heel size in [m]
L = d1 + d2; % foot size in [m]

% pm  is a 2 x 5 array with the joint cartesian position of each body
% pcm is a 2 x 5 array with the Center of Mass position of each body
[pm, pcm] = computeFK(q);

% state space representation
theta1 = x1;
theta2 = x2;
theta3= x3;
theta4 = x4;
theta5 = x5;

theta1_dot = x6;
theta2_dot = x7;
theta3_dot = x8;
theta4_dot  = x9;
theta5_dot = x10;
```

```
% Computing ZMP

% jacobian
J=[

cos(theta1)*(L1 - r1),
0,
0,
0,                                                  0; ...

-sin(theta1)*(L1 - r1),
0,
0,
0,                                                  0; ...

cos(theta1 + theta2)*(L2 - r2) + cos(theta1)*(L1 - r1),
cos(theta1 + theta2)*(L2 - r2),
0,
0,                                                  0; ...

- sin(theta1 + theta2)*(L2 - r2) - sin(theta1)*(L1 - r1),
-sin(theta1 + theta2)*(L2 - r2),
0,
0,                                                  0; ...
                                                              cos(theta1 + theta2)*(L2
- r2) + cos(theta1)*(L1 - r1) + r3*cos(theta1 + theta2 + theta3),
cos(theta1 + theta2)*(L2 - r2) + r3*cos(theta1 + theta2 + theta3),
r3*cos(theta1 + theta2 + theta3),
0,                                                  0; ...
                                                                     - r3*sin(theta1 + theta2 +
theta3) - sin(theta1 + theta2)*(L2 - r2) - sin(theta1)*(L1 - r1),
- r3*sin(theta1 + theta2 + theta3) - sin(theta1 + theta2)*(L2 - r2),
-r3*sin(theta1 + theta2 + theta3),
0,                                                  0; ...
                                                       cos(theta1 + theta2)*(L2 - r2) +
cos(theta1)*(L1 - r1) - r4*cos(theta1 + theta2 + theta3 - theta4),
cos(theta1 + theta2)*(L2 - r2) - r4*cos(theta1 + theta2 + theta3 - theta4),
-r4*cos(theta1 + theta2 + theta3 - theta4),
r4*cos(theta1 + theta2 + theta3 - theta4),
0; ...
                                                       r4*sin(theta1 + theta2 + theta3 -
theta4) - sin(theta1)*(L1 - r1) - sin(theta1 + theta2)*(L2 - r2),
r4*sin(theta1 + theta2 + theta3 - theta4) - sin(theta1 + theta2)*(L2 - r2),
r4*sin(theta1 + theta2 + theta3 - theta4),
-r4*sin(theta1 + theta2 + theta3 - theta4),
0; ...
 cos(theta1 + theta2)*(L2 - r2) - r5*cos(theta1 + theta2 + theta3 - theta4 + theta5) +
cos(theta1)*(L1 - r1) - r4*cos(theta1 + theta2 + theta3 - theta4), cos(theta1 +
theta2)*(L2 - r2) - r5*cos(theta1 + theta2 + theta3 - theta4 + theta5) - r4*cos(theta1 +
theta2 + theta3 - theta4), - r5*cos(theta1 + theta2 + theta3 - theta4 + theta5) -
r4*cos(theta1 + theta2 + theta3 - theta4),   r5*cos(theta1 + theta2 + theta3 - theta4 +
theta5) + r4*cos(theta1 + theta2 + theta3 - theta4), -r5*cos(theta1 + theta2 + theta3 -
theta4 + theta5); ...
```

```
 r5*sin(theta1 + theta2 + theta3 - theta4 + theta5) - sin(theta1 + theta2)*(L2 - r2) -
sin(theta1)*(L1 - r1) + r4*sin(theta1 + theta2 + theta3 - theta4), r5*sin(theta1 +
theta2 + theta3 - theta4 + theta5) - sin(theta1 + theta2)*(L2 - r2) + r4*sin(theta1 +
theta2 + theta3 - theta4),   r5*sin(theta1 + theta2 + theta3 - theta4 + theta5) +
r4*sin(theta1 + theta2 + theta3 - theta4), - r5*sin(theta1 + theta2 + theta3 - theta4 +
theta5) - r4*sin(theta1 + theta2 + theta3 - theta4),  r5*sin(theta1 + theta2 + theta3 -
theta4 + theta5); ...

-1,
0,
0,
0,                                                    0; ...

-1,
-1,
0,
0,                                                    0; ...

-1,
-1,
-1,
0,                                                    0; ...

-1,
-1,
-1,
1,                                                    0; ...

-1,
-1,
-1,
1,                                                   -1];


% jacobian derivative
Jd =[
-theta1_dot*sin(theta1)*(L1 - r1),
0,
0,
0,
0; ...

-theta1_dot*cos(theta1)*(L1 - r1),
0,
0,
0,
0;...

- theta1_dot*(sin(theta1 + theta2)*(L2 - r2) + sin(theta1)*(L1 - r1)) -
theta2_dot*sin(theta1 + theta2)*(L2 - r2),
- theta1_dot*sin(theta1 + theta2)*(L2 - r2) - theta2_dot*sin(theta1 + theta2)*(L2 - r2),
0,
0,
0;...
```

```
- theta1_dot*(cos(theta1 + theta2)*(L2 - r2) + cos(theta1)*(L1 - r1)) -
theta2_dot*cos(theta1 + theta2)*(L2 - r2),
- theta1_dot*cos(theta1 + theta2)*(L2 - r2) - theta2_dot*cos(theta1 + theta2)*(L2 - r2),
0,
0,
0;...

- theta1_dot*(r3*sin(theta1 + theta2 + theta3) + sin(theta1 + theta2)*(L2 - r2) +
sin(theta1)*(L1 - r1)) - theta2_dot*(r3*sin(theta1 + theta2 + theta3) + sin(theta1 +
theta2)*(L2 - r2)) - r3*theta3_dot*sin(theta1 + theta2 + theta3),
- theta1_dot*(r3*sin(theta1 + theta2 + theta3) + sin(theta1 + theta2)*(L2 - r2)) -
theta2_dot*(r3*sin(theta1 + theta2 + theta3) + sin(theta1 + theta2)*(L2 - r2)) -
r3*theta3_dot*sin(theta1 + theta2 + theta3),
- r3*theta1_dot*sin(theta1 + theta2 + theta3) - r3*theta2_dot*sin(theta1 + theta2 +
theta3) - r3*theta3_dot*sin(theta1 + theta2 + theta3),
0,
0;...

- theta1_dot*(cos(theta1 + theta2)*(L2 - r2) + cos(theta1)*(L1 - r1) + r3*cos(theta1 +
theta2 + theta3)) - theta2_dot*(cos(theta1 + theta2)*(L2 - r2) + r3*cos(theta1 + theta2
+ theta3)) - r3*theta3_dot*cos(theta1 + theta2 + theta3),
- theta1_dot*(cos(theta1 + theta2)*(L2 - r2) + r3*cos(theta1 + theta2 + theta3)) -
theta2_dot*(cos(theta1 + theta2)*(L2 - r2) + r3*cos(theta1 + theta2 + theta3)) -
r3*theta3_dot*cos(theta1 + theta2 + theta3),
- r3*theta1_dot*cos(theta1 + theta2 + theta3) - r3*theta2_dot*cos(theta1 + theta2 +
theta3) - r3*theta3_dot*cos(theta1 + theta2 + theta3),
0,
0;...

r4*theta3_dot*sin(theta1 + theta2 + theta3 - theta4) - theta2_dot*(sin(theta1 +
theta2)*(L2 - r2) - r4*sin(theta1 + theta2 + theta3 - theta4)) - theta1_dot*(sin(theta1
+ theta2)*(L2 - r2) + sin(theta1)*(L1 - r1) - r4*sin(theta1 + theta2 + theta3 - theta4))
- r4*theta4_dot*sin(theta1 + theta2 + theta3 - theta4),
r4*theta3_dot*sin(theta1 + theta2 + theta3 - theta4) - theta2_dot*(sin(theta1 +
theta2)*(L2 - r2) - r4*sin(theta1 + theta2 + theta3 - theta4)) - theta1_dot*(sin(theta1
+ theta2)*(L2 - r2) - r4*sin(theta1 + theta2 + theta3 - theta4)) -
r4*theta4_dot*sin(theta1 + theta2 + theta3 - theta4),
r4*theta1_dot*sin(theta1 + theta2 + theta3 - theta4) + r4*theta2_dot*sin(theta1 + theta2
+ theta3 - theta4) + r4*theta3_dot*sin(theta1 + theta2 + theta3 - theta4) -
r4*theta4_dot*sin(theta1 + theta2 + theta3 - theta4),
r4*theta4_dot*sin(theta1 + theta2 + theta3 - theta4) - r4*theta2_dot*sin(theta1 + theta2
+ theta3 - theta4) - r4*theta3_dot*sin(theta1 + theta2 + theta3 - theta4) -
r4*theta1_dot*sin(theta1 + theta2 + theta3 - theta4),
0;...

r4*theta3_dot*cos(theta1 + theta2 + theta3 - theta4) - theta2_dot*(cos(theta1 +
theta2)*(L2 - r2) - r4*cos(theta1 + theta2 + theta3 - theta4)) - theta1_dot*(cos(theta1
+ theta2)*(L2 - r2) + cos(theta1)*(L1 - r1) - r4*cos(theta1 + theta2 + theta3 - theta4))
- r4*theta4_dot*cos(theta1 + theta2 + theta3 - theta4),
r4*theta3_dot*cos(theta1 + theta2 + theta3 - theta4) - theta2_dot*(cos(theta1 +
theta2)*(L2 - r2) - r4*cos(theta1 + theta2 + theta3 - theta4)) - theta1_dot*(cos(theta1
+ theta2)*(L2 - r2) - r4*cos(theta1 + theta2 + theta3 - theta4)) -
r4*theta4_dot*cos(theta1 + theta2 + theta3 - theta4),
```

```
r4*theta1_dot*cos(theta1 + theta2 + theta3 - theta4) + r4*theta2_dot*cos(theta1 + theta2
+ theta3 - theta4) + r4*theta3_dot*cos(theta1 + theta2 + theta3 - theta4) -
r4*theta4_dot*cos(theta1 + theta2 + theta3 - theta4),
r4*theta4_dot*cos(theta1 + theta2 + theta3 - theta4) - r4*theta2_dot*cos(theta1 + theta2
+ theta3 - theta4) - r4*theta3_dot*cos(theta1 + theta2 + theta3 - theta4) -
r4*theta1_dot*cos(theta1 + theta2 + theta3 - theta4),
0;...
 theta3_dot*(r5*sin(theta1 + theta2 + theta3 - theta4 + theta5) + r4*sin(theta1 + theta2
+ theta3 - theta4)) - theta4_dot*(r5*sin(theta1 + theta2 + theta3 - theta4 + theta5) +
r4*sin(theta1 + theta2 + theta3 - theta4)) + theta1_dot*(r5*sin(theta1 + theta2 + theta3
- theta4 + theta5) - sin(theta1 + theta2)*(L2 - r2) - sin(theta1)*(L1 - r1) +
r4*sin(theta1 + theta2 + theta3 - theta4)) + theta2_dot*(r5*sin(theta1 + theta2 + theta3
- theta4 + theta5) - sin(theta1 + theta2)*(L2 - r2) + r4*sin(theta1 + theta2 + theta3 -
theta4)) + r5*theta5_dot*sin(theta1 + theta2 + theta3 - theta4 + theta5),
theta3_dot*(r5*sin(theta1 + theta2 + theta3 - theta4 + theta5) + r4*sin(theta1 + theta2
+ theta3 - theta4)) - theta4_dot*(r5*sin(theta1 + theta2 + theta3 - theta4 + theta5) +
r4*sin(theta1 + theta2 + theta3 - theta4)) + theta1_dot*(r5*sin(theta1 + theta2 + theta3
- theta4 + theta5) - sin(theta1 + theta2)*(L2 - r2) + r4*sin(theta1 + theta2 + theta3 -
theta4)) + theta2_dot*(r5*sin(theta1 + theta2 + theta3 - theta4 + theta5) - sin(theta1 +
theta2)*(L2 - r2) + r4*sin(theta1 + theta2 + theta3 - theta4)) +
r5*theta5_dot*sin(theta1 + theta2 + theta3 - theta4 + theta5), theta1_dot*(r5*sin(theta1
+ theta2 + theta3 - theta4 + theta5) + r4*sin(theta1 + theta2 + theta3 - theta4)) +
theta2_dot*(r5*sin(theta1 + theta2 + theta3 - theta4 + theta5) + r4*sin(theta1 + theta2
+ theta3 - theta4)) + theta3_dot*(r5*sin(theta1 + theta2 + theta3 - theta4 + theta5) +
r4*sin(theta1 + theta2 + theta3 - theta4)) - theta4_dot*(r5*sin(theta1 + theta2 + theta3
- theta4 + theta5) + r4*sin(theta1 + theta2 + theta3 - theta4)) +
r5*theta5_dot*sin(theta1 + theta2 + theta3 - theta4 + theta5), theta4_dot*(r5*sin(theta1
+ theta2 + theta3 - theta4 + theta5) + r4*sin(theta1 + theta2 + theta3 - theta4)) -
theta2_dot*(r5*sin(theta1 + theta2 + theta3 - theta4 + theta5) + r4*sin(theta1 + theta2
+ theta3 - theta4)) - theta3_dot*(r5*sin(theta1 + theta2 + theta3 - theta4 + theta5) +
r4*sin(theta1 + theta2 + theta3 - theta4)) - theta1_dot*(r5*sin(theta1 + theta2 + theta3
- theta4 + theta5) + r4*sin(theta1 + theta2 + theta3 - theta4)) -
r5*theta5_dot*sin(theta1 + theta2 + theta3 - theta4 + theta5), r5*theta1_dot*sin(theta1
+ theta2 + theta3 - theta4 + theta5) + r5*theta2_dot*sin(theta1 + theta2 + theta3 -
theta4 + theta5) + r5*theta3_dot*sin(theta1 + theta2 + theta3 - theta4 + theta5) -
r5*theta4_dot*sin(theta1 + theta2 + theta3 - theta4 + theta5) + r5*theta5_dot*sin(theta1
+ theta2 + theta3 - theta4 + theta5);...
 theta3_dot*(r5*cos(theta1 + theta2 + theta3 - theta4 + theta5) + r4*cos(theta1 + theta2
+ theta3 - theta4)) - theta4_dot*(r5*cos(theta1 + theta2 + theta3 - theta4 + theta5) +
r4*cos(theta1 + theta2 + theta3 - theta4)) + theta1_dot*(r5*cos(theta1 + theta2 + theta3
- theta4 + theta5) - cos(theta1 + theta2)*(L2 - r2) - cos(theta1)*(L1 - r1) +
r4*cos(theta1 + theta2 + theta3 - theta4)) + theta2_dot*(r5*cos(theta1 + theta2 + theta3
- theta4 + theta5) - cos(theta1 + theta2)*(L2 - r2) + r4*cos(theta1 + theta2 + theta3 -
theta4)) + r5*theta5_dot*cos(theta1 + theta2 + theta3 - theta4 + theta5),
theta3_dot*(r5*cos(theta1 + theta2 + theta3 - theta4 + theta5) + r4*cos(theta1 + theta2
+ theta3 - theta4)) - theta4_dot*(r5*cos(theta1 + theta2 + theta3 - theta4 + theta5) +
r4*cos(theta1 + theta2 + theta3 - theta4)) + theta1_dot*(r5*cos(theta1 + theta2 + theta3
- theta4 + theta5) - cos(theta1 + theta2)*(L2 - r2) + r4*cos(theta1 + theta2 + theta3 -
theta4)) + theta2_dot*(r5*cos(theta1 + theta2 + theta3 - theta4 + theta5) - cos(theta1 +
theta2)*(L2 - r2) + r4*cos(theta1 + theta2 + theta3 - theta4)) +
r5*theta5_dot*cos(theta1 + theta2 + theta3 - theta4 + theta5), theta1_dot*(r5*cos(theta1
+ theta2 + theta3 - theta4 + theta5) + r4*cos(theta1 + theta2 + theta3 - theta4)) +
theta2_dot*(r5*cos(theta1 + theta2 + theta3 - theta4 + theta5) + r4*cos(theta1 + theta2
+ theta3 - theta4)) + theta3_dot*(r5*cos(theta1 + theta2 + theta3 - theta4 + theta5) +
```

```
r4*cos(theta1 + theta2 + theta3 - theta4)) - theta4_dot*(r5*cos(theta1 + theta2 + theta3
- theta4 + theta5) + r4*cos(theta1 + theta2 + theta3 - theta4)) +
r5*theta5_dot*cos(theta1 + theta2 + theta3 - theta4 + theta5), theta4_dot*(r5*cos(theta1
+ theta2 + theta3 - theta4 + theta5) + r4*cos(theta1 + theta2 + theta3 - theta4)) -
theta2_dot*(r5*cos(theta1 + theta2 + theta3 - theta4 + theta5) + r4*cos(theta1 + theta2
+ theta3 - theta4)) - theta3_dot*(r5*cos(theta1 + theta2 + theta3 - theta4 + theta5) +
r4*cos(theta1 + theta2 + theta3 - theta4)) - theta1_dot*(r5*cos(theta1 + theta2 + theta3
- theta4 + theta5) + r4*cos(theta1 + theta2 + theta3 - theta4)) -
r5*theta5_dot*cos(theta1 + theta2 + theta3 - theta4 + theta5), r5*theta1_dot*cos(theta1
+ theta2 + theta3 - theta4 + theta5) + r5*theta2_dot*cos(theta1 + theta2 + theta3 -
theta4 + theta5) + r5*theta3_dot*cos(theta1 + theta2 + theta3 - theta4 + theta5) -
r5*theta4_dot*cos(theta1 + theta2 + theta3 - theta4 + theta5) + r5*theta5_dot*cos(theta1
+ theta2 + theta3 - theta4 + theta5);...

0,
0,
0,
0,
0;...

0,
0,
0,
0,
0;...

0,
0,
0,
0,
0;...

0,
0,
0,
0,
0;...

0,
0,
0,
0,
0];



vel = J*qd; % absolut velocity of each body

angvel= [vel(11,1);vel(12,1);vel(13,1);vel(14,1);vel(15,1)]; % absolut angular velocity
of each body

acc = J*dot(qd) +Jd*qd; % absolute acceleration of each body

x_acc = [acc(1); acc(3); acc(5); acc(7); acc(9)]; % acceleration in x direction
```

```
y_acc = [acc(2); acc(4); acc(6); acc(8); acc(10)]; % acceleration in y direction

 xZMP_num = 0;
 xZMP_den = 0;
 for i=1:5
     xZMP_num = xZMP_num + m(i)*(y_acc(i)+g)*pcm(1,i) - m(i)*x_acc(i)*pcm(2,i) -
Izz(i)*angvel(i);
     xZMP_den = xZMP_den + m(i)*(y_acc(i)+g);
 end
 xZMP = xZMP_num/xZMP_den;

%xCOM = computeCM(pcm(1,:));

%xZMP = computeCM(pcm(1,:)) - x_acc*computeCM(pcm(2,:))/g;

cbox = { 0.1 <= tf <= 10
    -pi <= icollocate(x1) <= pi
    -pi <= icollocate(x3) <= pi
    -pi <= icollocate(x4) <= pi
    -pi <= icollocate(x2) <= 0
     0 <= icollocate(x5) <= pi
    -10*pi <= icollocate(qd) <= 10*pi
    -d1 <= icollocate(xZMP) <= d2
    -200 <= collocate(u1) <= 200
    -200 <= collocate(u2) <= 200
    -200 <= collocate(u3) <= 200
    -200 <= collocate(u4) <= 200
    -200 <= collocate(u5) <= 200};

% 5th-link foot position
x_m5 = pm(1,5);
y_m5 = pm(2,5);

% Boundary constraints
cbnd = {final(x5) == initial(x2)
    final(x2) == initial(x5)
    final(x4) ==  initial(x3)
    final(x3) ==  initial(x4)
    final(x1) == (initial(x1) + initial(x2) + initial(x3) - initial(x4) + initial(x5))
    initial(x1) == (final(x1) + final(x2) + final(x3) - final(x4) + final(x5))
    initial(x_m5) == -L
    initial(y_m5) == 0
    initial(xZMP) == -d1
    final(xZMP) == d2
    final(x10) == initial(x7)
    final(x7) == initial(x10)
    final(x9) == -initial(x8)
    final(x8) == -initial(x9)
    final(x6) == initial(x6) + initial(x7) + initial(x8) - initial(x9) + initial(x10)
    initial(x6) == (final(x6) + final(x7) + final(x8) - final(x9) + final(x10))
    initial(x6) == 0
    final(x10) == 0};
```

```
% Equation of Motion in the form M*dot(qd) = RHS
% Using Newton-Euler
M(1,:) = [ Izz_1 + Izz_2 + Izz_3 + Izz_4 + Izz_5 + m5*(r5*sin(theta1 + theta2 + theta3 -
theta4 + theta5) - sin(theta1 + theta2)*(L2 - r2) - sin(theta1)*(L1 - r1) +
r4*sin(theta1 + theta2 + theta3 - theta4))^2 + m4*(cos(theta1 + theta2)*(L2 - r2) +
cos(theta1)*(L1 - r1) - r4*cos(theta1 + theta2 + theta3 - theta4))^2 + m3*(cos(theta1 +
theta2)*(L2 - r2) + cos(theta1)*(L1 - r1) + r3*cos(theta1 + theta2 + theta3))^2 +
m4*(sin(theta1 + theta2)*(L2 - r2) + sin(theta1)*(L1 - r1) - r4*sin(theta1 + theta2 +
theta3 - theta4))^2 + m5*(r5*cos(theta1 + theta2 + theta3 - theta4 + theta5) -
cos(theta1 + theta2)*(L2 - r2) - cos(theta1)*(L1 - r1) + r4*cos(theta1 + theta2 + theta3
- theta4))^2 + m2*(cos(theta1 + theta2)*(L2 - r2) + cos(theta1)*(L1 - r1))^2 +
m2*(sin(theta1 + theta2)*(L2 - r2) + sin(theta1)*(L1 - r1))^2 + m3*(r3*sin(theta1 +
theta2 + theta3) + sin(theta1 + theta2)*(L2 - r2) + sin(theta1)*(L1 - r1))^2 +
m1*cos(theta1)^2*(L1 - r1)^2 + m1*sin(theta1)^2*(L1 - r1)^2, Izz_2 + Izz_3 + Izz_4 +
Izz_5 + L2^2*m2 + L2^2*m3 + L2^2*m4 + L2^2*m5 + m2*r2^2 + m3*r2^2 + m3*r3^2 + m4*r2^2 +
m5*r2^2 + m4*r4^2 + m5*r4^2 + m5*r5^2 - 2*L2*m2*r2 - 2*L2*m3*r2 - 2*L2*m4*r2 -
2*L2*m5*r2 - L1*m4*r4*cos(theta2 + theta3 - theta4) - L1*m5*r4*cos(theta2 + theta3 -
theta4) - 2*L2*m5*r5*cos(theta3 - theta4 + theta5) + m4*r1*r4*cos(theta2 + theta3 -
theta4) + m5*r1*r4*cos(theta2 + theta3 - theta4) + 2*m5*r2*r5*cos(theta3 - theta4 +
theta5) + L1*m3*r3*cos(theta2 + theta3) + L1*L2*m2*cos(theta2) + L1*L2*m3*cos(theta2) +
L1*L2*m4*cos(theta2) + L1*L2*m5*cos(theta2) - m3*r1*r3*cos(theta2 + theta3) -
L1*m2*r2*cos(theta2) - L2*m2*r1*cos(theta2) - L1*m3*r2*cos(theta2) -
L2*m3*r1*cos(theta2) - L1*m4*r2*cos(theta2) - L2*m4*r1*cos(theta2) -
L1*m5*r2*cos(theta2) - L2*m5*r1*cos(theta2) + 2*L2*m3*r3*cos(theta3) -
L1*m5*r5*cos(theta2 + theta3 - theta4 + theta5) + m2*r1*r2*cos(theta2) +
m3*r1*r2*cos(theta2) + m4*r1*r2*cos(theta2) + m5*r1*r2*cos(theta2) -
2*m3*r2*r3*cos(theta3) + 2*m5*r4*r5*cos(theta5) - 2*L2*m4*r4*cos(theta3 - theta4) -
2*L2*m5*r4*cos(theta3 - theta4) + m5*r1*r5*cos(theta2 + theta3 - theta4 + theta5) +
2*m4*r2*r4*cos(theta3 - theta4) + 2*m5*r2*r4*cos(theta3 - theta4), Izz_3 + Izz_4 + Izz_5
+ m3*r3^2 + m4*r4^2 + m5*r4^2 + m5*r5^2 - L1*m4*r4*cos(theta2 + theta3 - theta4) -
L1*m5*r4*cos(theta2 + theta3 - theta4) - L2*m5*r5*cos(theta3 - theta4 + theta5) +
m4*r1*r4*cos(theta2 + theta3 - theta4) + m5*r1*r4*cos(theta2 + theta3 - theta4) +
m5*r2*r5*cos(theta3 - theta4 + theta5) + L1*m3*r3*cos(theta2 + theta3) -
m3*r1*r3*cos(theta2 + theta3) + L2*m3*r3*cos(theta3) - L1*m5*r5*cos(theta2 + theta3 -
theta4 + theta5) - m3*r2*r3*cos(theta3) + 2*m5*r4*r5*cos(theta5) - L2*m4*r4*cos(theta3 -
theta4) - L2*m5*r4*cos(theta3 - theta4) + m5*r1*r5*cos(theta2 + theta3 - theta4 +
theta5) + m4*r2*r4*cos(theta3 - theta4) + m5*r2*r4*cos(theta3 - theta4),
L1*m4*r4*cos(theta2 + theta3 - theta4) - Izz_5 - m4*r4^2 - m5*r4^2 - m5*r5^2 - Izz_4 +
L1*m5*r4*cos(theta2 + theta3 - theta4) + L2*m5*r5*cos(theta3 - theta4 + theta5) -
m4*r1*r4*cos(theta2 + theta3 - theta4) - m5*r1*r4*cos(theta2 + theta3 - theta4) -
m5*r2*r5*cos(theta3 - theta4 + theta5) + L1*m5*r5*cos(theta2 + theta3 - theta4 + theta5)
- 2*m5*r4*r5*cos(theta5) + L2*m4*r4*cos(theta3 - theta4) + L2*m5*r4*cos(theta3 - theta4)
- m5*r1*r5*cos(theta2 + theta3 - theta4 + theta5) - m4*r2*r4*cos(theta3 - theta4) -
m5*r2*r4*cos(theta3 - theta4), Izz_5 + m5*r5^2 - L2*m5*r5*cos(theta3 - theta4 + theta5)
+ m5*r2*r5*cos(theta3 - theta4 + theta5) - L1*m5*r5*cos(theta2 + theta3 - theta4 +
theta5) + m5*r4*r5*cos(theta5) + m5*r1*r5*cos(theta2 + theta3 - theta4 + theta5)];
M(2,:) = [ Izz_2 + Izz_3 + Izz_4 + Izz_5 + L2^2*m2 + L2^2*m3 + L2^2*m4 + L2^2*m5 +
m2*r2^2 + m3*r2^2 + m3*r3^2 + m4*r2^2 + m5*r2^2 + m4*r4^2 + m5*r4^2 + m5*r5^2 -
2*L2*m2*r2 - 2*L2*m3*r2 - 2*L2*m4*r2 - 2*L2*m5*r2 - L1*m4*r4*cos(theta2 + theta3 -
theta4) - L1*m5*r4*cos(theta2 + theta3 - theta4) - 2*L2*m5*r5*cos(theta3 - theta4 +
theta5) + m4*r1*r4*cos(theta2 + theta3 - theta4) + m5*r1*r4*cos(theta2 + theta3 -
theta4) + 2*m5*r2*r5*cos(theta3 - theta4 + theta5) + L1*m3*r3*cos(theta2 + theta3) +
L1*L2*m2*cos(theta2) + L1*L2*m3*cos(theta2) + L1*L2*m4*cos(theta2) +
L1*L2*m5*cos(theta2) - m3*r1*r3*cos(theta2 + theta3) - L1*m2*r2*cos(theta2) -
```

```
L2*m2*r1*cos(theta2) - L1*m3*r2*cos(theta2) - L2*m3*r1*cos(theta2) -
L1*m4*r2*cos(theta2) - L2*m4*r1*cos(theta2) - L1*m5*r2*cos(theta2) -
L2*m5*r1*cos(theta2) + 2*L2*m3*r3*cos(theta3) - L1*m5*r5*cos(theta2 + theta3 - theta4 +
theta5) + m2*r1*r2*cos(theta2) + m3*r1*r2*cos(theta2) + m4*r1*r2*cos(theta2) +
m5*r1*r2*cos(theta2) - 2*m3*r2*r3*cos(theta3) + 2*m5*r4*r5*cos(theta5) -
2*L2*m4*r4*cos(theta3 - theta4) - 2*L2*m5*r4*cos(theta3 - theta4) + m5*r1*r5*cos(theta2
+ theta3 - theta4 + theta5) + 2*m4*r2*r4*cos(theta3 - theta4) + 2*m5*r2*r4*cos(theta3 -
theta4), Izz_2 + Izz_3 + Izz_4 + Izz_5 + L2^2*m2 + L2^2*m3 + L2^2*m4 + L2^2*m5 + m2*r2^2
+ m3*r2^2 + m3*r3^2 + m4*r2^2 + m5*r2^2 + m4*r4^2 + m5*r4^2 + m5*r5^2 - 2*L2*m2*r2 -
2*L2*m3*r2 - 2*L2*m4*r2 - 2*L2*m5*r2 - 2*L2*m5*r5*cos(theta3 - theta4 + theta5) +
2*m5*r2*r5*cos(theta3 - theta4 + theta5) + 2*L2*m3*r3*cos(theta3) -
2*m3*r2*r3*cos(theta3) + 2*m5*r4*r5*cos(theta5) - 2*L2*m4*r4*cos(theta3 - theta4) -
2*L2*m5*r4*cos(theta3 - theta4) + 2*m4*r2*r4*cos(theta3 - theta4) +
2*m5*r2*r4*cos(theta3 - theta4), Izz_3 + Izz_4 + Izz_5 + m3*r3^2 + m4*r4^2 + m5*r4^2 +
m5*r5^2 - L2*m5*r5*cos(theta3 - theta4 + theta5) + m5*r2*r5*cos(theta3 - theta4 +
theta5) + L2*m3*r3*cos(theta3) - m3*r2*r3*cos(theta3) + 2*m5*r4*r5*cos(theta5) -
L2*m4*r4*cos(theta3 - theta4) - L2*m5*r4*cos(theta3 - theta4) + m4*r2*r4*cos(theta3 -
theta4) + m5*r2*r4*cos(theta3 - theta4), L2*m5*r5*cos(theta3 - theta4 + theta5) - Izz_5
- m4*r4^2 - m5*r4^2 - m5*r5^2 - Izz_4 - m5*r2*r5*cos(theta3 - theta4 + theta5) -
2*m5*r4*r5*cos(theta5) + L2*m4*r4*cos(theta3 - theta4) + L2*m5*r4*cos(theta3 - theta4) -
m4*r2*r4*cos(theta3 - theta4) - m5*r2*r4*cos(theta3 - theta4), Izz_5 + m5*r5^2 -
L2*m5*r5*cos(theta3 - theta4 + theta5) + m5*r2*r5*cos(theta3 - theta4 + theta5) +
m5*r4*r5*cos(theta5)];
M(3,:) = [ Izz_3 + Izz_4 + Izz_5 + m3*r3^2 + m4*r4^2 + m5*r4^2 + m5*r5^2 -
L1*m4*r4*cos(theta2 + theta3 - theta4) - L1*m5*r4*cos(theta2 + theta3 - theta4) -
L2*m5*r5*cos(theta3 - theta4 + theta5) + m4*r1*r4*cos(theta2 + theta3 - theta4) +
m5*r1*r4*cos(theta2 + theta3 - theta4) + m5*r2*r5*cos(theta3 - theta4 + theta5) +
L1*m3*r3*cos(theta2 + theta3) - m3*r1*r3*cos(theta2 + theta3) + L2*m3*r3*cos(theta3) -
L1*m5*r5*cos(theta2 + theta3 - theta4 + theta5) - m3*r2*r3*cos(theta3) +
2*m5*r4*r5*cos(theta5) - L2*m4*r4*cos(theta3 - theta4) - L2*m5*r4*cos(theta3 - theta4) +
m5*r1*r5*cos(theta2 + theta3 - theta4 + theta5) + m4*r2*r4*cos(theta3 - theta4) +
m5*r2*r4*cos(theta3 - theta4), Izz_3 + Izz_4 + Izz_5 + m3*r3^2 + m4*r4^2 + m5*r4^2 +
m5*r5^2 - L2*m5*r5*cos(theta3 - theta4 + theta5) + m5*r2*r5*cos(theta3 - theta4 +
theta5) + L2*m3*r3*cos(theta3) - m3*r2*r3*cos(theta3) + 2*m5*r4*r5*cos(theta5) -
L2*m4*r4*cos(theta3 - theta4) - L2*m5*r4*cos(theta3 - theta4) + m4*r2*r4*cos(theta3 -
theta4) + m5*r2*r4*cos(theta3 - theta4), Izz_3 + Izz_4 + Izz_5 + m3*r3^2 + m4*r4^2 +
m5*r4^2 + m5*r5^2 + 2*m5*r4*r5*cos(theta5), - Izz_4 - Izz_5 - m4*r4^2 - m5*r4^2 -
m5*r5^2 - 2*m5*r4*r5*cos(theta5), m5*r5^2 + m5*r4*cos(theta5)*r5 + Izz_5];
M(4,:) = [ L1*m4*r4*cos(theta2 + theta3 - theta4) - Izz_5 - m4*r4^2 - m5*r4^2 - m5*r5^2
- Izz_4 + L1*m5*r4*cos(theta2 + theta3 - theta4) + L2*m5*r5*cos(theta3 - theta4 +
theta5) - m4*r1*r4*cos(theta2 + theta3 - theta4) - m5*r1*r4*cos(theta2 + theta3 -
theta4) - m5*r2*r5*cos(theta3 - theta4 + theta5) + L1*m5*r5*cos(theta2 + theta3 - theta4
+ theta5) - 2*m5*r4*r5*cos(theta5) + L2*m4*r4*cos(theta3 - theta4) + L2*m5*r4*cos(theta3
- theta4) - m5*r1*r5*cos(theta2 + theta3 - theta4 + theta5) - m4*r2*r4*cos(theta3 -
theta4) - m5*r2*r4*cos(theta3 - theta4), L2*m5*r5*cos(theta3 - theta4 + theta5) - Izz_5
- m4*r4^2 - m5*r4^2 - m5*r5^2 - Izz_4 - m5*r2*r5*cos(theta3 - theta4 + theta5) -
2*m5*r4*r5*cos(theta5) + L2*m4*r4*cos(theta3 - theta4) + L2*m5*r4*cos(theta3 - theta4) -
m4*r2*r4*cos(theta3 - theta4) - m5*r2*r4*cos(theta3 - theta4), - Izz_4 - Izz_5 - m4*r4^2
- m5*r4^2 - m5*r5^2 - 2*m5*r4*r5*cos(theta5), Izz_4 + Izz_5 + m4*r4^2 + m5*r4^2 +
m5*r5^2 + 2*m5*r4*r5*cos(theta5), - m5*r5^2 - m5*r4*cos(theta5)*r5 - Izz_5];
M(5,:) = [ Izz_5 + m5*r5^2 - L2*m5*r5*cos(theta3 - theta4 + theta5) +
m5*r2*r5*cos(theta3 - theta4 + theta5) - L1*m5*r5*cos(theta2 + theta3 - theta4 + theta5)
+ m5*r4*r5*cos(theta5) + m5*r1*r5*cos(theta2 + theta3 - theta4 + theta5), Izz_5 +
m5*r5^2 - L2*m5*r5*cos(theta3 - theta4 + theta5) + m5*r2*r5*cos(theta3 - theta4 +
```

```
theta5) + m5*r4*r5*cos(theta5), m5*r5^2 + m5*r4*cos(theta5)*r5 + Izz_5, - m5*r5^2 -
m5*r4*cos(theta5)*r5 - Izz_5, m5*r5^2 + Izz_5];

RHS(1,1) = u1 - g*m5*(r5*sin(theta1 + theta2 + theta3 - theta4 + theta5) - sin(theta1 +
theta2)*(L2 - r2) - sin(theta1)*(L1 - r1) + r4*sin(theta1 + theta2 + theta3 - theta4)) +
g*m4*(sin(theta1 + theta2)*(L2 - r2) + sin(theta1)*(L1 - r1) - r4*sin(theta1 + theta2 +
theta3 - theta4)) + g*m2*(sin(theta1 + theta2)*(L2 - r2) + sin(theta1)*(L1 - r1)) +
g*m3*(r3*sin(theta1 + theta2 + theta3) + sin(theta1 + theta2)*(L2 - r2) +
sin(theta1)*(L1 - r1)) + g*m1*sin(theta1)*(L1 - r1) + m4*r1*r4*theta2_dot^2*sin(theta2 +
theta3 - theta4) + m4*r1*r4*theta3_dot^2*sin(theta2 + theta3 - theta4) +
m5*r1*r4*theta2_dot^2*sin(theta2 + theta3 - theta4) + m4*r1*r4*theta4_dot^2*sin(theta2 +
theta3 - theta4) + m5*r1*r4*theta3_dot^2*sin(theta2 + theta3 - theta4) +
m5*r1*r4*theta4_dot^2*sin(theta2 + theta3 - theta4) + m5*r2*r5*theta3_dot^2*sin(theta3 -
theta4 + theta5) + m5*r2*r5*theta4_dot^2*sin(theta3 - theta4 + theta5) +
m5*r2*r5*theta5_dot^2*sin(theta3 - theta4 + theta5) + L1*m3*r3*theta2_dot^2*sin(theta2 +
theta3) + L1*m3*r3*theta3_dot^2*sin(theta2 + theta3) + L1*L2*m2*theta2_dot^2*sin(theta2)
+ L1*L2*m3*theta2_dot^2*sin(theta2) + L1*L2*m4*theta2_dot^2*sin(theta2) +
L1*L2*m5*theta2_dot^2*sin(theta2) - m3*r1*r3*theta2_dot^2*sin(theta2 + theta3) -
m3*r1*r3*theta3_dot^2*sin(theta2 + theta3) - L1*m2*r2*theta2_dot^2*sin(theta2) -
L2*m2*r1*theta2_dot^2*sin(theta2) - L1*m3*r2*theta2_dot^2*sin(theta2) -
L2*m3*r1*theta2_dot^2*sin(theta2) - L1*m4*r2*theta2_dot^2*sin(theta2) -
L2*m4*r1*theta2_dot^2*sin(theta2) - L1*m5*r2*theta2_dot^2*sin(theta2) -
L2*m5*r1*theta2_dot^2*sin(theta2) + L2*m3*r3*theta3_dot^2*sin(theta3) -
L1*m5*r5*theta2_dot^2*sin(theta2 + theta3 - theta4 + theta5) -
L1*m5*r5*theta3_dot^2*sin(theta2 + theta3 - theta4 + theta5) -
L1*m5*r5*theta4_dot^2*sin(theta2 + theta3 - theta4 + theta5) -
L1*m5*r5*theta5_dot^2*sin(theta2 + theta3 - theta4 + theta5) +
m2*r1*r2*theta2_dot^2*sin(theta2) + m3*r1*r2*theta2_dot^2*sin(theta2) +
m4*r1*r2*theta2_dot^2*sin(theta2) + m5*r1*r2*theta2_dot^2*sin(theta2) -
m3*r2*r3*theta3_dot^2*sin(theta3) + m5*r4*r5*theta5_dot^2*sin(theta5) -
L2*m4*r4*theta3_dot^2*sin(theta3 - theta4) - L2*m4*r4*theta4_dot^2*sin(theta3 - theta4)
- L2*m5*r4*theta3_dot^2*sin(theta3 - theta4) - L2*m5*r4*theta4_dot^2*sin(theta3 -
theta4) + m5*r1*r5*theta2_dot^2*sin(theta2 + theta3 - theta4 + theta5) +
m5*r1*r5*theta3_dot^2*sin(theta2 + theta3 - theta4 + theta5) +
m5*r1*r5*theta4_dot^2*sin(theta2 + theta3 - theta4 + theta5) +
m5*r1*r5*theta5_dot^2*sin(theta2 + theta3 - theta4 + theta5) +
m4*r2*r4*theta3_dot^2*sin(theta3 - theta4) + m4*r2*r4*theta4_dot^2*sin(theta3 - theta4)
+ m5*r2*r4*theta3_dot^2*sin(theta3 - theta4) + m5*r2*r4*theta4_dot^2*sin(theta3 -
theta4) - L1*m4*r4*theta2_dot^2*sin(theta2 + theta3 - theta4) -
L1*m4*r4*theta3_dot^2*sin(theta2 + theta3 - theta4) - L1*m5*r4*theta2_dot^2*sin(theta2 +
theta3 - theta4) - L1*m4*r4*theta4_dot^2*sin(theta2 + theta3 - theta4) -
L1*m5*r4*theta3_dot^2*sin(theta2 + theta3 - theta4) - L1*m5*r4*theta4_dot^2*sin(theta2 +
theta3 - theta4) - L2*m5*r5*theta3_dot^2*sin(theta3 - theta4 + theta5) -
L2*m5*r5*theta4_dot^2*sin(theta3 - theta4 + theta5) - L2*m5*r5*theta5_dot^2*sin(theta3 -
theta4 + theta5) - 2*L1*m4*r4*theta1_dot*theta2_dot*sin(theta2 + theta3 - theta4) -
2*L1*m4*r4*theta1_dot*theta3_dot*sin(theta2 + theta3 - theta4) -
2*L1*m5*r4*theta1_dot*theta2_dot*sin(theta2 + theta3 - theta4) +
2*L1*m4*r4*theta1_dot*theta4_dot*sin(theta2 + theta3 - theta4) -
2*L1*m4*r4*theta2_dot*theta3_dot*sin(theta2 + theta3 - theta4) -
2*L1*m5*r4*theta1_dot*theta3_dot*sin(theta2 + theta3 - theta4) +
2*L1*m4*r4*theta2_dot*theta4_dot*sin(theta2 + theta3 - theta4) +
2*L1*m5*r4*theta1_dot*theta4_dot*sin(theta2 + theta3 - theta4) -
2*L1*m5*r4*theta2_dot*theta3_dot*sin(theta2 + theta3 - theta4) +
2*L1*m4*r4*theta3_dot*theta4_dot*sin(theta2 + theta3 - theta4) +
```

```
2*L1*m5*r4*theta2_dot*theta4_dot*sin(theta2 + theta3 - theta4) +
2*L1*m5*r4*theta3_dot*theta4_dot*sin(theta2 + theta3 - theta4) -
2*L2*m5*r5*theta1_dot*theta3_dot*sin(theta3 - theta4 + theta5) +
2*L2*m5*r5*theta1_dot*theta4_dot*sin(theta3 - theta4 + theta5) -
2*L2*m5*r5*theta2_dot*theta3_dot*sin(theta3 - theta4 + theta5) -
2*L2*m5*r5*theta1_dot*theta5_dot*sin(theta3 - theta4 + theta5) +
2*L2*m5*r5*theta2_dot*theta4_dot*sin(theta3 - theta4 + theta5) -
2*L2*m5*r5*theta2_dot*theta5_dot*sin(theta3 - theta4 + theta5) +
2*L2*m5*r5*theta3_dot*theta4_dot*sin(theta3 - theta4 + theta5) -
2*L2*m5*r5*theta3_dot*theta5_dot*sin(theta3 - theta4 + theta5) +
2*L2*m5*r5*theta4_dot*theta5_dot*sin(theta3 - theta4 + theta5) +
2*m4*r1*r4*theta1_dot*theta2_dot*sin(theta2 + theta3 - theta4) +
2*m4*r1*r4*theta1_dot*theta3_dot*sin(theta2 + theta3 - theta4) +
2*m5*r1*r4*theta1_dot*theta2_dot*sin(theta2 + theta3 - theta4) -
2*m4*r1*r4*theta1_dot*theta4_dot*sin(theta2 + theta3 - theta4) +
2*m4*r1*r4*theta2_dot*theta3_dot*sin(theta2 + theta3 - theta4) +
2*m5*r1*r4*theta1_dot*theta3_dot*sin(theta2 + theta3 - theta4) -
2*m4*r1*r4*theta2_dot*theta4_dot*sin(theta2 + theta3 - theta4) -
2*m5*r1*r4*theta1_dot*theta4_dot*sin(theta2 + theta3 - theta4) +
2*m5*r1*r4*theta2_dot*theta3_dot*sin(theta2 + theta3 - theta4) -
2*m4*r1*r4*theta3_dot*theta4_dot*sin(theta2 + theta3 - theta4) -
2*m5*r1*r4*theta2_dot*theta4_dot*sin(theta2 + theta3 - theta4) -
2*m5*r1*r4*theta3_dot*theta4_dot*sin(theta2 + theta3 - theta4) +
2*m5*r2*r5*theta1_dot*theta3_dot*sin(theta3 - theta4 + theta5) -
2*m5*r2*r5*theta1_dot*theta4_dot*sin(theta3 - theta4 + theta5) +
2*m5*r2*r5*theta2_dot*theta3_dot*sin(theta3 - theta4 + theta5) +
2*m5*r2*r5*theta1_dot*theta5_dot*sin(theta3 - theta4 + theta5) -
2*m5*r2*r5*theta2_dot*theta4_dot*sin(theta3 - theta4 + theta5) +
2*m5*r2*r5*theta2_dot*theta5_dot*sin(theta3 - theta4 + theta5) -
2*m5*r2*r5*theta3_dot*theta4_dot*sin(theta3 - theta4 + theta5) +
2*m5*r2*r5*theta3_dot*theta5_dot*sin(theta3 - theta4 + theta5) -
2*m5*r2*r5*theta4_dot*theta5_dot*sin(theta3 - theta4 + theta5) +
2*L1*m3*r3*theta1_dot*theta2_dot*sin(theta2 + theta3) +
2*L1*m3*r3*theta1_dot*theta3_dot*sin(theta2 + theta3) +
2*L1*m3*r3*theta2_dot*theta3_dot*sin(theta2 + theta3) +
2*L1*L2*m2*theta1_dot*theta2_dot*sin(theta2) +
2*L1*L2*m3*theta1_dot*theta2_dot*sin(theta2) +
2*L1*L2*m4*theta1_dot*theta2_dot*sin(theta2) +
2*L1*L2*m5*theta1_dot*theta2_dot*sin(theta2) -
2*m3*r1*r3*theta1_dot*theta2_dot*sin(theta2 + theta3) -
2*m3*r1*r3*theta1_dot*theta3_dot*sin(theta2 + theta3) -
2*m3*r1*r3*theta2_dot*theta3_dot*sin(theta2 + theta3) -
2*L1*m2*r2*theta1_dot*theta2_dot*sin(theta2) -
2*L2*m2*r1*theta1_dot*theta2_dot*sin(theta2) -
2*L1*m3*r2*theta1_dot*theta2_dot*sin(theta2) -
2*L2*m3*r1*theta1_dot*theta2_dot*sin(theta2) -
2*L1*m4*r2*theta1_dot*theta2_dot*sin(theta2) -
2*L2*m4*r1*theta1_dot*theta2_dot*sin(theta2) -
2*L1*m5*r2*theta1_dot*theta2_dot*sin(theta2) -
2*L2*m5*r1*theta1_dot*theta2_dot*sin(theta2) +
2*L2*m3*r3*theta1_dot*theta3_dot*sin(theta3) +
2*L2*m3*r3*theta2_dot*theta3_dot*sin(theta3) -
2*L1*m5*r5*theta1_dot*theta2_dot*sin(theta2 + theta3 - theta4 + theta5) -
2*L1*m5*r5*theta1_dot*theta3_dot*sin(theta2 + theta3 - theta4 + theta5) +
```

```
2*L1*m5*r5*theta1_dot*theta4_dot*sin(theta2 + theta3 - theta4 + theta5) -
2*L1*m5*r5*theta2_dot*theta3_dot*sin(theta2 + theta3 - theta4 + theta5) -
2*L1*m5*r5*theta1_dot*theta5_dot*sin(theta2 + theta3 - theta4 + theta5) +
2*L1*m5*r5*theta2_dot*theta4_dot*sin(theta2 + theta3 - theta4 + theta5) -
2*L1*m5*r5*theta2_dot*theta5_dot*sin(theta2 + theta3 - theta4 + theta5) +
2*L1*m5*r5*theta3_dot*theta4_dot*sin(theta2 + theta3 - theta4 + theta5) -
2*L1*m5*r5*theta3_dot*theta5_dot*sin(theta2 + theta3 - theta4 + theta5) +
2*L1*m5*r5*theta4_dot*theta5_dot*sin(theta2 + theta3 - theta4 + theta5) +
2*m2*r1*r2*theta1_dot*theta2_dot*sin(theta2) +
2*m3*r1*r2*theta1_dot*theta2_dot*sin(theta2) +
2*m4*r1*r2*theta1_dot*theta2_dot*sin(theta2) +
2*m5*r1*r2*theta1_dot*theta2_dot*sin(theta2) -
2*m3*r2*r3*theta1_dot*theta3_dot*sin(theta3) -
2*m3*r2*r3*theta2_dot*theta3_dot*sin(theta3) +
2*m5*r4*r5*theta1_dot*theta5_dot*sin(theta5) +
2*m5*r4*r5*theta2_dot*theta5_dot*sin(theta5) +
2*m5*r4*r5*theta3_dot*theta5_dot*sin(theta5) -
2*m5*r4*r5*theta4_dot*theta5_dot*sin(theta5) -
2*L2*m4*r4*theta1_dot*theta3_dot*sin(theta3 - theta4) +
2*L2*m4*r4*theta1_dot*theta4_dot*sin(theta3 - theta4) -
2*L2*m4*r4*theta2_dot*theta3_dot*sin(theta3 - theta4) -
2*L2*m5*r4*theta1_dot*theta3_dot*sin(theta3 - theta4) +
2*L2*m4*r4*theta2_dot*theta4_dot*sin(theta3 - theta4) +
2*L2*m5*r4*theta1_dot*theta4_dot*sin(theta3 - theta4) -
2*L2*m5*r4*theta2_dot*theta3_dot*sin(theta3 - theta4) +
2*L2*m4*r4*theta3_dot*theta4_dot*sin(theta3 - theta4) +
2*L2*m5*r4*theta2_dot*theta4_dot*sin(theta3 - theta4) +
2*L2*m5*r4*theta3_dot*theta4_dot*sin(theta3 - theta4) +
2*m5*r1*r5*theta1_dot*theta2_dot*sin(theta2 + theta3 - theta4 + theta5) +
2*m5*r1*r5*theta1_dot*theta3_dot*sin(theta2 + theta3 - theta4 + theta5) -
2*m5*r1*r5*theta1_dot*theta4_dot*sin(theta2 + theta3 - theta4 + theta5) +
2*m5*r1*r5*theta2_dot*theta3_dot*sin(theta2 + theta3 - theta4 + theta5) +
2*m5*r1*r5*theta1_dot*theta5_dot*sin(theta2 + theta3 - theta4 + theta5) -
2*m5*r1*r5*theta2_dot*theta4_dot*sin(theta2 + theta3 - theta4 + theta5) +
2*m5*r1*r5*theta2_dot*theta5_dot*sin(theta2 + theta3 - theta4 + theta5) -
2*m5*r1*r5*theta3_dot*theta4_dot*sin(theta2 + theta3 - theta4 + theta5) +
2*m5*r1*r5*theta3_dot*theta5_dot*sin(theta2 + theta3 - theta4 + theta5) -
2*m5*r1*r5*theta4_dot*theta5_dot*sin(theta2 + theta3 - theta4 + theta5) +
2*m4*r2*r4*theta1_dot*theta3_dot*sin(theta3 - theta4) -
2*m4*r2*r4*theta1_dot*theta4_dot*sin(theta3 - theta4) +
2*m4*r2*r4*theta2_dot*theta3_dot*sin(theta3 - theta4) +
2*m5*r2*r4*theta1_dot*theta3_dot*sin(theta3 - theta4) -
2*m4*r2*r4*theta2_dot*theta4_dot*sin(theta3 - theta4) -
2*m5*r2*r4*theta1_dot*theta4_dot*sin(theta3 - theta4) +
2*m5*r2*r4*theta2_dot*theta3_dot*sin(theta3 - theta4) -
2*m4*r2*r4*theta3_dot*theta4_dot*sin(theta3 - theta4) -
2*m5*r2*r4*theta2_dot*theta4_dot*sin(theta3 - theta4) -
2*m5*r2*r4*theta3_dot*theta4_dot*sin(theta3 - theta4);
RHS(2,1) = u2 + g*m4*(sin(theta1 + theta2)*(L2 - r2) - r4*sin(theta1 + theta2 + theta3 -
theta4)) + g*m3*(r3*sin(theta1 + theta2 + theta3) + sin(theta1 + theta2)*(L2 - r2)) -
g*m5*(r5*sin(theta1 + theta2 + theta3 - theta4 + theta5) - sin(theta1 + theta2)*(L2 -
r2) + r4*sin(theta1 + theta2 + theta3 - theta4)) + g*m2*sin(theta1 + theta2)*(L2 - r2) -
m4*r1*r4*theta1_dot^2*sin(theta2 + theta3 - theta4) - m5*r1*r4*theta1_dot^2*sin(theta2 +
theta3 - theta4) + m5*r2*r5*theta3_dot^2*sin(theta3 - theta4 + theta5) +
```

```
m5*r2*r5*theta4_dot^2*sin(theta3 - theta4 + theta5) + m5*r2*r5*theta5_dot^2*sin(theta3 -
theta4 + theta5) - L1*m3*r3*theta1_dot^2*sin(theta2 + theta3) -
L1*L2*m2*theta1_dot^2*sin(theta2) - L1*L2*m3*theta1_dot^2*sin(theta2) -
L1*L2*m4*theta1_dot^2*sin(theta2) - L1*L2*m5*theta1_dot^2*sin(theta2) +
m3*r1*r3*theta1_dot^2*sin(theta2 + theta3) + L1*m2*r2*theta1_dot^2*sin(theta2) +
L2*m2*r1*theta1_dot^2*sin(theta2) + L1*m3*r2*theta1_dot^2*sin(theta2) +
L2*m3*r1*theta1_dot^2*sin(theta2) + L1*m4*r2*theta1_dot^2*sin(theta2) +
L2*m4*r1*theta1_dot^2*sin(theta2) + L1*m5*r2*theta1_dot^2*sin(theta2) +
L2*m5*r1*theta1_dot^2*sin(theta2) + L2*m3*r3*theta3_dot^2*sin(theta3) +
L1*m5*r5*theta1_dot^2*sin(theta2 + theta3 - theta4 + theta5) -
m2*r1*r2*theta1_dot^2*sin(theta2) - m3*r1*r2*theta1_dot^2*sin(theta2) -
m4*r1*r2*theta1_dot^2*sin(theta2) - m5*r1*r2*theta1_dot^2*sin(theta2) -
m3*r2*r3*theta3_dot^2*sin(theta3) + m5*r4*r5*theta5_dot^2*sin(theta5) -
L2*m4*r4*theta3_dot^2*sin(theta3 - theta4) - L2*m4*r4*theta4_dot^2*sin(theta3 - theta4)
- L2*m5*r4*theta3_dot^2*sin(theta3 - theta4) - L2*m5*r4*theta4_dot^2*sin(theta3 -
theta4) - m5*r1*r5*theta1_dot^2*sin(theta2 + theta3 - theta4 + theta5) +
m4*r2*r4*theta3_dot^2*sin(theta3 - theta4) + m4*r2*r4*theta4_dot^2*sin(theta3 - theta4)
+ m5*r2*r4*theta3_dot^2*sin(theta3 - theta4) + m5*r2*r4*theta4_dot^2*sin(theta3 -
theta4) + L1*m4*r4*theta1_dot^2*sin(theta2 + theta3 - theta4) +
L1*m5*r4*theta1_dot^2*sin(theta2 + theta3 - theta4) - L2*m5*r5*theta3_dot^2*sin(theta3 -
theta4 + theta5) - L2*m5*r5*theta4_dot^2*sin(theta3 - theta4 + theta5) -
L2*m5*r5*theta5_dot^2*sin(theta3 - theta4 + theta5) -
2*L2*m5*r5*theta1_dot*theta3_dot*sin(theta3 - theta4 + theta5) +
2*L2*m5*r5*theta1_dot*theta4_dot*sin(theta3 - theta4 + theta5) -
2*L2*m5*r5*theta2_dot*theta3_dot*sin(theta3 - theta4 + theta5) -
2*L2*m5*r5*theta1_dot*theta5_dot*sin(theta3 - theta4 + theta5) +
2*L2*m5*r5*theta2_dot*theta4_dot*sin(theta3 - theta4 + theta5) -
2*L2*m5*r5*theta2_dot*theta5_dot*sin(theta3 - theta4 + theta5) +
2*L2*m5*r5*theta3_dot*theta4_dot*sin(theta3 - theta4 + theta5) -
2*L2*m5*r5*theta3_dot*theta5_dot*sin(theta3 - theta4 + theta5) +
2*L2*m5*r5*theta4_dot*theta5_dot*sin(theta3 - theta4 + theta5) +
2*m5*r2*r5*theta1_dot*theta3_dot*sin(theta3 - theta4 + theta5) -
2*m5*r2*r5*theta1_dot*theta4_dot*sin(theta3 - theta4 + theta5) +
2*m5*r2*r5*theta2_dot*theta3_dot*sin(theta3 - theta4 + theta5) +
2*m5*r2*r5*theta1_dot*theta5_dot*sin(theta3 - theta4 + theta5) -
2*m5*r2*r5*theta2_dot*theta4_dot*sin(theta3 - theta4 + theta5) +
2*m5*r2*r5*theta2_dot*theta5_dot*sin(theta3 - theta4 + theta5) -
2*m5*r2*r5*theta3_dot*theta4_dot*sin(theta3 - theta4 + theta5) +
2*m5*r2*r5*theta3_dot*theta5_dot*sin(theta3 - theta4 + theta5) -
2*m5*r2*r5*theta4_dot*theta5_dot*sin(theta3 - theta4 + theta5) +
2*L2*m3*r3*theta1_dot*theta3_dot*sin(theta3) +
2*L2*m3*r3*theta2_dot*theta3_dot*sin(theta3) -
2*m3*r2*r3*theta1_dot*theta3_dot*sin(theta3) -
2*m3*r2*r3*theta2_dot*theta3_dot*sin(theta3) +
2*m5*r4*r5*theta1_dot*theta5_dot*sin(theta5) +
2*m5*r4*r5*theta2_dot*theta5_dot*sin(theta5) +
2*m5*r4*r5*theta3_dot*theta5_dot*sin(theta5) -
2*m5*r4*r5*theta4_dot*theta5_dot*sin(theta5) -
2*L2*m4*r4*theta1_dot*theta3_dot*sin(theta3 - theta4) +
2*L2*m4*r4*theta1_dot*theta4_dot*sin(theta3 - theta4) -
2*L2*m4*r4*theta2_dot*theta3_dot*sin(theta3 - theta4) -
2*L2*m5*r4*theta1_dot*theta3_dot*sin(theta3 - theta4) +
2*L2*m4*r4*theta2_dot*theta4_dot*sin(theta3 - theta4) +
2*L2*m5*r4*theta1_dot*theta4_dot*sin(theta3 - theta4) -
```

```
2*L2*m5*r4*theta2_dot*theta3_dot*sin(theta3 - theta4) +
2*L2*m4*r4*theta3_dot*theta4_dot*sin(theta3 - theta4) +
2*L2*m5*r4*theta2_dot*theta4_dot*sin(theta3 - theta4) +
2*L2*m5*r4*theta3_dot*theta4_dot*sin(theta3 - theta4) +
2*m4*r2*r4*theta1_dot*theta3_dot*sin(theta3 - theta4) -
2*m4*r2*r4*theta1_dot*theta4_dot*sin(theta3 - theta4) +
2*m4*r2*r4*theta2_dot*theta3_dot*sin(theta3 - theta4) +
2*m5*r2*r4*theta1_dot*theta3_dot*sin(theta3 - theta4) -
2*m4*r2*r4*theta2_dot*theta4_dot*sin(theta3 - theta4) -
2*m5*r2*r4*theta1_dot*theta4_dot*sin(theta3 - theta4) +
2*m5*r2*r4*theta2_dot*theta3_dot*sin(theta3 - theta4) -
2*m4*r2*r4*theta3_dot*theta4_dot*sin(theta3 - theta4) -
2*m5*r2*r4*theta2_dot*theta4_dot*sin(theta3 - theta4) -
2*m5*r2*r4*theta3_dot*theta4_dot*sin(theta3 - theta4);
RHS(3,1) = u3 - g*m5*r5*sin(theta1 + theta2 + theta3 - theta4 + theta5) -
g*m4*r4*sin(theta1 + theta2 + theta3 - theta4) - g*m5*r4*sin(theta1 + theta2 + theta3 -
theta4) + g*m3*r3*sin(theta1 + theta2 + theta3) - m4*r1*r4*theta1_dot^2*sin(theta2 +
theta3 - theta4) - m5*r1*r4*theta1_dot^2*sin(theta2 + theta3 - theta4) -
m5*r2*r5*theta1_dot^2*sin(theta3 - theta4 + theta5) - m5*r2*r5*theta2_dot^2*sin(theta3 -
theta4 + theta5) - L1*m3*r3*theta1_dot^2*sin(theta2 + theta3) +
m3*r1*r3*theta1_dot^2*sin(theta2 + theta3) - L2*m3*r3*theta1_dot^2*sin(theta3) -
L2*m3*r3*theta2_dot^2*sin(theta3) + L1*m5*r5*theta1_dot^2*sin(theta2 + theta3 - theta4 +
theta5) + m3*r2*r3*theta1_dot^2*sin(theta3) + m3*r2*r3*theta2_dot^2*sin(theta3) +
m5*r4*r5*theta5_dot^2*sin(theta5) + L2*m4*r4*theta1_dot^2*sin(theta3 - theta4) +
L2*m4*r4*theta2_dot^2*sin(theta3 - theta4) + L2*m5*r4*theta1_dot^2*sin(theta3 - theta4)
+ L2*m5*r4*theta2_dot^2*sin(theta3 - theta4) - m5*r1*r5*theta1_dot^2*sin(theta2 + theta3
- theta4 + theta5) - m4*r2*r4*theta1_dot^2*sin(theta3 - theta4) -
m4*r2*r4*theta2_dot^2*sin(theta3 - theta4) - m5*r2*r4*theta1_dot^2*sin(theta3 - theta4)
- m5*r2*r4*theta2_dot^2*sin(theta3 - theta4) + L1*m4*r4*theta1_dot^2*sin(theta2 + theta3
- theta4) + L1*m5*r4*theta1_dot^2*sin(theta2 + theta3 - theta4) +
L2*m5*r5*theta1_dot^2*sin(theta3 - theta4 + theta5) + L2*m5*r5*theta2_dot^2*sin(theta3 -
theta4 + theta5) + 2*L2*m5*r5*theta1_dot*theta2_dot*sin(theta3 - theta4 + theta5) -
2*m5*r2*r5*theta1_dot*theta2_dot*sin(theta3 - theta4 + theta5) -
2*L2*m3*r3*theta1_dot*theta2_dot*sin(theta3) +
2*m3*r2*r3*theta1_dot*theta2_dot*sin(theta3) +
2*m5*r4*r5*theta1_dot*theta5_dot*sin(theta5) +
2*m5*r4*r5*theta2_dot*theta5_dot*sin(theta5) +
2*m5*r4*r5*theta3_dot*theta5_dot*sin(theta5) -
2*m5*r4*r5*theta4_dot*theta5_dot*sin(theta5) +
2*L2*m4*r4*theta1_dot*theta2_dot*sin(theta3 - theta4) +
2*L2*m5*r4*theta1_dot*theta2_dot*sin(theta3 - theta4) -
2*m4*r2*r4*theta1_dot*theta2_dot*sin(theta3 - theta4) -
2*m5*r2*r4*theta1_dot*theta2_dot*sin(theta3 - theta4);
RHS(4,:) = u4 + g*m5*r5*sin(theta1 + theta2 + theta3 - theta4 + theta5) +
g*m4*r4*sin(theta1 + theta2 + theta3 - theta4) + g*m5*r4*sin(theta1 + theta2 + theta3 -
theta4) + m4*r1*r4*theta1_dot^2*sin(theta2 + theta3 - theta4) +
m5*r1*r4*theta1_dot^2*sin(theta2 + theta3 - theta4) + m5*r2*r5*theta1_dot^2*sin(theta3 -
theta4 + theta5) + m5*r2*r5*theta2_dot^2*sin(theta3 - theta4 + theta5) -
L1*m5*r5*theta1_dot^2*sin(theta2 + theta3 - theta4 + theta5) -
m5*r4*r5*theta5_dot^2*sin(theta5) - L2*m4*r4*theta1_dot^2*sin(theta3 - theta4) -
L2*m4*r4*theta2_dot^2*sin(theta3 - theta4) - L2*m5*r4*theta1_dot^2*sin(theta3 - theta4)
- L2*m5*r4*theta2_dot^2*sin(theta3 - theta4) + m5*r1*r5*theta1_dot^2*sin(theta2 + theta3
- theta4 + theta5) + m4*r2*r4*theta1_dot^2*sin(theta3 - theta4) +
m4*r2*r4*theta2_dot^2*sin(theta3 - theta4) + m5*r2*r4*theta1_dot^2*sin(theta3 - theta4)
```

```
+ m5*r2*r4*theta2_dot^2*sin(theta3 - theta4) - L1*m4*r4*theta1_dot^2*sin(theta2 + theta3
- theta4) - L1*m5*r4*theta1_dot^2*sin(theta2 + theta3 - theta4) -
L2*m5*r5*theta1_dot^2*sin(theta3 - theta4 + theta5) - L2*m5*r5*theta2_dot^2*sin(theta3 -
theta4 + theta5) - 2*L2*m5*r5*theta1_dot*theta2_dot*sin(theta3 - theta4 + theta5) +
2*m5*r2*r5*theta1_dot*theta2_dot*sin(theta3 - theta4 + theta5) -
2*m5*r4*r5*theta1_dot*theta5_dot*sin(theta5) -
2*m5*r4*r5*theta2_dot*theta5_dot*sin(theta5) -
2*m5*r4*r5*theta3_dot*theta5_dot*sin(theta5) +
2*m5*r4*r5*theta4_dot*theta5_dot*sin(theta5) -
2*L2*m4*r4*theta1_dot*theta2_dot*sin(theta3 - theta4) -
2*L2*m5*r4*theta1_dot*theta2_dot*sin(theta3 - theta4) +
2*m4*r2*r4*theta1_dot*theta2_dot*sin(theta3 - theta4) +
2*m5*r2*r4*theta1_dot*theta2_dot*sin(theta3 - theta4);
RHS(5,:) = u5 - g*m5*r5*sin(theta1 + theta2 + theta3 - theta4 + theta5) -
m5*r2*r5*theta1_dot^2*sin(theta3 - theta4 + theta5) - m5*r2*r5*theta2_dot^2*sin(theta3 -
theta4 + theta5) + L1*m5*r5*theta1_dot^2*sin(theta2 + theta3 - theta4 + theta5) -
m5*r4*r5*theta1_dot^2*sin(theta5) - m5*r4*r5*theta2_dot^2*sin(theta5) -
m5*r4*r5*theta3_dot^2*sin(theta5) - m5*r4*r5*theta4_dot^2*sin(theta5) -
m5*r1*r5*theta1_dot^2*sin(theta2 + theta3 - theta4 + theta5) +
L2*m5*r5*theta1_dot^2*sin(theta3 - theta4 + theta5) + L2*m5*r5*theta2_dot^2*sin(theta3 -
theta4 + theta5) + 2*L2*m5*r5*theta1_dot*theta2_dot*sin(theta3 - theta4 + theta5) -
2*m5*r2*r5*theta1_dot*theta2_dot*sin(theta3 - theta4 + theta5) -
2*m5*r4*r5*theta1_dot*theta2_dot*sin(theta5) -
2*m5*r4*r5*theta1_dot*theta3_dot*sin(theta5) +
2*m5*r4*r5*theta1_dot*theta4_dot*sin(theta5) -
2*m5*r4*r5*theta2_dot*theta3_dot*sin(theta5) +
2*m5*r4*r5*theta2_dot*theta4_dot*sin(theta5) +
2*m5*r4*r5*theta3_dot*theta4_dot*sin(theta5);

% ODE and path constraints
ceq = collocate({M(1,:)*dot(qd) == RHS(1);
    M(2,:)*dot(qd) == RHS(2);
    M(3,:)*dot(qd) == RHS(3);
    M(4,:)*dot(qd) == RHS(4);
    M(5,:)*dot(qd) == RHS(5);
    qd == dot(q)});

% Function to be optimized
%objective = integrate(u1*u1 + u2*u2 + u3*u3 + u4*u4 + u5*u5);
objective = tf;

% Solve the problem
options = struct;
options.name = 'FiveLinkHumanoid';
options.solve = 'snopt';
solution = ezsolve(objective,{cbox,cbnd,ceq},x0, options);

% grabbing solution
opt.tf = subs(tf,solution);
opt.ti  = subs(icollocate(t),solution);
opt.tc = subs(collocate(t),solution);
opt.q = subs(icollocate(q),solution);
opt.qd = subs(icollocate(qd),solution);
opt.qd = subs(icollocate(qd),solution);
```

```
opt.u = subs(collocate(u),solution);
opt.objective = subs(objective,solution);

for i = 1:length(opt.ti)
    [opt.pm(:,:,i), opt.pcm(:,:,i)] = computeFK(opt.q(i,:));
    opt.xcm(i) = computeCM(opt.pcm(1,:,i)); % plot Center of Mass in x-position
end

% plotting joint angles
plot(opt.ti,opt.q);
legend('x1','x2','x3','x4','x5','Location','northeastoutside')
title('Criteria: ZMP - Objective Function: Maximum Velocity')
grid
% plotting joint velocities
figure
plot(opt.ti,opt.qd);
legend('x6','x7','x8','x9','x10','Location','northeastoutside')
title('Criteria: ZMP - Objective Function: Maximum Velocity')
grid
% plotting center of mass
figure
plot(opt.ti,opt.xcm');
grid
title('Criteria: ZMP - Objective Function: Maximum Velocity')
legend('xCOM','Location','northeastoutside')
% plotting torque
figure
plot(opt.tc,opt.u);
grid
title('Criteria: ZMP - Objective Function: Maximum Velocity')
legend('u1','u2','u3','u4','u5','Location','northeastoutside')

for i = 1:length(opt.tc)
    zmpplot(i) = computeZMP(opt.q(i,:),opt.qd(i,:),opt.u(i,:));
end
% plotting zmp
figure
plot(opt.tc,zmpplot);
grid
title('Criteria: ZMP - Objective Function: Maximum Velocity')
legend('xZMP','Location','northeastoutside')



figure
stickFigure(opt);

% figure
% filme= simulate(opt);
% movie2avi(filme,'Hum_movie')
```

Nome do arquivo: computeCMx.m

```
function xcm = computeCMx(x)
% computes the center of mass in the x-direction for the five link robot

mass = 50; % [kg]
m1 = 0.061 * mass; % [kg]
m2 = 0.1 * mass; % [kg]
m3 = 0.678 * mass; % [kg]
m4 = m2; % [kg]
m5 = m1; % [kg]

m = [m1 m2 m3 m4 m5]; % [kg]

xcm = ( m(1)*x(1) + m(2)*x(2) + m(3)*x(3) + m(4)*x(4) + m(5)*x(5) ) / sum(m); % [kg]


end
```

Nome do arquivo: computeFK.m

```
function [ pm, pcm] = computeFK(theta)
% Compute Forward Kinematics for a five link robot
% pm  is a 2 x 5 array with the joint cartesian position of each body
% pcm is a 2 x 5 array with the Center of Mass position of each body
% Last modification: 30/03/2014 (by Feliphe G. Galiza)

% parameters according to human body
height = 1.5; % [m]
L1 = 0.285*height; % [m]
L2 = (0.53 - 0.285)*height; % [m]
L3 = height - (L1 + L2); % [m]
L4 = L2; % [m]
L5 =L1; % [m]

r1 = L1/2; % [m]
r2 = L2/2; % [m]
r3 = L3/2; % [m]
r4 = L4/2; % [m]
r5 = L5/2; % [m]

mass = 50; % [kg]
m1 = 0.061 * mass; % [kg]
m2 = 0.1 * mass; % [kg]
m3 = 0.678 * mass; % [kg]
m4 = m2; % [kg]
m5 = m1; % [kg]

Izz_1 = m1 * 0.735^2; % [kg*m^2]
Izz_2 = m2 * 0.540^2; % [kg*m^2]
Izz_3 = m3 * 0.0798^2; % [kg*m^2]
Izz_4 = Izz_2; % [kg*m^2]
Izz_5 = Izz_1; % [kg*m^2]
```

```
% global orientations
o1 =  -theta(1);
o2 =  -theta(1) - theta(2);
o3 = -theta(1) - theta(2) - theta(3);
o4 = -theta(1) - theta(2) - theta(3) + theta(4);
o5 = -theta(1) - theta(2) -theta(3) +theta(4) - theta(5);


% global cartesian positions
pm1 = [ -L1 * sin(o1); L1* cos(o1)];
pm2 = pm1 + [-L2 * sin(o2); L2 *cos(o2)];
pm3 = pm2 + [-L3 * sin(o3); L3 * cos(o3)];
pm4 = pm2 + [L4 * sin(o4); -L4 * cos(o4)];
pm5 = pm4 + [L5 * sin(o5); -L5 * cos(o5)];
pm = [pm1 pm2 pm3 pm4 pm5];

% global center of mass positions
pcm1 = [ (-L1 + r1) * sin(o1); (L1 - r1)* cos(o1)];
pcm2 = pm1 + [(-L2 + r2) * sin(o2); (L2-r2) *cos(o2)];
pcm3 = pm2 + [-r3 * sin(o3); r3 * cos(o3)];
pcm4 = pm2 + [r4 * sin(o4); -r4 * cos(o4)];
pcm5 = pm4 + [r5 * sin(o5); -r5 * cos(o5)];
pcm = [pcm1 pcm2 pcm3 pcm4 pcm5];

end
```

Nome do Arquivo: simulate.m

```
function frames = simulate(opt)
% Simulates the five link robot
% in order to run the simulation run the file named as
% runSimulationFiveLink.
% Last modification: 30/03/2014 (by Feliphe G. Galiza)

close all

subplot(3,3,3);
plot(opt.ti,opt.q)
legend('x1','x2','x3','x4','x5')

subplot(3,3,6);
plot(opt.ti,opt.qd)
legend('xd1','xd2','xd3','xd4','xd5')

subplot(3,3,9);
plot(opt.ti,opt.xcm','r')
legend('xCOM')
% hold on
% plot(opt.ti,opt.xzmp','r')
% legend('xCOM','xZMP')
```

```
sim = subplot(3,3,[1 2 4 5 7 8]);

for step=1:1:1

for t=1:1:length(opt.ti)

    theta(1) = opt.q(t,1);
    theta(2) = opt.q(t,2);
    theta(3) = opt.q(t,3);
    theta(4) = opt.q(t,4);
    theta(5) = opt.q(t,5);

    [pm, pcm] = computeFK(theta);

    x = [0 pm(1,1:3)  pm(1,2) pm(1,4:5) ] ;
    y = [0 pm(2,1:3) pm(2,2) pm(2,4:5) ];

     xcm = pcm(1,:);
     ycm = pcm(2,:);

    delete(sim);
    sim = subplot(3,3,[1 2 4 5 7 8]);

    text = sprintf('Time: %0.2f sec',opt.ti(t));
    title(text)

    hold on
    plot(x,y,'b','LineWidth',3)
            hold on
            plot(xcm,ycm,'o',...
                'LineWidth',2,...
                'MarkerSize',10,...
                'MarkerEdgeColor','b',...
                'MarkerFaceColor','g')

            hold on
            plot(0,0,'^',...
                'LineWidth',2,...
                'MarkerSize',13,...
                'MarkerEdgeColor','k',...
                'MarkerFaceColor','b')

            hold on
            plot(opt.xcm(t),0,'x',...
                'LineWidth',3,...
                'MarkerSize',8,...
                'MarkerEdgeColor','r',...
                'MarkerFaceColor','r')

    axis equal

    grid
    drawnow
    pause(1/40)
```

```
    frames(t)=getframe(gcf);
end
end
end
```

Nome do Arquivo: LagrangianDynamics_EoM.m

```
% Computes the simbolic Equation of Motion of a Five Link Robot using Lagragian Dynamics
% Last modification: 30/03/2014 (by Feliphe G. Galiza)

syms theta1 theta2 theta3 theta4 theta5 theta1_dot theta2_dot theta3_dot theta4_dot
theta5_dot theta1_acc theta2_acc theta3_acc theta4_acc theta5_acc m1 m2 m3 m4 m5 m6 r1
r2 r3 r4 r5 L1 L2 L3 L4 L5 J1 J2 J3 J4 J5 u1 u2 u3 u4 u5

u = [u1; u2; u3; u4; u5]; % joint torques

% joints position velocities and accelerations
theta = [theta1; theta2; theta3; theta4 ;theta5]; % joint angles
theta_dot =[theta1_dot; theta2_dot; theta3_dot; theta4_dot; theta5_dot]; % joint
velocities
theta_acc =[theta1_acc; theta2_acc; theta3_acc; theta4_acc; theta5_acc]; % joint
accelerations

% moments of inertia in respect to the base cordinate frame. i.e. Inertial Frame
I(:,:,1) = [0, 0, 0; 0, 0, 0; 0,0, J1];
I(:,:,2) = [0, 0, 0; 0, 0, 0; 0,0, J2];
I(:,:,3) = [0, 0, 0; 0, 0, 0; 0,0, J3];
I(:,:,4) = [0, 0, 0; 0, 0, 0; 0,0, J4];
I(:,:,5) = [0, 0, 0; 0, 0, 0; 0,0, J5];

% orientations
o1 =  -theta(1);
o2 =  -theta(1) - theta(2);
o3 = -theta(1) - theta(2) - theta(3);
o4 = -theta(1) - theta(2) - theta(3) + theta(4);
o5 = -theta(1) - theta(2) -theta(3) +theta(4) - theta(5);
o =  [o1 o2 o3 o4 o5];


% cartesian position of each joint
pm1 = [ -L1 * sin(o1); L1* cos(o1)];
pm2 = pm1 + [-L2 * sin(o2); L2 *cos(o2)];
pm3 = pm2 + [-L3 * sin(o3); L3 * cos(o3)];
pm4 = pm2 + [L4 * sin(o4); -L4 * cos(o4)];
pm5 = pm4 + [L5 * sin(o5); -L5 * cos(o5)];

% cartesian position of the Center of Mass for each body
pcm1 = [ (-L1 + r1) * sin(o1); (L1 - r1)* cos(o1)];
pcm2 = pm1 + [(-L2 + r2) * sin(o2); (L2-r2) *cos(o2)];
pcm3 = pm2 + [-r3 * sin(o3); r3 * cos(o3)];
pcm4 = pm2 + [r4 * sin(o4); -r4 * cos(o4)];
```

```
pcm5 = pm4 + [r5 * sin(o5); -r5 * cos(o5)];
pcm = [pcm1 pcm2 pcm3 pcm4 pcm5];

% angular accelerations
w1_dot = -theta1_acc;
w2_dot =  -theta1_acc - theta2_acc;
w3_dot = -theta1_acc - theta2_acc - theta3_acc;
w4_dot = -theta1_acc - theta2_acc - theta3_acc + theta4_acc;
w5_dot = -theta1_acc - theta2_acc -theta3_acc +theta4_acc - theta5_acc;
w_dot =  [w1_dot w2_dot w3_dot w4_dot w5_dot];

% mass of each body
m = [m1; m2; m3; m4; m5];

% struct to store jacobian matrices
robotBody = struct('Jp',[],'Jo',[],'J',[]);

% building jacobian matrices
for corpo=1:5
    robotBody(corpo).Jp = [jacobian(pcm(:,corpo),theta.');zeros(1,5)];
    robotBody(corpo).Jo = [zeros(2,5);jacobian(o(corpo),theta.')];
    robotBody(corpo).J  = [robotBody(corpo).Jp; robotBody(corpo).Jo];
end

% building global inertia matrix
H =sym(zeros(5,5));
    for corpo=1:5
        H = H + m(corpo)*(robotBody(corpo).Jp.')*(robotBody(corpo).Jp) +
(robotBody(corpo).Jo.')*I(:,:,corpo)*(robotBody(corpo).Jo);
    end

% building Christoffel's Three-Index Symbol
C = sym(zeros(5,5,5));
 for k=1:5
     for i=1:5
         for j=1:5
            C(i,j,k) = diff(H(i,j),theta(k)) - 1/2*diff(H(j,k),theta(i));
        end
     end
 end

% building Centrifugal and Coriolis terms
 h = sym(zeros(5,1));
 for i=1:5
        for j=1:5
            for k=1:5
              h(i) = h(i) + C(i,j,k)*theta_dot(j)*theta_dot(k);
            end
        end
 end

 g = 9.81; % gravity [m/s^2]

% building potential energy
```

```
U=0;
for corpo=1:5
    U = U + m(corpo)*g*pcm(2,corpo);
end

% taking the potential energy derivative
dU=sym(zeros(1,5));
for corpo=1:5
    dU(corpo) = diff(U,theta(corpo));
end

rhs = u - h - dU.';

% Equation of Motion is in the form H*x_dot = rhs, where x_dot is the time
% derivative of the states [q qd]'
```

Nome do Arquivo: NewtonEulerDynamics_EoM.m

```
% Computes the simbolic Equation of Motion of a Five Link Robot using
% Newton-Euler Formulation
% Last modification: 30/03/2014 (by Feliphe G. Galiza)

% Symbolic variables
syms theta1 theta2 theta3 theta4 theta5 theta1_dot theta2_dot theta3_dot theta4_dot
theta5_dot theta1_acc theta2_acc theta3_acc theta4_acc theta5_acc
syms u1 u2 u3 u4 u5 % forces and moments
syms m1 m2 m3 m4 m5 r1 r2 r3 r4 r5 L1 L2 L3 L4 L5 J1 J2 J3 J4 J5 g

% Vector of generalized coordinates
q = [theta1; theta2; theta3; theta4; theta5];

% Vector of generalized velocities
qd = [theta1_dot; theta2_dot; theta3_dot; theta4_dot; theta5_dot];

% orientations
o1 =  -q(1);
o2 =  -q(1) - q(2);
o3 = -q(1) - q(2) - q(3);
o4 = -q(1) - q(2) - q(3) + q(4);
o5 = -q(1) - q(2) -q(3) +q(4) - q(5);
o =  [o1 o2 o3 o4 o5];

% position and orientation vector
r(1,1) = (-L1 + r1) * sin(o1); % x cm position of link-1
r(2,1) = (L1 - r1)* cos(o1); % y cm position of link-1
r(3,1) = r(1,1) + (-L2 + r2) * sin(o2);% x cm position of link-2
r(4,1) = r(2,1) + (L2-r2) *cos(o2); % y cm position of link-2
r(5,1) = r(3,1) - r3 * sin(o3);% x cm position of link-3
```

```
r(6,1) = r(4,1) + r3 * cos(o3);  % y cm position of link-3
r(7,1) = r(3,1) + r4 * sin(o4); % x cm position of link-4
r(8,1) = r(4,1) - r4 * cos(o4); % y cm position of link-4
r(9,1) = r(7,1) + r5 * sin(o5); % x cm position of link-5
r(10,1) = r(8,1)- r5 * cos(o5); % y cm position of link-5
r(11,1) = o(1); % orientation of link-1
r(12,1) = o(2); % orientation of link-2
r(13,1) = o(3); % orientation of link-3
r(14,1) = o(4); % orientation of link-4
r(15,1) = o(5); % orientation of link-5

% Inertia matrix
MM = diag([m1 m1 m2 m2 m3 m3 m4 m4 m5 m5 J1 J2 J3 J4 J5]);

% Vector of applied forces and moments, excluding the reaction forces and
% moments
Fe = [0; -m1*g; 0; -m2*g; 0; -m3*g; 0; -m4*g; 0; -m5*g; -u1+u2; -u2+u3; -u3-u4; u4+u5; -
u5];

% Jacobian
J = [diff(r, theta1), diff(r, theta2), diff(r,theta3), diff(r,theta4), diff(r,theta5)];

% Transposed Jacobian
Jt = J.';

% Derivative of Jacobian in time
for j = 1:length(J(1,:))
        Jd(:,j) = [diff(J(:,j),q(1)), diff(J(:,j),q(2)),
diff(J(:,j),q(3)),diff(J(:,j),q(4)), diff(J(:,j),q(5))]*qd;
end

% Mass matriz
M = Jt*MM*J;
M = simplify(M)

% Vector of generalized Coriolis, centrifugal and gyrocopic forces
k = Jt*MM*Jd*qd;
k = simplify(k)

% Vector of generalized applied forces
ke = Jt*Fe;
ke = simplify(ke)

% Right-hand-side of equations of motion
rhs = ke - k;
rhs = simplify(rhs)

% Equation of Motion is in the form M*x_dot = rhs, where x_dot is the time
% derivative of the states [q qd]'
```

Referências Bibliográficas

[1] H. Harry Asada, ***"Lecture Notes for MIT 2.12 (Fall 2005) : Introduction to Robotics, Chapter 7 Dynamics"***. Disponível em http://ocw.mit.edu/courses/mechanical-engineering/2-12-introduction-to-robotics-fall-2005/lecture-notes/chapter7.pdf Acesso em 02/11/2015.

[2] THORNTON, STEPHEN T., MARION, JERRY B.; ***"Classical Dynamics of Particles and Systems".***

[3] DEKKER, M. H. P. **"Zero-moment point method for stable biped walking.".** Eindhoven, July (2009).

[4] DE LUCA, ALESSANDRO. ***"Lecture Notes on Robotics 2 – Dynamic Model of Robots: Lagrangian approach"***. Disponível em: http://www.diag.uniroma1.it/~deluca/rob2_en/03_LagrangianDynamics_1.pdf Acesso em: 02/11/2015

[5] TEDRAKE, RUSS. ***"Underactuated Robotics: Learning, Planning, and Control for Efficient and Agile Machines". Course Notes for MIT 6.832***

[6] RUDY JOSEPH, ***"Zero Moment Point Walking Controller for Humanoid Walking using Darwin-OP"***. Disponível em: https://ame.nd.edu/undergrad-programs/RudyThesis.pdf Acesso em 02/11/2015.

[7] WINTERS, JACK M.; ***"Terminology and Foundations of Movement Science"*** **.**Disponível em: http://www.eng.mu.edu/wintersj/bien-168/Biomech-Neurocontrol_Chapter-1_JW.htm Acesso em 26/10/2015

[8] MAXIMO, Marcos Ricardo Omena de Albuquerque Maximo. **"*Otimização de Caminhada de Robôs Humanoides*",** 2012. Trabalho de Conclusão de Curso. (Graduação) - Instituto Tecnológico de Aeronáutica, São José dos Campos.

[9] VUKOBRATOVIC, M.; BOROVAC, B.; "***Zero-Moment Point — Thirty Five Years Of Its Life*",** International Journal of Humanoid Robotics, Vol. 1, No. 1, 157–173, 2004.

[10] SARDAIN, P.; BESSONET, G., "***Forces Acting on a Biped Robot. Center of Pressure - Zero Moment Point***", IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, Vol. 34, No. 5, 2004.

[11] CORTEZ, M., **"*Projeto Mecânico de um Robô Humanóide Futebol de Robôs – Humanóide League"***, Projeto de Iniciação Científica - Centro Universitário da FEI, São Bernardo do Campo, São Paulo, 2011.

[12] MELLO, E.; CORTEZ, M.; TONIDANDEL, F.; BIANCHI, R.; GONÇALVES, F. **"*RoboFEI 2013 Humanoid Team Description Paper*",** Competição Brasileira de Robótica, Fortaleza (CBR 2013), 2013.

[13] MELLO, E.; **"*Implementação do Sistema Eletrônico de um Robô Humanoide*"**, Projeto de Iniciação Científica - Centro Universitário da FEI, São Bernardo do Campo, São Paulo, 2012.

[14] ***RoboCup – Humanoid League*** . Disponível em: http://www.tzi.de/humanoid/bin/view/Website/WebHome Acesso em 11 de dezembro de 2013.

[15] ***RoboCup***. Disponível em: http://www.robocup.org Acesso em 11 de dezembro de 2013.

.
[16] **ROBOTIS-OP** (a.k.a. Darwin project); Disponível em:http://sourceforge.net/projects/darwinop/ Acesso em 26/10/2015.
[17] MENDES, EDUARDO M. A. M. ***"Ideias Básicas do Controle Ótimo"***. Disponível em: http://www.cpdee.ufmg.br/~emmendes/otimo.pdf. Acesso em 27/10/2015

[18] SHAFII, N.; REIS, L. P.; LAU, L.; **"*Biped Walking using Coronal and Sagittal Movements based on Truncated Fourier Series*"**, Faculty of Engineering of the University of Porto, Portugal; University of Aveiro, Portugal, 2009. RocoCup 2010: Robot Soccer World Cup XIV.

[19] Qiang Huang, Kazuhito Yokoi, Shuuji Kajita, Kenji Kanekom, Hirohiko Arai, Noriho Koyachi and Kazuo Tanie, ***"Planning walking patterns for a biped robot"***, IEEE Trans. Robotics and Automation. Vol.17, no.3, 2001.