UNIVERSITAT POLITÈCNICA DE CATALUNYA

Programa de Grau

GRAU EN ENGINYERIA EN TECNOLOGIES INDUSTRIALS

Treball de Fi de Grau

CONTROL D'UN PÈNDOL INVERTIT MITJANÇANT UNA PLATAFORMA PARAL·LELA ACTUADA PER CABLES

Pere Giró Perez

Director:
Lluís Ros Giralt

Ponent:
Josep Maria Font-Llagunes

Setembre 2018

Control d'un pèndol invertit mitjançant una plataforma paral·lela actuada per cables

per Pere Giró Perez

Aquest treball s'ha presentat a la Universitat Politècnica de Catalunya per a obtenir el títol de Grau

Programa de Grau:
Grau en Enginyeria en Tecnologies Industrials

Aquest treball s'ha realitzat a:
Institut de Robòtica i Informàtica Industrial, CSIC-UPC

Director:
Lluís Ros Giralt

Ponent:
Josep Maria Font-Llagunes

Tribunal avaluador:
President: Daniel Clos Costa
Vocal: Josep Maria Font Llagunes
Vocal: Montserrat Vallverdú Ferrer

Agraïments

M'agradaria donar les gràcies al Lluís Ros per la seva dedicació i suport durant aquesta última etapa del grau com a director d'aquest treball fi de grau. Sense la seva aportació el resultat seria molt diferent del que presento. També vull donar les gràcies al professor Federico Thomas per brindar-me suport en parts del projecte que requerien coneixement especialitzat, al professor Enric Celaya i l'enginyer de prototipus de l'IRI, Patrick Grosch, per ajudar-me en aspectes relatius als sensors i actuadors utilitzats, i al professor Josep M. Font-Llagunes, per suggerir proves de verificació dels models i per actuar com a ponent del treball. Finalment, estic molt agraït al Ricard Bordalba, doctorand de l'IRI, per proporcionar-me coneixements relatius a la simulació i a l'ús de Matlab i Simulink, i per ajudar-me en aspectes clau, com la integració de les equacions dels motors en el model dinàmic del sistema.

Índex

1 Introducció

1.1 Motivació

La Robòtica es troba en una etapa de gran expansió. L'aparició de mecanismes robòtics innovadors es contínua, i és previsible que el seu ús incrementi notablement en múltiples àmbits. Ja sigui en laboratoris d'investigació, en medicina, o a la indústria, trobem robots paral·lels, dispositius hàptics, vehicles aeris no tripulats, mans antropomorfes, humanoides, i moltes altres màquines realitzant funcions que difícilment podríem haver imaginat fa unes dècades. El món de la robòtica és, per tant, una sortida de futur prometedora per als enginyers industrials. A més és un camp interessant en el qual es poden consolidar i posar en pràctica de manera integrada múltiples competències que s'aprenen de forma relativament separada en els estudis del grau en enginyeria en tecnologies industrials. Aquestes competències, inclouen la modelització matemàtica de sistemes dinàmics amb equacions diferencials, l'obtenció d'aquestes equacions en el cas de sistemes mecànics complexos, l'ús del model dinàmic resultant per al disseny d'un controlador que estabilitzi els sistemes, la validació en simulació del controlador resultant i la implementació de programes de control dels sensors i actuadors d'un sistema. Per veure la interrelació d'aquestes competències se m'ha proposat aplicar-les a un problema de robòtica concret, que consisteix en l'estabilització d'un pèndol invertit utilitzant un robot paral·lel actuat per cables.

Així doncs la motivació principal d'aquest projecte és consolidar les competències esmentades, abordant la modelització dinàmica del sistema format pel robot i el pèndol, obtenint una llei de control adient per a la seva estabilització en una configuració d'equilibri, i validant aquesta llei mitjançant eines de simulació. En una fase posterior, ja fora de l'àmbit d'aquest projecte, s'implementarà la llei de control esmentada sobre el sistema robot-pèndol real.

1.2 Objectiu

L'objectiu d'aquest projecte és dissenyar un sistema de control per tal d'estabilitzar un pèndol invertit mitjançant una plataforma paral·lela actuada per cables (Fig. 1.1). El pèndol manté un contacte puntual sense lliscament amb la plataforma, la qual cosa complica substancialment el problema perquè, en comparació als sistemes carret-pèndol tradicionals, cal controlar dos graus de llibertat d'orientació i no un. La plataforma, però, està suspesa d'una estructura octaèdrica fixa mitjançant sis cables, les longituds dels quals es poden variar independentment utilitzant sis motors. D'aquesta manera es poden controlar els sis graus de llibertat de la plataforma, i corregir les desviacions del pèndol respecte de la posició vertical encara que s'apliquin forces de pertorbació no modelitzades sobre el pèndol. El robot paral·lel utilitzat es coneix amb el nom d'Hexacrane, i fou construït per un projecte previ del Grup de Cinemàtica i Disseny de Robots de l'IRI (Fig. 1.2). El sistema robot-pèndol de la Fig. 1.1 s'ha batejat amb el nom d'Hexapole.

Tot i que el problema de l'estabilització d'un pèndol invertit amb dos graus de llibertat ja s'ha tractat prèviament [1–3], fins ara no s'havia utilitzat un robot paral·lel actuat amb cables com a base estabilitzadora. Això fa que els models dinàmics i lleis de control obtinguts en aquest treball siguin, en gran mesura, originals.

1.3 Abast del treball

Habitualment, el desenvolupament d'un sistema de control s'aborda en dues etapes. En una primera etapa s'obté el model dinàmic del sistema a controlar, es dissenya la llei de control, i se n'avalua el funcionament en simulació. Després, en una segona etapa, s'implementa la llei de control obtinguda en el sistema real. Tenint en compte la complexitat del sistema Hexapole, s'ha decidit restringir l'abast d'aquest treball fi de grau a la primera d'aquestes dues etapes, deixant les tasques d'implementació sobre el sistema real per una etapa posterior. Així doncs, en l'àmbit d'aquest treball es duran a terme quatre tasques principals:

1. L'obtenció del model dinàmic de l'Hexapole.
2. La linealització del sistema al voltant d'un estat d'equilibri inestable corresponent al pèndol en posició vertical invertida i la plataforma en posició horitzontal.
3. El disseny d'una llei de control que estabilitzi el sistema en l'esmentat estat.
4. La verificació de la llei de control en simulació, utilitzant el model dinàmic no lineal del sistema en llaç tancat.

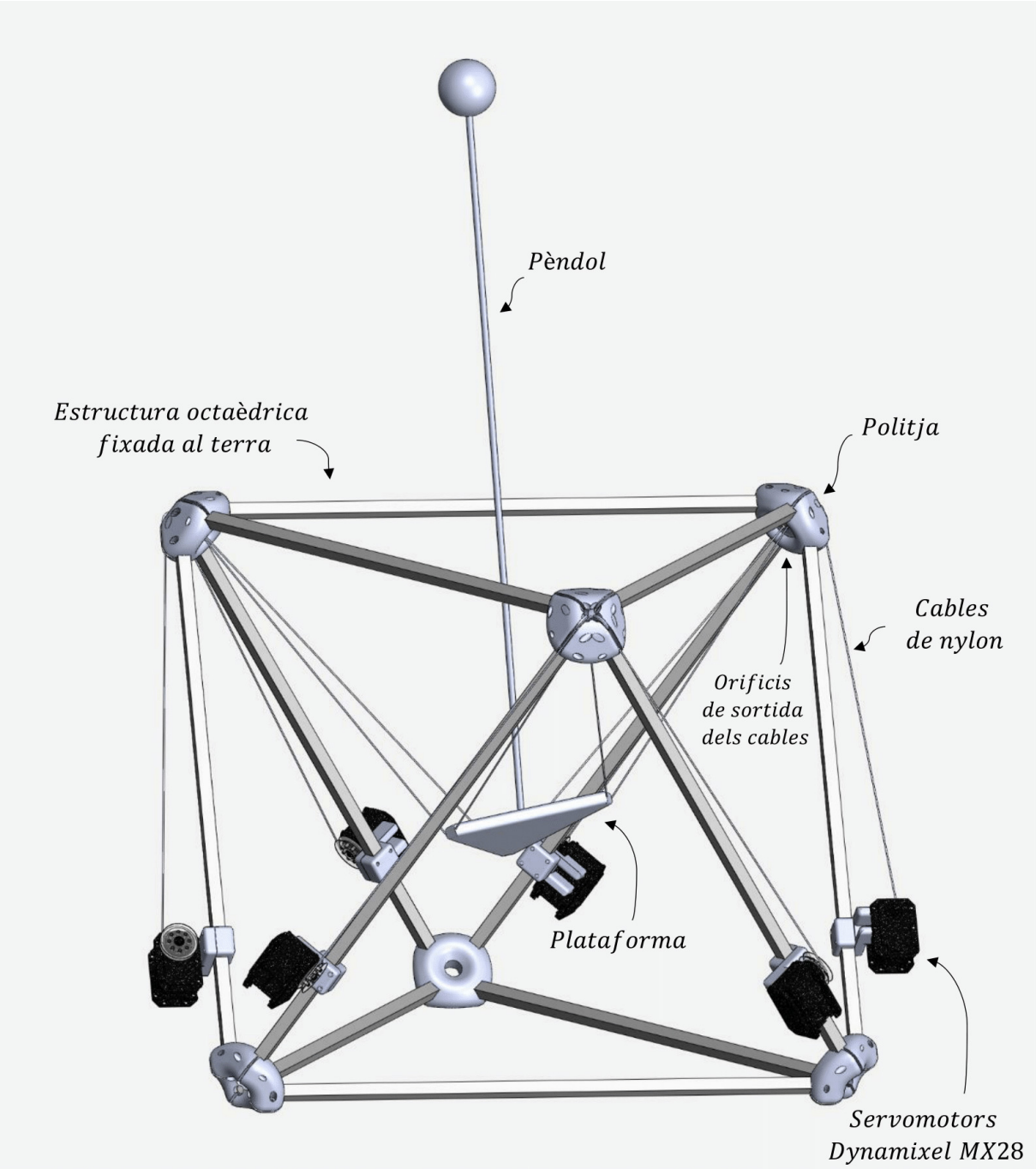


Figura 1.1: Estabilització d'un pèndol invertit mitjançant una plataforma paral·lela actuada per cables. Els servomotors han de realitzar accions correctores tota l'estona, garantint que el pèndol es manté en posició vertical, encara que se li apliquin petites pertorbacions de força no modelitzades.

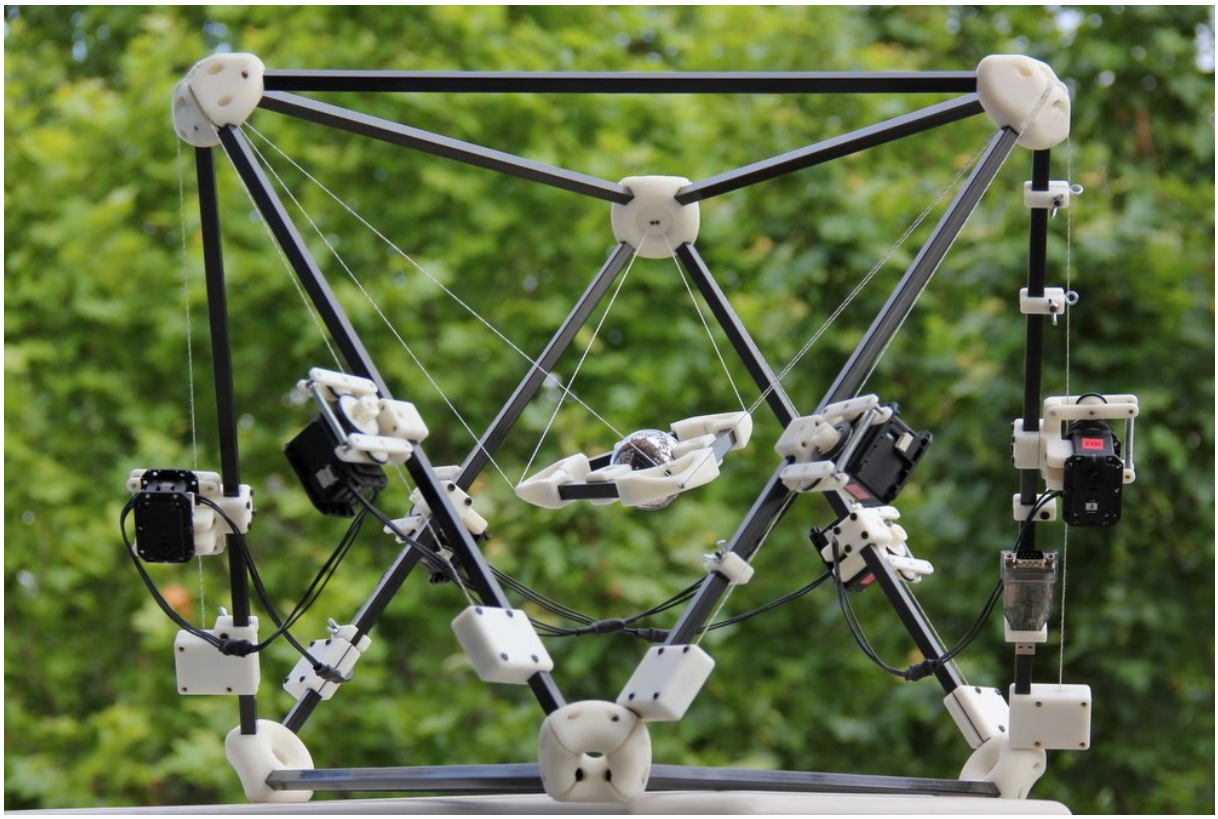


Figura 1.2: Robot paral·lel Hexacrane amb el que es vol realitzar el control del pèndol invertit.

En canvi, no es resoldran els següents aspectes d'implementació, que es deixen per a un projecte futur:

1. El disseny de sensors d'estat del sistema Hexapole.
2. La programació d'interfícies de control dels seus sensors i actuadors.
3. La implementació de la llei de control sobre el robot físic.
4. La verificació experimental del comportament d'aquesta llei.

El robot Hexacrane compta amb diverses limitacions que cal tenir en consideració en el moment d'abordar el problema de control plantejat. Primerament ens adonem que l'espai de treball en el qual podem moure i orientar la plataforma és força reduït. Aquest espai queda limitat pel fet que durant el moviment no podem perdre tensió en cap dels cables. La pèrdua de tensió suposaria pèrdua de part del control que tenim sobre la plataforma i en conseqüència la impossibilitat d'aplicar accions correctores per estabilitzar el pèndol. D'altra banda, els actuadors del robot tenen una capacitat limitada, tant pel que fa al parell com a les velocitats que

poden generar. Totes aquestes restriccions fan que l'Hexapole tingui les seves variables d'estat i d'acció limitades a prendre valors dins uns certs intervals, i el nostre objectiu serà dissenyar una llei de control que respecti aquests intervals. Per assolir aquest objectiu aplicarem la tècnica del regulador quadràtic lineal, que permet tractar aquestes limitacions de manera aproximada però efectiva a la pràctica. Donat un sistema dinàmic lineal, aquesta tècnica obté una llei de control que fa que la trajectòria seguida pel sistema cap a la configuració d'equilibri minimitzi una certa funció de cost, independentment de quines siguin les condicions inicials del sistema. Aquesta funció depèn de certs paràmetres que penalitzen els errors d'estat i d'acció, i ajustant aquests paràmetres amb procediments establerts podrem aconseguir que el controlador mantingui les variables d'estat i d'acció dins dels intervals permesos. Com en la majoria de sistemes, les equacions del moviment de l'Hexapole són no lineals, i per tant haurem de linealitzar el sistema al voltant de l'estat d'equilibri desitjat per poder aplicar aquesta tècnica de control. Aquesta llei, per tant, funcionarà bé sempre que les desviacions del pèndol respecte de l'estat d'equilibri no siguin excessives.

Pel que fa a la simulació del sistema, implementarem tota la infraestructura de funcions i *scripts* mitjançant MATLAB i Simulink. Aquests programaris ens proporcionen les eines necessàries per abordar i resoldre les tasques del projecte de forma prou senzilla. En canvi, per obtenir les equacions del moviment de l'Hexapole utilitzarem MAPLE, degut a les facilitats que ofereix aquest entorn per treballar amb expressions simbòliques, que posteriorment poden ser exportades a MATLAB i Simulink.

1.4 Hipòtesis de treball

1.4.1 Sensors d'estat

Durant aquest treball assumirem que el sistema Hexapole disposa de sensors suficients per determinar el seu estat mecànic en tot moment. Actualment, l'Hexacrane disposa de codificadors angulars magnètics que proporcionen la posició i velocitat angulars de l'eix de sortida dels seus motors amb una precisió de $0,088°$ i $1,347°/s$ respectivament. S'estima que aquests sensors són suficients per determinar la configuració i velocitat de la plataforma mòbil del robot amb prou precisió, utilitzant els mètodes de cinemàtica directa explicats als Annexos A.3 i A.4. El pèndol de l'Hexapole, però, encara està en procés de sensorització, i a efectes d'aquest treball hem suposat que podrem mesurar dos angles d'Euler que proporcionin l'orientació de la seva barra respecte d'una referència absoluta fixada a terra, i les derivades temporals d'aquests angles. Els dos angles utilitzats es defineixen a la Secció 2.2, i es poden obtenir de diverses maneres, ja sigui instal·lant una unitat de mesura inercial dins el pèndol [4], un sensor d'efecte Hall a l'extrem del

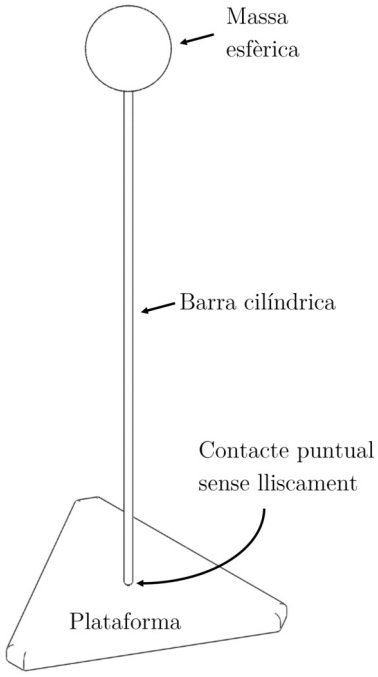


Figura 1.3: Model assumit per al pèndol i el seu contacte amb la plataforma.

pèndol [5, 6], o bé mitjançant una càmera externa i mètodes estàndar de visió per computador.

1.4.2 Model dinàmic

A efectes d'obtenir el model dinàmic de l'Hexapole hem fet les hipòtesis següents:

- Les masses dels cables i de les politges, en ser molt petites en comparació amb les de la resta d'elements, s'han considerat negligibles. Els únics elements amb massa no negligible seran la plataforma i el pèndol.
- La plataforma es modelitzarà com un sòlid rígid de geometria genèrica, i per tant amb tensor d'inèrcia també genèric.
- El pèndol, en canvi, es suposarà format per una barra cilíndrica, amb una massa esfèrica en el seu extrem.
- Suposarem que, en el seu extrem inferior, el pèndol manté un contacte puntual sense lliscament amb la plataforma (Fig. 1.3).
- Les úniques forces de fregament que es consideraran seran les degudes al fregament sec i viscós dels motors. La resta de forces de fregament, en ser força inferiors, s'han considerat

nul·les. Això inclou les forces de fregament entre els cables i les politges, les que es produeixin en el contacte pèndol-plataforma, i el fregament viscós del pèndol contra l'aire.

1.4.3 Sistema de control

Finalment, hem assumit que la freqüència a la que opera el llaç de control és prou ràpida, i que les mesures d'estat s'obtenen en temps molt breus, de manera que el sistema global format pel robot i el seu controlador es pugui assimilar a un sistema de temps continu. Aquesta hipòtesi ve suportada per experiments que s'han fet a l'IRI amb els servomotors de l'Hexacrane, i pel fet que la llei de control que obtindrem és molt senzilla, de manera que implementant-la sobre un computador prou potent s'hauria de poder avaluar en pocs milisegons.

1.5 Estructura de la memòria

La resta de la memòria seguirà la següent estructura.

- En el capítol 2 explicarem com s'ha obtingut el model mecànic de l'Hexapole. Per *model mecànic* entenem les equacions del moviment que relacionen les acceleracions de l'Hexapole amb els parells efectuats pels seus motors en el seu eix de sortida. Aquestes equacions, com veurem, negligeixen els efectes electromecànics interns dels motors, però serien ja suficients per a dissenyar lleis de control que estabilitzessin l'Hexapole en el cas que els motors acceptessin consignes de parell i les poguessin seguir de manera prou ràpida i acurada.
- En el capítol 3 ampliarem el model mecànic de l'Hexapole amb les equacions que descriuen el comportament electrodinàmic dels seus motors, tenint en compte, també, els efectes de fregament sec i viscós que aquests generen. L'objectiu és arribar a un model *electromecànic* global que descrigui la relació entre les acceleracions de l'Hexapole i els voltatges d'excitació dels seus motors. Aquest model resultarà de gran ajuda en el nostre cas, ja que els servomotors de l'Hexapole no admeten consignes de parell, però sí de voltatge d'excitació.
- En el capítol 4 detallarem com es poden linealitzar els models mecànic i electromecànic de l'Hexapole al voltant de l'estat d'equilibri desitjat (amb la plataforma parada en una configuració especificada, i amb el pèndol invertit en posició vertical).
- En el capítol 5 resumirem la tècnica del regulador quadràtic lineal i veurem com, utilitzant-la sobre els models lineals obtinguts en el capítol 4, podem dissenyar lleis de control que mantinguin l'Hexapole en l'estat d'equilibri esmentat. Detallarem també l'heurístic

de Bryson, que permet ajustar el comportament d'aquestes lleis, fent-les més o menys agressives segons quines siguin les nostres necessitats.

- En el capítol 6 verificarem la coherència dels models dinàmics desenvolupats, i el comportament de les lleis de control associades, realitzant diverses proves en simulació.
- En el capítol 7, finalment, descriurem les principals conclusions d'aquest treball, i enumerarem diversos aspectes que s'haurien de tractar en les seves extensions futures.

La memòria compta amb tres annexos que descriuen, respectivament, les relacions cinemàtiques de l'Hexacrane, el pressupost del projecte i els programes MAPLE i MATLAB implementants.

1.6 Convenis notacionals

En tota la memòria denotarem les magnituds escalars amb font normal, i les vectorials amb font en negreta. Els vectors es denotaran amb lletres minúscules, i les matrius amb lletres majúscules. Així, x denotarà un escalar, $\boldsymbol{x}$ un vector i $\mathbf{X}$ una matriu. Quan $\boldsymbol{x}$ sigui un vector que apareix en una operació, s'assumirà que és un vector columna.

Quan haguem de fer explícites les components d'un vector, normalment les escriurem en columna i entre claudàtors

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \tag{1.1}$$

però a vegades usarem també la notació $\boldsymbol{x} = (x_1, \cdots, x_n)$ per compacitat, que se suposarà equivalent a l'expressió de l'Eq. (1.1). De manera similar, usarem la notació $\boldsymbol{x} = (\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n)$ per referir-nos a la concatenació lineal dels vectors $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n$. Aquesta notació se suposarà equivalent a l'expressió $\boldsymbol{x} = \left[\boldsymbol{x}_1^T, \cdots, \boldsymbol{x}_n^T\right]^T$, on els vectors $\boldsymbol{x}_i$ es veuen com vectors columna.

Per agilitzar el text, sovint usarem claus horitzontals per definir nous símbols o fer aclariments en les expressions que escriurem. Per exemple, quan escrivim

$$\underbrace{\text{expressió}}_{\text{símbol}}$$

voldrem dir que, en endavant, el símbol especificat denotarà l'expressió indicada.

2

Model mecànic de l'Hexapole

En aquest capítol obtindrem el model mecànic de l'Hexapole. Aquest model proporciona la relació entre les acceleracions del sistema, i els parells efectuats pels seus motors al seu eix de sortida. El model s'obtindrà mitjançant la metodologia de les equacions de Lagrange. Començarem explicant l'expressió general d'aquestes equacions i la interpretació dels seus termes, i triarem unes coordenades d'estat i d'acció que facilitin la seva formulació en el cas de l'Hexapole. Seguidament obtindrem les expressions de les energies cinètica i potencial del sistema, i de la força generalitzada d'actuació. L'objectiu és arribar al que es coneix com la forma explícita de primer ordre de les equacions del moviment: una expressió de la forma $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u})$, on $\boldsymbol{x}$ és el vector d'estat del sistema, $\boldsymbol{u}$ és el vector que modelitza les accions dels actuadors, i $f(\boldsymbol{x}, \boldsymbol{u})$ és una funció que proporciona $\dot{\boldsymbol{x}} = \frac{d\boldsymbol{x}}{dt}$ en funció de $\boldsymbol{x}$ i de $\boldsymbol{u}$. Aquesta expressió és necessària perquè és l'assumida pels mètodes de simulació, linealització i control de sistemes que utilitzarem.

2.1 Equacions de Lagrange

L'estat mecànic d'un sistema robòtic es pot descriure mitjançant un vector d'estat $\boldsymbol{x} = (\boldsymbol{q}, \dot{\boldsymbol{q}})$ on $\boldsymbol{q}$ és un vector de $n_{\boldsymbol{q}}$ coordenades generalitzades que determinen la configuració del sistema en un cert instant de temps, i $\dot{\boldsymbol{q}}$ és la seva derivada temporal. Hi ha llibertat total en l'elecció de la forma i dimensió de $\boldsymbol{q}$, però $\boldsymbol{q}$ ha de descriure una i només una configuració del sistema.

Si denotem per $T(\boldsymbol{q}, \dot{\boldsymbol{q}})$ i $U(\boldsymbol{q})$ les funcions que donen, respectivament, l'energia cinètica i potencial del sistema en funció de $\boldsymbol{q}$ i $\dot{\boldsymbol{q}}$, l'expressió general de les equacions de Lagrange pren la forma [7]

$$\frac{d}{dt}\frac{\partial T}{\partial \dot{\boldsymbol{q}}} - \frac{\partial T}{\partial \boldsymbol{q}} + \frac{\partial U}{\partial \boldsymbol{q}} = \boldsymbol{\mathcal{F}}_a, \tag{2.1}$$

on T i U se suposen obtingudes en una referència Galileana. Cadascun dels termes de l'equació modelitza un conjunt de forces que intervenen en la dinàmica del sistema. Així, el terme, $\frac{d}{dt}\frac{\partial T}{\partial \dot{\boldsymbol{q}}} - \frac{\partial T}{\partial \boldsymbol{q}}$ engloba les forces generalitzades d'inèrcia de d'Alembert (canviades de signe segons

el conveni amb el qual hem escrit les equacions). Com que els únics elements de massa no negligible són la plataforma i el pèndol (vegi's la secció 1.4), en aquest terme només caldrà comptabilitzar l'energia cinètica d'aquests dos elements. El terme, $\frac{\partial U}{\partial \boldsymbol{q}}$, per la seva banda, engloba les forces generalitzades conservatives (també canviades de signe). Com que el robot Hexacrane no incorpora molles, l'única força conservativa que intervindrà és la gravetat. Finalment el membre dret de la igualtat, $\boldsymbol{\mathcal{F}}_a$, agrupa les forces generalitzades de la resta de forces no d'enllaç, que en general són les forces de fregament i les forces d'actuació del sistema. Com que les úniques forces de fregament que modelitzarem són les internes als motors, i aquests no es tindran en compte fins al Capítol 3, en aquest capítol $\boldsymbol{\mathcal{F}}_a$ modelitzarà només les forces d'actuació.

Tot seguit veurem els sistemes de referència, les bases vectorials i la notació principal amb què treballarem, i després les coordenades d'estat $\boldsymbol{x} = (\boldsymbol{q}, \dot{\boldsymbol{q}})$ i d'acció $\boldsymbol{u}$ que utilitzarem per formular les equacions de Lagrange.

2.2 Referències, bases vectorials, i notació principal

Definirem dos sistemes de referència, esquematitzats a la Fig. 2.1:

- El sistema de referència absolut, $\mathcal{F}_{abs}$, que és solidari amb el terra i té l'origen en un punt O predeterminat.
- El sistema de referència relatiu, $\mathcal{F}_{plat}$, que és solidari amb la plataforma i té l'origen al seu centre de masses, G_{plat}.

Cadascun d'aquests sistemes de referència duu una base ortonormal associada, utilitzada per expressar magnituds vectorials relatives al sistema en qüestió. Aquestes bases, també esquematitzades a la Fig. 2.1, són:

- La base B_{abs} solidària amb $\mathcal{F}_{abs}$, amb versors dirigits en les direccions x, y, z.
- La base B_{plat} solidària amb $\mathcal{F}_{plat}$, amb versors dirigits en les direccions x', y', z'.

Per orientar el pèndol en relació a B_{abs} definirem també una base vectorial B_{pole} fixa al pèndol, de versors dirigits en les direccions x''', y''' i z''' indicades a la Fig. 2.1 (el versor en z''' està alineat amb la barra del pèndol, i els altres dos versors apunten en direccions ortogonals a aquesta).

D'ara en endavant, denotarem per $\boldsymbol{p} = (x, y, z)$ la posició de G_{plat} relativa a $\mathcal{F}_{abs}$ expressada en la base B_{abs}. També suposarem que ψ, θ, φ denoten els tres angles d'Euler que orienten B_{plat} respecte de B_{abs}, utilitzant rotacions successives sobre els eixos x, y, i z. Això vol dir que la

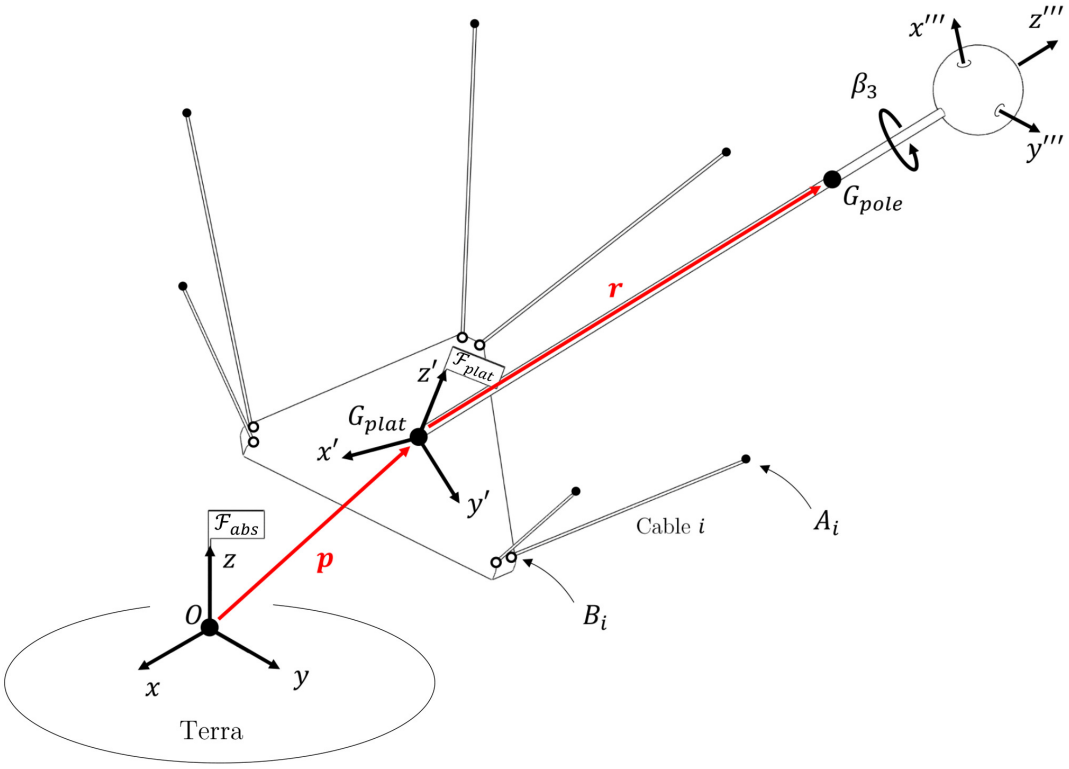


Figura 2.1: Sistemes de referència, bases vectorials i nomenclatura principal utilitzada en l'elaboració del model dinàmic de l'Hexapole.

matriu de rotació $\mathbf{R}$ que orienta B_{plat} respecte de B_{abs} vindrà donada per

$$\mathbf{R}_{plat} = \mathbf{R}_x(\psi) \cdot \mathbf{R}_y(\theta) \cdot \mathbf{R}_z(\varphi),$$

on $\mathbf{R}_a(\alpha)$ denota la matriu de rotació d'angle α al voltant de l'eix a. L'expressió detallada de $\mathbf{R}_{plat}$ es pot veure a l'Annex A.1.

Sovint agruparem els angles d'Euler de la plataforma en una tupla $\boldsymbol{\alpha} = (\psi, \theta, \varphi)$, i per tant la configuració de la plataforma vindrà donada per $\boldsymbol{t} = (\boldsymbol{p}, \boldsymbol{\alpha})$.

Tal i com hem fet amb B_{plat}, orientarem la base B_{pole} respecte de la base B_{abs} amb tres angles d'Euler $\beta_1, \beta_2, \beta_3$, de manera que la matriu de rotació $\mathbf{R}_{pole}$ que orienta B_{pole} respecte de B_{abs} serà (Fig. 2.2):

$$\mathbf{R}_{pole} = \mathbf{R}_x(\beta_1) \cdot \mathbf{R}_y(\beta_2) \cdot \mathbf{R}_z(\beta_3)$$

Si bé és cert que el pèndol pot adoptar una orientació arbitrària a l'espai, convé fer notar que l'angle β_3 és de fet irrellevant. Fixem-nos que el pèndol és un rotor simètric respecte de l'eix longitudinal de la barra, i la plataforma només pot transmetre forces (i no parells) al pèndol,

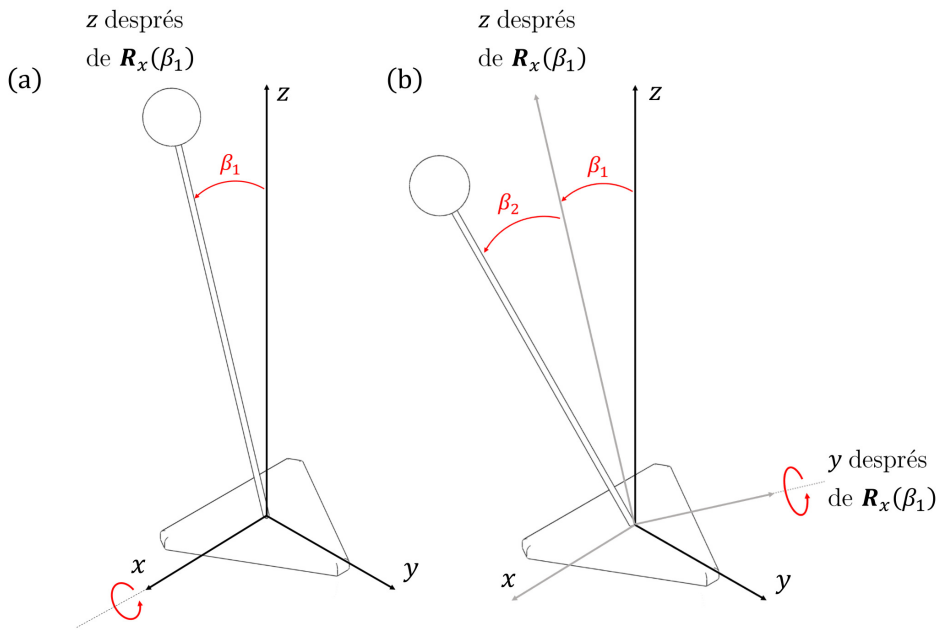


Figura 2.2: Orientació del pèndol respecte de la base B_{abs} (indicada amb els eixos negres). **(a)** Rotació que defineix l'angle β_1 respecte de l'eix x. **(b)** Rotació que defineix l'angle β_2 respecte de l'eix y després del gir de β_1. La tercera rotació d'angle β_3, que és la que es produiria al voltant de la barra, ja no s'indica.

perquè el contacte pèndol-plataforma s'ha suposat puntual. Això vol dir que les forces que actuen sobre el pèndol (la de contacte amb la plataforma i la del pes del pèndol) no poden generar cap parell que faci variar β_3. Per tant si a l'instant inicial tenim $\dot{\beta}_3 = 0$, llavors β_3 ha de romandre constant tota l'estona. Degut a això, al llarg del treball suposarem que $\beta_3 = 0$ i $\dot{\beta}_3 = 0$ al llarg del temps, amb la qual cosa serà

$$\mathbf{R}_{pole} = \mathbf{R}_x(\beta_1) \cdot \mathbf{R}_y(\beta_2).$$

A més, suposarem que β_1 i β_2 són els únics angles mesurats pels sensors del pèndol.

A banda de tot el que hem dit, denotarem per G_{pole} el centre de masses del pèndol, per $\boldsymbol{r}$ el vector que va des de G_{plat} a G_{pole} i per l_{pole} el seu mòdul. Pel que fa als tensors d'inèrcia utilitzats, a més, $\mathbf{I}_{plat}$ designarà el tensor central d'inèrcia de la plataforma en base B_{plat}, i $\mathbf{I}_{pole}$ el tensor central d'inèrcia del pèndol en base B_{pole}. Finalment, a la Taula 2.1 s'inclouen les coordenades dels punts A_i i B_i indicats a la Fig. 2.1. Els punts A_i indiquen els orificis de sortida dels cables a l'estructura octaèdrica externa (Fig. 1.1), i estan expressats en referència $\mathcal{F}_{abs}$ (en base B_{abs}). Els punts B_i són els d'ancoratge dels cables amb la plataforma, i estan expressats en la referència $\mathcal{F}_{plat}$ (en base B_{plat}). El vector de posició de A_i a $\mathcal{F}_{abs}$ i en base B_{abs} es denotarà

Terra	Plataforma
$A_1 = (-231.62, -136.18, 0)$	$B_1 = (0, -89.15, 0)$
$A_2 = (231.62, -136.18, 0)$	$B_2 = (0, -89.15, 0)$
$A_3 = (233.74, -132.50, 0)$	$B_3 = (77.21, 44.57, 0)$
$A_4 = (2.13, 268.67, 0)$	$B_4 = (77.21, 44.57, 0)$
$A_5 = (-2.13, 268.67, 0)$	$B_5 = (-77.21, 44.57, 0)$
$A_6 = (-233.74, -132.50, 0)$	$B_6 = (-77.21, 44.57, 0)$

Taula 2.1: Coordenades de A_i i B_i en mm, expressades en $\mathcal{F}_{abs}$ i $\mathcal{F}_{plat}$ respectivament.

per $\boldsymbol{a}_i$. Anàlogament, el vector de posició de B_i a $\mathcal{F}_{plat}$ i en base B_{plat} es denotarà per $\boldsymbol{b}_i$.

2.3 Selecció de coordenades d'estat i d'acció

Pel que fa a la selecció de coordenades d'estat $\boldsymbol{x} = (\boldsymbol{q}, \dot{\boldsymbol{q}})$, si bé tenim total llibertat en la tria de $\boldsymbol{q}$, a la pràctica és preferible fixar unes coordenades que siguin directament mesurables pels sensors del sistema, ja que així reduirem el nombre de càlculs que caldrà fer en cada iteració del llaç de control. Quan les coordenades d'estat no són directament mesurables, en canvi, cal fer operacions per obtenir-les a partir d'altres variables mesurades, reduint la freqüència del llaç de control i empobrint, potencialment, la resposta dinàmica del sistema.

Com ja hem explicat anteriorment, el robot Hexacrane disposa de codificadors angulars que proporcionen els angles dels seus sis motors, $\gamma_1, \cdots, \gamma_6$, i les seves respectives velocitats angulars, $\dot{\gamma}_1, \cdots, \dot{\gamma}_6$. Per altra banda, hem suposat que el pèndol està sensoritzat de manera que en coneixem els angles d'Euler β_1 i β_2 indicats a la Fig. 2.2 en tot moment. Per tant, l'ideal fora triar el vector d'estat

$$\boldsymbol{x} = (\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})$$

on

$$\boldsymbol{\theta} = (\underbrace{\gamma_1, \ldots, \gamma_6}_{\boldsymbol{\gamma}}, \underbrace{\beta_1, \beta_2}_{\boldsymbol{\beta}})$$

Aquest vector no inclou l'angle β_3 perquè, com ja hem explicat, aquest angle és irrellevant a efectes de controlar el pèndol en posició invertida.

És relativament fàcil de veure, però, que amb les anteriors coordenades no és possible obtenir expressions analítiques per a les energies cinètica i potencial del sistema. Aquestes expressions depenen de la configuració de la plataforma, que no es pot expressar en funció de $\boldsymbol{\gamma} = (\gamma_1, \cdots, \gamma_6)$ perquè el problema cinemàtic directe de l'Hexacrane no té solució tancada. Per

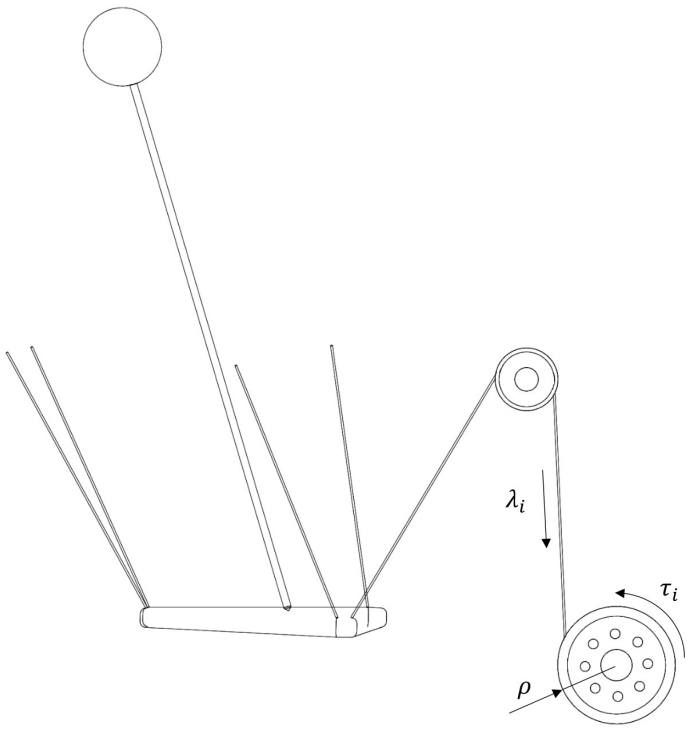


Figura 2.3: Definició dels sentits positius de les tensions dels cables λ_i i dels parells motors τ_i. Cada servomotor té muntada a l'eix de sortida una politja de radi ρ, i per tant $\tau_i = \lambda_i \cdot \rho$.

esquivar aquest inconvenient obtindrem primer les equacions de Lagrange utilitzant el vector d'estat alternatiu

$$\boldsymbol{x} = (\boldsymbol{q}, \dot{\boldsymbol{q}})$$

on

$$\boldsymbol{q} = (\underbrace{x, y, z}_{\boldsymbol{p}}, \underbrace{\psi, \theta, \varphi}_{\boldsymbol{\alpha}}, \underbrace{\beta_1, \beta_2}_{\boldsymbol{\beta}})$$

i després aplicarem un canvi de variable per reescriure les equacions en les coordenades $\boldsymbol{x} = (\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})$. Serà sobre aquestes darreres equacions que dissenyarem les lleis de control estabilitzadores de l'Hexapole.

Per facilitar les explicacions, d'ara endavant ens referirem a les coordenades $\boldsymbol{x} = (\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})$ com a *coordenades motor*, i a les coordenades $\boldsymbol{x} = (\boldsymbol{q}, \dot{\boldsymbol{q}})$ com a *coordenades plataforma*. A més, sovint usarem $\boldsymbol{x}_\theta$ o $\boldsymbol{x}_q$ per distingir unes coordenades de les altres. És a dir, utilitzarem

$$\boldsymbol{x}_\theta = (\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}),$$

o bé

$$\boldsymbol{x}_q = (\boldsymbol{q}, \dot{\boldsymbol{q}})$$

segons escaigui.

Pel que fa a les coordenades d'acció, fixem-nos que aquestes han de correspondre als senyals de consigna que donem als actuadors. En aquest capítol suposarem que els motors poden rebre consignes de parell i per tant el vector d'actuació tindrà la forma

$$\boldsymbol{u} = (\tau_1, \ldots, \tau_6),$$

on τ_i és el parell efectuat per l'i-èssim motor al seu eix de sortida. El criteri de signes que s'ha utilitzat per definir els sentits dels parells motors τ_i i les tensions dels cables t_i es pot veure a la Fig. 2.3. La relació entre aquestes dues variables ve donada per

$$\tau_i = \rho \cdot \lambda_i \tag{2.2}$$

on ρ és el radi de la politja connectada al motor.

Val la pena anticipar, però, que quan al Capítol 3 ampliem el model dinàmic per tenir en compte els motors, redefinirem $\boldsymbol{u}$ com

$$\boldsymbol{u} = (v_1, \ldots, v_6),$$

on v_i és el voltatge d'excitació de l'i-èssim motor. Per distingir un vector d'accions de l'altre, sovint usarem

$$\boldsymbol{u}_\tau = (\tau_1, \ldots, \tau_6)$$

per referir-nos als parells motor, i

$$\boldsymbol{u}_v = (v_1, \ldots, v_6)$$

per referir-nos als de voltatges d'excitació.

2.4 Model en coordenades plataforma

Amb la notació anterior, podem ja escriure les expressions de les energies cinètica i potencial del sistema en coordenades plataforma. El càlcul d'aquestes energies es farà en la referència $\mathcal{F}_{abs}$ fixa al terra, que considerarem Galileana als efectes d'aquest treball.

2.4.1 Energia cinètica

L'energia cinètica T del sistema es pot escriure com

$$T = T_{plat} + T_{pole},$$

on T_{plat} i T_{pole} són l'energia cinètica de la plataforma i del pèndol, respectivament. En ser de massa negligible, la resta d'elements del robot no contribueixen a l'expressió de T.

Desglossem ara cada un dels termes. Per una banda, T_{plat} es pot expressar com

$$T_{plat} = \underbrace{\frac{1}{2} m_{plat} |\boldsymbol{v}_{G_{plat}}|^2}_{\textit{Energia cinètica de translació}} + \underbrace{\frac{1}{2} \boldsymbol{\omega}_{plat}^T \mathbb{I}_{plat} \boldsymbol{\omega}_{plat}}_{\textit{Energia cinètica de rotació}}, \tag{2.3}$$

on $\boldsymbol{v}_{G_{plat}}$ és la velocitat absoluta del centre de masses G_{plat}, $\boldsymbol{\omega}_{plat}$ és la velocitat angular absoluta de la plataforma, $\mathbb{I}_{plat}$ és el tensor central d'inèrcia de la plataforma, i tots tres elements es suposen expressats en la base B_{abs}. Tenint en compte les definicions de la Secció 2.2, podem escriure

$$\boldsymbol{v}_{G_{plat}} = \dot{\boldsymbol{p}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} \tag{2.4}$$

i

$$\mathbb{I}_{plat} = \mathbf{R}_{plat} \mathbf{I}_{plat} \mathbf{R}_{plat}^T. \tag{2.5}$$

Per altra banda, a partir dels resultats de l'Annex A.2, la velocitat angular de la plataforma es pot expressar com

$$\boldsymbol{\omega}_{plat} = \mathbf{A}_m \dot{\boldsymbol{\alpha}}. \tag{2.6}$$

Substituint (2.4), (2.5) i (2.6) a (2.3) obtenim

$$T_{plat} = \frac{1}{2} m_{plat} \dot{\boldsymbol{p}}^T \dot{\boldsymbol{p}} + \frac{1}{2} \left(\mathbf{A}_m \dot{\boldsymbol{\alpha}}\right)^T \mathbf{R}_{plat} \mathbf{I}_{plat} \mathbf{R}_{plat}^T \left(\mathbf{A}_m \dot{\boldsymbol{\alpha}}\right). \tag{2.7}$$

Ens interessa però tenir l'Equació (2.7) en la forma $T_{plat} = \frac{1}{2} \dot{\boldsymbol{q}}^T \mathbf{H}_{plat} \dot{\boldsymbol{q}}$. Per aconseguir-ho, fixem-nos que d'una banda tenim

$$\dot{\boldsymbol{p}} = \underbrace{\begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} \end{bmatrix}}_{\mathbf{J}_{v_{plat}}} \begin{bmatrix} \dot{\boldsymbol{p}} \\ \dot{\boldsymbol{\alpha}} \\ \dot{\boldsymbol{\beta}} \end{bmatrix} = \mathbf{J}_{v_{plat}} \dot{\boldsymbol{q}}, \tag{2.8}$$

on $\dot{\boldsymbol{\beta}} = (\dot{\beta}_1, \dot{\beta}_2)$, i d'altra banda també tenim

$$\boldsymbol{\omega}_{plat} = \mathbf{A}_m \dot{\boldsymbol{\alpha}} = \underbrace{\begin{bmatrix} \mathbf{0} & \mathbf{A}_m & \mathbf{0} \end{bmatrix}}_{\mathbf{J}_{\omega_{plat}}} \begin{bmatrix} \dot{\boldsymbol{p}} \\ \dot{\boldsymbol{\alpha}} \\ \dot{\boldsymbol{\beta}} \end{bmatrix} = \mathbf{J}_{\omega_{plat}} \dot{\boldsymbol{q}}. \tag{2.9}$$

Per tant, si introduïm les equacions (2.8) i (2.9) a (2.7) obtenim

$$T_{plat} = \frac{1}{2} m_{plat} \left(\mathbf{J}_{v_{plat}} \dot{\boldsymbol{q}}\right)^T \left(\mathbf{J}_{v_{plat}} \dot{\boldsymbol{q}}\right) + \frac{1}{2} \left(\mathbf{J}_{\omega_{plat}} \dot{\boldsymbol{q}}\right)^T \mathbf{R}_{plat} \mathbf{I}_{plat} \mathbf{R}_{plat}^T \left(\mathbf{J}_{\omega_{plat}} \dot{\boldsymbol{q}}\right).$$

Si ara agrupem termes i ordenem, arribem a l'expressió

$$T_{plat} = \frac{1}{2} \dot{\boldsymbol{q}}^T \underbrace{\left[m_{plat} \mathbf{J}_{v_{plat}}^T \mathbf{J}_{v_{plat}} + \mathbf{J}_{\omega_{plat}}^T \mathbf{R}_{plat} \mathbf{I}_{plat} \mathbf{R}_{plat}^T \mathbf{J}_{\omega_{plat}} \right]}_{\mathbf{H}_{plat}} \dot{\boldsymbol{q}},$$

de manera que

$$T_{plat} = \frac{1}{2} \dot{\boldsymbol{q}}^T \mathbf{H}_{plat} \dot{\boldsymbol{q}}, \tag{2.10}$$

que ja és l'expressió que buscàvem.

Anàlogament a l'Eq. (2.3), per al pèndol podem escriure

$$T_{pole} = \underbrace{\frac{1}{2} m_{pole} |\boldsymbol{v}_{G_{pole}}|^2}_{\textit{Energia cinètica de translació}} + \underbrace{\frac{1}{2} \boldsymbol{\omega}_{pole}^T \mathbb{I}_{pole} \boldsymbol{\omega}_{pole}}_{\textit{Energia cinètica de rotació}}. \tag{2.11}$$

En aquesta expressió $\boldsymbol{v}_{G_{pole}}$ és la velocitat absoluta del centre de masses del pèndol, $\boldsymbol{\omega}_{pole}$ és la velocitat angular absoluta del pèndol, i $\mathbb{I}_{pole}$ és el tensor central d'inèrcia del pèndol, essent tots tres elements expressats en la base B_{abs}. Tenint en compte les definicions de la Secció 2.2,

$$\mathbb{I}_{pole} = \mathbf{R}_{pole} \mathbf{I}_{pole} \mathbf{R}_{pole}^T. \tag{2.12}$$

D'altra banda, per escriure l'Eq. (2.11) és necessari conèixer el valor de $\boldsymbol{v}_{G_{pole}}$. Aquesta velocitat es pot obtenir a partir de l'expressió

$$\boldsymbol{v}_{G_{pole}} = \frac{\mathrm{d}}{\mathrm{d}t} \boldsymbol{OG}_{pole}(\boldsymbol{q}) = \underbrace{\frac{\partial}{\partial \boldsymbol{q}} \boldsymbol{OG}_{pole}(\boldsymbol{q})}_{\mathbf{D}_{pole}} \dot{\boldsymbol{q}} = \mathbf{D}_{pole} \dot{\boldsymbol{q}}, \tag{2.13}$$

on el vector $\boldsymbol{OG}_{pole}$ s'obté a partir dels vectors $\boldsymbol{p}$ i $\boldsymbol{r}$ amb l'equació

$$\boldsymbol{OG}_{pole}(\boldsymbol{q}) = \underbrace{\begin{bmatrix} x \\ y \\ z \end{bmatrix}}_{\boldsymbol{p}} + \underbrace{\mathbf{R}_x(\beta_1)\mathbf{R}_y(\beta_2)\begin{bmatrix} 0 \\ 0 \\ l_{pole} \end{bmatrix}}_{\boldsymbol{r}} \tag{2.14}$$

i D_{pole} es pot obtenir amb l'ajut de MAPLE. Passem ara a calcular la velocitat angular del pèndol $\boldsymbol{\omega}_{pole}$. Com que a la Secció 2.2 hem suposat que $\dot{\beta}_3 = 0$, podem dir que

$$\boldsymbol{\omega}_{pole} = \begin{bmatrix} 0 \\ 0 \\ \dot{\beta}_1 \end{bmatrix} + \mathbf{R}_x(\beta_1)\begin{bmatrix} 0 \\ \dot{\beta}_2 \\ 0 \end{bmatrix} = \underbrace{\begin{bmatrix} & & 0 & \\ \mathbf{0} & \mathbf{0} & 0 & [\mathbf{R}_x(\beta_1)]_2 \\ & & 1 & \end{bmatrix}}_{\mathbf{J}_{\boldsymbol{\omega}_{pole}}} \underbrace{\begin{bmatrix} \dot{\boldsymbol{p}} \\ \dot{\boldsymbol{\alpha}} \\ \dot{\beta}_1 \\ \dot{\beta}_2 \end{bmatrix}}_{\dot{\boldsymbol{q}}} = \mathbf{J}_{\boldsymbol{\omega}_{pole}}\dot{\boldsymbol{q}}, \tag{2.15}$$

on $[\mathbf{R}_x(\beta_1)]_2$ denota la segona columna de la matriu $\mathbf{R}_x(\beta_1)$. Si substituïm les Eqs. (2.12), (2.13) i (2.15) a (2.11), arribem a la l'equació

$$T_{pole} = \frac{1}{2}m_{pole}(\mathbf{D}_{pole}\dot{\boldsymbol{q}})^T\mathbf{D}_{pole}\dot{\boldsymbol{q}} + \frac{1}{2}(\mathbf{J}_{\boldsymbol{\omega}_{pole}}\dot{\boldsymbol{q}})^T\mathbf{R}_{pole}\mathbf{I}_{pole}\mathbf{R}_{pole}^T(\mathbf{J}_{\boldsymbol{\omega}_{pole}}\dot{\boldsymbol{q}}). \tag{2.16}$$

De manera anàloga al que hem fet amb la plataforma ens interessa escriure l'equació (2.16) en la forma $T_{pole} = \frac{1}{2}\dot{\boldsymbol{q}}^T\mathbf{H}_{pole}\dot{\boldsymbol{q}}$. Per tant, si agrupem i ordenem els termes obtenim

$$T_{pole} = \frac{1}{2}\dot{\boldsymbol{q}}^T \underbrace{\left[m_{pole}\mathbf{D}_{pole}^T\mathbf{D}_{pole} + \mathbf{J}_{\boldsymbol{\omega}_{pole}}^T\mathbf{R}_{pole}\mathbf{I}_{pole}\mathbf{R}_{pole}^T\mathbf{J}_{\boldsymbol{\omega}_{pole}}\right]}_{\mathbf{H}_{pole}}\dot{\boldsymbol{q}},$$

que proporciona l'expressió que buscàvem

$$T_{pole} = \frac{1}{2}\dot{\boldsymbol{q}}^T\mathbf{H}_{pole}\dot{\boldsymbol{q}}. \tag{2.17}$$

Finalment, sumant les Eqs. (2.10) i (2.17) obtenim l'energia cinètica total del sistema,

$$T = T_{plat} + T_{pole} = \frac{1}{2}\dot{\boldsymbol{q}}^T(\mathbf{H}_{plat} + \mathbf{H}_{pole})\dot{\boldsymbol{q}}, \tag{2.18}$$

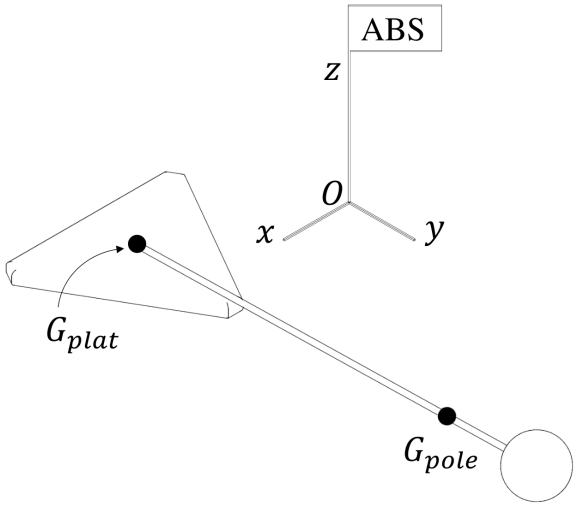


Figura 2.4: Definició de l'origen de potencial per la plataforma i el pèndol.

on $\mathbf{H} = \mathbf{H}_{plat} + \mathbf{H}_{pole}$ és l'anomenada matriu de massa del sistema, que en resum té l'expressió

$$\begin{aligned}\mathbf{H} = & \underbrace{\left[m_{plat}\mathbf{J}_{v_{plat}}^T\mathbf{J}_{v_{plat}} + \mathbf{J}_{\omega_{plat}}^T\mathbf{R}_{plat}\mathbf{I}_{plat}\mathbf{R}_{plat}^T\mathbf{J}_{\omega_{plat}}\right]}_{\mathbf{H}_{plat}} \\ & + \underbrace{\left[m_{pole}\mathbf{D}_{pole}^T\mathbf{D}_{pole} + \mathbf{J}_{\boldsymbol{\omega}_{pole}}^T\mathbf{R}_{pole}\mathbf{I}_{pole}\mathbf{R}_{pole}^T\mathbf{J}_{\boldsymbol{\omega}_{pole}}\right]}_{\mathbf{H}_{pole}}.\end{aligned} \tag{2.19}$$

2.4.2 Energia potencial

Per escriure l'expressió de l'energia potencial de l'Hexapole assumirem que l'origen de potencial es dóna quan la plataforma i el pèndol són coplanars amb el pla Oxy de $\mathcal{F}_{abs}$ (Fig. 2.4). Per tant, en una configuració genèrica $\boldsymbol{q}$ l'energia potencial d'aquests dos elements serà

$$U(\boldsymbol{q}) = -m_{plat}\ \boldsymbol{g}^T\boldsymbol{OG}_{plat} - m_{pole}\ \boldsymbol{g}^T\boldsymbol{OG}_{pole}, \tag{2.20}$$

on

$$\boldsymbol{g} = \begin{bmatrix} 0 \\ 0 \\ -9,8 \end{bmatrix} \mathrm{m/s^2}, \tag{2.21}$$

$$\boldsymbol{OG}_{plat} = \boldsymbol{p} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \tag{2.22}$$

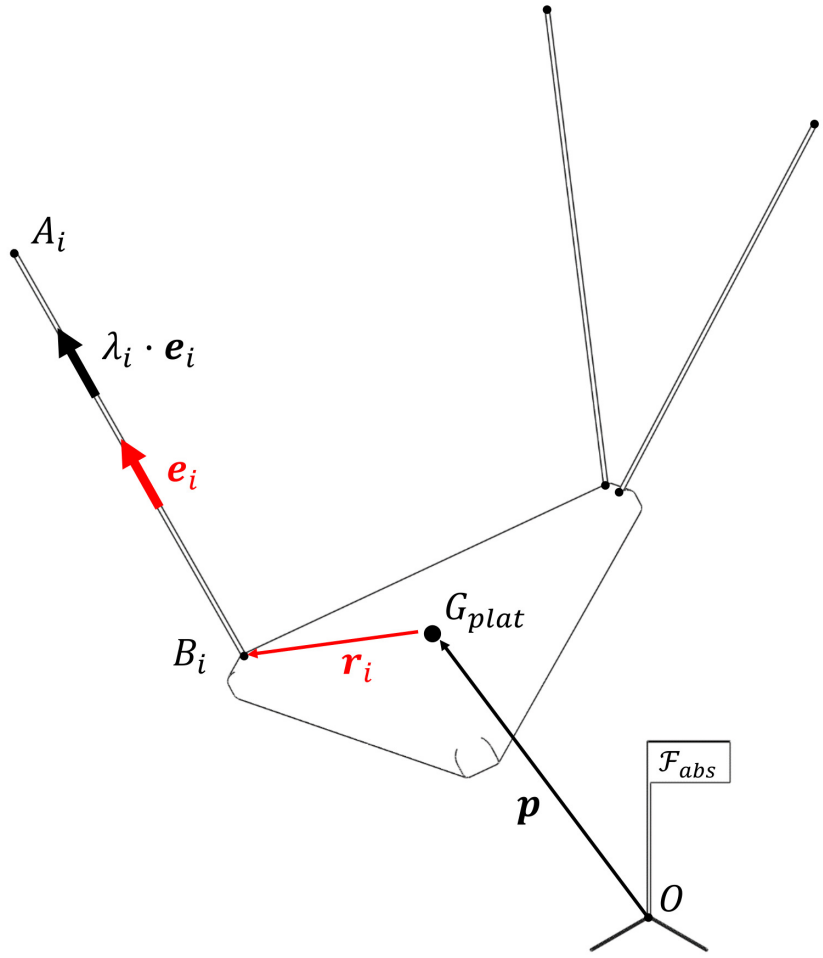


Figura 2.5: Esquema dels vectors que intervenen en el càlcul del Jacobià de forces.

i $\boldsymbol{OG}_{pole}$ ve donat per l'Eq. (2.14). Si substituïm (2.14), (2.21) i (2.22) a (2.20), i denotem per g l'acceleració de la gravetat, arribem a l'expressió que buscàvem:

$$U(\boldsymbol{q}) = m_{plat}\ g\ z - m_{pole}\ \boldsymbol{g}^T \left[\boldsymbol{p} + \mathbf{R}(\beta_1)\mathbf{R}_y(\beta_2) \begin{bmatrix} 0 \\ 0 \\ l_{pole} \end{bmatrix} \right]. \tag{2.23}$$

2.4.3 Força generalitzada d'actuació

Per escriure l'expressió de la força generalitzada d'actuació $\mathcal{F}_a$ utilitzarem el mètode de les potències virtuals. El mètode es basa en calcular la potència P_a generada per les forces d'actuació

sota una velocitat virtual $\dot{\boldsymbol{q}}$ a la que es mogui el mecanisme, i escriure-la en la forma

$$P_a = \boldsymbol{\mathcal{F}}_a^T \cdot \dot{\boldsymbol{q}}.$$

Per arribar a aquesta expressió, fixem-nos en primer lloc que

$$P_a = \hat{\mathbf{w}}_a^T \cdot \hat{\mathbf{T}}_{plat},$$

on $\hat{\mathbf{w}}_a$ és el torsor resultant de les forces que fan els cables sobre la plataforma, reduït al punt G_{plat}, i $\hat{\mathbf{T}}_{plat}$ és el torsor de velocitats de la plataforma, també reduït al punt G_{plat}. Per tant

$$\hat{\mathbf{w}}_a = \hat{\mathbf{w}}_1 + \ldots + \hat{\mathbf{w}}_6,$$

on $\hat{\mathbf{w}}_i$ és el torsor de força que exerceix el cable i sobre la plataforma. Aquest torsor té la forma

$$\hat{\mathbf{w}}_i = \begin{bmatrix} \lambda_i \boldsymbol{e}_i \\ \boldsymbol{r}_i \times \lambda_i \boldsymbol{e}_i \end{bmatrix},$$

on $\boldsymbol{e}_i$ és el vector unitari que indica la direcció i sentit de l'i-èssim cable,

$$\boldsymbol{e}_i = \frac{\boldsymbol{a}_i - \boldsymbol{p} - \mathbf{R}\boldsymbol{b}_i}{|\boldsymbol{a}_i - \boldsymbol{p} - \mathbf{R}\boldsymbol{b}_i|},$$

$\boldsymbol{r}_i$ és el vector de posició del punt d'ancoratge B_i en la referència $\mathcal{F}_{plat}$ (en base B_{abs}),

$$\boldsymbol{r}_i = \mathbf{R}\boldsymbol{b}_i,$$

i λ_i és el valor de la força que fa l'i-èssim cable sobre la plataforma. Aquestes magnituds és poden apreciar esquematitzades a la Fig. 2.5. Per tant, utilitzant les expressions anteriors (2.2) podem expressar $\hat{\mathbf{w}}_a$ de la següent manera

$$\hat{\mathbf{w}}_a = \underbrace{\begin{bmatrix} \boldsymbol{e}_1 & \cdots & \boldsymbol{e}_6 \\ \boldsymbol{r} \times \boldsymbol{e}_1 & \cdots & \boldsymbol{r}_6 \times \boldsymbol{e}_6 \end{bmatrix}}_{\mathbf{J}} \underbrace{\begin{bmatrix} \tau_1 \\ \vdots \\ \tau_6 \end{bmatrix}}_{\boldsymbol{u}_\tau} \frac{1}{\rho} = \frac{1}{\rho} \cdot \mathbf{J} \cdot \boldsymbol{u}_\tau \tag{2.24}$$

on $\mathbf{J}$ és el que anomenarem com Jacobià de forces de l'Hexacrane.

Pel que fa al torsor de velocitats, el podem obtenir fàcilment a partir del vector de velocitats

generalitzades $\dot{\boldsymbol{q}}$ i utilitzant l'Eq. (2.6):

$$\hat{\mathbf{T}}_{plat} = \begin{bmatrix} \boldsymbol{v}_{G_{plat}} \\ \boldsymbol{\omega}_{plat} \end{bmatrix} = \begin{bmatrix} \dot{\boldsymbol{p}} \\ \mathbf{A}_m \dot{\boldsymbol{\alpha}} \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_m & \mathbf{0} \end{bmatrix}}_{\mathbf{D}} \underbrace{\begin{bmatrix} \dot{\boldsymbol{p}} \\ \dot{\boldsymbol{\alpha}} \\ \dot{\boldsymbol{\beta}} \end{bmatrix}}_{\dot{\boldsymbol{q}}} = \mathbf{D} \cdot \dot{\boldsymbol{q}}. \tag{2.25}$$

Per tant, utilitzant les Eqs. (2.24), (2.25) obtenim

$$P_a = \hat{\mathbf{w}}_a^T \cdot \hat{\mathbf{T}}_{plat} = \underbrace{\frac{1}{\rho} \cdot (\mathbf{J} \cdot \boldsymbol{u}_\tau)^T \cdot \mathbf{D}}_{\boldsymbol{\mathcal{F}}_a^T} \cdot \dot{\boldsymbol{q}},$$

i la força generalitzada d'actuació és

$$\boldsymbol{\mathcal{F}}_a = \frac{1}{\rho} \cdot \mathbf{D}^T \cdot \mathbf{J} \cdot \boldsymbol{u}_\tau. \tag{2.26}$$

2.4.4 Equacions de Lagrange en la forma manipulador

En les subseccions anteriors ja hem obtingut les expressions de les energies cinètica i potencial, i de la força generalitzada d'actuació. Si ens fes falta, podríem aplicar l'equació de Lagrange (2.1) i obtindríem un sistema d'equacions diferencials que descriuria el comportament dinàmic de l'Hexapole. Tot i això en el món de la robòtica s'acostuma a treballar amb una forma especial de les equacions de Lagrange, coneguda com a *forma manipulador* [8, 9]. Aquesta forma ens facilitarà l'aïllament de les acceleracions generalitzades $\ddot{\boldsymbol{q}}$, l'obtenció de la forma de primer ordre de les equacions del moviment, i la seva posterior linealització.

La forma *manipulador* té la següent estructura

$$\mathbf{H} \cdot \ddot{\boldsymbol{q}} + \mathbf{C} \cdot \dot{\boldsymbol{q}} + \mathbf{G} = \mathbf{E} \cdot \boldsymbol{u} \tag{2.27}$$

on:

- $\mathbf{H}$ és la matriu de masses que caracteritza l'energia cinètica del sistema.
- $\mathbf{C}$ és l'anomenada matriu de Coriolis que, multiplicada per $\dot{\boldsymbol{q}}$, dóna lloc al vector amb els termes d'inèrcia de la forma $\dot{\boldsymbol{q}}_i \cdot \dot{\boldsymbol{q}}_j$ i $\dot{\boldsymbol{q}}_i^2$.
- $\mathbf{G}$ és la força generalitzada conservativa canviada de signe.
- $\mathbf{E}$ és una matriu tal que multiplicada pel vector d'accions $\boldsymbol{u}$ dóna la força generalitzada d'actuació $\boldsymbol{\mathcal{F}}_a$.

En el cas de l'Hexapole, l'expressió d'$\mathbf{H}$ ja s'ha calculat abans, i té la forma detallada a l'Eq. (2.19). La matriu de Coriolis es pot calcular de diverses maneres però, aprofitant que disposem de l'expressió simbòlica d'$\mathbf{H}$, calcularem les components de $\mathbf{C}$ mitjançant l'expressió (Murray i col. [9])

$$\mathbf{C}_{ij} = \frac{1}{2}\sum_{k=1}^{n}\left(\frac{\partial \mathbf{H}_{ij}}{\partial \boldsymbol{q}_k} + \frac{\partial \mathbf{H}_{ik}}{\partial \boldsymbol{q}_j} + \frac{\partial \mathbf{H}_{kj}}{\partial \boldsymbol{q}_i}\right)\dot{\boldsymbol{q}}_k \tag{2.28}$$

i amb l'ajut de MAPLE. El vector $\mathbf{G}$ també es pot obtenir mitjançant MAPLE tenint en compte l'Eq. (2.23) i el fet que

$$\mathbf{G}(\boldsymbol{q}) = \frac{\partial U(\boldsymbol{q})}{\partial \boldsymbol{q}}$$

Finalment, tenint en compte que en aquest capítol $\boldsymbol{u} = \boldsymbol{u}_\tau$, i identificant $\mathbf{E}\cdot\boldsymbol{u}_\tau$ amb l'expressió de l'Eq. (2.26), veiem que

$$\mathbf{E} = \frac{1}{\rho}\cdot\mathbf{D}^T\cdot\mathbf{J}.$$

2.5 Model en coordenades motor

Com ja hem indicat a la Secció 2.3, el següent pas és aplicar un canvi de variable sobre l'Eq. (2.27) per tal d'obtenir les equacions del moviment en les coordenades motor $\boldsymbol{x}_\theta = (\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})$. A tal efecte utilitzarem la relació cinemàtica

$$\dot{\boldsymbol{\theta}} = \mathbf{J}_i\cdot\dot{\boldsymbol{q}} \tag{2.29}$$

obtinguda a l'Annex A.3.

D'una banda, de l'Eq. (2.29) veiem que

$$\dot{\boldsymbol{q}} = \mathbf{J}_i^{-1}\cdot\dot{\boldsymbol{\theta}} \tag{2.30}$$

D'altra banda, si fem la derivada temporal de l'Eq. (2.29) obtenim l'expressió

$$\ddot{\boldsymbol{\theta}} = \dot{\mathbf{J}}_i\cdot\dot{\boldsymbol{q}} + \mathbf{J}_i\cdot\ddot{\boldsymbol{q}}, \tag{2.31}$$

la qual condueix a

$$\ddot{\boldsymbol{q}} = \mathbf{J}_i^{-1}\left(\ddot{\boldsymbol{\theta}} - \dot{\mathbf{J}}_i\cdot\dot{\boldsymbol{q}}\right), \tag{2.32}$$

i, utilitzant l'Eq. (2.30), a

$$\ddot{\boldsymbol{q}} = \mathbf{J}_i^{-1}\left(\ddot{\boldsymbol{\theta}} - \dot{\mathbf{J}}_i\cdot\mathbf{J}_i^{-1}\cdot\dot{\boldsymbol{\theta}}\right). \tag{2.33}$$

Si ara substituïm les expressions (2.30) i (2.33) a l'Eq. (2.27) i tenim en compte que $\boldsymbol{u} = \boldsymbol{u}_\tau$,

obtindrem

$$\underbrace{\mathbf{H}\mathbf{J}_i^{-1}}_{\mathbf{H}_\theta}\cdot\ddot{\boldsymbol{\theta}}+\underbrace{(\mathbf{C}-\mathbf{H}\mathbf{J}_i^{-1}\dot{\mathbf{J}}_i)\mathbf{J}_i^{-1}}_{\mathbf{C}_\theta}\cdot\dot{\boldsymbol{\theta}}+\mathbf{G}=\mathbf{E}\cdot\boldsymbol{u}_\tau, \tag{2.34}$$

que podem escriure de manera compacta com

$$\mathbf{H}_\theta\cdot\ddot{\boldsymbol{\theta}}+\mathbf{C}_\theta\cdot\dot{\boldsymbol{\theta}}+\mathbf{G}=\mathbf{E}\cdot\boldsymbol{u}_\tau \tag{2.35}$$

on $\mathbf{H}_\theta$ i $\mathbf{C}_\theta$ són les matrius definides a l'Eq. (2.34).

L'Eq. (2.35) proporciona l'equació del moviment en coordenades motor. Cal tenir en compte que en aquesta equació $\mathbf{H}_\theta$, $\mathbf{C}_\theta$, $\mathbf{G}$ i $\mathbf{E}$ són funcions de $\boldsymbol{q}$ i $\dot{\boldsymbol{q}}$, però ara l'equació ens descriu el comportament dinàmic de l'Hexapole en les coordenades $\boldsymbol{\theta}$ sota les accions $\boldsymbol{u}_\tau=(\tau_1,\ldots,\tau_6)$. Això ens permetrà dissenyar una llei de control en anell tancat que rebi $\boldsymbol{\theta}$ i $\dot{\boldsymbol{\theta}}$ com entrades, i retorni consignes de parell motor com a sortida. La introducció de les variables $\boldsymbol{\theta}$, a més, ens facilitarà la inclusió de les equacions diferencials dels motors en el model (Capítol 3).

2.6 Reducció a forma explícita de primer ordre

Finalment si aïllem les acceleracions generalitzades de l'Eq. (2.35) obtenim

$$\ddot{\boldsymbol{\theta}}=\mathbf{H}_\theta^{-1}\left(\mathbf{E}\cdot\boldsymbol{u}_\tau-\mathbf{C}_\theta\cdot\dot{\boldsymbol{\theta}}-\mathbf{G}\right)$$

que, juntament amb l'equació trivial

$$\dot{\boldsymbol{\theta}}=\dot{\boldsymbol{\theta}},$$

permet construir el sistema

$$\underbrace{\begin{bmatrix}\dot{\boldsymbol{\theta}}\\ \ddot{\boldsymbol{\theta}}\end{bmatrix}}_{\dot{\boldsymbol{x}}_\theta}=\underbrace{\begin{bmatrix}\dot{\boldsymbol{\theta}}\\ \mathbf{H}_\theta^{-1}\left(\mathbf{E}\cdot\boldsymbol{u}_\tau-\mathbf{C}_\theta\cdot\dot{\boldsymbol{\theta}}-\mathbf{G}\right)\end{bmatrix}}_{f(\boldsymbol{x}_\theta,\boldsymbol{u}_\tau)} \tag{2.36}$$

i arribar a la forma explícita de primer ordre que necessitàvem:

$$\dot{\boldsymbol{x}}_\theta=f(\boldsymbol{x}_\theta,\boldsymbol{u}_\tau). \tag{2.37}$$

3

Inclusió dels motors en el model

El model mecànic de l'Eq. (2.36) podria ser suficient per dissenyar una llei de control que estabilitzés l'Hexapole, però això requeriria disposar de motors capaços de rebre consignes de parell motor, i de seguir-les de forma prou ràpida i acurada. Malauradament, els motors de l'Hexapole no tenen aquesta capacitat. Són motors de baixes prestacions que no admeten consignes de parell, i la seva relació de reducció és molt alta, fet que augmenta els efectes del fregament sec i viscós dels motors, fent-los lents a l'hora de reaccionar. Per tal de tenir en compte aquestes limitacions, en aquest capítol ampliarem l'Eq. (2.36) amb les equacions que descriuen el comportament dinàmic d'aquests motors. El resultat serà un model electromecànic que descriurà el comportament global del sistema de manera més realista. Aquest model utilitzarà els voltatges d'excitació com a coordenades d'acció, i per tant permetrà dissenyar lleis de control que consignin aquests voltatges directament. Això facilitarà molt la implementació d'aquestes lleis sobre el sistema Hexapole real, ja que la interfície de programació dels seus motors permet consignar voltatges d'excitació de manera senzilla.

3.1 Equacions d'un motor de corrent continu

L'Hexapole incorpora sis servomotors Dynamixel MX-28 que inclouen un motor Maxon de corrent continu model 214897, un reductor, i un microcontrolador que implementa llaços de control PID de velocitat o posició (segons quin sigui el mode d'operació triat). En aquest treball, tanmateix, prescindirem d'aquests llaços de control, i explotarem el fet que es coneix un model força acurat dels motors [10], i que aquests motors es poden controlar consignant els seus voltatges d'excitació directament. Per modelitzar els motors utilitzarem els paràmetres dinàmics ja identificats a [10].

La dinàmica dels motors de l'Hexapole ve donada pels fenòmens elèctrics i mecànics esquematitzats a les Figs. 3.1 i 3.2. L'eix del motor està connectat a l'eix de sortida a través d'un

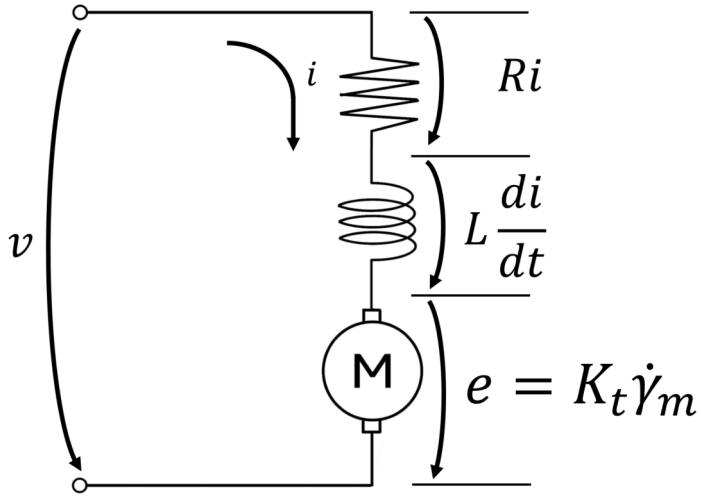


Figura 3.1: Esquema que modelitza la dinàmica elèctrica del servomotor.

reductor de relació $N = \frac{\gamma_m}{\gamma}$, on γ_m és l'angle girat per l'eix del motor, i γ és l'angle girat per l'eix de sortida.

Per arribar al model electromecànic de l'Hexapole expressarem el parell de càrrega τ_l que els cables fan sobre el motor en funció del voltatge d'excitació v del motor, l'angle γ, la velocitat $\dot{\gamma}$, i l'acceleració $\ddot{\gamma}$. Aquesta expressió, convenientment substituïda a l'Eq. (2.36), proporcionarà el model electromecànic desitjat.

Per obtenir aquest model analitzarem primer la part elèctrica dels motors (Fig. 3.1) utilitzant la següent nomenclatura. La caiguda de tensió en els pols de l'armadura la denotarem per v (perquè coincideix amb el voltatge d'excitació dels motors). La intensitat elèctrica que circula per l'induït la denotarem per i, la resistència del circuit per R, la inductància per L, la força contraelectromotriu en els pols del motor (f.c.e.m.) per e, i la constant de parell associada a aquesta força per K_t. Pel que fa a les velocitats dels motors, $\dot{\gamma}_m$ serà la velocitat angular a l'eix del motor (abans de la reducció), i $\dot{\gamma}_l$ serà la velocitat angular a l'eix de sortida del motor (després de la reducció). Finalment la reducció del motor vindrà donada per $N = \frac{R_l}{R_m}$ on R_l i R_m són els radis dels engranatges que es mostren a la Fig. 3.2. Amb aquesta nomenclatura, la caiguda de tensió v es pot expressar així.

$$v = Ri + L\frac{\mathrm{d}i}{\mathrm{d}t} + \underbrace{K_t \cdot \dot{\gamma}_m}_{f.c.e.m.}$$

Com en la majoria de motors de corrent continu, però, la inductància L es pot considerar

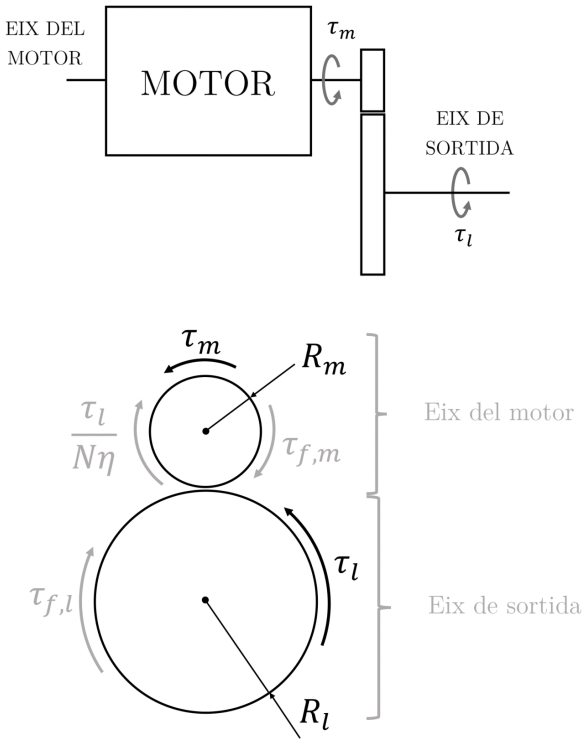


Figura 3.2: Esquema de la part mecànica dels servomotors.

negligible, amb la qual cosa l'expressió anterior queda reduïda a

$$v = R\, i + K_t\, \dot{\gamma}_m.$$

Si ara aïllem la intensitat i d'aquesta equació, ens queda

$$i = \frac{1}{R}(v - K_t \dot{\gamma}_m), \tag{3.1}$$

i substituint l'expressió

$$\dot{\gamma}_m = N\dot{\gamma} \tag{3.2}$$

a l'Eq. (3.1) obtenim l'expressió

$$i = \frac{1}{R}\left(v - K_t N \dot{\gamma}\right), \tag{3.3}$$

que proporciona el corrent que circula pel circuit en funció de v i $\dot{\gamma}$.

Anem ara a descriure la part mecànica. Farem servir la següent nomenclatura. La inèrcia a l'eix del motor es denotarà per J_m, el parell efectuat pel motor sobre el seu eix per τ_m, el parell de càrrega que els cables fan sobre l'eix de sortida per τ_l i el parell de fricció a l'eix del motor

per $\tau_{f,m}$. El rendiment mecànic del reductor el denotarem per η. Així, si apliquem el teorema del moment cinètic a l'eix del motor, podem escriure

$$J_m\,\ddot{\gamma}_m = \tau_m - \frac{\tau_l}{N\eta} - \tau_{f,m}, \tag{3.4}$$

on $\frac{\tau_l}{N\eta}$ és el parell de càrrega reduït a l'eix del motor i afectat pel coeficient de rendiment η. Aïllant τ_l de l'Eq. (3.4) i utilitzant el fet que $\tau_m = K_t \cdot i$, i que $\ddot{\gamma}_m = N\ddot{\gamma}$, obtenim

$$\tau_l = \eta N K_t i - \eta N \tau_{f,m} - \eta N^2 J_m \ddot{\gamma}. \tag{3.5}$$

Finalment, substituint l'Eq. (3.3) a la (3.5) obtenim τ_l en funció del voltatge d'excitació v, i de $\dot{\gamma}$ i $\ddot{\gamma}$.

$$\tau_l = \frac{\eta N K_t}{R}\, v - \frac{\eta N^2 K_t^2}{R}\,\dot{\gamma} - \eta N \tau_{f,m} - \eta N^2 J_m \ddot{\gamma} \tag{3.6}$$

Fixem-nos que en aquesta expressió $\eta\frac{\eta N K_t}{R}\ v - \eta\frac{\eta N^2 K_t^2}{R}\ \dot{\gamma}$ és el parell motor, $-\eta N \tau_{f,m}$ és el parell de fregament, i $-\eta N^2 J_m \ddot{\gamma}$ és el parell d'inèrcia, essent els tres parells reduïts a l'eix de sortida. Cal fer notar aquí que $N = 193$ en els servomotors Dynamixel MX-28, amb la qual cosa la reducció és molt elevada. Això fa que el parell d'inèrcia $-\eta N^2 J_m \ddot{\gamma}$ pugui tenir un efecte no menyspreable. Tot seguit veurem que es produirà un efecte semblant amb el parell de fricció $-\eta N \tau_{f,m}$, ja que també quedarà afectat pel factor N^2.

3.2 Incorporació del fregament sec i viscós

A l'Eq. (3.6) el terme $\tau_{f,m}$ modelitza el parell de fregament sec i viscós del motor. Hi ha fregament sec i viscós als dos eixos del reductor (el del motor, i el de sortida del reductor) però, només el fregament al primer eix sol tenir efectes apreciables, perquè és el que queda amplificat per la relació de reducció. Com que aquesta relació és molt alta en el cas de l'Hexapole (al Capítol 6 veurem que $N = 193$) en aquest treball considerarem que el fregament a l'eix de sortida és negligible.

El fregament sec i viscós a l'eix del motor sol ser una funció no lineal de $\dot{\gamma}_m$, amb l'aspecte que s'indica a la Fig. 3.3 **(a)**. Per simplificar, però, aproximarem aquesta funció amb la gràfica de la Fig. 3.3 **(b)**, i per tant suposarem que

$$\tau_{f,m} = \underbrace{b_m \dot{\gamma}_m}_{Frec\ viscós} + \underbrace{sgn\,(\dot{\gamma}_m)\,\tau_c}_{Frec\ sec}. \tag{3.7}$$

En aquesta equació, el terme $b_m\dot{\gamma}_m$ modelitza el frec viscós, mentre que el terme $sgn(\dot{\gamma}_m)\tau_c$

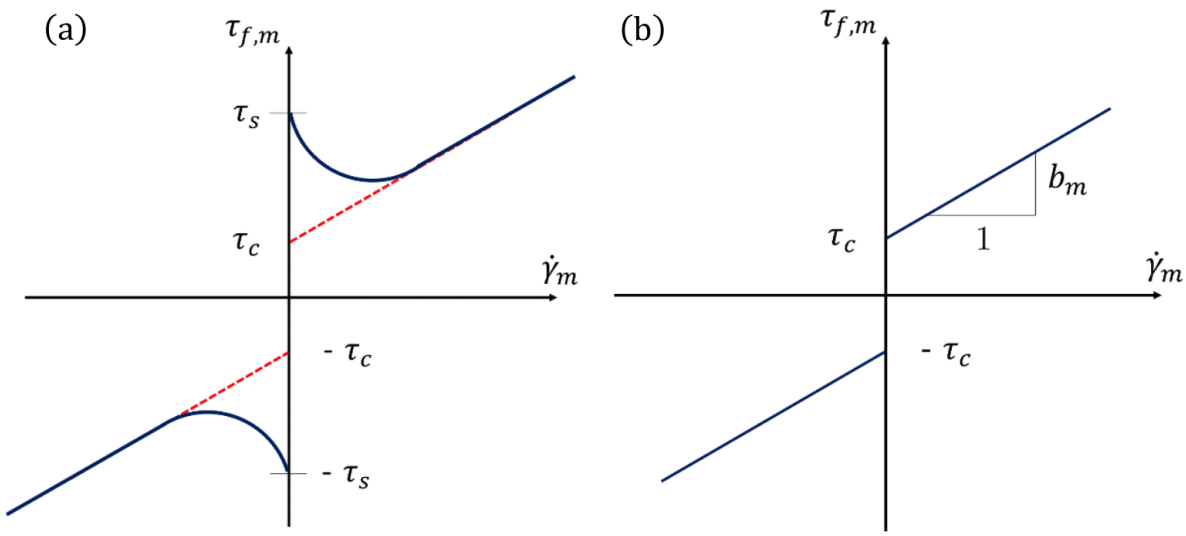


Figura 3.3: **(a)** Model acurat del parell de fricció $\tau_{f,m}$ en funció de $\dot{\theta}_m$. **(b)** Aproximació de la funció parell $\tau_{f,m} = \tau_{f,m}(\dot{\gamma}_m)$ de fricció que s'utilitza en aquest projecte.

modelitza el frec sec de Coulomb. La funció $sgn(\dot{\gamma}_m)$ val $+1$ si $\dot{\gamma}_m > 0$ i -1 si $\dot{\gamma}_m < 0$. Els coeficients b_m i τ_c són el coeficient de frec viscós i el parell de Coulomb, respectivament, i estan representats a la Fig. 3.3 **(b)**. Si ara multipliquem l'Eq. (3.7) per ηN i utilitzem el fet que $\dot{\gamma}_m = N\,\dot{\gamma}$, obtenim

$$\eta N \tau_{f,m} = \eta N^2 b_m \dot{\gamma} + sgn(\dot{\gamma}) \eta N \tau_c, \tag{3.8}$$

que ens dóna l'expressió detallada del terme $\eta N \tau_{f,m}$ de l'Eq. (3.6). Fixem-nos que, com ja avançàvem, quan el parell de fregament *es veu* des de l'eix de sortida, queda afectat pel coeficient N^2, fet que amplifica els seus efectes.

3.3 Model electromecànic global

Substituint l'Eq. (3.8) a l'Eq. (3.6) arribem a l'expressió

$$\tau_l = \frac{\eta N K_t}{R} v - \frac{\eta N^2 K_t^2}{R} \dot{\gamma} - \eta N^2 b_m \dot{\gamma} - sgn(\dot{\gamma}) \eta N \tau_c - \eta N^2 J_m \ddot{\gamma},$$

i reorganitzant i ordenant els seus termes obtenim

$$\tau_l = \underbrace{\frac{\eta N K_t}{R}}_{a_\tau} v \underbrace{- \left(\frac{\eta N^2 K_t^2}{R} + \eta N^2 b_m \right)}_{b_\tau} \dot{\gamma} \underbrace{- \eta N^2 J_m}_{c_\tau} \ddot{\gamma} \underbrace{- sgn\,(\dot{\gamma})\, \eta N \tau_c}_{d_\tau},$$

que de manera compacta podem escriure com

$$\tau_l = a_\tau v + b_\tau \dot{\gamma} + c_\tau \ddot{\gamma} + d_\tau. \tag{3.9}$$

L'Eq. (3.9) es pot escriure ara per cadascun dels motors, obtenint les equacions

$$\tau_{l,i} = a_\tau v_i + b_\tau \dot{\gamma}_i + c_\tau \ddot{\gamma}_i + d_{\tau,i}$$

per $i = 1, \ldots, 6$, que podem escriure vectorialment com

$$\boldsymbol{\tau}_l = a_\tau \boldsymbol{u}_v + b_\tau \dot{\boldsymbol{\gamma}} + c_\tau \ddot{\boldsymbol{\gamma}} + \boldsymbol{d}_\tau. \tag{3.10}$$

on, recordem, $\boldsymbol{u}_v = (v_1, \ldots, v_6)$.

Recordem ara que el model mecànic en coordenades motor ve donat per l'Eq. (2.35),

$$\mathbf{H}_\theta \cdot \ddot{\boldsymbol{\theta}} + \mathbf{C}_\theta \cdot \dot{\boldsymbol{\theta}} + \mathbf{G} = \mathbf{E} \cdot \boldsymbol{u}_\tau. \tag{3.11}$$

Com que $\boldsymbol{u}_\tau = \boldsymbol{\tau}_l$, si ara substituim l'Eq. (3.10) a la (3.11) obtenim

$$\mathbf{H}_\theta \cdot \ddot{\boldsymbol{\theta}} + \mathbf{C}_\theta \cdot \dot{\boldsymbol{\theta}} + \mathbf{G} = a_\tau \mathbf{E}\boldsymbol{u}_v + b_\tau \mathbf{E}\dot{\boldsymbol{\gamma}} + c_\tau \mathbf{E}\ddot{\boldsymbol{\gamma}} + \mathbf{E}\boldsymbol{d}_\tau, \tag{3.12}$$

i si definim les matrius $\mathbf{C}_m$ i $\mathbf{H}_m$ de la forma següent

$$b_\tau \mathbf{E}\dot{\boldsymbol{\theta}} = \underbrace{\begin{bmatrix} b_\tau \mathbf{E} & \mathbf{0} \end{bmatrix}}_{\mathbf{C}_m} \cdot \underbrace{\begin{bmatrix} \dot{\boldsymbol{\gamma}} \\ \dot{\boldsymbol{\beta}} \end{bmatrix}}_{\dot{\boldsymbol{\theta}}},$$

$$c_\tau \mathbf{E}\ddot{\boldsymbol{\theta}} = \underbrace{\begin{bmatrix} c_\tau \mathbf{E} & \mathbf{0} \end{bmatrix}}_{\mathbf{H}_m} \cdot \underbrace{\begin{bmatrix} \ddot{\boldsymbol{\gamma}} \\ \ddot{\boldsymbol{\beta}} \end{bmatrix}}_{\ddot{\boldsymbol{\theta}}},$$

podem reescriure l'Eq. (3.2) així

$$\underbrace{(\mathbf{H}_\theta - \mathbf{H}_m)}_{\mathbf{H}_{\theta,m}} \ddot{\boldsymbol{\theta}} + \underbrace{(\mathbf{C}_\theta - \mathbf{C}_m)}_{\mathbf{C}_{\theta,m}} \dot{\boldsymbol{\theta}} + \mathbf{G} = \mathbf{E}(a_\tau \boldsymbol{u}_v + \boldsymbol{d}_\tau),$$

i per tant arribem a l'expressió

$$\mathbf{H}_{\theta,m}\ddot{\boldsymbol{\theta}} + \mathbf{C}_{\theta,m}\dot{\boldsymbol{\theta}} + \mathbf{G} = \mathbf{E}(a_\tau \boldsymbol{u}_v + \boldsymbol{d}_\tau), \tag{3.13}$$

que constitueix el model electromecànic global que buscàvem. Cal tenir en compte, una vegada més que, tot i ser un model en coordenades $\boldsymbol{\theta}$, les magnituds $\mathbf{H}_{\theta,m}$, $\mathbf{G}$ i $\mathbf{E}$ són funcions de $\boldsymbol{q}$ i la matriu $\mathbf{C}_{\theta,m}$ és funció de $\boldsymbol{q}$ i $\dot{\boldsymbol{q}}$.

3.4 Cancelació del fregament sec via realimentació

Una estratègia típica per tractar amb el terme de fricció de Coulomb $\boldsymbol{d}_\tau$, és cancel·lar el seu efecte mitjançant la linealització per realimentació. Aquest mètode consisteix simplement a utilitzar la següent llei de control en el sistema real

$$\boldsymbol{u}_v = \boldsymbol{u}_{v,o} - \mathbf{K}\left(\boldsymbol{x}_\theta - \boldsymbol{x}_{\theta,o}\right) - \boldsymbol{d}_\tau, \tag{3.14}$$

on el terme $\boldsymbol{u}_{v,o} - \mathbf{K}(\boldsymbol{x}_\theta - \boldsymbol{x}_{\theta,o})$ és la llei de control que obtindríem si no hi hagués fregament sec de Coulomb (veure el Capítol 5) i $\boldsymbol{d}_\tau$ és el terme de Coulomb de l'Eq. (3.10). Per entendre aquesta estratègia, fixem-nos que si substituïm l'Eq. (3.14) a l'Eq. (3.13) el terme $-\boldsymbol{d}_\tau$ cancel·la $\boldsymbol{d}_\tau$ i l'efecte net és el d'un sistema sense fricció de Coulomb. Així doncs, en simulació això és equivalent a assumir que el sistema no té frec sec de Coulomb.

En aquest treball considerarem que en el sistema Hexapole real s'utilitzarà aquesta estratègia de control, i per tant suposarem que $\boldsymbol{d}_\tau = 0$ tant en la linealització com en la simulació del sistema, i també en el disseny de la llei de control. Així doncs, l'Eq (3.13) sense el terme $\boldsymbol{d}_\tau$ quedarà de la següent manera

$$\mathbf{H}_{\theta,m}\ddot{\boldsymbol{\theta}} + \mathbf{C}_{\theta,m}\dot{\boldsymbol{\theta}} + \mathbf{G} = a_\tau \mathbf{E}\boldsymbol{u}_v. \tag{3.15}$$

Per convertir aquesta equació a forma explícita de primer ordre només ens resta aïllar les acceleracions generalitzades de l'Eq (3.15)

$$\ddot{\boldsymbol{\theta}} = \mathbf{H}_{\theta,m}^{-1}\left[a_\tau \mathbf{E}\boldsymbol{u}_v - \mathbf{C}_{\theta,m}\dot{\boldsymbol{\theta}} - \mathbf{G}\right]$$

i adjuntar l'equació trivial

$$\dot{\boldsymbol{\theta}} = \dot{\boldsymbol{\theta}},$$

amb la qual cosa arribem a

$$\underbrace{\begin{bmatrix}\dot{\boldsymbol{\theta}}\\ \ddot{\boldsymbol{\theta}}\end{bmatrix}}_{\dot{\boldsymbol{x}}_\theta} = \underbrace{\begin{bmatrix}\dot{\boldsymbol{\theta}}\\ \mathbf{H}_{\theta,m}^{-1}\left[a_\tau \mathbf{E}\boldsymbol{u}_v - \mathbf{C}_{\theta,m}\dot{\boldsymbol{\theta}} - \mathbf{G}\right]\end{bmatrix}}_{f(\boldsymbol{x}_\theta,\, \boldsymbol{u}_v)}, \tag{3.16}$$

o, de manera compacta, a

$$\dot{\boldsymbol{x}}_\theta = f\left(\boldsymbol{x}_\theta,\ \boldsymbol{u}_v\right). \tag{3.17}$$

3.5 Identificació del fregament sec

A la pràctica, per identificar experimentalment el valor del parell de Coulomb τ_c podem excitar el motor sense càrrega ($\tau_l = 0$) i anar incrementant suaument els valors de voltatge v fins que l'eix de sortida comenci a rotar. La tensió crítica $v = v_c$ a la qual comenci aquesta rotació, ens donarà el parell τ_c (suposant, per simplificar, que els valors τ_s i τ_c de la Fig. 3.3 són semblant). Certament, en aquest punt crític tenim $\ddot{\gamma}_m = 0$, i per tant hi haurà els següents termes nuls a l'Eq. (3.4),

$$\underbrace{J_m\ddot{\gamma}_m}_{0} = \tau_m \underbrace{-\frac{\tau_l}{N\eta}}_{0} - \tau_{f,m},$$

de manera que en aquesta situació serà

$$\tau_{f,m} = \tau_m.$$

L'Eq. (3.3), a més, es reduirà a

$$i_c = \frac{v_c}{R}$$

on i_c és el valor del corrent crític corresponent a v_c. Com que $\tau_m = K_t \cdot i_c$, veiem que

$$\tau_{f,m} = \tau_m = K_t \cdot i_c = K_t \cdot \frac{v_c}{R},$$

i com que en aquest instant crític tota la fricció serà seca, el valor buscat de τ_c serà

$$\tau_c = K_t \cdot \frac{v_c}{R}.$$

4

Linealització del model

En aquest capítol detallarem com es poden linealitzar els models mecànic i electromecànic al voltant de l'estat d'equilibri desitjat (amb la plataforma parada en una configuració especificada, i amb el pèndol invertit en posició vertical). Aquesta linealització és necessària per després poder aplicar la teoria del control òptim a cadascun d'aquests models (Capítol 5).

4.1 Forma abstracta del model

Tant si considerem el model mecànic de l'Eq. (2.36), com l'electromecànic de l'Eq. (3.16), el procés de linealització és el mateix. Això és així perquè, tot i ser diferents, els dos models tenen la mateixa estructura matemàtica, que es pot resumir així

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}) = \begin{bmatrix} \dot{\boldsymbol{\theta}} \\ \mathbf{H}^{-1}\left(\mathbf{E}\cdot\boldsymbol{u} - \mathbf{C}\cdot\dot{\boldsymbol{\theta}} - \mathbf{G}\right) \end{bmatrix}, \tag{4.1}$$

on $\mathbf{H}$, $\mathbf{E}$, $\mathbf{C}$, i $\mathbf{G}$ fan referència a $\mathbf{H}_\theta$, $\mathbf{E}$, $\mathbf{C}_\theta$, i $\mathbf{G}$, en el cas de l'Eq. (2.36), o a $\mathbf{H}_{\theta,m}$, $\mathbf{E}$, $\mathbf{C}_{\theta,m}$, i $\mathbf{G}$, en el cas de l'Eq. (3.16), respectivament. Anàlogament, el vector $\boldsymbol{u}$ d'accions fa referència a $\boldsymbol{u}_\tau$, en el cas de l'Eq. (2.36), o a $\boldsymbol{u}_v$, en el cas de l'Eq. (3.16). En tot aquest capítol, per tant, suposarem que el sistema a linealitzar adopta l'estructura de l'Eq. (4.1), on $\boldsymbol{x} = (\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})$, i ens estalviarem els detalls de particularitzar la linealització obtinguda per cadascun dels models esmentats.

Abans d'abordar la linealització és important remarcar que, tant en l'Eq. (2.36) com en l'Eq. (3.16), els termes $\mathbf{H}$, $\mathbf{E}$, $\mathbf{C}$, i $\mathbf{G}$ venen donats en funció de les coordenades $(\boldsymbol{q}, \dot{\boldsymbol{q}})$, i no pas en les $(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})$. Aquest fet no tindrà excessiu impacte en el procés de linealització, però s'haurà de tenir en compte a l'hora de calcular certes derivades respecte de les coordenades $\boldsymbol{\theta}$.

4.2 El procés de linealització

Sigui $\boldsymbol{x}_o$ el vector que caracteritza l'estat d'equilibri desitjat, i $\boldsymbol{u}_0$ el vector d'accions necessari per mantenir aquest estat d'equilibri. En el cas de l'Hexapole, $\boldsymbol{x}_o$ codifica un estat en el qual la plataforma està en posició horitzontal enmig de l'espai de treball assolible, el pèndol es troba en posició vertical invertida, i ambdós sòlids tenen velocitat nul·la. En canvi, el vector $\boldsymbol{u}_o$ codifica les accions necessàries per compensar el pes de la plataforma i el pèndol en aquest estat, que seran valors de parell motor en el model de l'Eq. (2.36), o de voltatge d'excitació en el de l'Eq. (3.16). Els valors concrets assumits per $\boldsymbol{x}_o$ i $\boldsymbol{u}_o$ es donen a la Secció 6.3.1.

Per linealitzar el sistema de l'Eq. (4.1) al voltant de $(\boldsymbol{x}_o, \boldsymbol{u}_o)$ calcularem el desenvolupament en serie de Taylor de $f(\boldsymbol{x}, \boldsymbol{u})$ en el punt $(\boldsymbol{x}_o, \boldsymbol{u}_o)$ fins als termes de primer ordre:

$$\dot{\boldsymbol{x}} \approx \underbrace{f\left(\boldsymbol{x}_o, \boldsymbol{u}_o\right)}_{0} + \underbrace{\left.\frac{\partial f}{\partial \boldsymbol{x}}\right|_{\substack{\boldsymbol{x}=\boldsymbol{x}_o \\ \boldsymbol{u}=\boldsymbol{u}_o}}}_{\mathbf{A}} (\boldsymbol{x} - \boldsymbol{x}_o) + \underbrace{\left.\frac{\partial f}{\partial \boldsymbol{u}}\right|_{\substack{\boldsymbol{x}=\boldsymbol{x}_o \\ \boldsymbol{u}=\boldsymbol{u}_o}}}_{\mathbf{B}} (\boldsymbol{u} - \boldsymbol{u}_o) \tag{4.2}$$

Fixem-nos que en aquesta equació tenim $f(\boldsymbol{x}_o, \boldsymbol{u}_o) = \mathbf{0}$, perquè $\boldsymbol{x}_o$ és un estat d'equilibri quan s'aplica l'acció $\boldsymbol{u}_o$. Els Jacobians $\frac{\partial f}{\partial \boldsymbol{x}}$ i $\frac{\partial f}{\partial \boldsymbol{u}}$ avaluats a $(\boldsymbol{x}_o, \boldsymbol{u}_o)$ ens donen les matrius $\mathbf{A}$ i $\mathbf{B}$ del sistema dinàmic linealitzat. Aquest sistema, per tant, tindrà l'expressió

$$\dot{\boldsymbol{x}} = \mathbf{A} \cdot (\boldsymbol{x} - \boldsymbol{x}_o) + \mathbf{B} \cdot (\boldsymbol{u} - \boldsymbol{u}_o) \tag{4.3}$$

Vegem tot seguit com podem calcular $\mathbf{A} = \frac{\partial f}{\partial \boldsymbol{x}}$ i $\mathbf{B} = \frac{\partial f}{\partial \boldsymbol{u}}$ en el punt d'equilibri $(\boldsymbol{x}_o, \boldsymbol{u}_o)$, a partir dels termes de l'Eq. (4.1).

4.2.1 Derivades respecte de l'estat

Pel que fa a la matriu $\mathbf{A}$ tenim

$$\mathbf{A} = \frac{\partial f}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial \dot{\boldsymbol{\theta}}}{\partial \boldsymbol{\theta}} & \frac{\partial \dot{\boldsymbol{\theta}}}{\partial \dot{\boldsymbol{\theta}}} \\ \frac{\partial \ddot{\boldsymbol{\theta}}}{\partial \boldsymbol{\theta}} & \frac{\partial \ddot{\boldsymbol{\theta}}}{\partial \dot{\boldsymbol{\theta}}} \end{bmatrix}, \tag{4.4}$$

on la primera fila és trivial ja que

$$\frac{\partial \dot{\boldsymbol{\theta}}}{\partial \boldsymbol{\theta}} = \mathbf{0} \tag{4.5}$$

i

$$\frac{\partial \dot{\boldsymbol{\theta}}}{\partial \dot{\boldsymbol{\theta}}} = \mathbf{I}. \tag{4.6}$$

La segona fila es més laboriosa. La primera derivada parcial és

$$\frac{\partial \ddot{\boldsymbol{\theta}}}{\partial \boldsymbol{\theta}} = \frac{\partial \mathbf{H}^{-1}}{\partial \boldsymbol{\theta}} \cdot \underbrace{(\mathbf{E}\boldsymbol{u} - \mathbf{C}\dot{\boldsymbol{\theta}} - \mathbf{G})}_{\mathbf{0}} + \mathbf{H}^{-1} \cdot \frac{\partial}{\partial \boldsymbol{\theta}} \left(\mathbf{E}\boldsymbol{u} - \mathbf{C}\dot{\boldsymbol{\theta}} - \mathbf{G}\right). \tag{4.7}$$

Fixem-nos que en el segon membre de l'Eq (4.7), el terme $\mathbf{E}\boldsymbol{u} - \mathbf{C}\dot{\boldsymbol{\theta}} - \mathbf{G}$ és igual a $\mathbf{H}\ddot{\theta}$. Tenint en compte que l'expressió s'ha d'avaluar en el punt d'equilibri, i que $\ddot{\boldsymbol{\theta}}$ és nul·la en aquest punt, podem afirmar que $\mathbf{H}\ddot{\theta} = \mathbf{E}\boldsymbol{u} - \mathbf{C}\dot{\boldsymbol{\theta}} - \mathbf{G} = \mathbf{0}$ a $(\boldsymbol{x}_o, \boldsymbol{u}_o)$. Pel que fa al terme $\frac{\partial}{\partial \boldsymbol{\theta}} \left(\mathbf{E}\boldsymbol{u} - \mathbf{C}\dot{\boldsymbol{\theta}} - \mathbf{G}\right)$ fixem-nos que

$$\frac{\partial (\mathbf{C}\dot{\boldsymbol{\theta}})}{\partial \boldsymbol{\theta}} = \frac{\partial \mathbf{C}}{\partial \boldsymbol{\theta}} \cdot \dot{\boldsymbol{\theta}},$$

però com que $\dot{\boldsymbol{\theta}} = 0$ al punt d'equilibri, en aquest punt tindrem

$$\frac{\partial \mathbf{C}}{\partial \boldsymbol{\theta}} \cdot \dot{\boldsymbol{\theta}} = \mathbf{0}.$$

Si tenim en compte això, i el fet que les matrius $\mathbf{E}$ i $\mathbf{G}$ estan expressades com a funcions de $\boldsymbol{q}$, que alhora depèn de $\boldsymbol{\theta}$, la derivada $\frac{\partial \ddot{\boldsymbol{\theta}}}{\partial \boldsymbol{\theta}}$ serà

$$\frac{\partial \ddot{\boldsymbol{\theta}}}{\partial \boldsymbol{\theta}} = \mathbf{H}^{-1} \cdot \frac{\partial}{\partial \boldsymbol{\theta}} (\mathbf{E}\boldsymbol{u} - \mathbf{G}) = \mathbf{H}^{-1} \cdot \frac{\partial}{\partial \boldsymbol{q}} (\mathbf{E}\boldsymbol{u} - \mathbf{G}) \cdot \underbrace{\frac{\partial \boldsymbol{q}}{\partial \boldsymbol{\theta}}}_{\mathbf{J}_i^{-1}} = \mathbf{H}^{-1} \cdot \frac{\partial}{\partial \boldsymbol{q}} (\mathbf{E}\boldsymbol{u} - \mathbf{G}) \cdot \mathbf{J}_i^{-1} \tag{4.8}$$

on $\mathbf{J}_i$ és el Jacobià obtingut a l'Annex A.3, i $\frac{\partial}{\partial \boldsymbol{q}} (\mathbf{E}\boldsymbol{u} - \mathbf{G})$ es podrà calcular fàcilment amb l'ajut de MAPLE.

Per altra banda, la derivada $\frac{\partial \ddot{\boldsymbol{\theta}}}{\partial \dot{\boldsymbol{\theta}}}$ es pot desenvolupar així

$$\frac{\partial \ddot{\boldsymbol{\theta}}}{\partial \dot{\boldsymbol{\theta}}} = \frac{\partial \mathbf{H}^{-1}}{\partial \dot{\boldsymbol{\theta}}} \cdot \left(\mathbf{E}\boldsymbol{u} - \mathbf{C}\dot{\boldsymbol{\theta}} - \mathbf{G}\right) + \mathbf{H}^{-1} \cdot \frac{\partial}{\partial \dot{\boldsymbol{\theta}}} \left(\mathbf{E}\boldsymbol{u} - \mathbf{C}\dot{\boldsymbol{\theta}} - \mathbf{G}\right). \tag{4.9}$$

Si tenim en compte que en el punt d'equilibri $\dot{\boldsymbol{\theta}} = \mathbf{0}$ i $\mathbf{E}\boldsymbol{u} - \mathbf{C}\dot{\boldsymbol{\theta}} - \mathbf{G} = \mathbf{0}$, que $\mathbf{E}$ i $\mathbf{G}$ no depenen de $\dot{\boldsymbol{\theta}}$, podem reescriure l'Eq. (4.9) de la següent forma

$$\frac{\partial \ddot{\boldsymbol{\theta}}}{\partial \dot{\boldsymbol{\theta}}} = -\mathbf{H}^{-1} \left[\frac{\partial \mathbf{C}}{\partial \dot{\boldsymbol{\theta}}} \dot{\boldsymbol{\theta}} + \mathbf{C}\right] = -\mathbf{H}^{-1}\mathbf{C}. \tag{4.10}$$

Per tant substituint les Eqs. (4.5), (4.6), (4.8) i (4.10) a l'Eq. (4.4) obtenim

$$\mathbf{A} = \left.\frac{\partial f}{\partial \boldsymbol{x}}\right|_{\substack{\boldsymbol{x}=\boldsymbol{x}_o \\ \boldsymbol{u}=\boldsymbol{u}_o}} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{H}^{-1} \left(\frac{\partial}{\partial \boldsymbol{\theta}} (\mathbf{E}\boldsymbol{u} - \mathbf{G})\right) & -\mathbf{H}^{-1}\mathbf{C} \end{bmatrix}. \tag{4.11}$$

4.2.2 Derivades respecte de l'acció

Pel que fa a la matriu $\mathbf{B}$, tenim

$$\mathbf{B} = \frac{\partial f}{\partial \boldsymbol{u}} = \begin{bmatrix} \frac{\partial \dot{\boldsymbol{\theta}}}{\partial \boldsymbol{u}} \\ \frac{\partial \ddot{\boldsymbol{\theta}}}{\partial \boldsymbol{u}} \end{bmatrix}. \tag{4.12}$$

Detallant ara cada un dels seus termes, veiem que

$$\frac{\partial \dot{\boldsymbol{\theta}}}{\partial \boldsymbol{u}} = \mathbf{0}, \tag{4.13}$$

i que

$$\frac{\partial \ddot{\boldsymbol{\theta}}}{\partial \boldsymbol{u}} = \frac{\partial \mathbf{H}^{-1}}{\partial \boldsymbol{u}} \cdot \underbrace{\left(\mathbf{E}\boldsymbol{u} - \mathbf{C}\dot{\boldsymbol{\theta}} - \mathbf{G}\right)}_{0} + \mathbf{H}^{-1} \cdot \frac{\partial}{\partial \boldsymbol{u}} \left(\mathbf{E} \cdot \boldsymbol{u} - \mathbf{C}\dot{\boldsymbol{\theta}} - \mathbf{G}\right) = \mathbf{H}^{-1}\mathbf{E}. \tag{4.14}$$

ja que $\mathbf{C}$ i $\mathbf{G}$ no depenen de $\boldsymbol{u}$. Si substituïm les Eqs. (4.13) i (4.14) a l'Eq. (4.12) arribem finalment a l'expressió

$$\mathbf{B} = \left.\frac{\partial f}{\partial \boldsymbol{u}}\right|_{\substack{\boldsymbol{x}=\boldsymbol{x}_o \\ \boldsymbol{u}=\boldsymbol{u}_o}} = \begin{bmatrix} \mathbf{0} \\ \mathbf{H}^{-1}\mathbf{E} \end{bmatrix}. \tag{4.15}$$

5

Disseny de la llei de control

Amb el model de l'Eq. (4.3) podem ja dissenyar lleis de control lineals que estabilitzin l'Hexapole a l'estat desitjat. Tant si l'Eq. (4.3) prové del model mecànic (Eq. (2.36)) com de l'electromecànic (Eq. (3.16)), el procés de disseny serà el mateix. L'única cosa que variarà és el tipus de consigna que obtindrem pels motors, que serà un senyal de parell si usem l'Eq. (2.36), o de voltatge si usem l'Eq. (3.16). En aquest capítol explicarem com podem comprovar que el sistema sigui controlable, i com es pot dissenyar i ajustar una llei de control que compleixi amb les restriccions de l'Hexapole.

5.1 Anàlisi de controlabilitat

Abans de dissenyar la llei de control caldrà veure si el sistema linealitzat és controlable amb les accions de què disposem. Donat un sistema lineal de la forma

$$\dot{\boldsymbol{x}} = \mathbf{A} \cdot (\boldsymbol{x} - \boldsymbol{x}_o) + \mathbf{B} \cdot (\boldsymbol{u} - \boldsymbol{u}_o) \tag{5.1}$$

on A i B són matrius constants, $\boldsymbol{x}_o$ és un punt d'equilibri i $\boldsymbol{u}_o$ és l'acció necessària per mantenir aquest equilibri, es diu que el sistema és controlable si és possible dissenyar una llei de control que porti el sistema fins a l'estat $\boldsymbol{x}_o$ en temps finit, independentment de quines siguin les seves condicions inicials [11].

Per verificar si un sistema lineal és controlable podem utilitzar el següent resultat [11]. El sistema de l'Eq. (5.1) és controlable si, i només si, la matriu

$$\mathbf{C}_o = \begin{bmatrix} \mathbf{B} & \mathbf{AB} & \mathbf{A}^2\mathbf{B} & \dots & \mathbf{A}^{n-1}\mathbf{B} \end{bmatrix} \tag{5.2}$$

té rang n, on n és la dimensió de l'espai d'estats (16 en el cas de l'Hexapole). L'estimació del rang de la matriu $\mathbf{C}_o$, no obstant, és molt sensible als errors d'arrodoniment de les components

de $\mathbf{A}$. És per això que, per determinar la controlabilitat, normalment es transforma el sistema de l'Eq. (5.1) en el que s'anomena la seva forma esglaonada, en la qual els modes incontrolables apareixen factoritzats a la part superior esquerra de les matrius $\mathbf{A}$ i $\mathbf{B}$. Aleshores es pot afirmar que el sistema és controlable si i només si no apareixen aquests modes incontrolables en la forma esglaonada [1]. En aquest treball hem utilitzat aquest darrer test, que s'implementa fàcilment amb la comanda `ctrbf` de MATLAB.

5.2 Disseny d'un regulador quadràtic lineal

A l'hora de dissenyar un controlador per a mantenir un pèndol en la seva posició d'equilibri inestable cal tenir en compte les limitacions que hi pugui haver en l'actuació del robot. En el nostre cas, l'Hexapole té un espai de treball reduït, i els seus motors no poden efectuar un parell gaire elevat. Una tècnica de control que permet tenir en compte aquestes limitacions és la basada en els reguladors quadràtics lineals. Aquesta tècnica obté una llei de control que fa que la trajectòria $\boldsymbol{x}(t)$ del sistema cap al punt d'equilibri tingui cost mínim. S'assumeix que aquest cost té la forma

$$J\left(\boldsymbol{x}(t), t_o\right) = \int_{t_o}^{t} \left[\left(\boldsymbol{x} - \boldsymbol{x}_o\right)^T \mathbf{Q} \left(\boldsymbol{x} - \boldsymbol{x}_o\right) + \left(\boldsymbol{u} - \boldsymbol{u}_o\right)^T \mathbf{R} \left(\boldsymbol{u} - \boldsymbol{u}_o\right) \right] dt, \tag{5.3}$$

on t_o és l'instant inicial de la trajectòria $\boldsymbol{x}(t)$, $\boldsymbol{x}_o$ i $\boldsymbol{u}_o$ són l'estat d'equilibri desitjat i l'acció necessària per mantenir aquest estat, i $\mathbf{Q}$ i $\mathbf{R}$ són matrius que penalitzen l'error d'estat $(\boldsymbol{x} - \boldsymbol{x}_o)$ i d'acció $(\boldsymbol{u} - \boldsymbol{u}_o)$, i són conegudes a priori. $\mathbf{Q}$ ha de ser semidefinida positiva, i $\mathbf{R}$ ha de ser definida positiva. Normalment, aquestes matrius es trien utilitzant criteris heurístics (Secció 5.3).

Amb aquesta tècnica, la llei de control obtinguda té la forma

$$\boldsymbol{u}(t) = \boldsymbol{u}_o(t) - \mathbf{K} \cdot \left(\boldsymbol{x}(t) - \boldsymbol{x}_o\right) \tag{5.4}$$

on $\mathbf{K}$ és una matriu constant de guany que es calcula com [2]

$$\mathbf{K} = \mathbf{R}^{-1}\mathbf{B}\mathbf{P}, \tag{5.5}$$

on $\mathbf{P}$ és la solució de l'equació de Riccati de temps continu

$$\mathbf{A}^T\mathbf{P} + \mathbf{P}\mathbf{A} - \left(\mathbf{P}\mathbf{B}\right)\mathbf{R}^{-1}\left(\mathbf{B}^T\mathbf{P}\right) + \mathbf{Q} = \mathbf{0}. \tag{5.6}$$

[1] La matriu $\mathbf{C}_o$ es pot obtenir amb MATLAB via la comanda `ctrb`, i el test de controlabilitat basat en la forma esglaonada es pot realitzar amb la comanda `ctrbf` [12].

[2] La matriu $\mathbf{K}$ es pot calcular utilitzant la comanda LQR de MATLAB.

A [13] es demostra que si realimentem el sistema amb la llei de l'Eq. (5.4) i el valor de $\mathbf{K}$ esmentat, el sistema resultant,

$$\dot{\boldsymbol{x}} = \mathbf{A} \cdot (\boldsymbol{x} - \boldsymbol{x}_o) + \mathbf{B} \cdot [\boldsymbol{u}_o - \mathbf{K}(\boldsymbol{x} - \boldsymbol{x}_o)],$$

convergirà cap a $\boldsymbol{x}_o$ seguint trajectòries $\boldsymbol{x}(t)$ que minimitzen la funció de l'Eq. (5.3).

Fixem-nos que ajustant els valors de $\mathbf{Q}$ i $\mathbf{R}$, podem penalitzar més o menys els errors d'estat $(\boldsymbol{x} - \boldsymbol{x}_o)$ i d'acció $(\boldsymbol{u} - \boldsymbol{u}_o)$, i per tant obtenir lleis de control que mantinguin el sistema més o menys proper a $\boldsymbol{x}_o$ i $\boldsymbol{u}_o$.

5.3 Ajust del controlador

Amb el que s'ha vist a la secció anterior veiem que un aspecte clau a l'hora d'ajustar la llei de control serà la construcció adequada de $\mathbf{Q}$ i $\mathbf{R}$. A tal efecte, una de les metodologies que més s'utilitza és la basada en l'heurístic de *Bryson* [14]. Aquest heurístic és força útil a l'hora de tenir en compte les restriccions del sistema, tant les de posició (espai de treball) com les de velocitat i actuació (voltatges/parells limitats) com és el nostre cas. L'heurístic consisteix en fixar $\mathbf{Q}$ i $\mathbf{R}$ així

$$\mathbf{Q} = \begin{bmatrix} \frac{1}{(x_{1_{max}})^2} & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \frac{1}{(x_{n_{max}})^2} \end{bmatrix},$$

$$\mathbf{R} = \begin{bmatrix} \frac{1}{(u_{1_{max}})^2} & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \frac{1}{(u_{n_{max}})^2} \end{bmatrix},$$

on $x_{i_{max}}$ i $u_{i_{max}}$ són el valor màxim permès per a la variable x_i i la u_i, respectivament (se suposa que els intervals de factibilitat d'aquestes variables són de la forma $(-x_{i_{max}}, x_{i_{max}})$ i $(u_{i_{max}}, u_{i_{max}})$).

L'heurístic de Bryson no ens garanteix que aplicant la metodologia tal i com s'indica quedem sempre dins dels rangs especificats. Així i tot, per tal d'aconseguir una aproximació inicial de la llei de control és una bona eina, ja que amb els valors de $\mathbf{Q}$ i $\mathbf{R}$ esmentats sovint s'aconsegueix que el sistema es mogui respectant aquests rangs. Posteriorment, sempre podem aplicar altres ajustos de forma progressiva en funció dels resultats que es vagin obtenint en simulacions en anell tancat. Es tracta de procedir cíclicament amb la base que ofereix l'heurístic de Bryson fins a arribar als resultats que s'esperen. En última instància, si les condicions del problema són

massa estrictes, podria arribar a ser impossible mantenir el sistema en els rangs $(-x_{i_{max}}, x_{i_{max}})$ i $(-u_{i_{max}}, u_{i_{max}})$ desitjats. En aquest cas caldria plantejar-se la relaxació d'aquests rangs utilitzant motors més potents, o redissenyant el robot per ampliar el seu espai de treball.

5.4 Particularització a l'Hexapole

L'Hexapole s'ha modelitzat amb 8 coordenades generalitzades $\boldsymbol{\theta}$, que juntament amb les velocitats generalitzades $\dot{\boldsymbol{\theta}}$ conformen un espai d'estat de dimensió 16. A més, com que el sistema disposa de sis motors, el vector d'acció $\boldsymbol{u}$ serà de dimensió 6. Això vol dir que les matrius $\mathbf{A}$ i $\mathbf{B}$ seran 16×16 i 16×6, respectivament. Les matrius $\mathbf{Q}$ i $\mathbf{R}$ seran, per la seva banda, 16×16 i 6×6.

Per tal de controlar millor les variables que es volen penalitzar separarem la matriu $\mathbf{Q}$ en dos blocs $\mathbf{Q}_1$ i $\mathbf{Q}_2$. El primer bloc penalitzarà només les configuracions del sistema i prendrà la forma següent

$$\mathbf{Q}_1 = \begin{bmatrix} \frac{1}{(\theta_{1_{max}})^2} & \cdots & \mathbf{0} & & \\ \vdots & \ddots & \vdots & \mathbf{0} & \\ \mathbf{0} & \cdots & \frac{1}{(\theta_{6_{max}})^2} & & \\ & & & \frac{1}{(\beta_{1_{max}})^2} & 0 \\ & \mathbf{0} & & 0 & \frac{1}{(\beta_{2_{max}})^2} \end{bmatrix}. \tag{5.7}$$

El segon bloc penalitzarà només les velocitats i vindrà donat per

$$\mathbf{Q}_2 = \begin{bmatrix} \frac{1}{(\dot{\theta}_{1_{max}})^2} & \cdots & \mathbf{0} & & \\ \vdots & \ddots & \vdots & \mathbf{0} & \\ \mathbf{0} & \cdots & \frac{1}{(\dot{\theta}_{6_{max}})^2} & & \\ & & & \frac{1}{(\dot{\beta}_{1_{max}})^2} & 0 \\ & \mathbf{0} & & 0 & \frac{1}{(\dot{\beta}_{2_{max}})^2} \end{bmatrix}. \tag{5.8}$$

Amb aquests dos blocs, la matriu $\mathbf{Q}$ serà

$$\mathbf{Q} = \begin{bmatrix} \mathbf{Q}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_2 \end{bmatrix}$$

A més, per tenir més control de cara al post-ajust d'aquesta matriu en funció dels resultats de les

simulacions, introduirem dos coeficients ponderadors α_1 i α_2. D'aquesta manera la matriu que finalment considerarem a la funció de cost serà

$$\mathbf{Q} = \begin{bmatrix} \alpha_1 \mathbf{Q}_1 & \mathbf{0} \\ \mathbf{0} & \alpha_2 \mathbf{Q}_2 \end{bmatrix}.$$

Pel que fa a la matriu $\mathbf{R}$, es pot procedir de forma similar a com s'ha fet amb $\mathbf{Q}$. Tanmateix, com que el vector d'estat $\boldsymbol{u} = (u_1, \ldots, u_6)$ és homogeni en unitats, no serà necessari subdividir $\mathbf{R}$ en blocs, i prendrem

$$\mathbf{R} = \alpha_3 \begin{bmatrix} \frac{1}{(u_{1_{max}})^2} & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \frac{1}{(u_{n_{max}})^2} \end{bmatrix},$$

on α_3 és un coeficient ponderador que s'anirà ajustant en funció de les simulacions.

6 Simulació del sistema

En aquest capítol explicarem diverses simulacions que s'han fet per verificar la coherència dels models obtinguts i les lleis de control associades. Començarem explicant alguns aspectes que s'han de tenir en compte a l'hora d'integrar les equacions del moviment (Secció 6.1). Després veurem com s'han comprovat els models desenvolupats, simulant el moviment de l'Hexapole sota accions que compensin la gravetat (Secció 6.2). Finalment donarem els valors numèrics dels models linealitzats i les lleis dissenyades, i verificarem l'efectivitat d'aquestes lleis simulant el sistema des de diferents estats inicials, o sota pertorbacions de força no modelitzades. Les Taules 6.1, 6.2 i 6.5 detallen els paràmetres dinàmics que s'han assumit en tots els càlculs, i l'Annex C conté el codi font de tots els programes utilitzats.

6.1 Integració de les equacions del moviment

6.1.1 Nocions bàsiques

Donat un model dinàmic per a l'Hexapole

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}), \tag{6.1}$$

com pot ser el mecànic de l'Eq. (2.36), o l'electromecànic de l'Eq. (3.16), el nostre objectiu és ara simular-lo durant un cert interval de temps. És a dir, volem determinar la trajectòria $\boldsymbol{x}(t)$ que seguirà el sistema durant aquest interval, donat l'estat $\boldsymbol{x}_{ini}$ a l'inici de l'interval, i les accions $\boldsymbol{u} = \boldsymbol{u}(\boldsymbol{x})$ aplicades durant l'interval. Aquestes accions $\boldsymbol{u}(\boldsymbol{x})$ poden ser, per exemple, l'acció nul·la $\boldsymbol{u}(\boldsymbol{x}) = \boldsymbol{0}$, una acció $\boldsymbol{u}(\boldsymbol{x})$ que compensi la gravetat, o bé la llei de control $\boldsymbol{u} = \boldsymbol{u}_o - \mathbf{K}(\boldsymbol{x} - \boldsymbol{x}_o)$ de l'Eq. (5.4). Per dur a terme aquesta simulació es poden aplicar nombrosos mètodes, però independentment de quin sigui l'utilitzat, tots es basen en aplicar una versió discreta de l'Eq. (6.1). L'estratègia més simple és la del mètode d'Euler, que considera la

derivada $\dot{\boldsymbol{x}}$ com un quocient de petits increments

$$\dot{\boldsymbol{x}} = \frac{\boldsymbol{x}_{n+1} - \boldsymbol{x}_n}{h}, \tag{6.2}$$

on $\boldsymbol{x}_n$ i $\boldsymbol{x}_{n+1}$ són els estats del sistema per als instants t_n i $t_{n+1} = t_n + h$, i h és l'anomenat pas d'integració. Si h és prou petit, això permet escriure

$$\frac{\boldsymbol{x}_{n+1} - \boldsymbol{x}_n}{h} = f(\boldsymbol{x}_n, \boldsymbol{u}_n) \tag{6.3}$$

o, multiplicant per h i aillant $\boldsymbol{x}_{n+1}$,

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + h \cdot f(\boldsymbol{x}_n, \boldsymbol{u}_n), \tag{6.4}$$

que proporciona una regla recurrent per trobar la trajectòria $\boldsymbol{x}(t)$ a partir de $\boldsymbol{x}_{ini}$. La seqüència d'estats obtinguda, $\boldsymbol{x}_1 = \boldsymbol{x}_{ini}, \boldsymbol{x}_2, \boldsymbol{x}_3 \ldots$, serà tant més propera a la solució exacta $\boldsymbol{x}(t)$ com més petit sigui el pas h utilitzat.

Cal dir que el mètode d'Euler rarament s'utilitza perquè acumula errors ràpidament, fent que la seqüència $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \ldots$ es desvii ràpidament de la solució real $\boldsymbol{x}(t)$. Aquest mètode, però, il·lustra de manera senzilla el que fan altres mètodes més sofisticats. Tots ells implementen versions més o menys el·laborades de l'Eq. (6.4), avaluant f en un o més estats del sistema. El pas h que utilitzen, però, és sovint variable, per tal d'obtenir trajectòries acurades quan f varia ràpidament. Com en el mètode d'Euler, molts mètodes obtenen $\boldsymbol{x}_{n+1}$ com a funció explícita de $\boldsymbol{x}_n$, però també hi ha mètodes implícits en els quals $\boldsymbol{x}_{n+1}$ s'obté resolent numèricament una equació algebraica. A més, hi ha mètodes multi-pas que obtenen $\boldsymbol{x}_{n+1}$ a partir de diversos estats $\boldsymbol{x}_n, \boldsymbol{x}_{n-1}, \boldsymbol{x}_{n-2}, \ldots$ obtinguts prèviament. Sigui quin sigui el mètode que emprem, però, per calcular $\boldsymbol{x}_{n+1}$ haurem de proporcionar una rutina que avalui $f(\boldsymbol{x}, \boldsymbol{u})$. Vegem com podem fer-ho en el cas de l'Hexapole.

6.1.2 Càlcul de la derivada temporal de l'estat

Recordem que a l'Hexapole $f(\boldsymbol{x}, \boldsymbol{u})$ té la forma abstracta de l'Eq. (4.1), que repetim aquí per conveniència:

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}) = \begin{bmatrix} \dot{\boldsymbol{\theta}} \\ \mathbf{H}^{-1}\left(\mathbf{E} \cdot \boldsymbol{u} - \mathbf{C} \cdot \dot{\boldsymbol{\theta}} - \mathbf{G}\right) \end{bmatrix}. \tag{6.5}$$

Com que en cada pas d'integració disposarem de les coordenades motor $\boldsymbol{x} = (\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})$ i del valor de $\boldsymbol{u}$ aplicat (per exemple, $\boldsymbol{u}$ serà el valor proporcionat per la llei de control $\boldsymbol{u} = \boldsymbol{u}_o - \mathbf{K}(\boldsymbol{x} - \boldsymbol{x}_o)$), l'avaluació de (6.5) és en principi senzilla. Recordem però que a l'Eq. (6.5) els termes $\mathbf{H}$, $\mathbf{E}$, $\mathbf{C}$,

i $\mathbf{G}$ estan escrits en funció de les coordenades plataforma $\boldsymbol{q}$ i $\dot{\boldsymbol{q}}$. Així doncs, per avaluar aquests termes caldrà primer obtenir $\boldsymbol{q}$ i $\dot{\boldsymbol{q}}$ a partir de $\boldsymbol{\theta}$ i $\dot{\boldsymbol{\theta}}$.

D'una banda, el valor de $\boldsymbol{q}$ es pot obtenir a partir de $\boldsymbol{\theta}$ resolent el problema cinemàtic directe de l'Hexapole. La Secció A.4 de l'Annex A explica com es pot implementar una funció

$$\boldsymbol{q} = F_{FK}(\boldsymbol{\theta}, \boldsymbol{q}_a)$$

que resolgui aquest problema numèricament, partint d'una aproximació $\boldsymbol{q}_a$ de la solució buscada. Durant la simulació, $\boldsymbol{q}_a$ serà simplement el valor de $\boldsymbol{q}$ obtingut a la iteració anterior. Per altra banda, el valor de $\dot{\boldsymbol{q}}$ s'obté resolent el problema cinemàtic instantani directe de l'Hexapole. La Secció A.3.2 de l'Annex A explica que la solució d'aquest problema és simplement

$$\dot{\boldsymbol{q}} = \mathbf{J}_i^{-1}\ \dot{\boldsymbol{\theta}},$$

on $\mathbf{J}_i$ és el Jacobià obtingut a la Secció A.3.1. Un cop es tenen $\boldsymbol{q}$ i $\dot{\boldsymbol{q}}$, l'avaluació de l'Eq. (6.5) és immediata.

6.1.3 Mètodes d'integració utilitzats

A l'hora de resoldre l'Eq. (6.1) per a l'estat inicial $\boldsymbol{x}_{ini}$ es poden aplicar múltiples estratègies, però un mètode que sovint dóna molt bons resultats és el de Dormand-Prince [15]. Aquest mètode implementa una versió més elaborada de l'Eq. (6.4), basada en avaluar sis vegades la funció $f(\boldsymbol{x}, \boldsymbol{u})$. Donada l'estimació de la solució a l'instant t_n, $\boldsymbol{x}_n$, aquestes sis avaluacions s'utilitzen per estimar $\boldsymbol{x}_{n+1}$ dues vegades: una amb una fórmula d'ordre 4, i l'altra amb una fórmula d'ordre 5. El valor d'$\boldsymbol{x}_{n+1}$ obtingut amb la primera fórmula és el que es pren com a resultat de la integració, i la diferència entre els valors d'$\boldsymbol{x}_{n+1}$ obtinguts amb les dues fórmules es pren com l'error d'$\boldsymbol{x}_{n+1}$. Aquesta diferència permet adaptar el pas d'integració per tal que l'error de $\boldsymbol{x}_{n+1}$ es mantingui sempre dins de cotes admissibles. El mètode de Dormand-Prince, a més, és explícit ($\boldsymbol{x}_{n+1}$ s'obté com a funció explícita de $\boldsymbol{x}_n$) i d'un sol pas (per obtenir $\boldsymbol{x}_{n+1}$ utilitza $\boldsymbol{x}_n$ i no cap altre estat anterior) amb la qual cosa sol ser molt eficient a l'hora de calcular trajectòries $\boldsymbol{x}(t)$. És per aquests motius que és el mètode més recomanat a l'entorn MATLAB/SIMULINK. El mètode es pot aplicar cridant la comanda `ode45` de MATLAB.

A l'hora de simular tots els models dinàmics d'aquest projecte, el primer mètode aplicat ha estat sempre el de Dormand-Prince, i els resultats han estat satisfactoris en la majoria de casos, obtenint trajectòries $\boldsymbol{x}(t)$ força acurades amb temps de càlcul relativament curts.

Cal dir però que el mètode de Dormand-Prince només funciona bé en sistemes d'equacions diferencials que no siguin rígids (*non-stiff systems* en anglès). Es diu que un sistema és rígid

(o *stiff*) en relació a un mètode d'integració quan el mètode ha d'adoptar un pas d'integració extraordinàriament petit per mantenir la seva estabilitat, fins i tot quan les variacions de $f(\boldsymbol{x}, \boldsymbol{u})$ són suaus. La noció de rigidesa però, és qualitativa, ja que fins al moment no s'ha pogut caracteritzar de manera rigorosa, i tant sols es disposa de regles heurístiques per poder anticipar si un sistema d'equacions presentarà problemes de rigidesa o no. Per exemple, per un sistema lineal autònom

$$\dot{\boldsymbol{x}} = \mathbf{A} \cdot \boldsymbol{x},$$

on $\mathbf{A}$ té els valors pròpis $\lambda_1, \ldots, \lambda_n$, se sol considerar que el sistema és rígid quan el quocient

$$I_1 = \frac{max\{|Re(\lambda_1)|, \ldots, |Re(\lambda_n)|\}}{min\{|Re(\lambda_1)|, \ldots, |Re(\lambda_n)|\}} \tag{6.6}$$

té un valor alt. Alguns autors, però, remarquen que també cal tenir en compte l'interval de temps (t_o, t_f) en el qual s'efectua la integració, i consideren que el sistema és rígid quan el producte

$$I_2 = max\{|Re(\lambda_1)|, \ldots, |Re(\lambda_n)|\} \cdot (t_f - t_o) \tag{6.7}$$

és força més gran que 1. En aquestes situacions el mètode de Dormand-Prince no sol funcionar bé, i cal recórrer a mètodes d'integració implícits més sofisticats. En el nostre cas, com veurem, apareixeran problemes de rigidesa a l'hora de simular el model mecànic amb llei de control activada, i per solventar-los utilitzarem un mètode implícit d'ordre variable proposat per Shampine i Reichelt [16]. En aquest mètode, $\boldsymbol{x}_{n+1}$ s'obté amb fórmules de derivació implícita que es resolen numèricament. Aquestes fórmules donen lloc a aproximacions d'ordre entre 1 i 5 de la derivada $\dot{\boldsymbol{x}} = \frac{d\boldsymbol{x}}{dt}$, i quina fórmula es tria en cada moment depèn d'heurístics molt el·laborats [16]. El mètode es pot aplicar amb la comanda `ode15s` de MATLAB, utilitzant la mateixa sintaxi que la de la comanda `ode45` [12].

6.2 Simulació amb compensació gravitatòria

Abans de construir una llei de control cal assegurar que els models que hem desenvolupat es comporten de manera consistent. És per aquest motiu que hem orientat les primeres simulacions d'aquest treball a verificar les Eqs. (2.36) i (3.16) per als paràmetres concrets de l'Hexapole. Aquests paràmetres es mostren a les Taules 6.1 i 6.2.

Per verificar el model de l'Eq. (2.36) hem fet la prova següent. Suposem que en cada instant

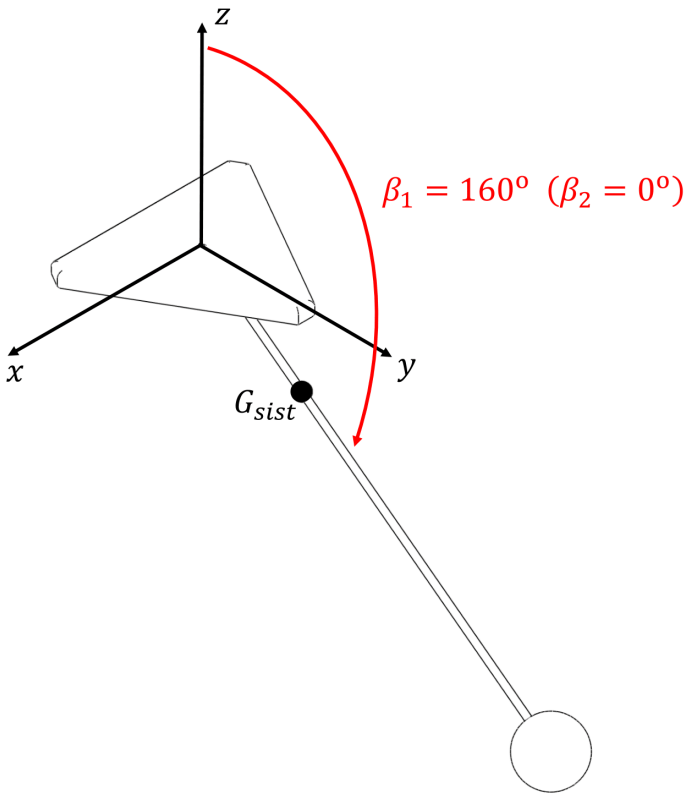


Figura 6.1: Condicions inicials de la simulació amb compensació gravitatòria.

l'Hexapole aplica uns parells $\boldsymbol{u}_\tau$ que generen, en conjunt, el torsor

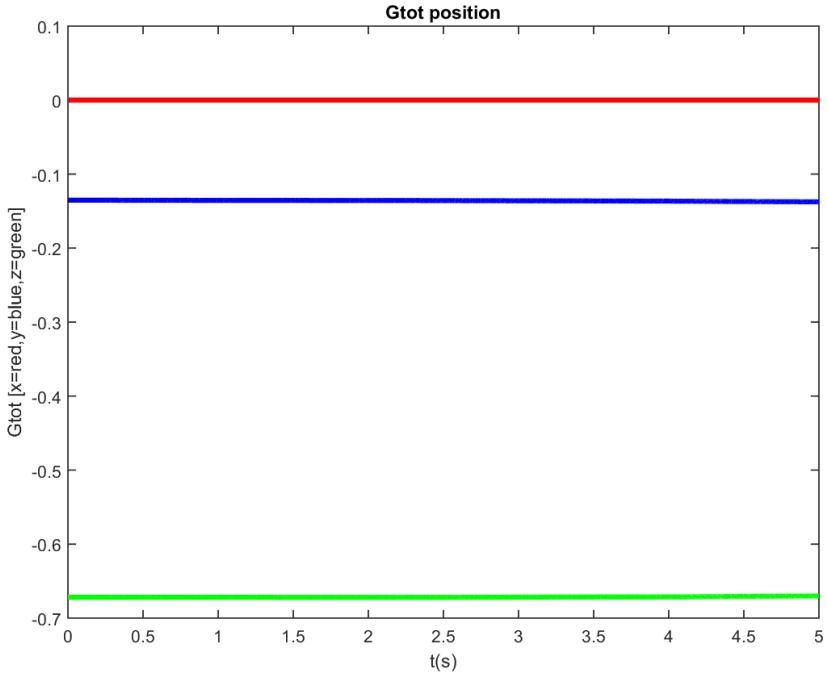
$$\hat{\mathbf{w}}_a = (\underbrace{0, 0, (m_{plat} + m_{pole}) \cdot g}_{\mathbf{F}_{comp}}, 0, 0, 0)$$

sobre la plataforma, referit al punt G_{plat}. Com que la component de força $\mathbf{F}_{comp}$ compensa exactament el pes total de la plataforma i el pèndol, si apliquem el teorema de la quantitat de moviment al conjunt format per la plataforma i el pèndol tindrem

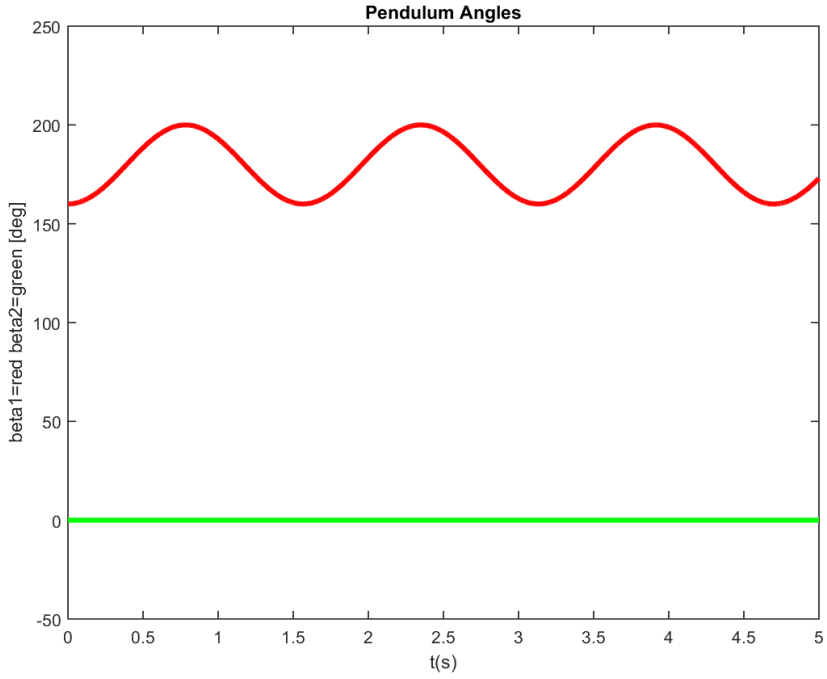
$$m_{tot} \cdot \boldsymbol{a}_{G_{tot}} = \mathbf{0},$$

on $\boldsymbol{a}_{G_{tot}}$ és l'acceleració del centre de masses del conjunt, que denotem per G_{tot}, i $m_{tot} = m_{plat} + m_{pole}$. Això vol dir que si deixem caure el pèndol des d'una posició com la indicada a la Fig. 6.1 (amb $\beta_1 = 160°$ i $\beta_2 = 0°$), i suposem que el contacte pèndol-plataforma és bilateral, G_{tot} s'hauria de mantenir sempre en la mateixa posició, i el pèndol hauria d'oscil·lar en el pla yz.

La simulació d'aquest moviment de caiguda s'ha dut a terme aplicant el mètode de Dormand-

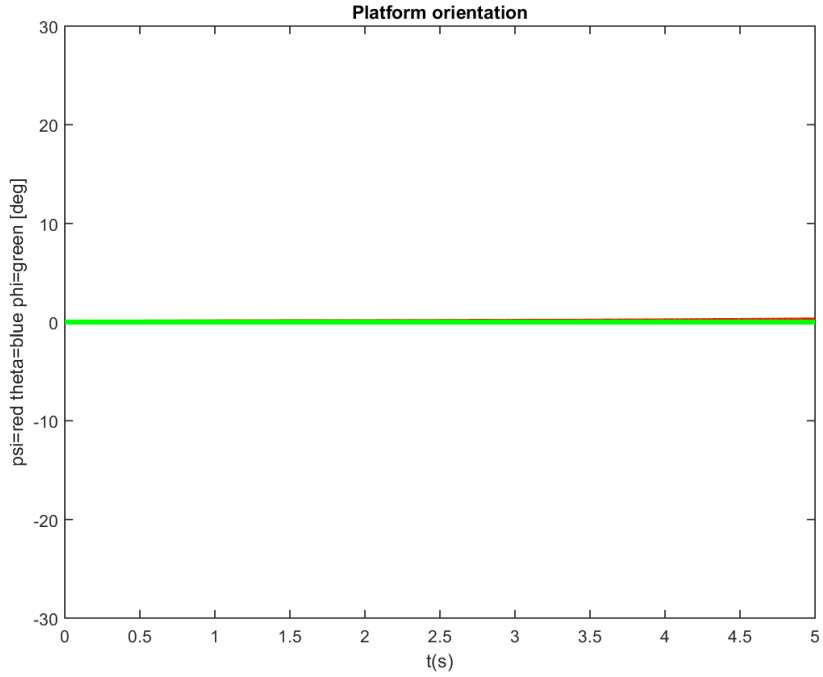


(a) Evolució temporal de la posició del G_{tot}.

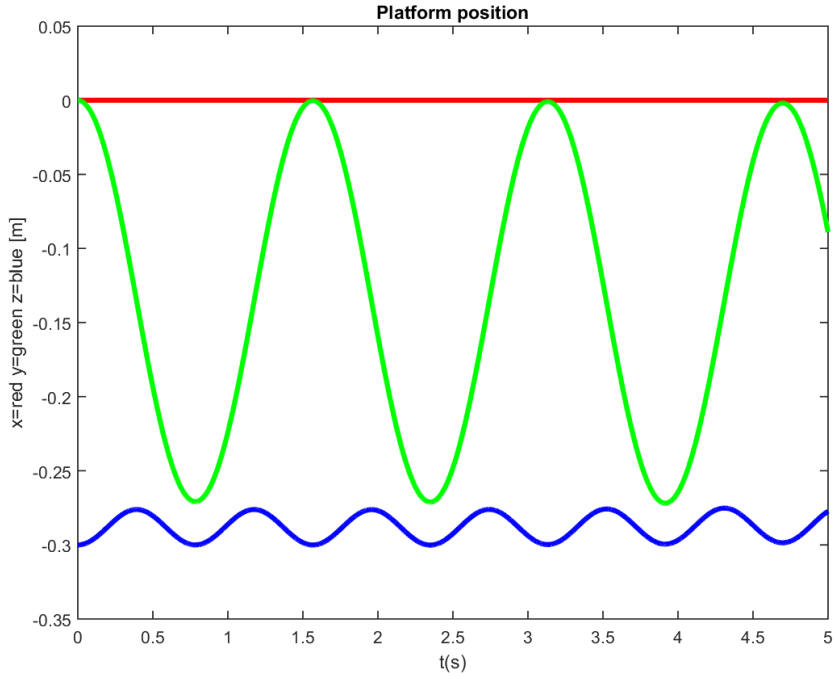


(b) Evolució temporal dels angles del pèndol β_1 i β_2.

Figura 6.2: Simulació amb compensació gravitatòria (1 de 3).

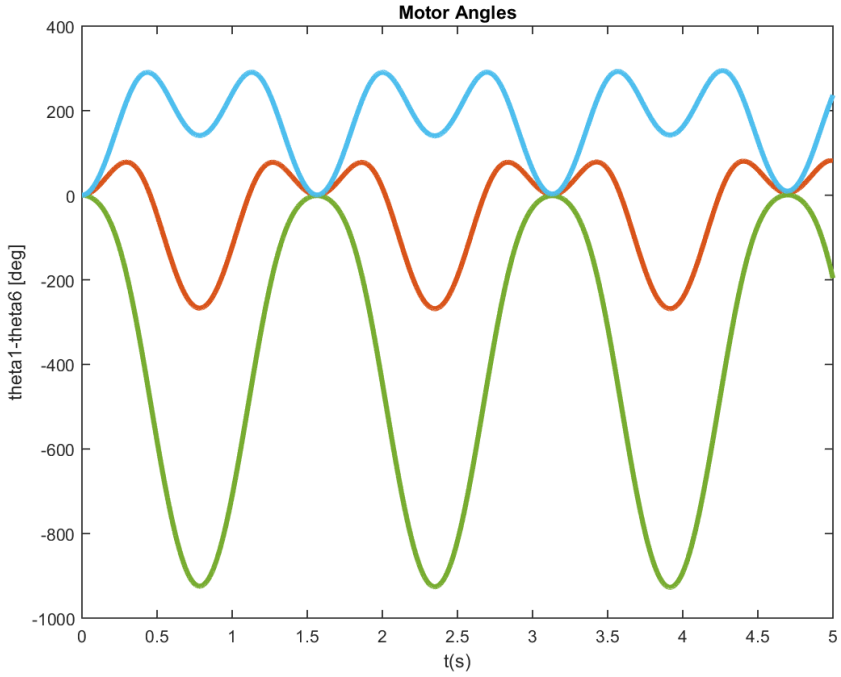


(c) Evolució temporal dels angles d'Euler ψ, θ, φ.

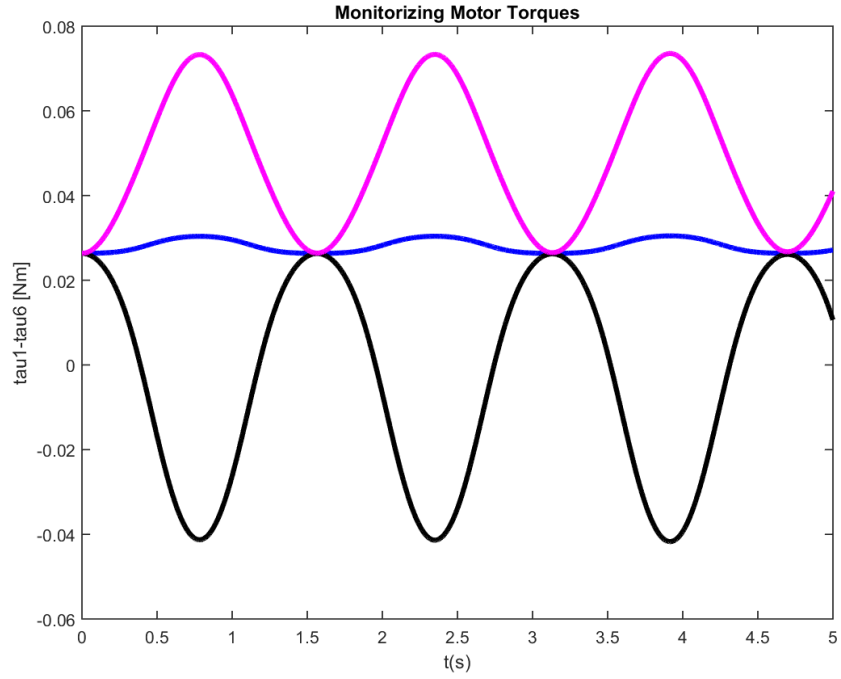


(d) Evolució temporal de la posició (x, y, z) del centre de masses de la plataforma G_{plat}.

Figura 6.2: Simulació amb compensació gravitatòria (2 de 3)



(e) Evolució temporal dels angles dels motors $\theta_1, \ldots, \theta_6$.



(f) Evolució temporal dels parells de compensació gravitatòria $\tau_1, \ldots, \tau_6$.

Figura 6.2: Simulació amb compensació gravitatòria (3 de 3)

Paràmetre	Símbol	Valor
Longitud del pèndol	l_{pole}	0.9672 m
Massa de la plataforma	m_{plat}	0.5989 kg
Massa del pèndol	m_{pole}	0.4147 kg
Tensor d'inèrcia plataforma	$\mathbf{I}_{plat}$	$\begin{bmatrix} 270725.71 & 0 & 0 \\ 0 & 270725.71 & 0 \\ 0 & 0 & 409645.04 \end{bmatrix} \times 10^{-9}$ kgm^2
Tensor d'inèrcia pèndol	$\mathbf{I}_{pole}$	$\begin{bmatrix} 11262205.59 & 0 & 0 \\ 0 & 11262205.59 & 0 \\ 0 & 0 & 77397.82 \end{bmatrix} \times 10^{-9}$ kgm^2

Taula 6.1: Paràmetres utilitzats en el primer model dinàmic de l'Hexapole.

Paràmetre	Símbol	Valor
Resistència	R	$8.3\ \Omega$
Inductància	L	2.03×10^{-3} H
Relació de transmissió	N	193
Rendiment de transmissió	η	0.836
Constant de velocitat	K_ω	93.1 rad/V
Constant de parell	K_t	0.0107 N m/A
Inèrcia eix motor	J_m	8.86×10^{-8} kg m^2
Fricció	b_m	8.87×10^{-8}N ms

Taula 6.2: Parametres utilitzats per modelitzar els servomotors MX-28.

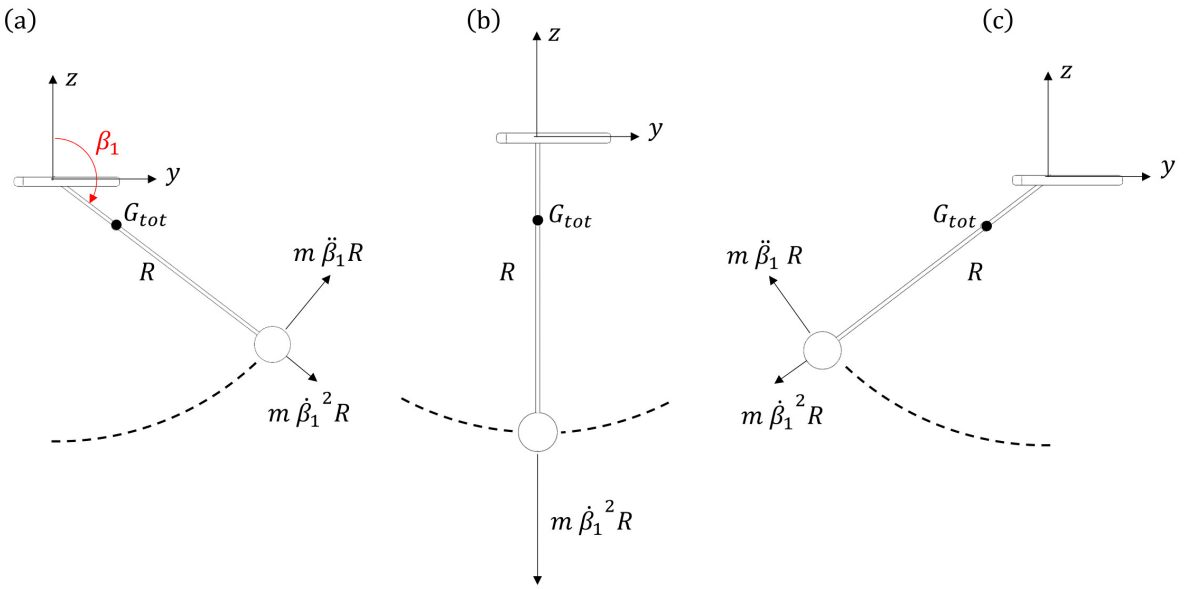


Figura 6.3: Esquema d'un període d'oscil·lació en z i mig en y. **(a)** Instant proper a l'inicial, en el qual la velocitat $\dot{\beta}_1 > 0$ és positiva però petita, i $\ddot{\beta}_1 > 0$. **(b)** Instant en que $\dot{\beta}_1$ és màxima i $\ddot{\beta}_1$ és zero. **(c)** Instant proper al final del període d'oscil·lació en z. En aquest instant segueix essent positiva però torna a ser petita, i $\ddot{\beta}_1 < 0$. El radi R indicat és la distància des de G_{plat} fins al centre de l'esfera del pèndol.

Prince esmentat anteriorment, utilitzant un pas d'integració màxim d'1 milisegon, i una finestra de simulació de 5 segons. Per fer la simulació, en cada instant ha calgut calcular l'acció $\boldsymbol{u}_\tau$ que genera el torsor $\hat{\mathbf{w}}_a$ anterior, però això és fàcil ja que de l'Eq. (2.24) deduïm ràpidament que

$$\boldsymbol{u}_\tau = \rho \cdot \mathbf{J}^{-1} \cdot \hat{\mathbf{w}}, \tag{6.8}$$

on $\mathbf{J}$ és el Jacobià de forces de l'Hexapole, i ρ és el radi de les seves politges motores[1].

La Fig. 6.2 mostra els resultats d'aquesta simulació. Tal i com esperàvem, el centre de masses G_{tot} es manté fix al llarg del temps (Fig. 6.2 **(a)**) i el pèndol es manté oscil·lant en el pla yz (Fig. 6.2 **(b)**). L'evolució de la plataforma es dedueix de les Figs. 6.2 **(c)** i **(d)**. Com que no hi ha parells externs que actuïn sobre la plataforma, aquesta no pot patir canvis d'orientació i els tres angles d'Euler ψ, θ, i φ es mantenen constants. La posició de la plataforma, en canvi, oscil·la periòdicament en el pla yz. Aquesta oscil·lació es deu a les forces d'inèrcia del pèndol, que es transmeten a la plataforma a través del contacte pèndol-plataforma. La Fig. 6.3 mostra aquestes forces durant un període de l'oscil·lació en z, que correspon a mig període de l'oscil·lació en

[1] Aquí cal tenir en compte que com que $\mathbf{J}$ és funció de la configuració de la plataforma, $\boldsymbol{u}_\tau$ prendrà un valor diferent a cada instant de temps.

y. Com es veu, cap al principi del període hi ha una força neta cap a la dreta i cap amunt (Fig. 6.3 **(a)**). És per això que la plataforma comença a oscil·lar cap a la dreta i cap amunt. El pèndol va caient, i en arribar a la posició vertical (Fig. 6.3 **(b)**) $\dot{\beta}_1$ ha de ser màxima (ja que posteriorment el pèndol es frena) i per tant $\ddot{\beta}_1$ ha de ser zero. Això vol dir que la força d'inèrcia és vertical, la coordenada z assoleix el seu valor màxim, i la velocitat en y comença a decréixer. A mida que el pèndol s'aproxima al valor màxim de β_1 (Fig.6.3 **(c)**) la coordenada z s'aproxima al seu valor mínim inicial, i la y cap el seu valor més a la dreta. Poc després, la força d'inèrcia tangencial canvia de signe i per tant provoca el retorn de la plataforma cap a l'esquerra. A les Figs. 6.2 **(e)** i **(f)**, finalment, es poden apreciar les evolucions dels angles dels motors i dels parells respectivament. La simetria de l'Hexapole fa que les corbes coincideixin de dues en dues, i per això només veiem tres corbes en aquestes figures.

Per verificar el model de l'Eq. (3.16) s'ha fet una prova similar a l'anterior, aplicant voltatges als motors que generessin, per qualsevol configuració de la plataforma, un torsor de compensació gravitacional. Els resultats obtinguts han estat també coherents com en el model mecànic, amb la diferència que el moviment de la plataforma queda notablement afectat pel frec viscós dels motors, donant lloc a una oscil·lació esmorteïda.

6.3 Simulació amb la llei de control activada

Una vegada hem verificat la consistència dels models de l'Hexapole, podem ja passar a dissenyar unes lleis de control que estabilitzin el sistema al voltant d'un estat d'equilibri desitjat. Tot seguit veurem com s'ha triat aquest estat (Secció 6.3.1), i com s'ha abordat el procés de linealització i control tant pel que fa al model mecànic (Secció 6.3.2) com a l'electromecànic (Secció 6.3.3). L'objectiu és comprovar l'efectivitat de les lleis de control dissenyades en simulació.

6.3.1 Estat d'equilibri triat

L'estat d'equilibri en el qual es vol estabilitzar el sistema s'ha triat de la següent manera. Com que volem mantenir la plataforma horitzontal i centrada enmig del seu espai de treball (per permetre desplaçaments laterals sense perdre tensió en els cables), els angles d'Euler ψ, θ, i φ s'han fixat a $0°$ i el punt G_{plat} a $(x, y, z) = (0, 0, -0.3)$ m. A més, com que el pèndol es vol mantenir en posició vertical invertida, β_1 i β_2 s'han fixat a $0°$. En ser un estat d'equilibri, les velocitats de la plataforma i del pèndol es volen totes a zero també. Això vol dir que, en unitats SI, l'estat d'equilibri vindrà donat pel vector de coordenades plataforma

$$\boldsymbol{x}_{\boldsymbol{q},o} = (\underbrace{0, 0, -0.3, 0, 0, 0, 0, 0}_{\boldsymbol{q}_o}, \underbrace{0, 0, 0, 0, 0, 0, 0, 0}_{\dot{\boldsymbol{q}}_o})$$

i pel vector de coordenades motor [2]

$$\boldsymbol{x}_{\boldsymbol{\theta},o} = (\underbrace{0,0,0,0,0,0,0,0}_{\boldsymbol{\theta}_o},\underbrace{0,0,0,0,0,0,0,0}_{\dot{\boldsymbol{\theta}}_o})$$

En aquest estat $\boldsymbol{x}_{\boldsymbol{\theta},o}$ l'Hexacrane adopta la configuració que es veu a la Fig. 6.4, on els sistemes de referència $Oxyz$ i $Ox'y'z'$ corresponen, respectivament, a les referències absoluta i relativa de la Fig. 2.1.

Les accions necessàries per mantenir aquest estat són

$$\boldsymbol{u}_{\tau,o} = (0.0263, 0.0263, 0.0263, 0.0263, 0.0263, 0.0263)\ \mathrm{Nm}$$

en el model mecànic, i

$$\boldsymbol{u}_{v,o} = (0.1267, 0.1267, 0.1267, 0.1267, 0.1267, 0.1267)\ \mathrm{V}$$

en el model electromecànic.

6.3.2 Linealització i control del model mecànic

La linealització del model mecànic s'ha dut a terme al voltant del punt $(\boldsymbol{x}_{\theta,o}, \boldsymbol{u}_{\tau,o})$ tenint en compte les coordenades de la Taula 2.1 i els paràmetres dinàmics de la Taula 6.1. Les masses i tensors d'inèrcia utilitzats provenen de suposar que

- La plataforma és un prisma triangular de titani amb densitat uniforme $(4510\ \mathrm{kg/m^3})$, costat de base igual a $147.92 \times 10^{-3}\ \mathrm{m}$ i alçada $10 \times 10^{-3}\ \mathrm{m}$.
- La barra del pèndol és de fibra de carboni de densitat uniforme $(1780\ \mathrm{kg/m^3})$, d'$1\ \mathrm{m}$ de llargada i $5 \times 10^{-3}\ \mathrm{m}$ de diàmetre.
- La massa a l'extrem del pèndol és considera una esfera metàlica massissa d'acer de densitat uniforme $(8000\ \mathrm{kg/m^3})$, i de $45 \times 10^{-3}\ \mathrm{m}$ de diàmetre.

Per donar els valors numèrics obtinguts per les matrius $\mathbf{A}$ i $\mathbf{B}$ del model linealitzat, recordem que en el Capítol 4 hem vist que

$$\mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{H}^{-1}\frac{\partial}{\partial \boldsymbol{\theta}}(\mathbf{E}\boldsymbol{u} - \mathbf{G}) & -\mathbf{H}^{-1}\mathbf{C} \end{bmatrix}$$

[2] La Secció A.4.1 de l'Annex A explica que els valors $(\gamma_1, \ldots, \gamma_6) = (0, \ldots, 0)$ dels angles dels motors són els que, per conveni, corresponen a la configuració $(x, y, z, \psi, \theta, \varphi) = (0, 0, -0.3\mathrm{m}, 0, 0, 0)$ de la plataforma.

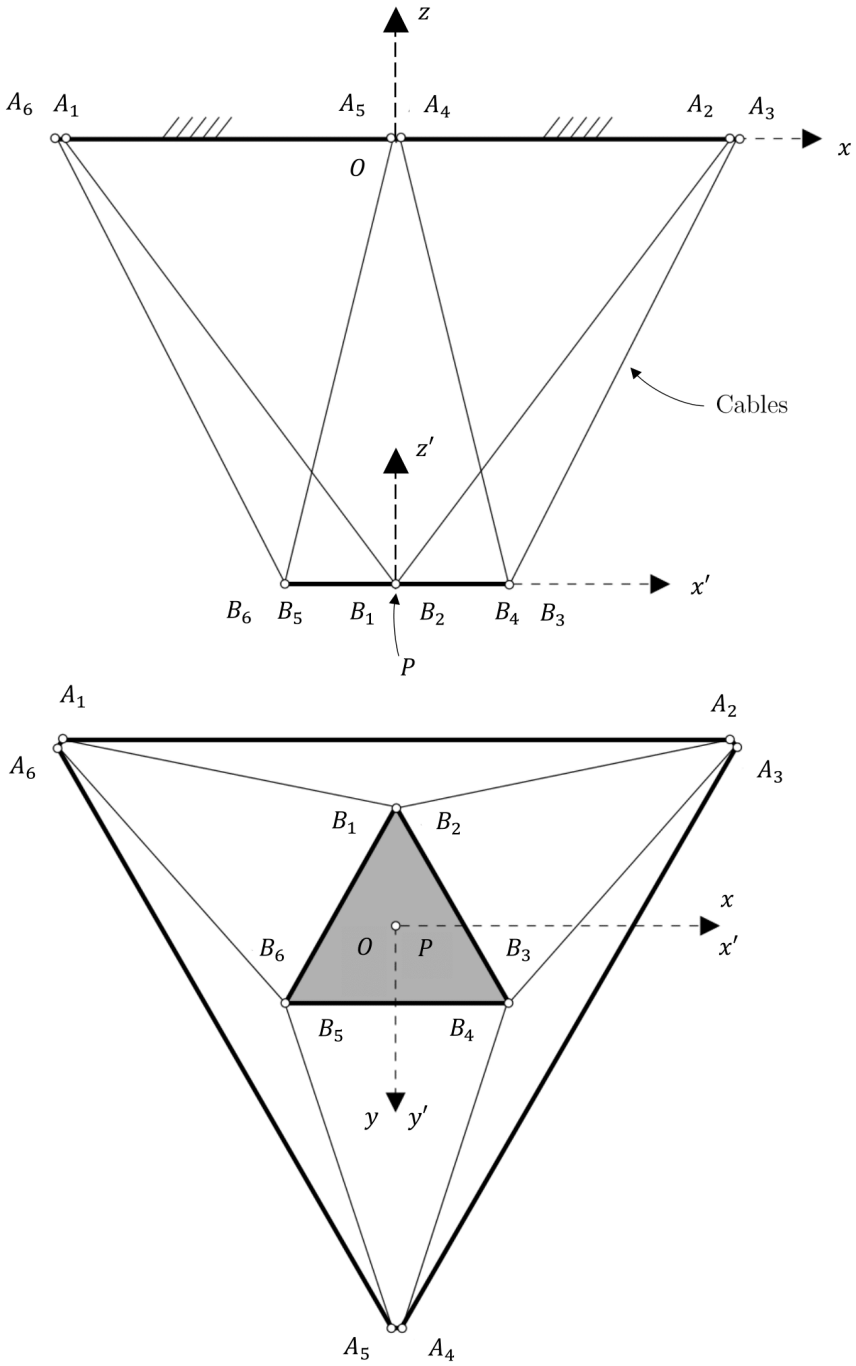


Figura 6.4: Configuració de l'Hexacrane a l'estat d'equilibri.

Bloc	Valor
$\mathbf{H}^{-1}\frac{\partial}{\partial \boldsymbol{\theta}}(\mathbf{E}\boldsymbol{u}-\mathbf{G})$	$\begin{bmatrix} -288,38 & -10,52 & -4,86 & 146,89 & -85,54 & 229,90 & -63,71 & 313,80 \\ -10,52 & -288,38 & 229,90 & -85,54 & 146,89 & -4,86 & -63,71 & -313,80 \\ -85,54 & 229,90 & -288,38 & -10,54 & -4,84 & 146,91 & -239,90 & -212,07 \\ 146,88 & -4,86 & -10,53 & -288,39 & 229,92 & -85,52 & 303,61 & 101,72 \\ -4,86 & 146,88 & -85,52 & 229,92 & -288,39 & -10,53 & 303,61 & -101,72 \\ 229,90 & -85,54 & 146,91 & -4,84 & -10,54 & -288,38 & -239,90 & 212,07 \\ -0,01 & -0,01 & 0,27 & -0,25 & -0,25 & 0,27 & 16,34 & 1,99\times10^{-16} \\ -0,30 & 0,30 & 0,14 & -0,16 & 0,16 & -0,14 & 1,15\times10^{-16} & 16,34 \end{bmatrix}$
$\mathbf{H}^{-1}\mathbf{C}$	$\mathbf{0}_{8\times8}$
$\mathbf{H}^{-1}\mathbf{E}$	$\begin{bmatrix} 169510,85 & 70460,99 & -10401,95 & -99242,18 & -10399,86 & -96529,59 \\ 70460,99 & 169510,85 & -96529,59 & -10399,86 & -99242,18 & -10401,95 \\ -10401,95 & -96529,59 & 169514,02 & 70466,20 & -10423,41 & -99262,06 \\ -99242,18 & -10399,86 & 70466,20 & 169515,66 & -96553,46 & -10423,41 \\ -10399,86 & -99242,18 & -10423,41 & -96553,46 & 169515,66 & 70466,20 \\ -96529,59 & -10401,95 & -99262,06 & -10423,41 & 70466,20 & 169514,02 \\ -16,20 & -16,20 & -61,02 & 77,23 & 77,23 & -61,02 \\ 79,82 & -79,82 & -53,94 & 25,87 & -25,87 & 53,94 \end{bmatrix}$

Taula 6.3: Blocs més rellevants d'$\mathbf{A}$ i $\mathbf{B}$ en el model mecànic linealitzat (en unitats SI). En aquest model, el bloc $-\mathbf{H}^{-1}\mathbf{C}$ és la matriu nul·la de dimensió 8×8.

i

$$\mathbf{B}=\begin{bmatrix}\mathbf{0}\\ \mathbf{H}^{-1}\mathbf{E}\end{bmatrix}.$$

Això vol dir que per especificar $\mathbf{A}$ i $\mathbf{B}$ n'hi ha prou amb donar els valors numèrics dels blocs $\mathbf{H}^{-1}\frac{\partial}{\partial \boldsymbol{\theta}}(\mathbf{E}\boldsymbol{u}-\mathbf{G})$, $-\mathbf{H}^{-1}\mathbf{C}$ i $\mathbf{H}^{-1}\mathbf{E}$. La Taula 6.3 proporciona aquests valors. Utilitzant aquests valors, i el test descrit a la Secció 5.1, es pot veure que el sistema linealitzat resultant és controlable.

El disseny de la llei de control no és una tasca senzilla. Tot i partir d'unes estimacions prou bones de les matrius $\mathbf{Q}$ i $\mathbf{R}$ com les que proporciona l'heurístic de Bryson, és inevitable entrar en un cicle d'assaig i error amb l'ajut de les eines de simulació de MATLAB. Per dissenyar i ajustar la llei s'ha tingut en compte que:

- L'angle γ_i de cada motor s'ha de mantenir en el rang $(-6.98, 6.98)$ rad per garantir que la plataforma no surti de l'espai de treball assolible amb els cables en tensió.
- Els motors Dynamixel MX-28 només poden proporcionar velocitats en el rang $(-7.01, 7.01)$ rad/s.
- A la sortida del reductor, el parell d'aquests motors ha d'estar en el rang $(-0.08, 0.08)$ Nm.
- Volem ser capaços d'estabilitzar orientacions del pèndol que es trobin dins un con d'obertura $3°$ respecte de la vertical.
- Orientativament, permetrem velocitats $\dot{\beta}_1$ i $\dot{\beta}_2$ en el rang $(-5, 5)$ rad/s.

Tenint en compte les restriccions anteriors aquests rangs, els valors màxims que hem considerat a les Eqs. (5.7) i (5.8) són els següents:

- $\gamma_{i,max} = 6.98$ rad
- $\dot{\gamma}_{i,max} = 7.01$ rad/s
- $\beta_{i,max} = 3$ rad
- $\dot{\beta}_{i,max} = 5$ rad/s
- $u_{i,max} = 0.08$ Nm

Si a més fixem $\alpha_1 = 200$, $\alpha_2 = 1 \times 10^9$, i $\alpha_3 = 1.5 \times 10^{12}$, la llei de control òptim obtinguda amb les Eqs. (5.5) i (5.6) és

$$\boldsymbol{u} = \boldsymbol{u}_o - \mathbf{K} \cdot (\boldsymbol{x} - \boldsymbol{x}_o) \tag{6.9}$$

on $\mathbf{K} = [\mathbf{K}_1, \mathbf{K}_2]$, i $\mathbf{K}_1$ i $\mathbf{K}_2$ són les dues matrius 6×8 indicades a la Taula 6.4.

Si ara introduïm aquesta llei en el model linealitzat obtenim

$$\dot{\boldsymbol{x}} = \mathbf{A} \cdot (\boldsymbol{x} - \boldsymbol{x}_o) + \mathbf{B} \cdot (\boldsymbol{u}_o - \mathbf{K} \cdot (\boldsymbol{x} - \boldsymbol{x}_o))$$

o, ordenant termes,

$$\dot{\boldsymbol{x}} = (\mathbf{A} - \mathbf{B} \cdot \mathbf{K}) \cdot (\boldsymbol{x} - \boldsymbol{x}_o) + \mathbf{B} \cdot \boldsymbol{u}_o$$

d'on es veu que el transitori del sistema ve caracteritzat pels valors propis de la matriu $\mathbf{A} - \mathbf{B} \cdot \mathbf{K}$.

Aquests valors són

$$\begin{aligned}
\lambda_1 &= -82.8404 + 0i,\\
\lambda_2 &= -82.8571 + 0i,\\
\lambda_3 &= -56.0832 + 0i,\\
\lambda_4 &= -13.2849 + 0i,\\
\lambda_5 &= -1.5338 + 6.0963i,\\
\lambda_6 &= -1.5338 - 6.0963i,\\
\lambda_7 &= -1.5338 + 6.0963i,\\
\lambda_8 &= -1.5338 - 6.0963i,\\
\lambda_9 &= -2.9587 + 1.9381i,\\
\lambda_{10} &= -2.9587 - 1.9381i,\\
\lambda_{11} &= -5.1196 + 0i,\\
\lambda_{12} &= -5.1194 + 0i,\\
\lambda_{13} &= -3.4760 + 0i,\\
\lambda_{14} &= -3.4760 + 0i,\\
\lambda_{15} &= -3.2581 + 0i,\\
\lambda_{16} &= -3.2581 + 0i.
\end{aligned}$$

Per tant, com que simularem el sistema durant 5 segons, els indicadors de rigidesa I_1 i I_2 de les Eqs. (6.6) i (6.7) són

$$I_1 = \frac{82.8571}{1.5338} = 54.02$$

i

$$I_2 = 82.8571 \cdot 5 = 414.28.$$

Com veiem, I_1 és bastant gran, i I_2 és molt més gran que 1, amb la qual cosa és possible que si realimentem el model no lineal de l'Eq. (2.36) amb la llei (6.9) tinguem problemes de rigidesa. En aplicar el mètode de Dormand-Prince amb `ode45` veiem que realment és així. Les simulacions progressen correctament però a un ritme extremadament lent, amb passos d'integració de centèssimes de milisegon en molts moments. És per això que per integrar aquest sistema és millor utilitzar el mètode de Shampine i Reichelt (`ode15`). En totes les proves realitzades aquest mètode ha obtingut simulacions prou acurades en pocs segons de càlcul, fixant un pas d'integració màxim de $1\ \mathrm{ms}$

Robustesa a condicions inicials diverses

Utilitzant el mètode de simulació que acabem d'explicar es pot veure que la llei de control dissenyada estabilitza correctament el pèndol quan aquest s'inicia amb orientacions dins un

Bloc	Valor
$\mathbf{K}_1$	$\begin{bmatrix} -2,42 & 2,47 & 0,68 & -0,89 & 1,74 & -1,57 & -106,73 & 521,09 \\ 2,47 & -2,42 & -1,57 & 1,74 & -0,89 & 0,68 & -106,73 & -521,09 \\ 1,74 & -1,57 & -2,42 & 2,46 & 0,68 & -0,89 & -397,90 & -352,98 \\ -0,89 & 0,68 & 2,46 & -2,42 & -1,57 & 1,74 & 504,63 & 168,10 \\ 0,68 & -0,89 & 1,74 & -1,57 & -2,42 & 2,46 & 504,63 & -168,10 \\ -1,57 & 1,74 & -0,89 & 0,68 & 2,46 & -2,42 & -397,90 & 352,98 \end{bmatrix} \times 10^{-3}$
$\mathbf{K}_2$	$\begin{bmatrix} -0,64 & 0,90 & 0,29 & -0,29 & 0,61 & -0,61 & -32,26 & 157,57 \\ 0,90 & -0,64 & -0,61 & 0,61 & -0,29 & 0,29 & -32,26 & -157,57 \\ 0,61 & -0,61 & -0,64 & 0,90 & 0,29 & -0,29 & -120,32 & -106,73 \\ -0,29 & 0,29 & 0,90 & -0,64 & -0,61 & 0,61 & 152,59 & 50,84 \\ 0,29 & -0,29 & 0,61 & -0,61 & -0,64 & 0,90 & 152,59 & -50,84 \\ -0,61 & 0,61 & -0,29 & 0,29 & 0,90 & -0,64 & -120,32 & 106,73 \end{bmatrix} \times 10^{-3}$

Taula 6.4: Matrius $\mathbf{K}_1$ i $\mathbf{K}_2$ de la llei de control obtinguda pel model mecànic (en unitats SI).

con d'obertura $3°$ respecte de la vertical (Fig. 6.5). A continuació es mostren les gràfiques que resulten d'una simulació de 5 s de duració en la qual es parteix de $\beta_1 = 3°$ i $\beta_2 = 0°$. A la Fig. 6.6 **(a)** veiem l'evolució dels angles β_1 i β_2. Com es pot apreciar, β_1 comença prenent un valor de $3°$ i la llei de control s'encarrega de corregir aquest error portant al pèndol cap a la seva posició d'equilibri vertical, deixant β_1 a $0°$. Un detall interessant és que, com és d'esperar, el pèndol és manté en tot moment en el pla yz, ja que β_2 és zero en tot moment. A la Fig. 6.6 **(b)** podem observar l'orientació de la plataforma al llarg del temps. Veiem com la llei de control, per tal de retornar el pèndol a la seva configuració vertical de forma efectiva, fa variar notablement l'orientació de la plataforma, variant l'angle ψ fins a $-40°$. A la Fig. 6.6 **(c)** s'observa a més com ha variat la posició del centre de gravetat de la plataforma al llarg del temps. Com és lògic, la plataforma fa el moviment corrector adequat per tal d'equilibrar el pèndol, i finalment retorna a la posició original. A la Fig. 6.6 **(d)** es pot apreciar com han variat els angles dels motors durant aquesta simulació. Anàlogament al que ens trobàvem a la Secció 6.2, els angles apareixen aparellats de dos en dos degut a la construcció simètrica del robot. Finalment, a la Fig. 6.6 **(f)** podem veure com evolucionen els parells τ_i generats pels motors. Com ja hem anat veient, i per qüestions de simetria, els parells surten emparellats de dos en dos. A més, són tots positius al llarg del temps, la qual cosa indica que els cables no perdran tensió en cap moment.

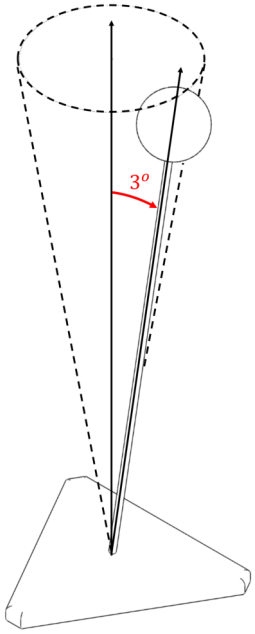


Figura 6.5: Con de $3°$ dins el qual podem estabilitzar el pèndol amb la llei de control obtinguda pel model mecànic.

Robustesa davant de pertorbacions de força no modelitzades

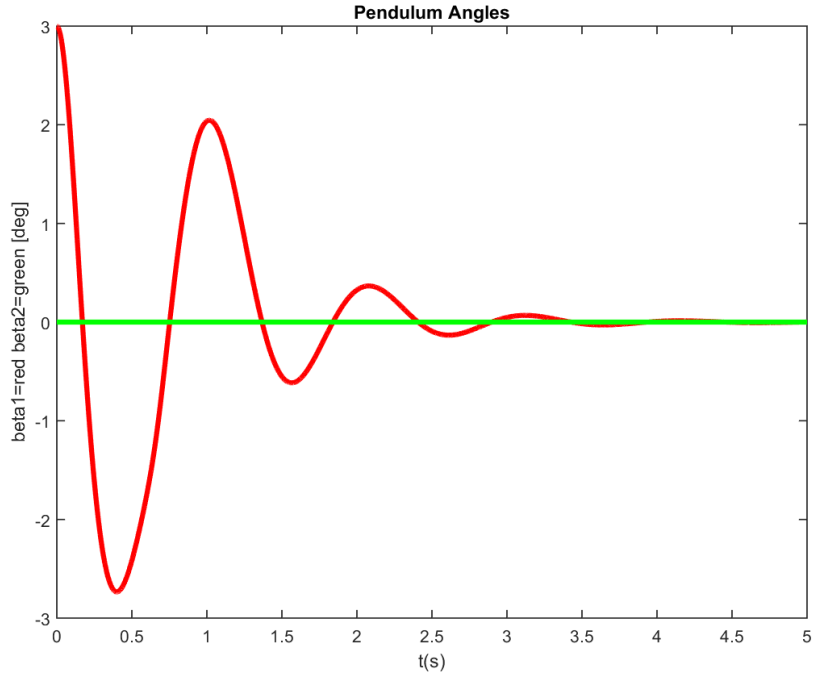
A continuació es vol posar a prova la robustesa del sistema realimentat davant de pertorbacions de força externes no incloses en el model. Per fer-ho simularem l'efecte d'aplicar petites forces externes sobre el centre de masses G_{pole} del pèndol, en diversos instants de temps del període de simulació.

Per simular l'efecte d'aquestes forces caldrà sumar la força generalitzada de pertorbació al membre dret de l'Eq. (2.35). Aquesta força, que denotarem per $\mathcal{F}_p$, s'obté de manera anàloga a com s'ha calculat la força generalitzada d'actuació $\mathcal{F}_a$ (Secció 2.4.3). Si $\boldsymbol{f}_p$ és la força de pertorbació considerada, la potència virtual d'aquesta força és

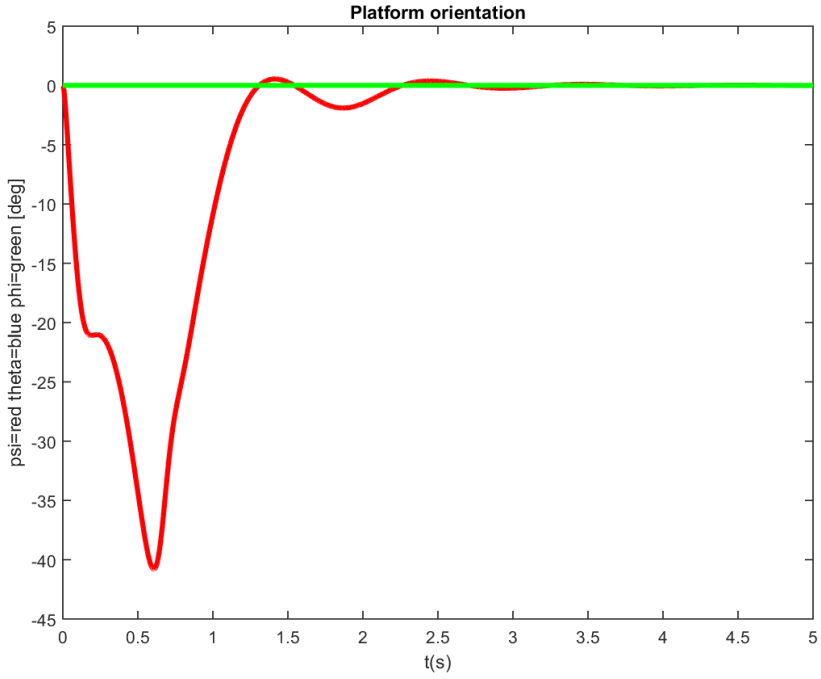
$$P_p = \boldsymbol{f}_p^T \cdot \boldsymbol{v}_{G_{pole}} \tag{6.10}$$

i l'objectiu serà escriure-la en la forma

$$P_p = \mathcal{F}_p^T \cdot \dot{\boldsymbol{q}}. \tag{6.11}$$

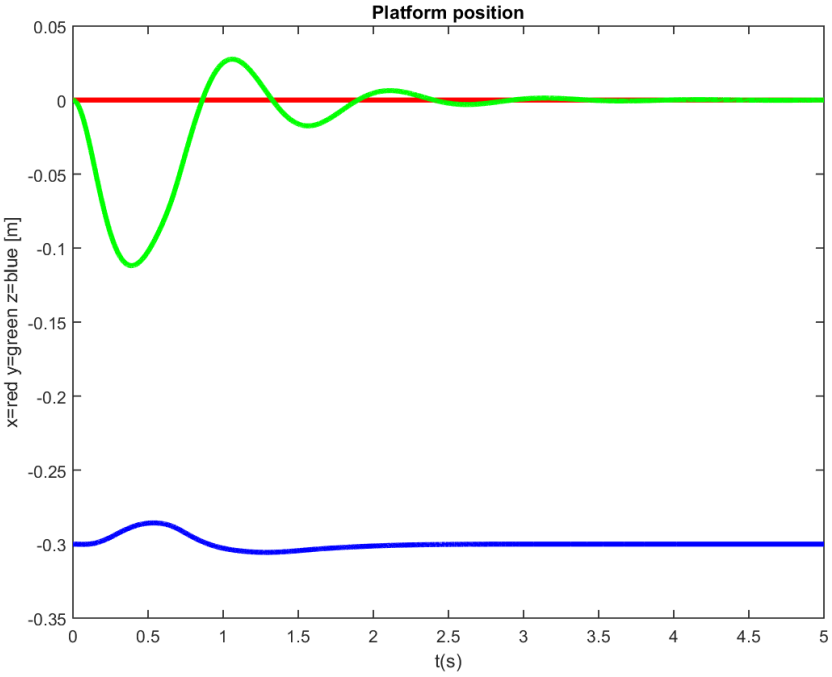


(a) Evolució dels angles β_1 i β_2.

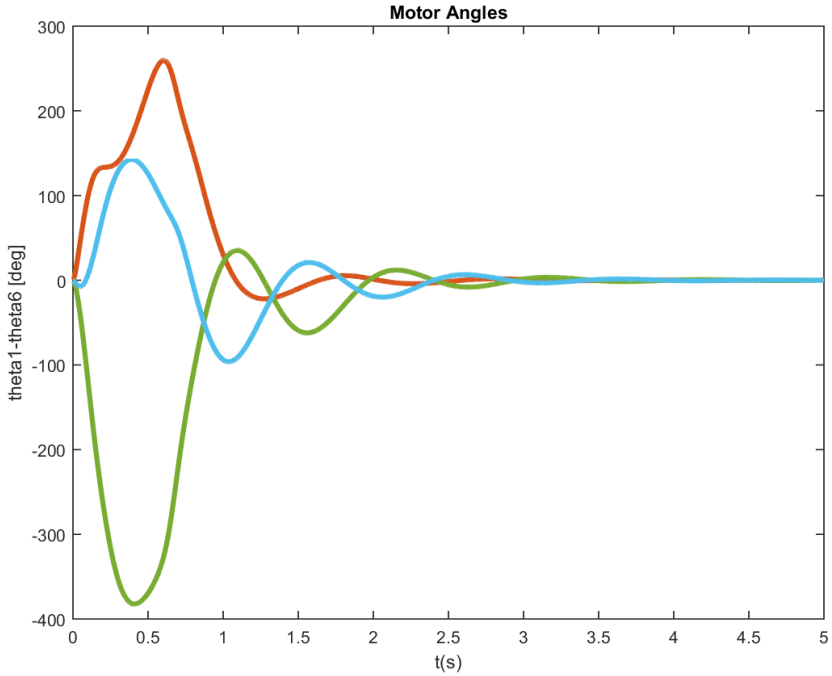


(b) Evolució dels angles d'Euler de la plataforma ψ, θ, φ.

Figura 6.6: Estabilització del sistema mecànic partint de $\beta_1 = 3°$ i $\beta_2 = 0°$ (1 de 3).

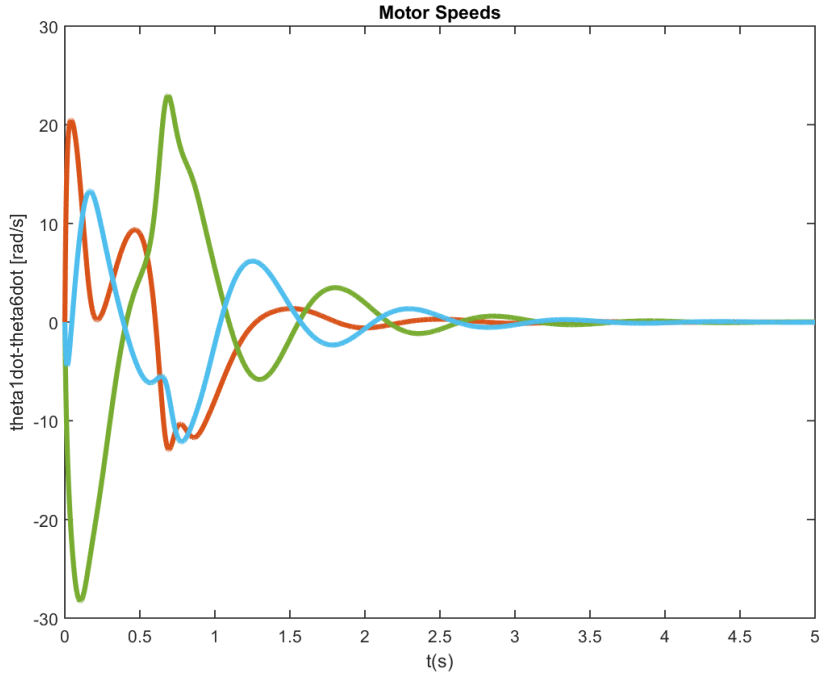


(c) Evolució de la posició del centre de gravetat de la plataforma G_{plat}.

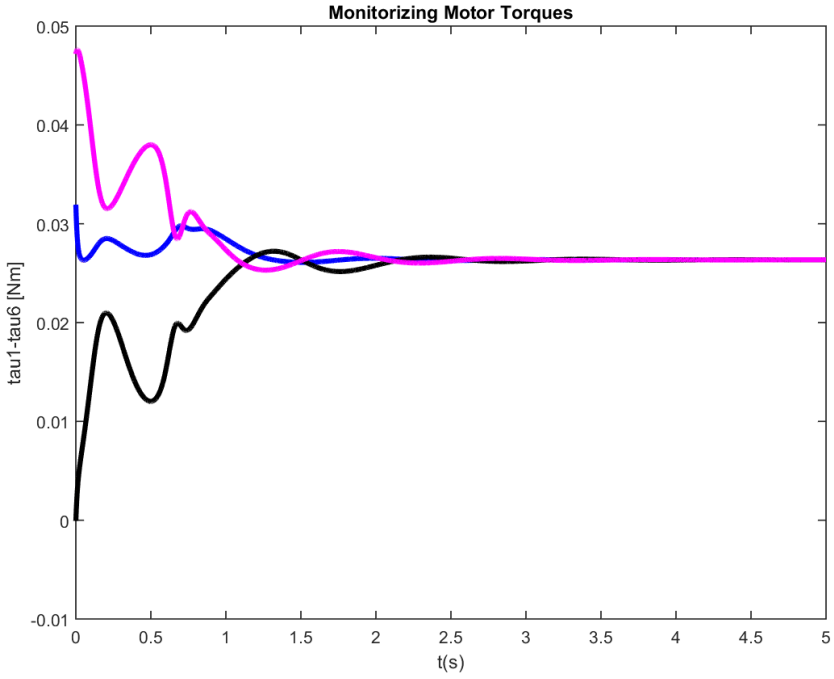


(d) Evolució dels angles dels motors $\theta_1, \ldots, \theta_6$.

Figura 6.6: Estabilització del sistema mecànic partint de $\beta_1 = 3°$ i $\beta_2 = 0°$ (2 de 3)



(e) Evolució de la velocitat angular dels motors $\dot{\gamma}_1, \ldots, \dot{\gamma}_6$.



(f) Evolució dels parells generats pels motors $\tau_1, \ldots, \tau_6$.

Figura 6.6: Estabilització del sistema mecànic partint de $\beta_1 = 3°$ i $\beta_2 = 0°$ (3 de 3)

A tal efecte, recordem que a la Secció 2.4 hem deduït que

$$\boldsymbol{v}_{G_{pole}} = \mathbf{D}_{pole} \cdot \dot{\boldsymbol{q}}, \tag{6.12}$$

Per tant, substituint l'Eq. (6.12) a la (6.10) obtenim

$$P_p = \boldsymbol{f}_p^T \cdot \mathbf{D}_{pole} \cdot \dot{\boldsymbol{q}} \tag{6.13}$$

i comparant l'Eq. (6.13) amb la (6.11) veiem que

$$\mathcal{F}_p = \mathbf{D}_{pole}^T \cdot \boldsymbol{f}_p \tag{6.14}$$

Per il·lustrar l'efecte d'aquestes forces, a continuació es mostraran els resultats d'aplicar-les durant una simulació de 15 s (Fig. 6.7). Durant aquesta simulació. La plataforma es troba inicialment en la configuració d'equilibri, però el pèndol està desviat amb $\beta_1 = 3°$ i $\beta_2 = 0$. La llei de control corregeix aquesta desviació amb poc menys de 5 s, retornant el pèndol a la configuració $\beta_1 = \beta_2 = 0°$. Poc després s'apliquen tres forces de pertorbació sobre G_{pole}:

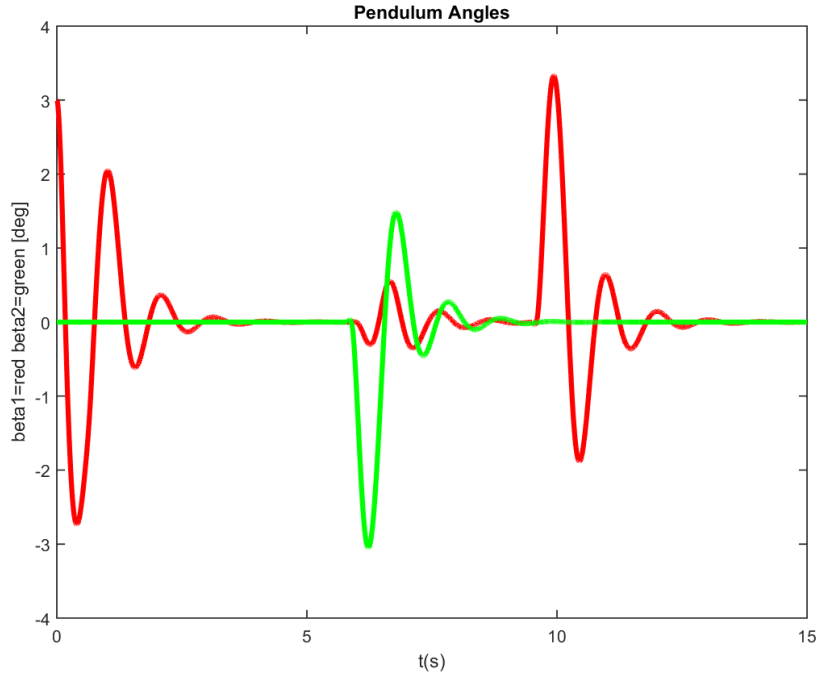
- Una força de 0.5 N en la direcció x, a l'instant $t = 5.8$ s.
- Una força també de 0.5 N però en la direcció y, a l'instant $t = 9.5$ s.
- Una força de 2 N en la direcció z, a l'instant $t = 13$ s.

Les tres forces s'apliquen de manera individual durant 100 ms a partir dels instants indicats.

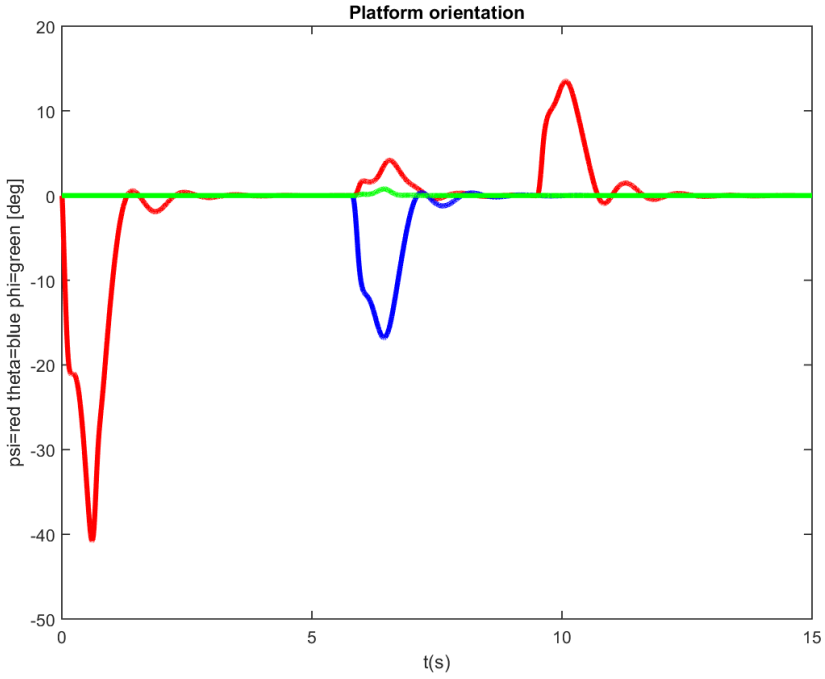
Dels gràfics de la Fig. 6.7 veiem com l'Hexapole corregeix totes les desviacions, tant la deguda a les condicions inicials com les provocades per les tres forces de pertorbació. Fixem-nos que mentre que les forces de pertorbació en les direccions x i y queden reflectides en tots els gràfics, la de pertorbació en z no fa variar els angles del pèndol, ni l'orientació de la plataforma, però sí la posició z de la plataforma i els parells aplicats pels motors. Això es deu a que aquesta força s'aplica quan el pèndol es troba en posició vertical, i per tant no pot provocar variacions en els angles del pèndol, ni en l'orientació de la plataforma, ni en les coordenades x i y. També és important adonar-se que la llei de control s'ha ajustat per tal de respectar els rangs esmentats a la Secció 6.3.2. En especial, fixem-nos que a la Fig. 6.7 **(d)** tots els parells motors es troben dins dels rangs permesos, i que a més són tots positius, de manera que tots els cables treballaran en tensió tal i com desitjàvem.

6.3.3 Linealització i control del model electromecànic

Un cop obtinguts uns resultats prou satisfactoris sobre el model mecànic, podem procedir a obtenir una llei de control pel model electromecànic complet. En fer-ho s'ha de tenir en compte

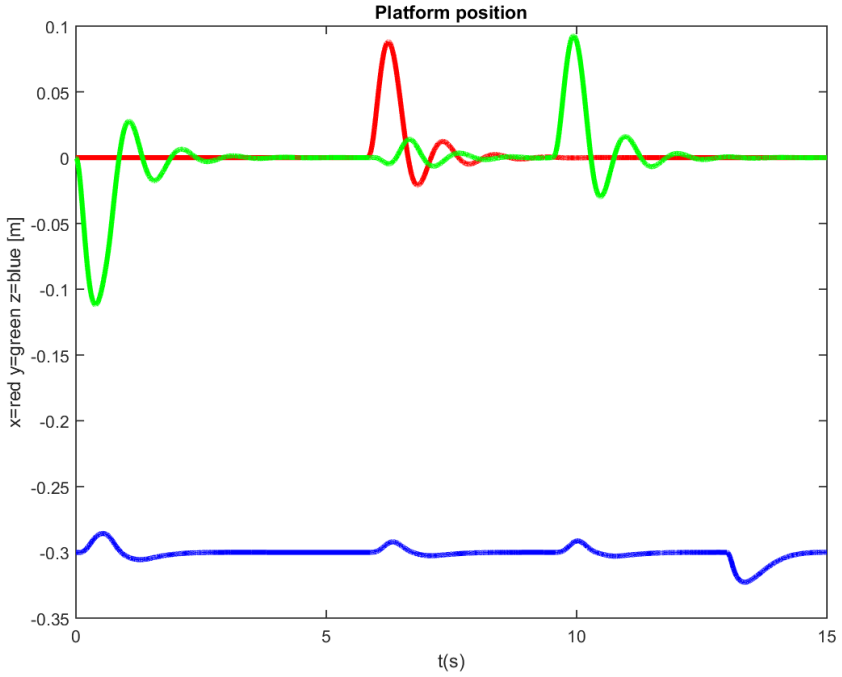


(a) Evolució dels angles β_1 i β_2.

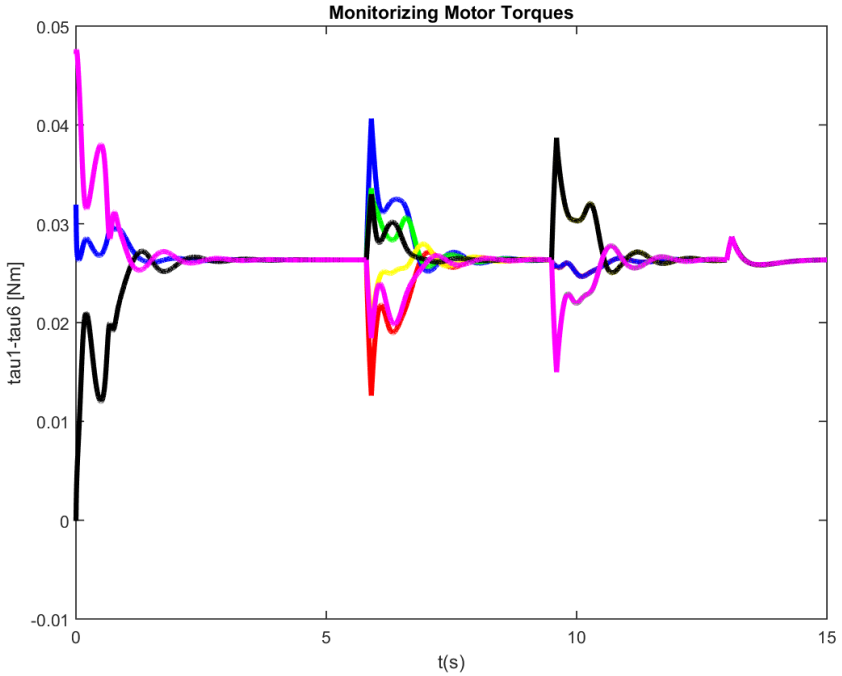


(b) Evolució dels angles d'Euler de la plataforma ψ, θ, φ.

Figura 6.7: Robustesa del sistema mecànic davant de pertorbacions no modelitzades (1 de 2).



(c) Evolució de la posició del centre de gravetat de la plataforma G_{plat}.



(d) Evolució dels parells $\tau_1, \ldots, \tau_6$ generats pels motors.

Figura 6.7: Robustesa del sistema mecànic davant de pertorbacions no modelitzades (2 de 2).

Paràmetre	Símbol	Valor
Longitud del pèndol	l_{pole}	0.7220 m
Massa de la plataforma	m_{plat}	0.1277 kg
Massa del pèndol	m_{pole}	0.7570 kg
Tensor d'inèrcia plataforma	$\mathbf{I}_{plat}$	$\begin{bmatrix} 152915.98 & 0 & 0 \\ 0 & 152915.98 & 0 \\ 0 & 0 & 291835.32 \end{bmatrix} \times 10^{-9}$ kgm^2
Tensor d'inèrcia pèndol	$\mathbf{I}_{pole}$	$\begin{bmatrix} 6184121.00 & 0 & 0 \\ 0 & 6184121.00 & 0 \\ 0 & 0 & 210893.40 \end{bmatrix} \times 10^{-9}$ kgm^2

Taula 6.5: Valors utilitzats en el model electromecànic realimentat de l'Hexacrane.

que la feina feta per ajustar la llei de control en la secció anterior no es pot aprofitar i cal començar de zero. Si bé és cert que també utilitzarem l'heurístic de Bryson com a primera aproximació, no podem perdre de vista que ara l'acció sobre el sistema es fa injectant voltatges en els motors, i que per tant els valors en la diagonal de la matriu $\mathbf{R}$ són completament diferents (Capítol 5). A més, degut a les baixes prestacions dels motors que utilitzem ens hem vist obligats a variar els valors dels paràmetres dinàmics amb els quals s'han realitzat les simulacions fins aquest moment. Així, s'ha reduït lleugerament la massa de la plataforma per poder accelerar-la més fàcilment. També s'ha augmentat la massa del pèndol. Per compensar la disminució de massa de la plataforma i poder mantenir la tensió dels cables, i s'ha escurçat la longitud del pèndol per reduir la seva inèrcia i facilitar la tasca de canviar-ne l'orientació. Aplicant aquests canvis s'intenta compensar el fet que aquest sistema té un comportament molt més feixuc que l'anterior a causa de la dinàmica dels motors i dels fregaments associats. A la Taula 6.5 es poden veure els nous paràmetres utilitzats. Amb aquests paràmetres, el model linealitzat obtingut ve donat pels valors de la Taula 6.6, que generen, novament, un sistema controlable. Tenint en compte els valors màxims indicats a la Secció 6.3.2, el fet que els motors accepten voltatges de fins a 14.8 V, i fixant $\alpha_1 = 1$, $\alpha_2 = 1$ i $\alpha_3 = 1100$, s'obté la llei de control

$$\boldsymbol{u}_v = \boldsymbol{u}_{v_o} - \mathbf{K}(\boldsymbol{x}_\theta - \boldsymbol{x}_{\theta,o}), \tag{6.15}$$

on $\mathbf{K} = [\mathbf{K}_1, \mathbf{K}_2]$, essent $\mathbf{K}_1$ i $\mathbf{K}_2$ les matrius indicades a la Taula 6.7 (en unitats SI).

Bloc	Valor
$\mathbf{H}^{-1}\frac{\partial}{\partial \boldsymbol{\theta}}(\mathbf{E}\boldsymbol{u}-\mathbf{G})$	$\begin{bmatrix} -0,72 & 0,39 & 0,10 & -0,10 & 0,10 & 0,05 & -0,002 & 18,43 \\ 0,39 & -0,72 & 0,05 & 0,10 & -0,10 & 0,10 & -0,002 & -18,43 \\ 0,10 & 0,05 & -0,72 & 0,39 & 0,10 & -0,10 & -15,96 & -9,21 \\ -0,10 & 0,10 & 0,39 & -0,72 & 0,05 & 0,10 & 15,96 & 9,21 \\ 0,10 & -0,10 & 0,10 & 0,05 & -0,72 & 0,39 & 15,96 & -9,21 \\ 0,05 & 0,10 & -0,10 & 0,10 & 0,39 & -0,72 & -15,96 & 9,21 \\ -0,001 & -0,001 & 0,01 & -0,009 & -0,009 & 0,01 & 13,88 & -4,05\times10^{-17} \\ -0,01 & 0,01 & 0,004 & -0,007 & 0,007 & -0,004 & -7,80\times^{-17} & 13,88 \end{bmatrix}$
$\mathbf{H}^{-1}\mathbf{C}$	$\begin{bmatrix} -159,13 & -0,01 & 0,17 & 0,52 & 0,17 & 0,50 & 0 & 0 \\ -0,01 & -159,13 & 0,50 & 0,17 & 0,52 & 0,17 & 0 & 0 \\ 0,17 & 0,50 & -159,12 & -0,01 & 0,17 & 0,52 & 0 & 0 \\ 0,52 & 0,17 & -0,01 & -159,12 & 0,50 & 0,17 & 0 & 0 \\ 0,17 & 0,52 & 0,17 & 0,50 & -159,12 & -0,01 & 0 & 0 \\ 0,50 & 0,17 & 0,52 & 0,17 & -0,01 & -159,12 & 0 & 0 \\ 0,0001 & 0,0001 & 1,28 & -1,28 & -1,28 & 1,28 & 0 & 0 \\ -1,48 & 1,48 & 0,74 & -0,74 & 0,74 & -0,74 & 0 & 0 \end{bmatrix}$
$\mathbf{H}^{-1}\mathbf{E}$	$\begin{bmatrix} 76,56 & 0,008 & -0,08 & -0,25 & -0,08 & -0,24 \\ 0,008 & 76,56 & -0,24 & -0,08 & -0,25 & -0,08 \\ -0,08 & -0,24 & 76,56 & 0,008 & -0,08 & -0,25 \\ -0,25 & -0,08 & 0,008 & 76,56 & -0,24 & -0,08 \\ -0,08 & -0,25 & -0,08 & -0,24 & 76,56 & 0,008 \\ -0,24 & -0,08 & -0,25 & -0,08 & 0,008 & 76,56 \\ -8,72\times^{-5} & -8,72\times^{-05} & -0,61 & 0,61 & 0,61 & -0,61 \\ 0,71 & -0,71 & -0,35 & 0,35 & -0,35 & 0,35 \end{bmatrix}$

Taula 6.6: Blocs rellevants d'$\mathbf{A}$ i $\mathbf{B}$ en el model electromecànic linealitzat (en unitats SI).

Robustesa a condicions inicials diverses

Després de dedicar-hi prou temps, s'han aconseguit reproduir uns resultats gairebé tan bons com els obtinguts en el primer model. En aquest cas, però, el frec viscós dels motors i la seva resposta temporal feixuga fan que el sistema acabi actuant de forma molt més lenta i requerint molts més recursos (voltatges de consigna alts) per efectuar moviments similars. És per això que, en aquest moment, el millor ajust trobat per a la llei de control ens permet iniciar el pèndol dins d'un con de $\sqrt{2}^{\circ} \approx 1,41^{\circ}$ d'obertura en comptes de 3° com s'ha obtingut en el primer model.

Bloc	Valor
$\mathbf{K}_1$	$\begin{bmatrix} 0,29 & 0,61 & 0,30 & -0,30 & 0,30 & -0,30 & -0,0003 & 619,64 \\ 0,61 & 0,29 & -0,30 & 0,30 & -0,30 & 0,30 & -0,0003 & -619,64 \\ 0,30 & -0,30 & 0,29 & 0,61 & 0,30 & -0,30 & -536,62 & -309,82 \\ -0,30 & 0,30 & 0,61 & 0,29 & -0,30 & 0,30 & 536,63 & 309,82 \\ 0,30 & -0,30 & 0,30 & -0,30 & 0,29 & 0,61 & 536,63 & -309,82 \\ -0,30 & 0,30 & -0,30 & 0,30 & 0,61 & 0,29 & -536,62 & 309,82 \end{bmatrix} \times 10^{-3}$
$\mathbf{K}_2$	$\begin{bmatrix} -1,54 & 1,55 & 0,77 & -0,77 & 0,77 & -0,77 & -7,28 \times 10^{-5} & 169,44 \\ 1,55 & -1,54 & -0,77 & 0,77 & -0,77 & 0,77 & -7,28 \times 10^{-5} & -169,44 \\ 0,77 & -0,77 & -1,54 & 1,55 & 0,77 & -0,77 & -146,74 & -84,72 \\ -0,77 & 0,77 & 1,55 & -1,54 & -0,77 & 0,77 & 146,74 & 84,72 \\ 0,77 & -0,77 & 0,77 & -0,77 & -1,54 & 1,55 & 146,74 & -84,72 \\ -0,77 & 0,77 & -0,77 & 0,77 & 1,55 & -1,54 & -146,74 & 84,72 \end{bmatrix} \times 10^{-3}$

Taula 6.7: Matrius $\mathbf{K}_1$ i $\mathbf{K}_2$ obtingudes amb el model electromecànic (en unitats SI).

Dins d'aquest con es garanteix que el pèndol retornarà a la seva posició original sense sortir de l'espai de treball i respectant les capacitats dels motors.

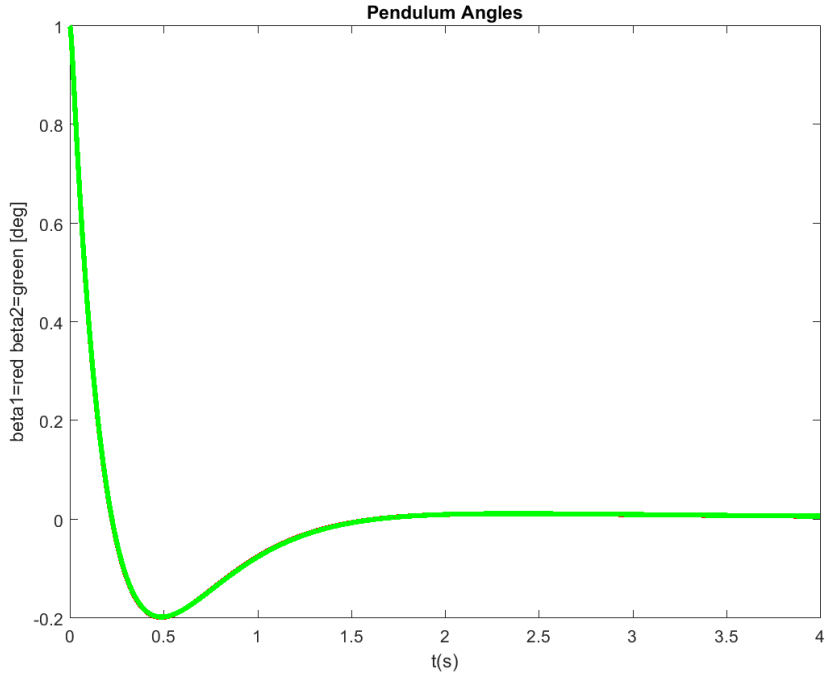
A tall d'exemple, la Fig. 6.8 mostra com s'estabilitza l'Hexapole quan iniciem el pèndol amb $\beta_1 = \beta_2 = 1°$. Degut a la simetria del robot i de les condicions inicials esmentades, β_1 i β_2 evolucionen de la mateixa manera amb el temps, i és per això que en a la Fig. 6.8 **(a)** les corbes d'aquests dos angles coincideixen. A la Fig. 6.8 **(b)** veiem com l'orientació de la plataforma varia una mica per corregir l'orientació del pèndol amb més facilitat. A diferència del que passava en el model mecànic, però, aquesta variació és ara petita. Això es deu a que el fregament dels motors, i les baixes prestacions que aquests tenen, fan que la resposta dinàmica del sistema sigui molt més lenta que abans, resultant molt més difícil provocar grans moviments de la plataforma. Veiem també que l'angle φ varia molt poc, en consistència amb el fet que una rotació en z de la plataforma és innecessària per a l'estabilització del pèndol. A la Fig. 6.8 **(c)** s'aprecia com el centre de masses de la plataforma es mou en diagonal, descrivint un moviment simètric en les coordenades x i y. Aquest moviment és coherent. Amb les condicions inicials triades el pèndol no es troba ni en el pla xz, ni a l'yz, sinó en un pla a $45°$ respecte d'aquests dos últims, i per tant és lògic que G_{plat} es mogui en aquest darrer pla. La gràfica de la Fig. 6.8 **(d)**, per la seva banda, mostra les consignes de voltatge que cal donar als motors. Com es veu, els voltatges no

excedeixen els $\pm 14.8\ V$ admisibles pels motors, indicant que la llei de control ha quedat ben ajustada. Les Figs. 6.8 **(e)** i 6.8 **(f)**, finalment, mostren les variacions de les velocitats angulars dels motors i les forces fetes pels cables sobre la plataforma. Com desitjàvem, les velocitats angulars es mantenen en el rangs admisibles i les tensions dels cables són sempre positives. Com que el vector d'accions ara està format pels voltatges d'excitació en lloc dels parells motors, l'evolució d'aquests darrers ja no es mostra.

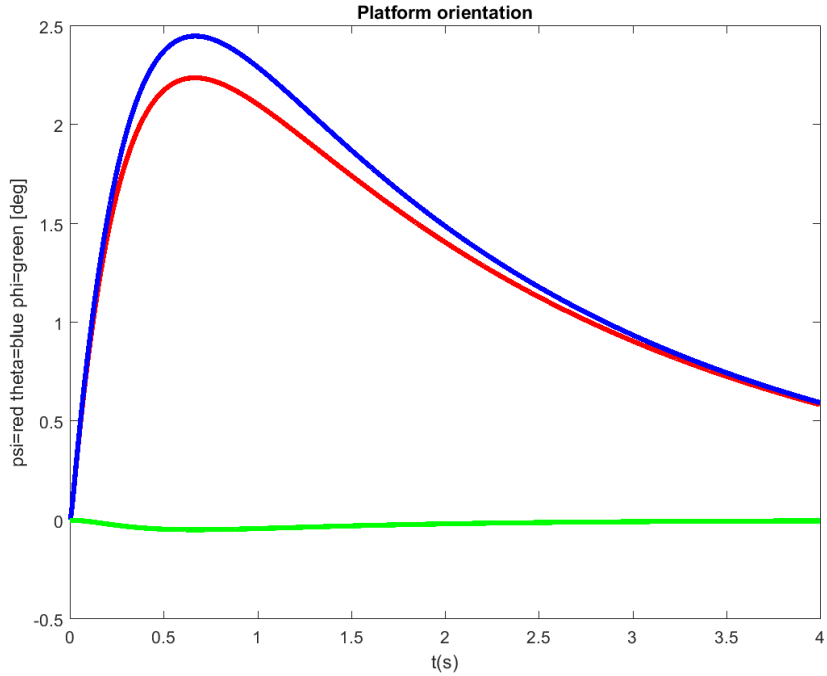
Tot i que la llei de control dissenyada amb aquest model és capaç d'estabilitzar l'Hexapole, veiem que els voltatges de consigna arriben fàcilment als màxims permesos pels actuadors Dynamixel MX-28. Això ja és així en situacions favorables com la descrita, en la qual la desviació del pèndol respecte de la posició d'equilibri és de poc més d'un grau. A efectes de controlar l'Hexapole real, per tant, el més aconsellable és canviar els seus motors per uns altres de majors prestacions i refer els càlculs de la llei de control d'acord amb aquestes prestacions. Aquests motors haurien de ser més ràpids, de major parell, i amb una menor relació de reducció, reduint així l'impacte del fregament sec i viscós en la resposta dinàmica del sistema.

Robustesa davant de pertorbacions no modelitzades

Com en el cas del model mecànic, la llei de control obtinguda pel model electromecànic és també robusta a petites pertorbacions de força no modelitzades. La Fig. 6.9 mostra els resultats de la mateixa prova feta abans pel model mecànic, en la qual s'apliquen forces de pertorbació en els instants 5.8s, 9.5s, i 13s, de 0.5N, 0.5N, i 2N respectivament, en les direccions x, y i z durant $100\ \text{ms}$. En aquest cas, però, les condicions inicials del pèndol són $\beta_1 = \beta_2 = 1°$. Com es veu, tot i ser de resposta més lenta, el sistema és capaç d'equilibrar la desviació inicial i eliminar les tres pertorbacions. De manera semblant al que passava en el sistema mecànic, la pertorbació de força en direcció z només queda reflectida a les gràfiques de tensions en els cables. Això es deu a que en el moment d'aplicar la pertorbació el pèndol es troba en posició vertical, i per tant els angles β_1 i β_2 no queden afectats. La plataforma tampoc varia gairebé la seva posició, perquè el fregament viscós dels motors, amplificat pel reductor, contraresta ràpidament la força de pertorbació. En conseqüència, el sistema es desvia molt poc en relació a la trajectòria que venia tenint, i no calen voltatges correctors més enllà dels que es venien aplicant.

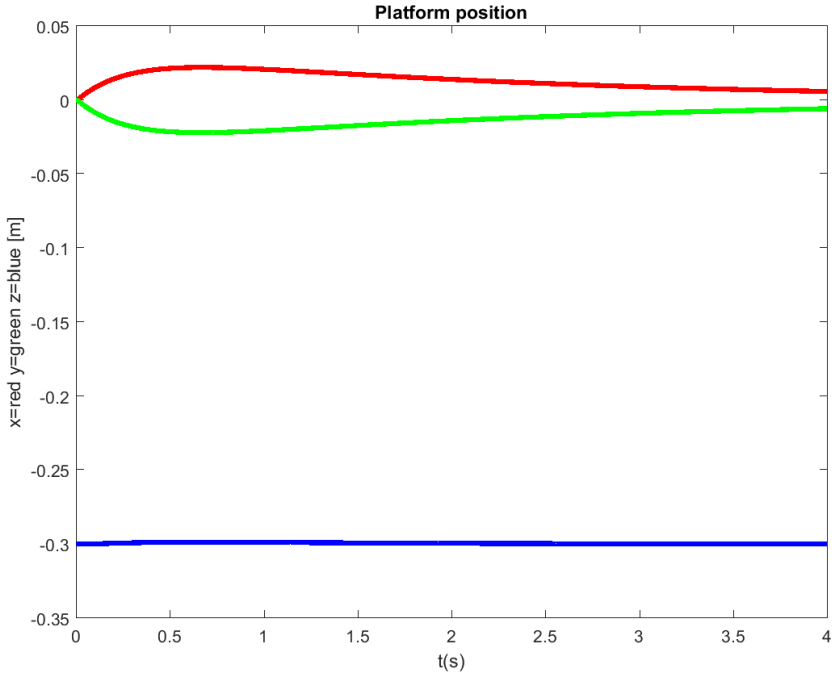


(a) Evolució dels angles β_1 i β_2.

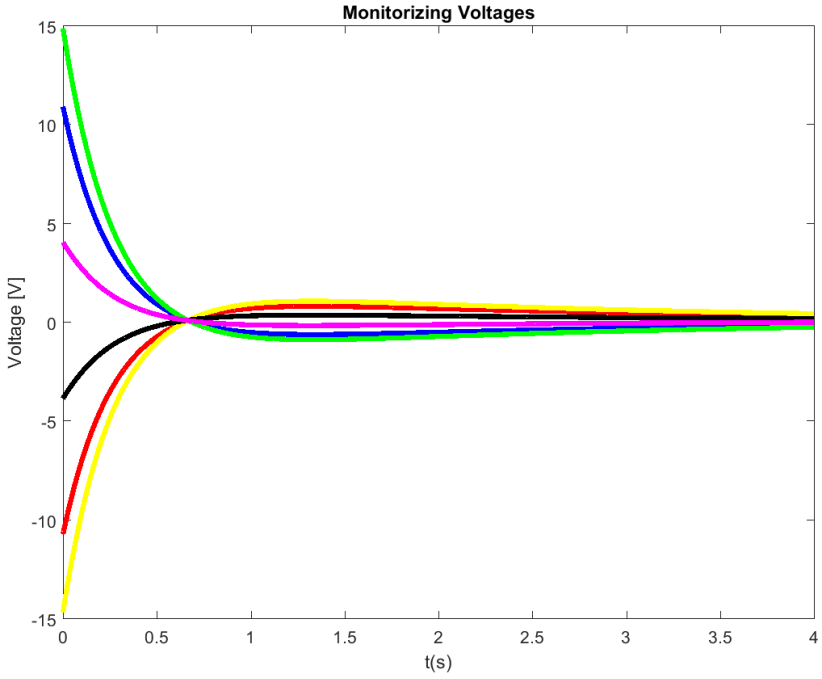


(b) Evolució dels angles d'Euler de la plataforma ψ, θ, φ.

Figura 6.8: Estabilització del sistema electromecànic partint de $\beta_1 = \beta_2 = 1^\circ$ (1 de 3).

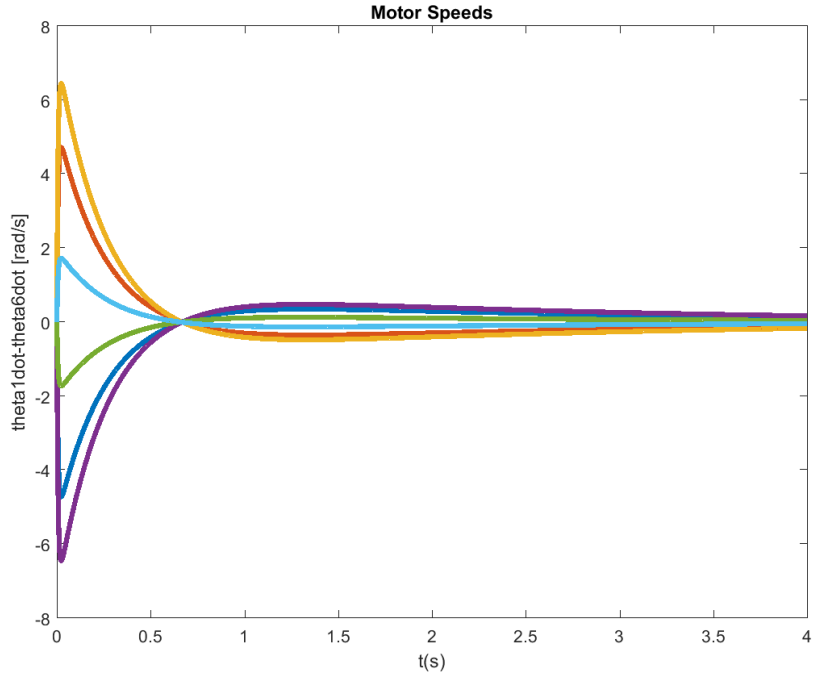


(c) Evolució de la posició del centre de gravetat de la plataforma G_{plat}.

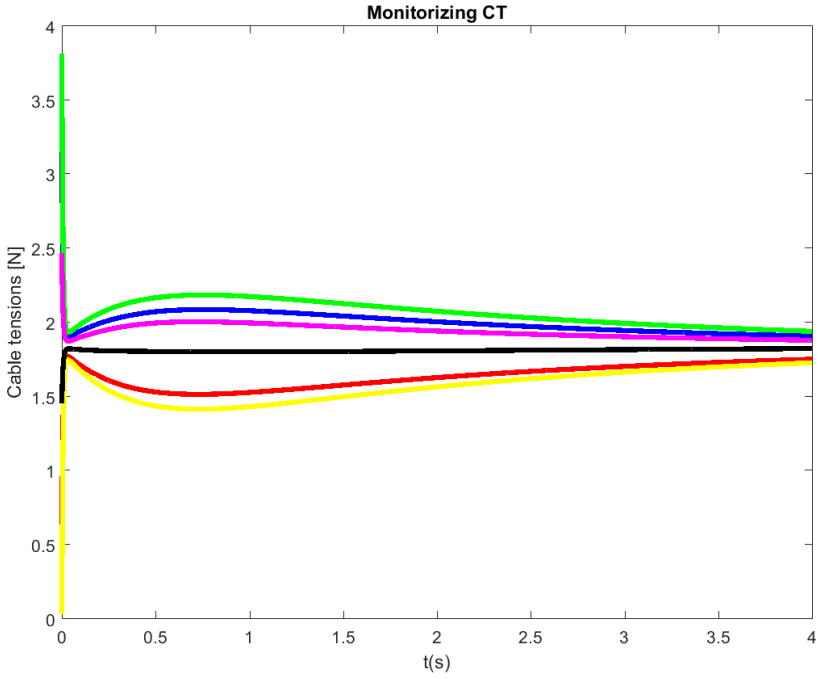


(d) Evolució de les consignes de voltatge dels motors $v_1, \ldots, v_6$.

Figura 6.8: Estabilització del sistema electromecànic partint de $\beta_1 = \beta_2 = 1°$ (2 de 3)

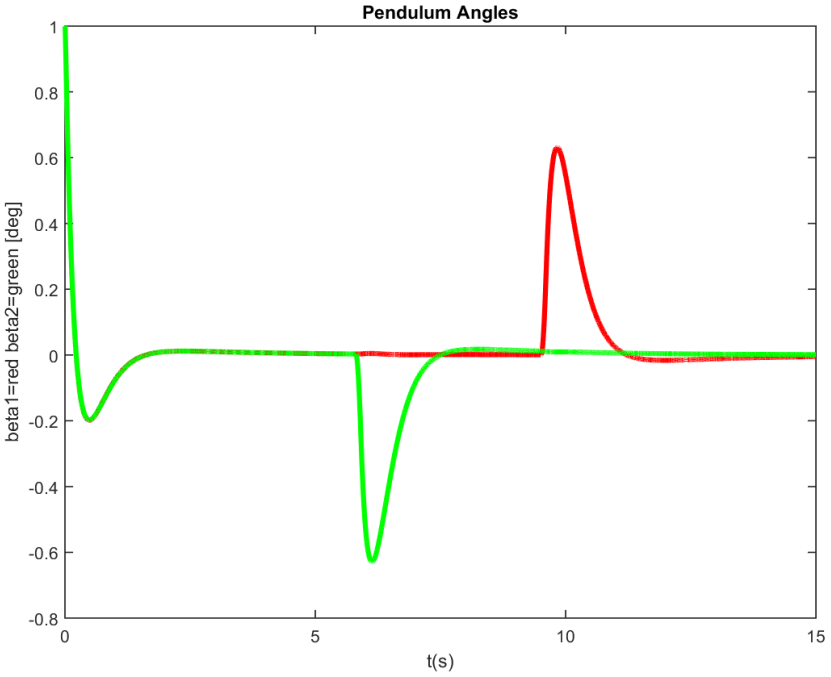


(e) Evolució de les velocitats angulars dels motors $\dot{\gamma}_1, \ldots, \dot{\gamma}_6$.

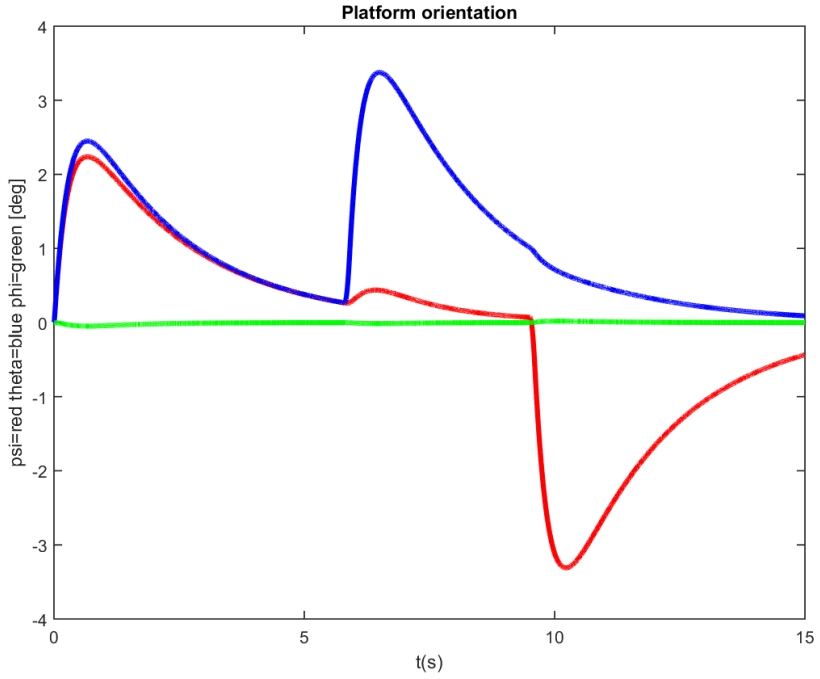


(f) Evolució de les forces transmeses pels cables a la plataforma.

Figura 6.8: Estabilització del sistema electromecànic partint de $\beta_1 = \beta_2 = 1°$ (3 de 3)

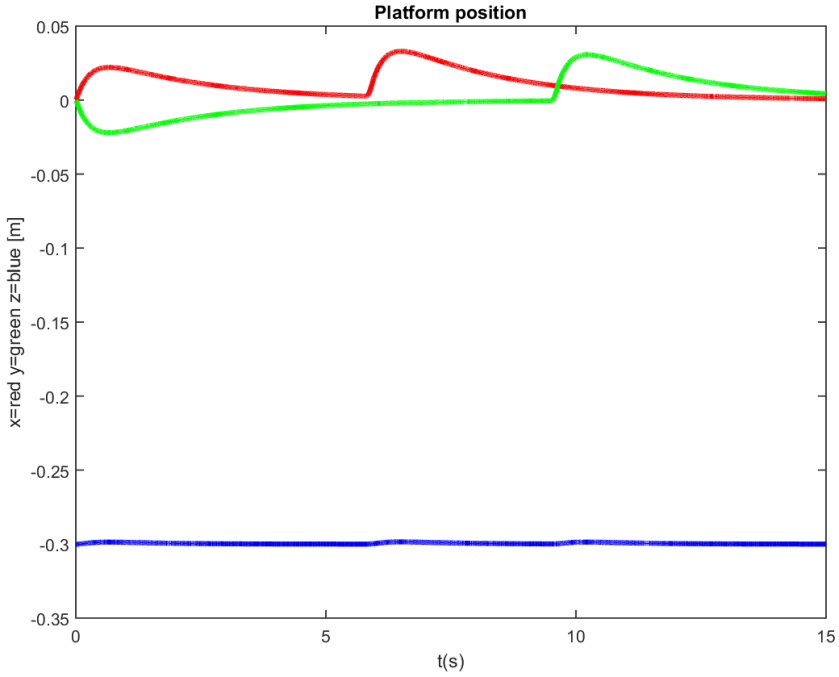


(a) Evolució dels angles β_1 i β_2.

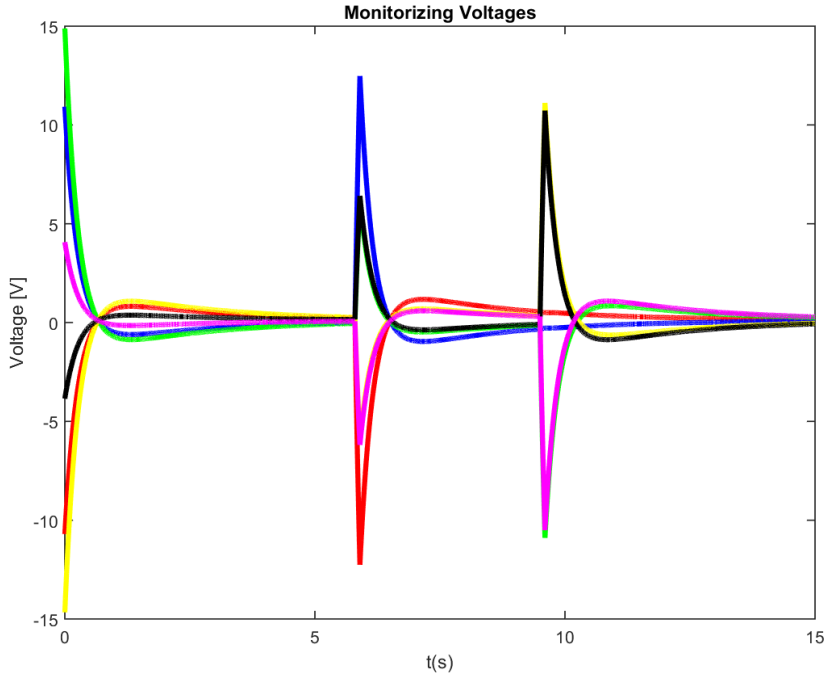


(b) Evolució dels angles d'Euler de la plataforma ψ, θ, φ.

Figura 6.9: Robustesa a pertorbacions no modelitzades en el model electromecànic (1 de 3).

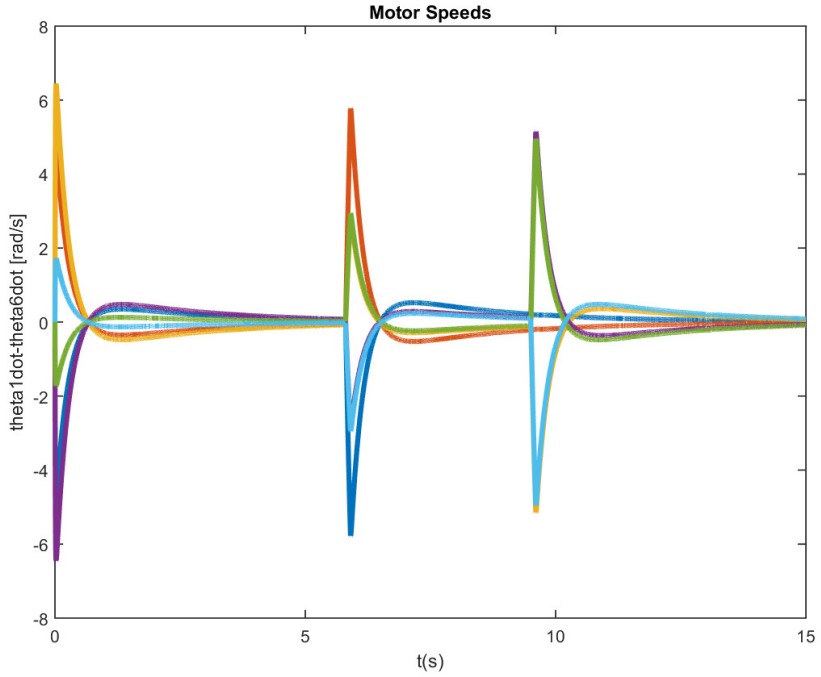


(c) Evolució de la posició del centre de gravetat de la plataforma G_{plat}.

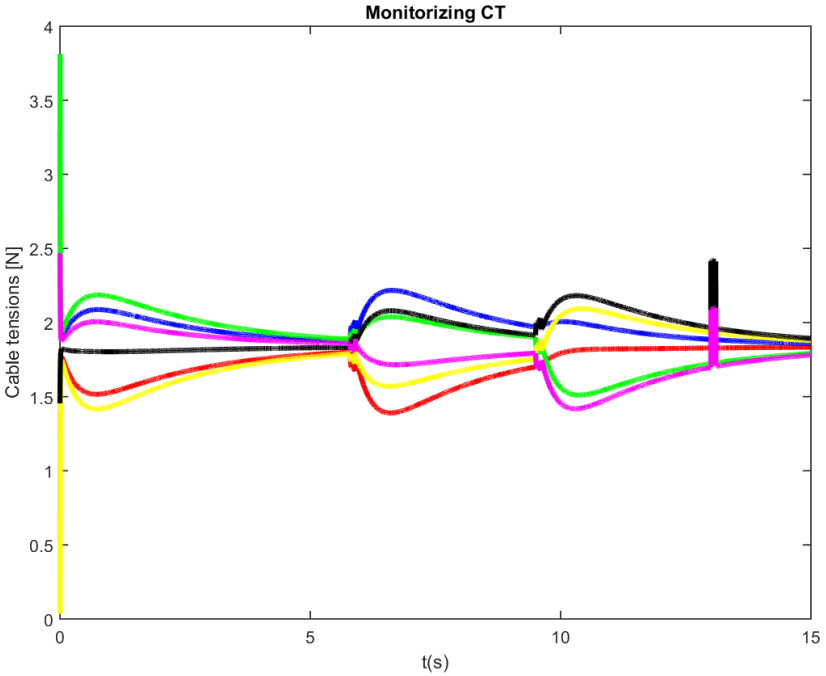


(d) Evolució de les consignes de voltatge dels motors $v_1, \ldots, v_6$.

Figura 6.9: Robustesa a pertorbacions no modelitzades en el model electromecànic (2 de 3)



(e) Evolució de la velocitat angular dels motors $\dot{\gamma}_1, \ldots, \dot{\gamma}_6$.



(f) Evolució temporal de les forces transmeses pels cables a la plataforma.

Figura 6.9: Robustesa a pertorbacions no modelitzades en el model electromecànic (3 de 3)

7

Conclusions

7.1 Contribucions

En aquest projecte s'han dissenyat dos sistemes de control per al sistema Hexapole. El primer sistema es basa en un model purament mecànic del robot, i no té en compte els fenòmens electrodinàmics a l'interior dels motors. La llei de control d'aquest sistema genera consignes de parell motor, i per tant s'assumeix implícitament que els motors són capaços d'acceptar i seguir aquestes consignes acuradament. Tot i aquesta assumpció, aquest sistema podria ser suficient en cas que el robot disposés de motors prou potents amb les prestacions esmentades. El segon sistema, en canvi, utilitza un model electromecànic que descriu millor l'Hexapole actual. Aquest model incorpora la dinàmica interna dels servomotors Dynamixel MX-28T, i permet avaluar si l'estabilització del pèndol és viable o no amb aquests actuadors. Les simulacions realitzades indiquen que l'estabilització és possible, però només per un rang molt petit de desviacions del pèndol respecte de la posició vertical. Això es deu a que els servomotors Dynamixel MX-28T són força lents, de parell baix, i reducció elevada, fent que el fregament del motor quedi massa amplificat a l'eix de sortida. Així doncs, la nostra recomanació és canviar aquests motors per altres de majors prestacions, i redissenyar la llei de control amb la metodologia exposada. Aquest redisseny hauria de ser relativament fàcil ja que es poden utilitzar tots els programes MAPLE i MATLAB desenvolupats, canviant únicament els valors dels paràmetres dinàmics emprats. A més, els paràmetres dels nous motors es podrien obtenir amb la metodologia descrita a [10], que tot i estar particularitzada pels servomotors Dynamixel MX-28T, és força genèrica.

7.2 Treball futur

Per implementar els sistemes de control descrits sobre l'Hexapole real caldria dur a terme les tasques següents:

1. Reequipar l'Hexacrane amb motors de majors prestacions, tal i com hem proposat a la secció anterior.

2. Actualitzar els model electromecànic tenint en compte els paràmetres d'aquests nous motors, i redissenyar una llei de control estabilitzadora amb la metodologia que hem exposat.

3. Afegir un sensor d'orientació en el pèndol. A l'IRI ja s'està desenvolupant un sistema per aquesta finalitat, basat en sensors d'efecte Hall col·locats a la plataforma on es recolza el pèndol. Aquests sensors s'han d'acabar de calibrar i ajustar, i s'han de polir els programes de comunicació per llegir els seus valors.

4. Implementar la llei de control estabilitzadora en llenguatge C.

5. Implementar, també en llenguatge C, programes de comunicació eficients entre un PC i els motors i sensors de l'Hexapole.

6. Compilar els anteriors programes i executar-los en un sistema operatiu de temps real per tal que el llaç de control no es vegi afectat per interrupcions provocades per altres tasques. Això permetria assegurar una certa freqüència en el llaç de control, que hauria de ser prou elevada si volem, com hem assumit, que el sistema es pugui assimilar a un de temps continu.

7. Validar el model electromecànic empíricament. Aquesta validació es podria fer acoplant el pèndol per la cara inferior de la plataforma i activant el sistema de control amb compensació gravitatòria, de manera semblant a com hem fet en simulació a la Secció 6.2. Els resultats de simulació haurien de concordar amb les oscil·lacions observades a la realitat.

8. Mesurar els retards que es produeixen des del moment en que es sol·liciten dades als sensors fins a la recepció d'aquestes dades. Mesurar també els retards de comunicació amb els motors. Els dos tipus de retards són clau, ja que podrien limitar la freqüència del llaç de control excessivament. Si fos així, el sistema no es podria assimilar a un de temps continu, i caldria redissenyar la llei de control mitjançant un regulador quadràtic lineal de temps discret.

9. Finalment, caldria validar el sistema de control experimentalment, veient la seva robustesa davant de pertorbacions de força aplicades sobre el pèndol, o davant de petites forces d'inèrcia provocades pel desplaçament de la base de l'Hexapole.

Aquestes tasques, com ja hem anticipat a la introducció, es deixen per a un treball futur.

Bibliografia

[1] M. Hehn and R. D'Andrea, "A flying inverted pendulum," in *Robotics and Automation (ICRA), 2011 IEEE international conference on*, pp. 763–770, IEEE, 2011.

[2] I. Soto and R. Campa, "Modelling and control of a spherical inverted pendulum on a five-bar mechanism," *International Journal of Advanced Robotic Systems*, vol. 12, no. 7, p. 95, 2015.

[3] S. Kouchaki, "Stabilization of an Inverted Pendulum with 2 Degrees of Freedom, using a Five Bar Linkage Mechanism," Master Thesis, University of California, San Diego, 2017.

[4] A. Kim and M. F. Golnaraghi, "A quaternion-based orientation estimation algorithm using an inertial measurement unit," in *PLANS 2004. Position Location and Navigation Symposium (IEEE Cat. No.04CH37556)*, pp. 268–272, April 2004.

[5] B. Sprenger, L. Kucera, and S. Mourad, "Balancing of an inverted pendulum with a SCARA robot," *IEEE/ASME transactions on mechatronics*, vol. 3, no. 2, pp. 91–97, 1998.

[6] T. Obenaus, A. Wilde, H. Priwitzer, J. Bretschneider, and O. Enge-Rosenblatt, "Design of multi-dimensional magnetic position sensor systems using the example of an inverse pendulum," in *Microelectronic Systems*, pp. 49–57, Springer, 2011.

[7] J. Agulló Batlle, *Mecànica de la Partícula i del Sòlid Rígid*. Publicacions OK Punt, 2002.

[8] R. Tedrake, "Underactuated Robotics. Course lecture notes. Available online through." http://underactuated.csail.mit.edu.2018.

[9] R. M.Murray, Z. Li, and S. Sastry., *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

[10] M. R. O. A. Maximo, C. H. C. Ribeiro, and R. J. M. Afonso, "Modeling of a position servo used in robotics applications," *XIII Simpósio Brasileiro de Automaçao Inteligente*, vol. 1, pp. 2032–2038, 2017.

[11] K. Ogata, *Modern Control Engineering*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 4th ed., 2001.

[12] M. Inc., *The Student Edition of Simulink : Dynamic System Simulation Software for Technical Education*. Prentice Hall, 1995.

[13] D. E. Kirk, *Optimal control theory: an introduction*. Dover Publications Inc., 2004.

[14] A. E. Bryson and Y.-C. HO, *Applied Optimal Control, Optimitzation, Estimation, and Control*. Taylor and Francis, 1975.

[15] J. Dormand and P. Prince, "A family of embedded Runge-Kutta formulae," *Appl. Math.*, vol. 6, pp. 19–26, 1980.

[16] L. F. Shampine and M. W. Reichelt, "The MATLAB ODE Suite," *Journal of Scientific Computing*, vol. 18, pp. 1–22, 1997.

[17] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C*. Cambridge University Press, 1986.

A

Relacions cinemàtiques de l'Hexapole

En aquest annex veurem diverses relacions cinemàtiques que són necessàries per a la formulació i linealització de les equacions del moviment de l'Hexapole, i per a la integració d'aquestes equacions en el temps. Durant tot l'annex tindrem en compte els sistemes de referència, les bases vectorials i tota la notació que s'ha introduït al capítol 2 (vegi's en especial les Seccions 2.2 i 2.3, i la Fig. 2.1).

A.1 Matriu de rotació de la plataforma

Construirem en primer lloc la matriu de rotació $\mathbf{R}_{plat}$ que ens dóna l'orientació de la base B_{plat} relativa a la base B_{abs} en funció dels angles d'Euler ψ, θ, φ de la plataforma. Si definim les següents rotacions axials

$$\mathbf{R}_x(\psi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\psi & -s_\psi \\ 0 & s_\psi & c_\psi \end{bmatrix},$$

$$\mathbf{R}_y(\theta) = \begin{bmatrix} c_\theta & 0 & s_\theta \\ 0 & 1 & 0 \\ -s_\theta & 0 & c_\theta \end{bmatrix},$$

i

$$\mathbf{R}_z(\varphi) = \begin{bmatrix} c_\varphi & -s_\varphi & 0 \\ s_\varphi & c_\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

on s_a i c_a denoten el sinus i el cosinus de l'angle a, respectivament, podem escriure

$$\mathbf{R}_{plat}(\psi, \theta, \varphi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\psi & -s_\psi \\ 0 & s_\psi & c_\psi \end{bmatrix} \cdot \begin{bmatrix} c_\theta & 0 & s_\theta \\ 0 & 1 & 0 \\ -s_\theta & 0 & c_\theta \end{bmatrix} \cdot \begin{bmatrix} c_\varphi & -s_\varphi & 0 \\ s_\varphi & c_\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{A.1}$$

i si operem

$$\mathbf{R}_{plat}(\psi, \theta, \varphi) = \begin{bmatrix} c_\theta c_\varphi & -c_\theta s_\varphi & s_\theta \\ c_\psi s_\varphi + s_\psi s_\theta c_\varphi & c_\psi c_\varphi - s_\psi s_\theta s_\varphi & -s_\psi c_\theta \\ s_\psi s_\varphi - c_{psi} s_\theta c_\varphi & s_{psi} c_\varphi + c_\psi s_\theta s_\varphi & c_\theta c_\psi \end{bmatrix}.$$

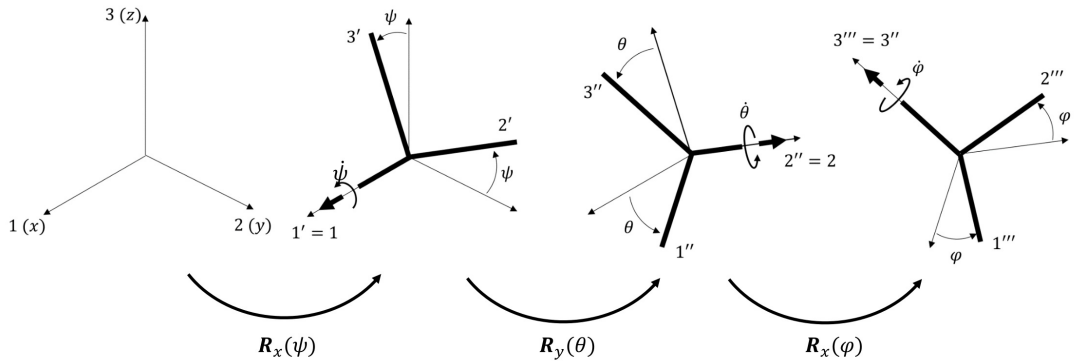


Figura A.1: Projecció de les velocitats angulars $\dot{\boldsymbol{\psi}}, \dot{\boldsymbol{\theta}}, \dot{\boldsymbol{\varphi}}$ sobre la base B_{abs} definida pels eixos 1, 2 i 3. Mentre el vector $\dot{\boldsymbol{\psi}}$ no queda afectat per cap rotació, els vectors $\dot{\boldsymbol{\theta}}$ i $\dot{\boldsymbol{\varphi}}$ queden afectats per la rotació $\mathbf{R}_x(\psi)$ i la rotació $\mathbf{R}_x(\psi) \cdot \mathbf{R}_y(\theta)$, respectivament.

A.2 Velocitat angular en funció dels angles d'Euler

Tenint en compte les matrius definides a la secció anterior, i les rotacions representades a la Fig. A.1, la velocitat angular absoluta de la plataforma en base B_{abs}, ω_{plat}, es pot expressar com

$$\boldsymbol{\omega}_{plat} = \underbrace{\begin{bmatrix} \dot{\psi} \\ 0 \\ 0 \end{bmatrix}}_{\dot{\boldsymbol{\psi}}} + \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\psi & -s_\psi \\ 0 & s_\psi & c_\psi \end{bmatrix}}_{\mathbf{R}_x(\psi)} \underbrace{\begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix}}_{\dot{\boldsymbol{\theta}}} + \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\psi & -s_\psi \\ 0 & s_\psi & c_\psi \end{bmatrix}}_{\mathbf{R}_x(\psi)} \underbrace{\begin{bmatrix} c_\theta & 0 & s_\theta \\ 0 & 1 & 0 \\ -s_\theta & 0 & c_\theta \end{bmatrix}}_{\mathbf{R}_y(\theta)} \underbrace{\begin{bmatrix} 0 \\ 0 \\ \dot{\varphi} \end{bmatrix}}_{\dot{\boldsymbol{\varphi}}}.$$

Si ara operem, obtenim

$$\boldsymbol{\omega}_{plat} = \begin{bmatrix} \dot{\psi} + \dot{\varphi} s_\theta \\ \dot{\theta} c_\psi - \dot{\varphi} s_\psi c_\theta \\ \dot{\theta} s_\psi + \dot{\varphi} c_\psi c_\theta \end{bmatrix},$$

i si separem el resultat en forma de producte entre matriu i vector

$$\boldsymbol{\omega}_{plat} = \underbrace{\begin{bmatrix} 1 & 0 & s_\theta \\ 0 & c_\psi & -s_\psi c_\theta \\ 0 & s_\psi & c_\psi c_\theta \end{bmatrix}}_{\mathbf{A}_m(\psi,\theta,\varphi)} \underbrace{\begin{bmatrix} \dot{\psi} \\ \dot{\theta} \\ \dot{\varphi} \end{bmatrix}}_{\dot{\boldsymbol{\alpha}}},$$

que de forma compacta expressarem com

$$\boldsymbol{\omega}_{plat} = \mathbf{A}_m \cdot \dot{\boldsymbol{\alpha}}. \tag{A.2}$$

A.3 Cinemàtica instantània

A.3.1 Solució del problema cinemàtic instantani invers

Per poder trobar el model en coordenades motor, a la Secció 2.5 es necessita l'expressió

$$\dot{\boldsymbol{\theta}} = \mathbf{J}_i \cdot \dot{\boldsymbol{q}} \tag{A.3}$$

que resol el problema cinemàtic instantani invers de l'Hexapole. Vegem tot seguit com podem trobar el Jacobià $\mathbf{J}_i$ indicat en aquesta expressió.

En primer lloc, si considerem les particions

$$\dot{\boldsymbol{q}} = (\underbrace{\dot{x}, \dot{y}, \dot{z}, \dot{\psi}, \dot{\theta}, \dot{\varphi}}_{\dot{\boldsymbol{t}}}, \underbrace{\dot{\beta}_1, \dot{\beta}_2}_{\dot{\boldsymbol{\beta}}})$$

i

$$\dot{\boldsymbol{\theta}} = (\underbrace{\dot{\gamma}_1, \ldots, \dot{\gamma}_6}_{\dot{\boldsymbol{\gamma}}}, \underbrace{\dot{\beta}_1, \dot{\beta}_2}_{\dot{\boldsymbol{\beta}}})$$

podem escriure l'Eq. (A.3) així

$$\begin{bmatrix} \dot{\boldsymbol{\gamma}} \\ \dot{\boldsymbol{\beta}} \end{bmatrix} = \begin{bmatrix} \mathbf{J}_i^{HC} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{2\times 2} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{t}} \\ \dot{\boldsymbol{\beta}} \end{bmatrix},$$

on $\mathbf{J}_i^{HC}$ és un Jacobià 6×6, i $\mathbf{I}_{2\times 2}$ és la matriu identitat de dimensió 2. Per trobar $\mathbf{J}_i$, per tant, només ens caldrà trobar el Jacobià $\mathbf{J}_i^{HC}$ de l'expressió

$$\dot{\boldsymbol{\gamma}} = \mathbf{J}_i^{HC} \cdot \dot{\boldsymbol{t}}, \tag{A.4}$$

que resol el problema cinemàtic invers de l'Hexacrane.

Obtindrem aquest Jacobià en dos passos. Primer buscarem la transformació que converteix $\dot{\boldsymbol{t}}$ en el torsor de velocitats de la plataforma reduït al punt G_{plat},

$$\hat{\boldsymbol{T}}_{plat} = \begin{bmatrix} \boldsymbol{v}_{G_{plat}} \\ \boldsymbol{\omega}_{plat} \end{bmatrix},$$

i després buscarem la transformació que converteix $\hat{\mathbf{T}}_{plat}$ en $\dot{\boldsymbol{\gamma}}$.

La primera transformació és fàcil de trobar, ja que de l'Eq. (2.25) tenim

$$\hat{\boldsymbol{T}}_{plat} = \begin{bmatrix} \boldsymbol{v}_{G_{plat}} \\ \boldsymbol{\omega}_{plat} \end{bmatrix} = \begin{bmatrix} \dot{\boldsymbol{p}} \\ \mathbf{A}_m \dot{\boldsymbol{\alpha}} \end{bmatrix},$$

i per tant

$$\hat{\boldsymbol{T}}_{plat} = \underbrace{\begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_m \end{bmatrix}}_{\mathbf{A}_e} \cdot \underbrace{\begin{bmatrix} \dot{\boldsymbol{p}} \\ \dot{\boldsymbol{\alpha}} \end{bmatrix}}_{\dot{\boldsymbol{t}}},$$

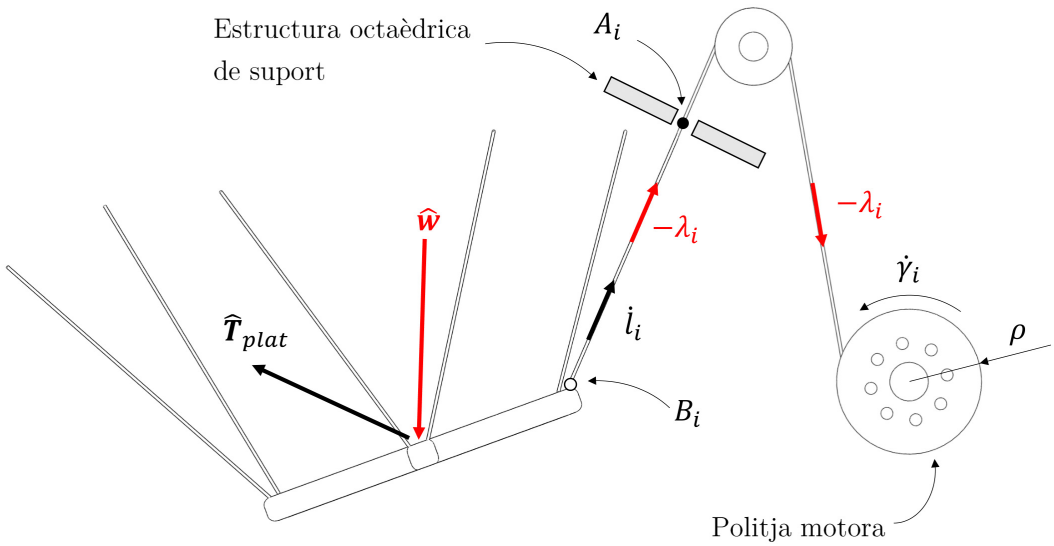


Figura A.2: Aplicació del principi de les potències virtuals al conjunt cables + plataforma. La plataforma es considera en equilibri sota un torsor de forces aplicat externament $\hat{\mathbf{w}}$, que queda compensat per les tensions $-\lambda_i$ efectuades per les politges motores sobre els cables. La velocitat virtual $\hat{\mathbf{T}}_{plat}$ indueix velocitats longitudinals $\dot{l}_i$ sobre els cables.

o, de manera compacta,

$$\hat{\mathbf{T}} = \mathbf{A}_e \cdot \dot{\boldsymbol{t}}. \tag{A.5}$$

Per trobar la segona transformació aplicarem el principi de les potències virtuals al subsistema format per la plataforma i els cables de l'Hexacrane, en condicions d'equilibri estàtic i gravetat nul·la (Fig. A.2). Per fer-ho, suposarem que en un cert instant de temps el robot es troba en equilibri, suportant un cert torsor $\hat{\mathbf{w}}$ de força aplicat des de l'exterior sobre la plataforma. Com a resultat d'aquest torsor, la plataforma aplicarà una tensió de magnitud λ_i sobre l'i-èssim cable, i per tant la politja del motor haurà d'aplicar una força equilibrant de magnitud $-\lambda_i$ sobre el cable. Considerarem, a més, que la plataforma es mou sota un cert torsor de velocitats virtual $\hat{\mathbf{T}}$ i que, com a conseqüència, l'i-èssim cable s'està arronsant a una certa velocitat $\dot{l}_i$, on l_i és la longitud del cable comptada des del punt A_i al B_i. Segons el principi de les potències virtuals, la potència de les forces exteriorment aplicades sobre el conjunt cables + plataforma ha de ser nul·la sota la velocitat virtual esmentada. Això vol dir que

$$-\boldsymbol{\lambda}^T \dot{\boldsymbol{l}} + \hat{\mathbf{w}}^T \cdot \hat{\mathbf{T}}_{plat} = 0, \tag{A.6}$$

on $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_6)$ i $\dot{\boldsymbol{l}} = (\dot{l}_1, \ldots, \dot{l}_6)$. En aquesta equació, $-\boldsymbol{\lambda}^T \dot{\boldsymbol{l}}$ és la potència generada per les forces equilibrants que fan les politges sobre els cables, i $\hat{\mathbf{w}}^T \cdot \hat{\mathbf{T}}_{plat}$ és la potència generada pel torsor $\hat{\mathbf{w}}$ sota la velocitat virtual $\hat{\mathbf{T}}_{plat}$. Si ara tenim en compte l'Eq. (2.24), ens adonem que $\hat{\mathbf{w}} = \mathbf{J} \cdot \boldsymbol{\lambda}$, i substituïnt aquesta expressió a l'Eq. (A.6) obtenim

$$\boldsymbol{\lambda}^T \cdot \dot{\boldsymbol{l}} = (\mathbf{J}\ \boldsymbol{\lambda})^T \cdot \hat{\mathbf{T}}_{plat},$$

i per tant

$$\boldsymbol{\lambda}^T \cdot \dot{\boldsymbol{l}} = \boldsymbol{\lambda}^T \cdot \mathbf{J}^T \cdot \hat{\mathbf{T}}$$

Com que aquesta expressió ha de ser vàlida independentment del torsor $\hat{\mathbf{w}}$ assumit inicialment, també ho ha de ser per qualsevol valor de $\boldsymbol{\lambda}$, i podem concloure que

$$\dot{\boldsymbol{l}} = \mathbf{J}^T \cdot \hat{\mathbf{T}}_{plat}$$

Ara bé, com que $\dot{l}_i = \gamma_i \cdot \rho$, llavors $\dot{\boldsymbol{l}} = \dot{\boldsymbol{\gamma}} \cdot \rho$, i per tant

$$\dot{\boldsymbol{\gamma}} = \frac{1}{\rho} \cdot \mathbf{J}^T \cdot \hat{\mathbf{T}}_{plat} \tag{A.7}$$

que és la segona transformació que buscàvem.

Utilitzant les Eqs. (A.5) i (A.7) arribem finalment a l'expressió

$$\dot{\boldsymbol{\gamma}} = \frac{1}{\rho} \cdot \mathbf{J}^T \cdot \mathbf{A}_e \cdot \dot{\boldsymbol{t}}$$

i identificant termes amb l'Eq. (A.4) veiem que

$$\mathbf{J}_i^{HC} = \frac{1}{\rho} \cdot \mathbf{J}^T \cdot \mathbf{A}_e$$

de manera que

$$\mathbf{J}_i = \begin{bmatrix} \frac{1}{\rho} \cdot \mathbf{J}^T \cdot \mathbf{A}_e & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{2\times 2} \end{bmatrix}. \tag{A.8}$$

A.3.2 Solució del problema cinemàtic instantani directe

Per poder simular el model en coordenades motor, a la Secció 6.1.2 necessitem problema cinemàtic instantani directe de l'Hexapole. Tenint en compte els resultats de la secció anterior, aquesta solució ve donada per

$$\dot{\boldsymbol{q}} = \mathbf{J}_i^{-1}\, \dot{\boldsymbol{\theta}}$$

on el Jacobià $\mathbf{J}_i$ té l'expressió indicada a l'Eq. (A.8). Aquest Jacobià és funció de la configuració $\boldsymbol{q}$ de l'Hexapole, i $\boldsymbol{q}$ es pot trobar a partir de $\boldsymbol{\theta}$ resolent el problema cinemàtic directe de l'Hexapole (Secció A.4).

A.4 Cinemàtica directa

A.4.1 Problema cinemàtic directe de l'Hexapole

Tal i com expliquem a la Secció 6.1.2, per poder avaluar $f(\boldsymbol{x}, \boldsymbol{u})$ es necessari resoldre el problema cinemàtic directe de l'Hexapole. Aquest problema consisteix en trobar el valor de

$$\boldsymbol{q} = (\underbrace{x, y, z, \psi, \theta, \varphi}_{\boldsymbol{t}}, \underbrace{\beta_1, \beta_2}_{\boldsymbol{\beta}})$$

que correspon a un cert valor de

$$\boldsymbol{\theta} = (\underbrace{\gamma_1, \ldots, \gamma_6}_{\boldsymbol{\gamma}}, \underbrace{\beta_1, \beta_2}_{\boldsymbol{\beta}})$$

tot satisfent les restriccions cinemàtiques del robot. La funció que transforma un valor de $\boldsymbol{\theta}$ en el valor corresponent de $\boldsymbol{q}$ s'anomena funció cinemàtica directa, i es denotarà així

$$\boldsymbol{q} = F_{FK}(\boldsymbol{\theta}).$$

Com veurem, però, en el cas de l'Hexapole aquesta funció només es podrà implementar numèricament partint d'una aproximació $\boldsymbol{q}_a$ prou bona de la solució, i per tant en realitat implementarem

$$\dot{\boldsymbol{q}} = F_{FK}(\boldsymbol{\theta}, \boldsymbol{q}_a).$$

Com que les dues darreres components de $\boldsymbol{\theta}$ i $\boldsymbol{q}$, β_1 i β_2, són idèntiques, per implementar $F_{FK}(\boldsymbol{\theta}, \boldsymbol{q}_a)$, per implementar $F_{FK}(\boldsymbol{\theta}, \boldsymbol{q}_a)$ només cal que expliquem com podem calcular el valor de la configuració de la plataforma,

$$\boldsymbol{t} = (x, y, z, \psi, \theta, \varphi),$$

a partir dels angles dels motors

$$\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_6).$$

És a dir, només cal trobar la solució del problema cinemàtic directe de l'Hexacrane.

A.4.2 Problema cinemàtic directe de l'Hexacrane

Resoldrem aquest problema en dues fases. Primer veurem com a partir dels angles γ_i dels motors podem calcular les longituds l_i dels cables (la distància entre els punts A_i i B_i (Fig. A.3), per $i = 1, \ldots, 6$), i després veurem com, a partir d'aquestes longituds, podem calcular el valor de $\boldsymbol{t} = (x, y, z, \psi, \theta, \varphi)$.

El primer pas és senzill. Si assumim que les politges dels motors estan muntades com en la Fig. A.3, un escurçament de l'i-èssim cable $\Delta l_i > 0$ correspondrà a una rotació en el sentit contrari de les agulles del rellotge $\Delta\gamma_i > 0$ de la politja del motor (Fig. A.4). Per tant, si ρ és el radi d'aquesta politja, podem escriure

$$\Delta l_i = \rho \cdot \Delta\gamma_i,$$

o, equivalentment

$$l_i - l_{i,o} = \rho \cdot (\gamma_i - \gamma_{i,o}),$$

on $l_{i,o}$ i $\gamma_{i,o}$ són, respectivament, la longitud de l'i-èssim cable i l'angle de l'i-èssim motor a la configuració d'equilibri desitjada, que prenem com a configuració de referència. Si fixem $\gamma_{i,o} = 0$ per $i = 1, \ldots, 6$, l'anterior expressió es redueix a

$$l_i = l_{i,o} + \rho \cdot \gamma_i,$$

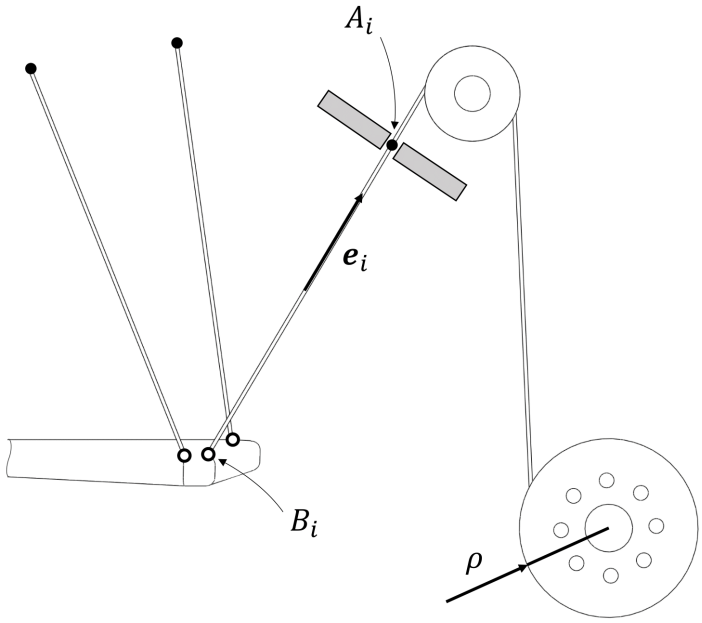


Figura A.3: Esquema del muntatge del sistema de cables i politges dels motors.

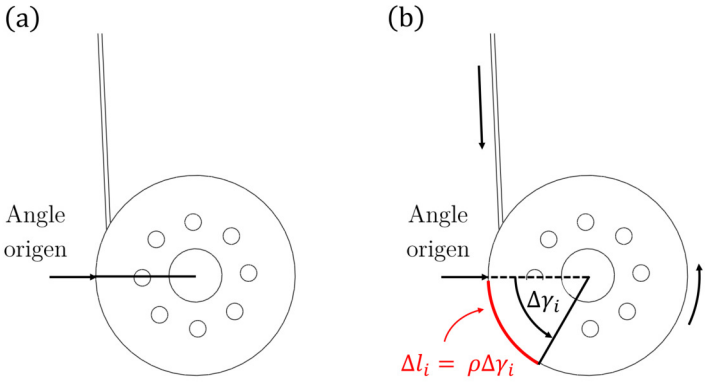


Figura A.4: **(a)** Politja a la posició d'origen on l'angle del motor $\theta = \theta_o = 0$, correspon a una longitud del cable $l_{o,i}$. **(b)** Politja en una posició angular θ genèrica després d'un escurçament del cable $\Delta\theta = \theta - \theta_o = \theta$.

que ens dóna la relació entre angles i longituds que buscàvem.

Per calcular $\boldsymbol{t} = (x, y, z, \psi, \theta, \varphi)$ a partir de $\boldsymbol{l} = (l_1, \ldots, l_6)$ només cal tenir en compte la següent relació

$$l_i^2 = |\boldsymbol{a}_i - (\boldsymbol{p} + \mathbf{R}_{plat}\boldsymbol{b}_i)|^2 \tag{A.9}$$

per $i = 1, \ldots, 6$, que es dedueix de la geometria de la Fig. A.5, i on $\boldsymbol{a}_i$ i $\boldsymbol{p}$ estan expressats en la base B_{abs} i $\boldsymbol{b}_i$ està expressat en la base B_{plat}.

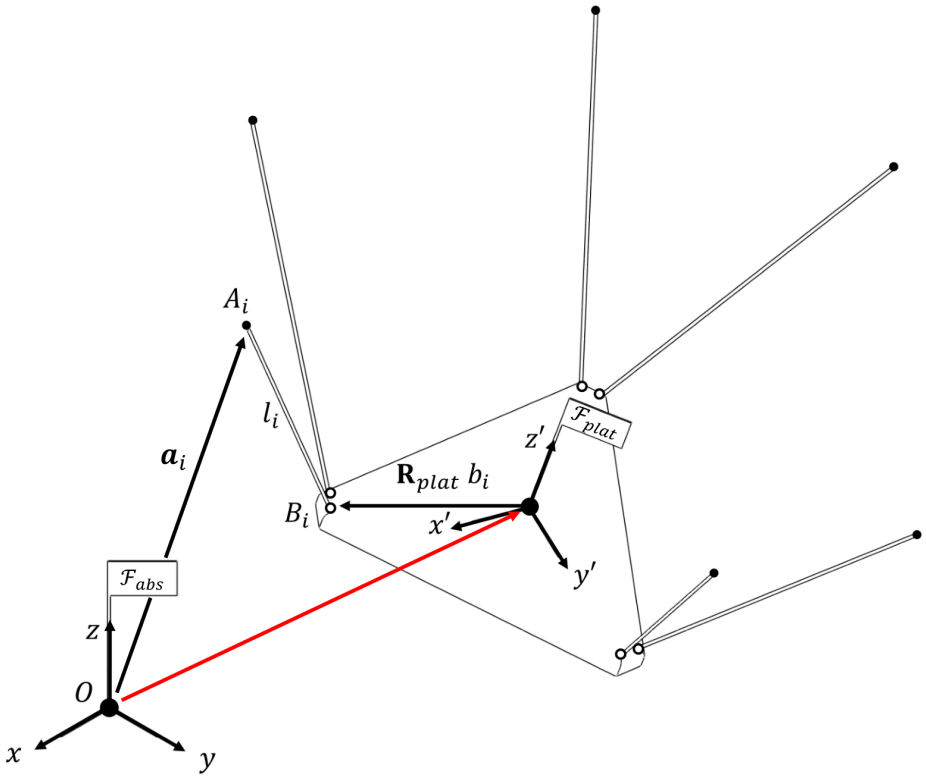


Figura A.5: Relació entre l_i i $\boldsymbol{a}_i$, $\boldsymbol{b}_i$, $\boldsymbol{p}$, i $\mathbf{R}_{plat}$.

Si reorganitzem l'Eq. (A.9) podem escriure

$$\mathbf{0} = |\boldsymbol{a}_i - \boldsymbol{p} - \mathbf{R}_{plat}\boldsymbol{b}_i|^2 - l_i^2 \tag{A.10}$$

i si agrupem les Eqs. (A.10) per tots els cables obtenim un sistema d'equacions no lineal de la forma

$$F(\underbrace{l_1, \cdots, l_6}_{\boldsymbol{l}}, \underbrace{\overbrace{x, y, z}^{\boldsymbol{p}}, \overbrace{\psi, \theta, \varphi}^{\boldsymbol{\alpha}}}_{\boldsymbol{t}}) = \mathbf{0} \tag{A.11}$$

Així doncs donat un vector de longituds de cables $\boldsymbol{l}$ caldrà trobar $\boldsymbol{t}$. Com és ben conegut a l'àmbit de la cinemàtica de robots paral·lels, l'Eq. (A.11) no té solució tancada per un valor d'$\boldsymbol{l}$ donat i per tant hem de recórrer a un mètode numèric per resoldre-la. A tal efecte, en aquest treball hem utilitzat el mètode de Newton-Raphson que, partint d'una estimació prou bona de la solució, té convergència quadràtica cap aquesta. A la Secció A.4.3 explicarem aquest mètode iteratiu, que per poder ser utilitzat requereix conèixer el Jacobià $\frac{\partial F}{\partial \boldsymbol{t}}$.

L'expressió de $\frac{\partial F}{\partial \boldsymbol{t}}$ es pot obtenir de la següent manera. Primer de tot la derivada temporal de l'Eq. (A.11) serà

$$\frac{\partial F}{\partial \boldsymbol{l}}\, \dot{\boldsymbol{l}} + \frac{\partial F}{\partial \boldsymbol{t}}\, \dot{\boldsymbol{t}} = \mathbf{0}, \tag{A.12}$$

on

$$\frac{\partial F}{\partial \boldsymbol{l}} = \begin{bmatrix} -2l_1 & & \\ & \ddots & \\ & & -2l_6 \end{bmatrix}, \tag{A.13}$$

i si aïllem $\dot{\boldsymbol{l}}$ de l'Eq. (A.12) serà

$$\dot{\boldsymbol{l}} = -\left(\frac{\partial F}{\partial \boldsymbol{l}}\right)^{-1} \frac{\partial F}{\partial \boldsymbol{t}} \cdot \dot{\boldsymbol{t}}. \tag{A.14}$$

D'altra banda, a la Secció A.3 es demostra que

$$\dot{\boldsymbol{l}} = \mathbf{J}^T \hat{\boldsymbol{T}}_{plat}, \tag{A.15}$$

on $\mathbf{J}$ i $\hat{\mathbf{T}}_{plat}$ són el Jacobià de forces de l'Hexacrane i el torsor de velocitats de la plataforma reduït al punt G_{plat}, les expressions dels quals ja s'han donat a la Secció 2.4.3. Així doncs, utilitzant l'Eq. (2.25) podem reescriure l'Eq. (A.15) de la següent manera

$$\dot{\boldsymbol{l}} = \mathbf{J}^T \begin{bmatrix} \dot{\boldsymbol{p}} \\ \mathbf{A}_m \dot{\boldsymbol{\alpha}} \end{bmatrix}.$$

Si tenim en compte que $\boldsymbol{t} = (\boldsymbol{p}, \boldsymbol{\alpha})$, però, l'anterior equació es pot expressar així

$$\dot{\boldsymbol{l}} = \mathbf{J}^T \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_m \end{bmatrix} \dot{\boldsymbol{t}}, \tag{A.16}$$

i identificant termes amb l'Eq. (A.14) tenim que

$$-\left(\frac{\partial F}{\partial \boldsymbol{l}}\right)^{-1} \cdot \frac{\partial F}{\partial \boldsymbol{t}} = \mathbf{J}^T \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_m \end{bmatrix},$$

de manera que

$$\frac{\partial F}{\partial \boldsymbol{t}} = -\frac{\partial F}{\partial \boldsymbol{l}} \cdot \mathbf{J}^T \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_m \end{bmatrix}.$$

Si en aquesta última expressió substituïm ara l'Eq. (A.13) i l'expressió del $\mathbf{J}$ de l'Eq. (2.24), obtenim

$$\frac{\partial F}{\partial \boldsymbol{t}} = 2 \begin{bmatrix} l_1 & & \\ & \ddots & \\ & & l_6 \end{bmatrix} \begin{bmatrix} \boldsymbol{e}_1 & \cdots & \boldsymbol{e}_6 \\ \boldsymbol{r}_1 \times \boldsymbol{e}_6 & \cdots & \boldsymbol{r}_6 \times \boldsymbol{e}_6 \end{bmatrix}^T \begin{bmatrix} \boldsymbol{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_m \end{bmatrix}. \tag{A.17}$$

Finalment, si $\boldsymbol{d}_i$ és el vector que va de B_i fins a A_i, tenim

$$\boldsymbol{d}_i = l_i \boldsymbol{e}_i,$$

i per tant $\boldsymbol{e}_i$ és

$$\boldsymbol{e}_i = \frac{1}{l_i} \boldsymbol{d}_i,$$

de manera que substituint aquesta última equació a l'Eq. (A.17) obtenim el Jacobià que neces-

sitem en el mètode de Newton:

$$\frac{\partial F}{\partial \boldsymbol{t}} = -2 \begin{bmatrix} \boldsymbol{d}_1 & \cdots & \boldsymbol{d}_6 \\ \boldsymbol{r}_1 \times \boldsymbol{d}_1 & \cdots & \boldsymbol{r} \times \boldsymbol{d}_6 \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_m \end{bmatrix}.$$

A.4.3 Mètode de Newton-Raphson

Per tal de poder donar solució numèrica a l'Eq. (A.11) implementarem el mètode de Newton-Raphson. En aquesta secció exposarem de forma genèrica en què consisteix aquest mètode.

L'objectiu del mètode és trobar una solució d'un sistema d'equacions $F(\boldsymbol{x}) = \mathbf{0}$, on $F : \mathbb{R}^n \to \mathbb{R}^n$ és una funció diferenciable, partint d'una aproximació prou bona de la solució, denotada per $\boldsymbol{x}_o$. Per veure com funciona el mètode, considerem primer el desenvolupament en sèrie de Taylor de $\boldsymbol{y} = F(\boldsymbol{x})$ en $\boldsymbol{x}_o$,

$$F(\boldsymbol{x}_o + \delta\boldsymbol{x}) = F(\boldsymbol{x}_o) + F_{\boldsymbol{x}}(\boldsymbol{x}_o)\delta\boldsymbol{x} + O(\delta\boldsymbol{x}^2), \tag{A.18}$$

on $F_{\boldsymbol{x}}(\boldsymbol{x}_o)$ és el Jacobià $\frac{\partial F}{\partial \boldsymbol{x}}$ avaluat a $\boldsymbol{x}_o$. En un entorn local d'$\boldsymbol{x}_o$ podem considerar que l'aproximació de Taylor fins als termes de primer ordre és prou bona, de manera que $O(\delta\boldsymbol{x}^2)$ és prou petit i l'Eq. (A.18) es pot reescriure com

$$F(\boldsymbol{x}_o + \delta\boldsymbol{x}) \approx F(\boldsymbol{x}_o) + F_{\boldsymbol{x}}(\boldsymbol{x}_o) \cdot \delta\boldsymbol{x}.$$

Com que es vol trobar el punt on $F(\boldsymbol{x}_o + \delta\boldsymbol{x}) = 0$ resoldrem

$$0 = F(\boldsymbol{x}_o) + F_{\boldsymbol{x}}(\boldsymbol{x}_o) \cdot \delta\boldsymbol{x},$$

i per tant l'increment $\delta\boldsymbol{x}$ que compleixi

$$F_{\boldsymbol{x}}(\boldsymbol{x}_o) \cdot \delta\boldsymbol{x} = -F(\boldsymbol{x}_o) \tag{A.19}$$

és el que en principi ens acostarà més a la solució buscada. El sistema (A.19) és lineal i es pot resoldre utilitzant el mètode de descomposició LU per exemple [17]. La nova aproximació de la solució serà

$$\boldsymbol{x}_{nova} = \boldsymbol{x}_o + \delta\boldsymbol{x}.$$

Si ara fem $\boldsymbol{x}_o = \boldsymbol{x}_{nova}$ i iterem el procés, convergirem ràpidament a la solució buscada. La Fig. A.6 il·lustra una iteració del mètode en el cas unidimensional.

A.5 Criteris de signes utilitzats

Tal i com com s'ha construït el Jacobià $\mathbf{J}$, queden implícitament definits els següents criteris de signes. En primer lloc, quan el cable i s'allarga, la velocitat del cable $\dot{l}_i$ és negativa ($\dot{l}_i < 0$). Aquest fet és degut a que la velocitat d'allargament és $\dot{l}_i \cdot \boldsymbol{e}_i$ i $\boldsymbol{e}_i$ s'ha definit com un vector orientat des del punt B_i cap a l'A_i (Fig. A.3). Es pot verificar aquest criteri amb un simple càlcul. Imaginem que el torsor de velocitats de la plataforma és

$$\hat{\boldsymbol{T}}_{plat} = (0, 0, -1, 0, 0, 0)$$

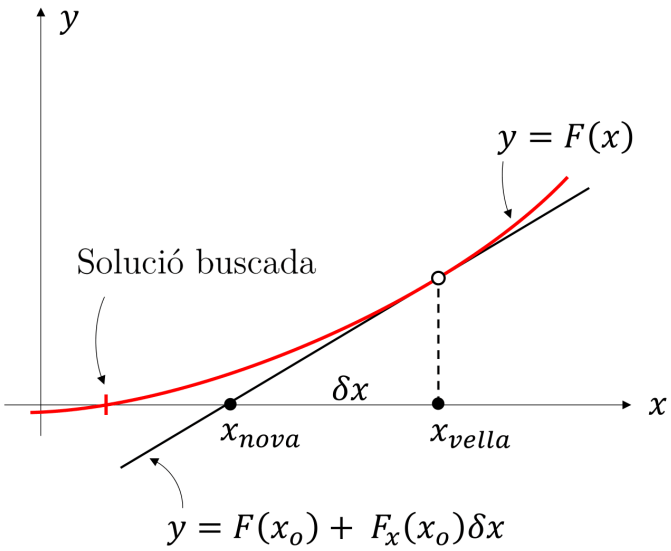


Figura A.6: Una iteració del mètode de Newton-Raphson en el cas unidimensional.

i que, per tant, la plataforma es trasllada en la direcció z negativa a $1\ m/s$ sense rotar. Sota aquest moviment, els cables necessàriament s'allarguen. Si ara calculem la velocitat dels cables resolent el problema cinemàtic invers

$$\underbrace{\begin{bmatrix} \dot{l}_1 \\ \vdots \end{bmatrix}}_{\dot{l}} = \underbrace{\begin{bmatrix} (\boldsymbol{e}_1)^T & (\boldsymbol{r}_1 \times \boldsymbol{e}_1)^T \\ \vdots & \vdots \end{bmatrix}}_{\mathbf{J}^T} \underbrace{\begin{bmatrix} 0 \\ 0 \\ -1 \\ 0 \\ 0 \\ 0 \end{bmatrix}}_{\hat{\mathbf{T}}_{plat}}$$

veiem clarament que les $\dot{l}_i$ són iguals a la component z de e_i canviada de signe, i per tant són totes negatives.

Cal notar que els increments de longitud del cable i, $\Delta l_i = l_i - l_{i,o}$, tenen el mateix signe que $\dot{l}_i$, ja que si considerem un increment de temps positiu prou petit podem escriure

$$\dot{l}_i = \frac{\Delta l_i}{\Delta t} \Leftrightarrow \Delta l_i = \dot{l}_i \Delta t$$

i per tant Δl_i serà del mateix signe que $\dot{l}_i$. Podem concloure que un allargament del cable i, per tant, implica una variació negativa Δl_i. L'única manera de complir amb aquest criteri és assumir que les longituds dels cables són sempre negatives i per tant la diferencia $\Delta l_i = l_i - l_{i,o}$ serà entre un valor d'origen negatiu $l_{i,o}$ i un valor negatiu l_i que serà o bé més petit que $l_{i,o}$, si el cable s'allarga, o bé més gran que $l_{i,o}$, si el cable s'escurça. Per tal de complir amb aquestes

longituds negatives caldrà que d'entre les dues solucions del problema cinemàtic invers,

$$l_i = \pm\sqrt{|\boldsymbol{a}_i - \boldsymbol{p} - \mathbf{R}_{plat}\boldsymbol{b}_i|^2},$$

escollim sempre la negativa.

B
Pressupost del projecte

El pressupost que presentem a continuació avalua el cost d'el·laboració del programari desenvolupat, tant en mitjans humans com materials.

B.1 Estimacions de partida

En aquest apartat especificarem els costos horaris per a la confecció del pressupost. Considerem dos tipus de costos:

- El cost horari del personal.
- El cost horari d'utilització dels equips.

B.1.1 Cost horari del personal

Estimem els següents salaris:

$$\begin{array}{ll} Analista & \ldots \quad 15,62\ EUR/hora \\ Programador & \ldots \quad 13,97\ EUR/hora \\ Operador & \ldots \quad 10,41\ EUR/hora \end{array}$$

B.1.2 Cost horari d'utilització dels equips

Per a la valoració del cost horari d'utilització dels equips cal tenir en compte: la seva amortització, el cost del personal de l'Institut de Robòtica i Informàtica Industrial (CSIC-UPC) necessari per al bon funcionament de l'equip, i els contractes de manteniment, si n'hi ha.

L'únic equip que s'ha utilitzat en aquest projecte es un PC de sobretaula, amb els costos associats següents:

1. **Cost de l'equip:** El PC utilitzat inclou una torre PC HP ProDesk 400 G4, amb dic dur SATA de 7.200 rpm, 1TB 3,5", placa Intel HD Graphics 630, targeta de xarxa Realtek RTL8111 HSH GbE LOM, 2 ports USB 3.1 Gen1, 4 ports USB 2.0, CPU Intel Core i5-7500 3,4GHz-3,8Ghz, memòria SDRAM de 8Gb DDR4-2400 (1 x 8 Gb), teclat compacte HP USB Business, ratolí óptic USB d'HP, monitor HP 20kd 19,5"(1440x900 a 60Hz), i sistema operatiu Windows 7. El cost d' aquest equip és aproximadament de 746,75 EUR. Aquesta quantitat inclou també les despeses d'instal·lació de la màquina. Si considerem que es

vol amortitzar l'equip en cinc anys, amb uns interessos del 15%, l'anualitat d'amortització resulta de:

$$C(equip) = 746,75 \times \frac{0,15 \times 1,15^5}{1,15^5 - 1} = 222,76\ EUR/any$$

2. **Manteniment:** El cost de manteniment anual de l'equip complet s'avalua en un 10% del seu preu de compra:

$$C(manteniment) = 746,75 \times 0,1 = 74,67\ EUR/any$$

3. **Consum elèctric:** S'avalua en:

$$C(consum) = 100\ EUR/any$$

4. **Cost del personal del IRI:** Es consideren els costos d'un operador treballant 0,5 hores setmanals durant 40 setmanes cada any:

$$C(pers\ IRI) = 208\ EUR/any$$

A partir dels anteriors costos parcials, el cost total anual serà:

$C(equip)$	...	$222,76\ EUR/any$
$C(manteniment)$	...	$74,67\ EUR/any$
$C(pers\ IRI)$	...	$208\ EUR/any$
$C(consum)$	...	$100\ EUR/any$
$C(anual)$	...	$605,43\ EUR/any$

El cost horari de CPU, considerant que es treballen 8 hores/dia i 5 dies/setmana durant 40 setmanes/any i que el temps d'utilització de la CPU és del 60%, resulta:

$$C(horariCPU) = \frac{605,43}{8 \times 5 \times 40 \times 0,6} = 0,63\ EUR/hora$$

B.2 Cost de desenvolupament

Tenint en compte els preus estipulats en l'apartat anterior, calcularem tot seguit el cost de desenvolupament del projecte, és a dir, el cost del personal, equips i despeses vàries. Cal dir que no es tenen en compte els costos referents a l'immobilitzat de l'IRI, sinó solament els de desenvolupament del projecte pel que fa a recursos humans i equips informàtics.

B.2.1 Cost del personal

S'estima que el temps esmerçat en el projecte ha estat de 23 setmanes amb dedicació completa de 40 hores/setmana. En el projecte hi ha treballat una persona cobrant (estimem) el sou d'un analista informàtic:

$$C(personal) = 920\ hores \times 15,62\ EUR/hora = 14.370,4\ EUR$$

B.2.2 Cost d'utilització dels equips

S'estima que del temps total del projecte, el 60% ha estat dedicat a la implementació dels programes. De fet cal descomptar un 35% d'aquest temps per temps morts de CPU davant l'ordinador. Es considera que quan s'utilitza la CPU, realment n'utilitzem el 80% d'ella (tot i ser multiusuari, el règim d'utilització és pràcticament monousuari). Per tant, podem avaluar els següents temps:

$$\begin{array}{lrcr} Temps\ d'ordinador & 920 \times 0,60 & = & 552\ hores, \\ Temps\ d'ordinador\ útil & 552 \times 0,65 & = & 358,80\ hores, \\ Temps\ de\ CPU & 358,80 \times 0,80 & = & 287,04\ hores, \end{array}$$

i el cost de CPU resulta ser de

$$C(CPU) = 287,04 \times 0,63 = 180,83\ EUR.$$

B.2.3 Despeses de documentació i impressió

Es consideren els conceptes següents:

$$\begin{array}{lcr} Documentació & \ldots & 90,00\ EUR \\ Paper\ i\ impressió & \ldots & 15,00\ EUR \\ \hline C(varis) & & 105,00\ EUR \end{array}$$

B.2.4 Cost total de desenvolupament

El cost total de desenvolupament del projecte, finalment, és la suma dels costos anteriors:

$$\begin{array}{lcr} C(personal) & \ldots & 14.370,40\ EUR \\ C(CPU) & \ldots & 180,83\ EUR \\ C(varis) & \ldots & 105,00\ EUR \\ \hline C(TOTAL) & \ldots & 14.656,23\ EUR \end{array}$$

C
Codi Font

Per tal de poder dissenyar el sistema de control i fer les simulacions corresponents, s'han creat una sèrie de funcions i *scripts* en MAPLE i MATLAB (Fig. C.1). Com ja s'ha esmentat en capítols anteriors, utilitzarem MAPLE, per obtenir els models dinàmics de l'Hexapole, i per tant el codi escrit en l'entorn MAPLE també quedarà exposat en aquest annex. L'esquema principal de funcionament és molt senzill. En primer lloc escrivim un script anomenat `WriteDataFile.m`, en el qual configurem els paràmetres dinàmics del model que necessiten ser ajustats al llarg del treball. Aquest script crearà un fitxer de text, *DynamicData.txt*, que podrà ser llegit tant per MATLAB com per MAPLE. Posteriorment aquest fitxer de dades serà llegit per MAPLE per tal de generar totes les matrius i vectors que conformen el model dinàmic en la forma manipulador. Una vegada construïdes aquestes matrius i vectors, caldrà executar l'script `MainDMMXcoord.m`, si es vol simular el model mecànic, o bé l'script `MainDMHexaMX28.m`, si es vol simular el model electromecànic. Cadascun d'aquests scripts llegeix el fitxer `DynamicData.txt`, i estableix els paràmetres geomètrics del robot. També configura els paràmetres de la simulació com la durada i la rutina d'integració que es vol utilitzar, quines variables volem que dibuixi, si es vol activar o no la llei de control, les condicions inicials, i si volem que ens generi un vídeo al final de la simulació o no. És en aquest script on també construïm les matrius $\mathbf{A}$ i $\mathbf{B}$ que linealitzen el model cridant la funció `LinealHexaAnalitic` implementada en un fitxer apart. D'aquesta funció n'existeixen també dues versions, una per cada model a simular. Posteriorment es defineixen les matrius $\mathbf{Q}$ i $\mathbf{R}$ per ajustar la llei de control. Amb aquestes dades trobem la matriu $\mathbf{K}$ utilitzant la comanda

```
[K,SolRic,Eigvals]=lqr(Alin,Blin,Q,R)
```

A partir d'aquí iniciem la simulació, cridant diverses funcions particularitzades per cada model. En primer lloc, es crida la funció que implementa el model dinàmic en forma explícita de primer ordre anomenada `xdot`. I per últim, necessitarem una funció anomenada `uopertwrench`, en la qual definim en quin instant i quina magnitud tindran les pertorbacions que volem aplicar sobre el pèndol.

Paral·lelament als scripts i funcions principals s'ha dissenyat un grup de funcions comunes que poden ser cridades per qualsevol dels grups de funcions o scripts anteriors. Aquestes funcions s'encarreguen de resoldre tres aspectes principals. En primer lloc, per simular el model necessitem solucionar diverses vegades el problema cinemàtic directe i el problema cinemàtic directe instantani, juntament amb les seves versions inverses. Un primer grup de funcions s'encarreguen d'aquesta tasca. En segon lloc, és necessari que a partir del vector de coordenades generalitzades puguem dibuixar la configuració del robot. Un segon grup

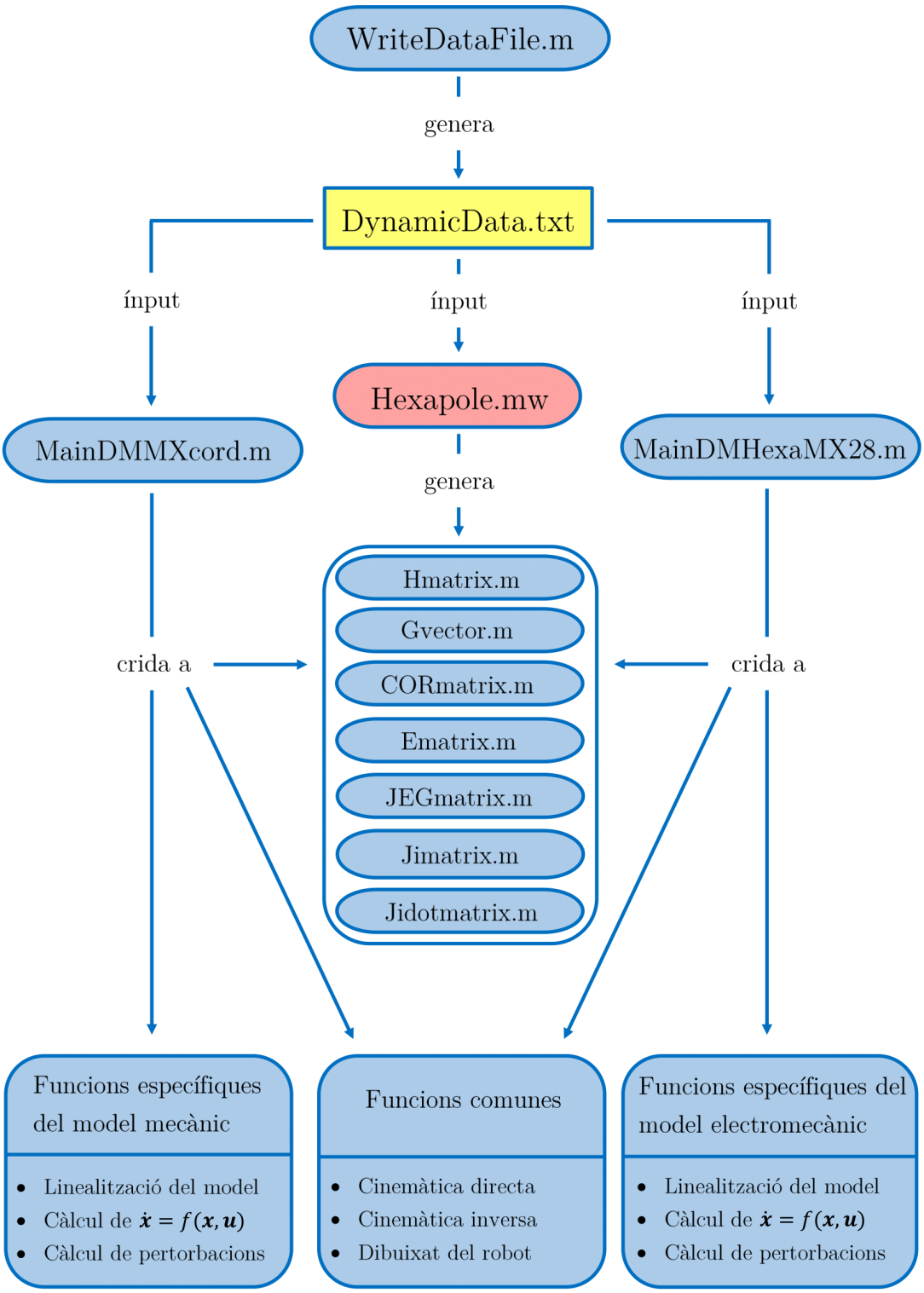


Figura C.1: Organigrama que descriu l'estructura del codi font implementat.

de funcions s'encarrega d'implementar aquesta tasca. En tercer lloc, la part final dels scripts principals s'escriu un tros de codi que permet cridar múltiples vegades aquestes funcions i d'aquesta manera s'aconsegueix generar un vídeo del moviment simulat.

C.1 Programa WriteDataFile.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Write_Data_file.m
%
% Purpose:
%
%    The aim of this script is to create the Dynamics_Data.txt file
%    containing the dynamic parameters of the Hexapole. These parameters
%    include those that are supposed to be adjusted during the project.
%    The remaining parameters are hard-coded in the main files Hexapole.mw,
%    MainDMMXcord.m and MainDMHexaMX28.m.
%
% User-specified parameters:
%
%    The pole length
%    The platform mass and inertia tensor
%    The pole mass and inertia tensor
%
% Output:
%
%    A file Dynamics_Data.txt containing the previous parameters in the format
%    assumed by the MATLAB programs reading this file.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Parameters of the model

ls=0.96724;   % length of the stick that links the ball with the platform,[m]
mp=0.59895;   % mass of the platform [kg]
mb=0.41472;   % mass of the ball+stick [kg]

% Values of the platform tensor

ITpxx=270725.71*1e-9; %Moment of inertia about the x axis [kg*m^2]
ITpyy=270725.71*1e-9; %Moment of inertia about the y axis [kg*m^2]
ITpzz=409645.04*1e-9; %Moment of inertia about the z axis [kg*m^2]

% Values of the pendulum tensor
```

```
ITbxx=11262205.59*1e-9; %Moment of inertia about the x axis [kg*m^2]
ITbyy=11262205.59*1e-9; %Moment of inertia about the y axis [kg*m^2]
ITbzz=77397.82*1e-9;    %Moment of inertia about the z axis [kg*m^2]

Dyn_Vector=[ls,mp,mb,ITpxx,ITpyy,ITpzz,ITbxx,ITbyy,ITbzz]';

Dyn_Param=to_file('Dynamic_Data',Dyn_Vector);
```

C.2 Programa Hexapole.mw

```
# -------------------------------------------------------------------------
# Initializations
# -------------------------------------------------------------------------

restart;
with(Student[MultivariateCalculus]):
with(Student[LinearAlgebra]):
with(CodeGeneration):
with(ArrayTools):
with(DynamicSystems):
interface(rtablesize=20):

# -------------------------------------------------------------------------
# Function to take partial derivatives with respect to x(t), y(t), ...
# -------------------------------------------------------------------------

sdiff := proc (expr, sym)
   local t;
   subs(t = sym, diff(subs(sym = t, expr), t))
end proc:

# -------------------------------------------------------------------------
# Function to compute the Jacobian of a vector
#
#    expr = a vector function, with variables that can be time-dependent
#    sym  = a list of variables with respect to which we take the derivatives
# -------------------------------------------------------------------------

sJacobian := proc (expr, sym)
    local out,i;
    out := Matrix(max(Size(expr)), max(Size(sym)));
    for i from 1 to max(Size(sym)) do
        out[ .. , i] := sdiff(expr, sym[i]);
    end do;
    out;
end proc:
Read data from file
Cdir:=currentdir();

currentdir("C:/Users/pgiro/Documents/MATLAB");
```

```
Geom_Vec:=readdata("Geom_Data.txt",float,9);

currentdir(Cdir);

# Platform mass, radius, and inertia tensor
mp:=Geom_Vec[1,2];
Ip := Matrix(3,3,[[Geom_Vec[1,4],0,0],[0,Geom_Vec[1,5],0],[0,0,Geom_Vec[1,6]]]);

# Ball-and-stick mass, radius, inertia tensor, and distance l from Gp to Gb
mb:=Geom_Vec[1,3];
Ib := Matrix(3,3,[[Geom_Vec[1,7],0,0],[0,Geom_Vec[1,8],0],[0,0,Geom_Vec[1,9]]]);
l := Geom_Vec[1,1];

# ------------------------------------------------------------------------
# Geometric and dynamic parameters in SI units
# ------------------------------------------------------------------------

# By columns, the anchor points at the base, in the absolute frame [m]

Apoints := 1e-3 *
           <<-231.6200 |  231.6200  |  233.7400  |   2.1300 |  -2.1300 | -233.7400 >,
            <-136.1800 | -136.1800  | -132.5000  | 268.6700 | 268.6700 | -132.5000 >,
            <   0      |    0       |    0       |   0      |   0      |    0      >>;

# By columns, the anchor points at the platform, in the relative frame [m]

Bpoints := 1e-3 *
           <<   0      |   0       | 77.2100 |  77.2100 | -77.2100  | -77.2100 >,
            < -89.1500 | -89.1500  | 44.5700 |  44.5700 |  44.5700  |  44.5700 >,
            <   0      |   0       |  0      |   0      |   0       |   0      >>;


# ------------------------------------------------------------------------
# Gravity constant [m/s^2]
#
# We shall use it as "gconst" below, in section "Potential Energy of the System".
# Uncomment the following if you wish to set it to a specific numerical value.
# ------------------------------------------------------------------------

gconst := 9.8;

# ------------------------------------------------------------------------
# Configuration coordinates and their derivatives
# ------------------------------------------------------------------------

n_q := 8:

q := Vector([ x(t),
              y(t),
              z(t),
              psi(t),
              theta(t),
              phi(t),
              beta1(t),
              beta2(t)]):
```

```
dq := diff(q,t):
ddq := diff(dq,t):

#########################################################################
#
# The equation of motion takes the form
#
#      Msys(q) * ddq + CC(q,dq) + G(q) = E(q) * u
#
# where:
#
#      Msys      =  Mass matrix of the system, of dim n_q x n_q
#      CC(q,dq)  =  Generalized Coriolis and Centrifugal force
#      G(q)      =  Generalized gravity force
#      E(q) * u  =  Generalized actuation force
#      u         =  Vector of dim n_u representing the actuator forces
#
# The term CC(q,dq) can also be decomposed as CC(q,dq) = COR(q,dq) · dq
#
# We shall compute each one of the terms Msys, G, CC, COR, and E.
#
#########################################################################

# -------------------------------------------------------------------------
# Mass matrix of the platform body (Mp)
# -------------------------------------------------------------------------
R1 := RotationMatrix(psi(t),   <1,0,0>): # Rot_x(psi)
R2 := RotationMatrix(theta(t), <0,1,0>): # Rot_y(theta)
R3 := RotationMatrix(phi(t),   <0,0,1>): # Rot_z(phi)

Rot := R1 . R2 . R3:

omBa :=           <diff(psi(t),t), 0, 0>   +
        R1 .      <0, diff(theta(t),t), 0> +
        R1 . R2 . <0, 0, diff(phi(t),t)>   :

Am := <  eval(omBa,[diff(psi(t),t) = 1, diff(theta(t),t) = 0, diff(phi(t),t) = 0]) |
         eval(omBa,[diff(psi(t),t) = 0, diff(theta(t),t) = 1, diff(phi(t),t) = 0]) |
         eval(omBa,[diff(psi(t),t) = 0, diff(theta(t),t) = 0, diff(phi(t),t) = 1])   >:
Jvp := < Matrix(3,3,shape=identity) | Matrix(3,5)>;

Jomp := < Matrix(3,3) | Am | Matrix(3,2) >:

Mp := simplify(mp * Jvp^+ . Jvp + Jomp^+ . Rot . Ip . Rot^+ . Jomp):

# -------------------------------------------------------------------------
# Mass matrix of the ball-and-stick body (Mb)
# -------------------------------------------------------------------------

R4 := RotationMatrix(beta1(t), <1,0,0>);
R5 := RotationMatrix(beta2(t), <0,1,0>):

p := <x(t),y(t),z(t)> :
```

```
r := R4 . R5 . <0,0,l> : # Last coord is a lowercase L!

OGb := p + r:

Db := sJacobian(OGb,q):

Jomb := < Matrix(3,3) | Matrix(3,3) | <<1>,<0>,<0>> | R4(..,2) >;

Rb := R4 . R5:

Mb := simplify(mb * Db^+ . Db + Jomb^+ . Rb . Ib . Rb^+ . Jomb):

# ------------------------------------------------------------------------
# Total mass matrix of the system (Msys)
# ------------------------------------------------------------------------

Msys := simplify(Mp + Mb):

xu_star := [x(t)     =      0,
            y(t)     =      0,
            z(t)     =   -0.3,
            psi(t)   =      0,
            theta(t) =      0,
            phi(t)   =      0,
            beta1(t) =      0,
            beta2(t) =      0,


            dq[1]    =      0,
            dq[2]    =      0,
            dq[3]    =      0,
            dq[4]    =      0,
            dq[5]    =      0,
            dq[6]    =      0,
            dq[7]    =      0,
            dq[8]    =      0,

            u[1]     =      0,
            u[2]     =      0,
            u[3]     =      (mp+mb)*gconst,
            u[4]     =      0,
            u[5]     =      0,
            u[6]     =      0
]:


Msysnum := eval(Msys,[op(xu_star), mp = 4, mb = 3, l=0.2]);
LinearAlgebra[Rank](Msysnum);
evalf(Determinant(Msysnum,method=float));
Msysnum_inverse := LinearAlgebra[MatrixInverse](Msysnum);
LinearAlgebra[Rank](Msysnum_inverse);

# ------------------------------------------------------------------------
# Kinetic energy of the system (T)
# ------------------------------------------------------------------------
```

```
T := simplify(1/2 * dq^+ . Msys . dq):

# ------------------------------------------------------------------------
# Potential energy of the system (U)
# ------------------------------------------------------------------------

grav := <0,0,-gconst>:
U := - mp * grav^+ . p - mb * grav^+ . OGb:

# ------------------------------------------------------------------------
# Equations of motion, assuming no actuation (dyn = 0)
# ------------------------------------------------------------------------
dyn := Vector(n_q):
for i from 1 to n_q do
  dyn[i] := simplify(sdiff(sdiff(T,dq[i]),t)-sdiff(T,q[i])+sdiff(U,q[i])):
end do:

# ------------------------------------------------------------------------
# Gravity term (G)
# ------------------------------------------------------------------------

# Initialize vector of generalized gravity forces
G := Vector(n_q):

# Set the velocity and acceleration functions identically to zero (dq=0 implies ddq=0 too)
dqzero := [seq(dq[i]=0, i=1..n_q)]:

# Evaluate the dynamic equations with all velocities and accelerations set to zero
G := simplify(combine(eval(dyn,dqzero))):

# ------------------------------------------------------------------------
# Coriolis and centrifugal term (CC)
# ------------------------------------------------------------------------
CC := simplify(combine(dyn-Msys.ddq-G)):

# ------------------------------------------------------------------------
# Coriolis matrix (COR)
# ------------------------------------------------------------------------

# Perform the decomposition CC(q,dq) = COR(q,dq)·dq
COR := Matrix(n_q,n_q):
for i from 1 to n_q do
    for j from 1 to n_q do
        for k from 1 to n_q do
            COR[i,j] := COR[i,j]  + dq[k]/2 * ( sdiff(Msys[i,j],q[k]) +
                                                sdiff(Msys[i,k],q[j]) -
                                                sdiff(Msys[k,j],q[i])   ):
        end do:
    end do:
end do:

# Check that CC = COR·dq
DIF := simplify(combine(CC - COR . dq)):

# ------------------------------------------------------------------------
```

```
# Generalized actuation force matrices (Ew and E)
# ------------------------------------------------------------------------

# Compute screw Jacobian
Jscrew := Matrix(6,6):
for j from 1 to 6 do

    dj := Apoints[..,j] - p - Rot . Bpoints[..,j];
    ej := dj / Norm(dj);
    rj := Rot . Bpoints[..,j];

    Jscrew[1..3,j] := ej;
    Jscrew[4..6,j] := rj &x ej;

end do:
Jscrew:

# Mount the D Jacobian
Djac := < < Matrix(3,3,shape = identity) | Matrix(3,3) | Matrix(3,2) >,
          < Matrix(3,3)                  |     Am      | Matrix(3,2) > >:

# Compute Ew(q) and E(q) matrices
Ew := simplify( Djac^+ ):
E  := simplify( Ew . Jscrew ):

# ------------------------------------------------------------------------
# Generalized pertubation Force (Eo) wrench is [Fx,Fy,Fz,Mx,My,Mz]
# ------------------------------------------------------------------------

#Compute Eo(q)
Eo:=simplify(Db^+):

# ------------------------------------------------------------------------
# Term "Ew · u - G" needed for the linearization (wrench)
# ------------------------------------------------------------------------

n_u := 6:
us := Vector(n_u,symbol=u):

JEwuG := sJacobian(Ew.us-G,q):
eval(JEwuG,xu_star):

# ------------------------------------------------------------------------
# Term "E·u-G" needed for the linealitzation (cable tensions)
# ------------------------------------------------------------------------

uts := Vector(n_u,symbol=ut):

u_star1         := eval(us,xu_star):
Jscrewstar:=eval(Jscrew,xu_star);

utstar:=LinearAlgebra[MatrixInverse](Jscrewstar).u_star1;

xut_star:=[x(t)     =     0,
           y(t)     =     0,
```

```
            z(t)      =  -0.3,
            psi(t)    =     0,
            theta(t)  =     0,
            phi(t)    =     0,
            beta1(t)  =     0,
            beta2(t)  =     0,


            dq[1]     =     0,
            dq[2]     =     0,
            dq[3]     =     0,
            dq[4]     =     0,
            dq[5]     =     0,
            dq[6]     =     0,
            dq[7]     =     0,
            dq[8]     =     0,

            ut[1]      =     utstar[1],
            ut[2]      =     utstar[2],
            ut[3]      =     utstar[3],
            ut[4]      =     utstar[4],
            ut[5]      =     utstar[5],
            ut[6]      =     utstar[6]
]:
JEuG := sJacobian(E.uts-G,q):
eval(JEuG,xut_star);

# ------------------------------------------------------------------------
# Term "Ero·tau-G" needed for the linealitzation (for motor coordinates model)
# ------------------------------------------------------------------------

ro:=0.025/2:
Ero:=(1/ro)*E:
utaus := Vector(n_u,symbol=utau):
utaustar:=ro*utstar:
JEtauG := sJacobian(Ero.utaus-G,q):
xutau_star:=[x(t)    =     0,
            y(t)     =     0,
            z(t)     =  -0.3,
            psi(t)   =     0,
            theta(t) =     0,
            phi(t)   =     0,
            beta1(t) =     0,
            beta2(t) =     0,


            dq[1]     =     0,
            dq[2]     =     0,
            dq[3]     =     0,
            dq[4]     =     0,
            dq[5]     =     0,
            dq[6]     =     0,
            dq[7]     =     0,
            dq[8]     =     0,
```

```
           utau[1]     =     utaustar[1],
           utau[2]     =     utaustar[2],
           utau[3]     =     utaustar[3],
           utau[4]     =     utaustar[4],
           utau[5]     =     utaustar[5],
           utau[6]     =     utaustar[6]
]:
eval(JEtauG,xutau_star);

# ------------------------------------------------------------------------
# Now we need the new term JEazuvG to linearize the dynamic model with motor
# coordinates (derivatives with respect to q)
# ------------------------------------------------------------------------
az:=(193*0.836*0.0107)/8.3:

Eaz:= Ero*az:
uvs:=Vector(n_u,symbol=uv):
uvstar:=(8.3/(0.0107*193*0.836))*utaustar;

JEazuvG:=sJacobian(Eaz.uvs-G,q):
xuv_star:=[x(t)     =     0,
           y(t)     =     0,
           z(t)     =  -0.3,
           psi(t)   =     0,
           theta(t) =     0,
           phi(t)   =     0,
           beta1(t) =     0,
           beta2(t) =     0,


           dq[1]    =     0,
           dq[2]    =     0,
           dq[3]    =     0,
           dq[4]    =     0,
           dq[5]    =     0,
           dq[6]    =     0,
           dq[7]    =     0,
           dq[8]    =     0,

           uv[1]     =     uvstar[1],
           uv[2]     =     uvstar[2],
           uv[3]     =     uvstar[3],
           uv[4]     =     uvstar[4],
           uv[5]     =     uvstar[5],
           uv[6]     =     uvstar[6]
]:
eval(JEazuvG,xuv_star);

# ------------------------------------------------------------------------
# Linearization, exact way (wrench):
# ------------------------------------------------------------------------

A_star := << Matrix(n_q,n_q)             | Matrix(n_q,n_q,shape=identity)   >,
          < M_star_inverse . JEwuG_star | - M_star_inverse . COR_star  >>:
B_star := <<  Matrix(n_q,n_u) >,
```

```
          <  M_star_inverse . Ew_star >>:
Co := ControllabilityMatrix(A_star,B_star):
LinearAlgebra[Rank](Co);

# ------------------------------------------------------------------------
# Calculus of Jacobian to pass form qdot to theta (Ji)
# ------------------------------------------------------------------------

ro:=0.025/2;
romat:=ro*LinearAlgebra:-IdentityMatrix(6,6):
roinvmat:=LinearAlgebra[MatrixInverse](romat):
Ae:=Matrix([[LinearAlgebra:-IdentityMatrix(3,3),(LinearAlgebra:-ZeroMatrix(3,3))],
[LinearAlgebra:-ZeroMatrix(3,3),Am]]);
Ji:=Matrix([[roinvmat.Jscrew^+.Ae,LinearAlgebra:-ZeroMatrix(6,2)],
[LinearAlgebra:-ZeroMatrix(2,6),LinearAlgebra:-IdentityMatrix(2,2)]]):

# ------------------------------------------------------------------------
# Computation of Jidot by parts
# ------------------------------------------------------------------------

#Compute lenght matrix
Limat := Matrix(6,6):
for j from 1 to 6 do

    dj := Apoints[..,j] - p - Rot . Bpoints[..,j];
    linvj := 1/Norm(dj);
    Limat[j,j] := linvj;

end do:
Limat:

#time derivative of Limat
Lidotmat:=diff~(Limat,t):
#compute Jacobian(d) Jd
Jd:= Matrix(6,6):
for j from 1 to 6 do

    dj := Apoints[..,j] - p - Rot . Bpoints[..,j];
    rj := Rot . Bpoints[..,j];

    Jd[1..3,j] := dj;
    Jd[4..6,j] := rj &x dj;

end do:
Jd:
#Compute time derivative of Jd matrix
Jddot:=diff~(Jd,t):
#compute time derivative of Ae matrix
Aedot:=diff~(Ae,t):
#first bloc of Jidot matrix
BLOC1:=Lidotmat.Jd^+.Ae+Limat.Jddot^+.Ae+Limat.Jd^+.Aedot:

#Now we build Jidot matrix
#Jidot:=(1/ro)*Matrix([[BLOC1,LinearAlgebra:-ZeroMatrix(6,2)],
[LinearAlgebra:-ZeroMatrix(2,6),LinearAlgebra:-ZeroMatrix(2,2)]]):
```

```
# ------------------------------------------------------------------------
# Export terms to Matlab
# ------------------------------------------------------------------------

# ------------------------------------------------------------------------
# We will do the following substitutions in order to:
#
#     (1) Avoid exporting the time dependence of each function
#     (2) Use proper variable names for the Matlab environment
#
# ------------------------------------------------------------------------

ft_to_f := [
               x(t)             = x[1],
               y(t)             = x[2],
               z(t)             = x[3],
               psi(t)           = x[4],
               theta(t)         = x[5],
               phi(t)           = x[6],
               beta1(t)         = x[7],
               beta2(t)         = x[8],

               diff(x(t),t)     = x[9],
               diff(y(t),t)     = x[10],
               diff(z(t),t)     = x[11],
               diff(psi(t),t)   = x[12],
               diff(theta(t),t) = x[13],
               diff(phi(t),t)   = x[14],
               diff(beta1(t),t) = x[15],
               diff(beta2(t),t) = x[16]

           ]:

# ------------------------------------------------------------------------
#
# Export these objects to Matlab:
#
#     MechEn = Mechanical energy of the system (T+U)
#
#     H      = Mass matrix (n_q x n_q)
#     G      = Generalized gravity force
#     CC     = Generalized Coriolis and centrifugal force
#     COR    = Coriolis matrix (such that CC = C · dq)
#     E      = Actuation matrix (assuming u = cable forces)
#     Ew     = Actuation matrix (assuming u = resultant cable wrench)
#     JEwuG = Jacobian of Ew·u-G wrt q (assuming u = resultant cable wrench)
# ------------------------------------------------------------------------


Set the path
#Set the path where you want the files to be written
files_path:="C:/Users/pgiro/Documents/MATLAB/WriteFile/Maple_Matrices";
   "C:/Users/pgiro/Documents/MATLAB/WriteFile/Maple_Matrices"
```

```
# ------------------------------------------------------------------------
# Export Mass Matrix
# ------------------------------------------------------------------------

Hmatrix := proc(x) end proc:

CDir:=currentdir();
currentdir(files_path);
interface(echo=0);
writeto("Hmatrix.m");
Matlab(Hmatrix);
Matlab(eval(Msys,    ft_to_f),resultname = "Hmatrixreturn"):
printf("end");
writeto(terminal);
currentdir(CDir);

# ------------------------------------------------------------------------
# Export Gravity Term
# ------------------------------------------------------------------------

Gmatrix := proc(x) end proc:

CDir:=currentdir();
currentdir(files_path);
interface(echo=0);
writeto("Gmatrix.m");
Matlab(Gmatrix);
Matlab(eval(Matrix(G), ft_to_f),resultname = "Gmatrixreturn"):
printf("end");
writeto(terminal);
currentdir(CDir);

# ------------------------------------------------------------------------
# Export COR Matrix
# ------------------------------------------------------------------------

CORmatrix := proc(x) end proc:

CDir:=currentdir();
currentdir(files_path);
interface(echo=0);
writeto("CORmatrix.m");
Matlab(CORmatrix);
Matlab(eval(COR,     ft_to_f), resultname="CORmatrixreturn"):
printf("end");
writeto(terminal);
currentdir(CDir);

# ------------------------------------------------------------------------
# Export E Matrix
# ------------------------------------------------------------------------

Ematrix := proc(x) end proc:

CDir:=currentdir();
```

```
currentdir(files_path);
interface(echo=0);
writeto("Ematrix.m");
Matlab(Ematrix);
Matlab(eval(E,       ft_to_f), resultname="Ematrixreturn"):
printf("end");
writeto(terminal);
currentdir(CDir);

# ------------------------------------------------------------------------
# Export Jacobian to linearize the system with (cable tensions actuation)
# ------------------------------------------------------------------------

JEazuvGmatrix := proc(x,uv) end proc:

CDir:=currentdir();
currentdir(files_path);
interface(echo=0);
writeto("JEazuvGmatrix.m");
Matlab(JEazuvGmatrix);
Matlab(eval(JEazuvG,    ft_to_f), resultname = "JEazuvGmatrixreturn"):
printf("end");
writeto(terminal);
currentdir(CDir);

# ------------------------------------------------------------------------
# Export Ji  to Matlab:
# ------------------------------------------------------------------------

Jimatrix := proc(x) end proc:

CDir:=currentdir();
currentdir(files_path);
interface(echo=0);
writeto("Jimatrix.m");
Matlab(Jimatrix);
Matlab(eval(Ji,    ft_to_f), resultname = "Jimatrixreturn"):
printf("end");
writeto(terminal);
currentdir(CDir);

# ------------------------------------------------------------------------
# Export terms to build Jidot matrix
# ------------------------------------------------------------------------

Jidotmatrix:= proc(x,m) end proc:

CDir:=currentdir();
currentdir(files_path);
interface(echo=0);
writeto("Jidotmatrix.m");
Matlab(Jidotmatrix):
Matlab(eval(Lidotmat,    ft_to_f), resultname = "Lidotmat"):
Matlab(eval(Jd,    ft_to_f), resultname = "Jd"):
Matlab(eval(Ae,    ft_to_f), resultname = "Ae"):
```

```
Matlab(eval(Limat,     ft_to_f), resultname = "Limat"):
Matlab(eval(Jddot,     ft_to_f), resultname = "Jddot"):
Matlab(eval(Aedot,     ft_to_f), resultname = "Aedot"):
printf("Jidotmatrixreturn=(1/m.ro)*[Lidotmat*Jd'*Ae+Limat*Jddot'*Ae+Limat*Jd'*Aedot,
zeros(6,2);
        zeros(2,6),                                                    zeros(2,2)];"):
printf("\n");
printf("end");
writeto(terminal);
currentdir(CDir);
```

C.3 Programes específics del model mecànic

C.3.1 MainDMMXcord

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Main_DM_MXcord.m
%
% Purpose:
%
%    This is the main script of the simulation used for the simulation of the
%    mechanical model, and for the design of its corresponding control law
%
% User-specified parameters:
%
%    Flags to specify various parameters of the simulation
%    Simulation time length
%    Integration maximum step size
%    Geometric and dynamic parameters
%    Define point of linearitzation
%    Bryson rule tuning constants
%
% Output:
%
%    Plots of the simulation results
%    Video of the simulation (Optional)
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%% Inputs %%%%%%%%%%%%%%%%%%%%%%%%

Flag='ODE15'; % Flag to define what you want the script to do
%               (Flag == ODE15)> integrate with ode15s
%               (Flag == VIDEO)> output a video of an animation of the
%               dynamic model
```

```
%               (Flag ==FinDif)> output AlinF and BlinF matrices to build
%               linealized model through FinDif

Flag00='NO'; % Flag to set if you want to simulate with %%CONTROL LAW%% or not
%               (Flag == YES)> simulation with %%CONTROL LAW%%
%               (Flag == NO)> simulation wthout %%CONTROL LAW%%

Flag01='PACK'; % Set how you want the plots to be displayed
%               (Flag01=='PACK')> set theta plots in packets
%               (Flag01=='FULL')> set theta plots individualy

Flag1='YES'; % Secondary flag to set if you want to monitorize or not x state
%              (q domain)
%                 (Flag2 == YES)> monitorize x
%                 (Flag2 == NO)> omit monitorizing of x

Flag2='YES'; % Secondary flag to set if you want to monitorize or not the
%              actuation cable tensions==> motor torques
%                 (Flag2 == YES)> monitorize cable tensions ==> motor
%                 torques
%                 (Flag2 == NO)> omit monitorizing of cable tensions ==>
%                 motor torques

global Flag3;
Flag3='NO'; % Secondary flag to set if you want parcial monitorizing of
%             xtheta/xq/xbarra
%                 (Flag3 == YES)>  Parcial monitorizing ENABLED
%                 (Flag3 == NO)> omit parcial monitorizing


if strcmp(Flag00, 'YES')
    Flag4='NO';% Secondary flag to set if you want external PERTRUB. or not
%                 (Flag4 == YES)>  external pertrub. ENABLED
%                 (Flag4 == NO)> omit external pertrub.
end

tf = 5;    % Final simulation time
h = 0.001; % Number of samples within final vectors (times and states)
opts = odeset('MaxStep',1e-3,'RelTol',1e-5,'AbsTol',1e-7); % max integration
                                                           % time step

% Gravity constant [m/s^2]

m.g=9.8;
```

```
Dyn_Vector=from_file('Dynamic_Data.txt');

m.ls=Dyn_Vector(1,1);    % Distance from the edge of the pendulum to G_pole,[m]
m.mp=Dyn_Vector(2,1);    % Mass of the platform [kg]
m.mb=Dyn_Vector(3,1);    % Mass of the ball+stick [kg]

m.ITpxx=Dyn_Vector(4,1); % XX axis inercia of the platform [kg*m^2]
m.ITpyy=Dyn_Vector(5,1); % YY axis inercia of the platform [kg*m^2]
m.ITpzz=Dyn_Vector(6,1); % ZZ axis inercia of the platform [kg*m^2]

m.ITbxx=Dyn_Vector(7,1); % XX axis inercia of the platform [kg*m^2]
m.ITbyy=Dyn_Vector(8,1); % YY axis inercia of the platform [kg*m^2]
m.ITbzz=Dyn_Vector(9,1); % ZZ axis inercia of the platform [kg*m^2]


% A and B matrices of the hexapole geometry, all distances in [m]

m.B=1e-3*[0,-89.15,0;0,-89.15,0;77.21,44.57,0;77.21,44.57,0; ...
                                    -77.21,44.57,0;-77.21,44.57,0]';
m.A=1e-3*[-231.62,-136.18,0;231.62,-136.18,0;233.74,-132.5,0; ...
                  2.13,268.67,0;-2.13,268.67,0;-233.74,-132.5,0]';

% Cable lenghts in home position

m.lc=-1*[0.381915756810321; 0.381915756810321; 0.381910232646364; ...
    0.381913362426611;0.381913362426611;0.381910232646364];


% Defining xstar vector

xstar(1,1)=0;
xstar(2,1)=0;
xstar(3,1)=-0.300;
xstar(4,1)=0;
xstar(5,1)=0;
xstar(6,1)=0;
xstar(7,1)=0;
xstar(8,1)=0;

xstar(9,1)=0;
xstar(10,1)=0;
xstar(11,1)=0;
xstar(12,1)=0;
xstar(13,1)=0;
xstar(14,1)=0;
```

```
xstar(15,1)=0;
xstar(16,1)=0;


% Bryson rule

CTE1=200; %contant for position
CTE2=1e9; %constant for speed
CTE3=(1.5e12); %contant for actuation 1e12


Q1=  [1/(deg2rad(360))^2*eye(6),             zeros(6,2);
          zeros(2,6),                 1/(deg2rad(5))^2*eye(2)];

Q2=[(1/(5.6)^2)*eye(6),             zeros(6,2);
          zeros(2,6),                 (1/10)*(1/(deg2rad(5))^2*eye(2))];

Q=[CTE1*Q1,       zeros(8,8);
    zeros(8,8),     CTE2*Q2];

R=CTE3*(1/(6.5*m.ro_utaustar(1))^2)*eye(6);



xthetastar(1:8,1)=[0,0,0,0,0,0,xstar(7),xstar(8)]';
xthetastar(9:16,1)=zeros(8,1);

[Alin,Blin]=Lineal_HexaMxcord_Analitic( xstar, utstar, utaustar, m, 'JEtauG');

[K,SolRic,Eigvals]=lqr(Alin,Blin,Q,R);


% Calculating our utstar (the tensions of the cables that are in
% our home state)

wrenchstar=[0,0,(m.mp+m.mb)*m.g,0,0,0]';

% Calculating Jscrew home position

Jscrewstar=Calc_Jscrew_MXcord(xstar);

% Defining ustar vector

utstar=inv(Jscrewstar)*wrenchstar;
utaustar=(m.ro)*utstar;
```

```
disp(utaustar)

% Rotation matrices

R1 = @(x) [1 0 0; 0 cos(x(4)) -sin(x(4)); 0 sin(x(4)) cos(x(4));]; % Rot_x(psi)
R2 = @(x) [cos(x(5)) 0 sin(x(5)); 0 1 0; -sin(x(5)) 0 cos(x(5));]; % Rot_y(theta)
R3 = @(x) [cos(x(6)) -sin(x(6)) 0; sin(x(6)) cos(x(6)) 0; 0 0 1;]; % Rot_z(phi)
R4 = @(x) [1 0 0; 0 cos(x(7)) -sin(x(7)); 0 sin(x(7)) cos(x(7));]; % Rot_x(beta1)
R5 = @(x) [cos(x(8)) 0 sin(x(8)); 0 1 0; -sin(x(8)) 0 cos(x(8));]; % Rot_t(beta2)

% Now that the main structures and parameters have been set we begin the part
% of the code to simulate the system

switch Flag
    case 'ODE15'

        % Simulate with ode15

        % Initial conditions

        x0=[0,0,-0.3,0,0,0,deg2rad(160),deg2rad(0),0,0,0,0,0,0,0,0]';

        global xq % Vector to make interations of newton easier
        xq=x0;

        % Time

        t = 0:h:tf;

        switch Flag3
            case 'YES'
                global xthetaplot   % State in theta domain
                xthetaplot(16,1)=0;

                global xbarrax      % State error for the control law
                xbarrax(16,1)=0;

                global xqplot
                xqplot=x0;
            otherwise
                disp('omit partial monitorizing')
        end

        xtheta0=x_to_xtheta(x0,m);
        display(xtheta0);
```

```
pause(3);


switch Flag00
    case 'NO'
        dxdt= @(t,xtheta) xdot_MXcord(t, xtheta, m, utaustar);
        [times,states]=ode15s(dxdt,t,xtheta0);
    case 'YES'
        if strcmp(Flag4, 'YES')
            dxdt= @(t,xtheta) xdot_MXcord(t, xtheta, m, utaustar, ...
                                xthetastar,K,@uo_pert_wrench_MXcord);
            [times,states]=ode15s(dxdt,t,xtheta0,opts);
        else
            dxdt= @(t,xtheta) xdot_MXcord(t, xtheta, m, utaustar, ...
                                                    xthetastar,K);
            [times,states]=ode15s(dxdt,t,xtheta0);
        end
    otherwise
        disp('Flag00 can not take this value');
end


switch Flag01
    case 'FULL'
        figure;
        plot(times,states(:,1))
        xlabel('t(s)'),ylabel('theta1(t)'),title('State')

        figure;
        plot(times,states(:,2))
        xlabel('t(s)'),ylabel('theta2(t)'),title('State')

        figure;
        plot(times,states(:,3))
        xlabel('t(s)'),ylabel('theta3(t)'),title('State')

        figure;
        plot(times,states(:,4))
        xlabel('t(s)'),ylabel('theta4(t)'),title('State')

        figure;
        plot(times,states(:,5))
        xlabel('t(s)'),ylabel('theta5(t)'),title('State')

        figure;
```

```
        plot(times,states(:,6))
        xlabel('t(s)'),ylabel('theta6(t)'),title('State')

        figure;
        plot(times,rad2deg(states(:,7)))
        xlabel('t(s)'),ylabel('beta1(t) [deg]'),title('State')

        figure;
        plot(times,rad2deg(states(:,8)))
        xlabel('t(s)'),ylabel('beta2(t) [deg]'),title('State')

        figure;
        plot(times,states(:,9))
        xlabel('t(s)'),ylabel('theta1dot(t)'),title('State')

        figure;
        plot(times,states(:,10))
        xlabel('t(s)'),ylabel('theta2dot(t)'),title('State')

        figure;
        plot(times,states(:,11))
        xlabel('t(s)'),ylabel('theta3dot(t)'),title('State')

        figure;
        plot(times,states(:,12))
        xlabel('t(s)'),ylabel('theta4dot(t)'),title('State')

        figure;
        plot(times,states(:,13))
        xlabel('t(s)'),ylabel('theta5dot(t)'),title('State')

        figure;
        plot(times,states(:,14))
        xlabel('t(s)'),ylabel('theta6dot(t)'),title('State')

        figure;
        plot(times,states(:,15))
        xlabel('t(s)'),ylabel('beta1dot(t)'),title('State')

        figure;
        plot(times,states(:,16))
        xlabel('t(s)'),ylabel('beta2dot(t)'),title('State')

    case 'PACK'
```

```
        figure;
        box on
        hold on
        plot(times,rad2deg(states(:,1)),'LineWidth',3)
        plot(times,rad2deg(states(:,2)),'LineWidth',3)
        plot(times,rad2deg(states(:,3)),'LineWidth',3)
        plot(times,rad2deg(states(:,4)),'LineWidth',3)
        plot(times,rad2deg(states(:,5)),'LineWidth',3)
        plot(times,rad2deg(states(:,6)),'LineWidth',3)

        xlabel('t(s)'),ylabel('theta1-theta6 [deg]'), ...
                                    title('Motor Angles')
        hold off

        figure;
        box on
        hold on
        plot(times,rad2deg(states(:,7)),'r','LineWidth',3)
        plot(times,rad2deg(states(:,8)),'g','LineWidth',3)
        xlabel('t(s)'),ylabel('beta1=red beta2=green [deg]'), ...
                                        title('Pendulum Angles')
        hold off

        figure;
        box on
        hold on
        plot(times,states(:,9),'LineWidth',3)
        plot(times,states(:,10),'LineWidth',3)
        plot(times,states(:,11),'LineWidth',3)
        plot(times,states(:,12),'LineWidth',3)
        plot(times,states(:,13),'LineWidth',3)
        plot(times,states(:,14),'LineWidth',3)
        xlabel('t(s)'),ylabel('theta1dot-theta6dot [rad/s]'), ...
                                          title('Motor Speeds')
        hold off

        figure;
        box on
        hold on
        plot(times,states(:,15),'r','LineWidth',3)
        plot(times,states(:,16),'g','LineWidth',3)
        xlabel('t(s)'),ylabel('beta1dot=red beta2dot=green [rad/s]' ...
                                          ),title('Pendulum Speeds')
        hold off
    otherwise
```

```
        disp('this Flag01 does not exist')
end

% Set if you want to monitorize or not x state (q domain)

switch Flag1
    case 'YES'
        % Building statesx
        statesx(1,:)=xtheta_to_xq(states(1,:)',xstar,m)';
        for i=2:max(size(states))
            statesx(i,:)=xtheta_to_xq(states(i,:)',statesx(i-1,:)',m)';
        end

% Here we start drawing the results of the simulation

        switch Flag01
            case 'FULL'
                figure;
                plot(times,statesx(:,1))
                xlabel('t(s)'),ylabel('x(t)'),title('State')

                figure;
                plot(times,statesx(:,2))
                xlabel('t(s)'),ylabel('y(t)'),title('State')

                figure;
                plot(times,statesx(:,3))
                xlabel('t(s)'),ylabel('z(t)'),title('State')

                figure;
                plot(times,statesx(:,4))
                xlabel('t(s)'),ylabel('psi(t)'),title('State')

                figure;
                plot(times,statesx(:,5))
                xlabel('t(s)'),ylabel('theta(t)'),title('State')

                figure;
                plot(times,statesx(:,6))
                xlabel('t(s)'),ylabel('phi(t)'),title('State')

                figure;
                plot(times,statesx(:,9))
                xlabel('t(s)'),ylabel('xdot(t)'),title('State')
```

```
        figure;
        plot(times,statesx(:,10))
        xlabel('t(s)'),ylabel('ydot(t)'),title('State')

        figure;
        plot(times,statesx(:,11))
        xlabel('t(s)'),ylabel('zdot(t)'),title('State')

        figure;
        plot(times,statesx(:,12))
        xlabel('t(s)'),ylabel('psidot(t)'),title('State')

        figure;
        plot(times,statesx(:,13))
        xlabel('t(s)'),ylabel('thetadot(t)'),title('State')

        figure;
        plot(times,statesx(:,14))
        xlabel('t(s)'),ylabel('phidot(t)'),title('State')

    case 'PACK'

       figure;
       box on
       hold on
       plot(times,statesx(:,1),'r','LineWidth',3)
       plot(times,statesx(:,2),'g','LineWidth',3)
       plot(times,statesx(:,3),'b','LineWidth',3)
       xlabel('t(s)'),ylabel('x=red y=green z=blue [m]'), ...
                                  title('Platform position')
       hold off


       figure;
       box on
       hold on
       plot(times,rad2deg(statesx(:,4)),'r','LineWidth',3)
       plot(times,rad2deg(statesx(:,5)),'b','LineWidth',3)
       plot(times,rad2deg(statesx(:,6)),'g','LineWidth',3)
       xlabel('t(s)'),ylabel( ...
           'psi=red theta=blue phi=green [deg]'),title( ...
                                    'Platform orientation')
       hold off

       figure;
```

```
        box on
        hold on
        plot(times,statesx(:,9),'r','LineWidth',3)
        plot(times,statesx(:,10),'b','LineWidth',3)
        plot(times,statesx(:,11),'g','LineWidth',3)
        xlabel('t(s)'), ...
            ylabel('xdot=red ydot=blue zdot=green [m/s]'), ...
                                        title('Platform speed')
        hold off

        figure;
        box on
        hold on
        plot(times,statesx(:,12),'r','LineWidth',3)
        plot(times,statesx(:,13),'b','LineWidth',3)
        plot(times,statesx(:,14),'g','LineWidth',3)
        xlabel('t(s)'), ylabel( ...
            'psidot=red thetadot=blue phidot=green [rad/s]'), ...
                                title('Platform rotation speed')
        hold off

   otherwise
        disp('Flag01 can not take this value')
end

% Ploting Gtot

figure;
box on
hold on
Gp(3,max(size(statesx)))=0;
Gb(3,max(size(statesx)))=0;
Gtot(3,max(size(statesx)))=0;
for i=1:max(size(statesx))
    Gp(1,i)=statesx(i,1);
    Gp(2,i)=statesx(i,2);
    Gp(3,i)=statesx(i,3);
end
for i=1:max(size(Gp))
    R4i=R4(statesx(i,:));
    R5i=R5(statesx(i,:));
    Gb(:,i)=Gp(:,i)+R4i*R5i*([0, 0, m.ls]');
end
for i=1:max(size(Gp))
    Gtot(:,i)=1/(m.mp+m.mb)*(m.mp*Gp(:,i)+m.mb*Gb(:,i));
```

```
        end
        plot(times,Gtot(1,:),'r','LineWidth',3)
        plot(times,Gtot(2,:),'b','LineWidth',3)
        plot(times,Gtot(3,:),'g','LineWidth',3)
        xlabel('t(s)'),ylabel('Gtot [x=red,y=blue,z=green]'), ...
                                            title('Gtot position')
        hold off

    otherwise
        disp('this input Flag1 does not exist')
end

% Monitorize motor torques

switch Flag2
    case 'YES'
        utauArray(6,max(size(times)))=0;

        switch Flag00

            case 'YES'
                for i=1:max(size(times))
                    utauArray(:,i)=u_function_MXcord(times(i), ...
                             states(i,:)',m,utaustar,xthetastar,K);
                end

            case 'NO'
                for i=1:max(size(times))
                    utauArray(:,i)=u_function_MXcord(times(i), ...
                                        states(i,:)',m,utaustar);
                end

            otherwise
                disp('This Flag00 does not exist')
        end

        figure;
        box on
        hold on
        plot(times,utauArray(1,:),'r','LineWidth',3)
        plot(times,utauArray(2,:),'b','LineWidth',3)
        plot(times,utauArray(3,:),'g','LineWidth',3)
        plot(times,utauArray(4,:),'y','LineWidth',3)
        plot(times,utauArray(5,:),'k','LineWidth',3)
        plot(times,utauArray(6,:),'m','LineWidth',3)
```

```
            xlabel('t(s)'),ylabel('tau1—tau6 [Nm]'), ...
                      title('Monitorizing Motor Torques')
            hold off

        case 'NO'
            disp('omitting monitorizing of torques')
        otherwise
            disp('this input Flag2 does not exist')
     end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

case 'VIDEO'

    time=cputime;
     % Animation
     % h is the sampling time
     % n is the scaling factor in order not to plot with the same step
     % than during the integration with ode45
     fs=30;
     n=round(1/(fs*h));

    % Set up the movie.
    writerObj = VideoWriter('hexapole_thetaway','MPEG—4'); % Name it.
    writerObj.FrameRate = fs; % How many frames per second.
    open(writerObj);


     for i = 1:n:length(times)
                q.x=statesx(i,1);
                q.y=statesx(i,2);
                q.z=statesx(i,3);
                q.psi=statesx(i,4);
                q.theta=statesx(i,5);
                q.phi=statesx(i,6);
                q.beta1=statesx(i,7);
                q.beta2=statesx(i,8);

                elapsed = cputime—time;
                if elapsed>200
                    disp(elapsed);
                    disp('took too long to generate the video')
                    break
                else
                    Draw_DM_MXcord(m,q)
```

```
                        % 'gcf' can handle if you zoom in to take a movie.
                        frame = getframe(gcf);
                        writeVideo(writerObj, frame);
                    end

        end
        close(writerObj); % Saves the movie.

    otherwise
        disp('This flag input does not exist');
end
```

C.3.2 LinealHexaMxcordAnalitic

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Lineal_HexaMxcord_Analitic.m
%
% Purpose:
%
%    This function builds the A and B matrices of the linearized model, using the
%    analitic way instead of finite differences
%
% User-specified parameters:
%
%    xstar:     State vector in vertical equilibrium (point of linearitzation)
%    utaustar:  Actuation vector (motor torques) in vertical equilibrium
%               state
%    m:         Structural parameter containing all geometric and dynamic
%               variables that are used to define the model (such as gravity
%               constant g, or anchor points Ai, Bi, and the moments of inertia)
%
% Output:
%
%    A:         Matrix that used to linearize the system (multiplies error of
%               state) A(x-xstar)
%
%    B:         Matrix that used to linearize the system (multiplies error of
%               actuation) B(u-ustar)
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [ A,B ] = Lineal_HexaMxcord_Analitic( xstar, utaustar, m )

    % Main matrices of our dynamic model needed for the linearization
    H=Hmatrix(xstar);
```

```
    Ji=Jimatrix(xstar);


    E=(1/m.ro)*Ematrix(xstar);
    COR=CORmatrix(xstar);
    Jidot=Jidotmatrix(xstar,m);

    invJi=pinv(Ji);

    Mtheta=H*invJi;
    Ctheta=(COR-H*invJi*Jidot)*invJi;

    JEtauG=JEtauGmatrix(xstar,utaustar);
    A21=inv(Mtheta)*JEtauG*invJi;
    A22=-inv(Mtheta)*Ctheta;


    % Compute A matrix (exact way)
    A=[zeros(8),eye(8);A21,A22];

    % Compute B matix (exact way)
    B=[zeros(8,6);(Mtheta)\E];

end
```

C.3.3 xdotMXcord

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% xdot_MXcord
%
% Purpose:
%
%     This function evaluates f(xtheta,utau) in the mechanical model
%
%               xthetadot = f(xtheta, utau)
%
%     of the Hexapole. This function is meant to be passed as a parameter to the
%     ode45 or ode15s simulation routines of Matlab. The value of utau is assumed
%     to be given by the optimal control law
%
%               utau = utaustar - K * ( xtheta - xthetastar )
%
%     defined by the input parameters. The function also allows the evaluation of
%     f(xtheta, utau) in the presence of perturbation forces applied to the
%     center of mass of the pole.
%
% Input parameters:
%
%     t:          The instant of time for which the evaluation of f has to be
%                 performed. This parameter is used to compute perturbation
%                 forces when present, which are time-dependent.
%     xtheta:     The state at which we evaluate f(xtheta, utau).
%     m:          A structure with all geometric and dynamic parameters of the
%                 Hexapole.
%     utaustar:   The steady-state action needed to keep the hexapole in
%                 equilibrium.
%     xthetastar: The equilibrium state desired for the Hexapole.
%     K:          The gain matrix of the control law.
%     u0_handle:  A handle to a function that produces non-modelled perturbation
%                 forces applied to the pole.
%
% Output parameters:
%
%     xthetadot:  The value of f(xtheta, utau).
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [ xthetadot ] = xdot_MXcord( t, xtheta, m, utaustar, xthetastar, K,
    uo_handle)
```

```
global Flag3;

global xq

p0=xq(1:3);
rpy0=xq(4:6);

[ pf,rpyf ] = FKfsolve( xtheta,p0,rpy0,m );

x(1:8,1)=[pf;rpyf;xtheta(7:8)];

H=Hmatrix(x);
G=Gvector(x);
E=Ematrix(x);
Ji=Jimatrix(x);

x(9:16,1)=Ji\xtheta(9:16);

switch Flag3
    case 'YES'
        global xthetaplot;
        xthetaplot=horzcat(xthetaplot,xtheta);

        global xqplot;
        xqplot=horzcat(xqplot,x);
end

xq=x;

COR=CORmatrix(x);
Jidot=Jidotmatrix(x,m);


invJi=pinv(Ji);

Mtheta=H*invJi;
Ctheta=(COR-H*invJi*Jidot)*invJi;


switch nargin
    case 4
        u=u_function_MXcord(t,xtheta,m,utaustar);
        thetadot=xtheta(9:16);
        thetadotdot=Mtheta\(E*(1/m.ro)*u-Ctheta*thetadot-G);
```

```
            disp('function evaluated at time:');
            disp(t);
            xthetadot=[thetadot; thetadotdot];

        case 6
            u=u_function_MXcord(t,xtheta,m,utaustar,xthetastar,K);
            thetadot=xtheta(9:16);
            thetadotdot=Mtheta\(E*(1/m.ro)*u-Ctheta*thetadot-G);

            disp('function evaluated at time:');
            disp(t);

            xthetadot=[thetadot; thetadotdot];

        case 7

            Eo=Eomatrix(x);

            u=u_function_MXcord(t,xtheta,m,utaustar,xthetastar,K);
            uo=uo_handle(t);
            thetadot=xtheta(9:16);
            thetadotdot=Mtheta\((1/m.ro)*E*u+Eo*uo-Ctheta*thetadot-G);
            disp('function evaluated at time:');
            disp(t);

            xthetadot=[thetadot; thetadotdot];


        otherwise
            disp('error in number of input variables');
    end

end
```

C.3.4 uopertwrenchMXcord

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% uo_pert_wrench_MXcord
%
% Purpose:
%
%     The aim of this function is to introduce unmodelled force perturbations
%     in a particular instant of time
%
% Input parameters:
```

```
%
%     t: Evaluation time of the function (the purpose of this input is to know
%        in which instant of time the perturbation has to be introduced)
%
% Output parameters:
%
%     uo: Vector of perturbation force
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [ uo ] = uo_pert_wrench_MXcord( t )

    if  t<5.8
        uo=[0,0,0]';

    elseif (t>=5.8 && t<5.9)
        uo=[0.5, 0, 0]';

    elseif (t>=9.5 && t<9.6)
        uo=[0, 0.5, 0]';

    elseif (t>=13 && t<13.1)
        uo=[0, 0, -2]';

    else
        uo=[0,0,0]';

    end

end
```

C.4 Programes específics del model electromecànic

C.4.1 MainDMHexaMX28

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Main_DM_HexaMX28.m
%
% Purpose:
%
%    This is the main script of the simulation used to simulate the
%    electromechanical model, and to design its associated control law.
%
% User-specified parameters:
%
```

```
%    Flags to specify various parameters of the simulation
%    Simulation time length
%    Integration maximum step size
%    Geometric and dynamic parameters
%    Define point of linearitzation
%    Bryson rule tunning constants
%
% Output:
%
%    Plots of the simulation results
%    Video of the simulation (Optional)
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%% Inputs %%%%%%%%%%%%%%%%%%%%%%%

Flag='NEW_VIDEO'; % Flag to define what you want the script to do
%                 (Flag == ODE45)> integrate with ode45
%                 (Flag == VIDEO)> output a video of an animation of the
%                 dynamic model (Only works if Flag1 is set to 'YES')
%                 (Flag == NEW_VIDEO)> output a video of an animation of the
%                  dynamic model with STL figure (Only works if Flag1 is set to '
    YES')
%                 (Flag ==FinDif)> output AlinF and BlinF matrices to build
%                 linealized model through FinDif

Flag00='YES';     % Flag to set if you want to simulate with %%CONTROL LAW%% or
    not
%                 (Flag == YES)> simulation with %%CONTROL LAW%%
%                 (Flag == NO)> simulation wthout %%CONTROL LAW%%

Flag01='PACK';
%                 (Flag01=='PACK')> set theta plots in packets
%                 (Flag01=='FULL')> set theta plots individualy

Flag1='YES';      % Secondary flag to set if you want to monitorize or
%                  not x state (q domain)
%                 (Flag2 == YES)> monitorize x
%                 (Flag2 == NO)> omit monitorizing of x

Flag2='YES';      % Secondary flag to set if you want to monitorize or not
%                 the actuation voltages
%                 (Flag2 == YES)> monitorize voltages
%                 (Flag2 == NO)> omit monitorizing voltages
```

```
global Flag3
Flag3='YES';      % Secondary flag to set if you want to monitorize or not the
%                 actuation cable tensions and motor accelerations
%                 (Flag2 == YES)> monitorize cable tensions and accelerations
%                 (Flag2 == NO)> omit monitorizing cable tensions and
    accelerations

if strcmp(Flag00, 'YES')
    Flag4='YES';  % Secondary flag to set if you want external PERTRUB. or not
%                 (Flag4 == YES)>  external pertrub. ENABLED
%                 (Flag4 == NO)> omit external pertrub.
end

% Final simulation time

tf = 15;

% Number of samples within final vectors (times and states)

h = 0.001;

% Max integration time step

opts = odeset('MaxStep',1e-3);


% Reading data from file

m.g=9.8;                % Gravity constant [m/s^2]

Dyn_Vector=from_file('Dynamic_Data.txt');

m.ls=Dyn_Vector(1,1);    % Length of the stick that links the ball with the
    platform,[m]
m.mp=Dyn_Vector(2,1);    % Mass of the platform [kg]
m.mb=Dyn_Vector(3,1);    % Mass of the ball+stick [kg]

m.ITpxx=Dyn_Vector(4,1); %XX axis inercia of the platform [kg*m^2]
m.ITpyy=Dyn_Vector(5,1); %YY axis inercia of the platform [kg*m^2]
m.ITpzz=Dyn_Vector(6,1); %ZZ axis inercia of the platform [kg*m^2]

m.ITbxx=Dyn_Vector(7,1); %XX axis inercia of the platform [kg*m^2]
m.ITbyy=Dyn_Vector(8,1); %YY axis inercia of the platform [kg*m^2]
m.ITbzz=Dyn_Vector(9,1); %ZZ axis inercia of the platform [kg*m^2]
```

```
% A and B matrices of the hexapole geometry, all distances in [m]

m.B=1e-3*[0,-89.15,0;0,-89.15,0;77.21,44.57,0;77.21,44.57,0;-77.21,44.57,0; ...
                                                            -77.21,44.57,0]';
m.A=1e-3*[-231.62,-136.18,0;231.62,-136.18,0;233.74,-132.5,0;2.13,268.67,0; ...
                                                  -2.13,268.67,0;-233.74,-132.5,0]';


% Defining motor-related constants

m.ro=0.025/2;   % Radius of the pulley fixed in MX-28 load axis [m]

m.Vs=12;        % Source voltage [V]


m.Res=8.3;      % Resistance [om]
m.L=2.03e-3;    % Inductance [H]
m.eff=0.836;    % Gear efficiency


m.N=193;        % Gearbox ratio
m.Kom=93.0959;  % Motor speed constant [rad/V]
m.Kt=0.0107;    % Torque constant [N*m/A]
m.Jm=8.68e-8;   % Inertia value at motor axis [kg*m^2]
m.Jl=2.896e-7;  % Inercia volue at load axis [kg*m^2]
m.bm=8.87e-8;   % Friction value at motor axis [N*m*s]
m.EN=m.N*m.eff; % Constant that relates tauload with taumotor


% Equation of MX-28 constants

m.a=(m.EN*m.Kt)/m.Res;
m.b=-(m.EN*(m.Kt)^2*m.N)/m.Res-m.EN*m.bm*m.N;
m.c=-m.EN*m.N*m.Jm;


% Vector of state in equilibrium position

xstar(1,1)=0;
xstar(2,1)=0;
xstar(3,1)=-0.300;
xstar(4,1)=0;
xstar(5,1)=0;
xstar(6,1)=0;
```

```
xstar(7,1)=0;
xstar(8,1)=0;

xstar(9,1)=0;
xstar(10,1)=0;
xstar(11,1)=0;
xstar(12,1)=0;
xstar(13,1)=0;
xstar(14,1)=0;
xstar(15,1)=0;
xstar(16,1)=0;

% Cable lenghts

 m.lc=IKsolve(xstar,m);

% Calculating our utstar (the tensions of the cables that are in
% our home state)
wrenchstar=[0,0,(m.mp+m.mb)*m.g,0,0,0]';

% Calculating Jscrew home position

Jscrewstar=Calc_Jscrew_HexaMX28(xstar);

% Defining utaustar vector

utstar=inv(Jscrewstar)*wrenchstar;
utaustar=(m.ro)*utstar;

% Now we build voltages in our home position vector

uvstar=(m.Res/(m.Kt*m.EN))*utaustar;
disp(uvstar)

% Bryson Rule

CTE1=1;    % Constant to adjust position terms of Q matrix
CTE2=1;    % Constant to adjust velocity terms of Q matrix
CTE3=1100; % Constant to adjust actuation terms of R matrix

Q1= [ 10/(deg2rad(60))^2*eye(6),  zeros(6,2)                ; ...
         zeros(2,6)           ,  1/(deg2rad(5))^2 * eye(2) ];


Q2= [ (1/50)*((1/((5.6)^2))*eye(6)) ,   zeros(6,2)               ; ...
```

```
        zeros(2,6)              ,   (1/10)*(1/(deg2rad(5))^2 * eye(2)) ];


Q = [ CTE1*Q1,          zeros(8,8)   ; ...
      zeros(8,8),  CTE2*Q2      ];

R=CTE3*(1/(10)^2)*eye(6);

xthetastar(1:8,1)=[0,0,0,0,0,0,xstar(7),xstar(8)]';
xthetastar(9:16,1)=zeros(8,1);

[Alin,Blin]=Lineal_HexaMX28_Analitic( xstar, uvstar, m);

[K,SolRic,Eigvals]=lqr(Alin,Blin,Q,R);

R1 = @(x) [1 0 0; 0 cos(x(4)) -sin(x(4)); 0 sin(x(4)) cos(x(4));]; % Rot_x(psi)
R2 = @(x) [cos(x(5)) 0 sin(x(5)); 0 1 0; -sin(x(5)) 0 cos(x(5));]; % Rot_y(theta)
R3 = @(x) [cos(x(6)) -sin(x(6)) 0; sin(x(6)) cos(x(6)) 0; 0 0 1;]; % Rot_z(phi)
R4 = @(x) [1 0 0; 0 cos(x(7)) -sin(x(7)); 0 sin(x(7)) cos(x(7));]; % Rot_x(beta1)
R5 = @(x) [cos(x(8)) 0 sin(x(8)); 0 1 0; -sin(x(8)) 0 cos(x(8));]; % Rot_t(beta2)

switch Flag
    case 'ODE45'

        % Simulate with ode45

        % Initial conditions

        x0=[0,0,-0.3,0,0,0,deg2rad(1),deg2rad(1),0,0,0,0,0,0,0,0]';

        global xq
        xq=x0;

        % Time
        t = 0:h:tf;

        global xbarrax
        xbarrax(16,1)=0;

        % Arrays needed to monitorize cable tensions later

        switch Flag3
            case 'YES'
                global xdotArray
                xdotArray(16,1)=0;
```

```
        global tvec
        tvec(1,1)=0;
end

xtheta0=x_to_xtheta(x0,m);
display(xtheta0);

switch Flag00
    case 'NO'
        dxdt= @(t,xtheta) xdot_HexaMX28(t, xtheta, m, uvstar);
        [times,states]=ode45(dxdt,t,xtheta0,opts);
    case 'YES'
        if strcmp(Flag4, 'YES')
            dxdt= @(t,xtheta) xdot_HexaMX28(t, xtheta, m, uvstar, ...
                xthetastar, K, @uo_pert_wrench_HexaMX28);
            [times,states]=ode45(dxdt,t,xtheta0,opts);
        else
            dxdt= @(t,xtheta) xdot_HexaMX28(t, xtheta, m, uvstar, ...
                xthetastar, K);
            [times,states]=ode45(dxdt,t,xtheta0,opts);
        end
    otherwise
        disp('this Flag00 does not exist');
end

switch Flag01
    case 'FULL'
        figure;
        plot(times,rad2deg(states(:,1)))
        xlabel('t(s)'),ylabel('theta1(t) [deg]'),title('State')

        figure;
        plot(times,rad2deg(states(:,2)))
        xlabel('t(s)'),ylabel('theta2(t) [deg]'),title('State')

        figure;
        plot(times,rad2deg(states(:,3)))
        xlabel('t(s)'),ylabel('theta3(t) [deg]'),title('State')

        figure;
        plot(times,rad2deg(states(:,4)))
        xlabel('t(s)'),ylabel('theta4(t) [deg]'),title('State')

        figure;
```

```
plot(times,rad2deg(states(:,5)))
xlabel('t(s)'),ylabel('theta5(t) [deg]'),title('State')

figure;
plot(times,rad2deg(states(:,6)))
xlabel('t(s)'),ylabel('theta6(t) [deg]'),title('State')

figure;
plot(times,rad2deg(states(:,7)))
xlabel('t(s)'),ylabel('beta1(t) [deg]'),title('State')

figure;
plot(times,rad2deg(states(:,8)))
xlabel('t(s)'),ylabel('beta2(t) [deg]'),title('State')

figure;
plot(times,states(:,9))
xlabel('t(s)'),ylabel('theta1dot(t)'),title('State')

figure;
plot(times,states(:,10))
xlabel('t(s)'),ylabel('theta2dot(t)'),title('State')

figure;
plot(times,states(:,11))
xlabel('t(s)'),ylabel('theta3dot(t)'),title('State')

figure;
plot(times,states(:,12))
xlabel('t(s)'),ylabel('theta4dot(t)'),title('State')

figure;
plot(times,states(:,13))
xlabel('t(s)'),ylabel('theta5dot(t)'),title('State')

figure;
plot(times,states(:,14))
xlabel('t(s)'),ylabel('theta6dot(t)'),title('State')

figure;
plot(times,states(:,15))
xlabel('t(s)'),ylabel('beta1dot(t)'),title('State')

figure;
plot(times,states(:,16))
```

```
        xlabel('t(s)'),ylabel('beta2dot(t)'),title('State')

    case 'PACK'
        figure;
        box on
        hold on
        plot(times,rad2deg(states(:,1)),'LineWidth',3)
        plot(times,rad2deg(states(:,2)),'LineWidth',3)
        plot(times,rad2deg(states(:,3)),'LineWidth',3)
        plot(times,rad2deg(states(:,4)),'LineWidth',3)
        plot(times,rad2deg(states(:,5)),'LineWidth',3)
        plot(times,rad2deg(states(:,6)),'LineWidth',3)

        xlabel('t(s)'),ylabel('theta1-theta6 [deg]'), ...
                                    title('Motor Angles')
        hold off

        figure;
        box on
        hold on
        plot(times,rad2deg(states(:,7)),'r','LineWidth',3)
        plot(times,rad2deg(states(:,8)),'g','LineWidth',3)
        xlabel('t(s)'),ylabel('beta1=red beta2=green [deg]'), ...
                                       title('Pendulum Angles')
        hold off

        figure;
        box on
        hold on
        plot(times,states(:,9),'LineWidth',3)
        plot(times,states(:,10),'LineWidth',3)
        plot(times,states(:,11),'LineWidth',3)
        plot(times,states(:,12),'LineWidth',3)
        plot(times,states(:,13),'LineWidth',3)
        plot(times,states(:,14),'LineWidth',3)
        xlabel('t(s)'),ylabel('theta1dot-theta6dot [rad/s]'), ...
                                         title('Motor Speeds')
        hold off

        figure;
        box on
        hold on
        plot(times,states(:,15),'r','LineWidth',3)
        plot(times,states(:,16),'g','LineWidth',3)
        xlabel('t(s)'),ylabel('beta1dot=red beta2dot=green [rad/s]' ...
```

```
                                                    ),title('Pendulum Speeds')
            hold off
        otherwise
            disp('this Flag01 does not exist')
    end

    switch Flag1

        case 'YES'

           % Building statesx
           statesx(1,:)=xtheta_to_xq(states(1,:)',xstar,m)';
           for i=2:max(size(states))
               statesx(i,:)=xtheta_to_xq(states(i,:)',statesx(i-1,:)',m)';
           end

           switch Flag01
               case 'FULL'
                   figure;
                   plot(times,statesx(:,1))
                   xlabel('t(s)'),ylabel('x(t)'),title('State')

                   figure;
                   plot(times,statesx(:,2))
                   xlabel('t(s)'),ylabel('y(t)'),title('State')

                   figure;
                   plot(times,statesx(:,3))
                   xlabel('t(s)'),ylabel('z(t)'),title('State')

                   figure;
                   plot(times,rad2deg(statesx(:,4)))
                   xlabel('t(s)'),ylabel('psi(t) [deg]'),title('State')

                   figure;
                   plot(times,rad2deg(statesx(:,5)))
                   xlabel('t(s)'),ylabel('theta(t) [deg]'),title('State')

                   figure;
                   plot(times,rad2deg(statesx(:,6)))
                   xlabel('t(s)'),ylabel('phi(t) [deg]'),title('State')

                   figure;
                   plot(times,statesx(:,9))
                   xlabel('t(s)'),ylabel('xdot(t)'),title('State')
```

```
        figure;
        plot(times,statesx(:,10))
        xlabel('t(s)'),ylabel('ydot(t)'),title('State')

        figure;
        plot(times,statesx(:,11))
        xlabel('t(s)'),ylabel('zdot(t)'),title('State')

        figure;
        plot(times,statesx(:,12))
        xlabel('t(s)'),ylabel('psidot(t)'),title('State')

        figure;
        plot(times,statesx(:,13))
        xlabel('t(s)'),ylabel('thetadot(t)'),title('State')

        figure;
        plot(times,statesx(:,14))
        xlabel('t(s)'),ylabel('phidot(t)'),title('State')

    case 'PACK'
        figure;
        box on
        hold on
        plot(times,statesx(:,1),'r','LineWidth',3)
        plot(times,statesx(:,2),'g','LineWidth',3)
        plot(times,statesx(:,3),'b','LineWidth',3)
        xlabel('t(s)'),ylabel('x=red y=green z=blue [m]'), ...
                                    title('Platform position')
        hold off


        figure;
        box on
        hold on
        plot(times,rad2deg(statesx(:,4)),'r','LineWidth',3)
        plot(times,rad2deg(statesx(:,5)),'b','LineWidth',3)
        plot(times,rad2deg(statesx(:,6)),'g','LineWidth',3)
        xlabel('t(s)'), ...
            ylabel('psi=red theta=blue phi=green [deg]'), ...
                                    title('Platform orientation')
        hold off

        figure;
```

```
            box on
            hold on
            plot(times,statesx(:,9),'r','LineWidth',3)
            plot(times,statesx(:,10),'b','LineWidth',3)
            plot(times,statesx(:,11),'g','LineWidth',3)
            xlabel('t(s)'), ...
                ylabel('xdot=red ydot=blue zdot=green [m/s]'), ...
                                            title('Platform speed')
            hold off

            figure;
            box on
            hold on
            plot(times,statesx(:,12),'r','LineWidth',3)
            plot(times,statesx(:,13),'b','LineWidth',3)
            plot(times,statesx(:,14),'g','LineWidth',3)
            xlabel('t(s)') ...
                ,ylabel('psidot=red thetadot=blue phidot=green [rad/s
                    ]' ...
                                    ),title('Platform rotation speed')
            hold off

        otherwise
            disp('this Flag01 does not exist')
    end


     %%Ploting Gtot
     figure;
     box on
     hold on
     Gp(3,max(size(statesx)))=0;
     Gb(3,max(size(statesx)))=0;
     Gtot(3,max(size(statesx)))=0;
     for i=1:max(size(statesx))
         Gp(1,i)=statesx(i,1);
         Gp(2,i)=statesx(i,2);
         Gp(3,i)=statesx(i,3);
     end
     for i=1:max(size(Gp))
         R4i=R4(statesx(i,:));
         R5i=R5(statesx(i,:));
         Gb(:,i)=Gp(:,i)+R4i*R5i*([0, 0, m.ls]');
     end
     for i=1:max(size(Gp))
```

```
                Gtot(:,i)=1/(m.mp+m.mb)*(m.mp*Gp(:,i)+m.mb*Gb(:,i));
            end
            plot(times,Gtot(1,:),'r','LineWidth',3)
            plot(times,Gtot(2,:),'b','LineWidth',3)
            plot(times,Gtot(3,:),'g','LineWidth',3)
            xlabel('t(s)'),ylabel('Gtot x=red,y=blue,z=green [m]'), ...
                                                title('Gtot position')
            hold off
    otherwise
        disp('this input Flag1 does not exist')
end

switch Flag2
    case 'YES'
        uvArray(6,max(size(times)))=0;
        for i=1:max(size(times))
            uvArray(:,i)=u_function_HexaMX28(times(i),states(i,:)', ...
                                                m,uvstar,xthetastar,K);
        end
        figure;
        box on
        hold on
        plot(times,uvArray(1,:),'r','LineWidth',3)
        plot(times,uvArray(2,:),'b','LineWidth',3)
        plot(times,uvArray(3,:),'g','LineWidth',3)
        plot(times,uvArray(4,:),'y','LineWidth',3)
        plot(times,uvArray(5,:),'k','LineWidth',3)
        plot(times,uvArray(6,:),'m','LineWidth',3)
        xlabel('t(s)'),ylabel('Voltage [V]'), ...
                    title('Monitorizing Voltages')
        hold off
    case 'NO'
        disp('omitting monitorizing of wrenches')
    otherwise
        disp('this input Flag2 does not exist')
end

% Monitorize cable tensions
switch Flag3
    case'YES'

        xdotts=timeseries(xdotArray,tvec);
        xdotresample=resample(xdotts,times);
        xdot_Array=xdotresample.Data;
        uCTArray(6,max(size(times)))=0;
```

```
for i=1:max(size(times))

    uCTArray(:,i)=(1/m.ro)*(m.a*uvArray(:,i)+m.b* ...
        xdot_Array(1:6,1,i)+m.c*xdot_Array(9:14,1,i));
end

figure;
box on
hold on
plot(times,uCTArray(1,:),'r','LineWidth',3)
plot(times,uCTArray(2,:),'b','LineWidth',3)
plot(times,uCTArray(3,:),'g','LineWidth',3)
plot(times,uCTArray(4,:),'y','LineWidth',3)
plot(times,uCTArray(5,:),'k','LineWidth',3)
plot(times,uCTArray(6,:),'m','LineWidth',3)
xlabel('t(s)'),ylabel('Cable tensions [N]'), ...
                        title('Monitorizing CT')
hold off

% Monitorize accelerations
for i=1:max(size(times))
    ACC_Array(:,i)=xdot_Array(9:16,1,i);
end


figure;
box on
hold on
plot(times,ACC_Array(1,:),'r','LineWidth',3)
plot(times,ACC_Array(2,:),'b','LineWidth',3)
plot(times,ACC_Array(3,:),'g','LineWidth',3)
plot(times,ACC_Array(4,:),'y','LineWidth',3)
plot(times,ACC_Array(5,:),'k','LineWidth',3)
plot(times,ACC_Array(6,:),'m','LineWidth',3)
xlabel('t(s)'),ylabel('theta1dotdot—theta6dotdot [rad/s^2]'),
    ...
                                    title('Motor Accelerations')
hold off

figure;
box on
hold on
plot(times,ACC_Array(7,:),'r','LineWidth',3)
plot(times,ACC_Array(8,:),'g','LineWidth',3)
```

```
            xlabel('t(s)'), ...
                ylabel('beta1dotdot=red beta2dotdot=green [rad/s^2]'), ...
                                            title('Pendulum Accelerations')
            hold off

        case 'NO'
            disp('omitting monitorizing of cable tenisons')
        otherwise
            disp('this input Flag3 does not exist')
    end

case 'FinDif'

    xthetastar(1:8,1)=[0,0,0,0,0,0,xstar(7),xstar(8)]';
    xthetastar(9:16,1)=zeros(8,1);

    display(xthetastar);
    display(utaustar);
    display(xstar);
    display(utstar);
    display(uvstar);

    [Alin,Blin]=Lineal_HexaMX28_Analitic( xstar, uvstar, m);

    [K,SolRic,Eigvals]=lqr(Alin,Blin,Q,R);


case 'VIDEO'

    time=cputime;
     % Animation
     % h is the sampling time
     % n is the scaling factor in order not to plot with the same step
     % than during the integration with ode45
     fs=30;
     n=round(1/(fs*h));

    % Set up the movie.
    writerObj = VideoWriter('hexapole_HexaMX28','MPEG-4'); % Name it.
    % writerObj.FileFormat = 'mp4';
    writerObj.FrameRate = fs; % How many frames per second.
    open(writerObj);

     for i = 1:n:length(times)
                q.x=statesx(i,1);
```

```
                q.y=statesx(i,2);
                q.z=statesx(i,3);
                q.psi=statesx(i,4);
                q.theta=statesx(i,5);
                q.phi=statesx(i,6);
                q.beta1=statesx(i,7);
                q.beta2=statesx(i,8);

                elapsed = cputime-time;
                if elapsed>200
                    disp(elapsed);
                    disp('took too long to generate the video')
                    break
                else
                    % 'gcf' can handle if you zoom in to take a movie.

                    Draw_DM_HexaMX28(m,q)
                    frame = getframe(gcf);
                    writeVideo(writerObj, frame);
                end

    end
    close(writerObj); % Saves the movie.

case 'NEW_VIDEO'

       time=cputime;
        % Animation
        % h is the sampling time
        % n is the scaling factor in order not to plot with the same step
        % than during the integration with ode45
        fs=30;
        n=round(1/(fs*h));

       % Set up the movie.
       writerObj = VideoWriter('hexapole_STL','MPEG-4'); % Name it.
       % writerObj.FileFormat = 'mp4';
       writerObj.FrameRate = fs; % How many frames per second.
       open(writerObj);


        % Draw external structure
        f100=Draw_STL();

        for i = 1:n:length(times)
```

```
                    q.x=statesx(i,1);
                    q.y=statesx(i,2);
                    q.z=statesx(i,3);
                    q.psi=statesx(i,4);
                    q.theta=statesx(i,5);
                    q.phi=statesx(i,6);
                    q.beta1=statesx(i,7);
                    q.beta2=statesx(i,8);

                    elapsed = cputime-time;
                    if elapsed>400
                        disp(elapsed);
                        disp('took too long to generate the video')
                        break

                    else
                        % 'gcf' can handle if you zoom in to take a movie.
                        [ Array_plot ]=Draw_mobile(m,q);
                        frame = getframe(gcf);
                        writeVideo(writerObj, frame);
                        Delete_plots(Array_plot);
                    end

            end
            close(writerObj); % Saves the movie.

    otherwise
        disp('This flag input does not exist');
end
```

C.4.2 LinealHexaMX28Analitic

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Lineal_HexaMX28_Analitic.m
%
% Purpose:
%
%    This function builds the A and B matrices of the linearized to linearize
%    the model, using the analytic way instead of finite differences.
%
% User-specified parameters:
%
%    xstar:     State vector in vertical equilibrium (point of linearitzation)
%    uvstar:    Actuation vector (motor voltages) in vertical equilibrium
%               state
%    m:         Structural parameter containing all geometric and dynamic
%               variables that are used to define the model (such as
%               gravity constant g, or anchor points Ai, Bi)
%
% Output:
%
%    A:         Matrix that used to linearize the system (multiplies error of
%               state) A(x-xstar)
%
%    B:         Matrix that used to linearize the system (multiplies error of
%               actuation) B(u-uvstar)
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [ A,B ] = Lineal_HexaMX28_Analitic( xstar, uvstar, m )

    % Jacobian JEazuvG of Maple will be used in this linearitzation

    % Main matrices of our dynamic model needed for the linearitzation

    H=Hmatrix(xstar);
    Ji=Jimatrix(xstar);

    E=Ematrix(xstar);
    COR=CORmatrix(xstar);
    Jidot=Jidotmatrix(xstar,m);

    invJi=pinv(Ji);

    % Now we build matrices needed for the HexaMX_28 model
```

```
    Hm=[m.c*(1/m.ro)*E,zeros(8,2)];
    Cm=[m.b*(1/m.ro)*E,zeros(8,2)];

    Mthetam=(H*invJi-Hm);
    Cthetam=((COR-H*invJi*Jidot)*invJi-Cm);


    JEazuvG=JEazuvGmatrix(xstar,uvstar);

    A21=inv(Mthetam)*JEazuvG*invJi;
    A22=-(Mthetam\Cthetam);

    % Compute A matix (exacte way)
    A=[zeros(8),eye(8);A21,A22];

    % Compute B matix (exacte way)
    B=[zeros(8,6);(Mthetam)\((1/m.ro)*E*m.a)];

end
```

C.4.3 xdotHexaMX28

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% xdot_HexaMX28
%
% Purpose:
%
%     This function evaluates f(xtheta,uv) in the electromechanical model
%
%               xthetadot = f(xtheta, uv)
%
%     of the Hexapole. This function is meant to be passed as a parameter to the
%     ode45 or ode15s simulation routines of Matlab. The value of utau is assumed
%     to be given by the optimal control law
%
%               uv = uvstar - K * ( xtheta - xthetastar )
%
%     defined by the input parameters. The function also allows the evaluation of
%     f(xtheta, uv) in the presence of perturbation forces applied to the
%     center of mass of the pole.
%
% Input parameters:
%
%     t:          The instant of time for which the evaluation of f has to be
%                 performed. This parameter is used to compute perturbation
```

```
%                forces when present, which are time-dependent.
%     xtheta:    The state at which we evaluate f(xtheta, uv).
%     m:         A structure with all geometric and dynamic parameters of the
%                Hexapole.
%     uvstar:    The steady-state action needed to keep the hexapole in
%                equilibrium.
%     xthetastar: The equilibrium state desired for the Hexapole.
%     K:         The gain matrix of the control law.
%     u0_handle: A handle to a function that produces non-modelled perturbation
%                forces applied to the pole.
%
% Output parameters:
%
%     xthetadot: The value of f(xtheta, uv).
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [ xthetadot ] = xdot_HexaMX28( t, xtheta, m, uvstar, xthetastar, K,
    uo_handle)

    % Global xthetaplot

    global xq

    global xdotArray

    global Flag3

    % Vector of times necesary to later monitorize cable tensions
    switch Flag3
        case 'YES'
            global tvec
            tvec=horzcat(tvec,t);
    end

    p0=xq(1:3);
    rpy0=xq(4:6);

    [ pf,rpyf ] = FKNewtonMeth( xtheta,p0,rpy0,m );



    x(1:8,1)=[pf;rpyf;xtheta(7:8)];

    H=Hmatrix(x);
```

```
G=Gvector(x);
E=Ematrix(x);
Ji=Jimatrix(x);

x(9:16,1)=Ji\xtheta(9:16);

xq=x;

COR=CORmatrix(x);
Jidot=Jidotmatrix(x,m);


invJi=pinv(Ji);

% Now we build needed matrices for HexaMX_28 model
Hm=[m.c*(1/m.ro)*E,zeros(8,2)];
Cm=[m.b*(1/m.ro)*E,zeros(8,2)];

Mthetam=(H*invJi-Hm);
Cthetam=((COR-H*invJi*Jidot)*invJi-Cm);


switch nargin
    case 4
        uv=u_function_HexaMX28(t,xtheta,m,uvstar);
        thetadot=xtheta(9:16);
        thetadotdot=Mthetam\((1/m.ro)*E*m.a*uv-Cthetam*thetadot-G);
        disp('function evaluated at time:');
        disp(t);

        xthetadot=[thetadot; thetadotdot];

    case 6
        uv=u_function_HexaMX28(t, xtheta, m, uvstar, xthetastar, K);
        thetadot=xtheta(9:16);
        thetadotdot=Mthetam\((1/m.ro)*E*m.a*uv-Cthetam*thetadot-G);
        disp('function evaluated at time:');
        disp(t);

        xthetadot=[thetadot; thetadotdot];

        switch Flag3
            case 'YES'
                xdotArray=horzcat(xdotArray,xthetadot);
        end
```

```
        case 7

            Eo=Eomatrix(x);

            uv=u_function_HexaMX28(t,xtheta,m,uvstar,xthetastar,K);
            uo=uo_handle(t);
            thetadot=xtheta(9:16);
            thetadotdot=Mthetam\((1/m.ro)*E*m.a*uv+Eo*uo-Cthetam*thetadot-G);
            disp('function evaluated at time:');
            disp(t);

            xthetadot=[thetadot; thetadotdot];

            switch Flag3
                case 'YES'
                    xdotArray=horzcat(xdotArray,xthetadot);
            end


        otherwise
            disp('error in number of input variables');
    end

end
```

C.4.4 uopertwrenchHexaMX28

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% uo_pert_wrench_HexaMX28
%
% Purpose:
%
%     The aim of this function is to introduce unmodelled force perturbations
%     in a particular instant of time
%
% Input parameters:
%
%     t: Evaluation time of the function (the purpose of this input is to know
%        in which instant of time the perturbation has to be introduced)
%
% Output parameters:
%
%     uo: Vector of perturbation force
%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [ uo ] = uo_pert_wrench_HexaMX28( t )

    if  t<5.8
        uo=[0,0,0]';

    elseif (t>=5.8 && t<5.9)
        uo=[0.5, 0, 0]';

    elseif (t>=9.5 && t<9.6)
        uo=[0, 0.5, 0]';

    elseif (t>=13 && t<13.1)
        uo=[0, 0, -2]';

    else
        uo=[0,0,0]';

    end

end
```

C.5 Funcions comunes

C.5.1 Solució del problema cinemàtic directe

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FKNewtonMeth
%
% Purpose:
%
%     Given a determinate ROW vector of xtheta state vector of our system and
%     a starting position of the platform vectors p0,rpy0, uses Newton method
%     to find the correct position of the platform that matches the ls lenghts.
%
% Input parameters:
%
%     xtheta:     The state at which we evaluate f(xtheta, utau).
%     p0:         Aproximation of the platform position (x,y,z) to start
%                 the iterations.
%     rpy0:       Aproximation of the platform orientation (psi,theta,phi)
%                 to start the iterations.
%     m:          A structure with all geometric and dynamic parameters of the
%                 Hexapole.
```

```
%
% Output parameters:
%
%     pf:         Solution of the platform position at the end of the iterations
%     rpyf:       Solution of the platform orientation at the end of the
%                 iterations
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [ pf,rpyf] = FKNewtonMeth( xtheta,p0,rpy0,m)

    % Precision you want to have in your solution (consider that By default,
    % MATLAB uses 16 digits of precision. )

    ep=1e-12;

    ls=theta_to_L(xtheta,m);

    % Limit number of iterations to avoid infinite loop

    t0=0;
    iterlim=15;

    % Defining our xold coordenates, our starting point, aproximate to the final
    % solution
    xo=[p0;rpy0];

        converged=false;

        % Iteration loop with 1e-9 tolerance error for instance
        while not(converged)

            t0=t0+1;

            % Fxo will be the value of our function for xold
            Fxo=FKkinematiceq1(ls,xo(1:3),xo(4:6),m);

            % Fx will be the Jacobian in xold point
            Fx=FKJacobianAnalitic(xo(1:3),xo(4:6),m);

            % Inverting Jacobian
            iFx=pinv(Fx);

            % Solving the linear system
            dx=-iFx*Fxo;
```

```
            % Updating the value of xold
            xo=xo+dx;

            converged=max(abs(dx))>ep;

            pf=xo(1:3);
            rpyf=xo(4:6);

            if t0>iterlim
                disp('max number of iterations reached')
                break
            end

        end

    disp(['number of iter. = '])
    disp(t0)

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FKkinematiceq1
%
% Purpose:
%
%     This functions simply implements de nonlinear equation we need to
%     solve using the Newton method
%
% Input parameters:
%
%     ls:         Length of the cables.
%     p:          Position vector of the central point of the platrom in ABS ref.
%     rpy:        Euler angles to describe the orientation of the platform.
%     m:          A structure with all geometric and dynamic parameters of the
%                 Hexapole.
%
% Output parameters:
%
%     O:          Is the value of the equation
%                 (F(x)=0; O=ls^2-(norm(ai-p-R*bi))^2)
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [ O ] = FKkinematiceq1( ls,p,rpy,m )
```

```
    R = [cos(rpy(2)) * cos(rpy(3)) -cos(rpy(2)) * sin(rpy(3)) sin(rpy(2)); ...
        sin(rpy(1)) * sin(rpy(2)) * cos(rpy(3)) + cos(rpy(1)) * sin(rpy(3)) ...
        -sin(rpy(1)) * sin(rpy(2)) * sin(rpy(3)) + cos(rpy(1)) * cos(rpy(3)) ...
        -sin(rpy(1)) * cos(rpy(2)); -cos(rpy(1)) * sin(rpy(2)) * cos(rpy(3)) ...
        + sin(rpy(1)) * sin(rpy(3)) cos(rpy(1)) * sin(rpy(2)) * sin(rpy(3)) +...
        sin(rpy(1)) * cos(rpy(3)) cos(rpy(1)) * cos(rpy(2));];

    % biabs == matrix with all the positions vectors that were B matrix, but now
    % in ABS ref. (matrix of platform position in ABS ref.)
    biabs = zeros(3,6);
    for i=1:6
        biabs(:,i) = p + R*m.B(:,i);
    end
    % XX is the second part of the equation we want to evaluate F(x)=0, it
    % is a column matrix, the first part is: (lengths of the cables)^2 == ls^2
    XX=zeros(6,1);
    for i=1:6
        XX(i,1)=(norm(m.A(:,i)-biabs(:,i))).^2;
    end

    % Value of the function F(x)
    O=XX-ls.^2;

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FKJacobianAnalitic
%
% Purpose:
%
%     Uses the screw jacobian method to find Jacobian matrix analyticaly
%     implemented with Maple
%
% Input parameters:
%
%     p:          Position vector of the central point of the platrom in ABS ref.
%     rpy:        Euler angles to describe the orientation of the platform.
%     m:          A structure with all geometric and dynamic parameters of the
%                 Hexapole.
%
% Output parameters:
%
%     ANAJ:       Jacobian matrix that is meant to be used with NewtonMeth
%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [ ANAJ ] = FKJacobianAnalitic( p,rpy,m )

    % We make sure that the input vectors will fit the matricial equations
    if isrow(p)
        p=p';
    end
    if isrow(rpy)
        rpy=rpy';
    end

    R = [cos(rpy(2)) * cos(rpy(3)) -cos(rpy(2)) * sin(rpy(3)) sin(rpy(2)); ...
        sin(rpy(1)) * sin(rpy(2)) * cos(rpy(3)) + cos(rpy(1)) * sin(rpy(3)) ...
        -sin(rpy(1)) * sin(rpy(2)) * sin(rpy(3)) + cos(rpy(1)) * cos(rpy(3)) ...
        -sin(rpy(1)) * cos(rpy(2)); -cos(rpy(1)) * sin(rpy(2)) * cos(rpy(3)) ...
        + sin(rpy(1)) * sin(rpy(3)) cos(rpy(1)) * sin(rpy(2)) * sin(rpy(3)) +...
        sin(rpy(1)) * cos(rpy(3)) cos(rpy(1)) * cos(rpy(2));];

    Am = [1 0 sin(rpy(2)); 0 cos(rpy(1)) -sin(rpy(1)) * cos(rpy(2)); 0 ...
        sin(rpy(1)) cos(rpy(1)) * cos(rpy(2));];

    for i=1:6
        r(:,i)=R*m.B(:,i);
        d(:,i)=m.A(:,i)-p-r(:,i);
        Jd(1:3,i)=d(:,i);
        Jd(4:6,i)=cross(r(:,i),(m.A(:,i)-p));
    end
    Ae=[eye(3),zeros(3);zeros(3),Am];

    Ft=2*Jd'*Ae;

    ANAJ=-Ft;

end
```

C.5.2 Solució del problema cinemàtic invers

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% IKsolve
%
% Purpose:
%
%     The aim of this function is give solution to the inverse kinematics
%     problem. Given a state x in platform coordinates we must obtain the
```

```
%     lengths of the cables
%
% Input parameters:
%
%     x:          The configuration of our system in platform coordinates.
%     m:          A structure with all geometric and dynamic parameters of the
%                 Hexapole.
%
% Output parameters:
%
%    lf:          The lengths of the cables matching the x configuration
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [ lf ] = IKsolve( x,m )

    p=x(1:3);
    rpy=x(4:6);

    R = [cos(rpy(2)) * cos(rpy(3)) -cos(rpy(2)) * sin(rpy(3)) sin(rpy(2));...
        sin(rpy(1)) * sin(rpy(2)) * cos(rpy(3)) + cos(rpy(1)) * sin(rpy(3))...
        -sin(rpy(1)) * sin(rpy(2)) * sin(rpy(3)) + cos(rpy(1)) * cos(rpy(3))...
        -sin(rpy(1)) * cos(rpy(2)); -cos(rpy(1)) * sin(rpy(2)) * cos(rpy(3))...
        + sin(rpy(1)) * sin(rpy(3)) cos(rpy(1)) * sin(rpy(2)) * sin(rpy(3)) ...
        + sin(rpy(1)) * cos(rpy(3)) cos(rpy(1)) * cos(rpy(2));];

    biabs = zeros(3,6);
    for i=1:6
        biabs(:,i) = p + R*m.B(:,i);
    end

    for i=1:6
        lf(i,1)=-1*norm(biabs(:,i)-m.A(:,i));
    end

end
```

C.5.3 Funcions de dibuixat del robot

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Draw_DM_MXcord
%
% Purpose:
%
%     This function plots the Hexapole at a particular configuration q
```

```
%     (in platform coordinates).
%
% Input parameters:
%
%     q:          Configuration vector in a particular instant of time, given in
%                 platform coordinates.
%     m:          A structure with all geometric and dynamic parameters of the
%                 Hexapole.
%
% Output parameters:
%
%     Plot of the Hexapole in a particular configuration.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [  ] = Draw_DM_MXcord( m, q )

    p=[q.x,q.y,q.z]';
    figure(100);
    pos_fig1 = [0 0 1920 1080];
    set(gcf,'Position',pos_fig1)


    l=m.ls;
    Bp=m.B;

    x(1)=q.x;
    x(2)=q.y;
    x(3)=q.z;
    x(4)=q.psi;
    x(5)=q.theta;
    x(6)=q.phi;
    x(7)=q.beta1;
    x(8)=q.beta2;


    R = [cos(x(5)) * cos(x(6)) -cos(x(5)) * sin(x(6)) sin(x(5)); ...
        sin(x(4)) * sin(x(5)) * cos(x(6)) + cos(x(4)) * sin(x(6)) ...
       -sin(x(4)) * sin(x(5)) * sin(x(6)) + cos(x(4)) * cos(x(6)) ...
       -sin(x(4)) * cos(x(5)); -cos(x(4)) * sin(x(5)) * cos(x(6)) ...
        + sin(x(4)) * sin(x(6)) cos(x(4)) * sin(x(5)) * sin(x(6)) +...
        sin(x(4)) * cos(x(6)) cos(x(4)) * cos(x(5));];

    %biabs == matrix with all the positions vectors that were B matrix, but now
    %in ABS ref. (matrix of platform position in ABS ref.)
```

```
%biabs[:,i]= p + m.B[:,i];
for i=1:6
    biabs(:,i)= p + R*m.B(:,i);
end
%drawing axis
trplot(eye(3), 'length', 0.1, 'color', 'k');
hold on;

%%%%%%%%%%%%%%%%%%
view([90,0]); % this part sets the view of the plot coment if you want
    isometric

%%%%%%%%%%%%%%%%%%
axis(1e-3*[-400 400 -400 400 -1300 200]); %[-400 400 -400 400 -500 700] for
    upper position
                                          % [-400 400 -400 400 -1300 200] for
                                          % lower

%drawing the poligons to see ABS and platform
%drawing ABS
fill3(m.A(1,:),m.A(2,:),m.A(3,:),(1/255)*[191,191,191]);

%drawing the original position of the platform over ABS (fixed platform
%just to have a reference
% fill3(m.B(1,:),m.B(2,:),m.B(3,:),'w');

%drawing the real platform
fill3(biabs(1,:),biabs(2,:),biabs(3,:),(1/255)*[127,127,127]);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%drawing the cables
%
for i=1:6
    plot3([biabs(1,i),m.A(1,i)],[biabs(2,i),m.A(2,i)],[biabs(3,i),m.A(3,i)],'
        color','r');
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%drawing the pendulum

%creating rs vector relative to platform;

OGb = [x(1) + sin(x(8)) * l; x(2) - sin(x(7)) * cos(x(8)) * l; x(3) + cos(x
    (7)) * cos(x(8)) * l;];
```

```
    [xx,yy,zz]=sphere;

    sph=surf(xx*0.025+OGb(1),yy*0.025+OGb(2),zz*0.025+OGb(3));
    sph.EdgeColor = 'none';
    sph.FaceColor = (1/255)*[127,127,127];

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %draw the stick
    plot3([p(1),OGb(1)],[p(2),OGb(2)],[p(3),OGb(3)],'color','k');

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %draw Gtot position
    rgtot=0.012;%radius of the point

    Gp=p;
    Gb=OGb;

    Gtot=1/(m.mp+m.mb)*(m.mp*Gp+m.mb*Gb);


    sph1=surf(xx*rgtot+Gtot(1),yy*rgtot+Gtot(2),zz*rgtot+Gtot(3));
    sph1.EdgeColor = 'none';
    sph1.FaceColor = 'r';

    hold off;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Draw_STL
%
% Purpose:
%
%     This function plots the octoedrical structure. In this case we use an stl
%     model obtained with SOLIDWORKS.
%
% Input parameters:
%
%     No inputs needed.
%
% Output parameters:
%
%     Plot of the octaedrical structure using stl model.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function [f100] = Draw_STL( )

    f100=figure(100);
    pos_fig1 = [0 0 1280 720];
    set(gcf,'Position',pos_fig1)

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    % Loads the UNIONS and translates and rotates it to its place

    [univ,unif] = stlRead('nomes unions.stl');
    univ2 = [univ(:,1:3)'; ones(size(univ(:,1)))'];

    % Warning my laptop function rotx is aleady in deg
    univ3 = (transl(-300, 300, -408)*trotx(90)*univ2)'; %(-300,300,-408)
    univ4 = univ3(:,1:3);

    % Loads BARS and translates and rotates it to its place
    [barsv,barsf] = stlRead('barrascarbono.stl');
    barsv2 = [barsv(:,1:3)'; ones(size(barsv(:,1)))'];

    % Warning my laptop function rotx is aleady in deg
    barsv3 = (transl(-300, 290, -400)*trotx(90)*barsv2)'; %(-300,300,-408)
    barsv4 = barsv3(:,1:3);


    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    grid on;
    trplot(eye(3), 'length', 100, 'color', 'k');

    hold on
    axis([-400 400 -400 400 -500 700]);
    daspect([1 1 1]);
    view(3);
    camlight;
    lighting gouraud;

    %draw bars
    patch('Faces',barsf,'Vertices',barsv4,'FaceColor', (1/255)*[89 89 89], ...
        'EdgeColor', 'none', 'FaceLighting', 'gouraud', 'AmbientStrength', 0.15);


    %draw unions
    patch('Faces',unif,'Vertices',univ4,'FaceColor', (1/255)*[252 255 121], ...
```

```
        'EdgeColor', 'none', 'FaceLighting', 'gouraud', 'AmbientStrength', 0.15);


    hold off

    saveas(figure(100), 'temp.fig')

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Draw_mobile
%
% Purpose:
%
%     This function has to be used with Draw_STL, to obtain the video of the
%     simulation. In particular, this functions draws de cables and the pole
%     given a vector of configuration q
%
% Input parameters:
%
%     m:          A structure with all geometric and dynamic parameters of the
%                 Hexapole.
%     q:          Configuration vector in a particular instant of time, given in
%                 platform coordinates
%
% Output parameters:
%
%     Plot of the system in a particular configuration.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [Array_plot ] = Draw_mobile( m,q )

    figure(100);

    % All units in [mm]

    x(1)=1e3*(q.x);
    x(2)=1e3*(q.y);
    x(3)=1e3*(q.z);
    x(4)=q.psi;
    x(5)=q.theta;
    x(6)=q.phi;
    x(7)=q.beta1;
    x(8)=q.beta2;
```

```
A=1e3*m.A;
l=1e3*m.ls;

% biabs == matrix with all the positions vectors that were B matrix, but now
% in ABS ref. (matrix of platform position in ABS ref.)

% biabs[:,i]= p + m.B[:,i]; in [mm]

for i=1:6
    biabs[:,i]= p + R*m.B[:,i];
end

hold on

% Drawing the real platform

plat_Draw=fill3(biabs(1,:),biabs(2,:),biabs(3,:),'k');

% Drawing the cables
w1=plot3([biabs(1,1),A(1,1)],[biabs(2,1),A(2,1)],[biabs(3,1),A(3,1)], ...
                                                        'color','y');
w2=plot3([biabs(1,2),A(1,2)],[biabs(2,2),A(2,2)],[biabs(3,2),A(3,2)], ...
                                                        'color','y');
w3=plot3([biabs(1,3),A(1,3)],[biabs(2,3),A(2,3)],[biabs(3,3),A(3,3)], ...
                                                        'color','y');

w4=plot3([biabs(1,4),A(1,4)],[biabs(2,4),A(2,4)],[biabs(3,4),A(3,4)], ...
                                                        'color','y');
w5=plot3([biabs(1,5),A(1,5)],[biabs(2,5),A(2,5)],[biabs(3,5),A(3,5)], ...
                                                        'color','y');
w6=plot3([biabs(1,6),A(1,6)],[biabs(2,6),A(2,6)],[biabs(3,6),A(3,6)], ...
                                                        'color','y');



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Drawing the pendulum

% Creating rs vector relative to platform;

OGb = [x(1) + sin(x(8)) * l; x(2) - sin(x(7)) * cos(x(8)) * l; x(3) + cos(x
    (7)) * cos(x(8)) * l;];
```

```
    [xx,yy,zz]=sphere;

    radius=0.025e3; % in [mm]

    sph=surf(xx*radius+OGb(1),yy*radius+OGb(2),zz*radius+OGb(3));
    sph.EdgeColor = 'none';
    sph.FaceColor = 'y';

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    % Draw the stick
    stick=plot3([x(1),OGb(1)],[x(2),OGb(2)],[x(3),OGb(3)],'color','k');


    hold off

    Array_plot = [plat_Draw,w1,w2,w3,w4,w5,w6, sph, stick  ];
    drawnow;

end
```