Industrial Robotics

Notes from the course 140440 - Industrial Robotics, University of Trento.
Course professor: Daniele Bortoluzzi.

The document has been developed starting from the exam of 11/02/2020).
The sections and subsections illustrate how to solve typical tasks on open-chain robots.

Utility

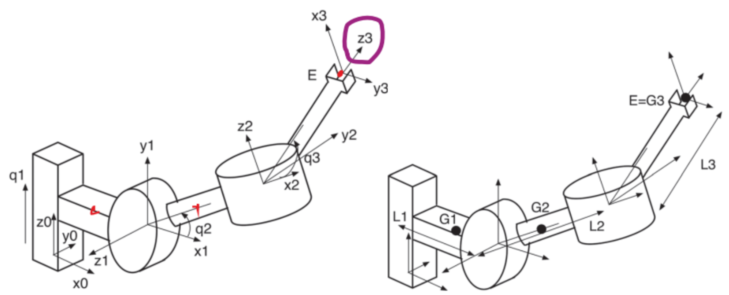
Configuration: Tools -> Options -> Display -> Input: "Maple notation", Output: "2-D Math notation"

```
> restart: with(LinearAlgebra): with(VectorCalculus): BasisFormat
  (false): with(plots):
```

```
> data:={L1=1,L2=0.6,L3=0.4,xE=1.765,yE=0.283,zE=0.541,m1=15,m2=7,
  m3=5,k1=20000,k2=5000,k3=10000,g=9.81,kpv=1000,kpd=15000};
```

$$data := \{L1 = 1, L2 = 0.6, L3 = 0.4, g = 9.81, k1 = 20000, k2 = 5000, k3 = 10000, kpd = 15000, kpv = 1000, m1 = 15, m2 = 7, m3 = 5, xE = 1.765, yE = 0.283, zE = 0.541\} \quad \textbf{(1.1)}$$

Exercise considered as reference



Rototranlsational matrices

Matrice rototraslazione su **Z**

```
> Mrotxtrasl:=(alpha,point)-> <
      <1,0,0,0>|
      <0,cos(alpha),sin(alpha),0>|
      <0,-sin(alpha),cos(alpha),0>|
      <point[1],point[2],point[3],1>
  >:
  "Mrotxtrasl"=Mrotxtrasl(theta(t),<a,b,c>);
```

$$\text{"Mrotxtrasl"} = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & \cos(\theta(t)) & -\sin(\theta(t)) & b \\ 0 & \sin(\theta(t)) & \cos(\theta(t)) & c \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(1.2.1)}$$

Matrice rototralsazione su **Y**

```
> Mrotytrasl:=(alpha,point)-> <
      <cos(alpha),0,-sin(alpha),0>|
      <0,1,0,0>|
      <sin(alpha),0,cos(alpha),0>|
```

```
    <point[1],point[2],point[3],1>
>:
"Mrotytrasl"=Mrotytrasl(theta(t),<a,b,c>);
```

$$"\text{Mrotytrasl}" = \begin{bmatrix} \cos(\theta(t)) & 0 & \sin(\theta(t)) & a \\ 0 & 1 & 0 & b \\ -\sin(\theta(t)) & 0 & \cos(\theta(t)) & c \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.2.2}$$

Matrice rototralsazione su **Z**

```
> Mrotztrasl:=(alpha,point)-> <
    <cos(alpha),sin(alpha),0,0>|
    <-sin(alpha),cos(alpha),0,0>|
    <0,0,1,0>|
    <point[1],point[2],point[3],1>
>:
"Mrotztrasl"=Mrotztrasl(theta(t),<a,b,c>);
```

$$"\text{Mrotztrasl}" = \begin{bmatrix} \cos(\theta(t)) & -\sin(\theta(t)) & 0 & a \\ \sin(\theta(t)) & \cos(\theta(t)) & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.2.3}$$

L matrices

```
> Lrotx   := <<0,0,0,0>|<0,0,1,0>|<0,-1,0,0>|<0,0,0,0>>:
  Lrotz   := <<0,1,0,0>|<-1,0,0,0>|<0,0,0,0>|<0,0,0,0>>:
  Ltraslx := <<0,0,0,0>|<0,0,0,0>|<0,0,0,0>|<1,0,0,0>>:
  Ltraslz := <<0,0,0,0>|<0,0,0,0>|<0,0,0,0>|<0,0,1,0>>:
  "Lrotx"=Lrotx,"Lrotz"=Lrotz, "Ltraslx"=Ltraslx;
```

$$"\text{Lrotx}" = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, "\text{Lrotz}" = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, "\text{Ltraslx}" = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{1.3.1}$$

Changing frame

Rototranslation

Mji:=MatrixInverse(Mij)

Velocity related

Wij_k:=Mki.Wij_i.MatrixInverse(Mki)

Hij_k:=Mki.Hij_i.MatrixInverse(Mki)

Lij_k:=Mki.Lij_i.MatrixInverse(Mki)

Pseudo inertia tensor

J_change_ref_func:=(M,J)->simplify(M.J.Transpose(M));

Jji:=Mij.Jii.Transpose(Mij)

Matrix of action

```
Phi_ji:=Mij.Phiii.Transpose(Mij)
```

Kinematic

Position analysis

Direct kinematic

Body 1

```
> M01:=Mrotxtrasl(0,<0,0,q1(t)>).Mrotxtrasl(Pi/2,<L1,0,0>);
```

$$M01 := \begin{bmatrix} 1 & 0 & 0 & L1 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & q1(t) \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.1.1.1}$$

Body 2

```
> M12:=Mrotztrasl(q2(t),0)
      .Mrotxtrasl(0,<L2,0,0>)
      .Mrotxtrasl(-Pi/2,0):
  M02:=M01.M12;
```

$$M02 := \begin{bmatrix} \cos(q2(t)) & 0 & -\sin(q2(t)) & \cos(q2(t))\,L2 + L1 \\ 0 & 1 & 0 & 0 \\ \sin(q2(t)) & 0 & \cos(q2(t)) & \sin(q2(t))\,L2 + q1(t) \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.1.1.2}$$

Body 3

```
> M23:=Mrotztrasl(q3(t),0)
      .Mrotxtrasl(0,<L3,0,0>)
      .Mrotytrasl(-Pi/2,0):
  Mrotztrasl(q3(t),0).Mrotxtrasl(0,<L3,0,0>).<<0,0,1,0>|<0,-1,0,
  0>|<1,0,0,0>|<0,0,0,1>>: "M23_first"=<%%>,"M23_second"=<%>;
  M03:=M02.M23:
```

$$\text{"M23\_first"} = \begin{bmatrix} 0 & -\sin(q3(t)) & -\cos(q3(t)) & \cos(q3(t))\,L3 \\ 0 & \cos(q3(t)) & -\sin(q3(t)) & \sin(q3(t))\,L3 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \text{"M23\_second"} \tag{2.1.1.3}$$

$$= \begin{bmatrix} 0 & \sin(q3(t)) & \cos(q3(t)) & \cos(q3(t))\,L3 \\ 0 & -\cos(q3(t)) & \sin(q3(t)) & \sin(q3(t))\,L3 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

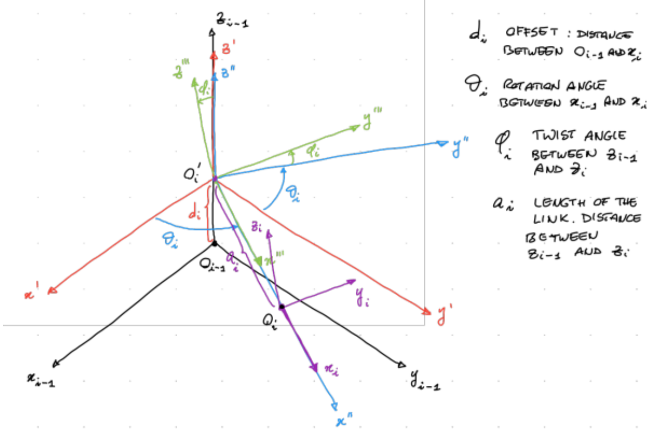
Variables

```
> vars:=[q1(t),q2(t),q3(t)];
```

$$vars := [q1(t), q2(t), q3(t)] \tag{2.1.1.4}$$

Denavit Hartenberg

Note: the DH method can only be used when, going back along x(i) you reach z(i-1)



$${}^{i-1}_{\;\;i}[M] = \text{TRANSL.}(z_{i-1}, d_i) \cdot \text{ROT}(z_{i-1}, \vartheta_i) \cdot \text{ROT}(x_i, \varphi_i) \cdot \text{TRANSL.}(x_i, a_i)$$

The approach does this transformation:

```
> Mrotztrasl(theta,<0,0,d>).Mrotxtrasl(phi,<a,0,0>);
```

$$\begin{bmatrix} \cos(\theta) & -\sin(\theta)\cos(\phi) & \sin(\theta)\sin(\phi) & \cos(\theta)\,a \\ \sin(\theta) & \cos(\theta)\cos(\phi) & -\cos(\theta)\sin(\phi) & \sin(\theta)\,a \\ 0 & \sin(\phi) & \cos(\phi) & d \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(2.1.2.1)}$$

I want a function to which I can pass <d,theta,phi,a>

```
> DH2M:=(dh)-> <
      <cos(dh[2]),sin(dh[2]),0,0>|
      <-sin(dh[2])*cos(dh[3]),cos(dh[2])*cos(dh[3]),sin(dh[3]),0>|
      <sin(dh[2])*sin(dh[3]),-cos(dh[2])*sin(dh[3]),cos(dh[3]),0>|
      <dh[4]*cos(dh[2]),dh[4]*sin(dh[2]),dh[1],1>
  >:
> M01_DH:=DH2M(<q1,0,Pi/2,L1>): "M01_DH"=M01_DH,"M01"=M01;
```

$$\text{"M01\_DH"} = \begin{bmatrix} 1 & 0 & 0 & L1 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & q1 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \text{"M01"} = \begin{bmatrix} 1 & 0 & 0 & L1 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & q1(t) \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(2.1.2.2)}$$

```
> M12_DH:=DH2M(<0,q2(t),-Pi/2,L2>): "M12_DH"=M12_DH,"M12"=M12;
```

$$\text{"M12\_DH"} = \begin{bmatrix} \cos(q2(t)) & 0 & -\sin(q2(t)) & \cos(q2(t))\,L2 \\ \sin(q2(t)) & 0 & \cos(q2(t)) & \sin(q2(t))\,L2 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \text{"M12"} \quad \textbf{(2.1.2.3)}$$

$$= \begin{bmatrix} \cos(q2(t)) & 0 & -\sin(q2(t)) & \cos(q2(t))\,L2 \\ \sin(q2(t)) & 0 & \cos(q2(t)) & \sin(q2(t))\,L2 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Not possible to code it because the last rotation is about y and DH does not provide anything for this.

```
> M23_DH:=...
```

$$M23\_DH := .. \tag{2.1.2.4}$$

Position of the end effector

```
> E:=M03.<0,0,0,1>;
```

$$E := \begin{bmatrix} \cos(q2(t))\cos(q3(t))\,L3 + \cos(q2(t))\,L2 + L1 \\ \sin(q3(t))\,L3 \\ \sin(q2(t))\cos(q3(t))\,L3 + \sin(q2(t))\,L2 + q1(t) \\ 1 \end{bmatrix} \tag{2.1.3.1}$$

Jacobian matrix

```
> JS:=subs(
        q1=q1(t),q2=q2(t),q3=q3(t),
        Jacobian(subs(q1(t)=q1,q2(t)=q2,q3(t)=q3,E[1..3]),[q1,q2,q3]
  )
  );
```

$$JS := \begin{bmatrix} 0 & -\sin(q2(t))\cos(q3(t))\,L3 - \sin(q2(t))\,L2 & -\cos(q2(t))\sin(q3(t))\,L3 \\ 0 & 0 & \cos(q3(t))\,L3 \\ 1 & \cos(q2(t))\cos(q3(t))\,L3 + \cos(q2(t))\,L2 & -\sin(q2(t))\sin(q3(t))\,L3 \end{bmatrix} \tag{2.1.4.1}$$

Singular configuration

```
> det:=Determinant(JS);
```

$$det := -\sin(q2(t))\,(\cos(q3(t))\,L3 + L2)\cos(q3(t))\,L3 \tag{2.1.5.1}$$

```
> SCs:=solve(det=0,vars);
```

$$SCs := \left[\left[q1(t) = q1(t), q2(t) = q2(t), q3(t) = \pi - \arccos\left(\frac{L2}{L3}\right)\right], \left[q1(t) = q1(t), q2(t) = q2(t), q3(t) = \frac{\pi}{2}\right], [q1(t) = q1(t), q2(t) = 0, q3(t) = q3(t)]\right] \tag{2.1.5.2}$$

```
> "sing1"=SCs[1], "sing2"=SCs[2], "sing3"=SCs[3];
```

$$\text{"sing1"} = \left[q1(t) = q1(t), q2(t) = q2(t), q3(t) = \pi - \arccos\left(\frac{L2}{L3}\right)\right], \text{"sing2"} = \left[q1(t) = q1(t), q2(t) = q2(t), q3(t) = \frac{\pi}{2}\right], \text{"sing3"} = [q1(t) = q1(t), q2(t) = 0, q3(t) = q3(t)] \tag{2.1.5.3}$$

Kiss:
- still even with activated motors
- impossibility to set a velocity
- ellipsoids to infinite force

Inverse kinematic with position

Find the joint coordinates which produce the coordinate E__desired=[...]

Given x,y,z find q1,q2,q3

```
> E__desired:=[x__E,y__E,z__E]:
> inv_kin_sols:=solve(subs(data,[E[1]=xE,E[2]=yE,E[3]=zE]),[q1(t),
  q2(t),q3(t)]);
```

$$inv\_kin\_sols := [[q1(t) = 0.1006496793, q2(t) = 0.5223011036, q3(t) = 0.7859544135], [q1(t) = 0.5410000000 - 0.6960864924\,I, q2(t) = 0. + 1.527041703\,I, q3(t) = 2.355638240]] \quad \textbf{(2.1.6.1)}$$

```
> nops(inv_kin_sols);
```

$$2 \quad \textbf{(2.1.6.2)}$$

The second solution is imaginary, so the only acceptable is the first

```
> sol:=inv_kin_sols[1];
```

$$sol := [q1(t) = 0.1006496793, q2(t) = 0.5223011036, q3(t) = 0.7859544135] \quad \textbf{(2.1.6.3)}$$

Alternatively, set other equations with the requested equality, as long as nop = neq.

Additional requests

Find the component of the z3 unit vector project in frame 0 when E is in the requested position

The projection of the z3 unit vector in frame 0 corresponds to the third column of the rototranslation matrix M03. The point here is to substitute the values of the just obtained solution inside the mentioned column

```
> evalf(subs(data,sol,M03[1..3,3]));
```

$$\begin{bmatrix} -0.6124897343 \\ -0.7075000000 \\ -0.3525621581 \end{bmatrix} \quad \textbf{(2.1.7.1)}$$

Velocity analysis

Velocity matrices

We can calulate them in two ways: using M and its derivative or using L

For the first way we need:

```
> W_func:=(M)->simplify(map(diff,M,t).MatrixInverse(M));
```

$$W\_func := M \mapsto simplify(map(VectorCalculus{:-}diff, M, t) \cdot LinearAlgebra{:-}MatrixInverse(M)) \quad \textbf{(2.2.1.1)}$$

The second method is useful only because we calculate Lij, used to calculate the **non lagrangian component**

Body 1

First way, using M

```
> W01:=W_func(M01);
```

$$W01 := \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{d}{dt} q1(t) \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \textbf{(2.2.1.1.1)}$$

Second way, using L

```
> L01:=Ltraslz: W01:=L01*diff(q1(t),t);
```

(2.2.1.1.2)

$$W01 := \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{d}{dt}\, q1(t) \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \textbf{(2.2.1.1.2)}$$

Body 2

First way, using M

```
> W02:=W_func(M02);
```

$$W02 := \begin{bmatrix} 0 & 0 & -\frac{d}{dt}\, q2(t) & \left(\frac{d}{dt}\, q2(t)\right) q1(t) \\ 0 & 0 & 0 & 0 \\ \frac{d}{dt}\, q2(t) & 0 & 0 & -L1 \left(\frac{d}{dt}\, q2(t)\right) + \frac{d}{dt}\, q1(t) \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \textbf{(2.2.1.2.1)}$$

Second way, using L

```
> L12_1:=Lrotz:                         # definition
  L12:=M01.L12_1.MatrixInverse(M01):    # change of rf
  W12:=simplify(L12*diff(q2(t),t)):     # definition
  W02:=simplify(W01+W12): "W02"=W02;    # Rival's theorem
```

$$\text{"W02"} = \begin{bmatrix} 0 & 0 & -\frac{d}{dt}\, q2(t) & \left(\frac{d}{dt}\, q2(t)\right) q1(t) \\ 0 & 0 & 0 & 0 \\ \frac{d}{dt}\, q2(t) & 0 & 0 & -L1 \left(\frac{d}{dt}\, q2(t)\right) + \frac{d}{dt}\, q1(t) \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \textbf{(2.2.1.2.2)}$$

Body 3

First way, using M

```
> W03:=W_func(M03);
```

$$W03 := \left[\left[0,\ -\left(\frac{d}{dt}\, q3(t)\right)\cos(q2(t)),\ -\frac{d}{dt}\, q2(t),\ \left(\frac{d}{dt}\, q2(t)\right) q1(t)\right],\right. \quad \textbf{(2.2.1.3.1)}$$

$$\left[\left(\frac{d}{dt}\, q3(t)\right)\cos(q2(t)),\ 0,\ \left(\frac{d}{dt}\, q3(t)\right)\sin(q2(t)),\ -\left(\frac{d}{dt}\, q3(t)\right)(L1\cos(q2(t)) \right.$$

$$\left. + \sin(q2(t))\, q1(t) + L2)\right],$$

$$\left[\frac{d}{dt}\, q2(t),\ -\left(\frac{d}{dt}\, q3(t)\right)\sin(q2(t)),\ 0,\ -L1\left(\frac{d}{dt}\, q2(t)\right) + \frac{d}{dt}\, q1(t)\right],$$

$$\left.\left[0,\ 0,\ 0,\ 0\right]\right]$$

Second way, using L

```
> L23_2:=Lrotz:                           # definition
  L23:=M02.L23_2.MatrixInverse(M02):      # change of rf
  W23:=simplify(L23.diff(q3(t),t)):       # definition
  W03:=simplify(W02+W23);                 # Rival's theorem
```

$$W03 := \left[\left[0, -\left(\frac{d}{dt}\, q3(t)\right)\cos(q2(t)), -\frac{d}{dt}\, q2(t), \left(\frac{d}{dt}\, q2(t)\right) q1(t)\right], \left[\left(\frac{d}{dt}\, q3(t)\right)\cos(q2(t)), 0, \left(\frac{d}{dt}\, q3(t)\right)\sin(q2(t)), -\left(\frac{d}{dt}\, q3(t)\right)(L1\cos(q2(t)) + \sin(q2(t))\, q1(t) + L2)\right], \left[\frac{d}{dt}\, q2(t), -\left(\frac{d}{dt}\, q3(t)\right)\sin(q2(t)), 0, -L1\left(\frac{d}{dt}\, q2(t)\right) + \frac{d}{dt}\, q1(t)\right], [0, 0, 0, 0]\right] \quad (2.2.1.3.2)$$

Direct kinematic

```
> VE:=simplify(combine(W03.E));
```

$$VE := \left[\left[-\sin(q2(t))\,(\cos(q3(t))\, L3 + L2)\left(\frac{d}{dt}\, q2(t)\right) - \cos(q2(t))\left(\frac{d}{dt}\, q3(t)\right)\sin(q3(t))\, L3\right], \left[\left(\frac{d}{dt}\, q3(t)\right)\cos(q3(t))\, L3\right], \left[\cos(q2(t))\,(\cos(q3(t))\, L3 + L2)\left(\frac{d}{dt}\, q2(t)\right) - \sin(q2(t))\left(\frac{d}{dt}\, q3(t)\right)\sin(q3(t))\, L3 + \frac{d}{dt}\, q1(t)\right], [0]\right] \quad (2.2.2.1)$$

For example, velocity of the end effector at **sol** with unitary join velocity

```
> evalf(subs(data,diff(q1(t),t)=1,diff(q2(t),t)=1,diff(q3(t),t)=1,
  sol,VE[1..3]));
```

$$\begin{bmatrix} -0.6856189241 \\ 0.2826853374 \\ 1.623818159 \end{bmatrix} \quad (2.2.2.2)$$

Inverse kinematic

Another way to get the jacobian....

```
> JS_bis,V:=GenerateMatrix([VE[1]=Vx,VE[2]=Vy,VE[3]=Vz],[diff(q1
  (t),t),diff(q2(t),t),diff(q3(t),t)]): "JS_bis"=JS_bis, "JS"=JS;
```

$$\text{"JS\_bis"} = \begin{bmatrix} 0 & -\sin(q2(t))\,(\cos(q3(t))\, L3 + L2) & -\cos(q2(t))\sin(q3(t))\, L3 \\ 0 & 0 & \cos(q3(t))\, L3 \\ 1 & \cos(q2(t))\,(\cos(q3(t))\, L3 + L2) & -\sin(q2(t))\sin(q3(t))\, L3 \end{bmatrix}, \text{"JS"} \quad (2.2.3.1)$$

$$= \begin{bmatrix} 0 & -\sin(q2(t))\cos(q3(t))\,L3 - \sin(q2(t))\,L2 & -\cos(q2(t))\sin(q3(t))\,L3 \\ 0 & 0 & \cos(q3(t))\,L3 \\ 1 & \cos(q2(t))\cos(q3(t))\,L3 + \cos(q2(t))\,L2 & -\sin(q2(t))\sin(q3(t))\,L3 \end{bmatrix}$$

The following relation between end effector velocity and joint variables velocities holds:
S_dot= JS.Q_dot

Position in q (vars) and velocity (Vx, Vy, Vz) have to be known...

```
> Q_dot:=MatrixInverse(JS).<Vx,Vy,Vz>;
```

$$Q\_dot := \tag{2.2.3.2}$$

$$\left[\left[\frac{\cos(q2(t))\,Vx}{\sin(q2(t))} + \frac{\left(\cos(q2(t))^2 + \sin(q2(t))^2\right)\sin(q3(t))\,Vy}{\sin(q2(t))\cos(q3(t))} + Vz\right],\right.$$

$$\left[-\frac{Vx}{\sin(q2(t))\,(\cos(q3(t))\,L3 + L2)} - \frac{\cos(q2(t))\sin(q3(t))\,Vy}{\sin(q2(t))\,(\cos(q3(t))\,L3 + L2)\cos(q3(t))}\right],$$

$$\left.\left[\frac{Vy}{\cos(q3(t))\,L3}\right]\right]$$

For example

```
> simplify(evalf(subs(data,sol,sol,Vx=1,Vy=1,Vz=1,Q_dot)));
```

$$\begin{bmatrix} 4.743991532 \\ -4.241255505 \\ 3.537502190 \end{bmatrix} \tag{2.2.3.3}$$

Acceleration analysis

```
> H_func:=(M)->simplify(map(diff,M,t,t).MatrixInverse(M));
```

$$H\_func := M \mapsto simplify(map(VectorCalculus:-diff, M, t, t) \cdot LinearAlgebra:-MatrixInverse(M)) \tag{2.3.1}$$

```
> H01:=H_func(M01):
> H02:=H_func(M02):
> H03:=H_func(M03):
> "H01"=H01,"H02"=...,"H03"=...
```

$$\text{"H01"} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{d^2}{dt^2}\,q1(t) \\ 0 & 0 & 0 & 0 \end{bmatrix}, \text{"H02"} = .., \text{"H03"} = .. \tag{2.3.2}$$

Auxiliary frame

By definition, the transformation from the auxiliary frame to the fixed one is quite simple:

- the **rotation block** is the identity matrix as there is no rotation
- the **translation vector** is the position of the end effector

Position

```
> M0a:=<<1,0,0,0>|<0,1,0,0>|<0,0,1,0>|<E[1],E[2],E[3],1>>:
```

Velocity

To get the velocity matrices w.r.t **a** two possibilities:
- change the frame of all the L matrices and multiply them by the join velocity
- change directly the coordinates of the W matrices from one fr to another

Using L

```
> L01_a:=MatrixInverse(M0a).L01.M0a:
> W01_a:=L01_a*diff(q1(t),t): "L01_a"=L01_a, "W01_a"=W01_a;
```

$$\text{"L01\_a"} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \text{"W01\_a"} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{d}{dt}\, q1(t) \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \textbf{(2.4.2.1.1)}$$

```
> L12_a:=MatrixInverse(M0a).L12.M0a:
> W12_a:=L12_a*diff(q2(t),t): "L12_a"=L12_a,"W12_a"=W12_a;
```

$$\text{"L12\_a"} = \begin{bmatrix} 0 & 0 & -1 & -\sin(q2(t))\cos(q3(t))\, L3 - \sin(q2(t))\, L2 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & \cos(q2(t))\cos(q3(t))\, L3 + \cos(q2(t))\, L2 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \text{"W12\_a"} \quad \textbf{(2.4.2.1.2)}$$

$$= \Big[\Big[0,\ 0,\ -\frac{d}{dt}\, q2(t),\ \left(\frac{d}{dt}\, q2(t)\right)\left(-\sin(q2(t))\cos(q3(t))\, L3 - \sin(q2(t))\, L2\right)\Big],$$
$$\Big[0,\ 0,\ 0,\ 0\Big],$$
$$\Big[\frac{d}{dt}\, q2(t),\ 0,\ 0,\ \left(\frac{d}{dt}\, q2(t)\right)\left(\cos(q2(t))\cos(q3(t))\, L3 + \cos(q2(t))\, L2\right)\Big],$$
$$\Big[0,\ 0,\ 0,\ 0\Big]\Big]$$

```
> L23_a:=MatrixInverse(M0a).L23.M0a:
> W23_a:=L23_a*diff(q3(t),t): "L23_a"="...long...","W12_a"=W12_a;
```

"L23_a" = "...long...", "W12_a" **(2.4.2.1.3)**

$$= \Big[\Big[0,\ 0,\ -\frac{d}{dt}\, q2(t),\ \left(\frac{d}{dt}\, q2(t)\right)\left(-\sin(q2(t))\cos(q3(t))\, L3 - \sin(q2(t))\, L2\right)\Big],$$

$$\left[0,\ 0,\ 0,\ 0\right],$$

$$\left[\frac{d}{dt}\ q2(t),\ 0,\ 0,\ \left(\frac{d}{dt}\ q2(t)\right)\ (\cos(q2(t))\cos(q3(t))\ L3+\cos(q2(t))\ L2)\right],$$

$$\left[0,\ 0,\ 0,\ 0\right]\Big]$$

```
> WO3_a:=W01_a+W12_a+W23_a:
```

Using M

```
> _W01_a:=MatrixInverse(M0a).W01.M0a:
> _W12_a:=MatrixInverse(M0a).W12.M0a:
> _W23_a:=MatrixInverse(M0a).W23.M0a:
> _W03_a:=_W01_a+_W12_a+_W23_a:
```

Manipulability ellipsoids

Need of another robot, consider the SCARA

```
> M01_scara:=Mrotztrasl(theta1(t),<0,0,h>).Mrotztrasl(0,<L1,0,0>):
  M12_scara:=Mrotztrasl(theta2(t),<0,0,0>).Mrotztrasl(0,<L2,0,0>):
  M23_scara:=Mrotztrasl(0,<0,0,-L(t)>):
  M03_scara:=M01_scara.M12_scara.M23_scara:
  E_scara:=M03_scara.<0,0,0,1>;
```

$$E\_scara := \begin{bmatrix} \cos(\theta 1(t))\cos(\theta 2(t))\ L2-\sin(\theta 1(t))\sin(\theta 2(t))\ L2+\cos(\theta 1(t))\ L1 \\ \sin(\theta 1(t))\cos(\theta 2(t))\ L2+\cos(\theta 1(t))\sin(\theta 2(t))\ L2+\sin(\theta 1(t))\ L1 \\ -L(t)+h \\ 1 \end{bmatrix} \quad \textbf{(2.5.1)}$$

We restrict E to the first two components to do the analysis.
We don't need to consider any time derivative, so no function of time t.

2D

```
> Ep_2D:=subs(theta1(t)=theta1,theta2(t)=theta2,E_scara[1..2]);
```

$$Ep\_2D := \begin{bmatrix} \cos(\theta 1)\cos(\theta 2)\ L2-\sin(\theta 1)\sin(\theta 2)\ L2+\cos(\theta 1)\ L1 \\ \sin(\theta 1)\cos(\theta 2)\ L2+\cos(\theta 1)\sin(\theta 2)\ L2+\sin(\theta 1)\ L1 \end{bmatrix} \quad \textbf{(2.5.1.1)}$$

```
> (JS_ell,det_ell):=Jacobian(Ep_2D,[theta1,theta2],'determinant'=
  true);
```

$$JS\_ell, det\_ell := [[-\sin(\theta 1)\cos(\theta 2)\ L2-\cos(\theta 1)\sin(\theta 2)\ L2-\sin(\theta 1)\ L1, \quad \textbf{(2.5.1.2)}$$

$$-\cos(\theta 1)\sin(\theta 2)\ L2-\sin(\theta 1)\cos(\theta 2)\ L2],$$

$$[\cos(\theta 1)\cos(\theta 2)\ L2-\sin(\theta 1)\sin(\theta 2)\ L2+\cos(\theta 1)\ L1,\ -\sin(\theta 1)\sin(\theta 2)\ L2$$

$$+\cos(\theta 1)\cos(\theta 2)\ L2]],\ L1\ L2\sin(\theta 1)^2\sin(\theta 2)+L1\ L2\cos(\theta 1)^2\sin(\theta 2)$$

```
> data_scara:={L1=0.3,L2=0.3}:
> ref:={theta1=Pi/6,theta2=Pi/3}:
```

Force

```
> F_ell:=<Fx,Fy>:
> AF:=JS_ell.Transpose(JS_ell):
```

```
> AF_num:=evalf(subs(data_scara,ref,AF)):
> (e_val_F,e_vec_F):=Eigenvectors(AF_num);
```

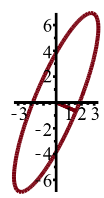
$$e\_val\_F, e\_vec\_F := \begin{bmatrix} 0.342249807638966 + 0.\,I \\ 0.0177501926410342 + 0.\,I \end{bmatrix}, \tag{2.5.1.1.1}$$

$$\begin{bmatrix} 0.920156303523247 + 0.\,I & 0.391551244521627 + 0.\,I \\ -0.391551244521627 + 0.\,I & 0.920156303523247 + 0.\,I \end{bmatrix}$$

```
> Ell_F:=Transpose(F_ell).AF_num.F_ell-Kf^2;
```

$$Ell\_F := Fx\,(0.2925000003\,Fx - 0.1169134295\,Fy) + Fy\,(-0.1169134295\,Fx + 0.06749999998\,Fy) - Kf^2 \tag{2.5.1.1.2}$$

```
> plots[display]([
          plot([[0,0],1/sqrt(Re(e_val_F[1]))*[Re(e_vec_F[1,1]),Re
  (e_vec_F[2,1])]]),
          plots[contourplot](subs(data_scara,ref,Kf=1,Ell_F),Fx=
  -10..10,Fy=-10..10,contours=[0],grid=[100,100])
      ],
      scaling=constrained);
```



Maximum force along a given direction alpha

Along a direction expressed with an angle the following realtion holds: **Fy/Fx=tan(alpha)**

```
> Fydir:=Fxdir*tan(alpha): angle:={alpha=Pi/3}:
```

There are two symmetric solutions.

```
> eq_dir:=evalf(subs(data_scara,ref,Kf=1,Fx=Fxdir,Fy=Fydir,angle,
  Ell_F));
```

$$eq\_dir := 0.09000000023\,Fxdir^2 - 1. \tag{2.5.1.1.3}$$

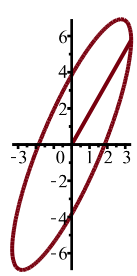
```
> Fx1,Fx2:=solve(eq_dir=0,Fxdir);
```

$$Fx1, Fx2 := 3.333333329, -3.333333329 \tag{2.5.1.1.4}$$

```
> Fy1:=evalf(subs(Fxdir=Fx1,angle,Fydir));
```

$$Fy1 := 5.773502682 \tag{2.5.1.1.5}$$

```
> plots[display]([
      plot([[0,0],[Fx1,Fy1]]),
      plots[contourplot](subs(data_scara,ref,Kf=1,Ell_F),Fx=-10.
  .10,Fy=-10..10,contours=[0],grid=[100,100],scaling=constrained)
  ]);
```




```
> sqrt(Fx1^2+Fy1^2);
```

(2.5.1.1.6)

$$6.666666656 \tag{2.5.1.1.6}$$

Velocity

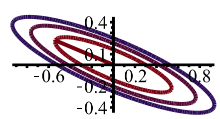
```
> V_ell:=<Vx,Vy>:
> AV:=combine(MatrixInverse(JS_ell.Transpose(JS_ell))):
> AV_num:=evalf(subs(data_scara,ref,AV)):
> (e_val_V,e_vec_V):=Eigenvectors(AV_num);
```

$$e\_val\_V, e\_vec\_V := \begin{bmatrix} 2.92184241987786 + 0.\,I \\ 56.3374168401221 + 0.\,I \end{bmatrix}, \begin{bmatrix} -0.920156303499675 + 0.\,I & -0.391551244577021 + 0.\,I \\ 0.391551244577021 + 0.\,I & -0.920156303499675 + 0.\,I \end{bmatrix} \tag{2.5.1.2.1}$$

```
> Ell_V:=Transpose(V_ell).AV_num.V_ell-Kv^2;
```

$$Ell\_V := Vx\,(11.11111111\,Vx + 19.24500895\,Vy) + Vy\,(19.24500895\,Vx + 48.14814815\,Vy) - Kv^2 \tag{2.5.1.2.2}$$

```
> plots[display]([
         plot([[0,0],1/sqrt(Re(e_val_V[1]))*[Re(e_vec_V[1,1]),Re
  (e_vec_V[2,1])]]),
         plots[contourplot](subs(data_scara,ref,Kv=1,Ell_V),Vx=-1.
  .1,Vy=-1..1,contours=[0,1,2],grid=[100,100])
      ],
      scaling=constrained);
```



Compliance

```
> dS_ell:=<dx,dy>:
> Kq:=k*<<1,0>|<0,2>>:
> Ks:=JS_ell.Kq.Transpose(JS_ell):
> AC:=combine(Transpose(MatrixInverse(Ks)).MatrixInverse(Ks)):
> AC_num:=evalf(subs(data_scara,ref,k=1,AC));
```

$$AC\_num := \begin{bmatrix} 123.4567900 & 356.3890549 \\ 356.3890549 & 1083.676269 \end{bmatrix} \tag{2.5.1.3.1}$$

```
> (e_val_C,e_vec_C):=Eigenvectors(AC_num);
```

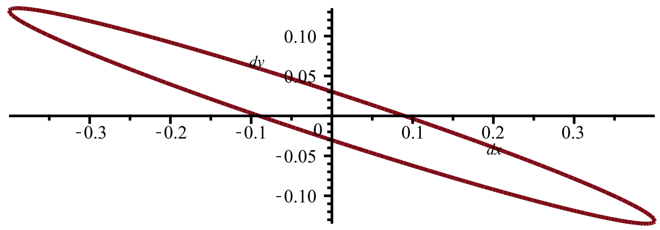
$$e\_val\_C, e\_vec\_C := \begin{bmatrix} 5.63800499621539 + 0.\,I \\ 1201.49505400378 + 0.\,I \end{bmatrix}, \begin{bmatrix} -0.949461707692608 + 0.\,I & -0.313882885206309 + 0.\,I \\ 0.313882885206309 + 0.\,I & -0.949461707692608 + 0.\,I \end{bmatrix} \tag{2.5.1.3.2}$$

```
> Ell_C:=Transpose(dS_ell).AC_num.dS_ell-Ka^2;
```

$$Ell\_C := dx\,(123.4567900\,dx + 356.3890549\,dy) + dy\,(356.3890549\,dx + 1083.676269\,dy) - Ka^2 \tag{2.5.1.3.3}$$

```
> plots[contourplot](
```

```
    eval(subs(data_scara,ref,Ka=1,k=1,Ell_C)),
    dx=-0.5..0.5,dy=-0.5..0.5,contours=[0],grid=[300,300],
    scaling=constrained
);
```



Manipulability ellipsoids (3D)

Force, velocity, whatever along a direction u

```
> ref3D:={theta1=Pi/6,theta2=Pi/3,L_scara=0.1};
```

$$ref3D := \left\{L\_scara = 0.1, \theta 1 = \frac{\pi}{6}, \theta 2 = \frac{\pi}{3}\right\} \tag{2.5.2.1}$$

Vector along which calculate the velocity:

```
> u:=<-2,1,0>;
```

$$u := \begin{bmatrix} -2 \\ 1 \\ 0 \end{bmatrix} \tag{2.5.2.2}$$

Velocity budget

```
> Ep3D:=subs(theta1(t)=theta1,theta2(t)=theta2,L(t)=L_scara,E_scara
  [1..3]);
```

$$Ep3D := \begin{bmatrix} \cos(\theta 1)\cos(\theta 2) L2 - \sin(\theta 1)\sin(\theta 2) L2 + \cos(\theta 1) L1 \\ \sin(\theta 1)\cos(\theta 2) L2 + \cos(\theta 1)\sin(\theta 2) L2 + \sin(\theta 1) L1 \\ -L\_scara + h \end{bmatrix} \tag{2.5.2.3}$$

```
> JS3D,det:=Jacobian(Ep3D,[theta1,theta2,L_scara],'determinant'=
  true):
> V3D:=<Vx,Vy,Vz>:
> AV3D:=combine(MatrixInverse(JS3D.Transpose(JS3D))):
> AV3D_num:=evalf(subs(data_scara,ref,AV3D)):
> Ell_V3D:=Transpose(V3D).AV3D_num.V3D-Kv^2;
```

$$Ell\_V3D := Vx\,(11.11111111\,Vx + 19.24500895\,Vy) + Vy\,(19.24500895\,Vx \tag{2.5.2.4}$$

$+ 48.14814815\, Vy) + 1.\, Vz^2 - Kv^2$

Two solutions..

```
> u_stretch:=evalf(solve(subs(Kv=3,Vx=u[1]*t,Vy=u[2]*t,Vz=u[3]*t,
  Ell_V3D),t))[1];
```

$$u\_stretch := 0.7592490185 \quad \textbf{(2.5.2.5)}$$

```
> Vx3D:=u[1]*u_stretch: Vy3D:=u[2]*u_stretch: Vz3D:=u[3]*u_stretch:
  "V3Dmax"=<Vx3D,Vy3D,Vz3D>;
```

$$\text{"V3Dmax"} = \begin{bmatrix} -1.518498037 \\ 0.7592490185 \\ 0. \end{bmatrix} \quad \textbf{(2.5.2.6)}$$

Kineto-statics

It is the part of the mechanics that deals witht the determination of the constrained reactions to which a partially constrained moving body i is subject.
It represents a first steps toward dynamic but, differently to this one, it considers the boyd to remain still.
Our object has been to find a relation between the a generic vector of actions Fs and the corresponding set of acitons Fq that the joints would have had to apply to balance the external ones.
Applying the principle of virtual work we found the **two equations of kineto statics** for which:
- Fq= -Transpose(JS).Fs
- Fs=-Inverse(Transpose(JS)).Fq

Notes:
- the Jacobian has to be calculated for the point in which the force acts
- the external force F is expressed with reference to the fixed frame

Steps:
- convert the coordinates of the points where forces are applied into the fixed frame coordinates
- calculate the Jacobian matrix for the obtained point
- apply the formula to get the joint forces and torques given the external actions (or weight)

Calculate the joint forces and torques necessary to balance the weight

Given centres of mass

```
> G11:=<-L1/2,0,0,1>: G22:=<-L2/2,0,0,1>: G33:=<0,0,0,1>:
> "G11"=G11,"G22"=G22,"G33"=G33;
```

$$\text{"G11"} = \begin{bmatrix} -\frac{L1}{2} \\ 0 \\ 0 \\ 1 \end{bmatrix}, \text{"G22"} = \begin{bmatrix} -\frac{L2}{2} \\ 0 \\ 0 \\ 1 \end{bmatrix}, \text{"G33"} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \quad \textbf{(2.6.1.1)}$$

Convert the coordinates to the fixed reference frame

```
> G1:=M01.G11: G2:=M02.G22: G3:=M03.G33:
> "G1"=G1,"G2"=G2,"G3"=...
```

$$"G1" = \begin{bmatrix} \frac{L1}{2} \\ 0 \\ q1(t) \\ 1 \end{bmatrix}, "G2" = \begin{bmatrix} \frac{\cos(q2(t))\, L2}{2} + L1 \\ 0 \\ \frac{\sin(q2(t))\, L2}{2} + q1(t) \\ 1 \end{bmatrix}, "G3" =.. \tag{2.6.1.2}$$

Calculate the jacobian for the obtained points (projected in the fixed frame)

```
> JS_1:=subs(
        q1=q1(t),q2=q2(t),q3=q3(t),
        Jacobian(subs(q1(t)=q1,q2(t)=q2,q3(t)=q3,G1[1..3]),[q1,q2,
  q3])
  ):
> JS_2:=subs(
        q1=q1(t),q2=q2(t),q3=q3(t),
        Jacobian(subs(q1(t)=q1,q2(t)=q2,q3(t)=q3,G2[1..3]),[q1,q2,
  q3])
  ):
> JS_3:=subs(
        q1=q1(t),q2=q2(t),q3=q3(t),
        Jacobian(subs(q1(t)=q1,q2(t)=q2,q3(t)=q3,G3[1..3]),[q1,q2,
  q3])
  ):
> "JS_1"=JS_1,"JS_2"=JS_2,"JS_3"=...
```

$$"JS\_1" = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}, "JS\_2" = \begin{bmatrix} 0 & -\frac{\sin(q2(t))\, L2}{2} & 0 \\ 0 & 0 & 0 \\ 1 & \frac{\cos(q2(t))\, L2}{2} & 0 \end{bmatrix}, "JS\_3" =.. \tag{2.6.1.3}$$

Calculate the joint forces and torques using: Fq= -Transpose(JS).Fs

```
> Fs_G1:=<0,0,-m1*g>:
  Fs_G2:=<0,0,-m2*g>:
  Fs_G3:=<0,0,-m3*g>:
  "Fs_G1"=Fs_G1,"Fs_G2"=Fs_G2,"Fs_G3"=Fs_G3;
```

$$"Fs\_G1" = \begin{bmatrix} 0 \\ 0 \\ -m1\, g \end{bmatrix}, "Fs\_G2" = \begin{bmatrix} 0 \\ 0 \\ -m2\, g \end{bmatrix}, "Fs\_G3" = \begin{bmatrix} 0 \\ 0 \\ -m3\, g \end{bmatrix} \tag{2.6.1.4}$$

```
> Fq_G1:=-Transpose(JS_1).Fs_G1:
  Fq_G2:=-Transpose(JS_2).Fs_G2:
  Fq_G3:=-Transpose(JS_3).Fs_G3:
  "Fq_G1"=Fq_G1,"Fq_G2"=Fq_G2,"Fq_G3"=Fq_G3;
```

$$"Fq\_G1" = \begin{bmatrix} m1\, g \\ 0 \\ 0 \end{bmatrix}, "Fq\_G2" = \begin{bmatrix} m2\, g \\ \frac{\cos(q2(t))\, L2\, m2\, g}{2} \\ 0 \end{bmatrix}, "Fq\_G3" \tag{2.6.1.5}$$

$$= \begin{bmatrix} m3\, g \\ (\cos(q2(t))\cos(q3(t))\, L3 + \cos(q2(t))\, L2)\, m3\, g \\ -\sin(q2(t))\sin(q3(t))\, L3\, m3\, g \end{bmatrix}$$

Calculate the joint forces and torques necessary to balance a given force

The force is applied to the end effector, so we already have the corresponding jacobian

```
> Fe:=<Fx,Fy,Fz>;
```

$$Fe := \begin{bmatrix} Fx \\ Fy \\ Fz \end{bmatrix} \tag{2.6.2.1}$$

```
> Fq:=-Transpose(JS).Fe;
```

$$Fq := [[-Fz], [-(-\sin(q2(t))\cos(q3(t))\, L3 - \sin(q2(t))\, L2)\, Fx - (\cos(q2(t))\cos(q3(t))\, L3 + \cos(q2(t))\, L2)\, Fz], [\cos(q2(t))\sin(q3(t))\, L3\, Fx - \cos(q3(t))\, L3\, Fy + \sin(q2(t))\sin(q3(t))\, L3\, Fz]] \tag{2.6.2.2}$$

Sometimes it is better to collect the terms:

```
> collect(expand(Fq[1..3]),[Fx,Fy,Fz]):
```

Motion planning

Given the manouvre time Tmax

Write the joint time profiles required to perform in 2s the motion at constant acceleration from the initial (all joint 0) to the found sol.

```
> Tmax:=2:
```

Base profile given T

```
> base_profile:=piecewise(
      t>=0 and t<=T/2, qin+2*(qfin-qin)*(t/T)^2,
      t>T/2 and t<T, qfin-2*(qfin-qin)*(t-T)^2/T^2
  );
```

$$base\_profile := \begin{cases} qin + \dfrac{(2\, qfin - 2\, qin)\, t^2}{T^2} & 0 \le t \le \dfrac{T}{2} \\ qfin - \dfrac{(2\, qfin - 2\, qin)\, (t-T)^2}{T^2} & \dfrac{T}{2} < t < T \end{cases} \tag{2.7.1.1}$$

```
> q1_profile:=subs(qin=0,qfin=q1(t),sol,T=Tmax,base_profile);
```

$$q1\_profile := \begin{cases} 0.05032483965\, t^2 & 0 \le t \le 1 \\ 0.1006496793 - 0.05032483965\, (t-2)^2 & 1 < t < 2 \end{cases} \tag{2.7.1.2}$$

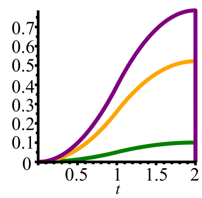
```
> q2_profile:=subs(qin=0,qfin=q2(t),sol,T=Tmax,base_profile);
```

$$q2\_profile := \begin{cases} 0.2611505518\, t^2 & 0 \le t \le 1 \\ 0.5223011036 - 0.2611505518\, (t-2)^2 & 1 < t < 2 \end{cases} \tag{2.7.1.3}$$

```
> q3_profile:=subs(qin=0,qfin=q3(t),sol,T=Tmax,base_profile);
```

$$q3\_profile := \begin{cases} 0.3929772068\, t^2 & 0 \le t \le 1 \\ 0.7859544135 - 0.3929772068\, (t-2)^2 & 1 < t < 2 \end{cases} \quad \textbf{(2.7.1.4)}$$

```
> display([
        plot(q1_profile,t=0..Tmax),
        plot(q2_profile,t=0..Tmax),
        plot(q3_profile,t=0..Tmax)
    ],
    color=["Green","Orange","Purple"],
    size=[150,150]
);
```



Given the joint accelerations a

Introduction and minimum total time

The minimum time requested by the system to perform the repositioning manouver corresponds to the maximum time takne by the slowest joint to perform its repositioning movement exploiting its maximum acceleration.

To calculate the minimum time for each joint variable it is sufficient to apply the formula obtained reworking the base_profile time history. The formula is the following:

```
> Tmin:=sqrt(4*abs(deltaq)/amax);
```

$$Tmin := 2\sqrt{\frac{|deltaq|}{amax}} \quad \textbf{(2.7.2.1)}$$

Given data:

```
> a1_max:=2: a2_max:=2: a3_max:=3:
```

Time taken by the 3 bodies

```
> T1_min:=evalf(subs(deltaq=qfin-qin,qfin=q1(t),qin=0,sol,amax=
  a1_max,Tmin)):
> T2_min:=evalf(subs(deltaq=qfin-qin,qfin=q2(t),qin=0,sol,amax=
  a1_max,Tmin)):
> T3_min:=evalf(subs(deltaq=qfin-qin,qfin=q3(t),qin=0,sol,amax=
  a1_max,Tmin)):
> "T1_min"=T1_min,"T2_min"=T2_min,"T3_min"=T3_min;
```

"T1_min" = 0.4486639706, "T2_min" = 1.022057829, "T3_min" = 1.253757882 **(2.7.2.2)**

```
> Ttot:=max([T1_min,T2_min,T3_min]);
```

$$Ttot := 1.253757882 \quad \textbf{(2.7.2.3)}$$

Repositioning time choice: the slowest one is the last, so we take as reference for Ttot its value and we rescaled the accelerations of the other joints in order to make them to finish at the same time.

Rescaling the accelerations: simply invert the previous formula.

```
> a_max_rescaled:=4*deltaq/T^2;
```

$$a\_max\_rescaled := \frac{4\, deltaq}{T^2} \quad \textbf{(2.7.2.4)}$$

```
> a1_rescaled:=evalf(subs(deltaq=qfin-qin,qfin=q1(t),qin=0,sol,T=
  Ttot,a_max_rescaled));
```

$$a1\_rescaled := 0.2561209088 \quad (2.7.2.5)$$

```
> a2_rescaled:=evalf(subs(deltaq=qfin-qin,qfin=q2(t),qin=0,sol,T=
  Ttot,a_max_rescaled));
```

$$a2\_rescaled := 1.329087526 \quad (2.7.2.6)$$

Plotting base_profile

Note: always check that the acceleration has the same direction of the minimum distance between the initial and final position. In other words: **pay attention to the sign of amax**!

Base profile given a

```
> base_profile_a:=piecewise(
      t>=0 and t<=T/2, qin+1/2*amax*t^2,
      t>T/2 and t<T, qin+amax*T^2/4-(T-t)^2*amax*1/2
  );
```

$$base\_profile\_a := \begin{cases} qin + \frac{amax\, t^2}{2} & 0 \le t \le \frac{T}{2} \\ qin + \frac{amax\, T^2}{4} - \frac{(T-t)^2\, amax}{2} & \frac{T}{2} < t < T \end{cases} \quad (2.7.2.7)$$

```
> q1_profile_a:=subs(amax=a1_rescaled,T=Ttot,qin=0,base_profile_a);
```

$$q1\_profile\_a := \begin{cases} 0.1280604544\, t^2 & 0 \le t \le 0.6268789410 \\ 0.1006496793 - 0.1280604544\,(1.253757882 - t)^2 & 0.6268789410 < t < 1.253757882 \end{cases}$$

```
> q2_profile_a:=subs(amax=a2_rescaled,T=Ttot,qin=0,base_profile_a);
```

$$q2\_profile\_a := \begin{cases} 0.6645437630\, t^2 & 0 \le t \le 0.6268789410 \\ 0.5223011035 - 0.6645437630\,(1.253757882 - t)^2 & 0.6268789410 < t < 1.253757882 \end{cases}$$

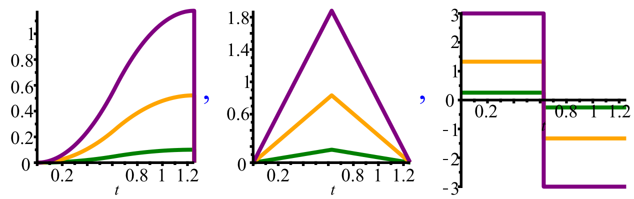
```
> q3_profile_a:=subs(amax=a3_max,T=Ttot,qin=0,base_profile_a);
```

$$q3\_profile\_a := \quad (2.7.2.10)$$

$$\begin{cases} \frac{3\, t^2}{2} & 0 \le t \le 0.6268789410 \\ 1.178931620 - \frac{3\,(1.253757882 - t)^2}{2} & 0.6268789410 < t < 1.253757882 \end{cases}$$

```
> display([
          plot(q1_profile_a,t=0..Ttot),
          plot(q2_profile_a,t=0..Ttot),
          plot(q3_profile_a,t=0..Ttot)
      ],
```

```
        color=["Green","Orange","Purple"]
    ),
    display([
            plot(diff(q1_profile_a,t),t=0..Ttot),
            plot(diff(q2_profile_a,t),t=0..Ttot),
            plot(diff(q3_profile_a,t),t=0..Ttot)
        ],
        color=["Green","Orange","Purple"]
    ),
    display([
            plot(diff(q1_profile_a,t,t),t=0..Ttot),
            plot(diff(q2_profile_a,t,t),t=0..Ttot),
            plot(diff(q3_profile_a,t,t),t=0..Ttot)
        ],
        color=["Green","Orange","Purple"],
        size=[150,150]
    );
```



With velocity saturation

```
> a1_max:=3: a2_max:=2: a3_max:=1:
> v1_max:=1: v2_max:=1.5: v3_max:=1:
```

Given the joint accelerations (given a, get Tmin_sat)

Minimum time and saturated velocity as soon as possible

```
> threshold_s:=vmax^2/2*(a+d)/(a*d);
```

$$threshold\_s := \frac{vmax^2\,(a+d)}{2\,a\,d} \qquad (2.7.3.1.1)$$

```
> delta_t_under_threshold:=sqrt(2*delta_S*(a+d)/(a*d));
```

$$delta\_t\_under\_threshold := \sqrt{2}\,\sqrt{\frac{delta\_S\,(a+d)}{a\,d}} \qquad (2.7.3.1.2)$$

```
> delta_t_over_threshold:=delta_S/vm+(a+d)/(2*a*d)*vm;
```

$$delta\_t\_over\_threshold := \frac{delta\_S}{vm} + \frac{(a+d)\,vm}{2\,a\,d} \qquad (2.7.3.1.3)$$

If a=d

```
> threshold_s_simplified:=vmax^2/a;
```

$$threshold\_s\_simplified := \frac{vmax^2}{a} \qquad (2.7.3.1.4)$$

```
> delta_t_under_threshold_simplified:=2*sqrt(delta_S/a);
```

$$delta\_t\_under\_threshold\_simplified := 2\,\sqrt{\frac{delta\_S}{a}} \qquad (2.7.3.1.5)$$

```
> delta_t_over_threshold_simplified:=delta_S/vm+vm/a;
```

$$delta\_t\_over\_threshold\_simplified := \frac{delta\_S}{vm} + \frac{vm}{a} \qquad (2.7.3.1.6)$$

Joint 1:

```
> threshold_s_1:=evalf(subs(a=a1_max,d=a1_max,vmax=v1_max,
  threshold_s));
```

$$threshold\_s\_1 := 0.3333333333 \tag{2.7.3.1.7}$$

```
> threshold_s_1_simplified:=evalf(subs(a=a1_max,vmax=v1_max,
  threshold_s_simplified));
```

$$threshold\_s\_1\_simplified := 0.3333333333 \tag{2.7.3.1.8}$$

```
> delta_s_1:=subs(qfin=q1(t),sol,qin=0,qfin-qin): <%>,
  "threshold_s_1"=threshold_s_1;
```

$$\left[\begin{array}{c} 0.1006496793 \end{array}\right], \text{"threshold\_s\_1"} = 0.3333333333 \tag{2.7.3.1.9}$$

```
> delta_t_1:=evalf(subs(a=a1_max,d=a1_max,delta_S=delta_s_1,
  delta_t_under_threshold));
```

$$delta\_t\_1 := 0.3663325981 \tag{2.7.3.1.10}$$

```
> delta_t_1_simplified:=evalf(subs(a=a1_max,vmax=v1_max,delta_S=
  delta_s_1,delta_t_under_threshold_simplified));
```

$$delta\_t\_1\_simplified := 0.3663325981 \tag{2.7.3.1.11}$$

Joint 2:

```
> threshold_s_2:=evalf(subs(a=a2_max,d=a2_max,vmax=v2_max,
  threshold_s));
```

$$threshold\_s\_2 := 1.125000000 \tag{2.7.3.1.12}$$

```
> threshold_s_2_simplified:=evalf(subs(a=a2_max,vmax=v2_max,
  threshold_s_simplified));
```

$$threshold\_s\_2\_simplified := 1.125000000 \tag{2.7.3.1.13}$$

```
> delta_s_2:=subs(qfin=q2(t),sol,qin=0,qfin-qin): <%>,
  "threshold_s_2"=threshold_s_2;
```

$$\left[\begin{array}{c} 0.5223011036 \end{array}\right], \text{"threshold\_s\_2"} = 1.125000000 \tag{2.7.3.1.14}$$

```
> delta_t_2:=evalf(subs(a=a2_max,d=a2_max,delta_S=delta_s_2,
  delta_t_under_threshold));
```

$$delta\_t\_2 := 1.022057829 \tag{2.7.3.1.15}$$

```
> delta_t_2_simplified:=evalf(subs(a=a2_max,vmax=v2_max,delta_S=
  delta_s_2,delta_t_under_threshold_simplified));
```

$$delta\_t\_2\_simplified := 1.022057829 \tag{2.7.3.1.16}$$

Joint 3:

```
> threshold_s_3:=evalf(subs(a=a3_max,d=a3_max,vmax=v3_max,
  threshold_s));
```

$$threshold\_s\_3 := 1. \tag{2.7.3.1.17}$$

```
> threshold_s_3_simplified:=evalf(subs(a=a3_max,vmax=v1_max,
  threshold_s_simplified));
```

$$threshold\_s\_3\_simplified := 1. \tag{2.7.3.1.18}$$

```
> delta_s_3:=subs(qfin=q3(t),sol,qin=0,qfin-qin): <%>,
  "threshold_s_3"=threshold_s_3;
```

$$\left[\begin{array}{c} 0.7859544135 \end{array}\right], \text{"threshold\_s\_3"} = 1. \tag{2.7.3.1.19}$$

```
> delta_t_3:=evalf(subs(a=a3_max,d=a3_max,delta_S=delta_s_3,
  delta_t_under_threshold));
```

$$delta\_t\_3 := 1.773081401 \tag{2.7.3.1.20}$$

```
> delta_t_3_simplified:=evalf(subs(a=a3_max,vmax=v3_max,delta_S=
  delta_s_3,delta_t_under_threshold_simplified));
```

$$delta\_t\_3\_simplified := 1.773081401 \quad \textbf{(2.7.3.1.21)}$$

Minimum repositioning time for the manouvre:

```
> T_min_sat:=max([delta_t_1,delta_t_2,delta_t_3]);
```

$$T\_min\_sat := 1.773081401 \quad \textbf{(2.7.3.1.22)}$$

Motion profiles (a and d the same)

```
> base_profile_v_sat:=piecewise(
      t>=0 and t<=t1, qin+1/2*amax*t^2,
      t>t1 and t<t2, qin+1/2*amax*t1^2+v*(t-t1),
      t>t2 and t<T, qin+1/2*amax*t1^2+v*(t-t1)-1/2*amax*(t-t2)^2
  );
```

$$base\_profile\_v\_sat := \quad \textbf{(2.7.3.1.23)}$$

$$\begin{cases} qin + \dfrac{amax\, t^2}{2} & 0 \le t \le t1 \\[2ex] qin + \dfrac{amax\, t1^2}{2} + v\,(-t1 + t) & t1 < t < t2 \\[2ex] qin + \dfrac{amax\, t1^2}{2} + v\,(-t1 + t) - \dfrac{amax\,(-t2 + t)^2}{2} & t2 < t < T \end{cases}$$

Body 1

```
> v_1:=rhs(op(solve(
      subs(delta_T=T_min_sat,amax=a1_max,delta_S=qfin-qin,qfin=q1
  (t),qin=0,sol,
      [delta_T=delta_S/vm+vm/amax]),vm
  )[1]));
```

$$v\_1 := 0.05738447114 \quad \textbf{(2.7.3.1.24)}$$

```
> t1_1_a_sat:=v_1/a1_max: t2_1_a_sat:=T_min_sat-t1_1_a_sat: "t1"=
  <%%>,"t2"=<%>;
```

$$\text{"t1"} = \left[\ 0.01912815705\ \right], \text{"t2"} = \left[\ 1.753953244\ \right] \quad \textbf{(2.7.3.1.25)}$$

```
> q1_profile_a_sat:=subs(amax=a1_max,
      t1=t1_1_a_sat,t2=t2_1_a_sat,T=T_min_sat,
      v=v_1,qin=0,
      base_profile_v_sat);
```

$$q1\_profile\_a\_sat := \begin{cases} \dfrac{3\,t^2}{2} & 0 \le t \le 0. \\[2ex] -0.0005488295878 + 0.05738447114\, t & 0.01912815705 \\[2ex] -0.0005488295878 + 0.05738447114\, t - \dfrac{3\,(-1.753953244 + t)^2}{2} & 1.753953244 < \end{cases}$$

Body 2

```
> v_2:=rhs(op(solve(
      subs(delta_T=T_min_sat,amax=a2_max,delta_S=qfin-qin,qfin=q2
  (t),qin=0,sol,
      [delta_T=delta_S/vm+vm/amax]),vm
  )[1]));
```

$$v\_2 := 0.3242144480 \quad \textbf{(2.7.3.1.27)}$$

```
> t1_2_a_sat:=v_2/a2_max: t2_2_a_sat:=T_min_sat-t1_2_a_sat: "t1"=
  <%%>,"t2"=<%>;
```

$$"t1" = \left[\ 0.1621072240\ \right], "t2" = \left[\ 1.610974177\ \right] \quad \textbf{(2.7.3.1.28)}$$

```
> q2_profile_a_sat:=subs(amax=a2_max,
      t1=t1_2_a_sat,t2=t2_2_a_sat,T=T_min_sat,
      v=v_2,qin=0,
      base_profile_v_sat);
```

$$q2\_profile\_a\_sat := \begin{cases} t^2 & 0 \le t \le 0.162107 \\ -0.02627875208 + 0.3242144480\, t & 0.1621072240 < t < 1. \\ -0.02627875208 + 0.3242144480\, t - (-1.610974177 + t)^2 & 1.610974177 < t < 1.7 \end{cases}$$

Body 3 (simply base_profile as it does not saturate)

```
> q3_profile_a_sat:=subs(amax=a3_max,T=T_min_sat,qin=0,
  base_profile_a);
```

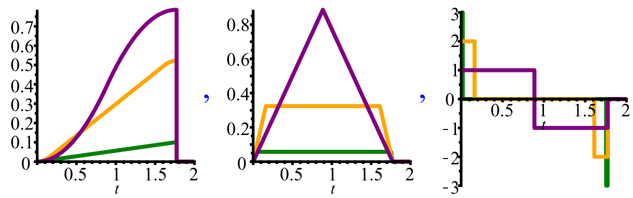
$$q3\_profile\_a\_sat := \quad \textbf{(2.7.3.1.30)}$$

$$\begin{cases} \frac{t^2}{2} & 0 \le t \le 0.8865407005 \\ 0.7859544138 - \frac{(1.773081401 - t)^2}{2} & 0.8865407005 < t < 1.773081401 \end{cases}$$

Display (given a, get Tmin_sat)

```
> display([
          plot(q1_profile_a_sat,t=0..2),
          plot(q2_profile_a_sat,t=0..2),
          plot(q3_profile_a_sat,t=0..2)
      ],
      color=["Green","Orange","Purple"]
  ),
  display([
          plot(diff(q1_profile_a_sat,t),t=0..2),
          plot(diff(q2_profile_a_sat,t),t=0..2),
          plot(diff(q3_profile_a_sat,t),t=0..2)
      ],
      color=["Green","Orange","Purple"]
  ),
  display([
```

```
            plot(diff(q1_profile_a_sat,t,t),t=0..2),
            plot(diff(q2_profile_a_sat,t,t),t=0..2),
            plot(diff(q3_profile_a_sat,t,t),t=0..2)
        ],
        color=["Green","Orange","Purple"],
        size=[150,150]
    );
```




```
> sol;
```

$$[q1(t) = 0.1006496793, q2(t) = 0.5223011036, q3(t) = 0.7859544135] \quad \textbf{(2.7.3.1.31)}$$

Given repositioning time (given Ttot_sat)

Saturated velocity reached as soon as possible and manouvre time fixed by Ttot_sat

```
> Ttot_sat:=2;
```

$$Ttot\_sat := 2 \quad \textbf{(2.7.3.2.1)}$$

Same as before, with **a=d**

```
> base_profile_v_sat:=piecewise(
        t>=0 and t<=t1, qin+1/2*amax*t^2,
        t>t1 and t<t2, qin+1/2*amax*t1^2+v*(t-t1),
        t>t2 and t<T, qin+1/2*amax*t1^2+v*(t-t1)-1/2*amax*(t-t2)^2
    );
```

$$base\_profile\_v\_sat := \quad \textbf{(2.7.3.2.2)}$$

$$\begin{cases} qin + \dfrac{amax\, t^2}{2} & 0 \le t \le t1 \\ qin + \dfrac{amax\, t1^2}{2} + v\,(-t1 + t) & t1 < t < t2 \\ qin + \dfrac{amax\, t1^2}{2} + v\,(-t1 + t) - \dfrac{amax\,(-t2 + t)^2}{2} & t2 < t < T \end{cases}$$

Body 1

```
> vel_1:=rhs(op(solve(
        subs(delta_T=Ttot_sat,amax=a1_max,delta_S=qfin-qin,qfin=q1
    (t),qin=0,sol,
        [delta_T=delta_S/vm+vm/amax]),vm
    )[1]));
```

$$vel\_1 := 0.05075417062 \quad \textbf{(2.7.3.2.3)}$$

```
> t1_1_t_sat:=vel_1/a1_max: t2_1_t_sat:=Ttot_sat-t1_1_a_sat: "t1"=
<%%>,"t2"=<%>;
```

$$\text{"t1"} = \left[\ 0.01691805687\ \right], \text{"t2"} = \left[\ 1.980871843\ \right] \quad \textbf{(2.7.3.2.4)}$$

```
> q1_profile_t_sat:=subs(amax=a1_max,
        t1=t1_1_t_sat,t2=t2_1_t_sat,T=Ttot_sat,
        v=vel_1,qin=0,
        base_profile_v_sat);
```

$$q1\_profile\_t\_sat := \begin{cases} \frac{3\,t^2}{2} & 0 \le t \le 0.0\ldots \\ -0.0004293309725 + 0.05075417062\,t & 0.01691805687 <\ldots \\ -0.0004293309725 + 0.05075417062\,t - \frac{3\,(-1.980871843 + t)^2}{2} & 1.9808718\ldots \end{cases}$$

Body 2

```
> vel_2:=rhs(op(solve(
      subs(delta_T=Ttot_sat,amax=a2_max,delta_S=qfin-qin,qfin=q2
  (t),qin=0,sol,
      [delta_T=delta_S/vm+vm/amax]),vm
  )[1]));
```

$$vel\_2 := 0.2808729562 \quad \textbf{(2.7.3.2.6)}$$

```
> t1_2_t_sat:=vel_1/a1_max: t2_2_t_sat:=Ttot_sat-t1_2_a_sat: "t1"=
  <%%>,"t2"=<%>;
```

$$\text{"t1"} = \left[\ 0.01691805687\ \right], \text{"t2"} = \left[\ 1.837892776\ \right] \quad \textbf{(2.7.3.2.7)}$$

```
> q2_profile_t_sat:=subs(amax=a2_max,
      t1=t1_1_t_sat,t2=t2_1_t_sat,T=Ttot_sat,
      v=vel_2,qin=0,
      base_profile_v_sat);
```

$$q2\_profile\_t\_sat := \begin{cases} t^2 & 0 \le t \le 0.01691\ldots \\ -0.004465603998 + 0.2808729562\,t & 0.01691805687 < t <\ldots \\ -0.004465603998 + 0.2808729562\,t - (-1.980871843 + t)^2 & 1.980871843 <\ldots \end{cases}$$

Body 3

```
> vel_3:=rhs(op(solve(
      subs(delta_T=Ttot_sat,amax=a3_max,delta_S=qfin-qin,qfin=q3
  (t),qin=0,sol,
      [delta_T=delta_S/vm+vm/amax]),vm
  )[1]));
```

$$vel\_3 := 0.5373493905 \quad \textbf{(2.7.3.2.9)}$$

```
> t1_3_t_sat:=vel_3/a3_max: t2_3_t_sat:=Ttot_sat-t1_3_t_sat: "t1"=
  <%%>,"t2"=<%>;
```

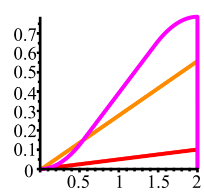
$$\text{"t1"} = \left[\ 0.5373493905\ \right], \text{"t2"} = \left[\ 1.462650610\ \right] \quad \textbf{(2.7.3.2.10)}$$

```
> q3_profile_t_sat:=subs(amax=a3_max,
     t1=t1_3_t_sat,t2=t2_3_t_sat,T=Ttot_sat,
     v=vel_3,qin=0,
     base_profile_v_sat);
```

$$q3\_profile\_t\_sat := \begin{cases} \frac{t^2}{2} & 0 \le t \le 0.537349 \\ -0.1443721837 + 0.5373493905\, t & 0.5373493905 < t < 1. \\ -0.1443721837 + 0.5373493905\, t - \frac{(-1.462650610 + t)^2}{2} & 1.462650610 < t \end{cases}$$

Display (given Ttot_sat)

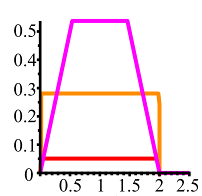
```
> display([
        plot(q1_profile_t_sat,t=0..Ttot_sat),
        plot(q2_profile_t_sat,t=0..Ttot_sat),
        plot(q3_profile_t_sat,t=0..Ttot_sat)
     ],
     color=["Red","DarkOrange","Magenta"],
     size=[150,150]
  );
```




```
> sol;
```

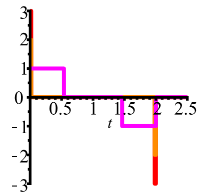
$$[q1(t) = 0.1006496793, q2(t) = 0.5223011036, q3(t) = 0.7859544135] \quad \textbf{(2.7.3.2.12)}$$

```
> display([
        plot(diff(q1_profile_t_sat,t),t=0..Ttot_sat+0.5),
        plot(diff(q2_profile_t_sat,t),t=0..Ttot_sat+0.5),
        plot(diff(q3_profile_t_sat,t),t=0..Ttot_sat+0.5)
     ],
     color=["Red","DarkOrange","Magenta"],
     size=[150,150]
  );
```




```
> display([
        plot(diff(q1_profile_t_sat,t,t),t=0..Ttot_sat+0.5),
        plot(diff(q2_profile_t_sat,t,t),t=0..Ttot_sat+0.5),
```

```
        plot(diff(q3_profile_t_sat,t,t),t=0..Ttot_sat+0.5)
    ],
    color=["Red","DarkOrange","Magenta"],
    size=[150,150]
);
```



Dynamic

The objective here is to calculate the equations of motion through the Lageangian approach.

Velocity matrices

I put another time this chapter because usually we need it in the dynamic.
To see it entirely see the kinematic part

```
> W_func:=(M)->simplify(map(diff,M,t).MatrixInverse(M));
```

$$W\_func := M \mapsto simplify(map(VectorCalculus{:-}diff, M, t) \cdot LinearAlgebra{:-}MatrixInverse(M)) \tag{3.1.1}$$

Body 1

```
> W01:=W_func(M01):
> L01:=Ltraslz: W01:=L01*diff(q1(t),t):
```

Body 2

```
> W02:=W_func(M02);
```

$$W02 := \begin{bmatrix} 0 & 0 & -\frac{d}{dt}q2(t) & \left(\frac{d}{dt}q2(t)\right)q1(t) \\ 0 & 0 & 0 & 0 \\ \frac{d}{dt}q2(t) & 0 & 0 & -L1\left(\frac{d}{dt}q2(t)\right)+\frac{d}{dt}q1(t) \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.1.2}$$

```
> L12_1:=Lrotz:                           # definition
  L12:=M01.L12_1.MatrixInverse(M01):      # change of rf
  W12:=simplify(L12*diff(q2(t),t)):       # definition
  W02:=simplify(W01+W12): "W02"=W02;      # Rival's theorem
```

$$"W02" = \begin{bmatrix} 0 & 0 & -\frac{d}{dt}q2(t) & \left(\frac{d}{dt}q2(t)\right)q1(t) \\ 0 & 0 & 0 & 0 \\ \frac{d}{dt}q2(t) & 0 & 0 & -L1\left(\frac{d}{dt}q2(t)\right)+\frac{d}{dt}q1(t) \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.1.3}$$

Body 3

```
> W03:=W_func(M03);
```

$$W03 := \left[\left[0, -\left(\frac{d}{dt}q3(t)\right)\cos(q2(t)), -\frac{d}{dt}q2(t), \left(\frac{d}{dt}q2(t)\right)q1(t)\right], \left[\left(\frac{d}{dt}q3(t)\right)\cos(q2(t)), 0, \left(\frac{d}{dt}q3(t)\right)\sin(q2(t)), -\left(\frac{d}{dt}q3(t)\right)(L1\cos(q2(t)) + \sin(q2(t))\,q1(t) + L2)\right], \tag{3.1.4}$$

$$\left[\frac{d}{dt}\, q2(t),\ -\left(\frac{d}{dt}\, q3(t)\right) \sin(q2(t)),\ 0,\ -L1 \left(\frac{d}{dt}\, q2(t)\right) + \frac{d}{dt}\, q1(t)\right],$$

$$\left[0,\ 0,\ 0,\ 0\right]\Big]$$

```
> L23_2:=Lrotz:                              # definition
  L23:=M02.L23_2.MatrixInverse(M02):         # change of rf
  W23:=simplify(L23.diff(q3(t),t)):          # definition
  W03:=simplify(W02+W23);
```

$$W03 := \Big[\left[0,\ -\left(\frac{d}{dt}\, q3(t)\right) \cos(q2(t)),\ -\frac{d}{dt}\, q2(t),\ \left(\frac{d}{dt}\, q2(t)\right) q1(t)\right], \tag{3.1.5}$$

$$\left[\left(\frac{d}{dt}\, q3(t)\right) \cos(q2(t)),\ 0,\ \left(\frac{d}{dt}\, q3(t)\right) \sin(q2(t)),\ -\left(\frac{d}{dt}\, q3(t)\right) (L1 \cos(q2(t)) + \sin(q2(t))\, q1(t) + L2)\right],$$

$$\left[\frac{d}{dt}\, q2(t),\ -\left(\frac{d}{dt}\, q3(t)\right) \sin(q2(t)),\ 0,\ -L1 \left(\frac{d}{dt}\, q2(t)\right) + \frac{d}{dt}\, q1(t)\right],$$

$$\left[0,\ 0,\ 0,\ 0\right]\Big]$$

Pseudo inertial tensor

Depending on the mass distribution, more ways to calculate it.

The general form is:

```
> J_local_frame:=<<Ixx,Ixy,Ixz,mi*xGii>|<Iyx,Iyy,Iyz,mi*yGi>|<Izx,
  Izy,Izz,mi*zGi>|<mi*xGi,mi*yGi,mi*zGi,mi>>;
```

$$J\_local\_frame := \begin{bmatrix} Ixx & Iyx & Izx & mi\, xGi \\ Ixy & Iyy & Izy & mi\, yGi \\ Ixz & Iyz & Izz & mi\, zGi \\ mi\, xGii & mi\, yGi & mi\, zGi & mi \end{bmatrix} \tag{3.2.1}$$

Note: as long as its local, also the coordinates of G has to be defined locally

Masses concentrated in a single point

```
> Ixx_lumped:=m*L^2/4;
```

$$Ixx\_lumped := \frac{m\, L^2}{4} \tag{3.2.1.1}$$

```
> J_lumped_func:=(G,m)->G.Transpose(G)*m;
```

$$J\_lumped\_func := (G, m) \mapsto (G \cdot G^{+}) \cdot m \tag{3.2.1.2}$$

Example:

```
> J_lumped_func(<a,b,c,1>,m);
```

(3.2.1.3)

$$\begin{bmatrix} m a^2 & m a b & m a c & m a \\ m a b & m b^2 & m b c & m b \\ m a c & m b c & m c^2 & m c \\ m a & m b & m c & m \end{bmatrix} \quad \textbf{(3.2.1.3)}$$

Masses distributed along an axis

We can also have the C.o.M. out of a specific axis but always along a single one.
I_distributed has: Ixx, Iyy, Izz

Given the mass

```
> Ixx_distributed:=m*L^2/3;
```

$$Ixx\_distributed := \frac{m L^2}{3} \quad \textbf{(3.2.2.1.1)}$$

Given the density

```
> Ixx_distributed:=int(rho*x^2,x=-L1..0);
```

$$Ixx\_distributed := \frac{\rho L1^3}{3} \quad \textbf{(3.2.2.2.1)}$$

```
> I_distributed_func:=(len,m)->m*(len^2)/3;
```

$$I\_distributed\_func := (len, m) \mapsto m \cdot len^2 \cdot \left(\frac{1}{3}\right) \quad \textbf{(3.2.2.1)}$$

Example:

```
> Jii:=<
      <I_distributed_func(Li,mi),0,0,Gii[1]*mi>|
      <0,0,0,Gii[2]*mi>|
      <0,0,0,Gii[3]*mi>|
      <Gii[1]*mi,Gii[2]*mi,Gii[3]*m,m>
  >;
```

$$Jii := \begin{bmatrix} \frac{mi\, \mathrm{Li}^2}{3} & 0 & 0 & Gii_1\, mi \\ 0 & 0 & 0 & Gii_2\, mi \\ 0 & 0 & 0 & Gii_3\, m \\ Gii_1\, mi & Gii_2\, mi & Gii_3\, mi & m \end{bmatrix} \quad \textbf{(3.2.2.2)}$$

Negleting the moment of inertia

If the text says that it means to define Jii as follows:

```
> Jii:=<<0,0,0,m*xG>|<0,0,0,m*yG>|<0,0,0,m*zG>|<m*xG,m*yG,m*zG,m>>;
```

$$Jii := \begin{bmatrix} 0 & 0 & 0 & m\, xG \\ 0 & 0 & 0 & m\, yG \\ 0 & 0 & 0 & m\, zG \\ m\, xG & m\, yG & m\, zG & m \end{bmatrix} \quad \textbf{(3.2.3.1)}$$

Solving the exercise:

Projected locally

```
> J11:=J_lumped_func(G11,m1):
> J22:=J_lumped_func(G22,m2):
> J33:=J_lumped_func(G33,m3):
> "J11"=J11,"J22"=J22,"J33"=J33;
```

$$"J11" = \begin{bmatrix} \frac{m1\,L1^2}{4} & 0 & 0 & -\frac{m1\,L1}{2} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -\frac{m1\,L1}{2} & 0 & 0 & m1 \end{bmatrix}, "J22" = \begin{bmatrix} \frac{m2\,L2^2}{4} & 0 & 0 & -\frac{m2\,L2}{2} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -\frac{m2\,L2}{2} & 0 & 0 & m2 \end{bmatrix}, "J33" \tag{3.2.4.1}$$

$$= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & m3 \end{bmatrix}$$

Projected in the fixed reference frame

```
> J_change_ref_func:=(M,J)->simplify(M.J.Transpose(M));
```

$$J\_change\_ref\_func := (M, J) \mapsto simplify(M \cdot J \cdot M^{+}) \tag{3.2.5.1}$$

```
> J10:=J_change_ref_func(M01,J11):
> J20:=J_change_ref_func(M02,J22):
> J30:=J_change_ref_func(M03,J33):
> "J10"=J10,"J20"=J20,"J30"=...;
```

$$"J10" = \begin{bmatrix} \frac{m1\,L1^2}{4} & 0 & \frac{q1(t)\,m1\,L1}{2} & \frac{m1\,L1}{2} \\ 0 & 0 & 0 & 0 \\ \frac{q1(t)\,m1\,L1}{2} & 0 & q1(t)^2\,m1 & q1(t)\,m1 \\ \frac{m1\,L1}{2} & 0 & q1(t)\,m1 & m1 \end{bmatrix}, "J20" = \Big[\tag{3.2.5.2}$$

$$\left[\frac{m2\,(\cos(q2(t))\,L2 + 2\,L1)^2}{4}, 0, \right.$$

$$\frac{m2\,(\sin(q2(t))\,L2 + 2\,q1(t))\,(\cos(q2(t))\,L2 + 2\,L1)}{4}, \frac{m2\,(\cos(q2(t))\,L2 + 2\,L1)}{2}$$

$$\Big],$$

$$\Big[0, 0, 0, 0\Big],$$

$$\left[\frac{m2\,(\sin(q2(t))\,L2+2\,q1(t))\,(\cos(q2(t))\,L2+2\,L1)}{4}, 0, \frac{m2\,(\sin(q2(t))\,L2+2\,q1(t))^2}{4}, \frac{m2\,(\sin(q2(t))\,L2+2\,q1(t))}{2}\right],$$
$$\left[\frac{m2\,(\cos(q2(t))\,L2+2\,L1)}{2}, 0, \frac{m2\,(\sin(q2(t))\,L2+2\,q1(t))}{2}, m2\right]\Bigg], \text{"J30"} =..$$

Just a check: the last is the same as calculating..
Pay attention to the last element depending on how J_lumped_func has been defined

```
> J30:=J_lumped_func(E,m3);
```

$$J30 := [[(\cos(q2(t))\cos(q3(t))\,L3+\cos(q2(t))\,L2+L1)^2\,m3,\ (\cos(q2(t))\cos(q3(t))\,L3+\cos(q2(t))\,L2+L1)\,m3\sin(q3(t))\,L3,\ (\cos(q2(t))\cos(q3(t))\,L3+\cos(q2(t))\,L2+L1)\,m3\,(\sin(q2(t))\cos(q3(t))\,L3+\sin(q2(t))\,L2+q1(t)),\ (\cos(q2(t))\cos(q3(t))\,L3+\cos(q2(t))\,L2+L1)\,m3],$$
$$[(\cos(q2(t))\cos(q3(t))\,L3+\cos(q2(t))\,L2+L1)\,m3\sin(q3(t))\,L3,\ \sin(q3(t))^2\,L3^2\,m3,\ \sin(q3(t))\,L3\,m3\,(\sin(q2(t))\cos(q3(t))\,L3+\sin(q2(t))\,L2+q1(t)),\ \sin(q3(t))\,L3\,m3],$$
$$[(\cos(q2(t))\cos(q3(t))\,L3+\cos(q2(t))\,L2+L1)\,m3\,(\sin(q2(t))\cos(q3(t))\,L3+\sin(q2(t))\,L2+q1(t)),\ \sin(q3(t))\,L3\,m3\,(\sin(q2(t))\cos(q3(t))\,L3+\sin(q2(t))\,L2+q1(t)),\ (\sin(q2(t))\cos(q3(t))\,L3+\sin(q2(t))\,L2+q1(t))^2\,m3,\ (\sin(q2(t))\cos(q3(t))\,L3+\sin(q2(t))\,L2+q1(t))\,m3],$$
$$[(\cos(q2(t))\cos(q3(t))\,L3+\cos(q2(t))\,L2+L1)\,m3,\ \sin(q3(t))\,L3\,m3,\ (\sin(q2(t))\cos(q3(t))\,L3+\sin(q2(t))\,L2+q1(t))\,m3,\ m3]] \quad \textbf{(3.2.5.3)}$$

Kinetic energy

```
> T_func:=(W,J)->simplify(1/2*Trace(W.J.Transpose(W)));
```

$$T\_func := (W, J) \mapsto simplify\left(\frac{1}{2}\cdot LinearAlgebra:-Trace(W\bullet J\bullet W^{+})\right) \quad \textbf{(3.3.1)}$$

```
> T1:=T_func(W01,J10);
```

$$T1 := \frac{\left(\frac{d}{dt}\,q1(t)\right)^2 m1}{2} \quad \textbf{(3.3.2)}$$

```
> T2:=T_func(W02,J20);
```

$$T2 := \frac{1}{8}\left(m2\left(4\left(\frac{d}{dt}\,q1(t)\right)^2+L2^2\left(\frac{d}{dt}\,q2(t)\right)^2+4\left(\frac{d}{dt}\,q2(t)\right)L2\cos(q2(t))\left(\frac{d}{dt}\,q1(t)\right)\right)\right) \quad \textbf{(3.3.3)}$$

```
> T3:=T_func(W03,J30): # more complex..
```

Potential energy

Neglicible

As the gravity force can be neglcted, there is not potential energy.

Gravity

```
> U_func:=(H,J)->simplify(Trace(-H.J));
```

$$U\_func := (H, J) \mapsto simplify(LinearAlgebra{:-}Trace(-H \cdot J)) \tag{3.4.2.1}$$

```
> Hg:=<<0,0,0,0>|<0,0,0,0>|<0,0,0,0>|<0,0,-g,0>>;
```

$$Hg := \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -g \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.4.2.2}$$

```
> Hg1:=Hg: Hg2:=Hg: Hg3:=Hg:
> Ug1:=U_func(Hg1,J10);
```

$$Ug1 := g\, q1(t)\, m1 \tag{3.4.2.3}$$

```
> Ug2:=U_func(Hg2,J20);
```

$$Ug2 := \frac{g\, m2\, (\sin(q2(t))\, L2 + 2\, q1(t))}{2} \tag{3.4.2.4}$$

```
> Ug3:=U_func(Hg3,J30);
```

$$Ug3 := m3\, (\sin(q2(t))\, (\cos(q3(t))\, L3 + L2) + q1(t))\, g \tag{3.4.2.5}$$

Elastic

```
> Uel:=1/2*Ktheta*(thetam(t)-theta(t))^2;
```

$$Uel := \frac{Ktheta\, (thetam(t) - \theta(t))^2}{2} \tag{3.4.3.1}$$

Non lagrangian component

Skip if the only forces considered in the problem are those coming from actuators.
This is because in that case it is simpler to just put the actuator' forces unaltered on the right hand side of the equation of the related generalised variable.
If this is not the case (there are additional forces), the procedure is the following.

We use the general formula for serial robot to calculate them.
The idea behind this formula is that all the actions applied to a robot matter for a joint variable as long as they perform work along it.
To consider only the action that perform work we assign a velocity different from zero to the considered joint variable while keeping the velocity of the others to zero.

Note:
- in it we collect all the forces and torques applied to the considered body
- each body has its own matrix of actions
- these actions have to be projected in the local frame (body 1, frame 1)
- remember that **forces causes torques**.
- torques are expressed w.r.t. the origin of the frame, the pole

- if defining the potential energy of a given force we can neglect it here (e.g. gravity)

How do we fill the matrix?

- considering the forces, simply project them on the x,y,z axes of the frame
- considering the torques, apply the formula to get the magnite: |c| = |r| |F| sin(theta)

Generic shape:

```
> Phi:=<<0,cz,-cy,-fx>|<-cz,0,cx,-fy>|<cy,-cx,0,-fz>|<fx,fy,fz,0>>;
```

$$\Phi := \begin{bmatrix} 0 & -cz & cy & fx \\ cz & 0 & -cx & fy \\ -cy & cx & 0 & fz \\ -fx & -fy & -fz & 0 \end{bmatrix} \quad (3.5.1)$$

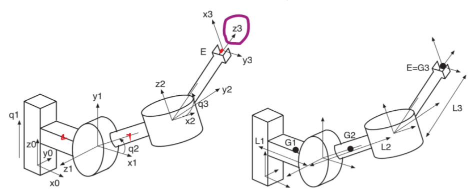
Only in this phase, I add the actuators' actions to show the result. Later I'll show directly how to include them without this passage.

Assume that we still have the force Fe used in the Kineto-Statics applied to the end effector:

```
> Transpose(Fe);
```

$$\begin{bmatrix} Fx & Fy & Fz \end{bmatrix} \quad (3.5.2)$$

Action matrices projected locally




```
> Phi11:=<<0,-F1*L1,0,0>|<F1*L1,0,0,-F1>|<0,0,0,0>|<0,F1,0,0>>:
> Phi22:=<<0,0,C2,0>|<0,0,0,0>|<-C2,0,0,0>|<0,0,0,0>>:
> Phi33:=<<0,0,0,-Fe[1]>|<0,0,C3,-Fe[2]>|<0,-C3,0,-Fe[3]>|<Fe[1],Fe
  [2],Fe[3],0>>:
> "Phi11"=Phi11,"Phi22"=Phi22,"Phi33"=Phi33,"Phi"=Phi;
```

$$\text{"Phi11"} = \begin{bmatrix} 0 & F1\,L1 & 0 & 0 \\ -F1\,L1 & 0 & 0 & F1 \\ 0 & 0 & 0 & 0 \\ 0 & -F1 & 0 & 0 \end{bmatrix}, \text{"Phi22"} = \begin{bmatrix} 0 & 0 & -C2 & 0 \\ 0 & 0 & 0 & 0 \\ C2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \text{"Phi33"} \quad (3.5.1.1)$$

$$= \begin{bmatrix} 0 & 0 & 0 & Fx \\ 0 & 0 & -C3 & Fy \\ 0 & C3 & 0 & Fz \\ -Fx & -Fy & -Fz & 0 \end{bmatrix}, \text{"Phi"} = \begin{bmatrix} 0 & -cz & cy & fx \\ cz & 0 & -cx & fy \\ -cy & cx & 0 & fz \\ -fx & -fy & -fz & 0 \end{bmatrix}$$

Action matrices projected in the fixed frame

```
> Phi10:=simplify(M01.Phi11.Transpose(M01)):
> Phi20:=simplify(M02.Phi22.Transpose(M02)):
> Phi30:=simplify(M03.Phi33.Transpose(M03)):
> "Phi10"=Phi10, "Phi20"=Phi20, "Phi30"=...
```

$$"Phi10" = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & F1 \\ 0 & 0 & -F1 & 0 \end{bmatrix}, "Phi20" = \begin{bmatrix} 0 & 0 & -C2 & 0 \\ 0 & 0 & 0 & 0 \\ C2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, "Phi30" =.. \tag{3.5.2.1}$$

Action projected in a frame a (on the end effector)

```
> Phiaa:=<<0,Cext,0,0>|<Cext,0,0,0>|<0,0,0,Fa>|<0,0,-Fa,0>>;
```

$$Phiaa := \begin{bmatrix} 0 & Cext & 0 & 0 \\ Cext & 0 & 0 & 0 \\ 0 & 0 & 0 & -Fa \\ 0 & 0 & Fa & 0 \end{bmatrix} \tag{3.5.3.1}$$

Remember, already defined

```
> M0a:=Mrotztrasl(0,E);
```

$$M0a := \begin{bmatrix} 1 & 0 & 0 & \cos(q2(t))\cos(q3(t))\,L3 + \cos(q2(t))\,L2 + L1 \\ 0 & 1 & 0 & \sin(q3(t))\,L3 \\ 0 & 0 & 1 & \sin(q2(t))\cos(q3(t))\,L3 + \sin(q2(t))\,L2 + q1(t) \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.5.3.2}$$

In fixed frame..

```
> Phia0:=M0a.Phi3aa.Transpose(M0a): "Phia0"=...
```

$$"Phia0" =.. \tag{3.5.3.3}$$

The matrix just obtained has to be summed to the other for calculating Qi.

Couples: (3,2) - (1,3) - (1,2) - (1,4) - (2,4) - (3,4) (page 13 personal notes)

```
> PSS:=(mat1,mat2)->
      mat1[3,2]*mat2[3,2]+mat1[1,3]*mat2[1,3]+mat1[1,2]*mat2[1,2]+
  # c_ij*L_ij
      mat1[1,4]*mat2[1,4]+mat1[2,4]*mat2[2,4]+mat1[3,4]*mat2[3,4];
  # f_ij*L_ij
```

$$PSS := (mat1, mat2) \mapsto mat1_{3,2} \cdot mat2_{3,2} + mat1_{1,3} \cdot mat2_{1,3} + mat1_{1,2} \cdot mat2_{1,2} + mat1_{1,4} \cdot mat2_{1,4} + mat1_{2,4} \cdot mat2_{2,4} + mat1_{3,4} \cdot mat2_{3,4} \tag{3.5.3}$$

The generic formula is: **Qi := sum_(i=1)^i Phi_i_0 PSS L_i_0**

```
> Q1:=PSS(Phi10+Phi20+Phi30,L01);
```

$$Q1 := F1 + \cos(q2(t))\,Fx - \sin(q2(t))\sin(q3(t))\,Fy - \sin(q2(t))\cos(q3(t))\,Fz \tag{3.5.4}$$

```
> Q2:=simplify(PSS(Phi20+Phi30,L12));
```

$$Q2 := L3\,Fx\cos(q3(t)) + Fx\,L2 + C2 \tag{3.5.5}$$

```
> Q3:=simplify(PSS(Phi30,L23));
```

$$Q3 := Fy\,L3 + C3 \tag{3.5.6}$$

As it can be seen, a force at the end effector produce work for any joint velocity.

Lagrange equation

```
> Lagr:=simplify(T1+T2+T3-Ug1-Ug2-Ug3);
```

$$Lagr := \frac{(4\,m1+4\,m2+4\,m3)\left(\frac{d}{dt}\,q1(t)\right)^2}{8} + \frac{1}{8}\Bigg(\Bigg(8\left(\cos(q3(t))\,L3\,m3 + \frac{L2\,(m2+2\,m3)}{2}\right)\cos(q2(t))\left(\frac{d}{dt}\,q2(t)\right) - 8\left(\frac{d}{dt}\,q3(t)\right)\sin(q2(t))\sin(q3(t))\,L3\,m3\Bigg)\left(\frac{d}{dt}\,q1(t)\right)\Bigg) + \frac{\left(4\cos(q3(t))^2\,L3^2\,m3 + 8\cos(q3(t))\,L2\,L3\,m3 + L2^2\,(m2+4\,m3)\right)\left(\frac{d}{dt}\,q2(t)\right)^2}{8} + \frac{m3\,L3^2\left(\frac{d}{dt}\,q3(t)\right)^2}{2} - \left(L3\cos(q3(t))\sin(q2(t))\,m3 + \frac{L2\,(m2+2\,m3)\sin(q2(t))}{2} + q1(t)\,(m1+m2+m3)\right)g \tag{3.6.1}$$

Equations of motion

Function introduced because in Maple it is not possible to perform derivatives w.r.t other functions.

```
> diffF:=(f,x)->subs(y=x,diff(subs(x=y,f),y));
```

$$diffF := (f, x) \rightarrow subs(y = x, VectorCalculus\text{:-}diff(subs(x = y, f), y)) \tag{3.7.1}$$

The **first term** is defined as: **d[**dLagr/(dtheta/dt)**]/dt** and represent the time derivative of the partial derivative of the Lagrande function w.r.t. the joint velocity.

The **second term** is: dLagr/dtheta and represents the partial derivative of the Lagrange function w.r.t a joint variable.

For the following equations of motion I show the shape of the equations substracting the non lagrange terms.
However, as the problem never mentioned the external force Fe on the end effector, I'll overwrite the equations putting directly the actuators' actions the same way we would have followed in the case of non lagrangian.

Body 1

```
> ddtddq1dot_Lagr:=diff(diffF(Lagr,diff(q1(t),t)),t);
```

$$ddtddq1dot\_Lagr := \frac{(4\,m1+4\,m2+4\,m3)\left(\frac{d^2}{dt^2}\,q1(t)\right)}{4} - 2\left(\frac{d}{dt}\,q3(t)\right)\sin(q3(t))\,L3\,m3\cos(q2(t))\left(\frac{d}{dt}\,q2(t)\right) - \left(\cos(q3(t))\,L3\,m3 + \frac{L2\,(m2+2\,m3)}{2}\right)\left(\frac{d}{dt}\,q2(t)\right)^2\sin(q2(t)) + \left(\cos(q3(t))\,L3\,m3 + \frac{L2\,(m2+2\,m3)}{2}\right)\cos(q2(t))\left(\frac{d^2}{dt^2}\,q2(t)\right) - \left(\frac{d^2}{dt^2}\right. \tag{3.7.1.1}$$

$$q3(t)\Bigg)\sin(q2(t))\sin(q3(t))\,L3\,m3-\left(\frac{d}{dt}\,q3(t)\right)^2\sin(q2(t))\cos(q3(t))\,L3\,m3$$

```
> ddq1_Lagr:=diffF(Lagr,q1(t));
```

$$ddq1\_Lagr := -(m1+m2+m3)\,g \qquad \textbf{(3.7.1.2)}$$

Hypotetic one

```
> eqn1:=simplify(ddtddq1dot_Lagr-ddq1_Lagr-Q1);
```

$$eqn1 := \frac{\left(\frac{d^2}{dt^2}\,q1(t)\right)(2\,m1+2\,m2+2\,m3)}{2} \qquad \textbf{(3.7.1.3)}$$

$$+\frac{(2\cos(q3(t))\,L3\,m3+L2\,(m2+2\,m3))\cos(q2(t))\left(\frac{d^2}{dt^2}\,q2(t)\right)}{2}-\Bigg(\frac{d^2}{dt^2}$$

$$q3(t)\Bigg)\sin(q2(t))\sin(q3(t))\,L3\,m3$$

$$-\frac{(2\cos(q3(t))\,L3\,m3+L2\,(m2+2\,m3))\left(\frac{d}{dt}\,q2(t)\right)^2\sin(q2(t))}{2}-2\,\Bigg(\frac{d}{dt}$$

$$q3(t)\Bigg)\sin(q3(t))\,L3\,m3\cos(q2(t))\left(\frac{d}{dt}\,q2(t)\right)-\Bigg(\frac{d}{dt}$$

$$q3(t)\Bigg)^2\sin(q2(t))\cos(q3(t))\,L3\,m3$$

$$+\frac{(2\cos(q3(t))\,Fz+2\sin(q3(t))\,Fy)\sin(q2(t))}{2}+m2\,g+m3\,g+m1\,g$$

$$-\cos(q2(t))\,Fx-F1$$

Used one

```
> eqn1:=simplify(ddtddq1dot_Lagr-ddq1_Lagr-F1);
```

$$eqn1 := \frac{\left(\frac{d^2}{dt^2}\,q1(t)\right)(2\,m1+2\,m2+2\,m3)}{2}+\Bigg(\cos(q3(t))\,L3\,m3 \qquad \textbf{(3.7.1.4)}$$

$$+\frac{L2\,(m2+2\,m3)}{2}\Bigg)\cos(q2(t))\left(\frac{d^2}{dt^2}\,q2(t)\right)-\Bigg(\frac{d^2}{dt^2}$$

$$q3(t)\Bigg)\sin(q2(t))\sin(q3(t))\,L3\,m3-\Bigg(\cos(q3(t))\,L3\,m3$$

$$+\frac{L2\,(m2+2\,m3)}{2}\Bigg)\left(\frac{d}{dt}\,q2(t)\right)^2\sin(q2(t))-2\,\Bigg(\frac{d}{dt}$$

$$q3(t)\Bigg)\sin(q3(t))\,L3\,m3\cos(q2(t))\left(\frac{d}{dt}\,q2(t)\right)-\Bigg(\frac{d}{dt}$$

$$q3(t)\Big)^2 \sin(q2(t))\cos(q3(t))\,L3\,m3 + m1\,g + m2\,g + m3\,g - F1$$

Body 2

```
> ddtddq2dot_Lagr:=diff(diffF(Lagr,diff(q2(t),t)),t);
```

$$ddtddq2dot\_Lagr := -\left(\frac{d}{dt}\,q3(t)\right)\sin(q3(t))\,L3\,m3\cos(q2(t))\left(\frac{d}{dt}\,q1(t)\right) - \left(\cos(q3(t))\,L3\,m3 + \frac{L2\,(m2+2\,m3)}{2}\right)\left(\frac{d}{dt}\,q2(t)\right)\sin(q2(t))\left(\frac{d}{dt}\,q1(t)\right) + \left(\cos(q3(t))\,L3\,m3 + \frac{L2\,(m2+2\,m3)}{2}\right)\cos(q2(t))\left(\frac{d^2}{dt^2}\,q1(t)\right) + \frac{1}{4}\Bigg(\Bigg(-8\cos(q3(t))\,L3^2\,m3\left(\frac{d}{dt}\,q3(t)\right)\sin(q3(t)) - 8\left(\frac{d}{dt}\,q3(t)\right)\sin(q3(t))\,L2\,L3\,m3\Bigg)\left(\frac{d}{dt}\,q2(t)\right)\Bigg) + \frac{1}{4}\left(\left(4\cos(q3(t))^2 L3^2\,m3 + 8\cos(q3(t))\,L2\,L3\,m3 + L2^2\,(m2+4\,m3)\right)\left(\frac{d^2}{dt^2}\,q2(t)\right)\right) \quad \textbf{(3.7.2.1)}$$

```
> ddq2_Lagr:=diffF(Lagr,q2(t));
```

$$ddq2\_Lagr := \frac{1}{8}\Bigg(\Bigg(-8\left(\cos(q3(t))\,L3\,m3 + \frac{L2\,(m2+2\,m3)}{2}\right)\sin(q2(t))\left(\frac{d}{dt}\,q2(t)\right) - 8\left(\frac{d}{dt}\,q3(t)\right)\cos(q2(t))\sin(q3(t))\,L3\,m3\Bigg)\left(\frac{d}{dt}\,q1(t)\right)\Bigg) - \left(L3\cos(q3(t))\cos(q2(t))\,m3 + \frac{L2\,(m2+2\,m3)\cos(q2(t))}{2}\right)g \quad \textbf{(3.7.2.2)}$$

Hypotetic one

```
> eqn2:=simplify(ddtddq2dot_Lagr-ddq2_Lagr-Q2);
```

$$eqn2 := \frac{\left(4\cos(q3(t))^2 L3^2\,m3 + 8\cos(q3(t))\,L2\,L3\,m3 + L2^2\,(m2+4\,m3)\right)\left(\frac{d^2}{dt^2}\,q2(t)\right)}{4} + \left(\cos(q3(t))\,L3\,m3 + \frac{L2\,(m2+2\,m3)}{2}\right)\cos(q2(t))\left(\frac{d^2}{dt^2}\,q1(t)\right) - 2\sin(q3(t))\left(\frac{d}{dt}\,q3(t)\right)L3\,m3\,(\cos(q3(t))\,L3 + L2)\left(\frac{d}{dt}\,q2(t)\right) + \left(\cos(q3(t))\,L3\,m3 + \frac{L2\,(m2+2\,m3)}{2}\right)g\cos(q2(t)) - L3\,Fx\cos(q3(t)) - Fx\,L2 - C2 \quad \textbf{(3.7.2.3)}$$

Used one

```
> eqn2:=simplify(ddtddq2dot_Lagr-ddq2_Lagr-C2);
```

$$eqn2 := \frac{\left(4\cos(q3(t))^2 L3^2\, m3 + 8\cos(q3(t))\, L2\, L3\, m3 + L2^2\,(m2+4\, m3)\right)\left(\frac{d^2}{dt^2}\, q2(t)\right)}{4} + \left(\cos(q3(t))\, L3\, m3 + \frac{L2\,(m2+2\, m3)}{2}\right)\cos(q2(t))\left(\frac{d^2}{dt^2}\, q1(t)\right) - 2\sin(q3(t))\left(\frac{d}{dt}\, q3(t)\right) L3\, m3\,(\cos(q3(t))\, L3 + L2)\left(\frac{d}{dt}\, q2(t)\right) + \left(\cos(q3(t))\, L3\, m3 + \frac{L2\,(m2+2\, m3)}{2}\right) g\cos(q2(t)) - C2 \quad (3.7.2.4)$$

Body 3

```
> ddtddq3dot_Lagr:=diff(diffF(Lagr,diff(q3(t),t)),t);
```

$$ddtddq3dot\_Lagr := -\left(\frac{d}{dt}\, q2(t)\right)\cos(q2(t))\sin(q3(t))\, L3\, m3\left(\frac{d}{dt}\, q1(t)\right) - \sin(q2(t))\left(\frac{d}{dt}\, q3(t)\right)\cos(q3(t))\, L3\, m3\left(\frac{d}{dt}\, q1(t)\right) - \sin(q2(t))\sin(q3(t))\, L3\, m3\left(\frac{d^2}{dt^2}\, q1(t)\right) + m3\, L3^2\left(\frac{d^2}{dt^2}\, q3(t)\right) \quad (3.7.3.1)$$

```
> ddq3_Lagr:=diffF(Lagr,q3(t));
```

$$ddq3\_Lagr := \frac{1}{8}\left(\left(-8\sin(q3(t))\, L3\, m3\cos(q2(t))\left(\frac{d}{dt}\, q2(t)\right) - 8\left(\frac{d}{dt}\, q3(t)\right)\sin(q2(t))\cos(q3(t))\, L3\, m3\right)\left(\frac{d}{dt}\, q1(t)\right)\right) + \frac{\left(-8\cos(q3(t))\, L3^2\, m3\sin(q3(t)) - 8\sin(q3(t))\, L2\, L3\, m3\right)\left(\frac{d}{dt}\, q2(t)\right)^2}{8} + \sin(q2(t))\sin(q3(t))\, L3\, m3\, g \quad (3.7.3.2)$$

Hypotetic one

```
> eqn3:=simplify(ddtddq3dot_Lagr-ddq3_Lagr-Q3);
```

$$eqn3 := -\sin(q2(t))\sin(q3(t))\, L3\, m3\left(\frac{d^2}{dt^2}\, q1(t)\right) + m3\, L3^2\left(\frac{d^2}{dt^2}\, q3(t)\right) + \sin(q3(t))\, L3\, m3\,(\cos(q3(t))\, L3 + L2)\left(\frac{d}{dt}\, q2(t)\right)^2 - \sin(q2(t))\sin(q3(t))\, L3\, m3\, g - Fy\, L3 - C3 \quad (3.7.3.3)$$

Used one

```
> eqn3:=simplify(ddtddq3dot_Lagr-ddq3_Lagr-C3);
```

(3.7.3.4)

$$eqn3 := -\sin(q2(t))\sin(q3(t))\,L3\,m3\left(\frac{d^2}{dt^2}q1(t)\right) + m3\,L3^2\left(\frac{d^2}{dt^2}q3(t)\right) + \sin(q3(t))\,L3\,m3\,(\cos(q3(t))\,L3 + L2)\left(\frac{d}{dt}q2(t)\right)^2 - \sin(q2(t))\sin(q3(t))\,L3\,m3\,g - C3 \quad \textbf{(3.7.3.4)}$$

```
> motion_eqns:={eqn1,eqn2,eqn3}:
```

Direct dynamic

Find the motion of the joints given the data and the actions.

In the following, eventually add the definition of the symbolic forces introduced previously (e.g. frictions, viscous forces etc)

```
> actions:={F1=10,C2=5,C3=3};
```

$$actions := \{C2 = 5, C3 = 3, F1 = 10\} \quad \textbf{(3.8.1)}$$

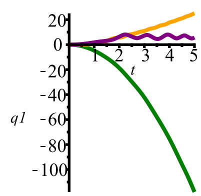
```
> ICs:={q1(0)=0,D(q1)(0)=0,q2(0)=0,D(q2)(0)=0,q3(0)=0,D(q3)(0)=0};
```

$$ICs := \{q1(0) = 0, q2(0) = 0, q3(0) = 0, D(q1)(0) = 0, D(q2)(0) = 0, D(q3)(0) = 0\} \quad \textbf{(3.8.2)}$$

```
> dsol:=dsolve(subs(data,actions,motion_eqns) union ICs,numeric);
```

$$dsol := \textbf{proc}(x\_rkf45) \;...\; \textbf{end proc} \quad \textbf{(3.8.3)}$$

```
> display([
          plots[odeplot](dsol,[t,q1(t)],0..5,numpoints=100),
          plots[odeplot](dsol,[t,q2(t)],0..5,numpoints=100),
          plots[odeplot](dsol,[t,q3(t)],0..5,numpoints=100)
      ],
      color=["Green","Orange","Purple"],
      size=[150,150]
  );
```



Inverse dynamic

Find the actions that produce a certain motion (i.e. most of the time the repositioning).

Manouvre

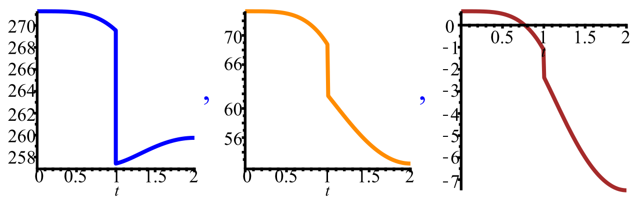
```
> profiles:={q1(t)=q1_profile,q2(t)=q2_profile,q3(t)=q3_profile}:
```

Solving

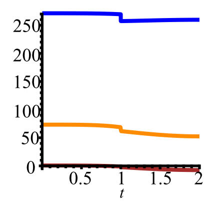
```
> f1:=solve(subs(data,profiles,eqn1)=0,F1):
> c2:=solve(subs(data,profiles,eqn2)=0,C2):
> c3:=solve(subs(data,profiles,eqn3)=0,C3):
```

Plotting

```
> plot(f1,t=0..Tmax,color="Blue"),plot(c2,t=0..Tmax,color=
  "DarkOrange"),plot(c3,t=0..Tmax,color="Brown");
```



```
> display([
          plot(f1,t=0..Tmax),
          plot(c2,t=0..Tmax),
          plot(c3,t=0..Tmax)
      ],
      color=["Blue","DarkOrange","Brown"],
      size=[150,150]
  );
```



Values at given time

```
> action_profiles:={"f1"=f1,"c2"=c2,"c3"=c3}:
```

t1 = 0.1ms

```
> action_t1:=evalf(subs(t=0.0001,action_profiles));
```

$$action\_t1 := \{"c2" = 73.30616794, "c3" = 0.6287635309, "f1" = 271.2958791\} \quad \mathbf{(3.9.1.1)}$$

t2 = 999.9ms

```
> action_t2:=evalf(subs(t=0.9999,action_profiles));
```

$$action\_t2 := \{"c2" = 68.69597967, "c3" = -1.127774079, "f1" = 269.5469708\} \quad \mathbf{(3.9.1.2)}$$

t3 = 1000.1ms

```
> action_t3:=evalf(subs(t=1.001,action_profiles));
```

$$action\_t3 := \{"c2" = 61.76161038, "c3" = -2.353615443, "f1" = 257.4124325\} \quad \mathbf{(3.9.1.3)}$$

t4 = 1999.9ms

```
> action_t4:=evalf(subs(t=1.9999,action_profiles));
```

$$action\_t4 := \{"c2" = 52.44566133, "c3" = -7.482683223, "f1" = 259.7588739\} \quad \mathbf{(3.9.1.4)}$$

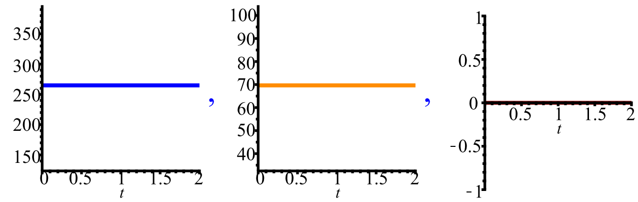
Static compensation

What are the values of the actions to keep the robot still with the joint variable at zero?

```
> positions:={q1(t)=0,q2(t)=0,q3(t)=0};
```

$$positions := \{q1(t) = 0, q2(t) = 0, q3(t) = 0\} \quad \mathbf{(3.9.2.1)}$$

```
> f1_still:=solve(subs(data,positions,eqn1)=0,F1):
> c2_still:=solve(subs(data,positions,eqn2)=0,C2):
> c3_still:=solve(subs(data,positions,eqn3)=0,C3):
> plot(f1_still,t=0..Tmax,color="Blue"),
  plot(c2_still,t=0..Tmax,color="DarkOrange"),
  plot(c3_still,t=0..Tmax,color="Brown");
```



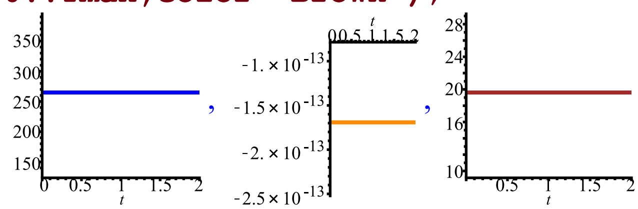
Different final position

What are the values of the actions to keep the robot still with the joint variable at zero?

```
> positions:={q1(t)=0,q2(t)=-Pi/2,q3(t)=Pi/2};
```

$$positions := \left\{q1(t) = 0, q2(t) = -\frac{\pi}{2}, q3(t) = \frac{\pi}{2}\right\} \quad \textbf{(3.9.3.1)}$$

```
> f1_still:=solve(subs(data,positions,eqn1)=0,F1):
> c2_still:=solve(subs(data,positions,eqn2)=0,C2):
> c3_still:=solve(subs(data,positions,eqn3)=0,C3):
> plot(f1_still,t=0..Tmax,color="Blue"),
  plot(c2_still,t=0..Tmax,color="DarkOrange"),
  plot(c3_still,t=0..Tmax,color="Brown");
```

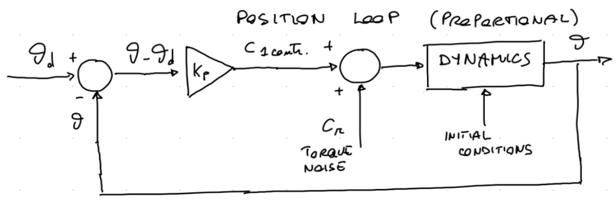


Control

The type of the loop define the quantities we want to control.
Obviously, in our case we are interessed in the position, so the position will be always considered.

The type of the controller define how we adjust them (in our cases always with a proportional action).

Position loop




```
> pos_cntr_actions:={
      F1=kp*(q1_profile-q1(t)),
      C2=kp*(q2_profile-q2(t)),
      C3=kp*(q3_profile-q3(t))
  }:
```

```
> ICs:={q1(0)=0,D(q1)(0)=0,q2(0)=0,D(q2)(0)=0,q3(0)=0,D(q3)(0)=0};
```

$$ICs := \{q1(0)=0, q2(0)=0, q3(0)=0, \mathrm{D}(q1)(0)=0, \mathrm{D}(q2)(0)=0, \mathrm{D}(q3)(0)=0\} \quad \textbf{(4.1.1)}$$

Caclulate and plot the joint motion profiles with different k

```
> dsol_pos_cntr_k1:=dsolve(subs(pos_cntr_actions,kp=k1,data,
  motion_eqns) union ICs,numeric);
```

$$dsol\_pos\_cntr\_k1 := \textbf{proc}(x\_rkf45) \;\ldots\; \textbf{end proc} \quad \textbf{(4.1.1.1)}$$

```
> dsol_pos_cntr_k2:=dsolve(subs(pos_cntr_actions,kp=k2,data,
  motion_eqns) union ICs,numeric);
```

$$dsol\_pos\_cntr\_k2 := \textbf{proc}(x\_rkf45) \;\ldots\; \textbf{end proc} \quad \textbf{(4.1.1.2)}$$

```
> dsol_pos_cntr_k3:=dsolve(subs(pos_cntr_actions,kp=k3,data,
  motion_eqns) union ICs,numeric);
```

$$dsol\_pos\_cntr\_k3 := \textbf{proc}(x\_rkf45) \;\ldots\; \textbf{end proc} \quad \textbf{(4.1.1.3)}$$

t1 = 0s

```
> dsol_pos_cntr_k1(0)[2],dsol_pos_cntr_k1(0)[4],dsol_pos_cntr_k1(0)
  [6];
```

$$q1(t)=0., q2(t)=0., q3(t)=0. \quad \textbf{(4.1.1.4)}$$

```
> dsol_pos_cntr_k2(0)[2],dsol_pos_cntr_k2(0)[4],dsol_pos_cntr_k2(0)
  [6];
```

$$q1(t)=0., q2(t)=0., q3(t)=0. \quad \textbf{(4.1.1.5)}$$

```
> dsol_pos_cntr_k3(0)[2],dsol_pos_cntr_k3(0)[4],dsol_pos_cntr_k3(0)
  [6];
```

$$q1(t)=0., q2(t)=0., q3(t)=0. \quad \textbf{(4.1.1.6)}$$

t2 = 1s

```
> dsol_pos_cntr_k1(1)[2],dsol_pos_cntr_k1(1)[4],dsol_pos_cntr_k1(1)
  [6];
```

$$q1(t)=0.0432759456483061, q2(t)=0.259155258954400, q3(t)=0.392998680827408 \quad \textbf{(4.1.1.7)}$$

```
> dsol_pos_cntr_k2(1)[2],dsol_pos_cntr_k2(1)[4],dsol_pos_cntr_k2(1)
  [6];
```

$$q1(t)=0.0432176581487227, q2(t)=0.259017933513634, q3(t)=0.392745546617047 \quad \textbf{(4.1.1.8)}$$

```
> dsol_pos_cntr_k3(1)[2],dsol_pos_cntr_k3(1)[4],dsol_pos_cntr_k3(1)
  [6];
```

$q1(t) = 0.0488976660912792, q2(t) = 0.262390674258094, q3(t) = 0.392943431004056$ **(4.1.1.9)**

t3 = 2s

```
> dsol_pos_cntr_k1(2)[2],dsol_pos_cntr_k1(2)[4],dsol_pos_cntr_k1(2)
  [6];
```

$q1(t) = 0.0771049356635189, q2(t) = 0.517748270983592, q3(t) = 0.786662899311100$ **(4.1.1.10)**

```
> dsol_pos_cntr_k2(2)[2],dsol_pos_cntr_k2(2)[4],dsol_pos_cntr_k2(2)
  [6];
```

$q1(t) = 0.0664494073760428, q2(t) = 0.515837518509738, q3(t) = 0.786988990847899$ **(4.1.1.11)**

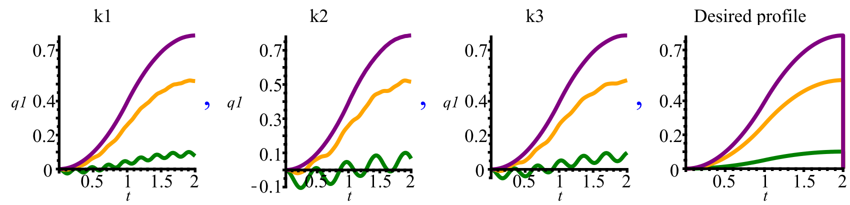
```
> dsol_pos_cntr_k3(2)[2],dsol_pos_cntr_k3(2)[4],dsol_pos_cntr_k3(2)
  [6];
```

$q1(t) = 0.0980338981030037, q2(t) = 0.522909466210480, q3(t) = 0.786008238770385$ **(4.1.1.12)**

Plot

```
> display([
          plots[odeplot](dsol_pos_cntr_k1,[t,q1(t)],0..2,numpoints=
  100),
          plots[odeplot](dsol_pos_cntr_k1,[t,q2(t)],0..2,numpoints=
  100),
          plots[odeplot](dsol_pos_cntr_k1,[t,q3(t)],0..2,numpoints=
  100)
      ],
      color=["Green","Orange","Purple"],
      size=[250,250],
      title="k1"
  ),
  display([
          plots[odeplot](dsol_pos_cntr_k2,[t,q1(t)],0..2,numpoints=
  100),
          plots[odeplot](dsol_pos_cntr_k2,[t,q2(t)],0..2,numpoints=
  100),
          plots[odeplot](dsol_pos_cntr_k2,[t,q3(t)],0..2,numpoints=
  100)
      ],
      color=["Green","Orange","Purple"],
      size=[250,250],
      title="k2"
  ),
  display([
          plots[odeplot](dsol_pos_cntr_k3,[t,q1(t)],0..2,numpoints=
  100),
          plots[odeplot](dsol_pos_cntr_k3,[t,q2(t)],0..2,numpoints=
  100),
          plots[odeplot](dsol_pos_cntr_k3,[t,q3(t)],0..2,numpoints=
  100)
      ],
      color=["Green","Orange","Purple"],
      size=[250,250],
      title="k3"
  ),
  # Pay attention!!! This is the desired one
  display([
          plot(q1_profile,t=0..Tmax),
```

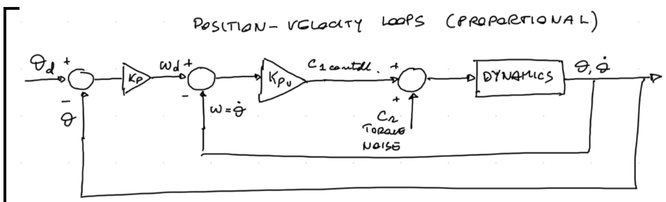
```
            plot(q2_profile,t=0..Tmax),
            plot(q3_profile,t=0..Tmax)
        ],
        color=["Green","Orange","Purple"],
        size=[250,250],
        title="Desired profile"
);
```



Incresing kp means:
- increase of bandwidth
- increase of noise rejection

Even if increasing it seems always a good idea, in reality there are additional poles that causes our system to become unstable.

Position-velocity loop



In this case the error in the joint position is converted into a target velocity and comparing this with the actual velocity we get the torque we have to apply as the result of this double feedback loop.

General substitution:

```
> C__theta:=kpv*(kp*(theta__desired(t)-theta(t))-diff(theta(t),t));
```

$$C_{\theta} := kpv\left(kp\left(\theta_{desired}(t) - \theta(t)\right) - \frac{d}{dt}\,\theta(t)\right) \quad (4.2.1)$$

```
> pos_vel_cntr_actions:={
      F1=kpv*(kp*(q1_profile-q1(t))-diff(q1(t),t)),
      C2=kpv*(kp*(q2_profile-q2(t))-diff(q2(t),t)),
      C3=kpv*(kp*(q3_profile-q3(t))-diff(q3(t),t))
  }:
```

```
> ICs:={q1(0)=0,D(q1)(0)=0,q2(0)=0,D(q2)(0)=0,q3(0)=0,D(q3)(0)=0};
```

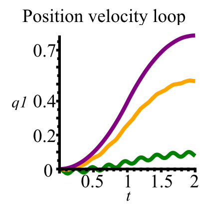
$$ICs := \{q1(0) = 0, q2(0) = 0, q3(0) = 0, \mathrm{D}(q1)(0) = 0, \mathrm{D}(q2)(0) = 0, \mathrm{D}(q3)(0) = 0\} \quad (4.2.2)$$

```
> dsol_pos_vel_cntr:=dsolve(subs(pos_cntr_actions,kp=k1,data,
  motion_eqns) union ICs,numeric);
```

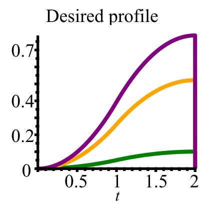
$$dsol\_pos\_vel\_cntr := \mathbf{proc}(x\_rkf45)\ \ldots\ \mathbf{end\ proc} \quad (4.2.3)$$

```
> display([
            plots[odeplot](dsol_pos_vel_cntr,[t,q1(t)],0..2,
  numpoints=100),
            plots[odeplot](dsol_pos_vel_cntr,[t,q2(t)],0..2,
  numpoints=100),
            plots[odeplot](dsol_pos_vel_cntr,[t,q3(t)],0..2,
  numpoints=100)
        ],
        color=["Green","Orange","Purple"],
```

```
        size=[250,250],
        title="Position velocity loop"
    ),
    # Pay attention!!! This is the desired one
    display([
            plot(q1_profile,t=0..Tmax),
            plot(q2_profile,t=0..Tmax),
            plot(q3_profile,t=0..Tmax)
        ],
        color=["Green","Orange","Purple"],
        size=[250,250],
        title="Desired profile"
    );
```



,



Sfizio: note that proceeding the same way as in other exercise gives the same substitution

```
> eqv:=subs(F1=kpv*(omegad-diff(q1(t),t)),eqn1):
> eqp:=subs(omegad=kp*(q_prof-q1(t)),eqv);
```

$$eqp := \frac{\left(\frac{d^2}{dt^2} q1(t)\right)(2\, m1 + 2\, m2 + 2\, m3)}{2} + \left(\cos(q3(t))\, L3\, m3 + \frac{L2\,(m2 + 2\, m3)}{2}\right)\cos(q2(t))\left(\frac{d^2}{dt^2} q2(t)\right) - \left(\frac{d^2}{dt^2} q3(t)\right)\sin(q2(t))\sin(q3(t))\, L3\, m3 - \left(\cos(q3(t))\, L3\, m3 + \frac{L2\,(m2 + 2\, m3)}{2}\right)\left(\frac{d}{dt} q2(t)\right)^2 \sin(q2(t)) - 2\left(\frac{d}{dt} q3(t)\right)\sin(q3(t))\, L3\, m3 \cos(q2(t))\left(\frac{d}{dt} q2(t)\right) - \left(\frac{d}{dt} q3(t)\right)^2 \sin(q2(t))\cos(q3(t))\, L3\, m3 + m1\, g + m2\, g + m3\, g - kpv\left(kp\,(q\_prof - q1(t)) - \frac{d}{dt} q1(t)\right) \quad \textbf{(4.2.4)}$$

```
> kpv*(kp*(q_prof - q1(t)) - diff(q1(t), t));
```

$$kpv\left(kp\,(q\_prof - q1(t)) - \frac{d}{dt} q1(t)\right) \quad \textbf{(4.2.5)}$$

Position proportional derivative loop

No drawing up to now.

General substitution:

```
> C__alpha:=kp*(alpha__desired(t)-alpha(t))+kpd*(diff
```

```
(alpha__desired(t),t)-diff(alpha(t),t));
```

$$C_{\alpha} := kp\left(\alpha_{desired}(t) - \alpha(t)\right) + kpd\left(\frac{d}{dt}\,\alpha_{desired}(t) - \frac{d}{dt}\,\alpha(t)\right) \tag{4.3.1}$$

```
> pos_vel_cntr_actions:={
      F1=kp*(q1_profile-q1(t))+kpd*(diff(q1_profile,t)-diff(q1(t),
  t)),
      C2=kp*(q2_profile-q2(t))+kpd*(diff(q2_profile,t)-diff(q2(t),
  t)),
      C3=kp*(q3_profile-q3(t))+kpd*(diff(q3_profile,t)-diff(q3(t),
  t))
  }:
> ICs:={q1(0)=0,D(q1)(0)=0,q2(0)=0,D(q2)(0)=0,q3(0)=0,D(q3)(0)=0};
```

$$ICs := \{q1(0) = 0, q2(0) = 0, q3(0) = 0, D(q1)(0) = 0, D(q2)(0) = 0, D(q3)(0) = 0\} \tag{4.3.2}$$

```
> dsol_pos_der_cntr:=dsolve(subs(pos_cntr_actions,kp=k1,data,
  motion_eqns) union ICs,numeric);
```

$$dsol\_pos\_der\_cntr := \mathbf{proc}(x\_rkf45)\ \ldots\ \mathbf{end\ proc} \tag{4.3.3}$$

```
> display([
          plots[odeplot](dsol_pos_der_cntr,[t,q1(t)],0..2,
  numpoints=100),
          plots[odeplot](dsol_pos_der_cntr,[t,q2(t)],0..2,
  numpoints=100),
          plots[odeplot](dsol_pos_der_cntr,[t,q3(t)],0..2,
  numpoints=100)
      ],
      color=["Green","Orange","Purple"],
      size=[250,250],
      title="Position derivative loop"
  ),
  # Pay attention!!! This is the desired one
  display([
          plot(q1_profile,t=0..Tmax),
          plot(q2_profile,t=0..Tmax),
          plot(q3_profile,t=0..Tmax)
      ],
      color=["Green","Orange","Purple"],
      size=[250,250],
      title="Desired profile"
  );
```