ISSN 0280-5316
ISRN LUTFD2/TFRT--5689--SE

Estimation of Inertial Parameters

Javier A. Martinez Serradell

Department of Automatic Control
Lund Institute of Technology
June 2002

Estimation of inertial parameters

Javier A. Martinez Serradell

June 26, 2002

Contents

1. Introduction

1.1 Estimation of inertial parameters . . . .

1.2 Identification procedure . . . .

2. Different algorithm types applied to estimation. . . . .

2.1 Least squares . . . .

2.1.1 Statistical interpretation . . . .

2.1.2 Statistical properties of least-squares estimation . . . .

2.2. Linear estimation algorithm for affine parametric non-linear system proposed by Middleton and Goodwin . . . .

2.3 Algorithm:Unnormalized least squares estimation . . . .

3. Method's to obtain the models . . . .

3.1 Newton-Euler iterative method . . . .

3.2 Using the Lagrange method . . . .

3.2.1 How to obtain the matrices A,C,Q . . . .

3.2.2 Computation of the potential energy . . . .

3.2.3 Dynamic model properties. . . . .

3.2.4 Example . . . .

4. Programs in Maple. . . . .

4.1 Program using Dynamic model . . . .

4.2 Program using Newton-Euler Iterative method . . . .

5. Estimation's examples . . . .

5.1 Pendulum model . . . .

5.1.1 Pendulum model adding filters. . . . .

5.1.2 Pendulum model adding noises . . . .

5.2 Two-link manipulator . . . .

5.2.1 Two-link manipulator scheme. . . . .

5.2.2 Two-link manipulator adding filters . . . .

5.2.3 Two-link manipulator adding noises. . . . .

5.3 Inertia wheel model . . . .

5.3.1 Inertia wheel model scheme. . . . .

5.3.2 Inertia wheel with noises . . . .

5.4 Inertia wheel plant . . . .

Appendix . . . .

A. Commands used in the programs . . . .

A.1 Introduction . . . .

A.2 Commands used in Newton-Euler and Lagrange equation method . . . .

B. Bibliography

1. Introduction

The purpose of this master thesis consist of two differents parts:

1. By means of the maple program, and using two differents method's, obtain the dynamical model of a serial Robot that in our case will have three or two links.
2. Estimate inertial parameters of this Robot ,in a pendulum and in a real plant, that in our case will be the inertia-wheel.

1.1 Estimation of inertial parameters

There are three main methods for restimating the inertial parameters,
First ,physical experiments: If we could disassemble the robot to isolate each part of it, then the following parameters could be obtain by physical experiments (Armstrong):

- The mass could be weighed directly
- The coordinates of the center-of –mass could be estimated by determining counterbalanced of points of the link.
- The diagonal elements of the inertia tensor could be obtained by pendulum motion.

Second, using CAD/CAM models: all robotics CAD/CAM packages provides tools to calculate the inertia from 3D models. This method is prone to errors due to the fact that the geometry of the links is complicated to define precisely, and those certain parts such bearings,bolts,nuts and washers are generally neglected.
Third,identification,this approach is based on the analysis of the "input/output" behaviour of the robot on some planned motion and on estimation of the parameters values minimizing the differences between a function of the real robot variables and its mathematical model. This method has been used extensively and was found to be the best in terms of ease of experimentation and precision of the obtained values.

1.1 Identification procedure

Several schemes have been proposed in the literature to identify the dynamic parameters. These methods presents the following features:

- The use of a linear model in the dynamic parameters
- The construction of an over determined linear systems equations by applying the identification model at sufficient number of points along some trajectories of the robot. In general a constant sampling rate is used between the different points
- The estimation of the parameters values using linear regression technique

2. Different algorithm types applied to estimation.

2.1 Least squares

The least-squares method can be applied to large variety of problems. It is particularly simple for mathematical model that can be written in the form[1],

$$y(i) = \varphi_1(i)\theta_1^0 + \varphi_2(i)\theta_2^0 + \cdots + \varphi_n(i)\theta_n^0 = \varphi^T(i)\theta^0 \tag{2.1}$$

where y is the observed variable $\theta_1^0, \theta_2^0, \ldots, \theta_n^0$ are parameters of the model to be determined, and $\varphi_1, \varphi_2, \ldots, \varphi_n$ are known functions that may depend on other know variables. The vectors

$$\varphi^T(i) = \begin{pmatrix} \varphi_1(i) & \varphi_2(i) & \cdots & \varphi_n(i) \end{pmatrix} \tag{2.2}$$

$$\theta^0 = \begin{pmatrix} \theta_1^0 & \theta_2^0 & \cdots & \theta_n^0 \end{pmatrix}^T \tag{2.3}$$

$$\hat{\theta} = \left(\Phi^T \Phi\right)^{-1} \Phi^T Y \tag{2.4}$$

have also been introduced. The model is indexed by the variable *i*, which often denotes time. The variables φ_i are called the regression variables or the regressors and the model(2.1) is called regression model. Pairs of observations and regressor are obtained from n experiment. The problem is to determine the parameters in such a way that the outputs computed from the model are as closely as possible with the measured variables *y(i)* in the sense of least-squares.

Least squares has several attractive features for purposes of identification:

1. Large errors are penalized.
2. The least-squares estimates can be obtained by straightforward matrix algebra.
3. The least-squares criterion is related to statistical variance, and the properties can be analysed according to statistical criteria.

2.2 Linear estimation algorithm for affine parametric non-linear system proposed by Middleton and Goodwin.

Consider the following parametric non-linear model

$$\begin{cases} \dot{x}_1 = f_0(x) + F^T(x,u)\theta \\ \dot{x}_2 = x_1 \end{cases} \tag{2.5}$$

with

$x_1, x_2 \in R^n$	state variables
$f_0(x) \in R^n, F^T(x,u) \in R^{n\times p}$	non-linear functions
$\theta \in R^p$	vector of unknown constant parameters

and suppose that x_1 and x_2 are known, but not $\dot{x}_1$.Let us apply the stable filter $\frac{1}{s+1}$ to the first equation of (2.5), in order to avoid the calculation of the joint accelerations, then we have

$$\frac{s}{s+1}x_1 = \frac{1}{s+1}f_0(x) + \frac{1}{s+1}F^T(x,u)\theta \tag{2.6}$$

Denoting the filtered versions of the known quantities x_1, $f_0(x)$ and $F^T(x,u)$ by

$$x_{1,f} = \frac{1}{s+1}x_1 \tag{2.7}$$

$$f_{0,f}(x) = \frac{1}{s+1}f_0(x) \tag{2.8}$$

$$F_f^T(x,u) = \frac{1}{s+1}F^T(x,u) \tag{2.9}$$

we can rewrite (2.6) as

$$x_1 = f_{0,f}(x) + F_f^T(x,u)\theta + x_{1,f} \tag{2.10}$$

If instead of the unknown θ we use its estimated $\hat{\theta}$,the corresponding predicted value of x_1 is

$$\hat{x}_1 = f_{0,f}(x) + F_f^T(x,u)\hat{\theta} + x_{1,f} \tag{2.11}$$

and the prediction error e is related to the estimation error $\tilde{\theta} = \theta - \hat{\theta}$ as follows:

$$e = x_1 - \hat{x}_1 = \tilde{x}_1 = F_f^T(x,u)\tilde{\theta} \tag{2.12}$$

We propose the following unnormalized gradient-type estimation algorithm:

$$\dot{\hat{\theta}} = h \cdot F_f(x,u) \cdot e \tag{2.13}$$

with h positive constant
The following lemma establishes some of the properties of the above estimator.

Lemma 1 The estimator(2.13) applied to the system(2.12),yields the following properties

1. $\tilde{\theta}$ is bounded.
2. If $\operatorname{rank}(F_f(x,u)F_f^T(x,u)) = p$ then $\lim_{t\to\infty} \hat{\theta} = 0$.

Proof. Consider the Lyapunov function

$$V(\tilde{\theta}) = \frac{1}{2}\tilde{\theta}^T\tilde{\theta} \tag{2.14}$$

using (2.12) and (2.13) we can show that

$$\dot{V} = -h\tilde{\theta}^T F_f(x,u)F_f^T(x,u)\tilde{\theta} \tag{2.15}$$

$\dot{V}$ is negative definite whenever $\operatorname{rank}(F_f(x,u)F_f^T(x,u)) = p$,otherwise is negative semidefinite, than we will see the desidered result in the next chapter.

2.3 Unnormalized least squares estimation

The dynamic equations of motion for a general rigid link manipulator having n degrees of freedom can be describes as follows:

$$\frac{d}{dt}\left\{I(\theta)\dot{\theta}\right\} = \underset{\sim}{\tau}_G(\theta) + \underset{\sim}{\tau}_M + \frac{1}{2}\frac{d}{d\theta}\left[\dot{\theta}^T I(\theta)\dot{\theta}\right] \tag{2.16}$$

where

$\underset{\sim}{\theta} \in R^n$ is the robot joint angles;

$\underset{\sim}{\dot{\theta}} \in R^n$ is the robot joint angular velocities;

$I(\underset{\sim}{\theta}) = I^T(\underset{\sim}{\theta}) \;>\; 0, \mathrm{I}(\underset{\sim}{\theta}) \in R^{n\times n}$ is the inertia matrix

$\underset{\sim}{\tau}_G \in R^n$ is the gravity torque vector;

$\underset{\sim}{\tau}_m \in R^n$ is the motor or actuatortorque vector

We will assume that some parameters are known, by physical mesurements, and we take the view that is illogical to estimate known quantities.
We now define:

$$I(\theta) = J(\theta) + \sum_{k=1}^{N} K_k(\theta) m_k \tag{2.17}$$

where $J(\theta)$ represents the known portion of the inertia matrix and $K_k(\theta)$ is the gravity matrix corresponding to the location of the kth mass.
We can rewrite then:

$$\utilde{\tau}_G(\utilde{\theta}) = \utilde{G}(\utilde{\theta}) + \sum_{k=1}^{N} \utilde{H}_k(\utilde{\theta}) m_k \tag{2.18}$$

and in view of (2.17) we shall rewrite

$$\frac{1}{2}\frac{d}{d\utilde{\theta}}\left[\dot{\utilde{\theta}}^T I(\theta)\dot{\utilde{\theta}}\right] = \utilde{D}(\utilde{\theta},\dot{\utilde{\theta}}) + \sum_{k=1}^{N} E_k(\utilde{\theta},\dot{\utilde{\theta}}) m_k \tag{2.19}$$

Using (2.17),(2.18) and (2.19) we can rewrite (2.16) as:

$$\phi^T \utilde{M} = \utilde{Y} + \tau_M \tag{2.19}$$

where

$$\utilde{Y} = -\frac{d}{dt}\left\{J(\utilde{\theta})\dot{\utilde{\theta}} + \utilde{G}(\utilde{\theta}) + \utilde{D}(\utilde{\theta},\dot{\utilde{\theta}})\right\} \tag{2.20}$$

$$\utilde{M}^T = \begin{bmatrix} m_1 & m_2 \cdots\cdots & m_N \end{bmatrix} \tag{2.21}$$

$\phi \in R^{N\times n}$,and the kth row of ϕ is

$$\phi_k = \left[\frac{d}{dt}\left\{K_k(\utilde{\theta})\dot{\utilde{\theta}} - \utilde{H}_k(\utilde{\theta}) - \utilde{E}(\utilde{\theta},\dot{\utilde{\theta}})\right\}\right]^T \tag{2.22}$$

Note that equation (2.19) is linear in the parameter vector, $\utilde{M}$,and so linear estimation can be used provided ϕ , $\utilde{Y}$ and $\utilde{\tau}_m$ are available.
Now we will suppose that only velocities and position are kown.In order to achieve this, we operate on the left of (2.22) by $\dfrac{\omega}{D+\omega}$ where $D = \dfrac{d}{dt}$ and ω is some constant.

$$\psi^T \utilde{M} = \utilde{Y}_F + \tau_F \tag{2.23}$$

where

$$\psi^T = \left(\frac{\omega}{D+\omega}\right)\phi \tag{2.24}$$

$$\underset{\sim}{Y}_F = \left(\frac{\omega}{D+\omega}\right)\underset{\sim}{Y} \tag{2.25}$$

and

$$\underset{\sim}{\tau}_F = \left(\frac{\omega}{D+\omega}\right)\underset{\sim}{\tau}_m \tag{2.26}$$

ψ and $\underset{\sim}{Y}_F$ are functions of $\underset{\sim}{\theta}$ and $\dot{\underset{\sim}{\theta}}$,but not $\ddot{\underset{\sim}{\theta}}$.The dependence of ψ and $\underset{\sim}{Y}_F$ on $\ddot{\underset{\sim}{\theta}}$ has been eliminated by the filtering.
We propose that estimation be based on (2.23) as follows. Given an estimated $\hat{\underset{\sim}{M}}$, of $\underset{\sim}{M}$ define the vector prediction error, $\underset{\sim}{e}(\hat{\underset{\sim}{M}})$ by

$$\underset{\sim}{e}(\hat{\underset{\sim}{M}}) = Y_F + \underset{\sim}{\tau}_F - \psi^T(\hat{\underset{\sim}{M}}) = -\psi^T(\tilde{\underset{\sim}{M}}) \tag{2.27}$$

where $\tilde{\underset{\sim}{M}} = \hat{\underset{\sim}{M}} - \underset{\sim}{M}$

A large number of parameter estimation procedures can be derived on the above equations by simple extensions to the estimators in Goodwin and Mayne(1985).Since the robot equations are coupled and non-linear, additional problem arise in the stability analysis of adaptive controllers. We propose the following unnormalized least squares estimation algorithm:

$$\dot{\hat{\underset{\sim}{M}}} = \alpha \cdot P \cdot \Psi \cdot \underset{\sim}{e} \tag{2.28}$$

$$\dot{P} = -\alpha \cdot P \cdot \Psi \cdot \Psi^T \cdot P \tag{2.29}$$

where $P \in R^{N\times N}, P(0) = P(0)^T > 0$ and α is a positive ,constant. The following lemma established some of the properties of the above estimator.

Lemma 1 The estimator(2.28),(2.29) applied to the system(2.27),yields the following properties

1. $\hat{M}$ is bounded.
2. $\underset{\sim}{e}$ belongs to L_2

Proof. Consider the Lyapunov function

$$V = \underset{\sim}{\tilde{M}}^T P^{-1} \underset{\sim}{\tilde{M}} \tag{2.30}$$

using (2.27) to (2.29) we can show that

$$\dot{V} = -\alpha \underset{\sim}{e}^T \underset{\sim}{e} \tag{2.31}$$

The desired results then follow.
Note also that any modification to (2.29) can be included provided:

1. The modification is positive Semi-definite.
2. The modification is such that *P* remains bounded for all time.

3. Method's to obtain the model

3.1 Newton-Euler iterative method

Consist of two different steps

First, Outwards iterations to compute velocities and accelerations.
In order to compute inertial forces acting on the links it is necessary to compute the rotational velocity and linear and rotational acceleration of the center of mass of each link of the manipulator at given at any instant.
These computations will be done in an iterative nature starting with link 1 and moving sucessively, link by link to, outward to link n.
The propagation of rotational velocity from link to link is given by,

$$^{i+1}\omega_{i+1} = {}^{i+1}_{i}R\,{}^{i}\omega_i + \dot{\theta}_{i+1}\,{}^{i+1}\hat{Z}_{i+1} \tag{3.1}$$

The propagation of rotational acceleration from link to the next link is given by,

$$^{i+1}\dot{\omega}_{i+1} = {}^{i+1}_{i}R\,{}^{i}\dot{\omega}_i + {}^{i+1}_{i}R\,{}^{i}\omega_i \times \dot{\theta}_{i+1}\,{}^{i+1}\hat{Z}_{i+1} + \ddot{\theta}_{i+1}\,{}^{i+1}\hat{Z}_{i+1} \tag{3.2}$$

When the joint is *i+1* is prismatic, this simplifies to

$$^{i+1}\dot{\omega}_{i+1} = {}^{i+1}_{i}R\,{}^{i}\dot{\omega}_i \tag{3.3}$$

The linear acceleration of each link frame origin is

$$^{i+1}\dot{v}_{i+1} = {}^{i+1}_{i}R\left[{}^{i}\dot{\omega}_i \times {}^{i+1}P_{i+1} + {}^{i}\omega_i \times \left({}^{i}\omega_i \times {}^{i}P_{i+1}\right) + {}^{i}\dot{v}_i\right] \tag{3.4}$$

which for prismatic joint *i+1* becomes

$$
\begin{aligned}
{}^{i+1}\dot{v}_{i+1} = {}^{i+1}_{i}R\left[{}^{i}\dot{\omega}_i \times {}^{i+1}P_{i+1} + {}^{i}\omega_i \times \left({}^{i}\omega_i \times {}^{i}P_{i+1}\right) + {}^{i}\dot{v}_i\right] \\
+ 2\,{}^{i+1}\omega_{i+1} \times \dot{d}_{i+1}\,{}^{i+1}\hat{Z}_{i+1} + \ddot{d}_{i+1}\,{}^{i+1}\hat{Z}_{i+1}
\end{aligned}
\tag{3.5}
$$

We also need the linear acceleration of the center of mass of each link, which also can be found by applying

$$
{}^{i+1}\dot{v}_{c_{i+1}} = {}^{i+1}\dot{\omega}_{i+1} \times {}^{i+1}P_{c_{i+1}} + {}^{i+1}\omega_{i+1} \times \left({}^{i+1}\omega_{i+1} \times {}^{i}P_{c_{i+1}}\right) + {}^{i+1}\dot{v}_{i+1} \tag{3.6}
$$

Where we imagine a frame, $\{C_i\}$, attached to each link with its origin located at the center of mass of the link, and with the same orientation as the link frame, $\{i\}$.
Note that the application of the equations to link 1 is especially simple since

$$
{}^{0}\omega_0 = {}^{0}\dot{\omega}_0 = 0 \tag{3.7}
$$

Having computed the linear and angular accelerations of the mass center of each link, we can apply the Newton-Euler equations to compute the inertial force and torque acting at the center of mass of each link. Thus we have,

$$
F_i = m\dot{v}_{c_i} \tag{3.8}
$$

$$
N_i = {}^{c_i}I\dot{\omega}_i + \omega_i \times {}^{c_i}I\omega_i \tag{3.9}
$$

Where $\{C_i\}$ has its origin at the center of mass of the link, and has the same orientation as the link frame, $\{i\}$.
Second, Inwards iterations to compute forces and torques.

Having computed the forces and torques acting on each link, it now remains to calculate the joint torques which will result in these net forces and torques being applied to each link.
We can do this by writing a force balance and moment balance equation based on a free body diagram of a typical link.
Each link has forces and torques exerted on it by its neighbors,and in addition experiences an inertial the force and torque.
The complete algorithm for computing joint torques from the motion of the joints is composed of two parts. First, link velocities and accelerations are iteratively computed from link 1 to link n and the Newton-Euler equation are applied to each link. Second, forces and torques of interaction and joint actuator torques are computed recursively from link n back to link 1.The equations are summarized below from the case of all joints rotational

Outward iterations: i: 0 to 2

$$ {}^{i+1}\omega_{i+1} = {}^{i+1}_{i}R\,{}^{i}\omega_i + \dot{\theta}_{i+1}\,{}^{i+1}\hat{Z}_{i+1} \tag{3.10} $$

$$ {}^{i+1}\dot{\omega}_{i+1} = {}^{i+1}_{i}R\,{}^{i}\dot{\omega}_i + {}^{i+1}_{i}R\,{}^{i}\omega_i \times \dot{\theta}_{i+1}\,{}^{i+1}\hat{Z}_{i+1} + \ddot{\theta}_{i+1}\,{}^{i+1}\hat{Z}_{i+1} \tag{3.11} $$

$$ {}^{i+1}\dot{v}_{i+1} = {}^{i+1}_{i}R\left({}^{i}\dot{\omega}_i \times {}^{i}P_{i+1} + {}^{i}\omega_i \times \left({}^{i}\dot{\omega}_i \times {}^{i}P_{i+1}\right) + {}^{i}\dot{v}_i\right) \tag{3.12} $$

$$ {}^{i+1}\dot{v}_{c_{i+1}} = {}^{i+1}\dot{\omega}_{i+1} \times {}^{i+1}P_{c_{i+1}} + {}^{i+1}\omega_{i+1} \times \left({}^{i+1}\omega_{i+1} \times {}^{i}P_{c_{i+1}}\right) + {}^{i+1}\dot{v}_{i+1} \tag{3.13} $$

$$ {}^{i+1}F_{i+1} = m_{i+1}\,{}^{i+1}\dot{v}_{c_{i+1}} \tag{3.14} $$

$$ {}^{i+1}N_{i+1} = {}^{c_{i+1}}I_{i+1}\,{}^{i+1}\dot{\omega}_{i+1} + {}^{i+1}\omega_{i+1} \times {}^{c_{i+1}}I_{i+1}\,{}^{i+1}\omega_{i+1} \tag{3.15} $$

Inward iterations: i: 3 to 1

$$ {}^{i}f_i = {}^{i}_{i+1}R\,{}^{i+1}f_{i+1} + {}^{i}F_i \tag{3.16} $$

$$ {}^{i}n_i = {}^{i}N_i + {}^{i}_{i+1}R\,{}^{i+1}n_{i+1} + {}^{i}P_{c_{i+1}} \times {}^{i}F_i + {}^{i}P_{i+1} \times {}^{i}_{i+1}R\,{}^{i+1}f_{i+1} \tag{3.17} $$

$$ \tau_i = {}^{i}n_i^{T}\,{}^{i}\hat{Z}_i \tag{3.18} $$

Mass distribution

$$ A_I = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yx} & I_{yy} & I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{bmatrix} \tag{3.19} $$

$$ I_{xx} = \int_0^w\int_0^h\int_0^l \left(y^2+z^2\right)\rho\,dxdydz \qquad I_{xy} = \int_0^w\int_0^h\int_0^l (xy)\rho\,dxdydz $$

$$ I_{yy} = \int_0^w\int_0^h\int_0^l \left(x^2+z^2\right)\rho\,dxdydz \qquad I_{xz} = \int_0^w\int_0^h\int_0^l (xz)\rho\,dxdydz $$

$$ I_{zz} = \int_0^w\int_0^h\int_0^l \left(x^2+y^2\right)\rho\,dxdydz \qquad I_{yz} = \int_0^w\int_0^h\int_0^l (yz)\rho\,dxdydz $$

3.1.1 Example

We assume that the mass distribution is simple; all mass exists as a point mass at the distal end of each link. The masses are m_1 and m_2.
The vectors which locate the center of mass for each link are

$$ {}^{1}P_{C_1} = l_1\hat{X}_1 \tag{3.20} $$

$$^{2}P_{C_2} = l_2\hat{X}_2 \tag{3.21}$$

Because of the point assumption, the inertia tensor written at the center of mass for each link is the zero matrix:

$$^{C_1}I_C = 0 \tag{3.22}$$
$$^{C_2}I_C = 0 \tag{3.23}$$

There are no forces acting on the end effector, and so we have

$$f_3 = 0 \tag{3.24}$$
$$n_3 = 0 \tag{3.25}$$

The base of the robot is not rotating, and hence we have

$$\omega_0 = 0 \tag{3.26}$$
$$\dot{\omega}_0 = 0 \tag{3.27}$$

To include gravity forces we will use

$$^{0}\dot{v}_0 = g\hat{Y}_0 \tag{3.28}$$

The rotation between succesives link frames is given by

$$_{i+1}^{\;\;i}R = \begin{bmatrix} c\theta_{i+1} & -s\theta_{i+1} & 0 \\ s\theta_{i+1} & c\theta_{i+1} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.29}$$

$$_{\;\;i}^{i+1}R = \begin{bmatrix} c\theta_{i+1} & s\theta_{i+1} & 0 \\ -s\theta_{i+1} & c\theta_{i+1} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.30}$$

And the transformation is given by

$$ {}^{i-1}_{i}T = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i \cdot c\alpha_{i-1} & c\theta_i \cdot c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1} \cdot d_i \\ s\theta_i \cdot s\alpha_{i-1} & c\theta_i \cdot s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1} \cdot d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.31} $$

The outwards iterations for link 1 are as follows:

$$ {}^{1}\omega_1 = \dot{\theta}_1 \hat{Z}_1 = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 \end{bmatrix} \tag{3.32} $$

$$ {}^{1}\dot{\omega}_1 = \ddot{\theta}_1 \hat{Z}_1 = \begin{bmatrix} 0 \\ 0 \\ \ddot{\theta}_1 \end{bmatrix} \tag{3.33} $$

$$ {}^{1}v_1 = \begin{bmatrix} c_1 & s_1 & 0 \\ -s_1 & c_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ g \\ 0 \end{bmatrix} = \begin{bmatrix} gs_1 \\ gc_1 \\ 0 \end{bmatrix} \tag{3.34} $$

$$ {}^{1}\dot{v}_{C_1} = \begin{bmatrix} 0 \\ l_1\ddot{\theta}_1 \\ 0 \end{bmatrix} + \begin{bmatrix} -l_1\dot{\theta}_1^2 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} gs_1 \\ gc_1 \\ 0 \end{bmatrix} \tag{3.35} $$

$$ {}^{1}F_1 = \begin{bmatrix} -m_1 l_1 \dot{\theta}_1^2 + m_1 g s_1 \\ -m_1 l_1 \ddot{\theta}_1 + m_1 g c_1 \\ 0 \end{bmatrix} \tag{3.36} $$

$$ {}^{1}N_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.37} $$

The outwards iterations for link 2 are as follows:

$$ {}^{2}\omega_2 = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 + \dot{\theta}_2 \end{bmatrix} \tag{3.38} $$

$$
{}^{2}\dot{\omega}_2 = \begin{bmatrix} 0 \\ 0 \\ \ddot{\theta}_1 + \ddot{\theta}_2 \end{bmatrix} \tag{3.39}
$$

$$
{}^{2}v_2 = \begin{bmatrix} c_2 & s_2 & 0 \\ -s_2 & c_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} gs_1 - l_1\dot{\theta}^2 \\ gc_1 + l_1\ddot{\theta}_1 \\ 0 \end{bmatrix} = \begin{bmatrix} l_1\ddot{\theta}_1 s_2 - l_1\dot{\theta}^2 c_2 + gs_{12} \\ l_1\ddot{\theta}_1 c_2 + l_1\dot{\theta}^2 s_2 + gc_{12} \\ 0 \end{bmatrix} \tag{3.40}
$$

$$
{}^{2}\dot{v}_{C_2} = \begin{bmatrix} 0 \\ l_2(\ddot{\theta}_1 + \ddot{\theta}_2) \\ 0 \end{bmatrix} + \begin{bmatrix} -l_1(\dot{\theta}_1 + \dot{\theta}_2) \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} l_1\ddot{\theta}_1 s_2 - l_1\dot{\theta}^2 c_2 + gs_{12} \\ l_1\ddot{\theta}_1 c_2 + l_1\dot{\theta}^2 s_2 + gc_{12} \\ 0 \end{bmatrix} \tag{3.41}
$$

$$
{}^{2}F_2 = \begin{bmatrix} m_2 l_1\ddot{\theta}_1 s_2 - m_2 l_1\dot{\theta}^2 c_2 + m_2 g s_{12} - m_2 l_2(\dot{\theta}_1 + \dot{\theta}_2) \\ m_2 l_1\ddot{\theta}_1 c_2 - m_2 l_1\dot{\theta}^2 s_2 + m_2 g c_{12} - m_2 l_2(\ddot{\theta}_1 + \ddot{\theta}_2) \\ 0 \end{bmatrix} \tag{3.42}
$$

$$
{}^{2}N_2 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.43}
$$

Inwards iterations for link 2 are as follows:

$$
{}^{2}f_2 = {}^{2}F_2 \tag{3.44}
$$

$$
{}^{2}n_2 = \begin{bmatrix} 0 \\ 0 \\ m_2 l_1 l_2 c_2\ddot{\theta}_1 + m_2 l_1 l_2 s_2\dot{\theta}_1^2 + m_2 l_2 g c_{12} + m_2 l_2^2(\ddot{\theta}_1 + \ddot{\theta}_2) \end{bmatrix} \tag{3.45}
$$

Inwards iterations for link 1 are as follows:

$$
\begin{aligned}
{}^{1}f_1 = {} & \begin{bmatrix} c_2 & s_2 & 0 \\ -s_2 & c_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} m_2 l_1\ddot{\theta}_1 s_2 - m_2 l_1\dot{\theta}^2 c_2 + m_2 g s_{12} - m_2 l_2(\dot{\theta}_1 + \dot{\theta}_2) \\ m_2 l_1\ddot{\theta}_1 c_2 - m_2 l_1\dot{\theta}^2 s_2 + m_2 g c_{12} - m_2 l_2(\ddot{\theta}_1 + \ddot{\theta}_2) \\ 0 \end{bmatrix} \\
& + \begin{bmatrix} -m_1 l_1\dot{\theta}_1^2 + m_1 g s_1 \\ -m_1 l_1\ddot{\theta}_1 + m_1 g c_1 \\ 0 \end{bmatrix}
\end{aligned} \tag{3.46}
$$

$$
\begin{aligned}
{}^{1}n_1 &= \begin{bmatrix} 0 \\ 0 \\ m_2 l_1 l_2 c_2 \ddot{\theta}_1 + m_2 l_1 l_2 s_2 \dot{\theta}_1^2 + m_2 l_2 g c_{12} + m_2 l_2^2 (\ddot{\theta}_1 + \ddot{\theta}_2) \end{bmatrix} \\
&+ \begin{bmatrix} 0 \\ 0 \\ m_2 l_1^2 \ddot{\theta}_1 + m_1 l_1 g c_1 \end{bmatrix} \\
&+ \begin{bmatrix} 0 \\ 0 \\ m_2 l_1^2 \ddot{\theta}_1 - m_2 l_1 l_2 s_2 (\dot{\theta}_1 + \dot{\theta}_2)^2 + m_2 l_2 g s_2 s_{12} + m_2 l_1 l_2 c_2 (\ddot{\theta}_1 + \ddot{\theta}_2)^2 + m_2 l_2 g c_2 c_{12} \end{bmatrix}
\end{aligned} \tag{3.47}
$$

Extracting the $\hat{Z}$ components of the ${}^{i}n_i$,we find the joints torques:

$$
\begin{aligned}
\tau_1 &= m_2 l_2^2 (\ddot{\theta}_1 + \ddot{\theta}_2) + m_2 l_1 l_2 c_2 (2\ddot{\theta}_1 + \ddot{\theta}_2) + (m_1 + m_2) l_1^2 \ddot{\theta}_1 \\
&\quad - 2 m_2 l_1 l_2 s_2 \dot{\theta}_1 \dot{\theta}_2 + m_2 l_2 g c_{12} + (m_1 + m_2) l_1 g c_1
\end{aligned} \tag{3.48}
$$

$$
\tau_2 = m_2 l_2^2 (\ddot{\theta}_1 + \ddot{\theta}_2) + m_2 l_1 l_2 c_2 \ddot{\theta}_1 + m_2 l_1 l_2 s_2 \dot{\theta}_1^2 + m_2 l_2 g c_{12} + (m_1 + m_2) l_1 g c_1 \tag{3.49}
$$

3.2 Using the Lagrange method

To present the general form of the dynamic model of robots and to get an insight into its properties. We consider an ideal system without friction or elasticity, exerting neither forces nor moments on the environment.
The Lagrange formulation describes the behaviour of a dynamic system in terms of work and energy stored in system. The Lagrange equations are commonly written in the form:

$$
\Gamma_i = \frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} \quad for\ i = 1, \ldots n \tag{3.50}
$$

where *L* is the Lagrangian of the robot defined as the difference between the kinetic energy *E* and the potential energy *U* of the system:

$$
L=E-U \tag{3.51}
$$

Define the Kinetic energy as,

$$
\frac{1}{2}\dot{q}^T A q^T \tag{3.52}
$$

where A is the (*nxn*) symmetric and positive definite inertia matrix of the robot. Its elements are function of the joint positions.
Since the potential energy is a fuction of the joints positions equations (3.51) and (3.52) lead to,

$$\Gamma = A(q)\ddot{q} + C(q,\dot{q})\dot{q} + Q(q) \tag{3.53}$$

where

$C(q,\dot{q})\dot{q}$ is the (*nx1*) vector of coriolis and centrifugal torques, such that:

$$C\dot{q} = \dot{A}\dot{q} - \frac{\partial E}{\partial q} \tag{3.54}$$

$Q = \begin{bmatrix} Q_1 & \dots & Q_n \end{bmatrix}^T$ is the vector of gravity torques.

Consequently, the dynamic model of a robot is described by n coupled and non-linear second order differential equations.
There exits several forms for the vector $C(q,\dot{q})\dot{q}$ .Using the Christoffell symbols $c_{i,j}$,the (*i,j*) element of the matrix *C* can be rewritten as

$$\begin{cases} C_{ij} = \sum_{K=1}^{n} c_{i,jk}\dot{q}_k & (3.55) \\ c_{i,jk} = \frac{1}{2}\left[\frac{\partial A_{ij}}{\partial q_k} + \frac{\partial A_{ik}}{\partial q_j} + \frac{\partial A_{jk}}{\partial q_i}\right] & (3.56) \end{cases}$$

The elements of *A*,*C and Q* are functions of the geometric and inertial parameters of the robots .

3.2.1 How to obtain the matrices *A*,*C*,*Q*

To compute the elements of *A*,*C* and *Q,*we begin by symbolically computing the expressions of the kinetic and potential energies of all the links of the robot.

Computation of the kinetic energy:

The kinetic energy is given as follow,

$$E = \sum_{j=1}^{n} E_j \tag{3.57}$$

where E_j denotes the kinetic energy of link j,which can be computed by:

$$E_j = \frac{1}{2}\left[\omega_j^T I_{Gj}\omega_j + M_j V_{Gj}{}^T V_{Gj}\right] \qquad (3.58)$$

If observe the graph() the velocity of the center of mass can be expressed as

$$V_{G_j} = V_j + \omega_j \times S_j \qquad (3.59)$$

and

$$J_j = I_{Gj} - M_j \hat{S}_j \hat{S}_j \qquad (3.60)$$

then the Eq. becomes:

$$E = \frac{1}{2}\left[\omega_j^T J_j \omega_j + M_j V_j^T V_j + 2MS_j^T\left(V_j \times \omega_j\right)\right] \qquad (3.61)$$

Equation(3.58) is not linear in the coordinates of the vector S_j.On the contrary equation(3.61) is linear in the elements of M_j, MS_j *and* J_j,which we call the standard inertial parameters. The linear and angular velocities V_j and ω_j are computing using the following recursive equations

$$\omega_j = \omega_{j-1} + \bar{\sigma}\dot{q}_j a_j \qquad (3.62)$$

$$V_j = V_{j-1} + \omega_{j-1} \times L_j + \sigma_j \dot{q}_j a_j \qquad (3.63)$$

where $\sigma_j = 0$ if joint j is revolute, $\sigma_j = 1$ if joint j is prismatic, and $\bar{\sigma}_j = 1 - \sigma_j$

If the base of the robot is fixed, the previous equations are initialized by $V_0 = 0$ *and* $\omega_0 = 0$.
All the elements appearing in equation(3.61) must be expressed in the same frame. The most efficient way to express them relative to frame R_j.Therefore, the equations (3.61),(3.62)and (3.63) are rewritten as,

$$E = \sum_{j=1}^{n} E_j = \frac{1}{2}\left[{}^j\omega_j^T\,{}^jJ_j\,{}^j\omega_j + M_j\,{}^jV_j^T\,{}^jV_j + 2\,{}^jMS_j^T\left({}^jV_j x\,{}^j\omega_j\right)\right] \qquad (3.64)$$

$${}^j\omega_j = {}^jA_{j-1}\,{}^{j-1}\omega_{j-1} + \bar{\sigma}_j \dot{q}_j\,{}^ja_j \qquad (3.65)$$

$${}^jV_j = {}^jA_{j-1}\left({}^{j-1}V_{j-1} + {}^{j-1}\omega_{j-1} x\,{}^{j-1}P_j\right) + \bar{\sigma}_j \dot{q}_j\,{}^ja_j \qquad (3.66)$$

The parameters ${}^{j}J_{j}$ and ${}^{j}MS_{j}$ are constants.

3.2.2 Computation of the potential energy

The potential energy is given by

$$U=\sum_{j=1}^{n}U_{j}=\sum_{j=1}^{n}-M_{j}g^{T}\left(L_{0,j}+S_{j}\right) \tag{3.67}$$

where $L_{0,j}$ is the position vector from the origin O_{0} *to* O_{j}.Projecting the vectors appearing in (3.67) into frame R_{0},we obtain:

$$U_{j}=-M_{j}\,{}^{0}g^{T}\left({}^{0}P_{j}+{}^{0}A_{j}\,{}^{j}S_{j}\right) \tag{3.68}$$

an expression then can be rewritten linearly in M_{j}, and elements of $MS_{j}\ as$:

$$U_{j}=-{}^{0}g^{T}\left(M_{j}\,{}^{0}P_{j}+{}^{0}A_{j}\,{}^{j}MS_{j}\right) \tag{3.69}$$

Since the kinetic and potential energies are linear in the elements of $M_{j}, MS_{j}\ and\ J_{j}$,we deduce that the dynamic model is also linear in these parameters.

3.2.3 Dynamic model properties

In this section we summarize some properties of the dynamic model of robots:

1. The inertia matrix A is symmetric and positive definite
2. The energy of link j is function of $\left(q_{1},\ \ldots\ ,q_{j}\right)$ and $\left(\dot{q}_{1},\ \ldots\ ,\dot{q}_{j}\right)$
3. The element A_{ij} is a function of $q_{k+1},\ldots,q_{n}$ with *k*=*min*(*i,j*),and of the inertial parameters of links $r,\ldots,n\ with$ r=*max*(*i,j*)
4. Γ_{i} is a function of the inertial parameters of links $i,\ldots,n$
5. The matrix $\left[\frac{d}{dt}A-2C\left(q,\dot{q}\right)\right]$ is skew-symmetric for the choice of the matrix C given in equation(3.55) and (3.56)

6. The inverse dynamic model is linear in the elements of the standard inertial parameters M_j, MS_j *and* J_j.This property is exploited to identify the dynamic parameters, to reduce the computation burden of the dynamic model.
7. There exist some positive real numbers $a_1, \ldots, a_7$ such that for any values of q and $\dot{q}$ we have:

$$\|A(q)\| \le a_1 + a_2\|q\| + a_3\|q\|^2 \tag{3.70}$$

$$\|C(q,\dot{q})\| \le \|\dot{q}\left(a_4 + a_5\|q\|\right)\| \tag{3.71}$$

$$\|Q\| \le a_6 + a_7\|q\| \tag{3.72}$$

where $\|*\|$ indicates a matrix or vector nrm.If the robot has only revolute joints, these relations becomes,

$$\|A(q)\| \le a_1 \tag{3.73}$$

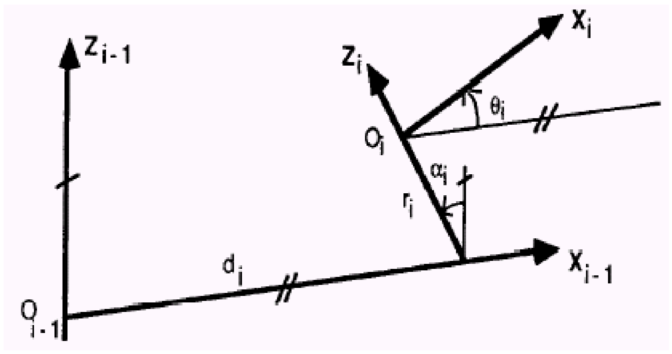
$$\|C(q,\dot{q})\| \le a_4\|\dot{q}\| \tag{3.74}$$

$$\|Q\| \le a_6 \tag{3.75}$$

8. A robot is a passive system which dissipates energy

3.2.4 Example

First let consider the following coordinated system in relation with the next frame



Using the transformation matrix that follow

$$ {}^{i-1}_{i}T = \begin{bmatrix} {}^{i-1}A_i & {}^{i-1}P_i \\ 000 & 1 \end{bmatrix} = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i \cdot c\alpha_i & c\theta_i \cdot c\alpha_i & -s\alpha_i & -s\alpha_i \cdot d_i \\ s\theta_i \cdot s\alpha_i & c\theta_i \cdot s\alpha_i & c\alpha_i & c\alpha_i \cdot d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.76}$$

where:

${}^{i-1}A_i$ defines the orientation of frame i with respect to frame i-1

${}^{i-1}P_i$ defines the position of the origin of the frame i with respect to frame i-1

And considering a three-link manipulator with the following geometrical description:

j	σ_j	α_j	d_j	θ_j	r_j
1	0	0	0	θ_1	0
2	0	$\pi/2$	0	θ_2	0
3	0	0	D_3	θ_3	0

First of all we need to compute the angular and linear velocities

$$ {}^0\omega_0 = 0 \tag{3.77}$$

$$ {}^1\omega_1 = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 \end{bmatrix} \tag{3.78}$$

$$ {}^2\omega_2 = {}^2A_1\,{}^1\omega_1 + \dot{\theta}_2\,{}^2a_2 = \begin{bmatrix} c_2 & 0 & s_2 \\ -s_2 & 0 & c_2 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_2 \end{bmatrix} = \begin{bmatrix} s_2\dot{\theta}_1 \\ c_2\dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} \tag{3.79}$$

$$ {}^3\omega_3 = {}^3A_2\,{}^2\omega_2 + \dot{\theta}_3\,{}^3a_3 = \begin{bmatrix} c_3 & s_3 & 0 \\ -s_3 & c_3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_2\dot{\theta}_1 \\ c_2\dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_3 \end{bmatrix} = \begin{bmatrix} s_{23}\dot{\theta}_1 \\ c_{23}\dot{\theta}_1 \\ \dot{\theta}_1 + \dot{\theta}_2 \end{bmatrix} \tag{3.80}$$

$$ {}^0v_0 = 0 \tag{3.81}$$

$$ {}^1v_1 = 0 \tag{3.82}$$

$$ {}^1v_2 = {}^1v_1 + {}^1\omega_1 \times {}^1P_2 = 0 \tag{3.83}$$

$$^{2}v_{2} = 0 \tag{3.84}$$

$$^{2}v_{3} = {}^{2}\omega_{2} \times {}^{2}P_{3} = \begin{bmatrix} 0 \\ D_{3}\dot{\theta}_{2} \\ -c_{2}D_{3}\dot{\theta}_{1} \end{bmatrix} \tag{3.85}$$

$$^{3}v_{3} = {}^{3}A_{2}\,{}^{2}v_{3} = \begin{bmatrix} s_{3}D_{3}\dot{\theta}_{2} \\ c_{3}D_{3}\dot{\theta}_{2} \\ -c_{2}D_{3}\dot{\theta}_{1} \end{bmatrix} \tag{3.86}$$

Second step is to compute the inertia matrix *A*,

$$^{j}MS_{j} = \begin{bmatrix} MX_{j} \\ MY_{j} \\ MZ_{j} \end{bmatrix} \tag{3.87}$$

$$^{j}J_{j} = \begin{bmatrix} XX_{j} & XY_{j} & XZ_{j} \\ XY_{j} & YY_{j} & YZ_{j} \\ XZ_{j} & ZY_{j} & ZZ_{j} \end{bmatrix}, I_{a} = \begin{bmatrix} I_{a_1} & 0 & 0 \\ 0 & I_{a_2} & 0 \\ 0 & 0 & I_{a_3} \end{bmatrix} \tag{3.88}$$

Obtaining the following elements of the inertia matrix

$$\begin{aligned} A_{11} &= Ia_{1} + ZZ_{1} + ss_{2}XZ_{2} + 2cs_{2}XY_{2} + cc_{2}YY_{2} + ss_{23}XX_{3} \\ &\quad + 2cs_{23}XY_{3} + cc_{23}YY_{3} + 2c_{2}cc_{23}D_{3}MX_{3} - 2c_{2}ss_{23}D_{3}MY_{3} + cc_{2}D_{3}^{2}M_{3} \end{aligned} \tag{3.89}$$

$$A_{12} = s_{2}XZ_{2} + \frac{1}{2}c_{2}XZ_{2} + \frac{1}{2}c_{2}YZ_{2} + s_{23}XZ_{3} + \frac{1}{2}c_{23}YZ_{3} + \frac{1}{2}c_{23}XY_{3} - s_{3}c_{23}D_{3}MZ_{3} \tag{3.90}$$

$$A_{13} = s_{23}XZ_{3} + \frac{1}{2}c_{23}YZ_{3} + \frac{1}{2}c_{23}XY_{3} \tag{3.91}$$

$$A_{22} = Ia_{2} + ZZ_{2} + ZZ_{3} + 2c_{3}D_{3}MX_{3} - 2s_{3}D_{3}MY_{3} + D_{3}^{2}M_{3} \tag{3.92}$$

$$A_{23} = ZZ_{3} + c_{3}D_{3}MX_{3} - s_{3}D_{3}MY_{3} \tag{3.93}$$

$$A_{33} = Ia_{3} + ZZ_{3} \tag{3.94}$$

Computation of the gravity forces

We assume that the gravitational acceleration is given as

$$^{0}g = \begin{bmatrix} 0 \\ 0 \\ g_3 \end{bmatrix}^T \qquad (3.95)$$

and using the equation(3.68) the following solutions are obtained as

$$U = -g_3\left(MZ_1 + s_2 MX_2 + c_2 MY_2 + s_{23} MX_3 + c_{23} MY_3 + D_3 s_2 M_3\right) \qquad (3.96)$$

$$Q_1 = 0 \qquad (3.97)$$

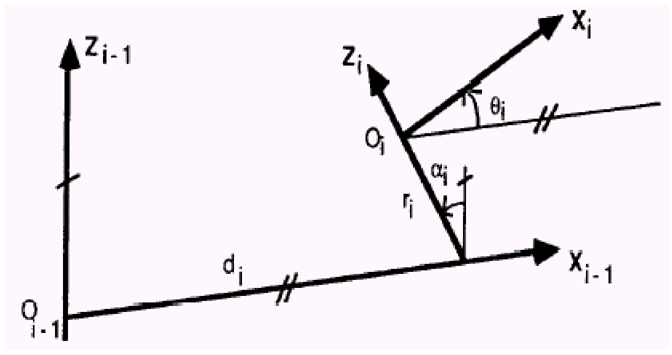
$$Q_2 = -g_3\left(c_2 MX_2 - s_2 MY_2 + c_{23} MX_3 - s_{23} MY_3 + D_3 c_2 M_3\right) \qquad (3.98)$$

$$Q_3 = -g_3\left(c_{23} MX_3 - s_{23} MY_3\right) \qquad (3.99)$$

4. Programs in Maple

4.1 Program using Dynamic model

Before build the program in maple is very important to know our coordinated system In relation with the next frame



$$^{i-1}_{\ \ i}T = \begin{bmatrix} ^{i-1}A_i & ^{i-1}P_i \\ 000 & 1 \end{bmatrix} = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i \cdot c\alpha_i & c\theta_i \cdot c\alpha_i & -s\alpha_i & -s\alpha_i \cdot d_i \\ s\theta_i \cdot s\alpha_i & c\theta_i \cdot s\alpha_i & c\alpha_i & c\alpha_i \cdot d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (4.1)$$

where:

${}^{i-1}A_i$ defines the orientation of frame i with respect to frame i-1

${}^{i-1}P_i$ defines the position of the origin of the frame i with respect to frame i-1

In the next code we have developed a program to obtain the model of a robot with the next geometric parameters:

j	σ_j	α_j	d_j	θ_j	r_j
1	0	0	0	θ_1	0
2	0	$\pi/2$	0	θ_2	0

```
 with(linalg):
with(linalg,randmatrix):
c[1]:=0:
c[2]:=0:
alpha[1]:=0:
cos(alpha[2]):=0:
sin(alpha[2]):=1:
d[1]:=0:
d[2]:=0:
r[1]:=0:
r[2]:=0:
Ia:=matrix(3,3,[Ia1,0,0,0,Ia2,0,0,0,Ia3]):
for j from 1 by 1 to 2 do
ms[j]:=matrix(3,1,[mx[j],my[j],mz[j]]):
A[j]:=transpose(matrix(3,3,[cos(q[j]),-
sin(q[j]),0,cos(alpha[j])*sin(q[j]),cos(alpha[j])*cos(q[
j]),-
sin(alpha[j]),sin(alpha[j])*sin(q[j]),sin(alpha[j])*cos(
q[j]),cos(alpha[j])])):
J[j]:=matrix(3,3,[XX[j],XY[j],XZ[j],XY[j],YY[j],YZ[j],XZ
[j],XY[j],ZZ[j]]);
a[j]:=matrix(3,1,[0,0,1]):
od:
for j from 1 by 1 to 2 do
p[j]:=matrix(3,1,[d[j],-
r[j]*sin(alpha[j]),r[j]*cos(alpha[j])]):
if (j=1) then
   v[1]:=matrix(3,1,[0,0,0]):
w[1]:=evalm((1-c[1])*diff(q[1],q)*a[1]):
else
```

```
w[j]:=evalm(A[j]&*w[j-1]+(1-c[j])*diff(q[j],q)*a[j]);
v[j]:=evalm(A[j]&*(v[j-1]+convert(crossprod(convert(w[j-
1],vector),convert(p[j],vector)),matrix))+c[j]*diff(q
[j],q)*a[j]);
fi:
od:
 for j from 1 by 1 to 2 do

E[j]:=evalm(1/2*(transpose(w[j])&*J[j]&*w[j]+m[j]*transp
ose(v[j])&*v[j]+2*transpose(ms[j])&*convert((crossprod(c
onvert(v[j],vector),convert(w[j],vector)),matrix))));
od:
Kinetic:=evalm(E[1]+E[2]):
M[1,1]:=coeff(Kinetic[1,1],diff(q[1],q)^2)*2:
vvv:=coeff(Kinetic[1,1],diff(q[1],q)):
M[1,2]:=coeff(vvv,diff(q[2],q)):
M[2,1]:=M[1,2]:
M[2,2]:=coeff(Kinetic[1,1],diff(q[2],q)^2)*2:
Inertia:=matrix(2,2,[M[1,1],M[1,2],M[2,1],M[2,2]])+evalm
(Ia);


for j from 1 by 1 to 2 do
T[j]:=evalm(matrix(4,4,[cos(q[j]),-
sin(q[j]),0,d[j],cos(alpha[j])*sin(q[j]),cos(alpha[j])*c
os(q[j]),-sin(alpha[j]),-
r[j]*sin(alpha[j]),sin(alpha[j])*sin(q[j]),sin(alpha[j])
*cos(q[j]),cos(alpha[j]),r[j]*cos(alpha[j]),0,0,0,1]));
od:
TR[1]:=evalm(T[1]):
TR[2]:=evalm(TR[1]&*T[2]):
for j from 1 by 1 to 2 do
U[j]:=-
matrix(1,4,[0,0,g2,0])&*TR[j]&*matrix(4,1,[ms[j][1,1],ms
[j][2,1],ms[j][3,1],m[j]]);

od:
Potencial:=evalm(U[1]+U[2]):
for j from 1 by 1 to 2 do

    Q[j]=map(diff,evalm(Potencial[1,1]),q[j]):
od:
gravity:=matrix(2,1,[evalm(Q[1]),evalm(Q[2])]):
 for i from 1 by 1 to 2 do
for j from 1 by 1 to 2 do
for k from 1 by 1 to 2 do
        evalm(M[i,k]);
tmp1[k]:=diff(M[i,k],q[j]);
        tmp2[k]:=diff(M[i,j],q[k]):
tmp3[k]:=diff(M[j,k],q[i]):
        res[k]:=1/2*(tmp1[k]+tmp2[k]-tmp3[k]);
   od;
od;
od;
for k from 1 by 1 to 2 do
coriolis:=res[K]*diff(q[k],q);
od:
 for k from 1 by 1 to 2 do
for i from 1 by 1 to 2 do
for j from 1 by 1 to 2 do
tmp1[i,j,k]:=diff(M[i,k],q[j]);
tmp2[i,j,k]:=diff(M[i,j],q[k]);
tmp3[i,j,k]:=diff(M[j,k],q[i]);
res[i,j,k]:=1/2*(tmp1[i,j,k]+tmp2[i,j,k]-tmp3[i,j,k]);
od;
od;
od;
 for i from 1 by 1 to 2 do
for j from 1 by 1 to 2 do
X[i,j]:=res[i,j,1]*diff(q[1],q)+res[i,j,2]*diff(q[2],q);
od;od;
coriolis:=matrix(2,2,[X[1,1],X[1,2],X[2,1],X[2,2]]);
```

After apply this program we obtain:

The inertia matrix:

$M(q) =$

$$A_{11} = Ia_1 + ZZ_1 + ss_2XZ_2 + cs_2XY_2 + cc_2YY_2 + cs_2XY_2 \tag{4.2}$$

$$A_{12} = s_2XZ_2 + \frac{1}{2}c_2XZ_2 + \frac{1}{2}c_2YZ_2 \tag{4.3}$$

$$A_{21} = s_2XZ_2 + \frac{1}{2}c_2XZ_2 + \frac{1}{2}c_2YZ_2 \tag{4.4}$$

$$A_{22} = Ia_2 + ZZ_2 \tag{4.5}$$

where

The coriolis matrix has the following elements:

$C(q,\dot{q}) =$

$$C_{1,1} = \frac{1}{2}((\cos(q_2)XX_2\text{-}\sin(q_2)XY_2)\sin(q_2)+(\sin(q_2)XX_2+ \\ \cos(q_2)XY_2)\cos(q_2)+\frac{1}{2}(\cos(q_2)XY_2\text{-}\sin(q_2)YY_2)\cos(q_2)\text{-} \\ (\sin(q_2)XY_2+\cos(q_2)YY_2)\sin(q_2))\frac{\partial q_2}{\partial t} \tag{4.6}$$

$$C_{1,2} = \frac{1}{2}((\cos(q_2)\ XX_2\ \text{-}\ \sin(q_2)\ XY_2)\ \sin(q_2)+ \\ (\sin(q_2)\ XX_2\ +\ \cos(q_2)\ XY_2)\ \cos(q_2) \\ +\ (\cos(q_2)\ XY_2\ \text{-}\ \sin(q_2)\ YY_2)\ \cos(q_2)\text{-} \\ (\sin(q_2)\ XY_2+\cos(q_2)YY_2) \\ \sin(q_2)))\frac{\partial q_1}{\partial t}+(\cos(q_2)XZ_2\text{-}\frac{1}{2}\sin(q_2)XY_2\text{-}\frac{1}{2}\sin(q_2)YZ_2)\frac{\partial q_2}{\partial t} \tag{4.7}$$

$$C_{2,1} = \frac{1}{2}(\text{-}(\cos(q_2)XX_2\text{-}\sin(q_2)XY_2)\sin(q_2)\text{-} \\ (\sin(q_2)XX_2+\cos(q_2)XY_2)\cos(q_2) \\ \text{-}\frac{1}{2}(\cos(q_2)XY_2\text{-}\sin(q_2)YY_2)\cos(q_2)+ \\ (\sin(q_2)XY_2+\cos(q_2)YY_2)\sin(q_2))\frac{\partial q_1}{\partial t} \tag{4.8}$$

$$C_{2,2} = 0 \tag{4.9}$$

And the gravity matrix:

$G(q) =$

$$\begin{bmatrix} 0 \\ \text{-}g\cos(q_2)\ mx_2 + g\sin(q_2)\ my_2 \end{bmatrix} \tag{4.10}$$

4.2 Program using Newton-Euler Iterative method

To implement the next program we need to use the following coordinated system
In relation with the next frame

$$^{1}P_{C_1} = l_1 \hat{X}_1 \tag{4.11}$$

$$^{2}P_{C_2} = l_2 \hat{X}_2 \tag{4.12}$$

Because of the point assumption, the inertia tensor written at the center of mass for each link is the zero matrix:

$$^{C_1}I_C = 0 \tag{4.13}$$

$$^{C_2}I_C = 0 \tag{4.14}$$

There are no forces acting on the end effector, and so we have

$$f_3 = 0 \tag{4.15}$$

$$n_3 = 0 \tag{4.16}$$

The base of the robot is not rotating, and hence we have

$$\omega_0 = 0 \tag{4.17}$$

$$\dot{\omega}_0 = 0 \tag{4.18}$$

To include gravity forces we will use

$$^{0}\dot{v}_0 = g\hat{Y}_0 \tag{4.19}$$

The rotation between succesives link frames is given by

$$_{i+1}^{\ \ i}R = \begin{bmatrix} c\theta_{i+1} & -s\theta_{i+1} & 0 \\ s\theta_{i+1} & c\theta_{i+1} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.20}$$

$$_{\ \ i}^{i+1}R = \begin{bmatrix} c\theta_{i+1} & s\theta_{i+1} & 0 \\ -s\theta_{i+1} & c\theta_{i+1} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.21}$$

And the transformation is given by

$$ {}^{i-1}_{\ \ i}T = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i \cdot c\alpha_{i-1} & c\theta_i \cdot c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1} \cdot d_i \\ s\theta_i \cdot s\alpha_{i-1} & c\theta_i \cdot s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1} \cdot d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.22} $$

The second program we have built, were built using the Newton-Euler iterative algorithm and are as follow

```
with(linalg):
pc[1]:=vector([l1,0,0]):
pc[2]:=vector([l2,0,0]):
p[1]:=vector([0,0,0]):
p[2]:=vector([l1,0,0]):
n:=diff(v[0],v)=matrix(3,1,[g,0,0]):
alpha[0]:=0;
alpha[1]:=0;
alpha[2]:=0;
 for j from 0 by 1 to 2 do
RT[j]:=matrix(3,3,[cos(theta[j+1]),-
sin(theta[j+1]),0,sin(theta[j+1])*cos(alpha[j]),cos(thet
a[j+1])*cos(alpha[j]),-
sin(alpha[j]),sin(theta[j+1])*sin(alpha[j]),cos(theta[j+
1])*sin(alpha[j]),cos(alpha[j])]):
R[j]:=transpose(RT[j]):
z[j]:=matrix(3,1,[0,0,1]):
od:
for j from 0 by 1 to 1 do
omega[0]:=0:
omega[j+1]:=evalm(R[j]&*omega[j]+evalm(Diff(theta[j+1],t
)*z[j]));
temp2[j]:=convert(evalm(Diff(theta[j+1],t)*z[j]),vector)
:
if (j<>0) then
temp1[j]:=convert(evalm(R[j]&*omega[j]),vector):
else
temp1[j]:=vector([0,0,0]):
fi;
##X[j]=omega´s[j+1] derivative
od:
```

```
for j from 0 by 1 to 1 do
X[j]:=evalm(evalm(R[j]&*map(Diff,omega[j],t)+evalm(cross
prod(evalm(temp1[j]),temp2[j]))+evalm(Diff(Diff(theta[j+
1],t),t)*z[j])));
od:
 for j from 0 by 1 to 1  do
if (j=0) then
Y[j]:=evalm(evalm(R[j])&*matrix(3,1,[0,g,0]));
else
Y[j]:=evalm(evalm(R[j])&*convert(evalm((crossprod(conver
t(map(Diff,omega[j],t),vector),p[j+1]))+crossprod(conver
t(evalm(omega[j]),vector),crossprod(convert(evalm(omega[
j]),vector),p[j+1]))),matrix))+evalm(evalm(R[j])&*Y[j-
1]);
fi;
od:
 for j from 0 by 1 to 1  do
temp3[j]:=convert(evalm(X[j]),vector);
temp4[j]:=convert(evalm(omega[j+1]),vector);
YY[j+1]:=evalm(crossprod(temp3[j],pc[j+1])+crossprod(tem
p4[j],crossprod(temp4[j],pc[j+1]))+Y[j]);
F[j+1]:=evalm(evalm(m[j+1]*evalm(YY[j+1])));
N[j]:=array(1..3,[0,0,0]);
od:
 i:=2:
while(i<>0) do
f[3]:=matrix(3,1,[0,0,0]):
n[3]:=matrix(3,1,[0,0,0]);
f[i]:=evalm(evalm(evalm(RT[i])&*evalm(f[i+1]))+evalm(F[i
])):
i:=i-1:
od:
i:=2:
p[1]:=vector([0,0,0]):
p[2]:=vector([l1,0,0]):
while (i<>0) do
if (i=2) then
n[i]:=evalm(convert(evalm(RT[i]&*n[3]),vector)+crossprod
(pc[i],convert(evalm(F[i]),vector)));
else
n[i]:=convert(evalm(RT[i]&*n[i+1]),vector)+crossprod(pc[
i],convert(evalm(F[i]),vector))+crossprod(p[i+1],convert
(evalm(evalm(RT[i])&*evalm(f[i+1])),vector));
fi;
tau[i]:=simplify(evalm(transpose(convert(evalm(n[i]),mat
rix))&*z[i]),trig);
```

```
i:=i-1;
od:
M:=matrix(2,2,[coeff(tau[1][1,1],Diff(Diff(theta[1],t),t
)),coeff(tau[1][1,1],Diff(Diff(theta[2],t),t)),coeff(tau
[2][1,1],Diff(Diff(theta[1],t),t)),coeff(tau[2][1,1],Dif
f(Diff(theta[2],t),t))])&*matrix(2,1,[Diff(Diff(theta[1]
,t),t),Diff(Diff(theta[2],t),t)]);
tmp1:=coeff(tau[2][1,1],Diff(Diff(theta[1],t),t)):
tmp2:=simplify(tau[2][1,1]-
tmp1*Diff(Diff(theta[1],t),t)):
tmp3:=coeff(tau[1][1,1],Diff(Diff(theta[1],t),t)):
tmp4:=coeff(tau[1][1,1],Diff(Diff(theta[2],t),t)):
tmp5:=simplify(tau[1][1,1]-
tmp3*Diff(Diff(theta[1],t),t)-
tmp4*Diff(Diff(theta[2],t),t)):
C:=matrix(2,1,[coeff(tmp5,Diff(theta[2],t),2)*Diff(theta
[2],t)^2+coeff(tmp5,Diff(theta[1],t))*Diff(theta[1],t),c
oeff(tmp2,Diff(theta[1],t),2)*Diff(theta[1],t)^2]);
G:=matrix(2,1,[simplify(coeff(tau[1][1,1],g),trig),simpl
ify(coeff(tau[2][1,1],g),trig)])*g;
```

When the Newton-Euler equations are evaluated symbolically for any manipulator, they yield a dynamic equation, which can be written in the form

$$\tau = M(\Theta)\ddot{\Theta} + V(\Theta,\dot{\Theta}) + G(\Theta) \tag{4.23}$$

Where $M(\Theta)$ is the $n\times n$ mass matrix of the manipulator composed of all those terms which multiply $\ddot{\Theta}$, and it is a function of Θ ,$V(\Theta,\dot{\Theta})$ is $n\times 1$ vector and contains all those terms which have any dependence on joint velocity, terms which depends on the square of joint are caused by centrifugal force and those which contains product of two different joint velocities are caused by coriolis force, and $G(\Theta)$ is an $n\times 1$ vector and contains all those terms in which appear gravitational constant.

Where,

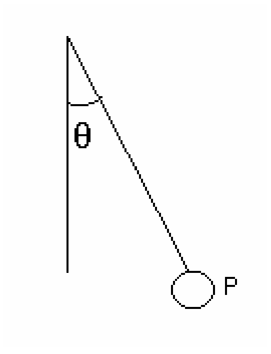
$$\mathrm{M}(\Theta)=\begin{bmatrix} (\mathrm{m}_1+\mathrm{m}_2)\mathrm{l}_1^2+\mathrm{m}_2\mathrm{l}_2^2+2\mathrm{c}_2\mathrm{m}_2\mathrm{l}_1\mathrm{l}_2 & \mathrm{m}_2\mathrm{l}_2^2+\mathrm{c}_2\mathrm{m}_2\mathrm{l}_1\mathrm{l}_2 \\ \mathrm{m}_2\mathrm{l}_2^2+\mathrm{c}_2\mathrm{m}_2\mathrm{l}_1\mathrm{l}_2 & \mathrm{m}_2\mathrm{l}_2^2 \end{bmatrix}.\begin{bmatrix}\ddot{\Theta}_1 \\ \ddot{\Theta}_2\end{bmatrix} \tag{4.24}$$

$$C\left(\Theta,\dot{\Theta}\right)\cdot\dot{\Theta}=\begin{bmatrix}-m_2l_1l_2s_2\dot{\theta}_2^2-2m_2l_1l_2s_2\dot{\theta}_1\dot{\theta}_2\\ m_2l_1l_2s_2\dot{\theta}_1^2\end{bmatrix} \tag{4.25}$$

$$G(\Theta)=\begin{bmatrix}m_2l_2gc_{12}+(m_1+m_2)l_1gc_1\\ m_2l_2gc_{12}\end{bmatrix} \tag{4.26}$$

5. Parameters estimation.

5.1 Pendulum model.



Find the pendulum's mass whether $q,\dot{q},\ddot{q},g$ and pendulum's length are known. First of all, we have to find the equation, which represent this model, to obtain this equation we use the energy's method, which says:

$$\frac{dE}{dt}=\tau \tag{5.1}$$

In our problem the energy is divided in kinetic and potential energy, to obtain this energy we must be careful with the disposition of our pendulum, and then we have:

$$E_K=\frac{1}{2}\cdot I\cdot\dot{q}^2 \tag{5.2}$$

$$E_p=m\cdot g\cdot l\cdot sen(q) \tag{5.3}$$

Now we have to derivate and the energy is obtained:

$$\frac{dE_K}{dt}=I\cdot\dot{q}\cdot\ddot{q} \tag{5.4}$$

$$\frac{dE_p}{dt} = m \cdot g \cdot l \cdot \dot{q} \cdot \cos(q) \tag{5.5}$$

Finally is obtained the model of the pendulum:

$$I \cdot \ddot{q} + m \cdot g \cdot l \cdot \cos(q) = \tau \tag{5.6}$$

Now we have to estimate the pendulum's mass, to estimate it the least square is used, then we have to apply the next formula, which say:

$$\hat{\theta} = (\varphi_N^T \cdot \varphi_N)^{-1} \cdot \varphi_N^T \cdot y_N \tag{5.7}$$

Where φ is the regressor, which in our case is:

$$\varphi_N = l^2 \cdot \ddot{q} + g \cdot l \cdot \cos(q) \tag{5.8}$$

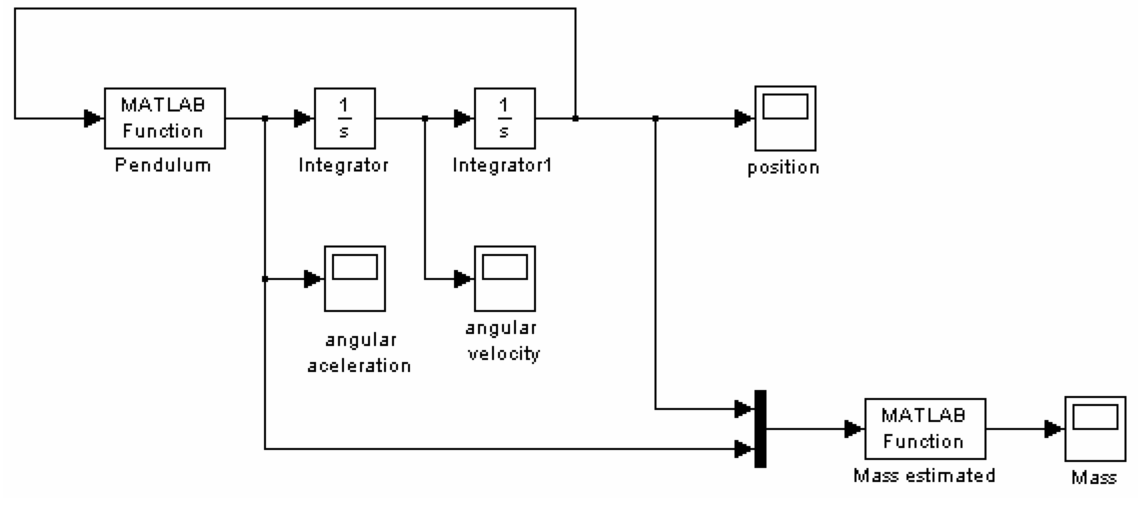
$$y_N = \tau \tag{5.9}$$

Then is obtained

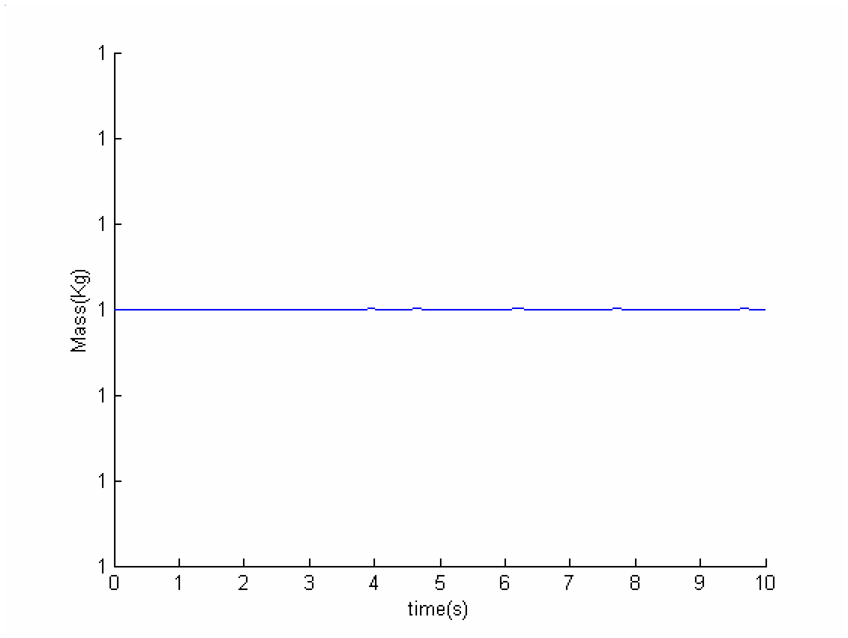
$$\theta = \frac{1}{l^2 \cdot \ddot{q} + l \cdot g \cdot \cos(q)} \cdot \tau \tag{5.10}$$

Now with this equation a simulation in matlab is made then the mass is estimated. Where the input of matlab function are:

$m = 0.5$ Kg $l = 0.2$ m $g = 9{,}82\ m/s^2$ $\tau = 0.5$ N.m



Applying those formulas the estimated mass is 1Kg,how is shown in the next plot:



Conclusion

After various simulations with differents values for the torque we observe that the pendulum's mass is independent of it and only depend on the values of the pendulum length and position, velocity and acceleration.

5.1.1 Pendulum model adding filters

Consider now that we are not able to measure the pendulum acceleration, $\ddot{q}$ is unknown, in order to resolve the problem, a low pass filter is needed it has the next expression:

$$\frac{w}{w+s} \tag{5.11}$$

Applying this filter to the model we obtain :

$$\frac{w}{w+s}\cdot I\cdot\ddot{q}+\frac{w}{w+s}\cdot m\cdot g\cdot l\cdot\cos(q)=\frac{w}{w+s}\cdot\tau \tag{5.12}$$

$$\frac{w}{w+s}\cdot\ddot{q}=\frac{w\cdot s}{w+s}\cdot\dot{q}=w\cdot s-\frac{w^2}{w+s} \tag{5.13}$$

$$m\left(l^2\left(w\cdot\dot{q}-\frac{w^2}{w+s}\cdot\dot{q}\right)+g\cdot l\cdot\frac{w}{w+s}\cdot\cos(q)\right)=\frac{w}{w+s}\cdot\tau \tag{5.14}$$

As in the previous case we will try to observe which of the following method obtain better result in the estimation. We will start inplementing the least-squares method, with the following results.

Now we will use to estimate the pendulum's mass, the algorithm describes in which says:

$$\dot{\hat{m}}=\alpha\cdot P\cdot\Psi\left(\tau_f-\Psi^T\cdot\hat{m}\right) \tag{5.15}$$

$$\dot{P}=-\alpha\cdot P\cdot\Psi\cdot\Psi^T\cdot P \tag{5.16}$$

Where φ is the regressor, which in our case is:

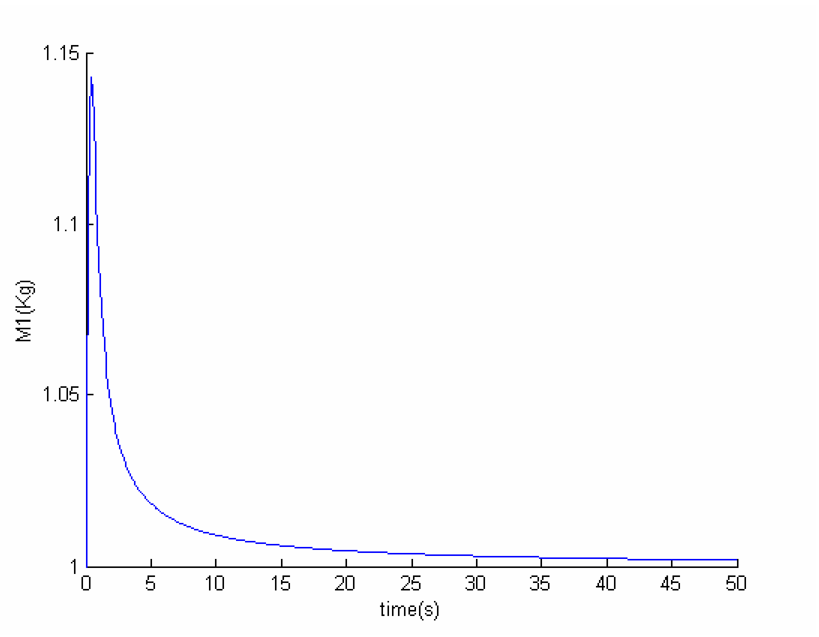
$$\Psi=\mathrm{l}^2\cdot\left(w\cdot\dot{\mathrm{q}}-\frac{\mathrm{w}^2}{\mathrm{s+w}}\cdot\dot{\mathrm{q}}\right)+\mathrm{g}\cdot\mathrm{l}\cdot\frac{\mathrm{w}}{\mathrm{s+w}}\cdot\cos(\mathrm{q}) \tag{5.17}$$

$$\tau_f=\frac{\mathrm{w}}{\mathrm{s+w}}\cdot\tau \tag{5.18}$$

Now with this equation a simulation in matlab is made then the mass is estimated, Where the input of matlab function are:

m = 1 Kg l = 0.2 m g = 9,82 m/s^2 τ =0.5 N.m w=10 alpha=15

Applying those formulas the estimated mass is 0.5Kg,how is shown in the next plot:

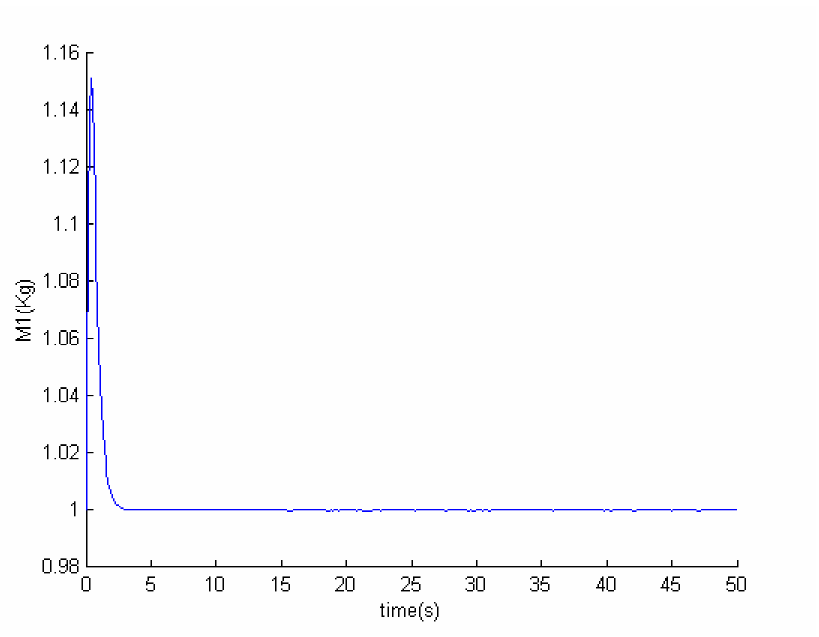


The last method to implement will be the gradient's method, which say[2]:

$$\dot{\hat{m}} = \alpha \cdot \Psi \cdot \left(\tau_f - \Psi^T \cdot \hat{m} \right) \tag{5.19}$$

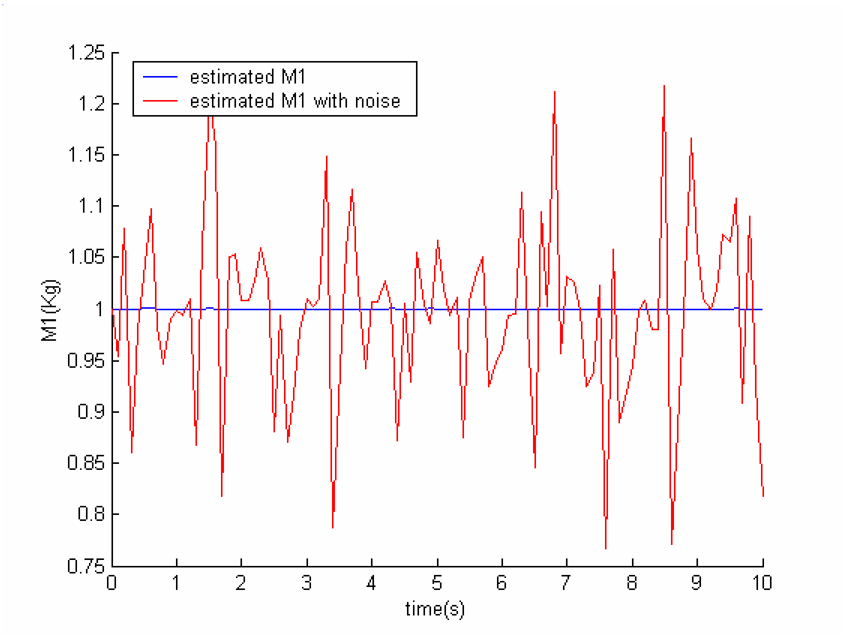
m = 1 Kg l = 0.2 m g = 9,82 m/s^2 τ =0.5 N.m w=20 alpha=20

We obtain the next plot:

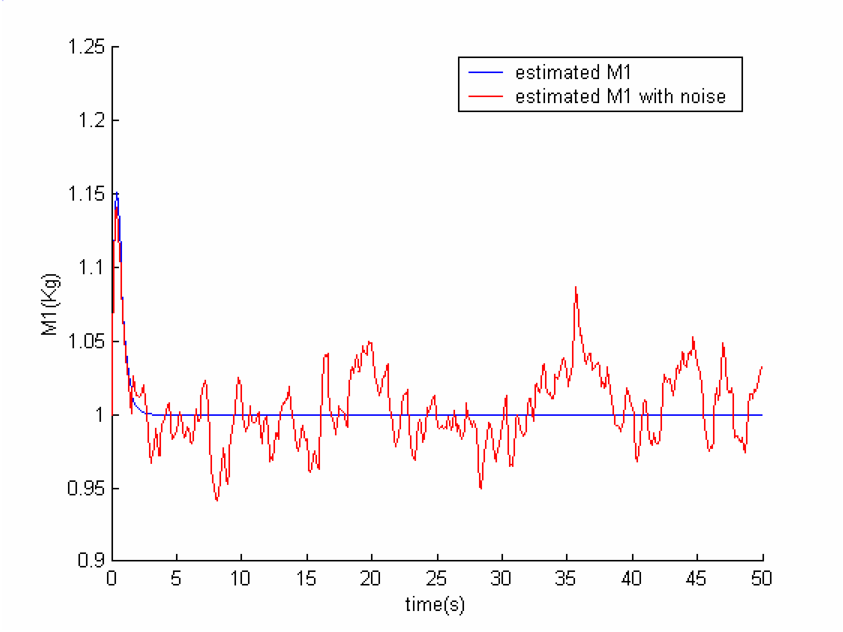


5.1.2 Pendulum model adding noises

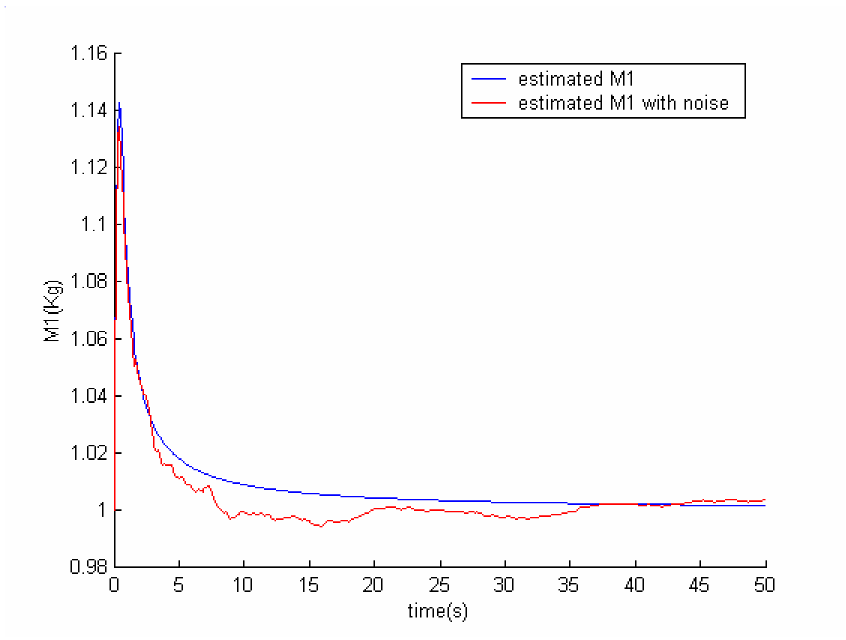
As in the previous chapter we will discuss the different response of the estimation algorithms in order to choose which will be the best method therefore ,the method that have better response when a noise is introduced to the system.
First method, Least-squares



Second method ,unnormalized least-squares method

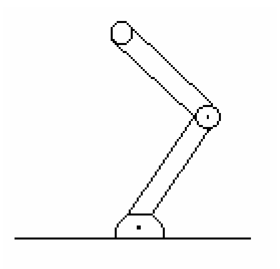


As we can see in those graphs the response of least squares methods is worse than gradient's method due to its elevate time to reach the estimated mass.
After others simulation in which I have included white noise the response of the system was better with the least squares method.
Third method, gradient



5.2 Two-link manipulator

5.2.1 Two-link manipulator scheme



5.2.2 Two-link manipulator adding filters

Following the method in ,we obtain the following equation

$$\tau = M(q)\ddot{q} + V(q,\dot{q}) + G(q) \tag{5.20}$$

where

$$\mathrm{M}(q) = \begin{bmatrix} (\mathrm{m}_1+\mathrm{m}_2)\mathrm{l}_1^2 & \mathrm{m}_2\mathrm{l}_2\mathrm{l}_1\cos(q_1-q_2) \\ \mathrm{m}_2\mathrm{l}_2\mathrm{l}_1\cos(q_1-q_2) & \mathrm{m}_2\mathrm{l}_2^2 \end{bmatrix} \cdot \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} \tag{5.21}$$

$$C(\mathrm{q},\dot{\mathrm{q}}) \cdot \dot{q} = \mathrm{m}_2\mathrm{l}_1\mathrm{l}_2\sin(q_2-q_1) \cdot \begin{bmatrix} -\dot{q}_2^2 + \dot{q}_1 \cdot \dot{q}_2 \\ \dot{q}_1^2 - \dot{q}_1 \cdot \dot{q}_2 \end{bmatrix} \tag{5.22}$$

$$\mathrm{G}(q) = \begin{bmatrix} -(\mathrm{m}_1+\mathrm{m}_2)\mathrm{l}_1\mathrm{g}\sin(q_1) \\ -\mathrm{m}_2\mathrm{l}_2\mathrm{g}\sin(q_2) \end{bmatrix} \tag{5.23}$$

As in the previous experiments we will try to made our experiments without the unknowledge of the aceleration,in our case $\ddot{\mathrm{q}}$,therefore we will use the same low pass filter as before:

$$\frac{w}{w+s} \tag{5.24}$$

In that case the result of the regressor is not so obvious because appear new terms like cosines and sinus not so easy to integer when we apply the low pass filter, we will need to use the integration by parts that says:

$$\frac{w}{w+s}\ddot{q}_1\cos(q_1-q_2) = \frac{ws}{w+s}\left(\frac{1}{s}\ddot{q}_1\cos(q_1-q_2)\right) \tag{5.25}$$

$$\int v \cdot du = u \cdot v - \int u \cdot dv \tag{5.26}$$

$$du = \ddot{q}_1 \qquad u = \dot{q}_1 \tag{5.27}$$

$$v = \cos(q_1-q_2) \qquad dv = -\sin(q_1-q_2)(\dot{q}_1-\dot{q}_2) \tag{5.28}$$

Now we have to estimate the two-link´s masses, to estimate it the least square [1] is used, also gradient method [2] and Goodwin method [3] known the Ψ regressor, which in our case is:

$$\Psi = \begin{bmatrix} l_1^2\dot{q}_1 w - l_1^2\dot{q}_1\frac{w^2}{s+w} - gl_1\frac{w}{s+w}\sin(q_1) \\ 0 \end{bmatrix} \tag{5.29}$$

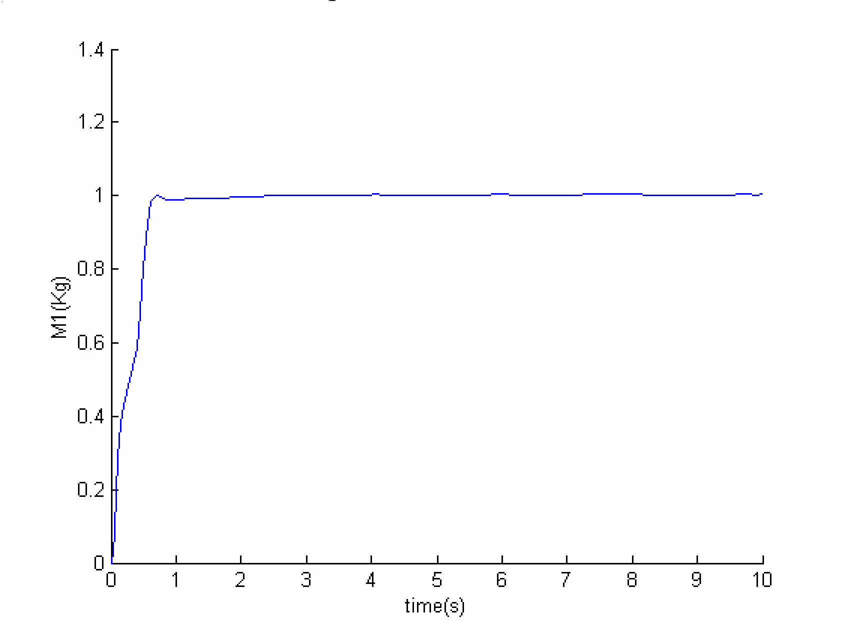
$$\left.\begin{matrix} l_1^2\dot{q}_1 w - l_1^2\dot{q}_1\dfrac{w^2}{s+w} - gl_1\dfrac{w}{s+w}\sin(q_1) + \dfrac{-w^2}{s+w}l_1l_2\dot{q}_2\cos(q_1-q_2) + wl_1l_2\dot{q}_2\cos(q_1-q_2) + \dfrac{w}{s+w}l_1l_2\dot{q}_2\sin(q_1-q_2) \\ l_2^2\dot{q}_2 w - l_2^2\dot{q}_2\dfrac{w^2}{s+w} - gl_2\dfrac{w}{s+w}\sin(q_2) + \dfrac{-w^2}{s+w}l_1l_2\dot{q}_1\cos(q_1-q_2) + wl_1l_2\dot{q}_1\cos(q_1-q_2) + \dfrac{w}{s+w}l_1l_2\dot{q}_1\sin(q_1-q_2) \end{matrix}\right] \cdot \begin{bmatrix} m_1 \\ m_2 \end{bmatrix}$$

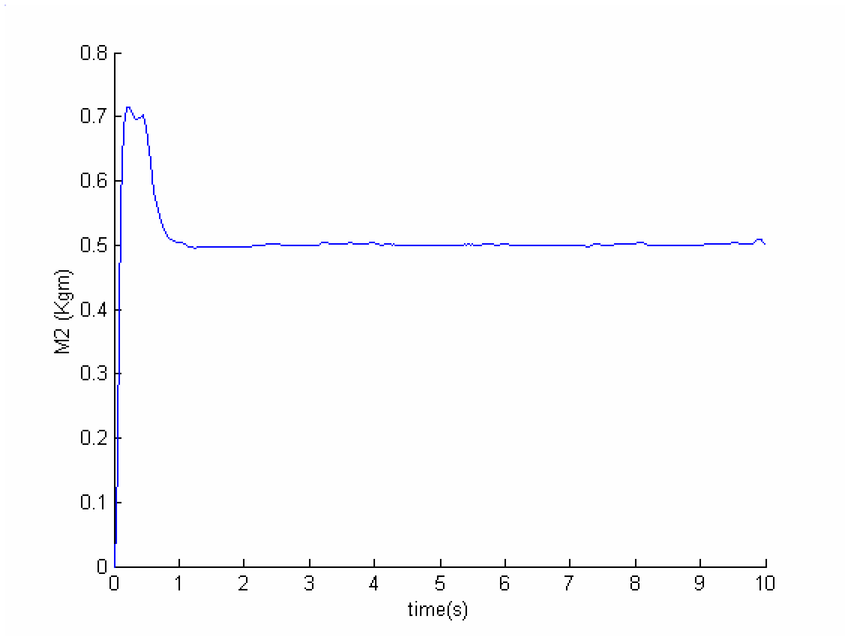
$$\tau_f = \frac{\mathrm{w}}{\mathrm{s+w}} \cdot \tau$$

Now with this equation a simulation in matlab is made then the mass is estimated. Where the input of matlab function are:

m1 = 1 Kg l1 = 0.2 m g = 9,82 m/s^2 τ2=step w=10 alpha=10
m2= 0.5 Kg   l2 = 0.2 m   τ1 =step

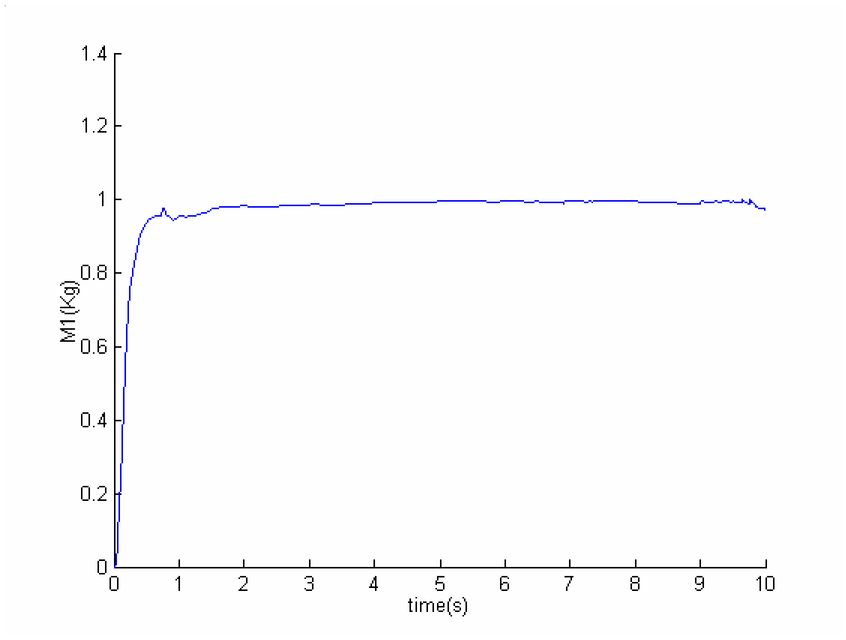
First with the unnormalized least-squares we obtain:

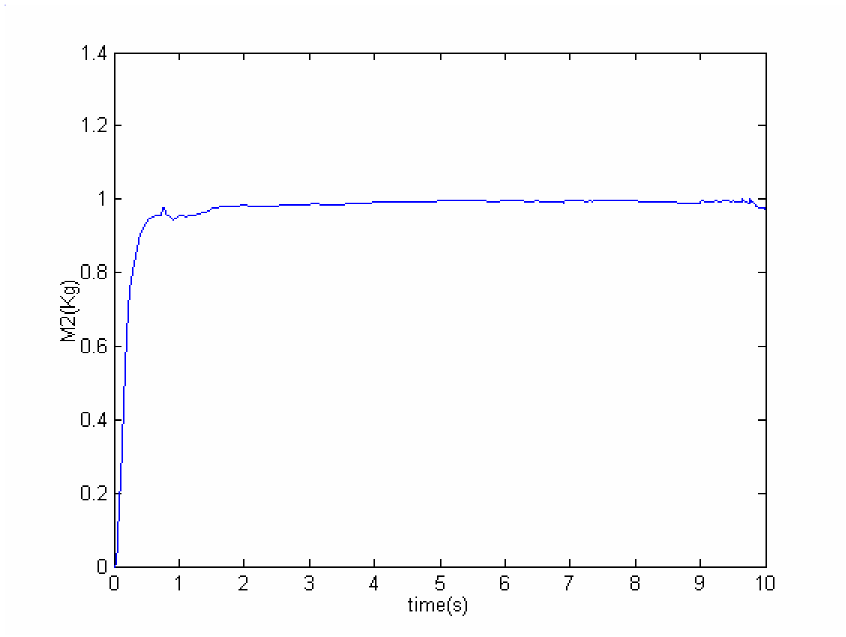




Second Gradient method

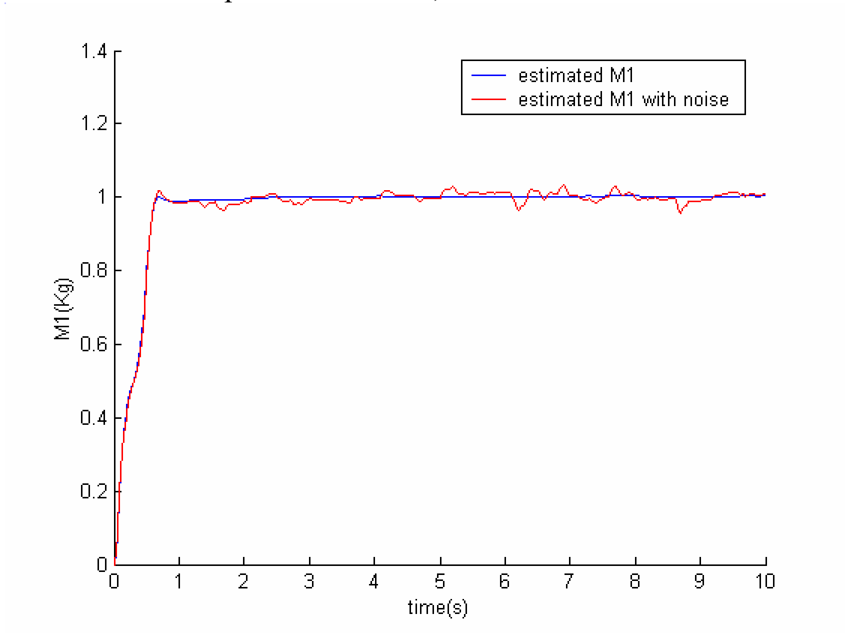
m1=1 Kg m2=1 Kg w=5 alpha=5



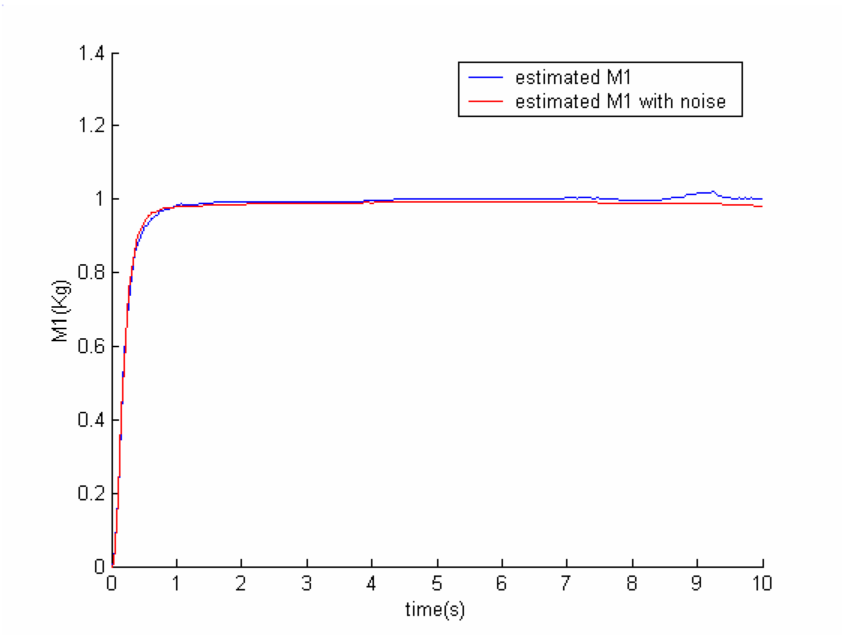


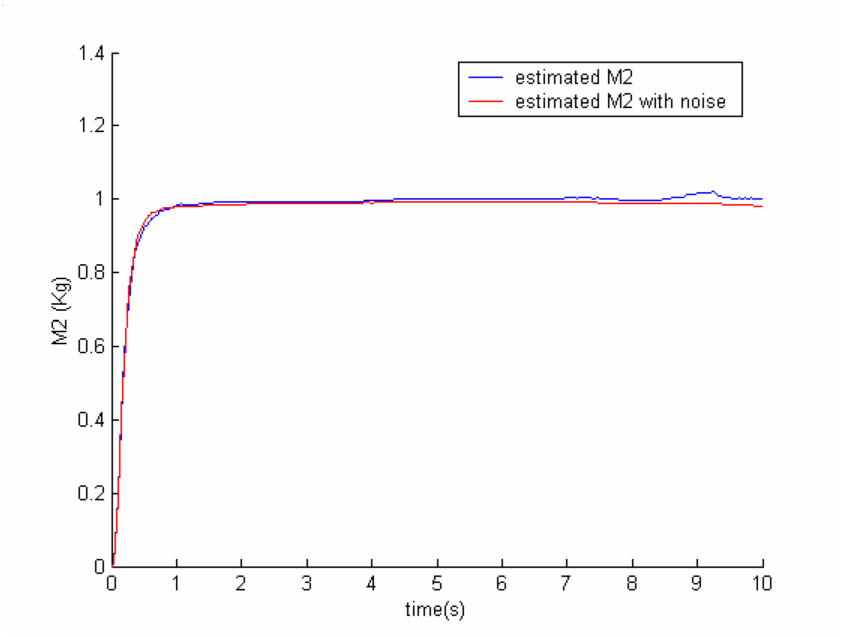
5.2.3 Twolink manipulator adding noises.

With unnormalized least-squares we obtain,



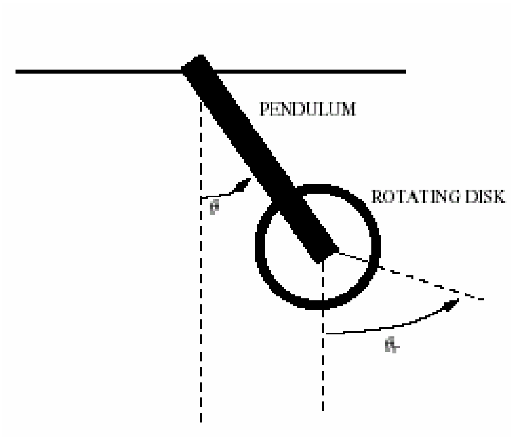
After others simulation in which I have included white noise the response of the system was better with Goodwing method.


estimated M1
estimated M1 with noise
M1(Kg)
time(s)
0
0.2
0.4
0.6
0.8
1
1.2
1.4
0
1
2
3
4
5
6
7
8
9
10


estimated M2
estimated M2 with noise
M2 (Kg)
time(s)
0
0.2
0.4
0.6
0.8
1
1.2
1.4
0
1
2
3
4
5
6
7
8
9
10

5.3Inertia wheel model

5.3.1Inertia wheel model scheme



In this section we will work with first we a model simulated in matlab and after that we will try to obtain the same results using a real plant described in [5]
First of all defines the physical equation of the model in the next form:

$$J\ddot{\theta} + mgl\sin(\theta) = -ku$$
(5.30)

$$J_r\ddot{\theta}_r = ku \qquad (5.40)$$

Where u is the control signal, and k is a proportionality constant scaling the input to the duty cycle range(0-500) of the PWM amplifier and

$$m = m_p + m_r \qquad (5.41)$$

$$ml = m_p l_p + m_r l_r \qquad (5.42)$$

$$J = J_p + m_p l_p^2 + m_r l_r^2 \qquad (5.43)$$

As in the previous works we will try to implement differents algorithms types to obtain the estimated parameters of the system. The regressor to estimate three parameters is as follow:

$$\psi = \begin{bmatrix} \ddot{\theta} & g\sin(\theta) & 0 \\ 0 & 0 & \ddot{\theta}_r \end{bmatrix} \qquad (5.44)$$

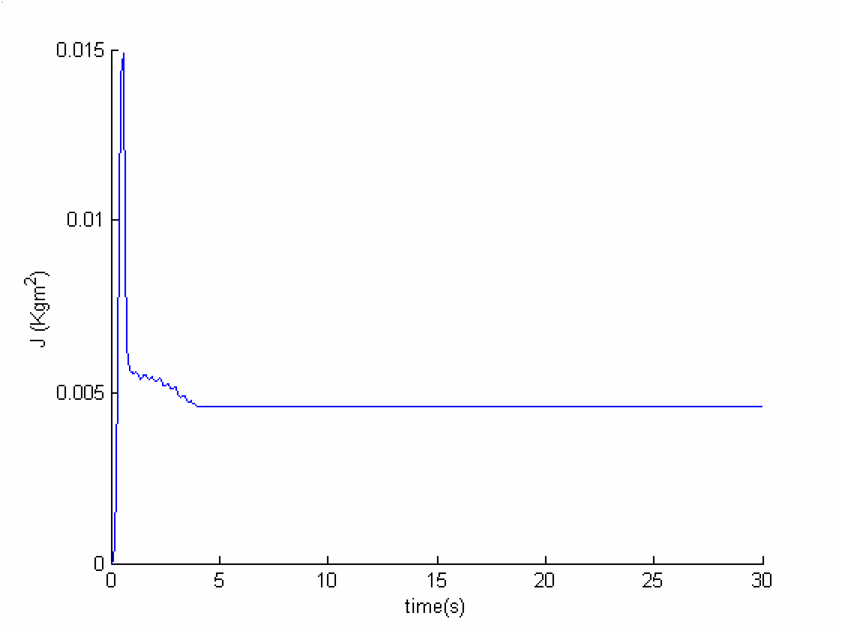
But let us consider that the acceleration and also the velocities are unknown, then we will need to implement two filters instead of one obtaining:

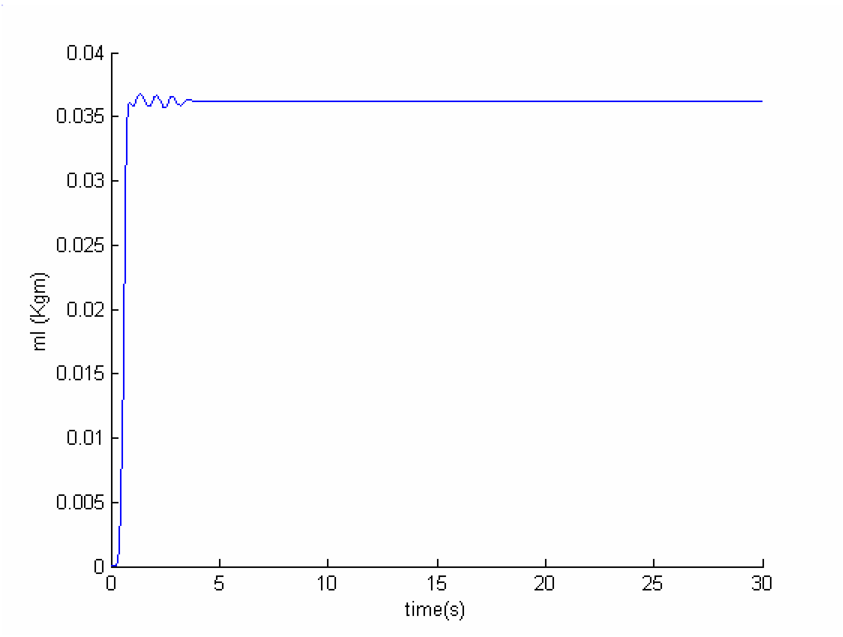
$$\ddot{\theta} = \theta\left(\omega s - \frac{\omega^2}{\omega + s}\right)^2 = \omega^2 - \frac{2\omega^3 s + \omega^4}{s^2 + 2\omega s + w^2} \qquad (5.45)$$

first we will implement the gradient algorithm that says:

$$\dot{\hat{m}} = \alpha \cdot \Psi \cdot \left(\tau_f - \Psi^T \cdot \hat{m}\right) \qquad (5.46)$$

obtained the following graphs and schemes:



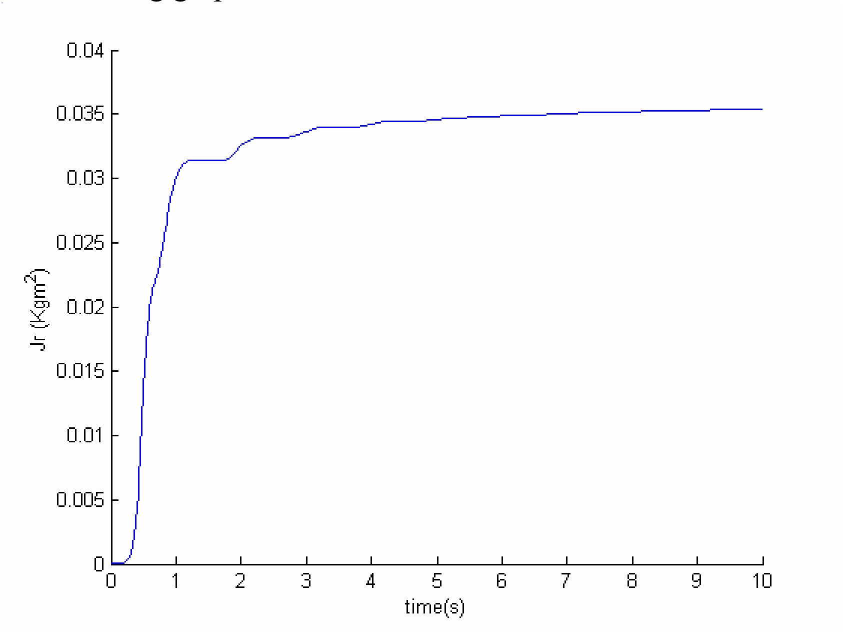


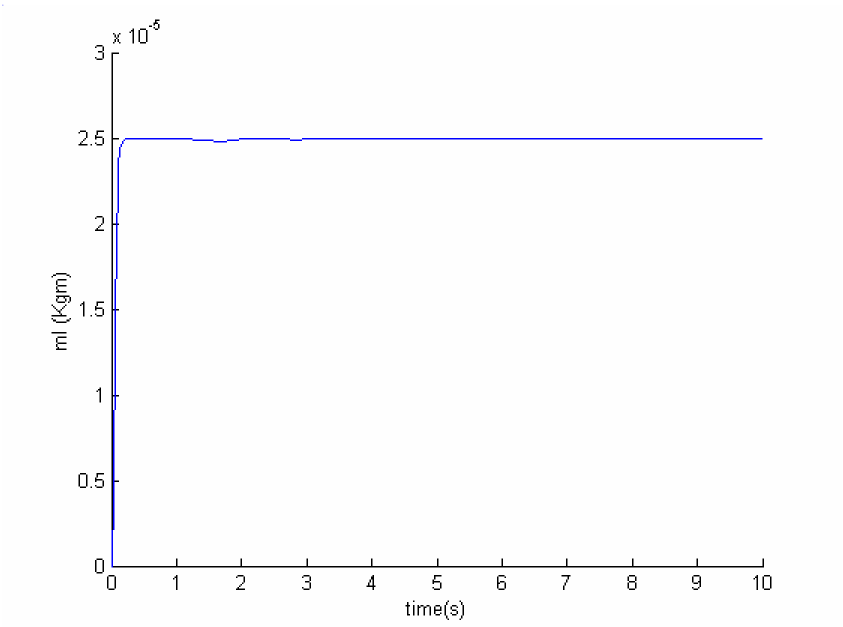
Trying to implement the gradient method we see that the system is unstable therefore is not possible to implement this algorithm to estimate three parameters.
Now with Goodwin algorithm

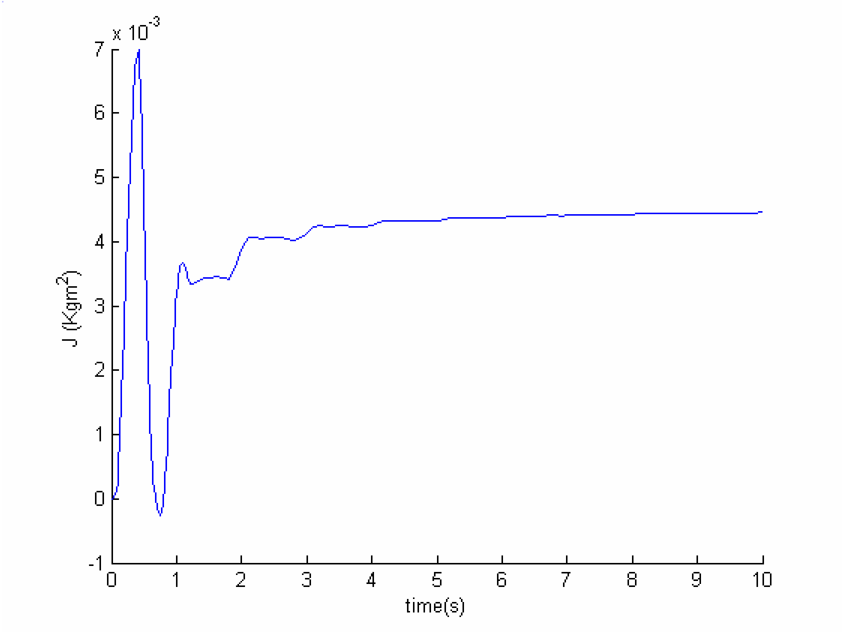
$$\dot{\hat{m}} = \alpha \cdot P \cdot \Psi\left(\tau_f - \Psi^T \cdot \hat{m}\right) \tag{5.47}$$

$$\dot{P} = -\alpha \cdot P \cdot \Psi \cdot \Psi^T \cdot P \tag{5.48}$$

obtained the following graphs and schemes:

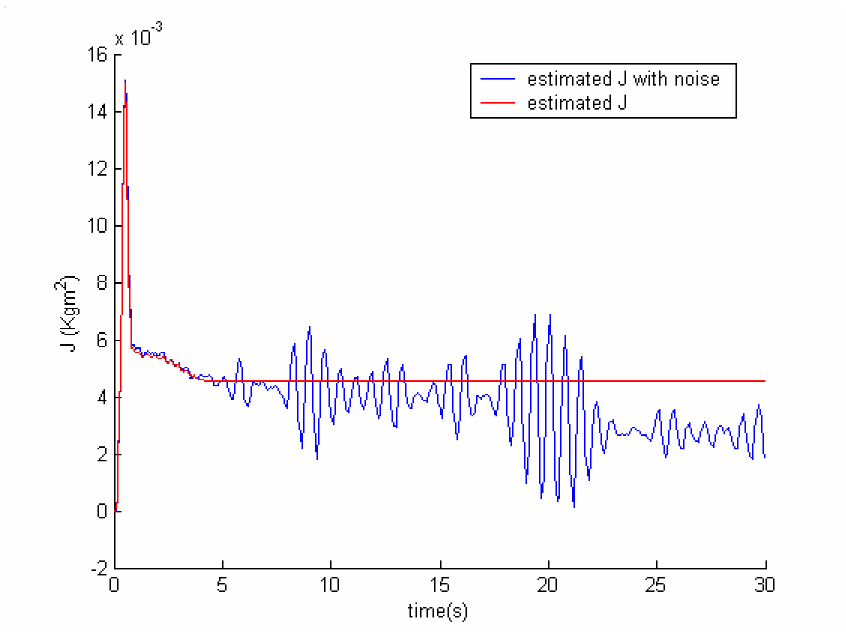


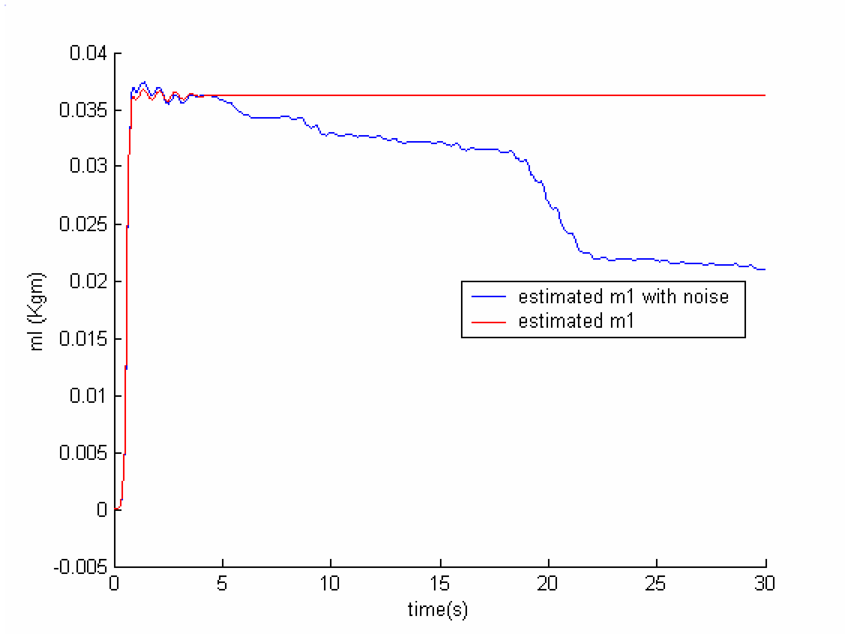




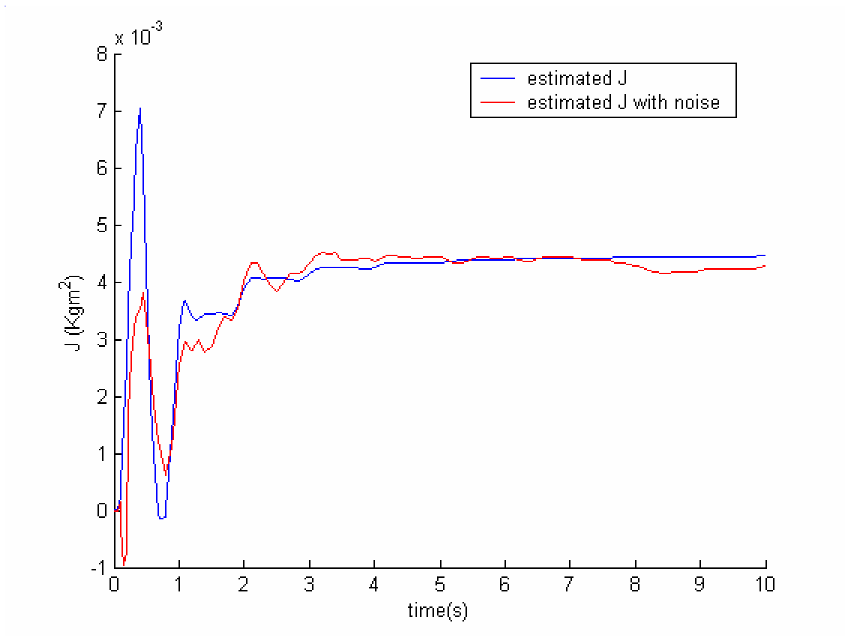
5.3.2 Inertia wheel with noises.

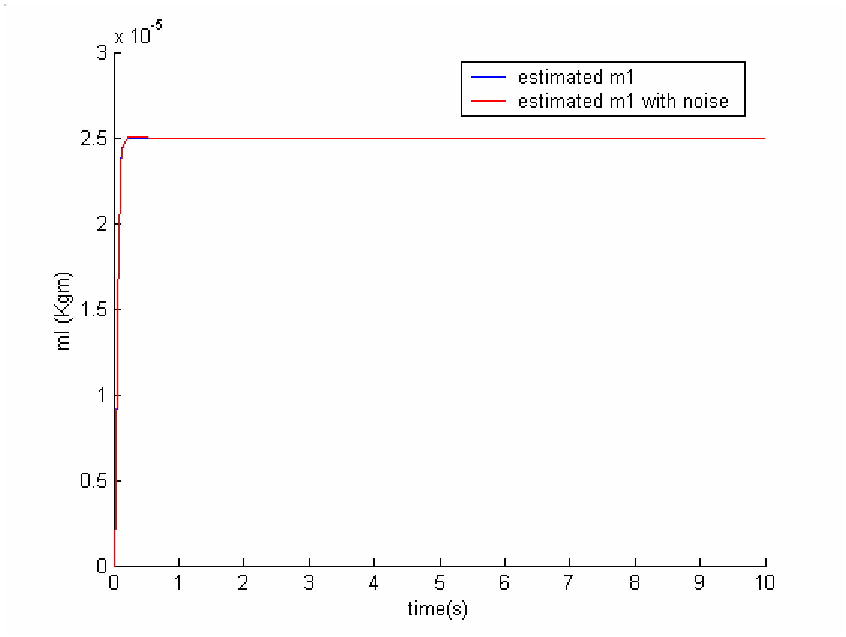
As in the previous chapter is not possible to estimated three parameters with the gradient method ,but is also unuseful in case that we only want to estimate two parameters because this method is very affected for this noises as we can see in the following graphs where the noises´ amplitude is 0.00001,

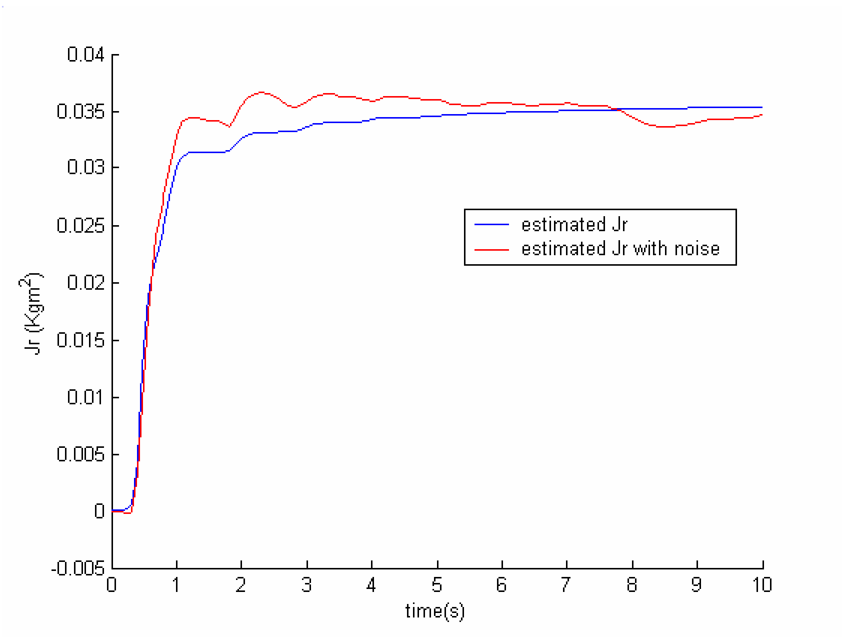




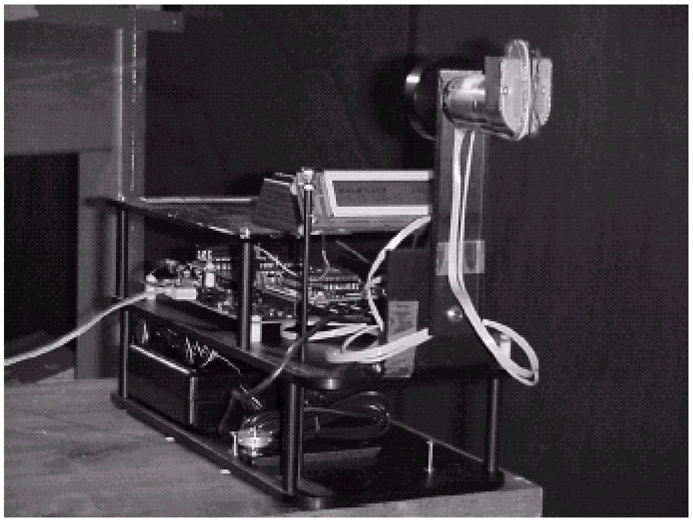
Using again the unnormalized least-squares we obtain


x 10^-3
J (Kgm^2)
time(s)
estimated J
estimated J with noise

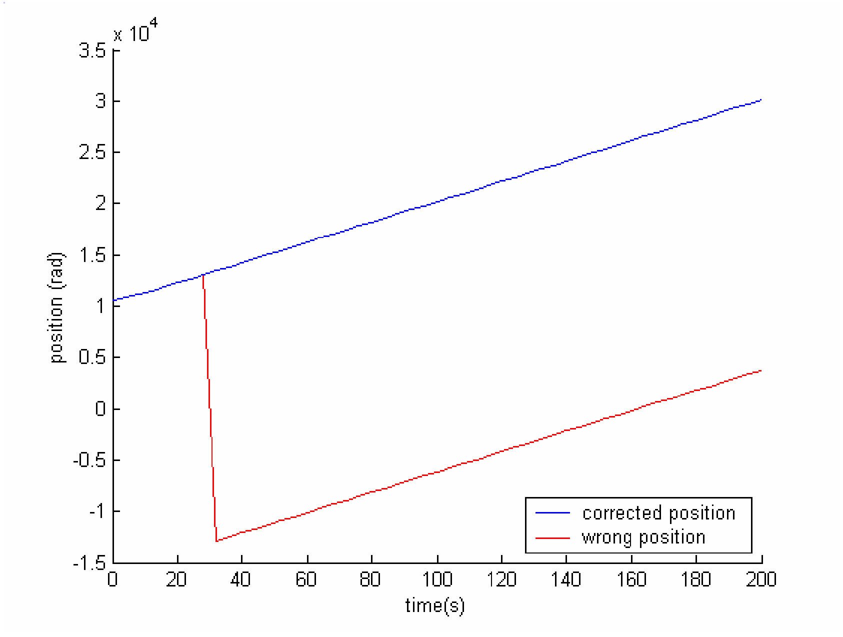

x 10^-5
ml (Kgm)
time(s)
estimated m1
estimated m1 with noise



5.4 Inertia wheel plant.



First of all we found a problem with the encoder, this problem that appear in the position measurements of the wheel was due to the appearance of a gap between to consecutive position of the wheel,we can se the problem and the solution in the following graph,



6. Concluding Remarks.

6.1 Conclusion.

First, the use of this maple program, that we have use will resolve a difficulties in the calculation due to hand made errors and also using it we reduce the time to built our models. About what program is better to obtain the models, we must say that the Lagrange formulation allow to study the characteristics and properties of the dynamic model of robots, but whether we want a model to be used on real time implementation our model must be Newton-Euler.

Second, in the case of the estimation procedures use in this paper, as were shown before, the first method use, least-squares estimation, was useful in the case we don't have noises due to the big influence of the noises in the least-squares method. The second method used, unnormalized least-squares, present worse response than the gradient method when a noise is introduced, excepts in the case of the inertia-wheel where the unnormalized least-squares gave us better results.

6.2 Future works.

Due to this estimation constitute the base of a wide kind of robots model once that we know the inertial parameters we will be able to control velocities, position ,torques,etc.,we can deduce the influence of inertial parameters on the positional accuracy of robots(19) that task is very important for example in the case that we use the robot for teleoperation.

Appendix A. Commands used in the programs

A.1 Introduction

First thing to do is open the maple program, depending on the platform that you are using, you need type maple in case that you are using Unix, or click in the maple icon if you are using windows machine.

How obtain help?

First of all how obtain help from the program, the only thing to do obtain this help is to type ,

```
>?<command>
>?
```

For a short **Maple Introduction**, first click anywhere on the Worksheet window, then type

```
>?intro
```

typing tutorial(1); at the ">" Maple prompt;

following the directions by typing Return when asked and SPACE when asked for "*More?* or type:

x to quit and type x again to quit and see the menu, selecting the last item to final quit;

after quitting the Tutorial mode before all the tutorials are finished and are back in Maple mode you may type tutorial(<n>); for <n> = 2 or 3 or .. or 16 for one of the other tutorial sections that was listed in the menu upon quitting.

Also maple contains on-line help facility

A.2 Commands used in Newton-Euler and Lagrange equation method

Cross product

linalg[crossprod] - vector cross product

Calling Sequence

crossprod(u, v)

Parameters

`u, v` - lists or vectors, each with three elements

Description

- This function computes the vector cross product of u and v, defined to be vector([u[2]*v[3]-u[3]*v[2], u[3]*v[1]-u[1]*v[3], u[1]*v[2]-u[2]*v[1]]).
- The command with(linalg,crossprod) allows the use of the abbreviated form of this command.

Matrix

linalg[matrix] - create a matrix

Calling Sequence

matrix(L)
matrix(m, n)
matrix(m, n, L)
matrix(m, n, f)
matrix(m, n, lv)

Parameters

`L`	-	list of lists or vectors of elements
`m,n`	-	positive integers (row and column dimensions)
`f`	-	a function used to create the matrix elements
`lv`	-	a list or vector of elements

Description

- The matrix function is part of the linalg package. It provides a simplified syntax for creating matrices. A general description of matrices in Maple is available under the heading matrix.
- The call matrix(m,n,L) creates an m by n matrix where the first row of the matrix is defined by the list/vector L[1], the second row by L[2], and so forth. The call matrix(L) is equivalent to matrix(m,n,L) where m = nops(L) and n = max(seq(nops(L[i]),i=1..m)) .
- The call matrix(m,n) creates an m by n matrix with unspecified elements.
- The call matrix(m,n,f) creates an m by n matrix whose elements are the result of the function f (possibly a constant) acting on the row and column index of the matrix. Thus, matrix(m,n,f) is equivalent to matrix([[f(1,1), ..., f(1,n)], ..., [f(m,1), ..., f(m,n)]]) .
- The call matrix(m,n,lv) creates an m by n matrix whose elements are read off from lv row by row, where lv is a list or vector of elements of type algebraic.
- Since matrices are represented as two-dimensional arrays, see array for further details about how to work with matrices.

- The command with(linalg,matrix) allows the use of the abbreviated form of this command.

Convert

Convert - convert an expression to a different form

Calling Sequence

convert(expr, form, arg3, ...)

Parameters

`expr`	-	any expression
`form`	-	a name
`arg3, ...`	-	(optional) other arguments

Description

- The convert function is used to convert an expression from one form to another. Some of the conversions are data-type conversions, for example convert([x,y], set) Others are form conversions, for example convert(x^3-3*x^2+7*x+9,horner,x) yields (((x^3)*x+7)*x)+9.

- A user can make his own conversions known to the convert function by defining a Maple procedure in the following way. If the procedure `convert/f` is defined, then the function call convert(a,f,x,y,...) will invoke `convert/f`(a,x,y,...); Note that the procedure may be indexed, for example **convert([1,2,3],** Vector[row]);

Map

map - apply a procedure to each operand of an expression

map2 - apply a procedure with a specified first argument to each operand of an expression

Calling Sequence

map(fcn, expr, arg2, ..., argn)
map2(fcn, arg1, expr, arg3, ..., argn)

Parameters

`fcn`	-	a procedure or a name
`expr`	-	any expression
`argi`	-	(optional) further arguments to **fcn**

Description

- The **map(fcn, expr)** function applies **fcn** to the operands of **expr**.
- The **i**th operand of **expr** is replaced by the result of applying **fcn** to the **i**th operand. This is done for all the operands of **expr**.
- For a table or array, **fcn** is applied to each element of the table or array.

- For an rtable, **fcn** is applied to each element of the rtable and a new rtable of the mapped result is returned.
- If **fcn** takes more than one argument, they are to be specified as additional arguments, **arg2**, **arg3**, ..., **argn**, which are simply passed through as the second, third, ..., **n**th arguments to **fcn**.
- The **map2** function is similar to map, except that for each operand of **expr**, **arg1** is passed as the first argument to **fcn**, the operand of **expr** is passed as the second argument, and **arg3**, ..., **argn** are passed as the third, ..., **n**th arguments.
- Since strings are atomic expressions in Maple, you cannot map a procedure over a string by using **map**. However, the StringTools package provides a **Map** export that delivers this functionality.
- **Note:** **map** and **map2** do not work well with functions that have special evaluation rules. For example, **map** may not return expected results if **fnc** uses parameters that are of type uneval. A workaround that is sometimes useful is to ``wrap" the procedure with special evaluation rules with one that has normal evaluation rules. (Nothing is lost, since **map** itself has normal evaluation rules, so the arguments to it will be evaluated anyway.)

Coefficient

coeff - extract a coefficient of a polynomial

Calling Sequence

coeff(p,x)
coeff(p,x,n)
coeff(p,x^n)

Parameters

p	-	a polynomial in x
x	-	the variable (an expression)
n	-	(optional) an integer

Description

- The coeff function extracts the coefficient of x^n in the polynomial p.
- If the third argument is omitted, it is determined by looking at the second argument. Thus coeff(p,x^n) is equivalent to coeff(p,x,n) for n <> 0.
- The cases of the second argument being a number or a product are disallowed since they do not make sense.
- The related functions lcoeff, tcoeff and coeffs extract the leading coefficient, trailing coefficient and all the coefficients of p in x respectively.

Diferenciate

diff or Diff - Differentiation or Partial Differentiation

Calling Sequence

diff(a, x1, x2, ..., xn)
Diff(a, x1, x2, ..., xn)
diff(a, [x1, x2, ..., xn])
Diff(a, [x1, x2, ..., xn])

Parameters

`a`	-	an algebraic expression
`x1, x2, ..., xn`	-	names

Description

- diff computes the partial derivative of the expression a with respect to x1, x2, ..., xn, respectively. The most frequent use is diff(f(x),x), which computes the derivative of the function f(x) with respect to x.
- Note that where n is greater than 1, the call to diff is the same as diff called recursively. Thus diff(f(x), x, y); is equivalent to the call diff(diff (f(x), x), y).
- diff has a user interface that will call the user's own differentiation functions. If the procedure `diff/f` is defined then the function call diff(f(x, y, z), y) will invoke `diff/f`(x,y,z,y) to compute the derivative. See example below.
- The sequence operator $ is useful for forming higher-order derivatives. diff(f(x),x$4), for example, is equivalent to diff(f(x),x,x,x,x) and diff(g(x,y),x$2,y$3) is equivalent to diff(g(x,y),x,x,y,y,y)
- The names with respect to which the differentiation is to be done can also be given as a list of names. This format allows for the special case of differentiation with respect to no variables, in the form of an empty list. In this case, the result is simply the original expression, a. This format is especially useful when used together with the sequence operator and sequences with potentially zero variables.
- If the derivative cannot be expressed (if the expression is an undefined function), the diff function call itself is returned. (The prettyprinter displays the diff function in a two-dimensional d/dx format.)
- The diff command assumes that partial derivatives commute.
- The capitalized function name Diff is the inert diff function, which simply returns unevaluated. The prettyprinter understands Diff to be equivalent to diff for printing purposes but formats the derivative in black to visually distinguish the inert case.
- The differential operator D is also defined in Maple; see D. For a comparison of D and diff see operators[D].

B. Bibliography

[1] R. Johansson: System modeling and identification (Prentice Hall 2002)

[2] W Khalil: Modeling,identification & control of robots (Hermes Penton 2002)

[3] Lennart Ljung: System identification,Theory for users (Prentice Hall 1999)

[4] J.Craig : Introduction to robotics:Mechanics and Control(2nd Edition Addison-Weslwy,1989)

[5] Luh J.Y.S: On-line computational scheme for mechanical manipulators (Journal of Dynamic Systems,Measurements and Control Vol.102/69 ,June 1980)

[6] M. Gautier: A direct determination of the minimum inertial parameters of robots

[7] M. Gautier: Direct caculation of minimum set of Inertial parameters of serial Robots(IEEE transactiions on robotics and automation Vol 6 NO 3,June 1990)

[8] M. Gautier: Identification of the minimum inertial parameters of robots

[9] M. Gautier: On the identification of the inertial parameters of robots (Proceedings of the 27th conference on Decision and Control, December 1988)

[10] M. Gautier: Numerical calculation of the base inertial parameters of robots

[11] M. Gautier: Identification of the dynamic parameters of a closed loop robot

[12] W. Spong: Mechanical Design and Control of the Pendubot(46th Annual Earthmoving Industry Conference, April 1995)

[13] W. Spong: The mechatronic Control Kit for Education and Research

[14] G.C. Goodwin: Adaptative computed torque control for rigid link manipulators (Proceedings of 25th Conference on Decision and Control, December 1986)

[15] Maple the future of mathematics. Waterloo Maple 1991

[16] P. Mohseni.Maple in action. And introductory handbook. June 1995

[17] John V. The Maple dictionary with examples.1995

[18] K.J. Astrom.Adaptative control (2nd edition Addison-Wesley 1995)

[19] B. Raucent.Influence of the inertial parameters on the positional accuracy of a robot