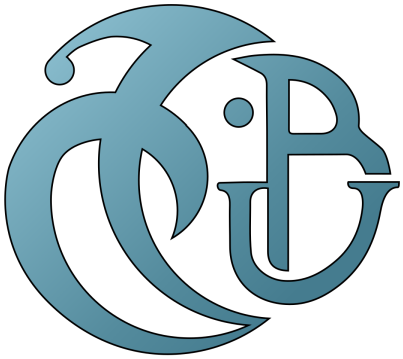
People's Democratic Republic of Algeria
Ministry of Higher education and Research

University of Saad Dahleb-Blida
Institute of Aeronautics and Space Studies

Avionics Major



Master thesis Submitted
In Partial Fulfillement of The Requirements
For **Masters Degree** of Science

Modern Attitude Control for Artificial Satellites

Under supervision of :
Dr. Bekhiti Belkacem
Co-supervisior :
Dr. Dilmi Smail

Presented by:
Bouaffane Zakaria
Zeddam Mohammed Amine

Academic Year 2019 - 2020

Dedication

To my dear an beloved parents, for their sacrifices, unconditional love and support.

To my dear brothers and my little sister for their constant encouragement, help and support.

May Allah protect and reward them all, I pray that Allah take care of them in this "Dunia" or in "Akhira".

To all my friends and class colleagues.

To my thesis partner Zeddam Mohammed-Amine.

Bouaffane Zakaria

Dedication

To my beloved parents who brought me to this world for their full time support, unconditional love and sacrifices.

To my sister and my familly.

To all my freinds.

To my thesis partner Bouaffane Zakaria.

Zeddam Mohammed Amine

Aknowledgment

In the name of Allah, the most beneficent, and the most merciful.
We praise and thank Allah, Lord of the worlds, for his greatness and the giving us the strength, courage, and patient to complete this work.

We would like to express our deepest gratitude to our thesis supervisor Dr.Bekhiti Belkacem For his precious advice and help throughout all the work's period.

Our heartfelt thanks go to the members of the jury for the interest in our research the willingness to review our work and the enrichment of their proposals.

Finally, we would also like to thank everyone who has been involved in this work from near or far.

Thank you everyone.

Abstract

In the past few decades till now, satellites are considered one of the hottest topics featuring Advanced technology and modern control designs, due to its vast utility in multiple domains. critically, control systems have helped satellite's importance continue its growth by the day. The idea of a good controller in a satellite (attitude-wise) is to keep it as stable as possible throughout its journey around the earth maintaining a good attitude precision with acceptable energy consumption. This thesis is concerned with designing an attitude controller for a chosen satellite using linear and nonlinear control theories. initially, the mathematical model of the satellite's dynamics is derived, then, a background review of the linear and nonlinear control theories that are used in the controller design, a multi-input multi-output PID (Proportional Integral Derivative) as linear, and feedback linearization as nonlinear. further, the thesis investigates an optimization method for the PID controller being particle swarm optimization. The designed controller's performance is tested and compared.

ملخص

في العقود القليلة الماضية حتى الآن ، تعتبر الأقمار الصناعية من أكثر المواضيع المثيرة للاهتمام من حيث التكنولوجيا المتقدمة وتصميمات التحكم الحديثة، بسبب فائدتها الواسعة في مجالات متعددة. بشكل حاسم ، أنظمة التحكم ساعدت القمر الصناعي على مواصلة نمو اهميته يومًا بعد يوم. فكرة المتحكم الجيد في القمر الصناعي (من حيث الموقف) هو إبقائه مستقرًا طوال رحلته حول الأرض مع الحفاظ على موقف جيدالدقة مع استهلاك طاقة مقبول. هذه الأطروحة معنية بتصميم جهاز تحكم في الموقف لقمر صناعي مختار باستخدام نظريات التحكم الخطية و غير الخطية. في البداية ، اشتقاق النموذج الرياضي لديناميات القمر الصناعي بعد ذلك ، مراجعة خلفية نظريات التحكم الخطية و غير الخطية المستخدمة في تـ صميم وحدة التحكم ، متعدد المدخلات متعدد المخرجات PID (مشتق تكاملي متناسب) كوحدة تحكم خطية، و ردود الفعل الخطية كوحدة تحكم غير خطية. علاوة على ذلك ، تبحث الأطروحة على تحسين طريقة التحكم PID باستعمال PSO . في الأخير يتم اختبار أداء وحدات التحكم المصممة ومقارنتها.

Contents

1 Background and Preliminaries **12**

1.1 Introduction 12

1.2 Satellite control background 12

1.2.1 Perturbation Theory 12

1.2.2 The attitude control 12

1.2.3 Orbit Control 13

1.2.4 Spacecraft Attitude Actuators 13

1.3 Mathematical Preliminaries 14

1.3.1 Equation of Motion of Rigid Body 14

1.3.2 The Inertia Tensor 16

1.3.3 Lyapunov Direct Method 16

1.3.4 General Rotations: 18

1.3.5 Lagrangian Function 19

1.4 Summary 19

2 Satellite's Dynamic modeling **20**

2.1 Introduction 20

2.2 Modeling Decomposition 20

2.3 Attitude Kinematics in inertial frame 21

2.3.1 Satellite Model Stability Properties 22

2.4 Orbital Frame Attitude Dynamics 23

2.5 Orbital Motion (Orbit Model) 25

2.5.1 Model Linearization 28

2.6 Summary 28

3 Classical Control Theory **29**

3.1 Introduction 29

3.2 Multi-Input Multi-Output PID Controller 30

3.3 Proportional derivative Integral Control 30

3.4 Matrix polynomials (λ - matrices) 31

3.4.1 State space representation 31

3.4.2 Matrix transfer function 31

3.5 Transformation to block controller form 32

3.6 Canonical right matrix fraction description 33

3.7 Solvents and Latent-structure 33

3.7.1 Left and right solvents 34

3.7.2 Complete set of solvents 34

3.8 Matrix polynomial coefficients construction 35

3.9 MIMO PID Control and Diophantine Equation 36

3.10 Summary 41

4 Nonlinear Control Theory **42**

4.1 Introduction 42

4.2 Feedback Linearization 42

4.3 Lie derivatives 43

4.4 Input-State Linearization 44

4.5 Input-Output Linearization 46

4.5.1 SISO nonlinear systems 46

4.5.2 MIMO nonlinear systems 49

4.6 Summary 51

5 Advanced Optimal Control Strategy **52**

5.1 Introduction 52

5.2 Optimal Control 52

5.3 Optimization 53

5.4 Meta-heuristic Methods 54

5.5 Swarm intelligence 54

5.6 Particle Swarm Optimization (PSO) 55
5.6.1 PSO and other evolutionary computation Techniques 56
5.6.2 Concept 56
5.7 Accelerated PSO 60
5.8 Summary 62

6 Implementation **63**
6.1 Introduction 63
6.2 Feedback Linearization Controller 63
6.2.1 Transient and Steady State response 63
6.2.2 Tracking Simulation 65
6.3 MIMO PID Controller 67
6.3.1 Transient and Steady State response 67
6.3.2 Tracking Simulation 69
6.4 MIMO PID Controller with PSO 70
6.5 Comparison Between PID and PID with PSO controllers 73
6.6 Summary 74

Appendices **77**

A Satellite Appendix **78**
A.1 Introduction 78
A.2 Sentinel-1 A 78
A.2.1 Propulsion System 79
A.2.2 Spacecraft Characteristics 79

B MATLAB Codes **80**
B.1 Introduction 80
B.2 Open Loop System 80
B.2.1 Function 82
B.3 Feedback Linearization 82
B.3.1 Function 84
6.4 MIMO PID with PSO Algorithm 85

6.4.1 funv 87

Bibliography **88**

List of Figures

1.1 Spacecraft Actuators 14
1.2 Rigid body 14

2.1 Satellite Orbital frame 23
2.2 Orbiting 25
2.3 Model Block Diagram 27

3.1 A PID Control System 31
3.2 Trajectory tracking control of MIMO PID controller 40

4.1 Input-State Linearization 45
4.2 Input-Output Lineariezed model 47

5.1 Optimization methods in a nature perspective 54
5.2 Representation of Swarm Intelligence 55
5.3 Position mechanism of a PSO 58
5.4 PSO Algorithm Flowchart 59
5.5 PSO and Accelerated PSO applied to f 61

6.1 Step response of feedback linearization control system 64
6.2 Error signals 65
6.3 Tracking trajectory with feedback linearization control. 66
6.4 Tracking Error 67
6.5 Step response with PID Controller 68
6.6 Error signals with PID Controller 69
6.7 Tracking trajectory with MIMO PID Controller 70
6.8 Tracking trajectory with enhanced PID controller 71

6.9 Error signals with Enhanced PID controller 72
6.10 Satellite's Orbit tracking 72

A.1 Sentinel 1-A Spacecraft 78

List of Tables

5.1 We can observe that results are almost identical with a slight difference to global best 61

6.1 Controller performance characteristics 65

6.2 MIMO PID Controller performance characteristics 69

A.1 Sentinel 1-A characteristics 79

Nomenclature

A	State matrix
B	Input matrix
C	Output matrix
D	Feedforward matrix
n	Number of states
m	Number of inputs
p	Number of outputs
$\dot{x}(t)$	Derivative of the state with respect to time
$x(t)$	State of the system
$u(t)$	Control input
$y(t)$	System Output
s	Laplace transformation parameter
$H(s)$	Transfer function
T_c/T_{c1}	Controller similarity transformation matrices
$N_r(s)$	Numerator of the RMFD
$D_r(s)$	Denominator of the RMFD
N_L	Numerator of the RMFD
D_L	Denominator of the LMFD
V_R	Vandermonde matrix
R	Solvent matrix
Ω_c	Block controllability matrix
V	Lyapunov function
Φ	Roll angle
Θ	Pitch angle
ψ	Yaw angle
L	Langragina function.
T	kinetic energy

V	Potential energy
I	Inertia tensor
ω	Angular velocity
$r(t)$	Radius angle
θ	Azimuth angle
ϕ	Elevation angle
$L_f h(x)$	Lie derivative

Acronyms

PID	Proportional Integral Derivative
MIMO	Multi Input Multi Output
SISO	Single Input Single Output
ORF	Orbital Reference Frame
BRF	Body Reference Frame
LMFD	Left Matrix Fraction Description
RMFD	Right Matrix Fraction Description
PSO	Particle Swarm Optimization
GA	Genetic Algorithm
BFO	Bacterial Foraging Optimization
ABC	Artificial Bee Colony

General Introduction

SINCE the launch of the first artificial earth satellite in 1957, the world has changed and a new era of political, military, technological, and scientific advancement. Throughout the years the satellite has played a major role in the development and the enhancement of multiple domains with that said the making of satellite it self was a tough challenge , modeling control and optimization processes are all key parts in delivering a perfect state of the art satellite which can do it's mission accurately and precisely.

A satellite's mission consists mainly of orbiting earth in order to complete it's task, during this phase orbit keeping is crucial in a satellite's journey. By providing a good controller we make sure that the satellite stays around the same trajectory and exploit its functionality at maximum level.

This thesis represents all the work needed to achieve the requested attitude for orbit keeping starting by obtaining dynamic model of satellite ,then the control process using linear and nonlinear methods ,finally applying optimization strategies is a must for a perfect control and orbit keeping.

Thesis Structure is ranked as follows:

- ⊛ Chapter 1 contains the mathematical tools and some satellite's background information needed in the overall view of a satellite's lifespan from the conception to the mission .
- ⊛ Chapter 2 is all about satellite dynamic modeling such as satellite attitude's model, control process, attitude control,orbit control and keeping.

- ⊛ Chapter 3 exposes the classical control theory which is MIMO PID controller with an academic example.;
- ⊛ Chapter 4 Introduces Feedback Linearization control theory followed by academic examples.
- ⊛ Chapter 5 explores advanced optimal control strategy Such as Particle Swarm Optimization and its Accelerated and faster version.
- ⊛ Chapter 6 is a pure application and implementation of our controllers and our optimization strategy with an analysis of their performance.

  finally we give a conclusion to draw up an assessment of simulation results obtained as well as prespectives to consider regarding potential applications experrimental and theorical research.

Chapter 1

Background and Preliminaries

1.1 Introduction

We Focus here primarily on introducing some background notions and concepts that allow a better understanding on the problem in hand.
the chapter has two parts one of them is some satellite basics and control related information and the other is concerned with the necessary mathematical notions used in the thesis for an easy reading process.

1.2 Satellite control background

1.2.1 Perturbation Theory

perturbation in an artificial satellite's orbit is a deviation from some normal or expected motion. The actual path will vary from the theoretical path due to perturbations caused by other mass bodies (such as the Moon) and additional forces not considered in Keplerian motion (such as nonspherical Earth)[1].

1.2.2 The attitude control

Attitude control is controlling the orientation of an object with respect to a specific frame of reference (i.e. they mean the way that the spacecraft is positioned in 3-dimensional space. Attitude is in effect, the way that the spacecraft is "pointed."). Attitude is controlled by tiny thrusters that contain compressed gasses, called monopropellants.

These tiny thrusters can be pointed in different directions. Firing those thrusters releases gas which makes the small adjustments needed to get the ship back on course [2].

1.2.3 Orbit Control

When a satellite has entered into its orbit around the Earth, it moves so fast that the centrifugal force caused by its motion is equal to the Earth's gravitational pull. In the absence of disturbing influences the satellite would continue along its trajectory forever. However, there are small perturbing forces that slow the satellite down and change its orbit(radiation Pressure of sunlight,photons,friction...etc)[3].

1.2.4 Spacecraft Attitude Actuators

We can conceptually divide actuator types into two general classes, passive and active. Passive actuators operate more or less open loop. In other words, after the spacecraft is in the desired attitude, **passive** actuators will keep it there with little or no additional torques needed. **Active** actuators, on the other hand, require continuous feedback and adjustment.

Thrusters

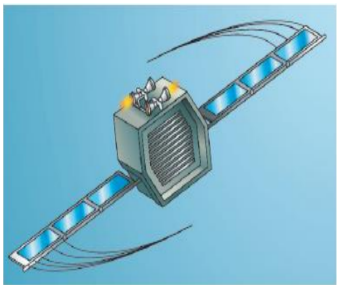
Thrusters are rockets that apply a force some distance away form the center of mass, causing a torque that rotates the spacecraft.

Reaction Wheels

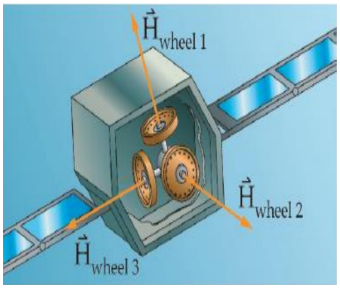
Reaction wheels are apart of a zero-bias system that uses three independent wheels, one along each axis, normally with zero or nearly zero momentum. To rotate the spacecraft or absorb disturbance toques, one or more wheels begin to spin.

Dampers

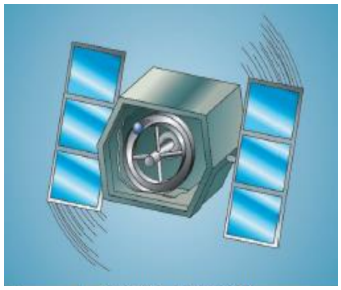
Dampers "absorb" unwanted angular momentum by converting the energy into friction, in much the same way



(a) Thruster



(b) Reaction Wheels



(c) Damper

Figure 1.1: Spacecraft Actuators

1.3 Mathematical Preliminaries

1.3.1 Equation of Motion of Rigid Body

With respect to an inertial frame of reference, let $\vec{r}$ be the vector defining the position of the center of mass of the flying object and $\vec{v}$ is its velocity vector. They are related by the kinematic relation $v = \frac{d\vec{r}}{dt}$. In this respect, we follow the motion of the center of mass under the application of various forces. For many problems in flight dynamics, an axis fixed to the earth can be used as an inertial frame (reference frame). Newton's second law can be applied [8]

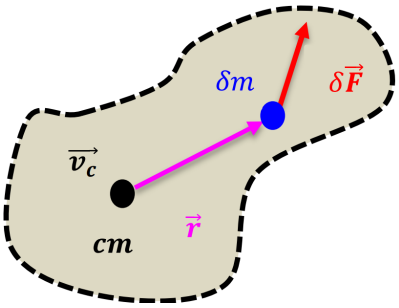


Figure 1.2: Rigid body

Moment equation: $\sum \vec{M} = \frac{d}{dt}(\vec{H})$, $\vec{H} = m(\vec{r} \times \vec{v})$ is the angular momentum

If we let δm be an element of mass of the rigid body, $\vec{v}$ be the velocity of the element mass relative to an absolute or inertial frame. $\delta\vec{M}$ be the resulting Moment acting on the elemental mass then

$$\delta\vec{M} = \frac{d}{dt}\delta\vec{H} = \frac{d}{dt}(\vec{r} \times \vec{v})\delta m \quad \text{with} \quad M = \sum \delta\vec{M}$$

The velocity of the differential mass δm is: $\vec{v} = \vec{v}_c + d\vec{r}/dt$. With $\vec{c}$ is the velocity of the center of mass of the rigid body and $\frac{d\vec{r}}{dt}$ is the velocity of the element relative to center of mass.

$$\vec{v} = \vec{v}_c + \frac{d\vec{r}}{dt} = \vec{v}_c + \vec{\omega} \times \vec{r}$$

Where $\vec{\omega}$ is the angular velocity of the rigid body and $\vec{r}$ is the position of the mass element measured from the center of mass. The total momentum can be written as:

$$\vec{H} = \sum \delta\vec{H} = \sum \delta m \vec{r} \times \vec{v}_c + \sum [\vec{r} \times (\vec{\omega} \times \vec{r})]\delta m$$

Because $\vec{r}$ is measured from the center of mass then $\sum \vec{r}\delta m = 0$ and therefore we get $\vec{H} = \sum [\vec{r} \times (\vec{\omega} \times \vec{r})]\delta m$.

Let : $\vec{\omega} = p\vec{i} + q\vec{j} + r\vec{k}$, $\vec{r} = x\vec{i} + y\vec{j} + z\vec{k}$ and $\vec{H} = H_x\vec{i} + H_y\vec{j} + H_z\vec{k}$.

$$\vec{H} = \sum [\vec{r} \times (\vec{\omega} \times \vec{r})]\delta m \rightarrow \begin{cases} H_x = p\sum(y^2+z^2)\delta m - q\sum xy\delta m - r\sum xz\delta m \\ H_y = q\sum(x^2+z^2)\delta m - p\sum xy\delta m - r\sum yz\delta m \\ H_z = r\sum(x^2+y^2)\delta m - p\sum xz\delta m - r\sum yz\delta m \end{cases}$$

$$\vec{H} = (pI_x - qI_{yx} - rI_{xz})\vec{i} + (qI_y - pI_{yx} - rI_{yz})\vec{j} + (rI_z - qI_{yz} - pI_{xz})\vec{k} \tag{1.1}$$

The sum is replaced by triple integral because $\delta m = \rho\ dxdydz = \rho\ dv$ with ρ is the density of mass and dv is an infinitesimal volume ,$\sum \delta m \equiv \iiint \rho\ dxdydz$.

$$I_x = \iiint (y^2+z^2)\delta m, \quad I_y = \iiint (x^2+z^2)\delta m, \quad I_z = \iiint (x^2+y^2)\delta m$$

$$I_{xy} = \iiint xy\delta m, \quad I_{xz} = \iiint xz\delta m, \quad I_{yz} = \iiint yz\delta m$$

$$I = \begin{pmatrix} I_x & -I_{xy} & -I_{xz} \\ -I_{xy} & I_y & -I_{yz} \\ -I_{xz} & -I_{yz} & I_z \end{pmatrix}$$

Now we must determine the derivatives of the vectors $\vec{v}$ and $\vec{H}$ referred to the rotating body frame of reference. It has been shown that the derivative of an arbitrary vector $\vec{X}$ referred to a rotating body frame having an angular velocity $\vec{\omega}$ can be represented by the following vector identity $\left(\frac{d\vec{X}}{dt}\right)_I = \left(\frac{d\vec{X}}{dt}\right)_B + \vec{\omega} \times \vec{X}$, where the where the subscript I and B refer to the inertial and body fixed frames respectively. Applying this identity to the equation derived before yields:

$$\vec{M} = \left(\frac{d\vec{H}}{dt}\right)_B + (\vec{\omega} \times \vec{H}) = \left(\frac{d\vec{H}}{dt}\right)_I \tag{1.2}$$

1.3.2 The Inertia Tensor

It is understood that the linear and angular velocity vectors $\vec{v}$ and $\vec{w}$ respectively, in the last expressions are expressed in the inertial frame, we try to find them in the body fixed frame, we know that$\vec{r}_I = R\vec{r}_B$ Where:
$\vec{r}_I$ is the position vector in inertial frame. $\vec{r}_B$ is the position vector in body fixed frame. R is the orientation-transformation matrix[8].

$$\vec{r}_I = R\vec{r}_B \leftrightarrow \frac{d\vec{r}_I}{dt} = \frac{dR}{dt}\vec{r}_B + R\frac{d\vec{r}_B}{dt} \tag{1.3}$$

But $\dfrac{d\vec{r}_B}{dt} = 0$ because $\vec{r}_B$ is constant in the body attached frame.

$$\vec{v}_I = \frac{d\vec{r}_I}{dt} = \frac{dR}{dt}\vec{r}_B = \dot{R}R^{-1}\vec{r}_I = \omega \times \vec{r}_I \quad \text{and} \quad R^T = R^{-1}$$

$$\vec{v_I} = (\dot{R}R^T)\vec{r}_I = \omega \times \vec{r}_I \tag{1.4}$$

Let we set $S(\omega) = (\dot{R}R^T)$ is a skew symmetric matrix satisfy

$$S(\omega) = -S^T(\omega) = \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix}$$

because : $d(RR^T)/dt = 0 \Rightarrow \dot{R}R^T + R\dot{R}^T = 0$. The inertia tensors J and I in the inertial frame and fixed frame respectively are related to each other by similarity transformation $J = RIR^T$. This is an important observation because the inertia matrix expressed in the body attached frame is a constant matrix independent on the motion of the body and easily can be computed.

1.3.3 Lyapunov Direct Method

Lyapunov stability theory deals with the behavior of unforced nonlinear system described by the differential equations

$$\dot{x} = \mathrm{f}(t, x(t)) \tag{1.5}$$

Where:
$t \geq 0, x \in \mathbb{R}^n$, $x(t) \in \mathbb{R}^n$ and $\mathrm{f}(.,.) : \mathbb{R}_+ \times \mathbb{R}^n \to \mathbb{R}$ is continuous.

⊛ The idea behind this theory is that if we consider an "isolated" system in the sense that there are no external forces acting on it, as equation 1.5, where without loss of generality, the origin is an equilibrium point. Then we suppose it is possible to define a function, so that it is zero at the origin and positive everywhere else. And describes, in some sense, the total energy of the system. If the system, originally at equilibrium, is perturbed to a new nonzero initial state, then there are several possibilities. If the system dynamics are such that the energy of the system is nonincreasing with time, then depending on the energy function, this may be sufficiently to conclude that the origin is stable. If the dynamics are such that the energy reduces to zero with time, then it may be sufficient to decide that the equilibrium point is asymptotically stable. Finally, if the dynamics are such that the energy increases beyond its initial values, then it is possible to conclude that the system is unstable[16].

⊛ Lyapunov was able extract from this reasoning a general theory that is applicable to any differential equation. This theory requires one to search for a function that satisfy some prespecified properties. This function is a generalization of the energy of mechanical systems, and is now commonly known as Lyapunov function.

⊛ Lyapunov theory will allow us to determine the stability of particular equilibrium point without actually solving the differential equation 1.5. Moreover, it will provide us with qualitative results the stability questions, which may be used in designing stabilizing controllers for nonlinear systems.

presenting a couple of basic Lyapunov theorems:

Theorem 1.3.1. *[9] The origin of the system 1.5 is uniformly stable, if there exist a continuously differential function C^1 and locally positive definite, $V : \mathbb{R}_+ \times \mathbb{R}^n \to \mathbb{R}$ and a constant $r > 0$ such that*

$$\dot{V}(t, x) \leq 0, \quad \forall t \geq t_0 \quad \textit{and } \forall x \in B_r \tag{1.6}$$

Where is evaluated along the trajectories of 1.5

Theorem 1.3.2. *[9]The origin of the system 1.5 is uniformly stable, if there exist C^1,decrescent, locally positive definite function $V : \mathbb{R}_+ \times \mathbb{R}^n \to \mathbb{R}$ and a constant $r > 0$ such that*

$$\dot{V}(t,x) \leq 0, \quad \forall t \geq 0 \quad \textit{and } \forall x \in B_r \tag{1.7}$$

Remark. the above theorems may be used to design controller that will stabilize a nonlinear system such as a robot manipulator. In fact, if one select a Lyapunov function candidate $V(t,x)$, then finding its total derivative $\dot{V}(t,x)$ will exhibit an explicit dependence on the control signal. By choosing the control signal to make $\dot{V}(t,x)$ negative definite, stability of the closed loop system is guaranteed.

1.3.4 General Rotations:

Theorem 1.3.3. *Euler's Theorem: The most general motion of a rigid body with one point fixed is a rotation about an axis through that point.*

The spacecraft attitude, expressed in terms of the Euler angles roll, pitch, and yaw, is here defined as a rotation of the BRF with respect to the ORF using the rotation matrices defined above. When the ORF is aligned with the ORF the roll, pitch, and yaw angles are zero. We denote by[8]:

$$\Phi = \text{roll angle}, \ \Theta = \text{pitch angle}, \ \Psi = \text{yaw angle}$$

When the body reference frame is not aligned with the orbital reference frame, the yaw, roll, and pitch angles are defined as the sequence of three rotations (Euler sequence) required to align the ORF axes to the BRF axes. The rotation angles are the following:

▷ We rotate by Ψ (Yaw) about the z axis we obtain (x',y',z').
▷ We rotate by Θ (Pitch) about the y axis we obtain (x",y",z").
▷We rotate by Φ (Roll) about the x axis we obtain (x,y,z).

The transformation matrix from ORF to BRF defined by the sequence of these rotations can be easily computed as the product of three consecutive single axis rotation matrices

$$R(\Phi,\Theta,\Psi) = R(\Phi)R(\Theta)R(\Psi) \tag{1.8}$$

$$R_{bo}(\Phi,\Theta,\Psi) = R_x(\Phi)R_y(\Theta)R_z(\Psi) = \begin{pmatrix} c_\Theta c_\Psi & c_\Theta s_\Psi & -s_\Theta \\ s_\varphi s_\Theta c_\Psi - c_\Phi s_\Psi & s_\Phi s_\Theta s_\Psi + c_\Phi c_\Psi & s_\Phi c_\Theta \\ c_\Phi s_\Theta c_\Psi + s_\Phi s_\Psi & c_\Phi s_\Theta s_\Psi - s_\Phi c_\Psi & c_\Phi c_\Theta \end{pmatrix}$$

Remark. ORF and BRF stand for Orbital and Body reference frame respectively.

1.3.5 Lagrangian Function

With T, V being the kinetic energy and the potential energy respectively. A Lagrangian is defined by $L = (T - V)$ and we can write[8]:

$$\left\{\frac{d}{dt}\left[\frac{\partial T}{\partial \dot{q}}\right] - \frac{\partial T}{\partial q}\right\} + \frac{\partial V}{\partial q} = 0 \Longleftrightarrow \frac{d}{dt}\left[\frac{\partial (T-V)}{\partial \dot{q}}\right] - \frac{\partial (T-V)}{\partial q} = 0$$

$$\frac{d}{dt}\left[\frac{\partial L}{\partial \dot{q}}\right] - \frac{\partial L}{\partial q} = 0 \tag{1.9}$$

When the system is forced by an external force F we obtain:

$$\frac{d}{dt}\left[\frac{\partial L}{\partial \dot{q}}\right] - \frac{\partial L}{\partial q} = F \tag{1.10}$$

With quantities q are the generalized coordinates.

1.4 Summary

The chapter was meant to clarify some background knowledge to prep us for the chapters to come. we exhibited satellite control related like attitude control, orbit control..etc, and showed the necessary but not common mathematical preliminaries such as rigid body's equation of motion, general rotation, Lyapunov function inertia tensor, Lagrangian function.

Chapter 2

Satellite's Dynamic modeling

2.1 Introduction

A spacecraft's attitude is generally described by the orientation of principal body axes fixed to a rigid body relative to a reference frame that may either be stationary (inertial) or rotating at a small, constant rate(orbital) [4].
this chapter is concerned with modeling the dynamics of the satellite using mathematical based strategies. We first discuss the attitude model in the inertial and the orbital frame. We further investigate the satellite's orbital motion where we simulate and solve the non-linear differential equations using MATLAB-Simulink. Finally we linearize the latter and get the state space system.

2.2 Modeling Decomposition

When dealing with satellite attitude and orbit control, one of the first design issue is the formulation of spacecraft dynamics. According to classical approach, rigid body motion can be decomposed into two parts:

- Attitude kinematics and dynamics, described by Euler parameters or Euler angles;
- Orbital motion, which depending on position and velocity of the satellite center of mass.

2.3 Attitude Kinematics in inertial frame

The problem of the attitude control design of an artificial satellite lies in the definition of a control law such that the attitude of the reference trajectory is obtained and maintained with certain tolerances. The torques, both predictable and unpredictable, will have to be compensated by the on-board actuation systems like thrusters (with chemical, cold gas or ion propellant), reaction or momentum wheels, control moment gyros, magnetic actuators, orientable solar sails, etc.

The rigid-body kinetics (as seen in previous chapter) gives the angular momentum vector H as $H = I_{CG}\omega$ or a moment itself $M = H = I_{CG}\dot{\omega}$, but in the inertial frame we have:

$$\left(\frac{d\vec{X}}{dt}\right)_I = \left(\frac{d\vec{X}}{dt}\right)_B + \vec{\omega} \times \vec{X} \Longleftrightarrow \tau = (I_{CG}\dot{\omega})_B + \omega \times (I_{CG}\omega) \tag{2.1}$$

In this equation, the subscript "I" indicates a derivative in the inertial frame, while the subscript "B" indicates a derivative in the rotating body frame, with $\omega = [\omega_x, \omega_y, \omega_z]^T$ and $I_{CG} = I_{CG}^T > 0$, is the inertia tensor about the center of gravity given by:

$$I_{CG} = \begin{pmatrix} I_x & -I_{xy} & -I_{xz} \\ -I_{xy} & I_y & -I_{yz} \\ -I_{xz} & -I_{yz} & I_z \end{pmatrix} \tag{2.2}$$

The satellite has three controllable moments $\tau = [M_x, M_y, M_z]^T$ which can be generated using different actuators.

The dynamic model can also be written in terms of Euler equation: $\tau = (I_{CG}\dot{\omega})_B - (I_{CG}\omega) \times \omega$, where we have used the fact that $a \times b = -b \times a$. Furthermore, it is possible to find a matrix $S(\omega)$ such that $S(\omega) = -S^T(\omega)$ is skew-symmetric. With other words:

$$\tau = (I_{CG}\dot{\omega})_B + S(\omega)\omega \tag{2.3}$$

The matrix$S(\omega)$ must satisfy the condition:

$$S(\omega)\omega = -(I_{CG}\omega) \times \omega \tag{2.4}$$

A matrix satisfying this condition is:

$$S(\omega) = \begin{pmatrix} 0 & (I_z\omega_z - I_{yz}\omega_y - I_{xz}\omega_x) & (I_{yz}\omega_z - I_y\omega_y + I_{xy}\omega_x) \\ (I_{yz}\omega_y + I_{xz}\omega_x) - (I_z\omega_z) & 0 & (I_x\omega_x - I_{xz}\omega_z - I_{xy}\omega_y) \\ (I_y\omega_y - I_{yz}\omega_z - I_{xy}\omega_x) & (I_{xz}\omega_z + I_{xy}\omega_y - I_x\omega_x) & 0 \end{pmatrix} \tag{2.5}$$

Assuming that X_B, Y_B, Z_B are the principal axes of inertia and performing the vector product, we obtain the expanded form of the Euler equation is:

$$M_x = I_x\dot{\omega}_x + I_{xy}\dot{\omega}_y + I_{xz}\dot{\omega}_z - (I_y - I_z)\omega_y\omega_z - I_{yz}(\omega_z^2 - \omega_y^2) - \omega_x(\omega_z I_{xy} - \omega_y I_{xz}) \tag{2.6}$$

$$M_y = I_{xy}\dot{\omega}_x + I_y\dot{\omega}_y + I_{yz}\dot{\omega}_z - (I_z - I_x)\omega_x\omega_z + I_{xz}(\omega_x^2 - \omega_z^2) - \omega_y(\omega_x I_{yz} - \omega_z I_{xy}) \tag{2.7}$$

$$M_z = I_{xz}\dot{\omega}_x + I_{yz}\dot{\omega}_y + I_z\dot{\omega}_z - (I_x - I_y)\omega_x\omega_y - I_{xy}(\omega_y^2 - \omega_x^2) - \omega_z(\omega_y I_{xz} - \omega_x I_{yz}) \tag{2.8}$$

In fundamental mechanics it is well-known that a principal axes frame is a body-fixed frame in which the moment of inertia matrix is diagonal, i.e

$$I_{CG} = \begin{pmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{pmatrix} \tag{2.9}$$

Now we deduce the following nonlinear coupled state space equation:

$$\begin{cases} M_x = I_x\dot{\omega}_x + \omega_y\omega_z(I_z - I_y) \\ M_y = I_y\dot{\omega}_y + \omega_x\omega_z(I_x - I_z) \Longleftrightarrow \\ M_z = I_z\dot{\omega}_z + \omega_x\omega_y(I_y - I_x) \end{cases} \tag{2.10}$$

$$\begin{cases} \dot{\omega}_x = (I_y - I_z I_x)\omega_y\omega_z + \frac{M_x}{I_x} \\ \dot{\omega}_y = (I_z - I_x I_y)\omega_x\omega_z + \frac{M_y}{I_y} \Longleftrightarrow \dot{x} = f(x,t) + g(u,t) \\ \dot{\omega}_z = (I_x - I_y I_z)\omega_x\omega_y + \frac{M_z}{I_z} \end{cases} \tag{2.11}$$

2.3.1 Satellite Model Stability Properties

Consider the Lyapunov function candidate $V = \frac{1}{2}\omega^T I_{CG}\omega + h(\theta)$ where $h(\theta)$ is a positive definite function depending on the attitude. Differentiation of V with respect to time

gives:

$$\begin{aligned}\dot{V} &= \omega^T I_{CG}\dot{\omega} + \dot{\theta}^T \frac{\partial h(\theta)}{\partial \theta} \\ &= \omega^T(\tau - S(\omega)\omega) + \omega^T J^T(\theta)\frac{\partial h(\theta)}{\partial \theta}\end{aligned} \tag{2.12}$$

Since $S(\omega) = -S^T(\omega)$; it follows that: $\omega^T S(\omega)\omega = 0 \quad \forall \quad \omega$

This suggests that the control input τ should be chosen as:

$$(\dot{V} = \omega^T(\tau + J(\theta))\frac{\partial h(\theta)}{\partial \theta}) \leqslant 0 \tag{2.13}$$

One control law satisfying this is:

$$\tau = -K_d\omega - J^T(\theta)\frac{\partial h(\theta)}{\partial \theta} \tag{2.14}$$

Where $K_d > 0$: This finally gives: $\dot{V} = -\omega^T K_d \omega \leqslant 0$, and stability and convergence follow from standard Lyapunov techniques.

Remark. : If the Euler angles Φ Θ,and Ψ of the orbiting frame (see next section) are used to represent attitude,the kinematics of the satellite become: $\dot{\theta} = J(\theta)\omega$, where $\theta = [\Phi, \Theta, \Psi]^T$ and:

$$J(\theta) = \begin{pmatrix} 1 & \sin(\Phi)\tan(\Theta) & \cos(\Phi)\tan(\Theta) \\ 0 & \cos(\Phi) & -\sin(\Phi) \\ 0 & \sin(\Phi)/\cos(\Theta) & \cos(\Phi)/\cos(\Theta) \end{pmatrix} \tag{2.15}$$

2.4 Orbital Frame Attitude Dynamics

For these types of spacecraft, it is useful to have one of the spacecraft body axes pointing down (called the nadir direction). Let us define an orbiting reference frame F_0 with basis vectors $\vec{x_0}$, $\vec{y_0}$ and $\vec{z_0}$ as follows:

$$\vec{x_0} = \vec{y_0} \times \vec{z_0}, \quad \vec{y_0} = -\frac{\vec{r} \times \vec{v}}{||\vec{r} \times \vec{v}||} \quad and \quad \vec{z_0} = -\frac{\vec{r}}{||\vec{r}||}$$

Where $\vec{r}$ and $\vec{v}$ are the inertial spacecraft position and velocity relative to the center of the Earth respectively. $\vec{x_0}$= roll axis ; $\vec{y_0}$= pitch axis; $\vec{z_0}$= yaw axis.

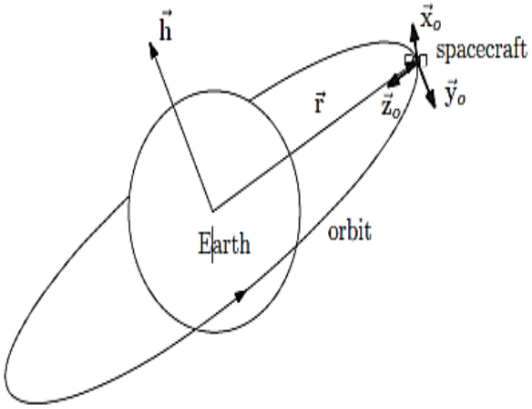


Figure 2.1: Satellite Orbital frame

If the spacecraft body frame F_B is aligned with the orbiting frame F_0, then the body z-axis points directly to nadir as desired for an Earth pointing satellite. The orbiting frame F_0 defines the desired attitude for the spacecraft. It is therefore of interest to study the attitude motion of the spacecraft relative to F_0.

Let us describe the spacecraft attitude relative to the orbiting frame by a $3-2-1$ attitude sequence (see previous chapter). Namely, yaw angle which is a rotation of Ψ about the original z-axis, and pitch angle which is a rotation of Θ about the intermediate y-axis, lastly a roll angle which is a rotation of φ about the transformed x-axis.

$$R_{bo}(\Phi,\Theta,\Psi) = R_x(\Phi)R_y(\Theta)R_z(\Psi) = \begin{pmatrix} c_\Theta c_\Psi & c_\Theta s_\Psi & -s_\Theta \\ s_\Phi s_\Theta c_\Psi - c_\Phi s_\Psi & s_\Phi s_\Theta s_\Psi + c_\Phi c_\Psi & s_\Phi c_\Theta \\ c_\Phi s_\Theta c_\Psi + s_\Phi s_\Psi & c_\Phi s_\Theta s_\Psi - s_\Phi c_\Psi & c_\Phi c_\Theta \end{pmatrix}$$

Where: $s_b = \sin(b) \quad and \quad c_b = \cos(b)$. The attitude kinematics relative to the orbiting frame are then given by:

$$\begin{cases} \vec{\omega}_{OB} = (J(\theta))^{-1}\vec{\omega}_{BO} \\ where \;\; \vec{\omega}_{BO} = \dot{\theta} \end{cases} \iff \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} \begin{pmatrix} 1 & 0 & \sin(\varphi) \\ 0 & \cos(\varphi) & \cos(\Theta)\sin(\varphi) \\ 0 & -\sin(\Theta) & \cos(\Theta)\cos(\varphi) \end{pmatrix} \begin{pmatrix} \dot{\varphi} \\ \dot{\Theta} \\ \dot{\Psi} \end{pmatrix} \iff$$

$$\ddot{\theta} = \frac{dJ(\theta)}{dt}\omega_{OB} + J(\theta)\frac{d\omega_{OB}}{dt} \tag{2.16}$$

Where $\vec{\omega}_{OB}$ is the angular velocity of the body frame F_B relative to the orbiting frame F_0 expressed in body coordinates.

Remark. We tend to use a circular Keplerian Orbit, to avoid the inertial angular velocity of F_0 for more simplicity.

2.5 Orbital Motion (Orbit Model)

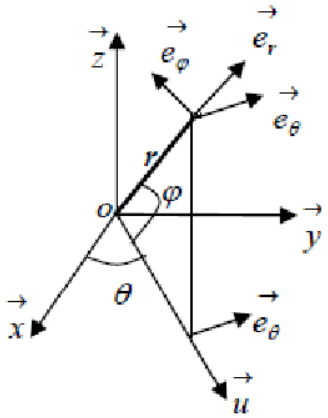
suppose that a satellite of unit mass m orbits the earth at a distance $r(t)$ from its center . Let $\theta(t)$ and $\phi(t)$ be the azimuth and elevation angle of the satellite at time t respectively, and the three forces acting on the satellite are the engines thrusts or forces: $u_r(t)$, $u_\theta(t)$ and u_ϕ with k being gravitational constant of attracting center [5][6][7].

Figure 2.2: Orbiting

Our objective to derive the dynamic model using the Lagrangian function: $L = T - V$.

where $T = m/2[v_x^2 + v_y^2 + v_z^2]$ and $V = km/\sqrt{x^2+y^2+z^2}$,k is a known constant.

Transform these energies from the Cartesian frame geometry to the spherical coordinate frame. From the next figure it is clear that:



$$R = r\cos(\phi) \quad and \quad z = r\sin(\phi),$$

$$x = R\cos(\theta) = r\cos(\theta)\cos(\phi),$$

$$y = R\sin(\theta) = r\sin(\theta)\cos(\phi).$$

We shall now find the expression for the speeds in spherical coordinates

$$\vec{r} = r\vec{e_r} = \mathrm{f}(r,\phi,\theta)$$

$$\vec{v} = \frac{\partial \vec{r}}{dt} = \frac{\partial \vec{r}}{\partial r}\frac{dr}{dt} + \frac{\partial \vec{r}}{\partial \phi}\frac{d\phi}{dt} + \frac{\partial \vec{r}}{\partial \theta}\frac{d\theta}{dt}$$

But we have

$$\frac{\partial \vec{r}}{\partial r}: \begin{cases} \frac{\partial x}{\partial r} = \cos(\phi)\cos(\theta) \\ \frac{\partial y}{\partial r} = \cos(\phi)\sin(\theta), \\ \frac{\partial z}{\partial r} = \sin(\phi) \end{cases} \quad \frac{\partial \vec{r}}{\partial \phi}: \begin{cases} \frac{\partial x}{\partial \phi} = -r\sin(\phi)\cos(\theta) \\ \frac{\partial y}{\partial \phi} = -r\sin(\phi)\sin(\theta), \\ \frac{\partial z}{\partial \phi} = r\cos(\phi) \end{cases} \quad \frac{\partial \vec{r}}{\partial \theta}: \begin{cases} \frac{\partial x}{\partial \theta} = -r\cos(\phi)\sin(\theta) \\ \frac{\partial y}{\partial \theta} = r\cos(\phi)\cos(\theta), \\ \frac{\partial z}{\partial \theta} = 0 \end{cases}$$

Let we set the following notations:

$$\frac{\partial \vec{r}}{\partial r} = \vec{e}_r, \quad \frac{\partial \vec{r}}{\partial \theta} = r\cos(\phi)\vec{e}_\theta, \quad \frac{\partial \vec{r}}{\partial \phi} = r\vec{e}_\phi \tag{2.17}$$

Where

$$\vec{e}_r = \cos(\phi)\cos(\theta)\vec{i} + \cos(\phi)\sin(\theta)\vec{j} + \sin(\phi)\vec{k}$$
$$\vec{e}_\phi = -\sin(\phi)\cos(\theta)\vec{i} - \sin(\phi)\sin(\theta)\vec{j} + \cos(\phi)\vec{k}$$
$$\vec{e}_\theta = -\sin(\theta)\vec{i} + \cos(\theta)\vec{j}$$

Finally we get the velocity equation

$$\vec{v} = \frac{dr}{dt}\vec{e_r} + r\frac{d\phi}{dt}\vec{e_\phi} + r\frac{d\theta}{dt}\cos(\phi)\vec{e_\theta} \tag{2.18}$$

We shall now find the expression for the kinetic and potential energies in spherical coordinates:

$$T = \frac{m}{2}\left[\dot{r}^2 + (r\dot{\phi})^2 + (r\dot{\theta}\cos(\phi))^2\right], \quad \text{and} \quad V = k\frac{m}{r}$$

Hence, in spherical coordinates, the Lagrangian is expressed as:

$$L = \frac{m}{2}\left[\dot{r}^2 + (r\dot{\phi})^2 + (r\dot{\theta}\cos(\phi))^2\right] - k\frac{m}{r} \tag{2.19}$$

It is well-known that:

$\frac{d}{dt}(\frac{\partial L}{\partial \dot{q}} - (\frac{\partial L}{\partial q})) = u(t)$, with q is the generalized coordinate vector and u(t) is the force vector.

In order to evaluate such equation we compute partially its elements 'one by one' and we gather them later on.

$$\begin{cases} \frac{\partial L}{\partial \dot{r}} = m\dot{r} \\ \frac{\partial L}{\partial \dot{\theta}} = m(r\cos(\phi))^2\dot{\theta}, \\ \frac{\partial L}{\partial \dot{\phi}} = mr^2\dot{\phi} \end{cases} \quad \begin{cases} \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{r}}\right) = m\ddot{r} \\ \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) = m(r\cos(\phi))^2\ddot{\theta} + 2m\dot{\theta}(r\cos(\phi))(\dot{r}\cos(\phi)) - r\dot{\phi}\sin(\phi)), \\ \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\phi}}\right) = mr^2\ddot{\phi} + 2m\dot{r}r\dot{\phi} \end{cases}$$

$$\begin{cases} \frac{\partial L}{\partial r} = m(\dot{\theta}\cos(\phi))^2 r - k\frac{m}{r^2} + m(\dot{\phi})^2 r \\ \frac{\partial L}{\partial \theta} = 0, \\ \frac{\partial L}{\partial \phi} = -m(r\theta)^2\cos(\phi)\sin(\phi) \end{cases}$$

The equations of motion can then be shown to be:

$$\begin{aligned}
\frac{d^2r}{dt^2} &= r(t)\dot{\theta}(t)^2\cos^2(\phi) - r(t)\phi(t)^2 - \frac{k}{r(t)^2} + \frac{u_r(t)}{m} \\
\frac{d^2\theta}{dt^2} &= -2\frac{\dot{r}(t)\dot{\theta}(t)}{r(t)} + 2\dot{\phi}(t)\dot{\theta}(t)\frac{\sin(\phi)}{\cos(\phi)} + \frac{u_\theta(t)}{m(r(t)(\cos(\Phi))^2} \\
\frac{d^2\phi}{dt^2} &= -\dot{\theta}(t)^2\cos(\phi)sin(\phi) - 2\frac{\dot{r}(t)\dot{\phi}(t)}{r(t)} + \frac{u_\phi(t)}{m(r(t))^2}
\end{aligned} \tag{2.20}$$

A free (undriven) solution of these equations corresponds to the satellite being in a circular equatorial orbit.

We can simulate the model and find the solution through MATLAB-Simulink. The block diagram is shown in the figure below:

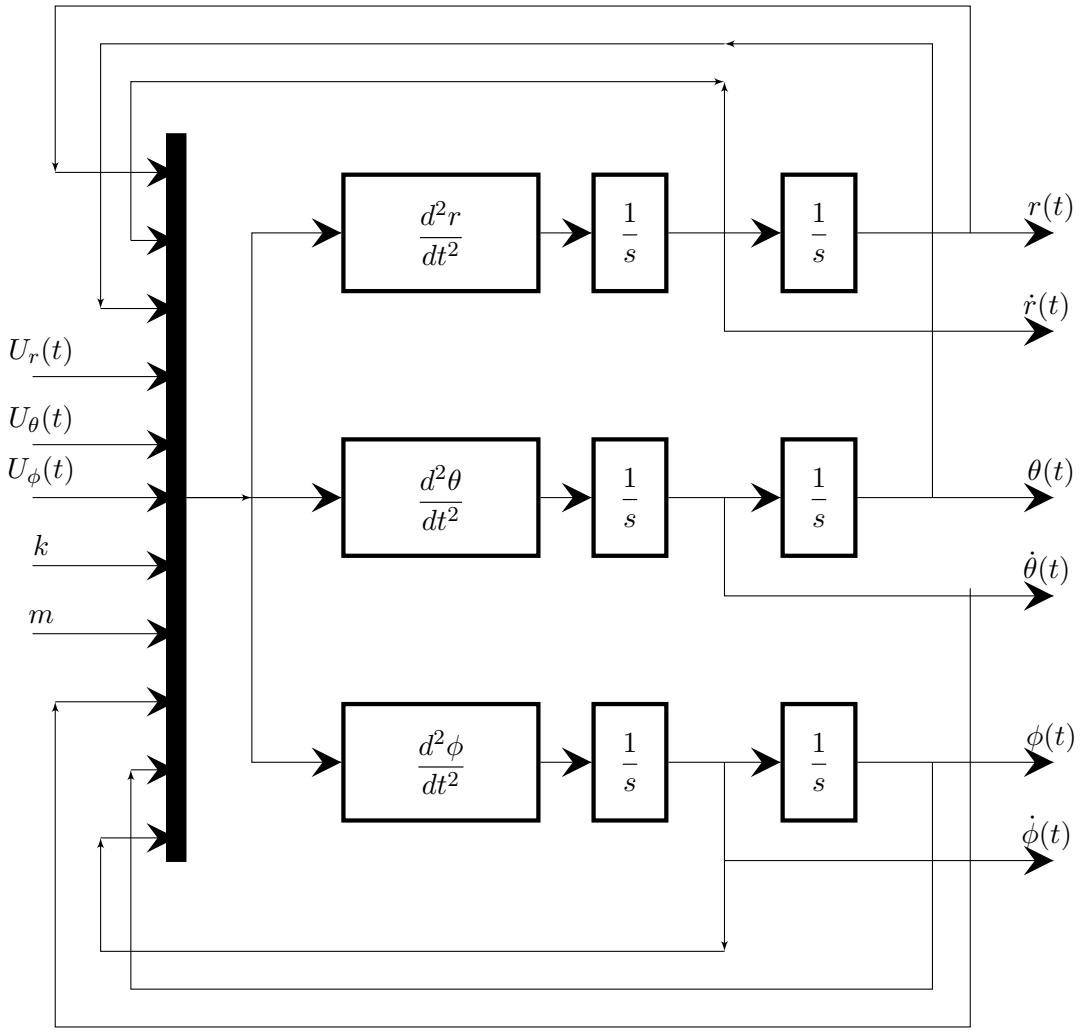


Figure 2.3: Model Block Diagram

2.5.1 Model Linearization

Since the Model is made with Simulink, the latter provides a "**Linear Analysis**" tool which grants us the linearized multi-input, multi-output model.
With $x(t) = \left[r(t), \dot{r}(t), \theta(t), \dot{\theta}(t), \phi(t), \dot{\phi}(t)\right]^T$ as states vector, the input thrusts or forces are written as $u(t) = [u_r(t), u_\theta(t), u_\phi(t)]^T$. the system is linearized around $u(t) = [0, 0, 0]^T$ at $x(t) = [r_0, 0, 0, \theta, 0, 0]^T$ initial condition.
the system matrices are given :

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0.0036 & 0 & 0 & 83.16 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & -1.73e-04 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -0.0036 & 0 \end{bmatrix}, C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0 & 0 \\ 4.57e-04 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 9.52e-10 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 9.52e-10 \end{bmatrix}, \quad D = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Remark. The satellite's parametrs are all obtained from Table A.1 in the appendix section.

2.6 Summary

The chapter presents an in-depth modeling process starting with an inertial frame model derived with angular momentum ant torque. an Orbital frame model is discussed and derived with euler angles accelerations and finishing with an Orbit model where the differential equations are derived through Langragian function, this model is linearized and its state space system is obtained.

Chapter 3

Classical Control Theory

3.1 Introduction

Control theory, a field that plays a major role in nearly every modern precision device. In the classical engineering world, everything from stereos and computers to chemical manufacturing and aircraft utilizes control theory.In a more natural setting, biological systems, even the smallest single-celled creatures, have evolved intricate, life-sustaining feedback mechanisms in the form of biochemical pathways.

The main objective of the theory of control is to design a controlled sculpt for controlling various systems by using optimal methods to maintain the stability. For this purpose, a well-functioned controlled system with the indispensable functionality is needed the most.

In this thesis we are going to be interested in designing a MIMO PID controller using Diophantine equation followed by a n example for further explanation.

3.2 Multi-Input Multi-Output PID Controller

Proportional-integral-derivative (PID) controllers are the most adopted controllers in industrial settings because of the advantageous cost/benefit ratio they are able to provide. Despite their long history and the know-how gained from years of experience, the availability of microprocessors and software tools and the increasing demand for higher product quality at reduced cost have stimulated researchers to devise new methodologies to improve their performance and make them easier to use.

In preparation for our Multi-input Multi-output controller application on our Satellite we must introduce the theory behind the controller design.

3.3 Proportional derivative Integral Control

The **Proportional** control produces an output value that is proportional to the current error value. The proportional response can be adjusted by multiplying the error by a proportional gain K_P. A proportional controller always come with a steady state error e_{ss}. The main function of an **Integral** action is to make sure that the process output agrees with the set-point in steady state, the sum of instantaneous error over time multiplied by the integral gain K_I will eliminate the e_{ss}, but may make the transient response worse. A **Derivative** control improves stability reduces the overshoot and also improves the transient response, it is calculated by determining the slope of the error over time and multiplying this rate of change by the derivative gain K_D.[10]

The "textbook" version of the PID algorithm can be described as:

$$u(t) = K_p \left(e(t) + \frac{1}{K_I} \int_0^t e(\tau)\, d\tau + K_D \frac{de(t)}{dt} \right). \tag{3.1}$$

where u is the control signal and e is the control error $(e = y_d - y)$.

This figure below (3.1) shows a normal PID control system's structure:

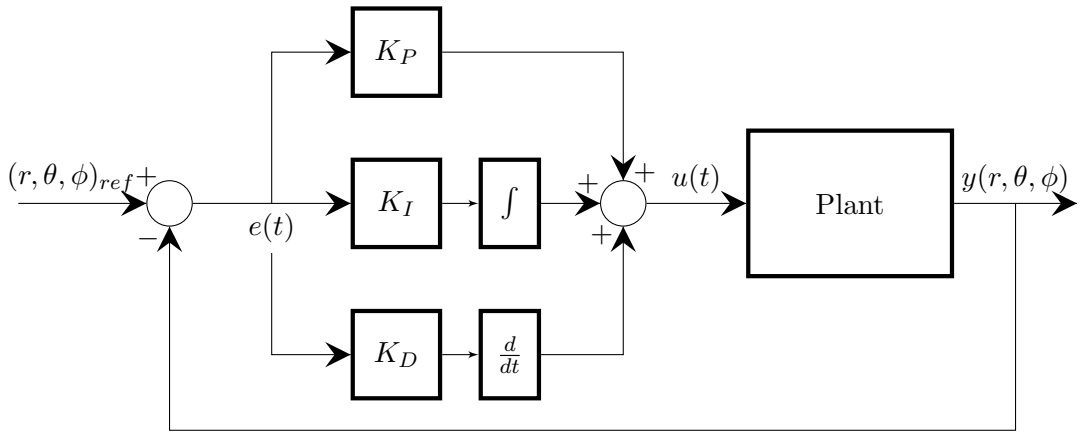


Figure 3.1: A PID Control System

3.4 Matrix polynomials (λ - matrices)

Definition 3.4.1. A matrix polynomial $A(\lambda)$ comes in form of :

$$A(\lambda) = A_0\lambda^l + A_1\lambda^{l-1} + \cdots + A_{l-1}\lambda + A_l \tag{3.2}$$

where $A_i, i = 0, 1, 2, \ldots, l$, is a constant m x m matrix , l represents the degree of a matrix polynomial $A(\lambda)$,while m signifies its order [11].

3.4.1 State space representation

the state equation below (3.3) represents a linear time-invvarient m-input p-output system :

$$\begin{cases} \dot{X}(t) = AX(t) + Bu(t) \\ Y(t) = CX(t) + Du(t) \end{cases} \tag{3.3}$$

where X, Y, u are state, output, and input vectors, respectively; A B C D are matrices of appropriate dimensions.

$$A \in \mathbb{R}^{n\times n} \quad , \quad B \in \mathbb{R}^{n\times m} \quad , \quad C \in \mathbb{R}^{p\times n} \quad , \quad D \in \mathbb{R}^{p\times m}$$

3.4.2 Matrix transfer function

To preform easily on a dynamic MIMO system we can transform its state equation to a specific canonical form [12]. This form i s called matrix fraction description MFD.

$$y(\lambda) = H(\lambda)u(\lambda) \tag{3.4}$$

where:

$$H(\lambda) = D_L^{-1}(\lambda)N_L(\lambda) = N_R(\lambda)D_R^{-1} \tag{3.5}$$

Equations 3.5 are referred to as left matrix fraction description (LMFD), and right matrix fraction description (RMFD) of the system respectively.

3.5 Transformation to block controller form

A system in Eq 3.3 can be transformed to block controller form if and only if :

- $\frac{n}{m} = l$ is an integer.
- The block controllability matrix Ω_c is full rank with :

$$\Omega_c = [B, AB, A^2B, \ldots, A^{l-1}B]$$

If the conditions are satisfied we use the similarity transformation $x_c = T_c x$ with T_c/T_{c1} being the similarity transformation matrix where :

$$Tc = \begin{pmatrix} T_{c1} \\ T_{c1}A \\ \vdots \\ T_{c1}A^{l-1} \end{pmatrix}, \quad T_{c1} = \begin{bmatrix} O_m, O_m, \ldots, I_m \end{bmatrix} \Omega^{-1} \tag{3.6}$$

$$\begin{cases} A_c = T_c A T_c^{-1} = \begin{pmatrix} O_m & I_m & \cdots & O_m \\ O_m & O_m & \cdots & O_m \\ \vdots & \vdots & \cdots & \vdots \\ O_m & O_m & \cdots & I_m \\ -A_l & -A_{l-1} & \cdots & -A_1 \end{pmatrix} \\ B_c = T_c B = \begin{pmatrix} O_m & O_m & \cdots & I_m \end{pmatrix}^T \\ C_c = C T_C^{-1} = \begin{pmatrix} C_l & C_{l-1} & \cdots & C_1 \end{pmatrix} \end{cases} \tag{3.7}$$

Finally we obtain :

$$\begin{aligned} \dot{X}_c &= A_c X_c(t) + B_c u(t) \\ Y(t) &= C_c X_c(t) + D_c u(t) \end{aligned} \tag{3.8}$$

3.6 Canonical right matrix fraction description

supposing the condition $l = \dfrac{n}{m} =$ **integer** is satisfied, we can define the canonical right MFD [11] :

$$\begin{pmatrix} D_l \\ D_{l-1} \\ \vdots \\ D_1 \end{pmatrix} = -(\Omega_c)^{-1} A^l B, \begin{pmatrix} N_l^T \\ N_{l-1}^T \\ \vdots \\ N_1^T \end{pmatrix}^T = C\Omega_c \mathcal{M}_c \tag{3.9}$$

$$D_R(\lambda) = I_m \lambda^l - \mathcal{S}_c \Omega_c^{-1} A^l B \quad \text{and} \quad N_R(\lambda) = C\Omega_c \mathcal{M}_c \mathcal{S}_c \tag{3.10}$$

with :

$$\mathcal{M}_c = \begin{pmatrix} D_{l-1} & \cdots & D_1 & D_0 \\ D_{l-2} & \cdots & D_0 & O_m \\ \vdots & \cdots & \vdots & \vdots \\ D_1 & \cdots & \cdots & O_m \\ D_0 & \cdots & \cdots & O_m \end{pmatrix}, \text{ and } \mathcal{S}_c = \begin{pmatrix} I_m \\ I_m \lambda \\ \vdots \\ I_m \lambda^{l-1} \end{pmatrix} \tag{3.11}$$

3.7 Solvents and Latent-structure

Definition 3.7.1. [13]: The complex number λ_i is called a latent root of matrix polynomial $A(\lambda)$ if it is a solution of the scalar polynomial equation $\det(A(\lambda)) = 0$ the vector p , solution of $A(\lambda_i)p = 0_m$ is called a primary right latent vector associated with λ_i. Similarly the vector q solution of $q^T A(\lambda_i) = 0_m$ is called a primary left latent vector associated with λ_i.

3.7.1 Left and right solvents

Definition 3.7.2. [13]: A right block root is also called solvent of λ-matrices and its an mxm real matrix R . Given the matrix polynomial as in Eq 3.2 whe have :

$$\begin{aligned} R^l + A_1 R^{l-1} + \cdots + A_{l-1} R + A_l = O_m \\ \Leftrightarrow A_R(R) = \sum_{i=0}^{l} A_i R^{l-i} = O_m \end{aligned} \tag{3.12}$$

similarly a left solvent is an mxm real matrix L thus :

$$\begin{aligned} L^l + A_1 L^{l-1} + \cdots + A_{l-1} L + A_l = O_m \\ \Leftrightarrow A_L(L) = \sum_{i=0}^{l} A_i L^{l-1} = O_m \end{aligned} \tag{3.13}$$

Remark. :

- Solvents of a matrix polynomial do not always exist.
- Generalized right (left) eigenvectors of a right (left) solvent are the generalized latent vectors of the corresponding matrix polynomial.

3.7.2 Complete set of solvents

Theorem 3.7.1. *[13] Consider the set of solvents $\{R_1, R_2, \ldots, R_l\}$ constructed from the eigenvalues $(\lambda_1, \lambda_2, \ldots, \lambda_l)$ of a matrix w_p. $\{R_1, R_2, \ldots, R_l\}$is a complete set of solvents if and only if :*

1. $\cup\sigma(R_i) = \sigma(w_p)$
2. $\sigma(R_i) \cap \sigma(R_j) = \emptyset$ *for* $i \neq j$ *and* $i = 1, 2, \ldots, l$ *and* $j = 1, 2, \ldots, l$
3. $det(V(R_1, R_2, \ldots, R_l)) \neq 0$

Where: σ is the spectrum of the matrix. V_R right Vandermonde matrix corresponding to $\{R_1, R_2, \ldots, R_l\}$ given as

$$V_R(R_1, R_2, \ldots, R_l) = \begin{pmatrix} I_m & I_m & \cdots & I_m \\ R_1 & R_2 & \cdots & R_l \\ \vdots & \vdots & \cdots & \vdots \\ R_1^{l-1} & R_2^{l-1} & \cdots & R_l^{l-1} \end{pmatrix} \tag{3.14}$$

Remark. : We can define a set of left solvents in the same way as the right solvent.

3.8 Matrix polynomial coefficients construction

We want to construct the matrix polynomial defined by $A(\lambda)$ from a set of right or left block roots. Suppose we have a desired complete set of solvents, the problem is to find the desired polynomial matrix or the characteristic equation of the block controller or block observer. If a complete set of l (right or left) solvents exist then we can determine, from this set, the matrix coefficients $A_i (i = 0, 1, \cdots, l)$

1. For each right solvent R_j of $A(\lambda)$ we have:

$$\begin{aligned} & R_j^l + A_1 R_j^{l-1} + \cdots + A_{l-1} R_j + A_l = O_m \\ \Leftrightarrow\ & A_1 R_j^{l-1} + \cdots + A_{l-1} R_j + A_l = -R_j^l, \quad j = 1, 2, \cdots, l \end{aligned} \tag{3.15}$$

   Or in vectorial form we can write

$$[A_l, A_{l-1}, \cdots, A_1] \begin{pmatrix} I_m \\ R_j \\ \vdots \\ R_j^{l-1} \end{pmatrix} = -R_j^l \quad j = 1, 2, \cdots, l \tag{3.16}$$

   If we collect all the set of equations $j = 1, 2, \cdots, l$ we get the more compact form:

$$[A_l, A_{l-1}, \cdots, A_1] = -[R_1^l, R_2^l, \cdots, R_l^l] V_R^{-1} \tag{3.17}$$

   with V_R has the shown format in Eq 3.14

2. For each left solvent L_j of $A(\lambda)$ we have

$$\begin{bmatrix} A_l \\ A_{l-1} \\ \vdots \\ A_1 \end{bmatrix} = -V_L^{-1} \begin{bmatrix} L_1^l \\ L_2^l \\ \vdots \\ L_l^l \end{bmatrix} \tag{3.18}$$

   With

$$V_L = \begin{pmatrix} I_m & L_1 & L_1^2 & \cdots & L_1^{l-1} \\ I_m & L_2 & L_2^2 & \cdots & L_2^{l-1} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ I_m & L_l & L_l^2 & \cdots & L_l^{l-1} \end{pmatrix}$$

3.9 MIMO PID Control and Diophantine Equation

The polynomial Diophantine matrix equation (PDME) plays a very important role in the analysis and design of control systems such as compensators design , multivariable adaptive feedback systems , predictive control, robust controllers design, etc.

Our purpose of this section here is the dynamic compensator design which relates inputs to outputs when the states are not measurable, the proposed compensator is of special type called three actions or MIMO PID controller[11] based on the solution of Diophantine equation to relocate some desired Block roots of matrix polynomial achieving needed control performances.

Given a system described by right or left MFD as

$$H(\lambda) = N_R(\lambda)D_R^{-1}(\lambda) = D_L^{-1}(\lambda)N(\lambda) \tag{3.19}$$

The matrix transfer function of the controller is

$$C(\lambda) = (K\lambda)^{-1}(K_I + K_P\lambda + K_D\lambda^2) = D_c^{-1}(\lambda)N_c(\lambda) \tag{3.20}$$

The control input signal is given by:

$$u(\lambda) = (K\lambda)^{-1}(K_I + K_P\lambda + K_D\lambda^2)e(\lambda) \tag{3.21}$$

Where $e(\lambda = r(\lambda) - y(\lambda))$ is the error between the input and the output. The closed loop transfer matrix is obtained as:

$$\begin{aligned} H_{closed}(\lambda) = N_R(\lambda)\left[D_c(\lambda)D_R(\lambda) + N_c(\lambda)N_R(\lambda)\right]^{-1}N_c(\lambda) \\ \Leftrightarrow H_{closed}(\lambda) = N_R(\lambda)D_f(\lambda)^{-1}N_c(\lambda) \end{aligned} \tag{3.22}$$

The matrix equation $D_f(\lambda)$ is called Diophantine equation where:

$$D_f(\lambda) = D_c(\lambda)D_R(\lambda) + N_c(\lambda)N_R(\lambda) \tag{3.23}$$

Expanding this last equation we get:

$$\begin{pmatrix} D_l^T & N_{l-1}^T & 0 & 0 \\ D_{l-1}^T & N_{l-2}^T & N_{l-1}^T & 0 \\ D_{l-2}^T & N_{l-3}^T & N_{l-2}^T & N_{l-1}^T \\ \vdots & \vdots & \vdots & \vdots \\ D_1^T & N_0^T & N_1^T & N_2^T \\ D_0^T & 0 & N_0^T & N_1^T \\ 0 & 0 & 0 & N_0^T \end{pmatrix} \begin{pmatrix} K_T \\ K_D^T \\ K_P^T \\ K_I^T \end{pmatrix} = \begin{pmatrix} D_{f(l+1)}^T \\ D_{fl}^T \\ \vdots \\ D_{f0}^T \end{pmatrix} \tag{3.24}$$

Remark. : The existence of MIMO PID controller using this procedure depends on the solvability of the last rectangular matrix equation.

Example 3.9.1. Consider A two input two output system in state space form given by

$$A = \begin{bmatrix} -3.8606 & 1.0659 & 3.4621 & -3.4180 \\ -0.8278 & -1.8554 & 3.2787 & -3.6134 \\ -0.2555 & -0.4818 & -1.5920 & 0.8892 \\ -0.2449 & -0.5802 & 1.5167 & -2.6921 \end{bmatrix}, B = \begin{bmatrix} 6.5669 & 4.2779 \\ 2.9019 & 2.6719 \\ 7.5454 & 7.5374 \\ 5.5812 & 8.9838 \end{bmatrix}$$

$$C = \begin{bmatrix} 10.9267 & 14.0747 & 7.9974 & 4.0162 \\ 6.1025 & 3.8314 & 14.3213 & 3.7513 \end{bmatrix}, D = \begin{bmatrix} 0.000 & 0.000 \\ 0.000 & 0.000 \end{bmatrix}$$

Find its corresponding matrix transfer function $H(\lambda)$, then design a MIMO PID controller which will achieve the desired set of latent structure with tracking conditions. The desired set of solvent (Latent structure) is given below[11]

$$R_1 = \begin{bmatrix} -3.9914 & -0.1381 \\ 0.0627 & -5.0086 \end{bmatrix}, R_2 = \begin{bmatrix} -2.9657 & 0.4224 \\ -0.3279 & -7.0343 \end{bmatrix}, R_3 = \begin{bmatrix} -7.9647 & -0.3234 \\ 0.1129 & -9.0353 \end{bmatrix}$$

The numerator $N_R(\lambda)$ and denominator $D_R(\lambda)$ of the proper rational matrix transfer function $H(\lambda)$ are conducted according to the next Matlab statement.

```
1 Omega = [B A*B]; Ds = -inv(Omega)*(A^2)*B;
2 D0 = Ds(1:2,:); D1 = Ds(3:4,:); D2 = eye(2,2);
3 Mc = [D1 D2;.. D2 0]; Ns = C*Omega*Mc;
4 N0=Ns(:,1:2); N1=Ns(:,3:4).
```

Note: Ds stands for$D(\lambda)$ and Ns stands for $N(\lambda)$.

Now we obtain the matrix transfer function of dynamic system given in its RMFD form:

$$H(\lambda) = N_R(\lambda)D_R^{-1} = \left(\sum_{i=0}^{2} N_i\lambda^i\right)\left(\sum_{i=0}^{2} D_i\lambda^i\right)^{-1}$$

where :

$$D_0 = \begin{bmatrix} 3.6640 & -6.2621 \\ -0.2896 & 7.0461 \end{bmatrix}, D_1 = \begin{bmatrix} 4.2082 & -2.1862 \\ -0.0377 & 5.7919 \end{bmatrix}, D_2 = \begin{bmatrix} 1.000 & 0.000 \\ 0.000 & 1.000 \end{bmatrix}$$

$$N_0 = \begin{pmatrix} 120.5373 & 78.3892 \\ 463.9935 & 314.0497 \end{pmatrix}, N_1 = \begin{bmatrix} 195.3567 & 180.7099 \\ 180.1895 & 177.9893 \end{bmatrix}, N_2 = \begin{bmatrix} 0.000 & 0.000 \\ 0.000 & 0.000 \end{bmatrix}$$

Let we define now the following matrices:

$$V_R = \begin{pmatrix} I_2 & I_2 & I_2 \\ R_1 & R_2 & R_3 \\ R_1^2 & R_2^2 & R_3^2 \end{pmatrix}, \quad M = \begin{pmatrix} D_2 & D_1 & D_0 & O_2 \\ N_1 & N_0 & O_2 & O_2 \\ O_2 & N_1 & N_0 & O_2 \\ O_2 & O_2 & N_1 & N_0 \end{pmatrix}$$

Construction of the desired matrix polynomial coefficients form those Block spectral data as we have seen previously in this chapter (Eq 3.17):

$$[D_{f2}, D_{f1}, D_{f0}] = -[R_1^3, R_2^3, R_3^3]V_R^{-1} \text{ and } D_{f3} = I_2$$

Here in this example the matrix M is full rank square matrix then:

$$[K, K_D, K_P, K_I] = [D_{f3}, D_{f2}, D_{f1}, D_{f0}]M^{-1}$$

$$[K, K_D, K_P, K_I] = \left(I_2 \ \vdots \ -[R_1^3, R_2^3, R_3^3] \begin{pmatrix} I_2 & I_2 & I_2 \\ R_1 & R_2 & R_3 \\ R_1^2 & R_2^2 & R_3^2 \end{pmatrix}^{-1} \right) \begin{pmatrix} D_2 & D_1 & D_0 & O_2 \\ N_1 & N_0 & O_2 & O_2 \\ O_2 & N_1 & N_0 & O_2 \\ O_2 & O_2 & N_1 & N_0 \end{pmatrix}$$

Finally the PID parameters are:

$$K = \begin{bmatrix} 1.2548 & 0.7286 \\ -20.4771 & -20.4111 \end{bmatrix}, K_D = \begin{bmatrix} 0.0389 & -0.0436 \\ -0.0966 & 0.2183 \end{bmatrix}$$

$$K_P = \begin{bmatrix} 0.1882 & -0.1352 \\ -0.4025 & 0.5964 \end{bmatrix}, K_I = \begin{bmatrix} 0.1945 & 0.0210 \\ -1.2949 & 1.3305 \end{bmatrix}$$

and the MIMO PID compensator equation is

$$C(\lambda) = \begin{bmatrix} 0.2831 & 0.1308 \\ -0.2205 & -0.1964 \end{bmatrix} \frac{1}{\lambda} + \begin{vmatrix} 0.0677 & -0.0684 \\ -0.0632 & 0.0579 \end{vmatrix} \lambda + \begin{bmatrix} 0.3318 & -0.2175 \\ -0.3132 & 0.1889 \end{bmatrix}$$

The corresponding MIMO PID compensator can be designed using the following MATLAB statements:

```
clear all; clc
A = [-3.8606  1.0659  3.4621  -3.4180;
     -0.8278  -1.8554  3.2787  -3.6134;
     -0.2555  -0.4818  -1.5920  0.8892;
     -0.2449  -0.5802  1.5167  -2.6921];
B = [6.5669  4.2779;2.9019  2.6719;
     7.5454  7.5374;  5.5812  8.9838];
C = [10.9267  14.0747  7.9974  4.0162;
     6.1025  3.8314  14.3213  3.7513];
D=zeros(2,2);I=eye(2,2);O=zeros(2,2);
Omega_c = [B A*B];Ds = -inv(Omega_c)*(A^2)*B;
D0=Ds(1:2,:); D1=Ds(3:4,:);Mc =[D1 I;I O];
Ns=C*Omega_c*Mc;N0=Ns(:,1:2);N1=Ns(:,3:4);
D2 = I;
s=tf('s');
H=(N1*s+N0)*inv(I*s^2+D1*s+D0);
Ts = 0.1; N = 60; t = [0:Ts:N]'; n = length(t);
u = 10*[1.5*(1.5-1.5*exp(-0.1*t)).*sin(0.3*t),
     3.0*(1-exp(-0.2*t)).*cos(-0.2*t+30)];
u1 = u(:,1); u2 = u(:,2);
R1 = [-3.9914  -0.1381;0.0627  -5.0086];
R2 = [-2.9657  0.4224;-0.3279  -7.0343];
R3 = [-7.9647  -0.3234;0.1129  -9.0353];
Vr=[I I I;R1 R2 R3;R1^2 R2^2 R3^2];
Df=-[R1^3 R2^3 R3^3]*inv(Vr);
```

```
Df3 = eye(2, 2); Df2 = Df(:, 5:6);
Df1 = Df(:,3:4);Df0 = Df(:,1:2);
M = [D2 D1 D0 O;N1 N0 O O;O N1 N0 O;O O N1 N0];
PID = [Df3, Df2, Df1, Df0]*inv(M);
K = PID(:,1:2);Kd = PID(:,3:4);
Kp = PID(:,5:6);Ki = PID(:,7:8);
PID_S=(inv(K)/s)*(Ki+Kp*s+Kd*s^2);
T_f=H*PID_S;
H_cl=feedback(T_f,I,-1); H_cl=minreal(H_cl);
y=lsim(H_cl,u,t);y1=y(1:n,1);y2=y(1:n,2);e=u-y;
plot(t,y1,'-r','linewidth',1.5),grid on,hold on
plot(t,u1,'-b','linewidth',1.5)
plot(t,y2,'-k','linewidth',1.5),grid on,hold on
plot(t,u2,'-g','linewidth',1.5)
xlabel('time');ylabel('magnitude Y1 Y2');
title('The output signals');
legend('y1','u1','y2','u2');
figure;plot(t,e,'-','linewidth',1.5),grid on
xlabel('time');ylabel('magnitude E1 E2');
title('The error signals');legend('e1','e2')
```

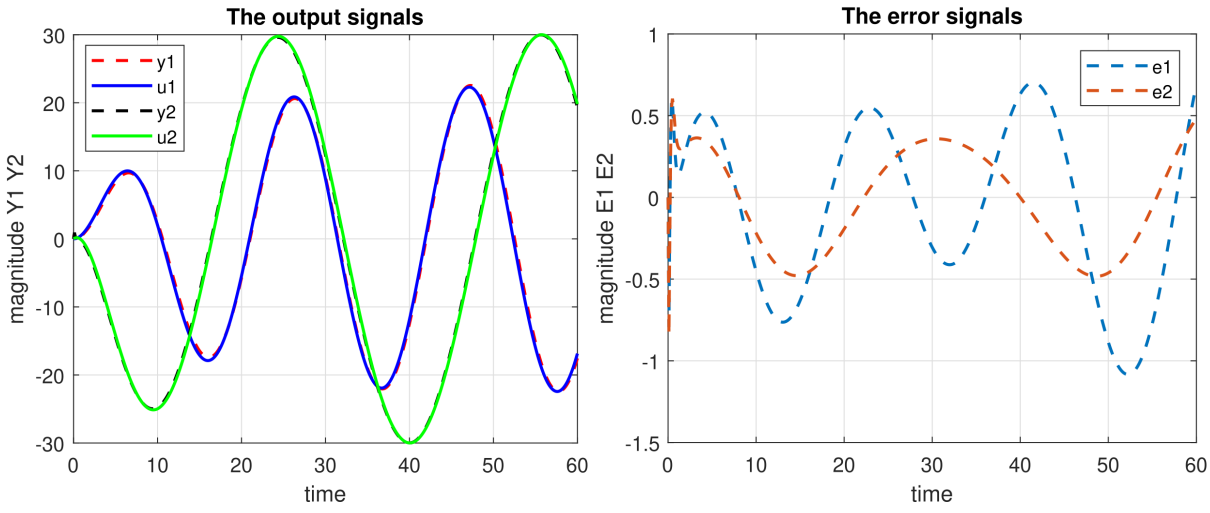


Figure 3.2: Trajectory tracking control of MIMO PID controller

The following case study illustrates best tracking , regulation and robustness with no oscillation and the ability of this MIMO PID controller to robustly maintain best dynamic performance and matching some desired latent structures. From the results obtained, in the above figures we see that the plant outputs coincides with its reference, no excess is recorded in both transient and permanent regimes which are well shown by the error signals. Finally the global stability is guaranteed because all desired Block roots are stable matrices having specific latent roots latent vectors which implies large design degree of freedom and/or much more flexibility in syntheses.

3.10 Summary

This chapter contained a basic knowledge about MIMO systems that lead into developing the theory behind our designed and tested PID controller with the goal of exploiting it throughout the next chapters.

Chapter 4

Nonlinear Control Theory

4.1 Introduction

Real world systems are inherently nonlinear in nature at least when considered over wide operating range although many of the systems are assumed to "behave" as linear in the vicinity of a certain operating point at low speeds under certain assumptions. Many physical processes are represented by nonlinear models. Examples include; coulomb friction, gravitational and electrostatic attraction... etc. Recently, many researchers from such broad areas like process control, bio-medical engineering, robotics, aircraft and spacecraft control have shown an active interest in the design and analysis of nonlinear control strategies. Thus, most of the real problems necessitate invariably bumping into nonlinearities[14].

Nonlinear control theory is concerned with the analysis and design of nonlinear control systems. When engineers analyze and design nonlinear dynamical systems in electrical circuits, mechanical systems, control systems, and other engineering disciplines, they need to absorb and digest a wide range of nonlinear analysis tools. Our main objective for this chapter is to introduce one of the tools (feedback linearization) linked with examples.

4.2 Feedback Linearization

Feedback linearization is an approach to nonlinear control design which has attracted a great deal of research interest in recent years. The principal of this approach is to

algebraically transform a nonlinear system dynamics into a (fully or partly) linear one, so that linear control techniques can be applied. This differs entirely from conventional linearization (Jacobian linearization.) in that feedback linearization is achieved by exact state transformations and feedback, rather than by linear approximations of the dynamics.

Feedback linearization has been used successfully to address some practical control problems. These include the control of helicopters, high performance aircraft, industrial robots, and biomedical devices. More applications of the methodology are being developed in industry. However, there are also a number of important shortcomings and limitations associated with the feedback linearization approach. Such problems are stiil very much topics of current research[15].

4.3 Lie derivatives

Given a scalar function $h(x)$ and a vector field $\mathbf{f}(x)$, we define a new scalar function $L_{\mathbf{f}}h(x)$, called the lie derivative (or simply, the drevative) of h with respect to $\mathbf{f}$[16].

Definition 4.3.1. [15]: $\mathbf{f}$ is called smooth if $\mathbf{f} \in \mathbb{R}^n$. That is, $\mathbf{f}$ is continuous and all derivatives of all order are continuous.

Definition 4.3.2. [15]: T is a diffeomorphism if T is smooth, and the inverse exists and is also smooth.

Definition 4.3.3. [15]: Let $h : \mathbb{R}^n \to \mathbb{R}$ be a smooth scalar function, and $\mathbf{f}$: $\mathbb{R}^n \to \mathbb{R}^n$ be a smooth vector field on $\mathbb{R}^n$, then the Lie derivative of h with respect to $\mathbf{f}$ is a scalar function defined $L_{\mathbf{f}}h(x) = (\nabla h)\mathbf{f}$ where $\nabla h = \dfrac{\partial h}{\partial x}$

Thus, the lie derivative $L_{\mathbf{f}}h$ is simply the directional derivative of h along the direction of the vector $\mathbf{f}$.

4.4 Input-State Linearization

Consider the problem of designing the control input u for a single-input nonlinear system of the form

$$\dot{x} = \mathbf{f}(x, u)$$

The technique of input-state linearization solves this problem in two steps. First, one finds a state transformation $z = z(x)$ and an input transformation $u = u(x, v)$ so that the nonlinear system dynamics is transformed into an equivalent *linear-time-invariant* dynamics, in the familiar form $\dot{z} = Az + bv$. Second, one uses standard linear techniques (such as pole placement) to design v.

Example 4.4.1. For a better understanding of this approach. Consider the second order system

$$\begin{cases} \dot{x}_1 = -2x_1 + ax_2 + \sin x_1 \\ \dot{x}_2 = -x_2 \cos x_1 + u \cos(2x_1) \end{cases} \tag{4.1}$$

Even though linear control design can stabilize the system in a small region around the equilibrium point (0, 0), it is not obvious at all what controller can stabilize it in a larger region. A specific difficulty is the non-linearity in the first equation, which cannot be directly canceled by the control input u.

However, if we consider the new set of state variables

$$z = T(x) \Leftrightarrow \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ ax_2 + \sin x_1 \end{bmatrix}$$

Then, the new state equations are

$$\begin{cases} \dot{z}_1 = -2z_1 + 2z_2 \\ \dot{z}_2 = -2z_1 \cos z_1 + \cos z_1 \sin z_1 + au \cos(2z_1) \end{cases} \tag{4.2}$$

Note that the new state equations also have an equilibrium point at (0, 0). Now we see that the nonlinearities can be canceled by the control law of the form

$$u = \frac{1}{a \cos(2z_1)}(v - \cos z_1 \sin z_1 + 2z_1 \cos z_1)$$

where v is an equivalent input to be designed (equivalent in the sense that determining v amounts to determining u, and vice versa), leading to a linear input-state relation

$$\begin{cases} \dot{z}_1 = -2z_1 + z_2 \\ \dot{z}_2 = v \end{cases} \tag{4.3}$$

Since the new dynamics is linear and controllable, it is well known hat the linear state feedback control law

$$v = -k_1 z_1 - k_2 z_2$$

We can place the poles anywhere with proper choices of feedback gains. For example, we may choose $v = -2z_2$ and we get the stable closed-loop dynamucs

$$\begin{cases} \dot{z}_1 = -2z_1 + z_2 \\ \dot{z}_2 = -2z_2 \end{cases} \tag{4.4}$$

whose poles are both placed at -2. In terms of the original state x_1 and x_2. this control law corresponds to the original input

$$u = \frac{1}{a\cos(2x_1)}(-2ax_2 - 2\sin x_1 - \cos x_1 \sin x_1 + 2x_1 \cos x_1)$$

The original state x is given from z by

$$\begin{cases} x_1 = z_1 \\ x_2 = \dfrac{z_2 - \sin z_1}{a} \end{cases} \tag{4.5}$$

Since both z_1 and z_2 converge to zero, the original state x converges to zero.

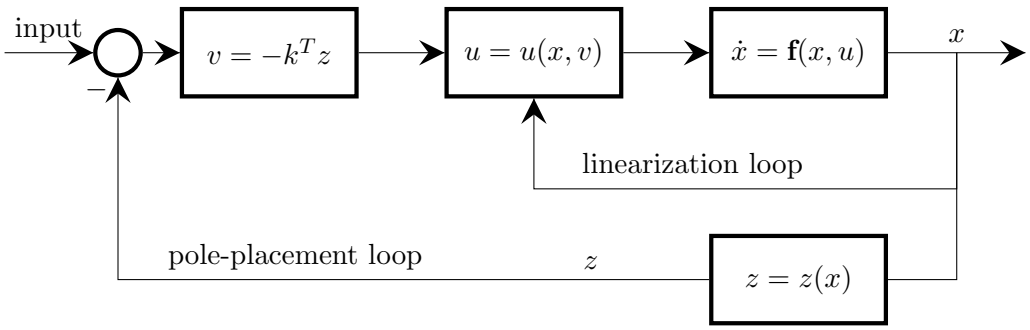


Figure 4.1: Input-State Linearization

The closed-loop system under the above control law is represented in the block diagram in 4.1. We can detect two loops in this control system, with the inner loop achieving the

linearization of the input-state relation, and the outer loop achieving the stabilization of the closed-loop dynamics.

Remark. : Tracking control can also be considered. However, the desired motion then needs to be expressed in terms of the full new state vector.

4.5 Input-Output Linearization

4.5.1 SISO nonlinear systems

In this section, we discuss input-output linearization for single-input nonlinear systems represented by the state and output equations

$$\begin{cases} \dot{x} = \mathbf{f}(x) + \mathbf{g}(x)u \\ y = h(x) \end{cases} \tag{4.6}$$

Where **f, g**, and h are sufficiently smooth in a domain $D \subset \mathbb{R}_n$. The mappings $\mathbf{f} : D \to \mathbb{R}_n$ and $\mathbf{g} : D \to \mathbb{R}_n$ are called vector fields on D [16].

Derive conditions which allow us to transform the system such that the input output map is linear.
the derivative $\dot{y}$ is given by :

$$\dot{y} = \frac{\partial h}{\partial x}[\mathbf{f}(x) + \mathbf{g}(x)u] = L_{\mathbf{f}}h(x) + L_{\mathbf{g}}h(x)u \tag{4.7}$$

Where

$$\frac{\partial h}{\partial x}\mathbf{f}(x) = L_{\mathbf{f}}h(x) \quad \text{and} \quad \frac{\partial h}{\partial x}\mathbf{g}(x) = L_{\mathbf{g}}h(x) \tag{4.8}$$

$L_{\mathbf{f}}h$ is called the Lie Derivative of h with respect to **f**.

if $L_{\mathbf{g}}h(x) = 0$, then $\dot{y} = L_{\mathbf{f}}h(x)$, independent of u. If we continue to calculate the second derivative of y, denoted by $y^{(2)}$, we obtain:

$$y^{(2)} = \frac{\partial (L_{\mathbf{f}}h)}{\partial x}[\mathbf{f}(x) + \mathbf{g}(x)u] = L_{\mathbf{f}}^2 h(x) + L_{\mathbf{g}}L_{\mathbf{f}}h(x)u \tag{4.9}$$

Once again, if $L_{\mathbf{g}}L_{\mathbf{f}}h(x) = 0$, then $y^{(2)} = L_{\mathbf{f}}^2 h(x)$ independent of u .Repeating this process,

we see that if $h(x)$ satisfies

$$\begin{cases} L_{\mathbf{g}} L_{\mathbf{f}}^{i-1} h(x) = 0 & i = 1, 2, \ldots, \rho - 1 \\ L_{\mathbf{g}} L_{\mathbf{f}}^{\rho} h(x) \neq 0 \end{cases} \tag{4.10}$$

Then u does not appear in the equations of $y, \dot{y}, \ldots, y^{(\rho-1)}$ and appears in the equation of $y^{(\rho)}$ with a nonzero coefficient:

$$y^{(\rho)} = L_{\mathbf{f}}^{\rho} h(x) + L_{\mathbf{g}} L_{\mathbf{f}}^{\rho-1} h(x) u \tag{4.11}$$

The forgoing equation shows clearly that the system is input-output linearizable, since the state feedback control

$$u = \frac{1}{L_{\mathbf{g}} L_{\mathbf{f}}^{\rho-1} h(x)} [-L_{\mathbf{f}}^{\rho} h(x) + v]$$

Reduces the input-output map to

$$y^{(2)} = v$$

Which is a chain of ρ integrators. In this case, the integer ρ is called the relative degree of the system.

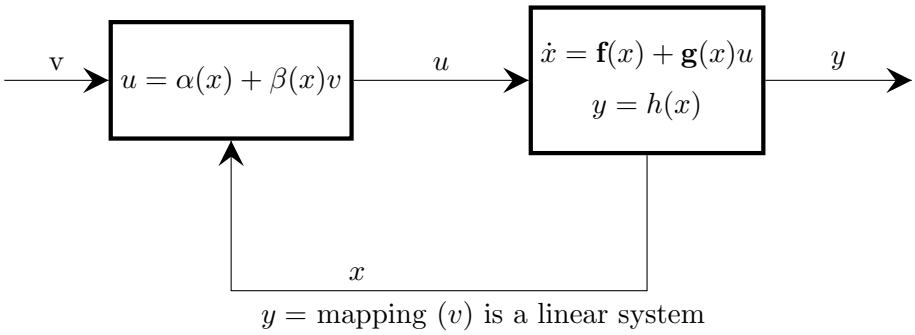


Figure 4.2: Input-Output Lineariezed model

Example 4.5.1. [16] Consider the controlled van der Pol equation

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -x_1 + \epsilon(1 - x_1^2)x_2 + u \end{cases}$$

With output $y = x_1$. Calculating the derivatives of the output, we obtain

$$\begin{cases} \dot{y} = \dot{x}_1 = x_2 \\ \ddot{y} = \dot{x}_2 = -x_1 + \epsilon(1 - x_1^2)x_2 + u \end{cases}$$

Hence, the system has relative degree two in $\mathbb{R}_2$. but for the output $y = x_1 + x_2^2$

$$\dot{y} = x_2 + 2x_2[-x_1 + \epsilon(1 - x_1^2)x_2 + u]$$

Then the new relative degree is one, therefore we conclude that the relative degree is also dependent on the choice of the output. And one can say the system has relative degree one in $D_0 = \{x \in \mathbb{R}^2 | \ x_2 \neq 0\}$

Definition 4.5.1. [15]: If $x = [x_1, x_2, \ldots, x_n]^T$ and $\mathbf{f}= [\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_n]^T$

$$L_{\mathbf{f}}h(x) = \frac{\partial h}{\partial x}\mathbf{f}(x) = \sum_{i=1}^{n} \frac{\partial h}{\partial x_i}\mathbf{f}_i(x) \tag{4.12}$$

$$[\mathbf{f}, \mathbf{g}] = \sum_{i=1}^{n} \frac{\partial \mathbf{g}}{\partial x_i}\mathbf{f}_i(x) \sum_{i=1}^{n} \frac{\partial \mathbf{f}}{\partial x_i}\mathbf{g}_i(x) = L_{\mathbf{f}}\mathbf{g}(x) - L_{\mathbf{f}}\mathbf{g}(x) \tag{4.13}$$

Example 4.5.2. : consider the third-order system

$$\begin{cases} \dot{x}_1 = \sin x_2 + (x_2 + 1)x_3 \\ \dot{x}_2 = x_1^5 + x_3 \\ \dot{x}_3 = x_1^2 + u \end{cases}$$

To generate a direct relationship between the output y and the input u, let us differentiate the output y

$$\dot{y} = \dot{x}_1 = \sin x_2 + (x_2 + 1)x_3$$

Since $\dot{y}$ is still not directly related to the input u, let us differentiate again. We now obtain

$$\ddot{y} = (x_2 + 1)u + \mathbf{f}_1(x)$$

Where $\mathbf{f}_1(x)$ is a function of the state defined by

$$\mathbf{f}_1(x) = (x_1^5 + x_3)(x_3 + \cos x_2) + (x_2 + 1)x_1^2$$

Clearly, the second derivative of output equation represents an explicit relationship between y and u. If we choose the control input to be in the form

$$u = \frac{1}{x_2 + 1}(v - \mathbf{f}_1)$$

Where v is a new input to be determined, and we obtain a simple linear double integrator relationship between the output and the new input v, $\ddot{y} = v$

The design of a tracking controller for this double-integrator relation is simple, because of the availability of linear control techniques. For instance, letting $e = y - y_d Be$ the tracking error, and choosing the new input v as

$$v = \ddot{y}_d - k_1 e - k_2 \dot{e} \Leftrightarrow \ddot{e} + k_1 e + k_2 \dot{e} = 0$$

Which represents an exponentially stable error dynamics.

Remark. :

- The control law is defined everywhere, except at the singularity points such that $x_2 = -1$
- Full state measurement is necessary in implementing the control law, because the computations of both the derivative y and the input transformation require the value of x.

However, one must remember that the error equation only accounts for part of the closed-loop dynamics, because it has only order 2, while the whole dynamics has order 3 (the same as that of the plant, because the controller introduces no extra dynamics). Therefore, a part of the system dynamics (described by one state component) has been rendered "unobservable" in the input-output linearization. This part of the dynamics will be called the **internal dynamics**, because it cannot be seen from the external input-output relationship.

4.5.2 MIMO nonlinear systems

The concepts used in the above sections for SISO systems, such as input-state linearization, input-output linearization, and so on, can be extended to MIMO systems. In the MIMO case, we consider, in a neighborhood of a point x_0, square systems (i.e., systems with the same number of inputs and outputs).

Consider the multi-input-multi-output system

$$\dot{x} = \mathbf{f}(x) + \mathbf{G}(x)u \quad \text{and} \quad y = h(x) \tag{4.14}$$

with

$$y = [y_1, y_2, \ldots, y_p]^T, h = [h_1, h_2, \ldots, h_p]^T, u = [u_1, u_2, \ldots, u_p]^T, x = [x_1, x_2, \ldots, x_n]^T$$

Where x is $n \times 1$ the state vector, u is the $p \times 1$ control input vector, y is the $p \times 1$ of system outputs, $\mathbf{f}$ and h are smooth vector fields, and $\mathbf{G}$ is $n \times p$ matrix whose colums are smooth vector fields $\mathbf{g}_i$[16].

Input-output linearization of MIMO systems is obtained similarly to the SISO case, by differentiating the outputs y_i until the inputs appear. Assume that ρ_i is the smallest integer such that at least one of the inputs appears in $y_i^{(\rho_i)}$ then

$$y_j^{(\rho_j)} = L_{\mathbf{f}}^{\rho_j} h_j(x) + \sum_{i=1}^{p} L_{\mathbf{g}_i} L_{\mathbf{f}}^{\rho_j - 1} h_j(x) u_i \tag{4.15}$$

$\rightarrow$ each output has a specific relative degree (ρ_i)

The nonlinear system is said to have vector relative degree $\{\rho_1, \rho_2, \ldots, \rho_p\}$ at the point x_0 if :

1. $L_{\mathbf{g}i} L_{\mathbf{f}}^k h_i(x) = 0$ for all $1 \leq i, j \leq p$, for all $k < \rho_i - 1$, and fro all x in a neighborhood of x_0(Let Ω_0 be the neighborhood of x_0).

2. the $p \times p$ decoupling matrux (invertibility matrix $D(x)$) is nonsingular at the point x_0

$$D(x) = \begin{pmatrix} L_{\mathbf{g}_1} L_{\mathbf{f}}^{\rho_1 - 1} h_1(x) & \cdots & L_{\mathbf{g}_p} L_{\mathbf{f}}^{\rho_1 - 1} h_1(x) \\ \vdots & \ddots & \vdots \\ L_{\mathbf{g}_1} L_{\mathbf{f}}^{\rho_p - 1} h_j(x) & \cdots & L_{\mathbf{g}_p} L_{\mathbf{f}}^{\rho_p - 1} h_j(x) \end{pmatrix} \tag{4.16}$$

With $L_{\mathbf{g}_i} L_{\mathbf{f}}^k h_i(x) \neq 0$ for at least one j, in a neighborhood Ω_0, of the point x_0 Performing the above procedure dor each output y_j yields

$$\begin{pmatrix} y_1^{(\rho_1)} \\ y_2^{(\rho_2)} \\ \vdots \\ y_p^{(\rho_p)} \end{pmatrix} = \begin{pmatrix} L_{\mathbf{f}}^{\rho_1} h_1(x) \\ L_{\mathbf{f}}^{\rho_2} h_1(x) \\ \vdots \\ L_{\mathbf{f}}^{\rho_p} h_p(x) \end{pmatrix} + D(x) \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_p \end{pmatrix} \tag{4.17}$$

Define then Ω as the intersection of the Ω_0. If, as assumed above, the partial "relative degrees"ρ_i are all well defined, then Ω is itself a finite neighborhood of x_0 . Furthermore,

if $D(x)$ is invertible over the region Ω, then, similarly to the SISO case, the input transformation

$$\begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_p \end{pmatrix} = \begin{pmatrix} L_{\mathbf{g}_1} L_{\mathbf{f}}^{\rho_1 - 1} h_1(x) & \cdots & L_{\mathbf{g}_p} L_{\mathbf{f}}^{\rho_1 - 1} h_1(x) \\ \vdots & \ddots & \vdots \\ L_{\mathbf{g}_1} L_{\mathbf{f}}^{\rho_p - 1} h_j(x) & \cdots & L_{\mathbf{g}_p} L_{\mathbf{f}}^{\rho_p - 1} h_j(x) \end{pmatrix}^{-1} \left[\begin{pmatrix} y_1^{(\rho_1)} \\ y_2^{(\rho_2)} \\ \vdots \\ y_p^{(\rho_p)} \end{pmatrix} - \begin{pmatrix} L_{\mathbf{f}}^{\rho_1} h_1(x) \\ L_{\mathbf{f}}^{\rho_1} h_1(x) \\ \vdots \\ L_{\mathbf{f}}^{\rho_p} h_p(x) \end{pmatrix} \right] \tag{4.18}$$

Yields p equations of the simple form

$$y_i^{(\rho_i)} = v_i$$

Since the input v_i only affects the output y_i, Eq.(4.18) is called a decoupling control law, and the invertible matrix $D(x)$ is called the decoupling matrix of the system in. The system (4.14) is then said to have relative degree $(\rho_1, \rho_2, \ldots, \rho_p)$ at x_0, and the scalar $\rho = \rho_1 + \rho_2 + \cdots + \rho_p$ is called the total relative degree of the system at x_0.

4.6 Summary

In this chapter we have acknowledged the nonlinear control theory, then we revealed one of its tools (Feed-back Linearization), so it can be our second control method in the next chapters.

Chapter 5

Advanced Optimal Control Strategy

5.1 Introduction

Many systems, physical, chemical, and economical, can be modeled by mathematical relations, such as deterministic and/or stochastic differential and/or difference equations. These systems then change with time or any other independent variable according to the dynamical relations. It is possible to steer these systems from one state to another state by the application of some type of external inputs or controls. If this can be done at all, there may be different ways of doing the same task. If there are different ways of doing the same task, then there may be one way of doing it in the "best" way. This best way can be minimum time to go from one state to another state, or maximum thrust developed by a rocket engine. The input given to the system corresponding to this best situation is called "optimal" control. The measure of "best" way or performance is called "performance index" or "cost function." Thus, we have an "optimal control system," when a system is controlled in an optimum way satisfying a given performance index [17].

5.2 Optimal Control

The main objective of optimal control is to determine control signals that will cause a process (plant) to satisfy some physical constraints and at the same time extremize (maximize or minimize) a chosen performance criterion [17].

The formulation of optimal control requires

1. a mathematical description (or model) of the process to be controlled (generally in state variable form),

2. a specification of the performance index,

3. a statement of boundary conditions and the physical constraints on the states and/or controls.

5.3 Optimization

The subject of optimization is quite general in the sense that it can be viewed in different ways depending on the approach (algebraic or geometric), the nature of the signals (deterministic or stochastic), and the stage (single or multiple) used in optimization. The main purpose of optimization is finding the best result possible for a given system. This result can be expressed with either a minimum or maximum. Further, optimization can be classified as static optimization and dynamic optimization [17][18].

- **Static Optimization** is concerned with controlling a plant under steady state conditions.

- **Dynamic Optimization** concerns with the optimal control of plants under dynamic conditions, the plant is described by differential (or difference) equations.

The figure(5.1) below also shows a handful methods thas has been generally used to solve optimization problems.

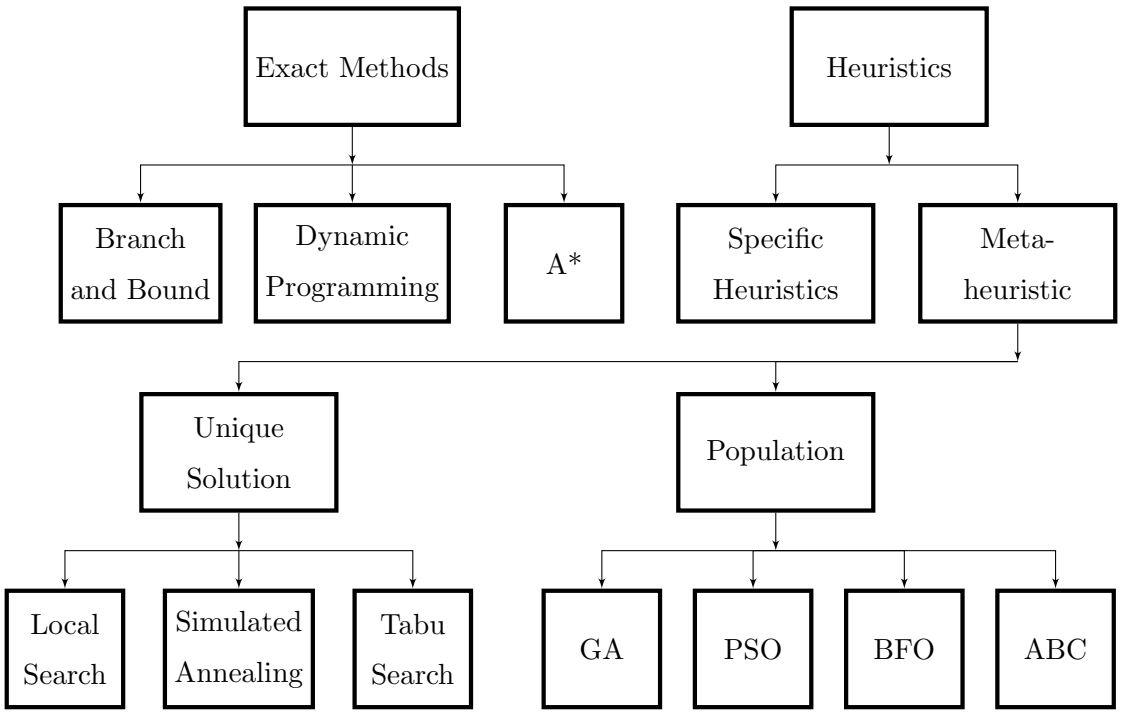


Figure 5.1: Optimization methods in a nature perspective

5.4 Meta-heuristic Methods

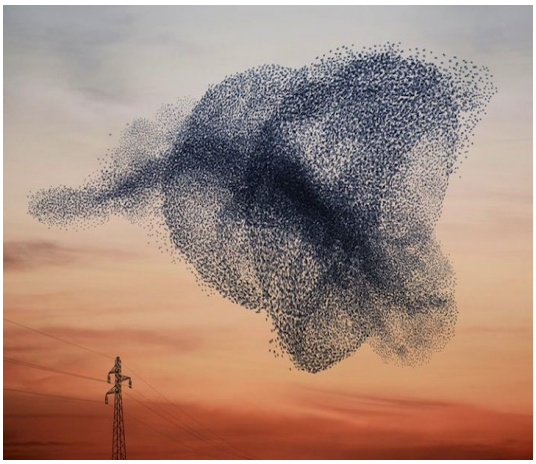
Meta-heuristics are defined as a stochastic iterative methods(or algorithms), inspired by natural and biological phenomena. These methods progress towards an optimum by sampling a function called "objective". We can describe it's behavior as a search method, first by learning the problem's characteristics then finding an approximated result to the best solution. As shown in 5.1 Meta-heuristics can be population-based or unique solution-based and also can do a simple local search or a complex global one.

5.5 Swarm intelligence

Swarm intelligence refers to a class of algorithms that simulates natural and artificial systems composed of many individuals that coordinate using decentralized control and self-organization. The algorithm focuses on the collective behaviors that result from the local interactions of the individuals with each other and with the environment where these individuals stay. Some common examples of systems involved in swarm intelligence are colonies of ants and termites, fish schools, bird flock, animal herds. A typical swarm

intelligence system has the following properties:

1. It consists of many individuals.
2. The individuals are relatively homogeneous (i.e. they are either all identical or they belong to a few typologies).
3. The interactions between the individuals are based on simple behavioral rules that exploit only local information that the individuals exchange directly or via the environment.
4. The global behavior of the system results from the local interactions of individuals with each other and with their environment.



(a) Brirds Flocking

(b) Fish Schooling

Figure 5.2: Representation of Swarm Intelligence

The main property of a swarm intelligence system is its ability to act in a coordinated way without the presence of a coordinator or of an external controller. Despite the lack of individuals in charge of the group, the swarm as a whole does show intelligent behavior. This is the result of interactions of spatially located neighboring individuals using simple rules [19].

5.6 Particle Swarm Optimization (PSO)

Particle swarm optimization has been applied to solve many optimization problems since it was originally developed by Kennedy and Eberhart in 1995. As an alternative of

genetic algorithms , PSO was inspired from research on swarm intelligence such as fish schooling and bird flocking coordinated movements[20].

The algorithm employs a population of search points that moves stochastically in the search space. Concurrently, the best position ever attained by each individual, also called its experience, is retained in memory. This experience is then communicated to part or the whole population, biasing its movement towards the most promising regions detected so far. The communication scheme is determined by a fixed or adaptive social network that plays a crucial role on the convergence properties of the algorithm [21].

5.6.1 PSO and other evolutionary computation Techniques

The most striking difference between PSO and the other evolutionary algorithms is that PSO chooses the path of cooperation over competition. The other algorithms commonly use some form of decimation, survival of the fittest. In contrast, the PSO population is stable and individuals are not destroyed or created. Individuals are influenced by the best performance of their neighbors. Individuals eventually converge on optimal points in the problem domain. In addition, the PSO traditionally does not have genetic operators like crossover between individuals and mutation, and other individuals never substitute particles during the run. Instead, the PSO refines its search by attracting the particles to positions with good solutions [22].

5.6.2 Concept

Fundamentally, PSO was developed as a simulation of flocking birds in bi-dimensional space. The agent(particle) position is represented on X and Y axis with their respective velocities such that V_x is the velocity on X axis and V_y is the velocity on Y.

Each recurrence, the agent proceed via an objective function towards the best value until it reaches ($pbest$) and its position.

This information is obtained by the analysis of personal experiences of each agent.the latter also knows the best global value of the group ($gbest$) amongst all ($pbests$). $gbest$

describes the value which other *pbests* are performing around. each agent changes his position according to the following information:

- Current position (x, y)
- Current velocity (V_x, V_y)
- Distance between current position and *pbest*
- Distance between current position and *gbest*

every modification written as follows [18]:

$$V_i^{k+1} = \mathrm{w}V_i^k + c_1 rand_1 \times (pbest_i - X_i^k) + c_2 rand_2 \times (gbest - X_i^k) \tag{5.1}$$

Where:

V_i^k: Velocity of agent i at iteration k.

w : inertia weight.

c_j : weighting factor.

rand : random value between 0 and 1.

X_i^k :current position of agent i at iteration k.

pbest : best agent i position.

gbest : best global position of all agents.

Generally the inertia weight function in Eq 5.1 is written as :

$$\mathrm{w} = \mathrm{w}_{\max} - \frac{\mathrm{w}_{\max} - \mathrm{w}_{\min}}{iter_{\max}} \times iter \tag{5.2}$$

Where :

$\mathrm{w_m}$ax : final weight,

$\mathrm{w_m}$in : initial weight,

$iter_{\max}$: maximum iterations, $iter$: current iteration value.

The model used in Eq 5.2 is called inertia weight approach (IWA)[22]. The current particle position is changed through the equation below :

$$X_i^{k+1} = X_i^k + V_i^{k+1} \tag{5.3}$$

With:

X_i^k : Current agent position.

X_i^{k+1} : next agent position.

V_k^{k+1} : Next agent velocity.

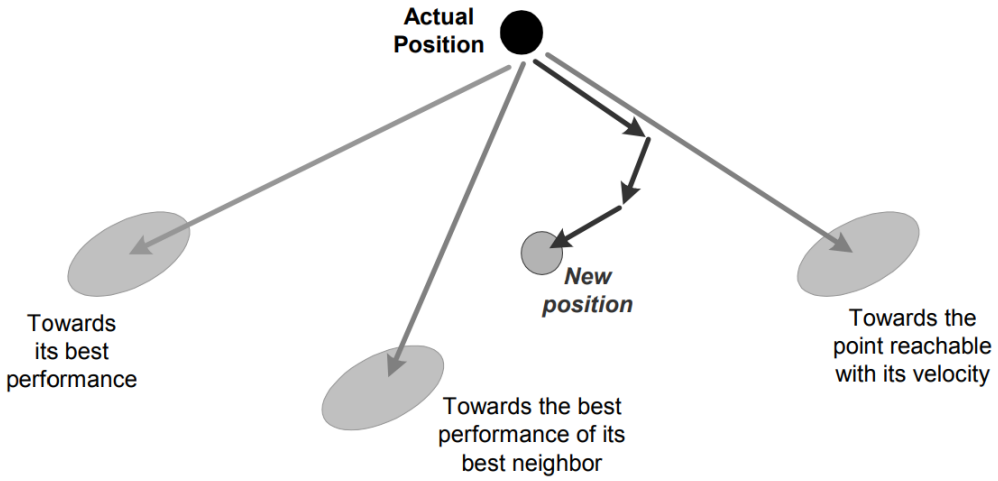


Figure 5.3: Position mechanism of a PSO

This presented PSO concept enables us to construct it's proper algorithm described in the figure below :

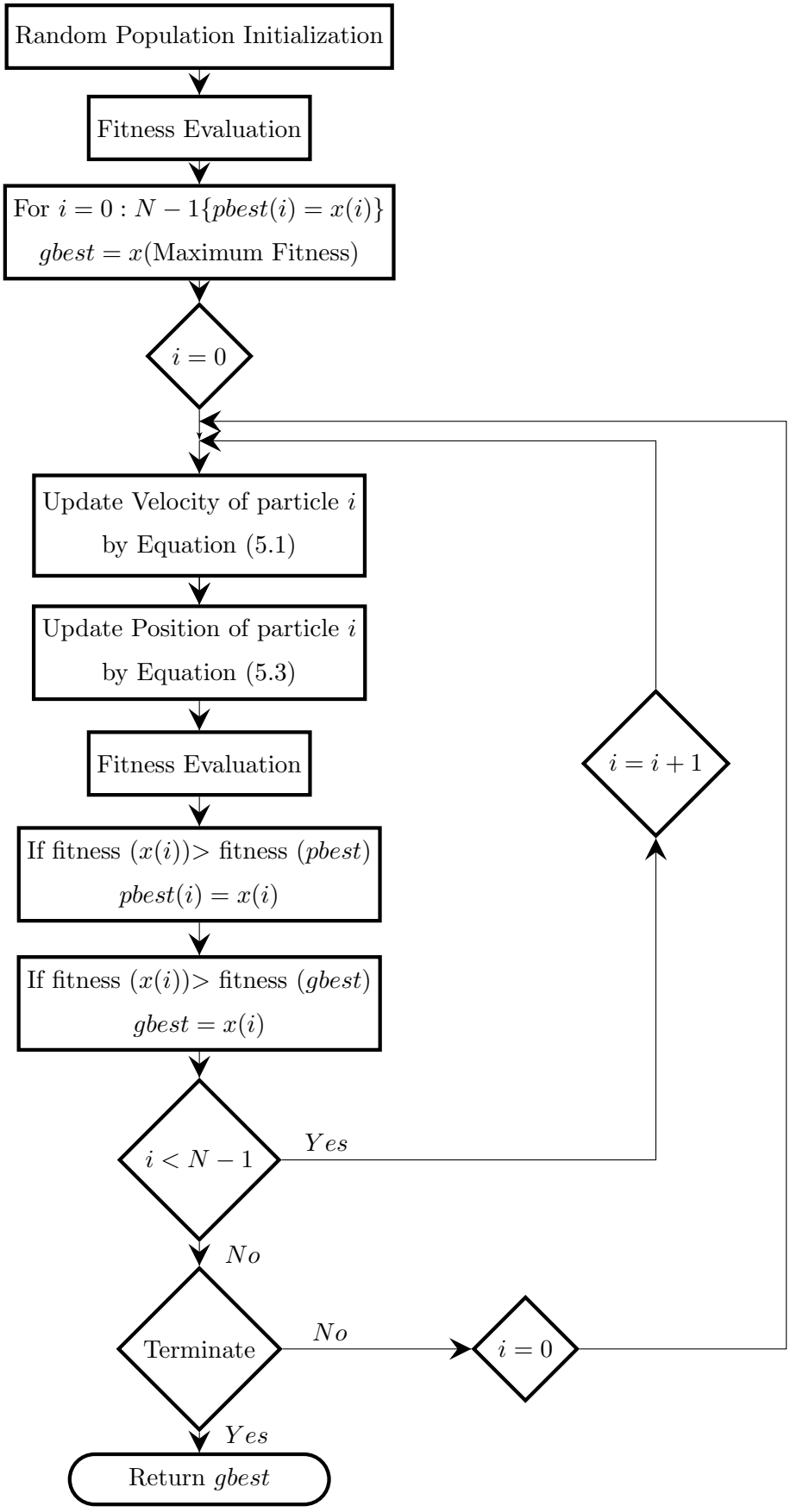


Figure 5.4: PSO Algorithm Flowchart

The algorithm can be applied to a variety of optimization problems in different domains, in order to implement the algorithm , it is required to adapt variables and parameters of the latter to the considered problem.

5.7 Accelerated PSO

There are many variations which extend the standard algorithm, and the most noticeably the algorithms use inertia function w so that V_i^k is replaced by wV_i^t where weight can be expressed as in Eq 5.2. This is equivalent to introduce a virtual mass to stabilize the motion of the particles, and thus the algorithm is expected to converge more quickly.

The standard particle swarm optimization uses both the current global best *gbest* and the individual best *pbest*. The reason of using the individual best is primarily to increase the diversity in the quality solution, however, this diversity can be simulated using the randomness. Subsequently, there is no compelling reason for using the individual best. A simplified version which could accelerate the convergence of the algorithm is to use the global best only [23]. Thus, in the accelerated particle swarm optimization, the velocity vector is generated by

$$V_i^{k+1} = V_i^k + c_1(rand - 0.5) + c_2(gbest - X_i^k), \tag{5.4}$$

where *rand* is a random variable with values from 0 to 1. The update of the position is simply

$$X_i^{k+1} = X_i^k + V_i^{k+1}. \tag{5.5}$$

In order to increase the convergence even further, we can also write the update of the location in a single step

$$X_i^{k+1} = (1 - c_2)X_i^k + c_2 gbest + c_1(rand - 0.5). \tag{5.6}$$

This simpler version will give the same order of convergence. the typical values for this accelerated PSO are $c_1 \approx 0.1 \sim 0.4$ and $c_2 \approx 0.1 \sim 0.7$, though $c_1 \approx 0.2$ and $c_2 \sim 0.5$ are recommended for most uni-modal objective objective functions.

A further accelerated PSO is to reduce the randomness as iterations proceed. This mean that we can use a monotonically decreasing function such as

$$c_1 = \alpha_0 \exp(-\gamma k), \qquad \text{or} \qquad c_1 = \alpha_0 \gamma^k, \quad (\gamma < 1) \tag{5.7}$$

where $\alpha_0 \approx 0.5 \sim 1$ is the initial value of the randomness parameter. k is the number of iterations or time steps. $\gamma < 1$ is a control parameter.

Example 5.7.1. Suppose we have to find the minimizing value of the following function [18]:

$$f = x^4 - 14x^3 + 60x^2 - 70x, \qquad \text{with } -10 < x < 10$$

Given the solution of this problem : $fminsearch$: $f_min(0.7809) = -24.3696$. The compared results are shown in the table below :

Method	Result
$fminsearch$	$f_{min}(0.7809) = -24.3696$
PSO	$f_{min}(0.7799) = -24.3696$
Accelerated PSO	$f_{min}(0.7788) = -24.3695$

Table 5.1: We can observe that results are almost identical with a slight difference to global best

We can further discuss the difference in the shown figure below :

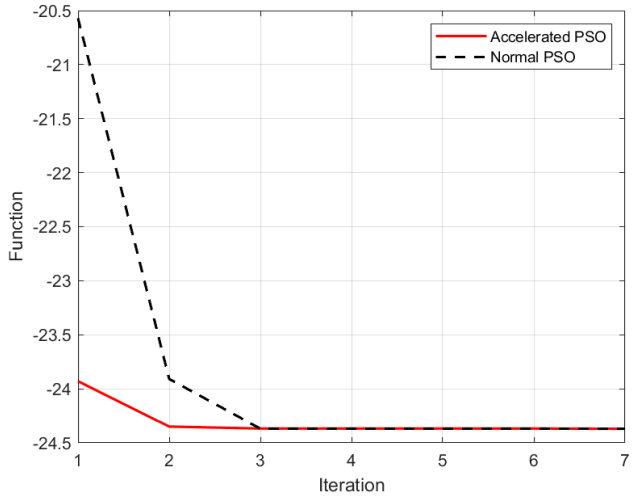


Figure 5.5: PSO and Accelerated PSO applied to f

Analyzing fig 5.5, we see that Accelerated PSO started way closer to the solution unlike the normal one, we also notice that it took only one iteration for the Accelerated PSO to reach global minimum, where the normal PSO took an additional iteration to reach it's minimum.

Remark. : the implementation of the algorithm in MATLAB are included in the appendices.

PSO algorithms can outperform genetic algorithms and other conventional algorithms for solving many optimization problems. This is partially due to that fact that the broadcasting ability of current best estimates gives a better and quicker convergence towards the optimality. However, this algorithm is almost memory-less since it does not record the movement paths of each particle, and it is expected that it can be further improved using short-term memory [23].

5.8 Summary

In this chapter we investigated optimal control and its relation to optimization, we saw Meta-heuristic optimization methods and their source of inspiration, dove into Particle swarm optimization method and developed it's algorithm. Furthermore, Accelerated PSO was discussed and it will be used in the next chapter.

Chapter 6

Implementation

6.1 Introduction

This chapter collects all what we have discussed in the previous ones from modeling the plant through linear and non-linear control up to optimization and put them to work on a chosen satellite which can fit our developed model characterizations to a good extent.

Implementation of the simulation, controllers , optimization algorithm are all done in MATLAB environment and presented in the appendix section.

6.2 Feedback Linearization Controller

Our first controller will be nonlinear, as we have discussed in the forth chapter , the idea is to linearize our system every time step with an input u throughout the control law :

$$v = \ddot{y}_d - k_1 e - k_2 \dot{e}$$

where k_1 k_2 are 3×1 real vectors, their value are manually obtained (easily).
performance tests contains analysis of the transient and steady state response and real time tracking.

6.2.1 Transient and Steady State response

After setting the reference radius, azimuth, and elevation all to a step we get:

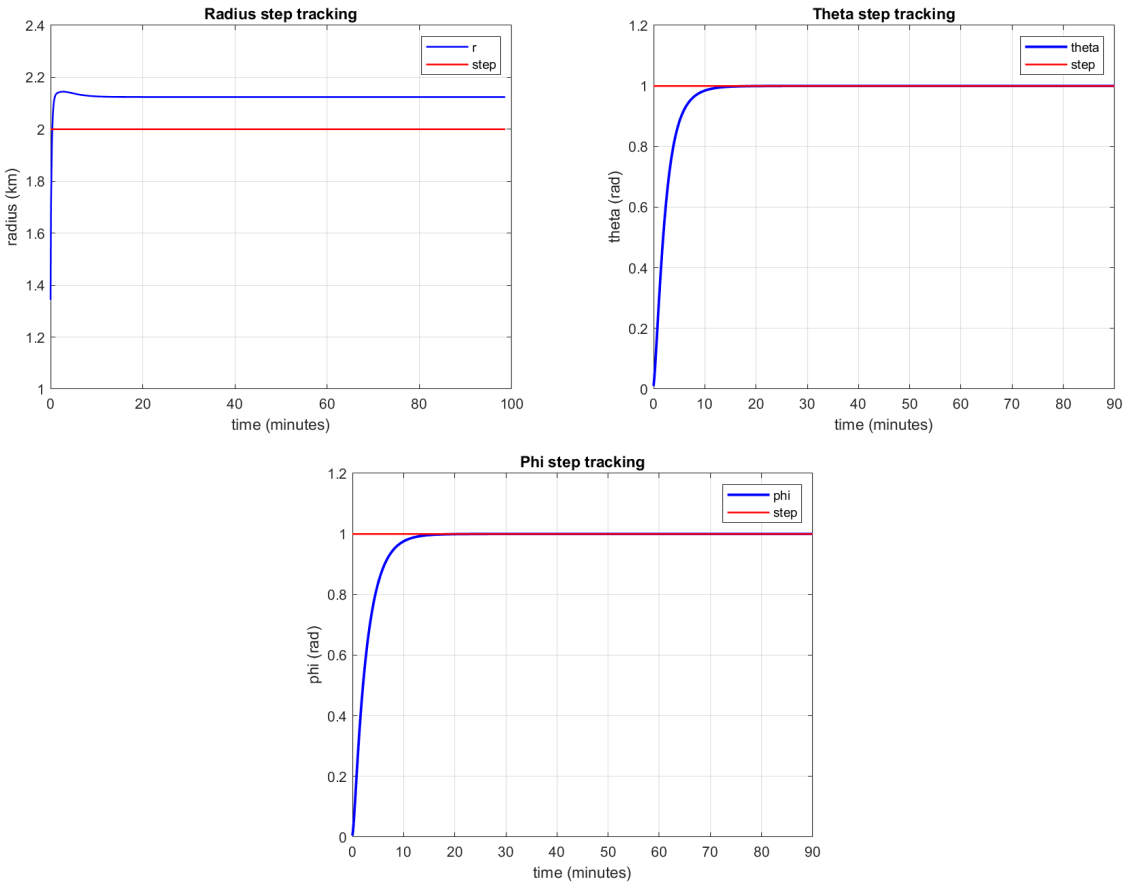
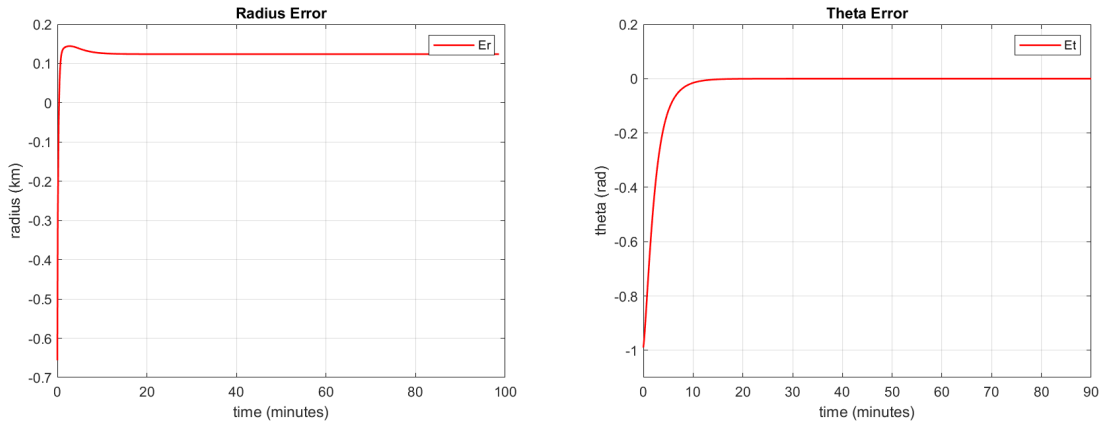


Figure 6.1: Step response of feedback linearization control system

If we look at the first output we can notice a good 0.5 minute rise time but has a 7.2% overshoot and 0.124 steady-state error. For the rest outputs (azimuth and elevation) the step response performance is near perfect, $5.1mn$ and $5.8mn$ rise time , no overshoot and a steady state error equals to 0.



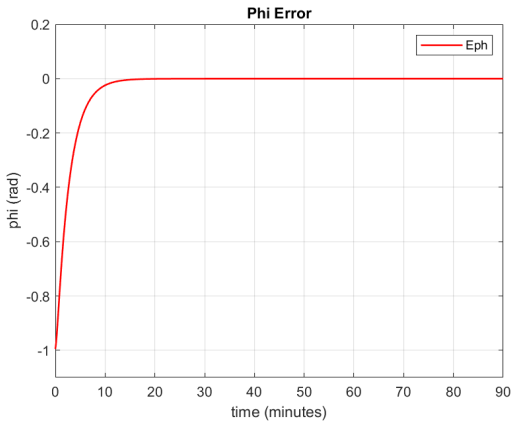


Figure 6.2: Error signals

The figure 6.2 above represent output error of the step response. First output error reaches rapidly 0 but it exceeds further up till 0.1445, and settles at 0.124. the rest outputs move up perfectly to 0.

We can summarize the performance of the controller in this table below

Variable	Output 1 (radius)	Output 2 (azimuth)	Output 3 (elevation)
Rise time	5.1	5.1	5.8
Overshoot	7.2%	0%	0%
Settling time	13	15.2	16.9
Delay-time	0.2	2	2.1
Peak time	2.9	15.2	16.9
Steady State error	0.124	0	0

Table 6.1: Controller performance characteristics

The gains k_1 and k_2 chosen are:

$$k_1 = [40 \quad 1 \quad 1] \qquad , \quad k_2 = [15 \quad 3 \quad 3]$$

6.2.2 Tracking Simulation

After setting the reference attitude for tracking purposes for a day duration of earth orbiting, Outputs are shown in the figure 6.3.

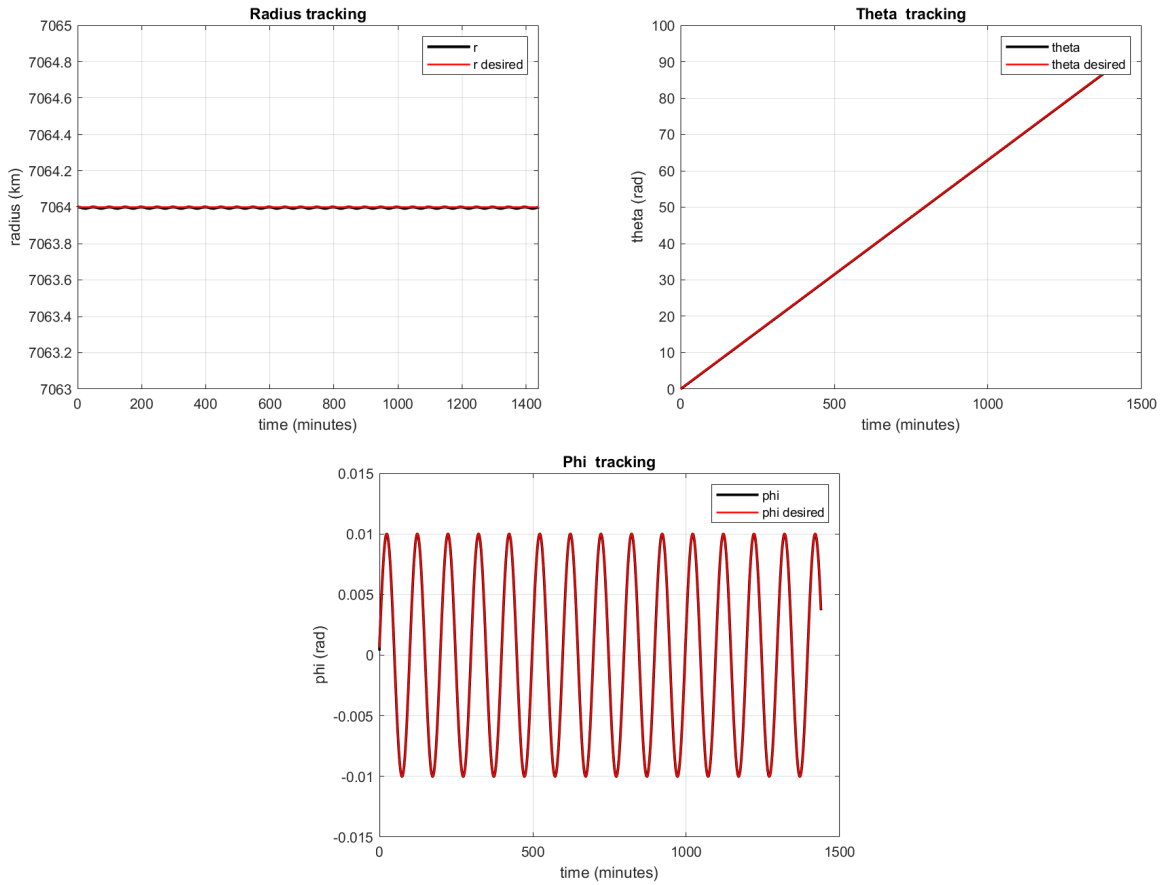
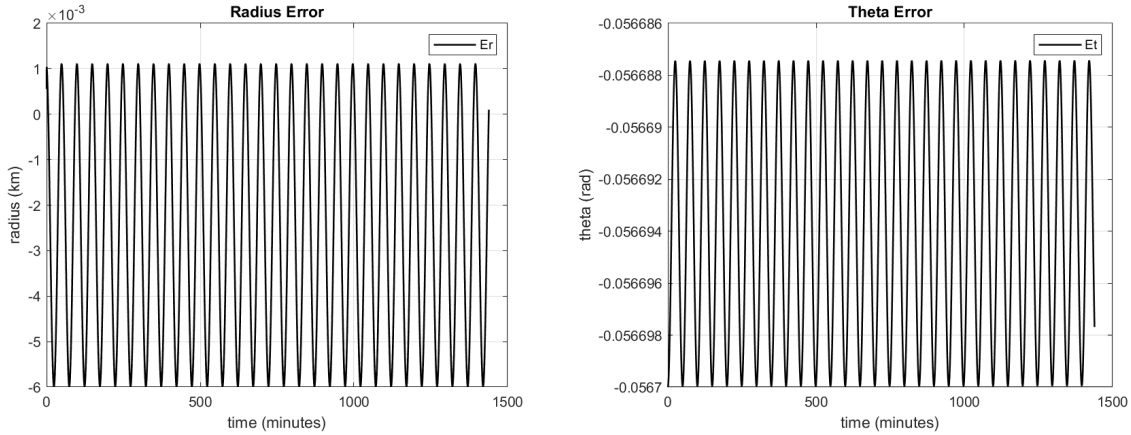


Figure 6.3: Tracking trajectory with feedback linearization control.

the figure 6.3 shows a good tracking to the desired radius, oscillation are present but very minimal and acceptable ($0.005km$ error). For the azimuth tracking the figure also shows great tracking to the reference with very minimal error. As for the elevation tracking, 6.3 reveals good tracking in both transient and steady state phase, with no spikes.



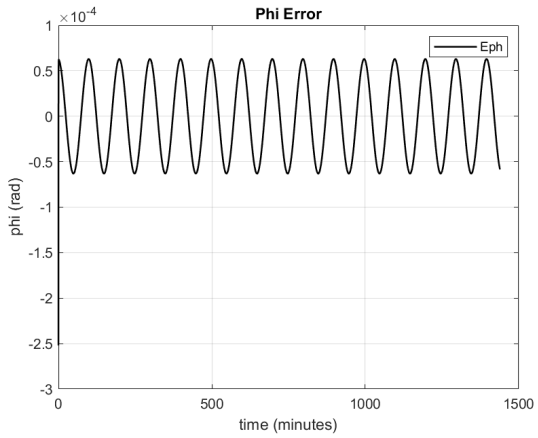


Figure 6.4: Tracking Error

the figure 6.4 shows oscillations in all outputs but very minimal, error varies form -0.006 to 0.001 in the first output, from-0.0567 to -0.05669 in the second output. for the last output the error starts from -2.5×10^4 goes rapidly to -6.2×10^5 then starts oscillating till the end.

The controller exhibits a promising results when it comes to performance.

6.3 MIMO PID Controller

As what we have investigated in the third chapter, our second controller consists of 4 parameters unlike the classic PID with 3 parameters. The mathematical representation comes in the form :

$$PID = (KS)^{-1}(K_I + K_P S + K_D S^2) \tag{6.1}$$

where K, K_I, K_P, K_D are 3×3real matrices manually obtained through some Solvent randomness. performance benchmark contains transient and steady state response and real time tracking.

6.3.1 Transient and Steady State response

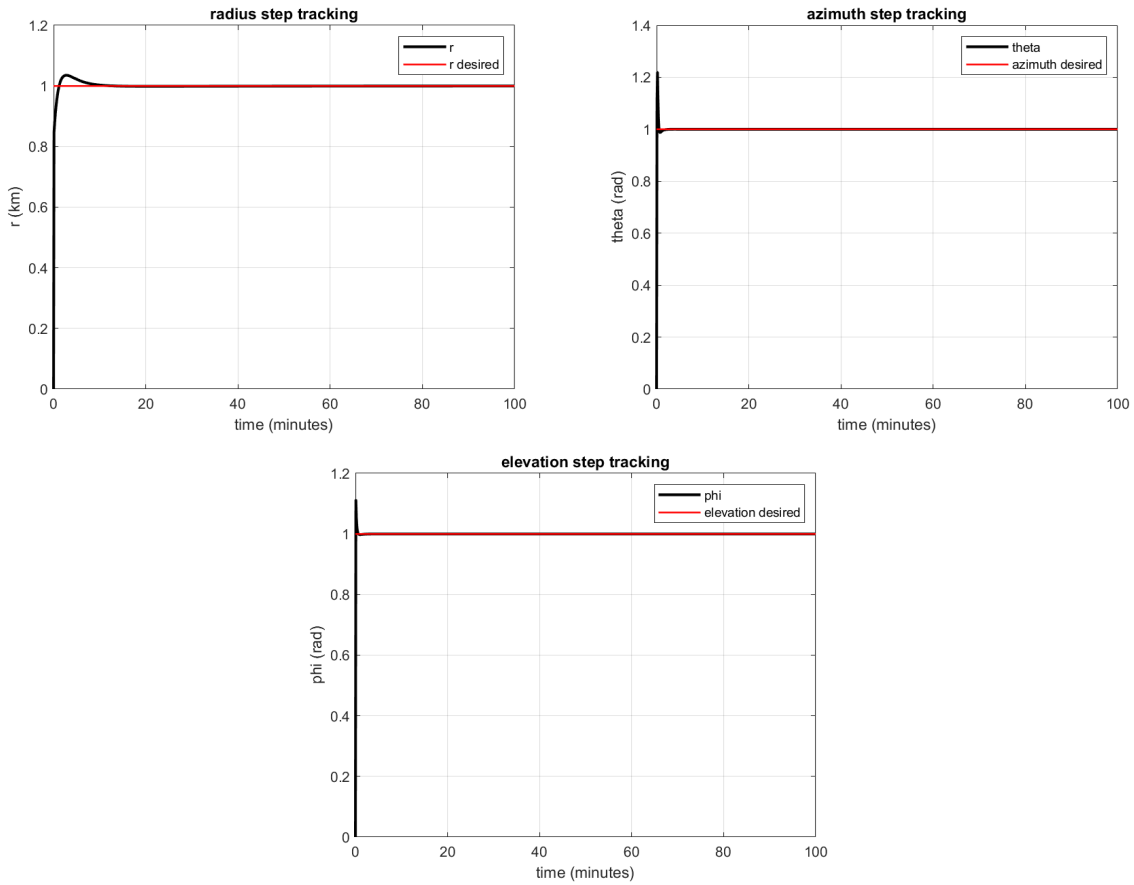
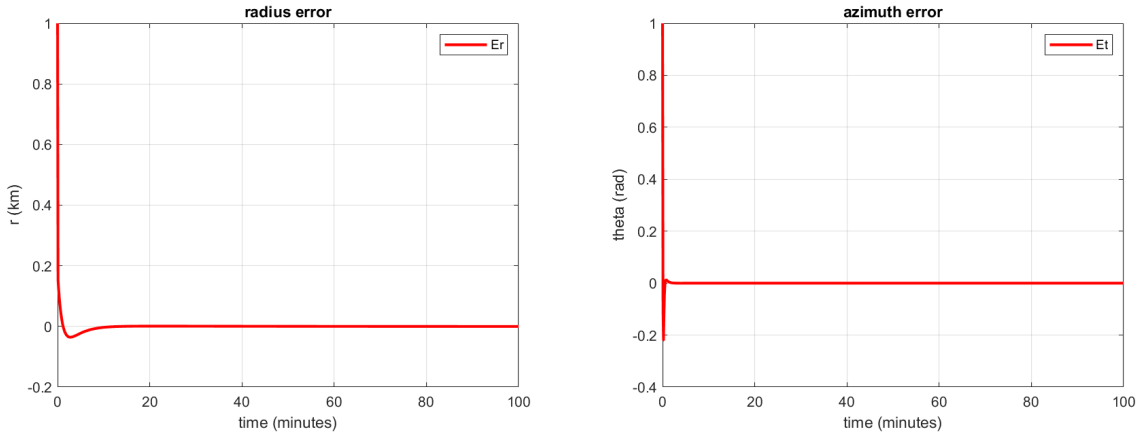


Figure 6.5: Step response with PID Controller

The figure 6.5 displays a $2mn$ rise time, 5% overshoot and no steady state error regarding the first output. For the second output, the rise time $20s$ with 20% overshoot and no steady state error. The rise time in the last output is around $30s$, its overshoot is over 10% and no steady state error.



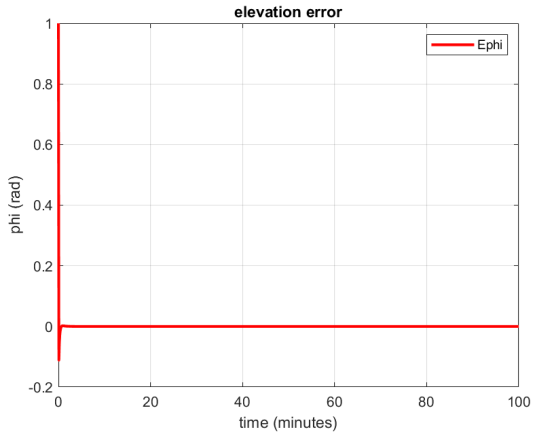


Figure 6.6: Error signals with PID Controller

As we can see in figure 6.6, the error rapdidly moves to 0 in all outputs exceed to -0.05,-0.2 and -0.1 respectively and settles down quickly to 0.

All performance aspects are presented in the table below:

Variable	Output 1 (radius)	Output 2 (azimuth)	Output 3 (elevation)
Rise time	2	$22s$	$35s$
Overshoot	5%	10%	20%
Settling time	14	4.1	3.5
Delay-time	50s	5s	7s
Peak time	4.8	1.1	0.9
Steady State error	0	0	0

Table 6.2: MIMO PID Controller performance characteristics

6.3.2 Tracking Simulation

After setting the attitude reference we run the controller and the results are shown in the figure below :

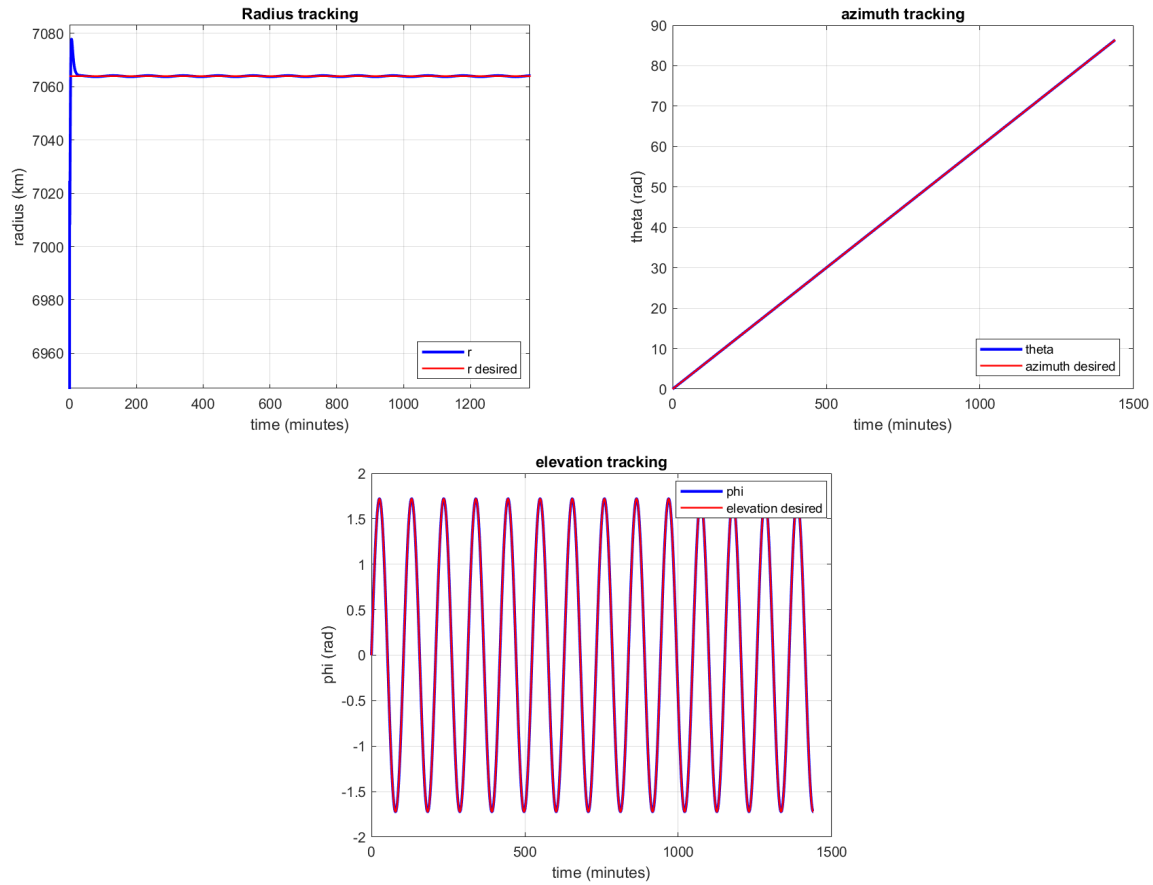


Figure 6.7: Tracking trajectory with MIMO PID Controller

The figure 6.7 shows a good tracking to the reference in steady state phase, but with a slight overshoot and minimal oscillation. The second and the third output shows also good tracking to the reference with no spikes.

6.4 MIMO PID Controller with PSO

Our final controller will be an Enhanced version of the previous one which is done by obtaining the PID gains with the particle swarm optimization technique.
As we Chose the IAE index (IAE= $\int_0^\infty |e(t)|dt$),with $e(t)$ is the range error. PSO algorithm is set to find all elements of the solvents for we can construct a PID controller using the Diophantine equation.

The gains in the figure 6.8 are obtained by the Accelerated PSO as it's discussed in chapter 5.

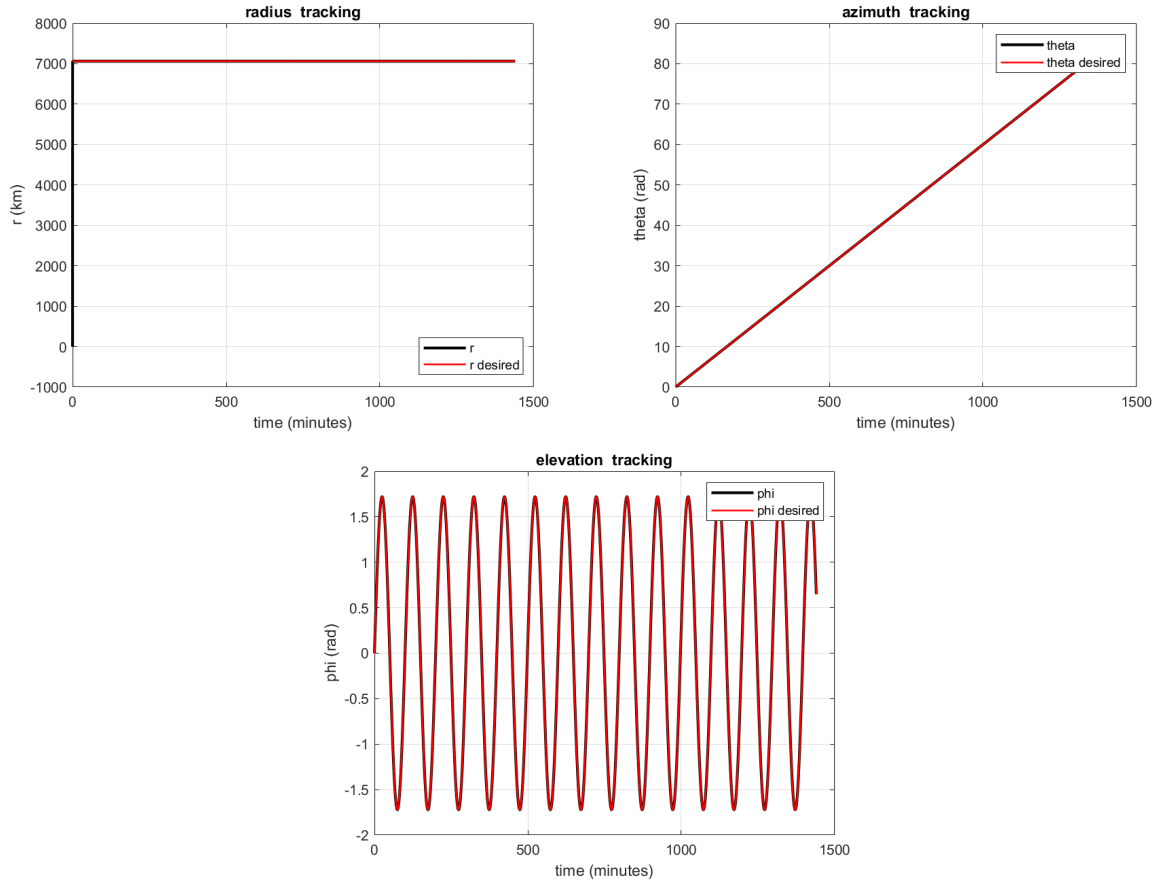
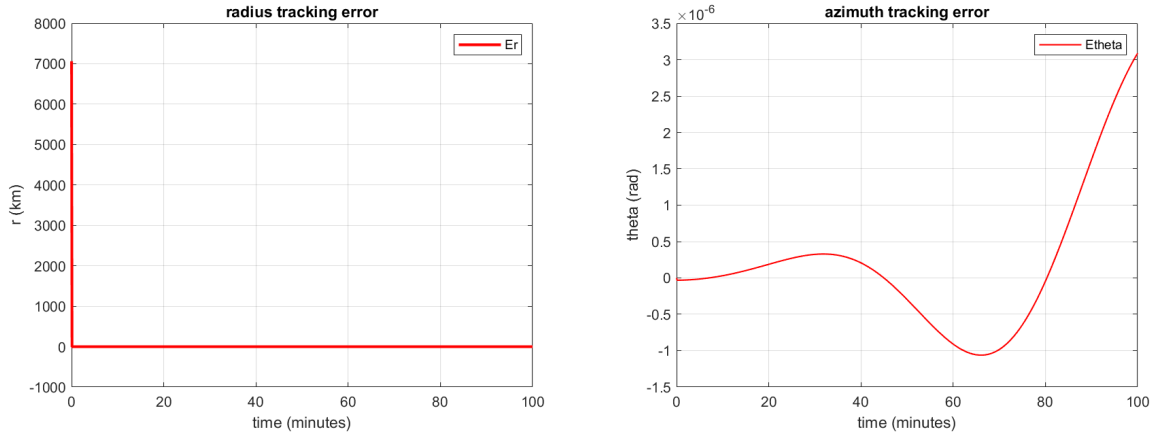


Figure 6.8: Tracking trajectory with enhanced PID controller

The simulation exhibits a great tracking in the steady state phase for the first output with no overshoot or oscillation. Second and third are in near perfect tracking with very minimal error.



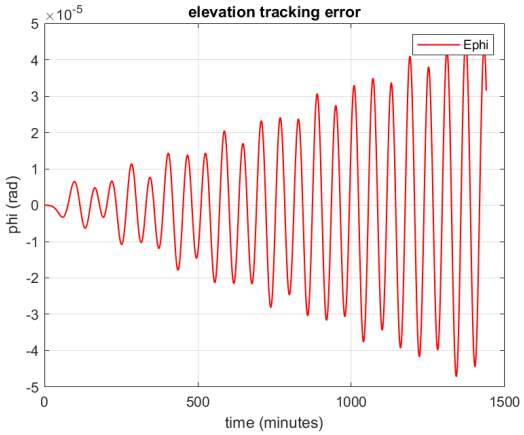


Figure 6.9: Error signals with Enhanced PID controller

The radius error in steady state phase is stable around 0 with no oscillation. azimuth error is considered near absent for which its peak is 3×10^{-6}.for the elevation error signal oscillations are present but very minimal as its higher value reaches 3×10^{-6}.

The 3D simulation of satellite's orbit tracking is as shown in the figure below:

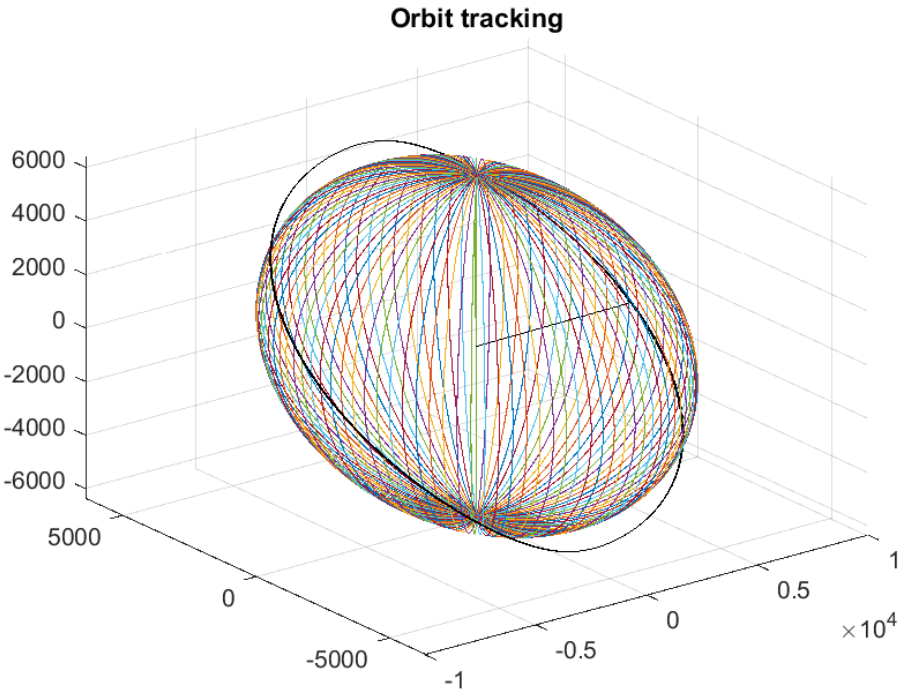


Figure 6.10: Satellite's Orbit tracking

With K, K_I, K_P, K_D values are :$(e + x = 10^x)$

$$K = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad , K_D = \begin{bmatrix} -6.39e+11 & 7e+17 & 8.28e+17 \\ -1.79e+12 & 9.42e+16 & -3.39e+17 \\ 7.55e+11 & 9.69e+16 & 3.28e+17 \end{bmatrix}$$

$$K_I = \begin{bmatrix} 4.6e+06 & 1.42e+12 & 1.22e+12 \\ -7.91e+05 & 2.76e+12 & 1.84e+11 \\ 1.85e+06 & -4.51e+11 & 2.45e+12 \end{bmatrix}$$

$$K_P = \begin{bmatrix} 3.08e+09 & 2.38e+15 & 2.35e+15 \\ -2.59e+09 & 2e+15 & -6.21e+13 \\ 2.58e+09 & -3.54e+14 & 1.48e+15 \end{bmatrix}$$

Remarks:

- We notice the appearance of a very high gain values , and that's mainly due to the control matrix's influence (Matrix B) as shown in the second chapter.
- Regarding the first output of the tracking simulation, the radial component shouldn't start from 0 because in real time the satellite is already at 693 km above earth, and that all because our system is causal (matrix D of the system is a null matrix). Consequently we ignored the analysis of radial transient response.

6.5 Comparison Between PID and PID with PSO controllers

⊛ PID Controllers main difficulty is to find a proper gains, PSO take care of that problem.

⊛ Regular PID Controller's gains comes in a random form (we use a specific random solvents) so the chance of getting good ones are fairly low,so its not reliable.

⊛ In terms of performance the Enhanced version is way better as seen figure 6.7 and figure 6.8.

⊛ PSO's algorithm is not difficult to link it with the regular PID as it is also easy to set up.

⊛ As we Introduced an accelerated PSO, obtaining results is not time consuming.

6.6 Summary

In this chapter we have implemented the controller theories that we have studied through MATLAB . We first displayed the feedback linearization tests of performance, where we saw a good tracking signals with average and acceptable oscillations in radial output. then we showed MIMO PID controller where its performance tests carried out a good tracking but with significant error. finally we introduced the enhanced MIMO PID controller where we Used PSO algorithm to obtain the PID gains and its results were way better and precise with very minimal error, we followed up a comparison between the two.

General Conclusion

In this thesis we have presented different control and optimization approaches in the design of an attitude controller for a particular class of satellites. The main objective was to obtain a dynamical model of our satellite, and then present the most recent, advanced and illustrative methods used in attitude control and optimization of satellites in a simpler and didactic manner for a quick,easy and efficient understanding for readers enabling the controllers conception.

We have considered two approach theories for attitude control, classical and nonlinear control. The classical one contains well known method of control for linear systems such as MIMO PID control. The nonlinear control approach concerns the Feedback Linearization method. Optimal control plays the alpha role in finding the best possible results by finding MIMO PID proper gains. Lastly the implementation shows all three tested controllers: feedback linearization,MIMO PID and MIMO PID using accelerated PSO. The tests revealed that all three satisfies our requirements from step response and tracking trajectory perspective to analyzing error signals in all three cases,but in conclusion only the MIMO PID with implemented PSO guaranteed a perfect tracking and enhanced performance with extremely minimal error.

The work conducted in this thesis can be extended in different directions. For instance, one may reconsider the satellite's model and use different approach. One also may use different control strategies for the model especially the nonlinear ones such as sliding mode. Moreover, in optimal control one can be exposed to more Advanced and developed PSO algorithms or exploring other meta-heuristic optimization methods mentioning Genetic Algorithm, Bacterial Foraging Optimization or Artificial Bee Colony.

Appendices

Appendix A

Satellite Appendix

A.1 Introduction

In this appendix we are going to present the satellite that has been used in simulations providing pictures, general information and table of characteristics.

A.2 Sentinel-1 A

Figure A.1: Sentinel 1-A Spacecraft

A.2.1 Propulsion System

Sentinel-1 is equipped with a monopropellant propulsion system consisting of 14 thrusters serving as Reaction Control system. Six of the 14 thrusters are facing the same direction to be able to perform delta-v burns for orbit control. The Propulsion System uses a large propellant tank that is installed in the center of the satellite to maintain the vehicle's Center of Gravity once the propellant is expended. The tank can hold up to 154 Kilograms of Hydrazine and is pressurized with gaseous Helium.

A.2.2 Spacecraft Characteristics

Characteristic	Description
Design life	7 Years (consumables for 12 years)
Orbit	. Near-polar Sun-Synchronus . 693 km altitude . 98.18 degrees of inclination . 12 day repeat cycle . 175 orbits per cycle
Orbital Period	98.6 minutes
Attitude stabilisation	3 axis stabilised
Launch Weight	2300 kg
Actual Weight	2187 kg
Dimenstions	$3900 \times 2600 \times 2500 mm^3$
Launch Date	3 april 2014

Table A.1: Sentinel 1-A characteristics

Appendix B

MATLAB Codes

B.1 Introduction

This appendix presents Matlab codes used for our controllers and for PSO algorithm.

B.2 Open Loop System

```
clear all
format long
k =6.6*10-11; % gravitational constant
m = 2187; %kg
w0 =0.063; %(radian/minute)
r0 = 693+6371; % radius in Km
step=0.1;states=[];
r = [];theta = [];phi = [];
tr = [];t= 1:0.1:1440;
n=length(t);inputs=[];
for i=1:n
y(1)=yo(1);y(2)=yo(2);y(3)=yo(3);
y(4)=yo(4);y(5)=yo(5);y(6)=yo(6);
    %% RUNGE.KUTTA
H = step;
HH = H/2;
```

```
H6 = H/6;
% first step
V = yo ;
dydx = equationss(V) ;
yt = yo + HH*dydx' ;
%  second step
V = yt ;
dyt = equationss(V) ;
yt = yo + HH*dyt';
% third step
V = yt ;
dym = equationss(V) ;
yt = yo + H*dym' ;
dym = dyt + dym ;
% fourth step
V = yt ;
dyt = equationss(V) ;
yout = yo + H6*(dydx' + dyt' + 2*dym');
d = yout';
yo=d';
% inputs=[u(1) u(2) u(3);inputs];
states=[states;d];
r = [r;d(1)];
theta = [theta;d(3)];
phi = [phi;d(5)];
% tr=[tr;d(1)+6371];
end
time=0:0.1:1439;
figure
plot(time,r,'k','linewidth',1.95)
figure
```

```
plot(time,theta,'k','linewidth',1.95)
figure
plot(time,phi,'k','linewidth',1.95)
```

B.2.1 Function

```
function yout = equationss(V)
% In this function we introduce the differential equations
% The output will the evaluation of these ODE's
x=V;
m = 2187;
k = 32;
u=[0,0,0];
%% differrential equations
dx(1)=x(2);
dx(2)=x(1)*(x(4)^2)*(cos(x(5)^2)
-(k./(x(1)^2)))+ u(1)/m;
dx(3)=x(4);
dx(4)=(-2*x(2)*x(4)/x(1))+2*x(4)*x(6)*tan(x(5))
+u(2)/(m*(x(1)*cos(x(5)))^2);
dx(5)=x(6);
dx(6)=-(x(4)^2)*cos(x(5))*sin(x(5))
-2*(x(2)*x(6)/x(1))+u(3)/(m*(x(1)^2));
yout = dx;
end
```

B.3 Feedback Linearization

```
clear all
format long
k =6.6*10-11; % gravitational constant
```

```
m = 2187; %kg
w0 =0.063; %(radian/minute)
r0 = 693+6371; % radius in Km
yo=[r0 0 0 w0 0 0]';step=0.1;
states=[];r = [];
theta = [];phi = [];tr = [];
t= 1:0.1:1440;n=length(t);
for i=1:n
y(1)=yo(1);y(2)=yo(2);y(3)=yo(3);
y(4)=yo(4);y(5)=yo(5);y(6)=yo(6);
%% Desired Attitude
yd=[(693+6371)*ones(n,1)
,0.063*t'.*ones(n,1) 0.01*sin(0.063*t')];
yp=[zeros(n,1) 0.063*ones(n,1)
, 0.01*0.063*cos(0.063*t')];
ypp=[zeros(n,1) zeros(n,1)
, -0.01*0.063^2*sin(0.063*t')];
%% V vector
a=ypp(i,1)-40*(y(1)-yd(i,1))-15*(y(2)-yp(i,1));
b=ypp(i,2)-1*(y(3)-yd(i,2))-3*(y(4)-yp(i,2));
c=ypp(i,3)-1*(y(5)-yd(i,3))-3*(y(6)-yp(i,3));
v = [a b c];
    %% RUNGE.KUTTA
H = step;
HH = H/2;
H6 = H/6;
% first step
V = yo ;
dydx = equations(V,v) ;
yt = yo + HH*dydx' ;
%  second step
```

```
V = yt  ;
dyt = equations(V,v)  ;
yt = yo + HH*dyt';
% third step
V = yt  ;
dym = equations(V,v)  ;
yt = yo + H*dym'  ;
dym = dyt + dym  ;
% fourth step
V = yt  ;
dyt = equations(V,v)  ;
yout = yo + H6*(dydx' + dyt' + 2*dym');
d = yout';
yo=d';
states=[states;d];
r = [r;d(1)];
theta = [theta;d(3)];
phi = [phi;d(5)];
end
```

B.3.1 Function

```
function yout = equations(V,ff)
% In this function we introduce
%the differential equations
% The output will the evaluation of these ODE's
x=V;m = 2187;k = 32;v = ff;
%% system  inputs
u(1)=m*(v(1)-(x(1)*x(4)^2*(cos(x(4))^2)
+x(1)*x(5)^2-k/x(1)^2));
u(2)=m*x(1)^2*(cos(x(5))^2)*(v(2)
```

```
−(−2∗x(2)∗x(4)/x(1))
+2∗x(4)∗x(6)∗tan(x(5)));
u(3)=m∗x(1)^2∗(v(3)
−(−(x(4)^2)∗cos(x(5))∗sin(x(5))
−2∗(x(2)∗x(6)/x(1))));
%% differrential equations
dx(1)=x(2);
dx(2)=x(1)∗(x(4)^2)∗(cos(x(5)^2)
−(k./(x(1)^2)))+ u(1)/m;
dx(3)=x(4);
dx(4)=(−2∗x(2)∗x(4)/x(1))+2∗x(4)∗x(6)∗tan(x(5))
+u(2)/(m∗(x(1)∗cos(x(5)))^2);
dx(5)=x(6);
dx(6)=−(x(4)^2)∗cos(x(5))∗sin(x(5))
−2∗(x(2)∗x(6)/x(1))+u(3)/(m∗(x(1)^2));
yout = dx;
end
```

6.4 MIMO PID with PSO Algorithm

```
clear all;clc
wmax=0.9;wmin=0.8;c1=2;c2=2;
n=20;itermax=6;
m = 27;v = zeros(m,n);
xmax= −10∗ones(1,m);
xmin = −1000∗ones(1,m);
rand('state',0);
%% Random Population Initialization
for i=1:n
for j=1:m
x(j,i)=xmin(j)+rand∗(xmax(j)−xmin(j));
```

```
end
fun_marge(i)=pso_control(x(1,i),x(2,i),x(3,i),x(4,i)
,x(5,i),x(6,i),x(7,i),x(8,i),x(9,i),x(10,i),x(11,i)
,x(12,i),x(13,i),x(14,i),x(15,i),x(16,i),x(17,i)
,x(18,i),x(19,i),x(20,i),x(21,i),x(22,i),x(23,i)
*,x(24,i),x(25,i),x(26,i),x(27,i));
end
%% Fitness Evaluation
xbest = x;
fbest = fun_marge;
fgbest = min(fun_marge);
gbest = x(:,find(fun_marge==fgbest));

for iter=1:itermax
w = wmax-(wmax-wmin)*iter/itermax;
%  c1 = c1_0*exp(-gamma*iter);
for i=1:n
%% Update Velocity of particle i
% v(:,i) = w*v(:,i)+c1*rand*(xbest(:,i)
%-x(:,i))+c2*rand*(gbest-x(:,i));
% uncomment for regular PSO
v(:,i) = v(:,i)+c1*(rand-0.5)+c2*(gbest-x(:,i));
%% Update Position of particle i
% x(:,i) = x(:,i)+v(:,i);%uncomment for regular PSO
x(:,i) = (1-c2)*x(:,i) +c2*gbest+c1*(rand-0.5);
for jj = 1:m
if x(jj,i)>xmax(jj)
x(jj,i)=xmax(jj);
end
if x(jj,i)<xmin(jj)
x(jj,i)=xmin(jj);
```

```
end
end
%% Fitness Evaluation
fun_marge(i)=pso_control(x(1,i),x(2,i),x(3,i),x(4,i)
,x(5,i),x(6,i),x(7,i),x(8,i),x(9,i),x(10,i),x(11,i)
,x(12,i),x(13,i),x(14,i),x(15,i),x(16,i),x(17,i)
,x(18,i),x(19,i),x(20,i),x(21,i),x(22,i),x(23,i)
,x(24,i),x(25,i),x(26,i),x(27,i));
if fun_marge(i) < fbest(i)
xbest(:,i)=x(:,i);
fbest(i)=fun_marge(i);
end
if fun_marge(i) < fgbest
gbest=x(:,i);
fgbest=fun_marge(i);
end
end
result(iter)=fgbest;
end
plot([1:itermax], result,'-r','linewidth',1.5)
xlabel('Iteration');ylabel('Function');grid on
```

6.4.1 MIMO PID Function

```
function ee=pso_control(k1,k2,k3,k4,k5,k6,k7,k8,k9,k10,k11,k12,k13,k14,k15,
format long
load('lasttry','linsys1')

A=linsys1.A;B=linsys1.B;
C=linsys1.C;D=linsys1.D;

```

```
I=eye(3,3);Z=zeros(3,3);
omega = [B A*B];
% Ds = -inv(omega)*(A^2)*B;
tet=-A^2*B*I;
Ds=linsolve(omega,tet);
D0=Ds(1:3,:);D1=Ds(4:6,:);
Mc =[D1 I;I Z];
Ns=C*omega*Mc;
N0=Ns(:,1:3);N1=Ns(:,4:6);

s=tf('s');
H1=C*inv(s*eye(6,6)-A)*B+D;
H2=(N1*s+N0)*inv(I*s^2+D1*s+D0);

t=0:0.1:1440;
n=length(t);
u1=(693+6371)*ones(1,n);
u2=0.06*t;
u3=((pi/2)+0.15)*sin(0.06*t);
u=[u1' u2' u3'];
R1 = [k1 k2 k3;k4 k5 k6;k7 k8 k9];
R2 = [k10 k11 k12;k13 k14 k15;k16 k17 k18];
R3 = [k19 k20 k21;k22 k23 k24;k25 k26 k27];
Vr=[I I I;R1 R2 R3;R1^2 R2^2 R3^2];
Df=-[R1^3 R2^3 R3^3]*inv(Vr);
M = [I D1 D0 Z;N1 N0 Z Z;Z N1 N0 Z;Z Z N1 N0];
PID = [I Df]*inv(M);
K=PID(:,1:3);
Kd=PID(:,4:6);
Kp=PID(:,7:9);
Ki=PID(:,10:12);
```

```
39
40 PID_S=(inv(K)/s)*(Ki+Kp*s+Kd*s^2);
41 T_f=H1*PID_S;
42 H_cl=feedback(T_f,I,-1);
43 H_cl=minreal(H_cl);
44
45 y=lsim(H_cl,u,t);
46 y1=y(1:n,1);
47 y2=y(1:n,2);
48 y3=y(1:n,3);
49 [xu,yu,zu] = sph2cart(u1,u2,u3);
50 [xs,ys,zs] = sph2cart(y1,y2,y3);
51 eee=[];
52 for i=1:n
53 ex= xu(i)-xs(i);
54 ey= yu(i)-ys(i);
55 ez= zu(i)-zs(i);
56 e=[ex ey ez];
57 erro=norm(e);
58 eee=[eee;erro];
59 end
60 ee=trapz(t,eee)
```

Bibliography

[1] R.R.Bate, D.D.Mueller, J.E.White *Fundamentals of Astrodynamics* , Dover, (1971).

[2] Damaren, C. De Ruiter, A.H.J.Forbes, James.R *Spacecraft Dynamics and Control: an Introduction* , Wiley, (2013).

[3] L. Mazzini *Flexible Spacecraft dynamics, Control and Guidance* , Springer, (2016).

[4] A. Tewari, *Advanced Control of Aircraft,Spacecraft and Rockets* , Aerospace Series, (2011).

[5] T. Kailath, *Linear Systems* , Prentice-hall, (1980).

[6] B. Datta, *Numerical Methods for Linear Control Systems* , Academic Press, (2003).

[7] A. E. Bryson, *Applied Optimal Control* , New York, (1965).

[8] B. Bekhiti, *Dynamic Modeling and Control of Large Space Structures* , Blida University (2019).

[9] M. Vidyasagar, *Nonlinear Systems Analysis* , (1992).

[10] K. J. Astrom and T. Hagglund, *Advanced PID Control*, Lund University, (2005).

[11] B. Bekhiti, *Multivariable control system design using The Theory of Matrix Polynomials* , M'Hamed Bougara University , Boumerdes, (2018).

[12] M. Akroum, *Multi input multi output system identification using matrix fraction description* , M'Hamed Bougara University, Boumerdes, (2018).

[13] B. Bekhiti, A. Dehimene, B. Nail, K. Hariche, A. Hamadouche, *On Block Roots of Matrix Polynomials Based MIMO Control System Design* , 4^{th} International Conference on Electrical Engineering , Boumerdes, (2015).

[14] I. Jamshed, M. Ullah, S.G. Khan, B. Khelifa, and S.Ćuković, *Nonlinear control systems – A brief overview of historical and recent advances* , De Gruyter , (2017).

[15] J .Slotine, W.li, *Applied Nonlinear Control* , Massachusetts Institute of Technology , (1991).

[16] The Amazon book.

[17] D. S. Naidu, *Optimal Control Systems* , Idaho State University , USA, (2002).

[18] S.Slami, *Introduction à l'Optimisation Méta-heuristique* , Dar Kifaya, Algeria (2019).

[19] J. Sun, C. Lai, X. Wu, *Particle Swarm Optimization : Classical and Quantum perspectives* , CRC , (2011).

[20] Y. LI, X. Zhang, Y. Chen, H. Zhou, *Particle Swarm Optimization for time-optimal control design* , Tsinghma University, beijing , (2012).

[21] K.E. Parsopoulos, M.N. Vrahatis, *Particle Swarm Optimization and intelligence : Advances and applications* , Tsinghma University, Greece , (2010).

[22] A. Lazinga (editor), *Particle Swarm Optimization* , year not montioned.

[23] X. Yang, *Introduction to Mathematical Optimization* , Cambridge University, (2008).