

# The stack of tasks

Florent Lamiriaux, Olivier Stasse and Nicolas Mansard

CNRS-LAAS, Toulouse, France

# The stack of tasks

Introduction

Theoretical foundations

Software

# Outline

Introduction

Theoretical foundations

Software

# Introduction

The stack of tasks provides a control framework for real-time redundant manipulator control

- ▶ implementation of a data-flow,
- ▶ control of the graph by python scripting,
- ▶ task-based hierarchical control,
- ▶ portable: tested on HRP-2, Nao, Romeo.

# Introduction

The stack of tasks provides a control framework for real-time redundant manipulator control

- ▶ implementation of a data-flow,
- ▶ control of the graph by python scripting,
- ▶ task-based hierarchical control,
- ▶ portable: tested on HRP-2, Nao, Romeo.

# Introduction

The stack of tasks provides a control framework for real-time redundant manipulator control

- ▶ implementation of a data-flow,
- ▶ control of the graph by python scripting,
- ▶ task-based hierarchical control,
- ▶ portable: tested on HRP-2, Nao, Romeo.

# Introduction

The stack of tasks provides a control framework for real-time redundant manipulator control

- ▶ implementation of a data-flow,
- ▶ control of the graph by python scripting,
- ▶ task-based hierarchical control,
- ▶ portable: tested on HRP-2, Nao, Romeo.

# Introduction

The stack of tasks provides a control framework for real-time redundant manipulator control

- ▶ implementation of a data-flow,
- ▶ control of the graph by python scripting,
- ▶ task-based hierarchical control,
- ▶ portable: tested on HRP-2, Nao, Romeo.



# Outline

Introduction

Theoretical foundations

Software

## Rigid body $\mathcal{B}$

- Configuration represented by an homogeneous matrix

$$M_{\mathcal{B}} = \begin{pmatrix} R_{\mathcal{B}} & \mathbf{t}_{\mathcal{B}} \\ 0 & 1 \end{pmatrix} \in SE(3)$$

$$R_{\mathcal{B}} \in SO(3) \Leftrightarrow R_{\mathcal{B}}^T R_{\mathcal{B}} = I_3$$

Point  $\mathbf{x} \in \mathbb{R}^3$  in local frame of  $\mathcal{B}$  is moved to  $\mathbf{y} \in \mathbb{R}^3$  in global frame:

$$\begin{pmatrix} \mathbf{y} \\ 1 \end{pmatrix} = M_{\mathcal{B}} \begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix}$$

## Rigid body $\mathcal{B}$

- Configuration represented by an homogeneous matrix

$$M_{\mathcal{B}} = \begin{pmatrix} R_{\mathcal{B}} & \mathbf{t}_{\mathcal{B}} \\ 0 & 1 \end{pmatrix} \in SE(3)$$

$$R_{\mathcal{B}} \in SO(3) \Leftrightarrow R_{\mathcal{B}}^T R_{\mathcal{B}} = I_3$$

Point  $\mathbf{x} \in \mathbb{R}^3$  in local frame of  $\mathcal{B}$  is moved to  $\mathbf{y} \in \mathbb{R}^3$  in global frame:

$$\begin{pmatrix} \mathbf{y} \\ 1 \end{pmatrix} = M_{\mathcal{B}} \begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix}$$

## Rigid body $\mathcal{B}$

- Configuration represented by an homogeneous matrix

$$M_{\mathcal{B}} = \begin{pmatrix} R_{\mathcal{B}} & \mathbf{t}_{\mathcal{B}} \\ 0 & 1 \end{pmatrix} \in SE(3)$$

$$R_{\mathcal{B}} \in SO(3) \Leftrightarrow R_{\mathcal{B}}^T R_{\mathcal{B}} = I_3$$

Point  $\mathbf{x} \in \mathbb{R}^3$  in local frame of  $\mathcal{B}$  is moved to  $\mathbf{y} \in \mathbb{R}^3$  in global frame:

$$\begin{pmatrix} \mathbf{y} \\ 1 \end{pmatrix} = M_{\mathcal{B}} \begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix}$$

## Rigid body $\mathcal{B}$

- ▶ Velocity represented by  $(\mathbf{v}_{\mathcal{B}}, \omega_{\mathcal{B}}) \in \mathbb{R}^6$  where

$$\dot{R}_{\mathcal{B}} = \hat{\omega}_{\mathcal{B}} R_{\mathcal{B}}$$

and

$$\hat{\omega} = \begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix}$$

is the matrix corresponding to the cross product operator

- ▶ Velocity of point P on  $\mathcal{B}$

$$\mathbf{v}_p = \dot{\mathbf{t}}_{\mathcal{B}} + \omega_{\mathcal{B}} \times O_{\mathcal{B}} \vec{P}$$

where  $O_{\mathcal{B}}$  is the origin of the local frame of  $\mathcal{B}$ .

## Rigid body $\mathcal{B}$

- ▶ Velocity represented by  $(\mathbf{v}_{\mathcal{B}}, \omega_{\mathcal{B}}) \in \mathbb{R}^6$  where

$$\dot{R}_{\mathcal{B}} = \hat{\omega}_{\mathcal{B}} R_{\mathcal{B}}$$

and

$$\hat{\omega} = \begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix}$$

is the matrix corresponding to the cross product operator

- ▶ Velocity of point P on  $\mathcal{B}$

$$\mathbf{v}_p = \dot{\mathbf{t}}_{\mathcal{B}} + \omega_{\mathcal{B}} \times \vec{O_{\mathcal{B}}P}$$

where  $O_{\mathcal{B}}$  is the origin of the local frame of  $\mathcal{B}$ .

## Rigid body $\mathcal{B}$

- ▶ Velocity represented by  $(\mathbf{v}_{\mathcal{B}}, \omega_{\mathcal{B}}) \in \mathbb{R}^6$  where

$$\dot{R}_{\mathcal{B}} = \hat{\omega}_{\mathcal{B}} R_{\mathcal{B}}$$

and

$$\hat{\omega} = \begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix}$$

is the matrix corresponding to the cross product operator

- ▶ Velocity of point P on  $\mathcal{B}$

$$\mathbf{v}_p = \dot{\mathbf{t}}_{\mathcal{B}} + \omega_{\mathcal{B}} \times O_{\mathcal{B}} \vec{P}$$

where  $O_{\mathcal{B}}$  is the origin of the local frame of  $\mathcal{B}$ .

# Configuration space

- ▶ Robot: set of rigid-bodies linked by joints  $\mathcal{B}_0, \dots \mathcal{B}_m$ .
- ▶ Configuration: position in space of each body.

$$\mathbf{q} = (\mathbf{q}_{waist}, \theta_1, \dots, \theta_{n-6}) \in SE(3) \times \mathbb{R}^{n-6}$$
$$\mathbf{q}_{waist} = (x, y, z, roll, pitch, yaw)$$



- ▶ Position of  $\mathcal{B}_i$  depends on  $\mathbf{q}$ :

$$M_{\mathcal{B}_i}(\mathbf{q}) \in SE(3)$$



# Configuration space

- ▶ Robot: set of rigid-bodies linked by joints  $\mathcal{B}_0, \dots \mathcal{B}_m$ .
- ▶ Configuration: position in space of each body.

$$\mathbf{q} = (\mathbf{q}_{waist}, \theta_1, \dots \theta_{n-6}) \in SE(3) \times \mathbb{R}^{n-6}$$
$$\mathbf{q}_{waist} = (x, y, z, roll, pitch, yaw)$$



- ▶ Position of  $\mathcal{B}_i$  depends on  $\mathbf{q}$ :

$$M_{\mathcal{B}_i}(\mathbf{q}) \in SE(3)$$

# Configuration space

- ▶ Robot: set of rigid-bodies linked by joints  $\mathcal{B}_0, \dots \mathcal{B}_m$ .
- ▶ Configuration: position in space of each body.

$$\mathbf{q} = (\mathbf{q}_{waist}, \theta_1, \dots \theta_{n-6}) \in SE(3) \times \mathbb{R}^{n-6}$$
$$\mathbf{q}_{waist} = (x, y, z, roll, pitch, yaw)$$



- ▶ Position of  $\mathcal{B}_i$  depends on  $\mathbf{q}$ :

$$M_{\mathcal{B}_i}(\mathbf{q}) \in SE(3)$$

# Velocity

► Velocity:

$$\begin{aligned}\dot{\mathbf{q}} &= (\dot{x}, \dot{y}, \dot{z}, \omega_x, \omega_y, \omega_z, \dot{\theta}_1, \dots, \dot{\theta}_{n-6}) \\ \omega &\in \mathbb{R}^3\end{aligned}$$

► Velocity of  $\mathcal{B}_i$



$$\begin{pmatrix} \mathbf{v}_{\mathcal{B}_i} \\ \omega_{\mathcal{B}_i} \end{pmatrix} (\mathbf{q}, \dot{\mathbf{q}}) = J_{\mathcal{B}_i}(\mathbf{q}) \cdot \dot{\mathbf{q}} \in \mathbb{R}^6$$

# Velocity

- Velocity:

$$\begin{aligned}\dot{\mathbf{q}} &= (\dot{x}, \dot{y}, \dot{z}, \omega_x, \omega_y, \omega_z, \dot{\theta}_1, \dots, \dot{\theta}_{n-6}) \\ \omega &\in \mathbb{R}^3\end{aligned}$$

- Velocity of  $\mathcal{B}_i$



$$\begin{pmatrix} \mathbf{v}_{\mathcal{B}_i} \\ \omega_{\mathcal{B}_i} \end{pmatrix} (\mathbf{q}, \dot{\mathbf{q}}) = J_{\mathcal{B}_i}(\mathbf{q}) \cdot \dot{\mathbf{q}} \in \mathbb{R}^6$$

# Velocity

- Velocity:

$$\begin{aligned}\dot{\mathbf{q}} &= (\dot{x}, \dot{y}, \dot{z}, \omega_x, \omega_y, \omega_z, \dot{\theta}_1, \dots, \dot{\theta}_{n-6}) \\ \omega &\in \mathbb{R}^3\end{aligned}$$

- Velocity of  $\mathcal{B}_i$



$$\begin{pmatrix} \mathbf{v}_{\mathcal{B}_i} \\ \omega_{\mathcal{B}_i} \end{pmatrix} (\mathbf{q}, \dot{\mathbf{q}}) = J_{\mathcal{B}_i}(\mathbf{q}) \cdot \dot{\mathbf{q}} \in \mathbb{R}^6$$

# Task

- ▶ Definition: function of the
  - ▶ robot configuration,
  - ▶ time and
  - ▶ possibly external parameters

that should converge to 0:

$$T \in C^\infty(\mathcal{C} \times \mathbb{R}, \mathbb{R}^m)$$

- ▶ Example: position tracking of an end-effector  $\mathcal{B}_{ee}$ 
  - ▶  $M(\mathbf{q}) \in SE(3)$  position of the end-effector,
  - ▶  $M^*(t) \in SE(3)$  reference position

$$T(\mathbf{q}, t) = \begin{pmatrix} \mathbf{t}(M^{*-1}(t)M(\mathbf{q})) \\ u_\theta(R^{*-1}(t)R(\mathbf{q})) \end{pmatrix}$$

where

- ▶  $\mathbf{t}()$  is the translation part of an homogeneous matrix,
- ▶  $R$  and  $R^*$  are the rotation part of  $M$  and  $M^*$

# Task

- ▶ Definition: function of the
  - ▶ robot configuration,
  - ▶ time and
  - ▶ possibly external parameters

that should converge to 0:

$$T \in C^\infty(\mathcal{C} \times \mathbb{R}, \mathbb{R}^m)$$

- ▶ Example: position tracking of an end-effector  $\mathcal{B}_{ee}$ 
  - ▶  $M(\mathbf{q}) \in SE(3)$  position of the end-effector,
  - ▶  $M^*(t) \in SE(3)$  reference position

$$T(\mathbf{q}, t) = \begin{pmatrix} \mathbf{t}(M^{*-1}(t)M(\mathbf{q})) \\ u_\theta(R^{*-1}(t)R(\mathbf{q})) \end{pmatrix}$$

where

- ▶  $\mathbf{t}()$  is the translation part of an homogeneous matrix,
- ▶  $R$  and  $R^*$  are the rotation part of  $M$  and  $M^*$

# Task

- ▶ Definition: function of the
  - ▶ robot configuration,
  - ▶ time and
  - ▶ possibly external parameters

that should converge to 0:

$$T \in C^\infty(\mathcal{C} \times \mathbb{R}, \mathbb{R}^m)$$

- ▶ Example: position tracking of an end-effector  $\mathcal{B}_{ee}$ 
  - ▶  $M(\mathbf{q}) \in SE(3)$  position of the end-effector,
  - ▶  $M^*(t) \in SE(3)$  reference position

$$T(\mathbf{q}, t) = \begin{pmatrix} \mathbf{t}(M^{*-1}(t)M(\mathbf{q})) \\ u_\theta(R^{*-1}(t)R(\mathbf{q})) \end{pmatrix}$$

where

- ▶  $\mathbf{t}()$  is the translation part of an homogeneous matrix,
- ▶  $R$  and  $R^*$  are the rotation part of  $M$  and  $M^*$



# Task

- ▶ Definition: function of the
  - ▶ robot configuration,
  - ▶ time and
  - ▶ possibly external parameters

that should converge to 0:

$$T \in C^\infty(\mathcal{C} \times \mathbb{R}, \mathbb{R}^m)$$

- ▶ Example: position tracking of an end-effector  $\mathcal{B}_{ee}$ 
  - ▶  $M(\mathbf{q}) \in SE(3)$  position of the end-effector,
  - ▶  $M^*(t) \in SE(3)$  reference position

$$T(\mathbf{q}, t) = \begin{pmatrix} \mathbf{t}(M^{*-1}(t)M(\mathbf{q})) \\ u_\theta(R^{*-1}(t)R(\mathbf{q})) \end{pmatrix}$$

where

- ▶  $\mathbf{t}()$  is the translation part of an homogeneous matrix,
- ▶  $R$  and  $R^*$  are the rotation part of  $M$  and  $M^*$

## Task

- ▶ Definition: function of the
  - ▶ robot configuration,
  - ▶ time and
  - ▶ possibly external parameters

that should converge to 0:

$$T \in C^\infty(\mathcal{C} \times \mathbb{R}, \mathbb{R}^m)$$

- ▶ Example: position tracking of an end-effector  $\mathcal{B}_{ee}$ 
  - ▶  $M(\mathbf{q}) \in SE(3)$  position of the end-effector,
  - ▶  $M^*(t) \in SE(3)$  reference position

$$T(\mathbf{q}, t) = \begin{pmatrix} \mathbf{t}(M^{*-1}(t)M(\mathbf{q})) \\ u_\theta(R^{*-1}(t)R(\mathbf{q})) \end{pmatrix}$$

where

- ▶  $\mathbf{t}()$  is the translation part of an homogeneous matrix,
- ▶  $R$  and  $R^*$  are the rotation part of  $M$  and  $M^*$ .

# Hierarchical task based control

Given

- ▶ a configuration  $\mathbf{q}$ ,
- ▶ two tasks of decreasing priorities:
  - ▶  $T_1 \in C^\infty(\mathcal{C} \times \mathbb{R}, \mathbb{R}^{m_1})$ ,
  - ▶  $T_2 \in C^\infty(\mathcal{C} \times \mathbb{R}, \mathbb{R}^{m_2})$ ,

compute a control vector  $\dot{\mathbf{q}}$

- ▶ that makes  $T_1$  converge toward 0 and
- ▶ that makes  $T_2$  converge toward 0 if possible.

# Hierarchical task based control

Given

- ▶ a configuration  $\mathbf{q}$ ,
- ▶ two tasks of decreasing priorities:
  - ▶  $T_1 \in C^\infty(\mathcal{C} \times \mathbb{R}, \mathbb{R}^{m_1})$ ,
  - ▶  $T_2 \in C^\infty(\mathcal{C} \times \mathbb{R}, \mathbb{R}^{m_2})$ ,

compute a control vector  $\dot{\mathbf{q}}$

- ▶ that makes  $T_1$  converge toward 0 and
- ▶ that makes  $T_2$  converge toward 0 if possible.

# Hierarchical task based control

Jacobian:

- ▶ we denote

- ▶  $J_i = \frac{\partial T_i}{\partial \mathbf{q}}$  for  $i \in \{1, 2\}$

- ▶ then

- ▶  $\forall \mathbf{q} \in \mathcal{C}, \forall t \in \mathbb{R}, \forall \dot{\mathbf{q}} \in \mathbb{R}^n, \dot{T}_i = J_i(\mathbf{q}, t)\dot{\mathbf{q}} + \frac{\partial T_i}{\partial t}(\mathbf{q}, t)$

We try to enforce

- ▶  $\dot{T}_1 = -\lambda_1 T_1 \Rightarrow T_1(t) = e^{-\lambda_1 t} T_1(0) \rightarrow 0$

- ▶  $\dot{T}_2 = -\lambda_2 T_2 \Rightarrow T_2(t) = e^{-\lambda_2 t} T_2(0) \rightarrow 0$

- ▶  $\lambda_1$  and  $\lambda_2$  are called the gains associated to  $T_1$  and  $T_2$ .

# Hierarchical task based control

Jacobian:

- ▶ we denote
  - ▶  $J_i = \frac{\partial T_i}{\partial \mathbf{q}}$  for  $i \in \{1, 2\}$
- ▶ then
  - ▶  $\forall \mathbf{q} \in \mathcal{C}, \forall t \in \mathbb{R}, \forall \dot{\mathbf{q}} \in \mathbb{R}^n, \dot{T}_i = J_i(\mathbf{q}, t)\dot{\mathbf{q}} + \frac{\partial T_i}{\partial t}(\mathbf{q}, t)$

We try to enforce

- ▶  $\dot{T}_1 = -\lambda_1 T_1 \Rightarrow T_1(t) = e^{-\lambda_1 t} T_1(0) \rightarrow 0$
- ▶  $\dot{T}_2 = -\lambda_2 T_2 \Rightarrow T_2(t) = e^{-\lambda_2 t} T_2(0) \rightarrow 0$
- ▶  $\lambda_1$  and  $\lambda_2$  are called the gains associated to  $T_1$  and  $T_2$ .

# Hierarchical task based control

Jacobian:

- ▶ we denote
  - ▶  $J_i = \frac{\partial T_i}{\partial \mathbf{q}}$  for  $i \in \{1, 2\}$
- ▶ then
  - ▶  $\forall \mathbf{q} \in \mathcal{C}, \forall t \in \mathbb{R}, \forall \dot{\mathbf{q}} \in \mathbb{R}^n, \dot{T}_i = J_i(\mathbf{q}, t)\dot{\mathbf{q}} + \frac{\partial T_i}{\partial t}(\mathbf{q}, t)$

We try to enforce

- ▶  $\dot{T}_1 = -\lambda_1 T_1 \Rightarrow T_1(t) = e^{-\lambda_1 t} T_1(0) \rightarrow 0$
- ▶  $\dot{T}_2 = -\lambda_2 T_2 \Rightarrow T_2(t) = e^{-\lambda_2 t} T_2(0) \rightarrow 0$
- ▶  $\lambda_1$  and  $\lambda_2$  are called the gains associated to  $T_1$  and  $T_2$ .

## Hierarchical task based control

Jacobian:

- ▶ we denote
  - ▶  $J_i = \frac{\partial T_i}{\partial \dot{\mathbf{q}}}$  for  $i \in \{1, 2\}$
- ▶ then
  - ▶  $\forall \mathbf{q} \in \mathcal{C}, \forall t \in \mathbb{R}, \forall \dot{\mathbf{q}} \in \mathbb{R}^n, \dot{T}_i = J_i(\mathbf{q}, t)\dot{\mathbf{q}} + \frac{\partial T_i}{\partial t}(\mathbf{q}, t)$

We try to enforce

- ▶  $\dot{T}_1 = -\lambda_1 T_1 \Rightarrow T_1(t) = e^{-\lambda_1 t} T_1(0) \rightarrow 0$
- ▶  $\dot{T}_2 = -\lambda_2 T_2 \Rightarrow T_2(t) = e^{-\lambda_2 t} T_2(0) \rightarrow 0$
- ▶  $\lambda_1$  and  $\lambda_2$  are called the gains associated to  $T_1$  and  $T_2$ .



# Hierarchical task based control

Jacobian:

- ▶ we denote
  - ▶  $J_i = \frac{\partial T_i}{\partial \mathbf{q}}$  for  $i \in \{1, 2\}$
- ▶ then
  - ▶  $\forall \mathbf{q} \in \mathcal{C}, \forall t \in \mathbb{R}, \forall \dot{\mathbf{q}} \in \mathbb{R}^n, \dot{T}_i = J_i(\mathbf{q}, t)\dot{\mathbf{q}} + \frac{\partial T_i}{\partial t}(\mathbf{q}, t)$

We try to enforce

- ▶  $\dot{T}_1 = -\lambda_1 T_1 \Rightarrow T_1(t) = e^{-\lambda_1 t} T_1(0) \rightarrow 0$
- ▶  $\dot{T}_2 = -\lambda_2 T_2 \Rightarrow T_2(t) = e^{-\lambda_2 t} T_2(0) \rightarrow 0$
- ▶  $\lambda_1$  and  $\lambda_2$  are called the gains associated to  $T_1$  and  $T_2$ .

## Moore Penrose pseudo-inverse

Given a matrix  $A \in \mathbb{R}^{m \times n}$ , the Moore Penrose pseudo inverse  $A^+ \in \mathbb{R}^{n \times m}$  of  $A$  is the unique matrix satisfying:

$$\begin{aligned}AA^+A &= A \\A^+AA^+ &= A^+ \\(AA^+)^T &= AA^+ \\(A^+A)^T &= A^+A\end{aligned}$$

Given a linear system:

$$Ax = b, \quad A \in \mathbb{R}^{m \times n}, \quad x \in \mathbb{R}^n, \quad b \in \mathbb{R}^m$$

$x = A^+b$  minimizes

- ▶  $\|Ax - b\|$  over  $\mathbb{R}^n$ ,
- ▶  $\|x\|$  over  $\operatorname{argmin} \|Ax - b\|$ .

## Moore Penrose pseudo-inverse

Given a matrix  $A \in \mathbb{R}^{m \times n}$ , the Moore Penrose pseudo inverse  $A^+ \in \mathbb{R}^{n \times m}$  of  $A$  is the unique matrix satisfying:

$$\begin{aligned}AA^+A &= A \\A^+AA^+ &= A^+ \\(AA^+)^T &= AA^+ \\(A^+A)^T &= A^+A\end{aligned}$$

Given a linear system:

$$Ax = b, \quad A \in \mathbb{R}^{m \times n}, \quad x \in \mathbb{R}^n, \quad b \in \mathbb{R}^m$$

$x = A^+b$  minimizes

- ▶  $\|Ax - b\|$  over  $\mathbb{R}^n$ ,
- ▶  $\|x\|$  over  $\operatorname{argmin} \|Ax - b\|$ .

## Moore Penrose pseudo-inverse

Given a matrix  $A \in \mathbb{R}^{m \times n}$ , the Moore Penrose pseudo inverse  $A^+ \in \mathbb{R}^{n \times m}$  of  $A$  is the unique matrix satisfying:

$$\begin{aligned}AA^+A &= A \\A^+AA^+ &= A^+ \\(AA^+)^T &= AA^+ \\(A^+A)^T &= A^+A\end{aligned}$$

Given a linear system:

$$Ax = b, \quad A \in \mathbb{R}^{m \times n}, \quad x \in \mathbb{R}^n, \quad b \in \mathbb{R}^m$$

$x = A^+b$  minimizes

- ▶  $\|Ax - b\|$  over  $\mathbb{R}^n$ ,
- ▶  $\|x\|$  over  $\operatorname{argmin} \|Ax - b\|$ .

## Moore Penrose pseudo-inverse

Given a matrix  $A \in \mathbb{R}^{m \times n}$ , the Moore Penrose pseudo inverse  $A^+ \in \mathbb{R}^{n \times m}$  of  $A$  is the unique matrix satisfying:

$$\begin{aligned}AA^+A &= A \\A^+AA^+ &= A^+ \\(AA^+)^T &= AA^+ \\(A^+A)^T &= A^+A\end{aligned}$$

Given a linear system:

$$Ax = b, \quad A \in \mathbb{R}^{m \times n}, \quad x \in \mathbb{R}^n, \quad b \in \mathbb{R}^m$$

$x = A^+b$  minimizes

- ▶  $\|Ax - b\|$  over  $\mathbb{R}^n$ ,
- ▶  $\|x\|$  over  $\operatorname{argmin} \|Ax - b\|$ .

# Hierarchical task based control

Resolution of the first constraint:

$$\dot{T}_1 = J_1 \dot{\mathbf{q}} + \frac{\partial T_1}{\partial t} = -\lambda_1 T_1 \quad (1)$$

$$J_1 \dot{\mathbf{q}} = -\lambda_1 T_1 - \frac{\partial T_1}{\partial t} \quad (2)$$

$$\dot{\mathbf{q}}_1 \triangleq -J_1^+ (\lambda_1 T_1 + \frac{\partial T_1}{\partial t}) \quad (3)$$

Where  $J_1^+$  is the (Moore Penrose) pseudo-inverse of  $J_1$ .  
 $\dot{\mathbf{q}}_1$  minimizes

- ▶  $\|J_1 \dot{\mathbf{q}} + \lambda_1 T_1 + \frac{\partial T_1}{\partial t}\| = \|\dot{T}_1 + \lambda_1 T_1\|$
- ▶  $\|\dot{\mathbf{q}}\|$  over  $\operatorname{argmin} \|J_1 \dot{\mathbf{q}} + \lambda_1 T_1 + \frac{\partial T_1}{\partial t}\|$

Hence,

- ▶ if  $\lambda_1 T_1 + \frac{\partial T_1}{\partial t}$  is in  $\operatorname{Im}(J_1)$ , (1) is satisfied

# Hierarchical task based control

Resolution of the first constraint:

$$\dot{T}_1 = J_1 \dot{\mathbf{q}} + \frac{\partial T_1}{\partial t} = -\lambda_1 T_1 \quad (1)$$

$$J_1 \dot{\mathbf{q}} = -\lambda_1 T_1 - \frac{\partial T_1}{\partial t} \quad (2)$$

$$\dot{\mathbf{q}}_1 \triangleq -J_1^+ (\lambda_1 T_1 + \frac{\partial T_1}{\partial t}) \quad (3)$$

Where  $J_1^+$  is the (Moore Penrose) pseudo-inverse of  $J_1$ .

$\dot{\mathbf{q}}_1$  minimizes

- ▶  $\|J_1 \dot{\mathbf{q}} + \lambda_1 T_1 + \frac{\partial T_1}{\partial t}\| = \|\dot{T}_1 + \lambda_1 T_1\|$
- ▶  $\|\dot{\mathbf{q}}\|$  over  $\operatorname{argmin} \|J_1 \dot{\mathbf{q}} + \lambda_1 T_1 + \frac{\partial T_1}{\partial t}\|$

Hence,

- ▶ if  $\lambda_1 T_1 + \frac{\partial T_1}{\partial t}$  is in  $\operatorname{Im}(J_1)$ , (1) is satisfied

## Hierarchical task based control

Resolution of the first constraint:

$$\dot{T}_1 = J_1 \dot{\mathbf{q}} + \frac{\partial T_1}{\partial t} = -\lambda_1 T_1 \quad (1)$$

$$J_1 \dot{\mathbf{q}} = -\lambda_1 T_1 - \frac{\partial T_1}{\partial t} \quad (2)$$

$$\dot{\mathbf{q}}_1 \triangleq -J_1^+ (\lambda_1 T_1 + \frac{\partial T_1}{\partial t}) \quad (3)$$

Where  $J_1^+$  is the (Moore Penrose) pseudo-inverse of  $J_1$ .  
 $\dot{\mathbf{q}}_1$  minimizes

- ▶  $\|J_1 \dot{\mathbf{q}} + \lambda_1 T_1 + \frac{\partial T_1}{\partial t}\| = \|\dot{T}_1 + \lambda_1 T_1\|$
- ▶  $\|\dot{\mathbf{q}}\|$  over  $\operatorname{argmin} \|J_1 \dot{\mathbf{q}} + \lambda_1 T_1 + \frac{\partial T_1}{\partial t}\|$

Hence,

- ▶ if  $\lambda_1 T_1 + \frac{\partial T_1}{\partial t}$  is in  $\operatorname{Im}(J_1)$ , (1) is satisfied



# Hierarchical task based control

Resolution of the first constraint:

$$\dot{T}_1 = J_1 \dot{\mathbf{q}} + \frac{\partial T_1}{\partial t} = -\lambda_1 T_1 \quad (1)$$

$$J_1 \dot{\mathbf{q}} = -\lambda_1 T_1 - \frac{\partial T_1}{\partial t} \quad (2)$$

$$\dot{\mathbf{q}}_1 \triangleq -J_1^+ (\lambda_1 T_1 + \frac{\partial T_1}{\partial t}) \quad (3)$$

Where  $J_1^+$  is the (Moore Penrose) pseudo-inverse of  $J_1$ .  
 $\dot{\mathbf{q}}_1$  minimizes

- ▶  $\|J_1 \dot{\mathbf{q}} + \lambda_1 T_1 + \frac{\partial T_1}{\partial t}\| = \|\dot{T}_1 + \lambda_1 T_1\|$
- ▶  $\|\dot{\mathbf{q}}\|$  over  $\operatorname{argmin} \|J_1 \dot{\mathbf{q}} + \lambda_1 T_1 + \frac{\partial T_1}{\partial t}\|$

Hence,

- ▶ if  $\lambda_1 T_1 + \frac{\partial T_1}{\partial t}$  is in  $\operatorname{Im}(J_1)$ , (1) is satisfied

## Hierarchical task based control

Resolution of the first constraint:

$$\dot{T}_1 = J_1 \dot{\mathbf{q}} + \frac{\partial T_1}{\partial t} = -\lambda_1 T_1 \quad (1)$$

$$J_1 \dot{\mathbf{q}} = -\lambda_1 T_1 - \frac{\partial T_1}{\partial t} \quad (2)$$

$$\dot{\mathbf{q}}_1 \triangleq -J_1^+ (\lambda_1 T_1 + \frac{\partial T_1}{\partial t}) \quad (3)$$

Where  $J_1^+$  is the (Moore Penrose) pseudo-inverse of  $J_1$ .  
 $\dot{\mathbf{q}}_1$  minimizes

- ▶  $\|J_1 \dot{\mathbf{q}} + \lambda_1 T_1 + \frac{\partial T_1}{\partial t}\| = \|\dot{T}_1 + \lambda_1 T_1\|$
- ▶  $\|\dot{\mathbf{q}}\|$  over  $\operatorname{argmin} \|J_1 \dot{\mathbf{q}} + \lambda_1 T_1 + \frac{\partial T_1}{\partial t}\|$

Hence,

- ▶ if  $\lambda_1 T_1 + \frac{\partial T_1}{\partial t}$  is in  $\operatorname{Im}(J_1)$ , (1) is satisfied

# Hierarchical task based control

In fact

$$\forall u \in \mathbb{R}^n, \quad J_1 (\dot{\mathbf{q}}_1 + (I_n - J_1^+ J_1) u) = J_1 \dot{\mathbf{q}}_1$$

therefore,

$$\dot{\mathbf{q}} = \dot{\mathbf{q}}_1 + (I_n - J_1^+ J_1) u$$

also minimizes  $\|J_1 \dot{\mathbf{q}} + \lambda_1 T_1 + \frac{\partial T_1}{\partial t}\|$ .

$P_1 = (I_n - J_1^+ J_1)$  is a projector on  $J_1$  kernel:

$$J_1 P_1 = 0$$

$\forall u \in \mathbb{R}^n$ , if  $\dot{\mathbf{q}} = P_1 u$ , then,  $\dot{T}_1 = \frac{\partial T_1}{\partial t}$ .

# Hierarchical task based control

In fact

$$\forall u \in \mathbb{R}^n, \quad J_1 (\dot{\mathbf{q}}_1 + (I_n - J_1^+ J_1) u) = J_1 \dot{\mathbf{q}}_1$$

therefore,

$$\dot{\mathbf{q}} = \dot{\mathbf{q}}_1 + (I_n - J_1^+ J_1) u$$

also minimizes  $\|J_1 \dot{\mathbf{q}} + \lambda_1 T_1 + \frac{\partial T_1}{\partial t}\|$ .

$P_1 = (I_n - J_1^+ J_1)$  is a projector on  $J_1$  kernel:

$$J_1 P_1 = 0$$

$\forall u \in \mathbb{R}^n$ , if  $\dot{\mathbf{q}} = P_1 u$ , then,  $\dot{T}_1 = \frac{\partial T_1}{\partial t}$ .

# Hierarchical task based control

In fact

$$\forall u \in \mathbb{R}^n, \quad J_1 (\dot{\mathbf{q}}_1 + (I_n - J_1^+ J_1) u) = J_1 \dot{\mathbf{q}}_1$$

therefore,

$$\dot{\mathbf{q}} = \dot{\mathbf{q}}_1 + (I_n - J_1^+ J_1) u$$

also minimizes  $\|J_1 \dot{\mathbf{q}} + \lambda_1 T_1 + \frac{\partial T_1}{\partial t}\|$ .

$P_1 = (I_n - J_1^+ J_1)$  is a projector on  $J_1$  kernel:

$$J_1 P_1 = 0$$

$\forall u \in \mathbb{R}^n$ , if  $\dot{\mathbf{q}} = P_1 u$ , then,  $\dot{T}_1 = \frac{\partial T_1}{\partial t}$ .

# Hierarchical task based control

In fact

$$\forall u \in \mathbb{R}^n, \quad J_1 (\dot{\mathbf{q}}_1 + (I_n - J_1^+ J_1)u) = J_1 \dot{\mathbf{q}}_1$$

therefore,

$$\dot{\mathbf{q}} = \dot{\mathbf{q}}_1 + (I_n - J_1^+ J_1)u$$

also minimizes  $\|J_1 \dot{\mathbf{q}} + \lambda_1 T_1 + \frac{\partial T_1}{\partial t}\|$ .

$P_1 = (I_n - J_1^+ J_1)$  is a projector on  $J_1$  kernel:

$$J_1 P_1 = 0$$

$\forall u \in \mathbb{R}^n$ , if  $\dot{\mathbf{q}} = P_1 u$ , then,  $\dot{T}_1 = \frac{\partial T_1}{\partial t}$ .

## Controlling the second task

We have

$$\dot{\mathbf{q}} = \dot{\mathbf{q}}_1 + P_1 u$$

$$\dot{T}_2 = J_2 \dot{\mathbf{q}} + \frac{\partial T_2}{\partial \mathbf{q}}$$

$$\dot{T}_2 = J_2 \dot{\mathbf{q}}_1 + \frac{\partial T_2}{\partial \mathbf{q}} + J_2 P_1 u$$

We want

$$\dot{T}_2 = -\lambda_2 T_2$$

Thus

$$-\lambda_2 T_2 = J_2 \dot{\mathbf{q}}_1 + \frac{\partial T_2}{\partial \mathbf{q}} + J_2 P_1 u$$

$$J_2 P_1 u = -\lambda_2 T_2 - J_2 \dot{\mathbf{q}}_1 - \frac{\partial T_2}{\partial \mathbf{q}}$$

## Controlling the second task

We have

$$\dot{\mathbf{q}} = \dot{\mathbf{q}}_1 + P_1 u$$

$$\dot{T}_2 = J_2 \dot{\mathbf{q}} + \frac{\partial T_2}{\partial \mathbf{q}}$$

$$\dot{T}_2 = J_2 \dot{\mathbf{q}}_1 + \frac{\partial T_2}{\partial \mathbf{q}} + J_2 P_1 u$$

We want

$$\dot{T}_2 = -\lambda_2 T_2$$

Thus

$$-\lambda_2 T_2 = J_2 \dot{\mathbf{q}}_1 + \frac{\partial T_2}{\partial \mathbf{q}} + J_2 P_1 u$$

$$J_2 P_1 u = -\lambda_2 T_2 - J_2 \dot{\mathbf{q}}_1 - \frac{\partial T_2}{\partial \mathbf{q}}$$



## Controlling the second task

We have

$$\dot{\mathbf{q}} = \dot{\mathbf{q}}_1 + P_1 u$$

$$\dot{T}_2 = J_2 \dot{\mathbf{q}} + \frac{\partial T_2}{\partial \mathbf{q}}$$

$$\dot{T}_2 = J_2 \dot{\mathbf{q}}_1 + \frac{\partial T_2}{\partial \mathbf{q}} + J_2 P_1 u$$

We want

$$\dot{T}_2 = -\lambda_2 T_2$$

Thus

$$-\lambda_2 T_2 = J_2 \dot{\mathbf{q}}_1 + \frac{\partial T_2}{\partial \mathbf{q}} + J_2 P_1 u$$

$$J_2 P_1 u = -\lambda_2 T_2 - J_2 \dot{\mathbf{q}}_1 - \frac{\partial T_2}{\partial \mathbf{q}}$$

## Controlling the second task

Thus

$$-\lambda_2 T_2 = J_2 \dot{\mathbf{q}}_1 + \frac{\partial T_2}{\partial \mathbf{q}} + J_2 P_1 \mathbf{u}$$

$$J_2 P_1 \mathbf{u} = -\lambda_2 T_2 - J_2 \dot{\mathbf{q}}_1 - \frac{\partial T_2}{\partial \mathbf{q}}$$

$$\mathbf{u} = -(J_2 P_1)^+ (\lambda_2 T_2 + J_2 \dot{\mathbf{q}}_1 + \frac{\partial T_2}{\partial \mathbf{q}})$$

$$\dot{\mathbf{q}}_2 \triangleq \dot{\mathbf{q}}_1 + P_1 \mathbf{u}$$

$$= \dot{\mathbf{q}}_1 - P_1 (J_2 P_1)^+ (\lambda_2 T_2 + J_2 \dot{\mathbf{q}}_1 + \frac{\partial T_2}{\partial \mathbf{q}})$$

minimizes  $\|\dot{T}_2 + \lambda_2 T_2\|$  over  $\dot{\mathbf{q}}_1 + \text{Ker } J_1$ .

## Controlling the second task

Thus

$$-\lambda_2 T_2 = J_2 \dot{\mathbf{q}}_1 + \frac{\partial T_2}{\partial \mathbf{q}} + J_2 P_1 u$$

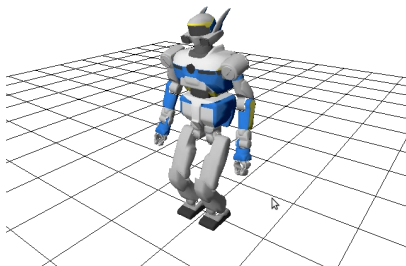
$$J_2 P_1 u = -\lambda_2 T_2 - J_2 \dot{\mathbf{q}}_1 - \frac{\partial T_2}{\partial \mathbf{q}}$$

$$u = -(J_2 P_1)^+ (\lambda_2 T_2 + J_2 \dot{\mathbf{q}}_1 + \frac{\partial T_2}{\partial \mathbf{q}})$$

$$\begin{aligned} \dot{\mathbf{q}}_2 &\triangleq \dot{\mathbf{q}}_1 + P_1 u \\ &= \dot{\mathbf{q}}_1 - P_1 (J_2 P_1)^+ (\lambda_2 T_2 + J_2 \dot{\mathbf{q}}_1 + \frac{\partial T_2}{\partial \mathbf{q}}) \end{aligned}$$

minimizes  $\|\dot{T}_2 + \lambda_2 T_2\|$  over  $\dot{\mathbf{q}}_1 + \text{Ker } J_1$ .

## Example



- ▶  $T_1$ : position of the feet + projection of center of mass,
- ▶  $T_2$ : position of the right wrist.

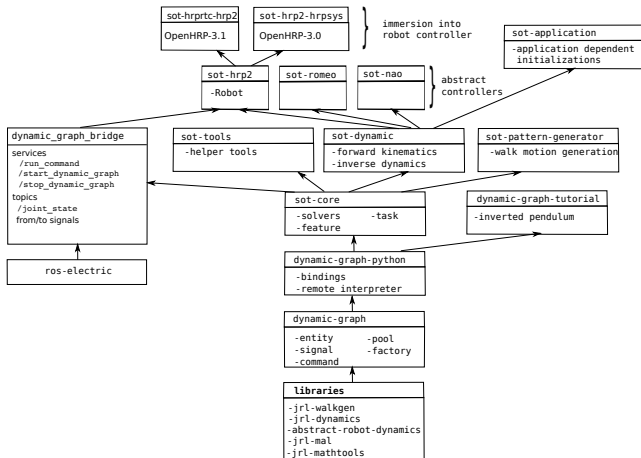
# Outline

Introduction

Theoretical foundations

Software

# Architecture overview



# Libraries

- ▶ `jrl-mathtools`: implementation of small size matrices,
  - ▶ to be replaced by Eigen
- ▶ `jrl-mal`: abstract layer for matrices,
  - ▶ to be replaced by Eigen
- ▶ `abstract-robot-dynamics`: abstraction for humanoid robot description,
- ▶ `jrl-dynamics`: implementation of the above abstract interfaces,
- ▶ `jrl-walkgen`: ZMP based dynamic walk generation.

# Libraries

- ▶ `jrl-mathtools`: implementation of small size matrices,
  - ▶ to be replaced by Eigen
- ▶ `jrl-mal`: **abstract layer for matrices**,
  - ▶ to be replaced by Eigen
- ▶ `abstract-robot-dynamics`: abstraction for humanoid robot description,
- ▶ `jrl-dynamics`: implementation of the above abstract interfaces,
- ▶ `jrl-walkgen`: ZMP based dynamic walk generation.



# Libraries

- ▶ `jrl-mathtools`: implementation of small size matrices,
  - ▶ to be replaced by Eigen
- ▶ `jrl-mal`: **abstract layer for matrices**,
  - ▶ to be replaced by Eigen
- ▶ `abstract-robot-dynamics`: **abstraction for humanoid robot description**,
- ▶ `jrl-dynamics`: implementation of the above abstract interfaces,
- ▶ `jrl-walkgen`: ZMP based dynamic walk generation.

# Libraries

- ▶ `jrl-mathtools`: implementation of small size matrices,
  - ▶ to be replaced by Eigen
- ▶ `jrl-mal`: **abstract** layer for matrices,
  - ▶ to be replaced by Eigen
- ▶ `abstract-robot-dynamics`: **abstraction** for humanoid robot description,
- ▶ `jrl-dynamics`: implementation of the above abstract interfaces,
- ▶ `jrl-walkgen`: ZMP based dynamic walk generation.

# Libraries

- ▶ `jrl-mathtools`: implementation of small size matrices,
  - ▶ to be replaced by Eigen
- ▶ `jrl-mal`: **abstract** layer for matrices,
  - ▶ to be replaced by Eigen
- ▶ `abstract-robot-dynamics`: **abstraction** for humanoid robot description,
- ▶ `jrl-dynamics`: implementation of the above abstract interfaces,
- ▶ `jrl-walkgen`: **ZMP** based dynamic walk generation.

# dynamic-graph

- ▶ Entity
  - ▶ Signal: synchronous interface
  - ▶ Command: asynchronous interface
- ▶ Factory
  - ▶ builds a new entity of requested type,
  - ▶ new entity types can be dynamically added (advanced).
- ▶ Pool
  - ▶ stores all instances of entities,
  - ▶ return reference to entity of given name.

# dynamic-graph

- ▶ Entity
  - ▶ Signal: synchronous interface
  - ▶ Command: asynchronous interface
- ▶ Factory
  - ▶ builds a new entity of requested type,
  - ▶ new entity types can be dynamically added (advanced).
- ▶ Pool
  - ▶ stores all instances of entities,
  - ▶ return reference to entity of given name.

# dynamic-graph

- ▶ Entity
  - ▶ Signal: synchronous interface
  - ▶ Command: asynchronous interface
- ▶ Factory
  - ▶ builds a new entity of requested type,
  - ▶ new entity types can be dynamically added (advanced).
- ▶ Pool
  - ▶ stores all instances of entities,
  - ▶ return reference to entity of given name.

# Signal

## Synchronous interface storing a given data type

- ▶ output signals:
  - ▶ recomputed by a callback function, or
  - ▶ set to constant value
  - ▶ **warning**: setting to constant value deactivate callback,
- ▶ input signals:
  - ▶ plugged by an output signal, or
  - ▶ set to constant value,
  - ▶ **warning**: setting to constant value unplugs,

# Signal

## Synchronous interface storing a given data type

- ▶ output signals:
  - ▶ recomputed by a callback function, or
  - ▶ set to constant value
  - ▶ **warning**: setting to constant value deactivate callback,
- ▶ input signals:
  - ▶ plugged by an output signal, or
  - ▶ set to constant value,
  - ▶ **warning**: setting to constant value unplugs,



# Signal

## Synchronous interface storing a given data type

- ▶ output signals:
  - ▶ recomputed by a callback function, or
  - ▶ set to constant value
  - ▶ **warning**: setting to constant value deactivate callback,
- ▶ input signals:
  - ▶ plugged by an output signal, or
  - ▶ set to constant value,
  - ▶ **warning**: setting to constant value unplugs,

# Signal

## Synchronous interface storing a given data type

- ▶ output signals:
  - ▶ recomputed by a callback function, or
  - ▶ set to constant value
  - ▶ **warning**: setting to constant value deactivate callback,
- ▶ input signals:
  - ▶ plugged by an output signal, or
  - ▶ set to constant value,
  - ▶ **warning**: setting to constant value unplugs,

# Signal

## Synchronous interface storing a given data type

- ▶ dependency relation:  $s_1$  depends on  $s_2$  if  $s_1$  callback needs the value of  $s_2$ ,
- ▶ each signal  $s$  stores time of last recomputation in member  $s.t_*$
- ▶  $s$  is said outdated at time  $t$  if
  - ▶  $t > s.t_*$ , and
  - ▶ one dependency  $s\_dep$  of  $s$ 
    - ▶ is out-dated or
    - ▶ has been recomputed later than  $s$ :  $s\_dep.t_* > s.t_*$ .
- ▶ reading an out-dated signal triggers recomputation.
- ▶ New types can be dynamically added (advanced)

# Signal

## Synchronous interface storing a given data type

- ▶ dependency relation:  $s_1$  depends on  $s_2$  if  $s_1$  callback needs the value of  $s_2$ ,
- ▶ each signal  $s$  stores time of last recomputation in member  $s.t_$
- ▶  $s$  is said outdated at time  $t$  if
  - ▶  $t > s.t_$ , and
  - ▶ one dependency  $s\_dep$  of  $s$ 
    - ▶ is out-dated or
    - ▶ has been recomputed later than  $s$ :  $s\_dep.t_ > s.t_$ .
- ▶ reading an out-dated signal triggers recomputation.
- ▶ New types can be dynamically added (advanced)

# Signal

## Synchronous interface storing a given data type

- ▶ dependency relation:  $s_1$  depends on  $s_2$  if  $s_1$  callback needs the value of  $s_2$ ,
- ▶ each signal  $s$  stores time of last recomputation in member  $s.t_*$
- ▶  $s$  is said outdated at time  $t$  if
  - ▶  $t > s.t_*$ , and
  - ▶ one dependency  $s\_dep$  of  $s$ 
    - ▶ is out-dated or
    - ▶ has been recomputed later than  $s$ :  $s\_dep.t_* > s.t_*$ .
- ▶ reading an out-dated signal triggers recomputation.
- ▶ New types can be dynamically added (advanced)

# Signal

## Synchronous interface storing a given data type

- ▶ dependency relation:  $s_1$  depends on  $s_2$  if  $s_1$  callback needs the value of  $s_2$ ,
- ▶ each signal  $s$  stores time of last recomputation in member  $s.t_*$
- ▶  $s$  is said outdated at time  $t$  if
  - ▶  $t > s.t_*$ , and
  - ▶ one dependency  $s_{dep}$  of  $s$ 
    - ▶ is out-dated or
    - ▶ has been recomputed later than  $s$ :  $s_{dep}.t_* > s.t_*$
- ▶ reading an out-dated signal triggers recomputation.
- ▶ New types can be dynamically added (advanced)

# Signal

## Synchronous interface storing a given data type

- ▶ dependency relation:  $s_1$  depends on  $s_2$  if  $s_1$  callback needs the value of  $s_2$ ,
- ▶ each signal  $s$  stores time of last recomputation in member  $s.t_$
- ▶  $s$  is said outdated at time  $t$  if
  - ▶  $t > s.t_$ , and
  - ▶ one dependency  $s\_dep$  of  $s$ 
    - ▶ is out-dated or
    - ▶ has been recomputed later than  $s$ :  $s\_dep.t_ > s.t_$ .
- ▶ reading an out-dated signal triggers recomputation.
- ▶ New types can be dynamically added (advanced)

# Signal

## Synchronous interface storing a given data type

- ▶ dependency relation:  $s_1$  depends on  $s_2$  if  $s_1$  callback needs the value of  $s_2$ ,
- ▶ each signal  $s$  stores time of last recomputation in member  $s.t_$
- ▶  $s$  is said outdated at time  $t$  if
  - ▶  $t > s.t_$ , and
  - ▶ one dependency  $s\_dep$  of  $s$ 
    - ▶ is out-dated or
    - ▶ has been recomputed later than  $s$ :  $s\_dep.t_ > s.t_$ .
- ▶ reading an out-dated signal triggers recomputation.
- ▶ New types can be dynamically added (advanced)



# Command

## Asynchronous interface

- ▶ input in a fixed set of types,
- ▶ trigger an action,
- ▶ returns a result in the same set of types.

# dynamic-graph-python

## Python bindings to dynamic-graph

- ▶ module `dynamic_graph` linked to `libdynamic-graph.so`
  - ▶ class `Entity`
    - ▶ each C++ entity class declared in the factory generates a python class of the same name,
    - ▶ signals are instance members,
    - ▶ commands are bound to instance methods
    - ▶ method `help` lists commands
    - ▶ method `displaySignals` displays signals
  - ▶ class `Signal`
    - ▶ property `value` to set and get signal value
- ▶ remote interpreter to be embedded into a robot controller (advanced)

# dynamic-graph-python

## Python bindings to dynamic-graph

- ▶ **module** `dynamic_graph` linked to `libdynamic-graph.so`
  - ▶ **class** `Entity`
    - ▶ each C++ entity class declared in the factory generates a python class of the same name,
    - ▶ signals are instance members,
    - ▶ commands are bound to instance methods
    - ▶ method `help` lists commands
    - ▶ method `displaySignals` displays signals
  - ▶ **class** `Signal`
    - ▶ property `value` to set and get signal value
- ▶ remote interpreter to be embedded into a robot controller (advanced)

# dynamic-graph-python

## Python bindings to `dynamic-graph`

- ▶ module `dynamic_graph` linked to `libdynamic-graph.so`
  - ▶ class `Entity`
    - ▶ each C++ entity class declared in the factory generates a python class of the same name,
    - ▶ signals are instance members,
    - ▶ commands are bound to instance methods
    - ▶ method `help` lists commands
    - ▶ method `displaySignals` displays signals
  - ▶ class `Signal`
    - ▶ property `value` to set and get signal value
- ▶ remote interpreter to be embedded into a robot controller (advanced)

# dynamic-graph-python

## Python bindings to `dynamic-graph`

- ▶ module `dynamic_graph` linked to `libdynamic-graph.so`
  - ▶ class `Entity`
    - ▶ each C++ entity class declared in the factory generates a python class of the same name,
    - ▶ signals are instance members,
    - ▶ commands are bound to instance methods
    - ▶ method `help` lists commands
    - ▶ method `displaySignals` displays signals
  - ▶ class `Signal`
    - ▶ property `value` to set and get signal value
- ▶ remote interpreter to be embedded into a robot controller (advanced)

# dynamic-graph-python

## Python bindings to `dynamic-graph`

- ▶ module `dynamic_graph` linked to `libdynamic-graph.so`
  - ▶ class `Entity`
    - ▶ each C++ entity class declared in the factory generates a python class of the same name,
    - ▶ signals are instance members,
    - ▶ commands are bound to instance methods
    - ▶ method `help` lists commands
    - ▶ method `displaySignals` displays signals
  - ▶ class `Signal`
    - ▶ property `value` to set and get signal value
- ▶ remote interpreter to be embedded into a robot controller (advanced)

# dynamic-graph-tutorial

## Simple use case for illustration

- ▶ Definition of 2 entity types
  - ▶ InvertedPendulum
    - ▶ input signal: force
    - ▶ output signal: state
  - ▶ FeedbackController
    - ▶ input signal: state
    - ▶ output signal: force

# dynamic-graph-tutorial

```
>>> from dynamic_graph.tutorial import InvertedPendulum, FeedbackController
>>> a = InvertedPendulum ('IP')
>>> b = FeedbackController ('K')
>>> a.displaySignals ()
--- <IP> signal list:
|-- <Sig:InvertedPendulum(IP)::input(double)::force (Type Cst) AUTOPLUGGED
'-- <Sig:InvertedPendulum(IP)::output(vector)::state (Type Cst)
>>> a.help ()
Classical inverted pendulum dynamic model
```

List of commands:

```
-----
getCartMass:           Get cart mass
getPendulumLength:     Get pendulum length
getPendulumMass:       Get pendulum mass
incr:                  Integrate dynamics for time step provided as input
setCartMass:           Set cart mass
setPendulumLength:     Set pendulum length
setPendulumMass:       Set pendulum mass
>>> a.help ('incr')
incr:
```

Integrate dynamics for time step provided as input

take one floating point number as input

```
>>>
```



# dynamic-graph-tutorial

```
>>> from dynamic_graph.tutorial import InvertedPendulum, FeedbackController
>>> a = InvertedPendulum ('IP')
>>> b = FeedbackController ('K')
>>> a.displaySignals ()
--- <IP> signal list:
|-- <Sig:InvertedPendulum(IP)::input(double)::force (Type Cst) AUTOPLUGGED
'-- <Sig:InvertedPendulum(IP)::output(vector)::state (Type Cst)
>>> a.help ()
Classical inverted pendulum dynamic model
```

List of commands:

```
-----
getCartMass:           Get cart mass
getPendulumLength:     Get pendulum length
getPendulumMass:       Get pendulum mass
incr:                  Integrate dynamics for time step provided as input
setCartMass:           Set cart mass
setPendulumLength:     Set pendulum length
setPendulumMass:       Set pendulum mass
>>> a.help ('incr')
incr:
```

Integrate dynamics for time step provided as input

take one floating point number as input

```
>>>
```

# dynamic-graph-tutorial

```
>>> from dynamic_graph.tutorial import InvertedPendulum, FeedbackController
>>> a = InvertedPendulum ('IP')
>>> b = FeedbackController ('K')
>>> a.displaySignals ()
--- <IP> signal list:
|-- <Sig:InvertedPendulum(IP)::input(double)::force (Type Cst) AUTOPLUGGED
'-- <Sig:InvertedPendulum(IP)::output(vector)::state (Type Cst)
>>> a.help ()
Classical inverted pendulum dynamic model
```

List of commands:

```
-----
getCartMass:           Get cart mass
getPendulumLength:     Get pendulum length
getPendulumMass:       Get pendulum mass
incr:                  Integrate dynamics for time step provided as input
setCartMass:           Set cart mass
setPendulumLength:     Set pendulum length
setPendulumMass:       Set pendulum mass
>>> a.help ('incr')
incr:
```

Integrate dynamics for time step provided as input

take one floating point number as input

```
>>>
```

# dynamic-graph-tutorial

```
>>> from dynamic_graph.tutorial import InvertedPendulum, FeedbackController
>>> a = InvertedPendulum ('IP')
>>> b = FeedbackController ('K')
>>> a.displaySignals ()
--- <IP> signal list:
|-- <Sig:InvertedPendulum(IP)::input(double)::force (Type Cst) AUTOPLUGGED
'-- <Sig:InvertedPendulum(IP)::output(vector)::state (Type Cst)
>>> a.help ()
Classical inverted pendulum dynamic model
```

List of commands:

```
-----
getCartMass:           Get cart mass
getPendulumLength:     Get pendulum length
getPendulumMass:       Get pendulum mass
incr:                  Integrate dynamics for time step provided as input
setCartMass:           Set cart mass
setPendulumLength:     Set pendulum length
setPendulumMass:       Set pendulum mass
>>> a.help ('incr')
incr:
```

Integrate dynamics for time step provided as input

take one floating point number as input

```
>>>
```

# dynamic-graph-tutorial

```
>>> from dynamic_graph.tutorial import InvertedPendulum, FeedbackController
>>> a = InvertedPendulum ('IP')
>>> b = FeedbackController ('K')
>>> a.displaySignals ()
--- <IP> signal list:
|-- <Sig:InvertedPendulum(IP)::input(double)::force (Type Cst) AUTOPLUGGED
'-- <Sig:InvertedPendulum(IP)::output(vector)::state (Type Cst)
>>> a.help ()
Classical inverted pendulum dynamic model
```

List of commands:

```
-----
getCartMass:           Get cart mass
getPendulumLength:     Get pendulum length
getPendulumMass:       Get pendulum mass
incr:                  Integrate dynamics for time step provided as input
setCartMass:           Set cart mass
setPendulumLength:     Set pendulum length
setPendulumMass:       Set pendulum mass
>>> a.help ('incr')
incr:
```

Integrate dynamics for time step provided as input

take one floating point number as input

```
>>>
```

# dynamic-graph-tutorial

## Package provides

- ▶ **C++ code of classes** `InvertedPendulum` and `FeedbackController`,
- ▶ explanation about how to create a new entity type in C++,
- ▶ information about how to create a command in C++,
- ▶ information about how to create a python module defining the bindings in cmake,
- ▶ python script that runs an example.

# dynamic-graph-tutorial

## Package provides

- ▶ **C++ code of classes** `InvertedPendulum` and `FeedbackController`,
- ▶ **explanation about how to create a new entity type in C++**,
- ▶ information about how to create a command in C++,
- ▶ information about how to create a python module defining the bindings in cmake,
- ▶ python script that runs an example.

# dynamic-graph-tutorial

## Package provides

- ▶ **C++ code of classes** `InvertedPendulum` and `FeedbackController`,
- ▶ **explanation about how to create a new entity type in C++**,
- ▶ **information about how to create a command in C++**,
- ▶ information about how to create a python module defining the bindings in cmake,
- ▶ python script that runs an example.

# dynamic-graph-tutorial

## Package provides

- ▶ C++ code of classes `InvertedPendulum` and `FeedbackController`,
- ▶ explanation about how to create a new entity type in C++,
- ▶ information about how to create a command in C++,
- ▶ information about how to create a python module defining the bindings in cmake,
- ▶ python script that runs an example.



# dynamic-graph-tutorial

## Package provides

- ▶ C++ code of classes `InvertedPendulum` and `FeedbackController`,
- ▶ explanation about how to create a new entity type in C++,
- ▶ information about how to create a command in C++,
- ▶ information about how to create a python module defining the bindings in cmake,
- ▶ python script that runs an example.

# sot-core

## Class FeatureAbstract

- ▶ function of the robot and environment states
  - ▶ position of an end-effector,
  - ▶ position of a feature in an image (visual servoing)
- ▶ with values in a Lie group  $G$  ( $SO(3)$ ,  $SE(3)$ ,  $\mathbb{R}^n, \dots$ ),
- ▶ with a mapping  $e$  from  $G$  into  $\mathbb{R}^m$  such that

$$e(0_G) = 0$$

# sot-core

## Class FeatureAbstract

- ▶ function of the robot and environment states
  - ▶ position of an end-effector,
  - ▶ position of a feature in an image (visual servoing)
- ▶ with values in a Lie group  $G$  ( $SO(3)$ ,  $SE(3)$ ,  $\mathbb{R}^n, \dots$ ),
- ▶ with a mapping  $e$  from  $G$  into  $\mathbb{R}^m$  such that

$$e(0_G) = 0$$

# sot-core

## Class FeatureAbstract

- ▶ function of the robot and environment states
  - ▶ position of an end-effector,
  - ▶ position of a feature in an image (visual servoing)
- ▶ with values in a Lie group  $G$  ( $SO(3)$ ,  $SE(3)$ ,  $\mathbb{R}^n$ ,...),
- ▶ with a mapping  $e$  from  $G$  into  $\mathbb{R}^m$  such that

$$e(0_G) = 0$$

# sot-core

## Class FeatureAbstract

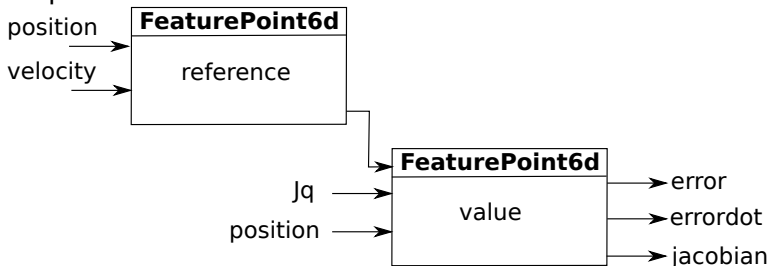
- ▶ function of the robot and environment states
  - ▶ position of an end-effector,
  - ▶ position of a feature in an image (visual servoing)
- ▶ with values in a Lie group  $G$  ( $SO(3)$ ,  $SE(3)$ ,  $\mathbb{R}^n, \dots$ ),
- ▶ with a mapping  $e$  from  $G$  into  $\mathbb{R}^m$  such that

$$e(0_G) = 0$$

# Feature

When paired with a reference, features become *tasks*.

► Example

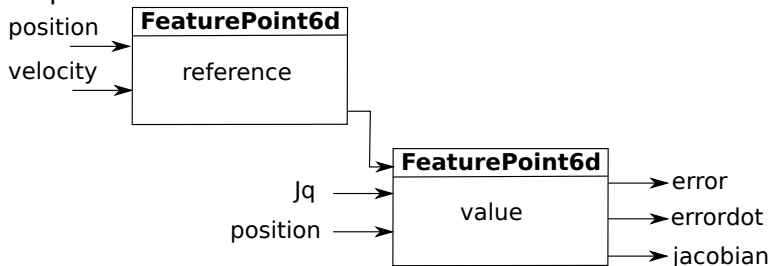


- $\text{error} = e(\text{value.position} \ominus \text{reference.position})$
- **error** $\dot{}$ : derivative of error when **value.position** is constant.

# Feature

When paired with a reference, features become *tasks*.

- ▶ Example

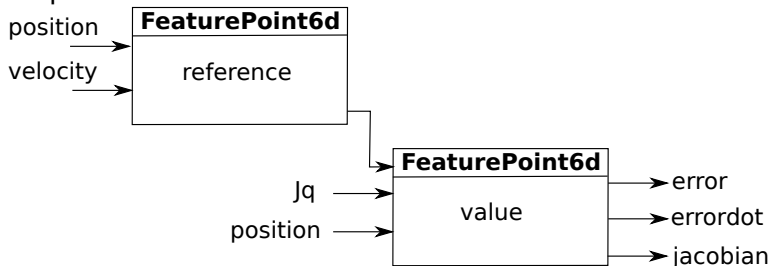


- ▶  $\text{error} = \mathbf{e}(\text{value.position} \ominus \text{reference.position})$
- ▶ `error dot`: derivative of error when `value.position` is constant.

# Feature

When paired with a reference, features become *tasks*.

- ▶ Example

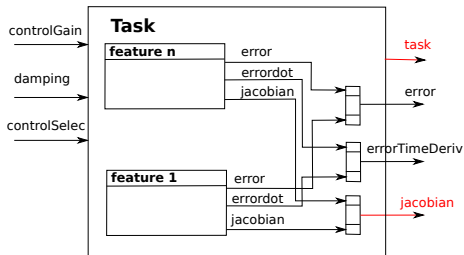


- ▶  $\text{error} = \mathbf{e}(\text{value.position} \ominus \text{reference.position})$
- ▶ **errordot**: derivative of error when `value.position` is constant.



# Task

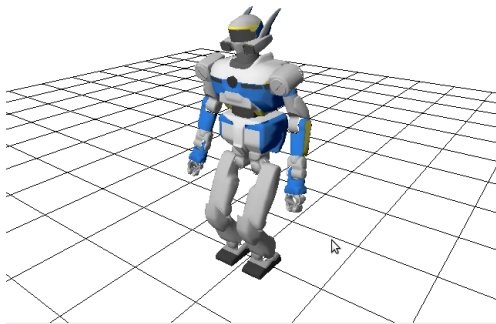
- ▶ Collection of features with a control gain,
- ▶ implements abstraction `TaskAbstract`



- ▶ `task = -controlGain.error`

# Solver SOT

## Hierarchical task solver



- computes robot joint velocity

## sot-dynamic

`dynamic_graph.sot.dynamics.Dynamic` builds a kinematic chain from a file and

- ▶ computes forward kinematics
  - ▶ position and Jacobian of end effectors (wrists, ankles),
  - ▶ position of center of mass
- ▶ computes dynamics
  - ▶ inertia matrix.

# sot-pattern-generator

`dynamic_graph.sot.pattern_generator`

- ▶ **Entity** `PatternGenerator` produces walk motions as
  - ▶ position and velocity of the feet
  - ▶ position and velocity of the center of mass

# sot-application

`dynamic_graph.sot.application`

- ▶ Provide scripts for standard control graph initialization
  - ▶ depends on application: control mode (velocity, acceleration)

## Packages specific to robots

`sot-hrp2`

- ▶ defines a class `Robot` that provides
  - ▶ ready to use features for feet, hands, gaze and center of mass,
  - ▶ ready to use tasks for the same end effectors,
  - ▶ an entity `Dynamic`,
  - ▶ an entity `Device` (interface with the robot control system)

`sot-hrprtc-hrp2`

- ▶ provide an RTC component to integrate `sot-hrp2` into the robot controller.

# Utilities

- ▶ `dynamic_graph.writeGraph (filename)`: **writes the current graph in a file using graphviz dot format.**
- ▶ `dynamic_graph.sot.core.FeaturePosition` wraps two `FeaturePoint6d`: a value and a reference,
- ▶ `MetaTask6d`:
- ▶ `MetaTaskPosture`:
- ▶ `MetaTaskKine6d`:
- ▶ `MetaTaskKinePosture`:
- ▶ `MetaTaskCom`:

# Utilities

- ▶ `dynamic_graph.writeGraph (filename)`: **writes the current graph in a file using graphviz dot format.**
- ▶ `dynamic_graph.sot.core.FeaturePosition` **wraps two** `FeaturePoint6d`: **a value and a reference,**
- ▶ `MetaTask6d`:
- ▶ `MetaTaskPosture`:
- ▶ `MetaTaskKine6d`:
- ▶ `MetaTaskKinePosture`:
- ▶ `MetaTaskCom`:



# Utilities

- ▶ `dynamic_graph.writeGraph (filename)`: **writes the current graph in a file using graphviz dot format.**
- ▶ `dynamic_graph.sot.core.FeaturePosition` **wraps two** `FeaturePoint6d`: **a value and a reference,**
- ▶ `MetaTask6d`:
- ▶ `MetaTaskPosture`:
- ▶ `MetaTaskKine6d`:
- ▶ `MetaTaskKinePosture`:
- ▶ `MetaTaskCom`:

# Utilities

- ▶ `dynamic_graph.writeGraph (filename)`: **writes the current graph in a file using graphviz dot format.**
- ▶ `dynamic_graph.sot.core.FeaturePosition` **wraps two** `FeaturePoint6d`: **a value and a reference,**
- ▶ `MetaTask6d`:
- ▶ `MetaTaskPosture`:
- ▶ `MetaTaskKine6d`:
- ▶ `MetaTaskKinePosture`:
- ▶ `MetaTaskCom`:

# Utilities

- ▶ `dynamic_graph.writeGraph (filename)`: **writes the current graph in a file using graphviz dot format.**
- ▶ `dynamic_graph.sot.core.FeaturePosition` **wraps two** `FeaturePoint6d`: **a value and a reference,**
- ▶ `MetaTask6d`:
- ▶ `MetaTaskPosture`:
- ▶ `MetaTaskKine6d`:
- ▶ `MetaTaskKinePosture`:
- ▶ `MetaTaskCom`:

# Utilities

- ▶ `dynamic_graph.writeGraph (filename)`: **writes the current graph in a file using graphviz dot format.**
- ▶ `dynamic_graph.sot.core.FeaturePosition` **wraps two** `FeaturePoint6d`: **a value and a reference,**
- ▶ `MetaTask6d`:
- ▶ `MetaTaskPosture`:
- ▶ `MetaTaskKine6d`:
- ▶ `MetaTaskKinePosture`:
- ▶ `MetaTaskCom`:

# Utilities

- ▶ `dynamic_graph.writeGraph (filename)`: **writes the current graph in a file using graphviz dot format.**
- ▶ `dynamic_graph.sot.core.FeaturePosition` **wraps two** `FeaturePoint6d`: **a value and a reference,**
- ▶ `MetaTask6d`:
- ▶ `MetaTaskPosture`:
- ▶ `MetaTaskKine6d`:
- ▶ `MetaTaskKinePosture`:
- ▶ `MetaTaskCom`:

# Installation

## Through robotpkg

```
▶ git clone http://trac.laas.fr/git/robots/robotpkg.git
   cd robotpkg
   ./bootstrap/bootstrap --prefix=<your_prefix>
   cd motion/sot-dynamic

   make install
```

# Installation

## Through github:

```
► git clone --recursive git://github.com/jrl-umi3218/jrl-mal.git
git clone --recursive git://github.com/jrl-umi3218/jrl-mathtools.git
git clone --recursive git://github.com/laas/abstract-robot-dynamics.git
git clone --recursive git://github.com/jrl-umi3218/jrl-dynamics.git
git clone --recursive git://github.com/jrl-umi3218/jrl-walkgen.git
git clone --recursive git://github.com/jrl-umi3218/dynamic-graph.git
git clone --recursive git://github.com/jrl-umi3218/dynamic-graph-python.git
git clone --recursive git://github.com/jrl-umi3218/sot-core.git
git clone --recursive git://github.com/laas/sot-tools.git
git clone --recursive git://github.com/jrl-umi3218/sot-dynamic.git
git clone --recursive git://github.com/jrl-umi3218/sot-pattern-generator.git
git clone --recursive git://github.com/stack-of-tasks/sot-application.git
git clone --recursive git://github.com/laas/sot-hrp2.git
git clone --recursive git://github.com/stack-of-tasks/sot-hrp2c-hrp2.git
```

## ► for each package,

```
mkdir package/build
cd package/build
cmake -DCMAKE_INSTALL_PREFIX=<your-prefix> ..

make install
```

# Installation

## Through github:

```
▶ git clone --recursive git://github.com/jrl-umi3218/jrl-mal.git
git clone --recursive git://github.com/jrl-umi3218/jrl-mathtools.git
git clone --recursive git://github.com/laas/abstract-robot-dynamics.git
git clone --recursive git://github.com/jrl-umi3218/jrl-dynamics.git
git clone --recursive git://github.com/jrl-umi3218/jrl-walkgen.git
git clone --recursive git://github.com/jrl-umi3218/dynamic-graph.git
git clone --recursive git://github.com/jrl-umi3218/dynamic-graph-python.git
git clone --recursive git://github.com/jrl-umi3218/sot-core.git
git clone --recursive git://github.com/laas/sot-tools.git
git clone --recursive git://github.com/jrl-umi3218/sot-dynamic.git
git clone --recursive git://github.com/jrl-umi3218/sot-pattern-generator.git
git clone --recursive git://github.com/stack-of-tasks/sot-application.git
git clone --recursive git://github.com/laas/sot-hrp2.git
git clone --recursive git://github.com/stack-of-tasks/sot-hrp2c-hrp2.git
```

## ▶ for each package,

```
mkdir package/build
cd package/build
cmake -DCMAKE_INSTALL_PREFIX=<your-prefix> ..

make install
```



# Installation

## Through installation script

```
▶ git clone git://github.com/stack-of-tasks/install-sot.git  
   cd install-sot/scripts  
  
   ./install_sot.sh
```

# Running the stack of tasks into OpenHRP-3.1

You need to install:

- ▶ `ros-electric`
- ▶ `OpenHRP-3.1`

you will find instructions in <https://wiki.laas.fr/robots/HRP/Software>

Then follow instructions in [sot-hrp2c/README.md](https://github.com/stack-of-tasks/sot-hrp2c-hrp2):

<https://github.com/stack-of-tasks/sot-hrp2c-hrp2>

# Running the stack of tasks into OpenHRP-3.0.7

## Assumptions

- ▶ OpenHRP 3.0.7 is installed
- ▶ The Stack of Tasks has been installed thanks to previous slide with **install\_sot.sh** in the directory:

```
/home/user/devel/ros_unstable
```

- ▶ Your `/opt/grx3.0/HRP2LAAS/bin/config.sh` is well setup.

## The golden commands

```
$>roscore  
#Launching HRP2 simulation with OpenHRP  
$>roslaunch hrp2_bringup openhrp_bridge.launch robot_machine_profile:=sim  
$>rosservice call /start_dynamic_graph  
$>roslaunch dynamic_graph_bridge run.command
```

# Running the stack of tasks into OpenHRP-3.0.7

## Initialize the application: create tracer and solver

```
[INFO] [WallTime: 1370854858.786392] waiting for  
service ...  
Interacting with remote server.  
>>> from dynamic_graph.sot.application.velocity.\  
precomputed_tasks import initialize  
>>> solver = initialize (robot)  
>>> robot.initializeTracer ()
```

# Running the stack of tasks into OpenHRP-3.0.7

## Build the graph including the pattern generator

```
[INFO] [WallTime: 1370854858.786392] waiting for
    service ...
Interacting with remote server.
>>> from
    dynamic_graph.sot.pattern_generator.walking
import CreateEverythingForPG, walkFewSteps
With meta selector
```

# Running the stack of tasks into OpenHRP-3.0.7

## Create the graph

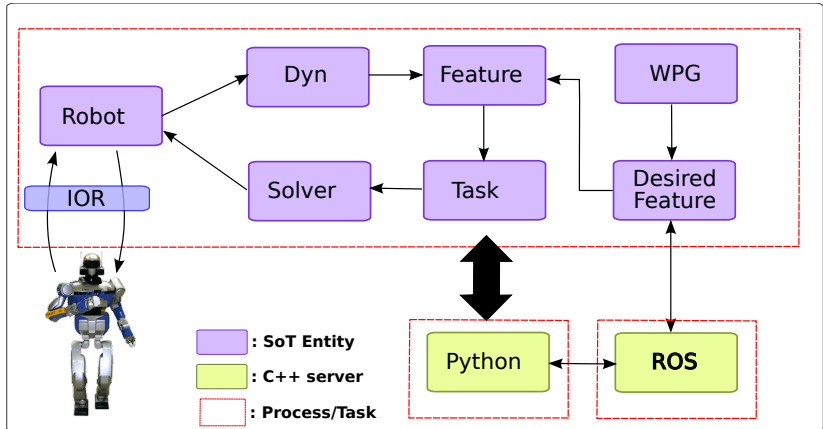
```
>>> CreateEverythingForPG(robot, solver)
At this stage
('modelDir:',
 '~/.devel/ros-unstable/install/share/hrp2-14')
('modelName:', 'HRP2JRLmainsmall.wrl')
('specificitiesPath:',
 'HRP2SpecificitiesSmall.xml')
('jointRankPath:', 'HRP2LinkJointRankSmall.xml')
After Task for Right and Left Feet
```

# Running the stack of tasks into OpenHRP-3.0.7

Switch to the new graph

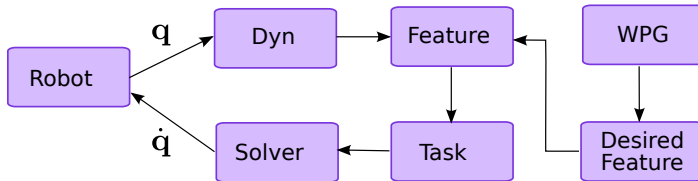
```
>>> walkFewSteps ( robot )  
>>>
```

# Software structure - Conceptual view

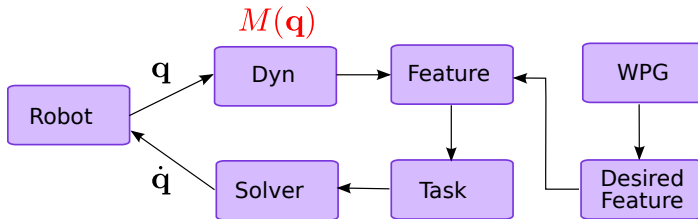




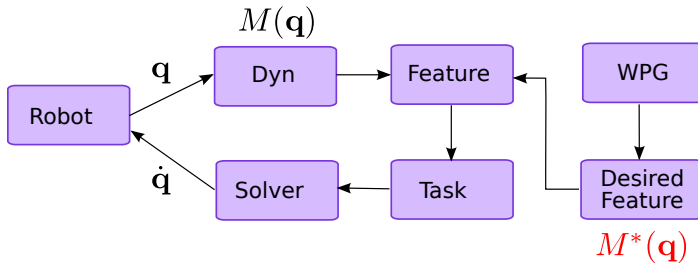
# Software structure - Link with Model



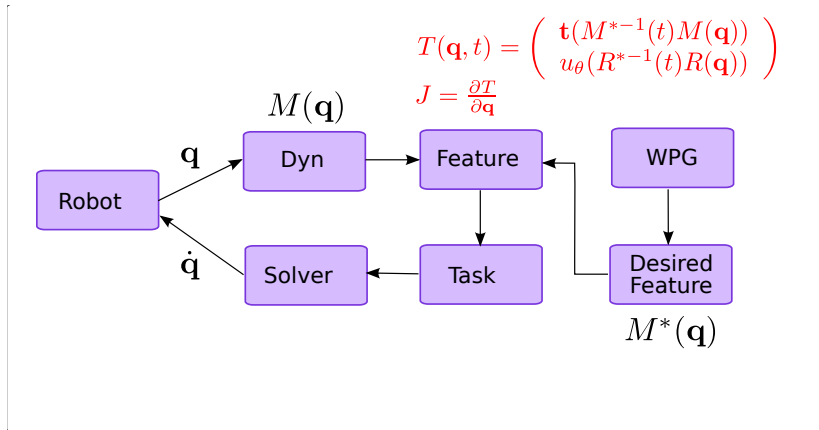
## Software structure - Link with Model



# Software structure - Link with Model



# Software structure - Link with Model



# Software structure - Link with Model

# Software structure - Link with Model

# Software structure - Repositories