ROBOOP
A Robotics Object Oriented Package in C++
version 1.31

Documentation

Richard Gourdeau
Département de génie électrique
École Polytechnique de Montréal
C.P. 6079, Succ. Centre-Ville,
Montréal, Québec, Canada, H3C 3A7

Contents

make

or

Upgraded the matrix library to NEWMAT11 (beta) April 2006 enabling compilation under GNU g++ 4.1.s.

- **Removed class** `mlink`**. DH and modified DH pagametegs age now included in** `link`**.**
- **Added** `kine_pd()`**.**
- **Cgeated a new** `torque` **member function that allowed to have load on last link.**
- **Fixed bug in modified DH dynamics.**
- **Added a class** `Quaternion`**.**
- **Added the pgoggam** `rtest` **to compage gesults with Peter Cogke MATLAB toolbox.**
- **Added member function** `set_plot2d` **to genegate plots using the** `Plot2d` **class.**
- **Added utility class** `IO_matrix_file` **dealing with data files (not documented yet).**

e on , **(2002/08/09) Moved the aggays of** `ColumnVector` **to the con-**

o ce		**the ROBOOP proTram source directory**
CMakeLists	txt	**ConfTuratcton file for CMake**
robot	h	**header file**
clik	h	**header file for** CLIK
config	h	

con **configuration files directory**
pd_

irotk

Syntax

```
ReturnMatrix irotk(const Matrix & R);
```

Description

Given a homogeneous transform matrix R, this function returns a column vector

$$\begin{bmatrix} \mathbf{k} \\ \theta \end{bmatrix} \tag{2.5}$$

with **k** a unit vector such that the 3×3 rotation bloc of the matrix

$$\mathrm{Rot}(\mathbf{k}, \theta) \tag{2.6}$$

is equal to the 3×3 rotation bloc of the matrix R.

Return Value

ColumnVector.

irpy

Syntax

(2.7)

```
ReturnMatrix irpy(const Matrix & R);
```

of the matrix

Description

Given a homogeneous transform matrix R**, this function returns a column vector**

β

α

rotk

yn

```
ReturnMatrix rotk(const Real theta,
                  const ColumnVector & k);
```

Dec on

This function returns the matrix of a rotation of an angle `theta` **around the vector**

rpy

yn

```
ReturnMatrix rpy(constColumnVector&a);
```

De c on

Giv a

rotx, roty, rotz

yn¿

```
ReturnMatrix rotx(const Real alpha);
```

2.2 The `Quaternion` class

The `Quaternion`

operators

yn

```
Quaternion   operator+(const Quaternion & q)const;
Quaternion   operator-(const Quaternion & q)const;
Quaternion   operator*(const Quaternion & q)const;
Quaternion   operator*(const ColumnVector & vec)const;
Quaternion   operator*(constReal c)const;
Quaternion   operator/(const Quaternion & q)const;
Quaternion   operator/(constReal c)const;
```

De c on

tRT

conjugate and inverse

$\mathbf{yn}_{\wr}$

quaternion time derivative

yn

```
Quaternion   dot(const ColumnVector & w, const short sign)const;
ReturnMatrix E(const short sign)const;
```

De c on

The quaternion time de&ivative is obtain f om the quate&nio

unit and norm

Rotation matrices

Omega, ω

$\mathbf{yn}\iota$

Squad

yn

```
Quaternion Squad(const Quaternion & p, const Quaternion & a,
                 const Quaternion & b, const Quaternion & r,
                 const Real t);
```

De c on

Squad stands for Spherical Cubic Interpolation. Squad is not a member

Squad_prime

Synopsis

```
Quaternion Squad_prime(const Quaternion & p, const Quaternion & a,
                       const Quaternion & b, const Quaternion & q,
                       const Real t);
```

Description

2.3 The `Robot` and `mRobot` classes

The `Robot` **and** `mRobot`

Table 2.1: The `Link` class data parameters

$n_{dof} \times$

constructors

get_q, get_qp, get_qpp

yn¿

```
ReturnMatrix get_q(void);
```

inv_kin

yn

```
ReturnMatrix inv_kin(const Matrix & Tobj, const int mj = 0);
ReturnMatrix inv_kin(const Matrix & Tobj, const int mj,
                     const int endlink, bool & converge);
```

De c on

The inerse kinematic model is computed using a Newton-Raphs on technique. If `mj == 0`, it is ased on the ollowing [6]:

$$
\begin{aligned}
{}^0T_n(q^*) &= {}^0T_n(q + \delta q) \approx {}^0T_n(q)\delta T(\delta q) = T_{oj} && (2.35)\\
\delta T(\delta q) &= ({}^0T_n(q))^{-1}T_{oj} = I \quad \blacktriangle && (2.36)
\end{aligned}
$$

$\blacktriangle$ =

inv_kin_

jacobian_dot

yn

```
ReturnMatrix jacobian_dot(const int ref=0);
```

De c on

The manipulator Jacobian time derivative can be used to compute the end

jacobian_DLS_inv

yn¿

2.3.3 Dynamics

The robotics manipulator dynamic model is given by (see appendix A or [4])

$$\tau = D(q)\ddot{q} + C(q,\dot{q}) + G(q) \tag{2.66}$$

inertia

yn

```
ReturnMatrix inertia(const ColumnVector & q);
```

Dec on

This function computes the robot inertia matrix $\mathbf{D}(\mathbf{q}$. A simplified RNE version computing

$$\boldsymbol{\tau} = \mathbf{D}(\mathbf{q}\ \ddot{\mathbf{q}} \tag{2.69}$$

torque_nотyeloci

yn{

G and C

2.3.4 Linearized dynamics

Murray and Neuman [13] have developed an efficient recursive linearized Newton-Euler formulation that can be used to compute (see appendix A)

dqp_torque

yn

```
void dqp_torque(const ColumnVector & q,
                const ColumnVector & qp,
                const ColumnVector & dqp,
                ColumnVector & torque,
                ColumnVector & dtorque);
```

De c on

This function computes

$$\mathbf{S}_1(\mathbf{q},\dot{\mathbf{q}})\delta\dot{\mathbf{q}} \tag{2.73}$$

e n e

None (`torque` **and** `dtorque`

dtau_dq

Syntax

```
ReturnMatrix dtau_dq(const ColumnVector & q,
                     const ColumnVector & qp,
                     const ColumnVector & qpp);
```

Description

This function computes

$$\frac{\partial \boldsymbol{\tau}}{\partial \mathbf{q}} = \mathbf{S}_2(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}) \tag{2.74}$$

Return value

Matrix

dtau_dqp

yn¿

```
ReturnMatrix dtau_dqp(const ColumnVector & q,
                      const ColumnVector & qp);
```

perturb_robot

2.5 The Spl_path class

Spl_path uses three instances of the class Spl_Cubic f986(28(21(o)0.0492351rc)-3120549(p)0.32898(a)0.

2.7 The Trajectory_Select class

set_trajectory

yn¿

```
void set_trajectory(const string & filename);
```

torque_cmd

yn¿

```
ReturnMatrix torque_cmd(Robot_basic & robot, const ColumnVector & qd,
                        const ColumnVector & qpd);
```

K_{0k}

2.10 The `Computed_torque_method` class

The Computed_torque_method class deals with the well known computed

torque_cmd

yn¿

```
ReturnMatrix torque_cmd(Robot_basic & robot, const ColumnVector & qd,
                        const ColumnVector & qpd);
```

K_d, K_p

Syntax

```
short set_Kp(const DiagonalMatrix & Kp);
short set_Kd(const DiagonalMatrix & Kd);
```

Description

These functions sets the joint position error gain matrix, K_p

torque

K_{pp}, K_{vp}, K_{po}, K_{vo}

Syntax

```
void set_Kpp(const double Kpp);
void set_Kvp(const double Kvp);
void set_Kpo(const double Kpo);
void set_Kvo(const double Kvo);
```

Description

control

yn

```
short control(const ColumnVector & pdpp, const ColumnVector & pdp,
              const ColumnVector & pd, const ColumnVector & wdp,
              const ColumnVector & wd, const Quaternion & qd,
              const ColumnVector & f, const ColumnVector & n,
              const Real dt);
```

Dec on

get_dof

yn

```
int get_dof();
```

De c on

**This function return the degree of185(f)812(f)0.304362(r)-0.210368(e)-0.234986(e)-0.234986(d)0.331218

set_control

yn

```
void set_control(const string & filename);
```

De c on

This function set the active controller.

e n e

None

2.14 The `Stewart` class

Coming soon ... (based on [17]).

read

yn¿

```
short read(const vector<Matrix> & data);
short read(const vector<Matrix> & data, const vector<string> & data_title);
```

Plot2d

addcommand

yn

```
void addcommand(const char * gcom);
```

Dec **on**

addcurve

yn¿

```
void addcurve(const Matrix & data,
              const char * label = "",
```

gnuplot

yn

```
void gnuplot(void);
```

Dec on

This function calls gnuplot with the current content of the object.

e n e

None

settitle

yn

```
void settitle(const char * t);
```

Dec ion

This function sets the title of the graph to the strin(n)148210368(i)-0.248413(n)(r)57Tf Td2584.369(t)

setxlabel

yn

```
void setxlabel(const char * t);
```

De c on

t.

e n e

None

setylabel

$\mathbf{yn}_{\{}$

set_plot2d

yn¿

```
void set_plot2d(const char *title_graph, const char *x_axis_title,
                const char *y_axis_title, const char *label, int type,
                const Matrix &xdata, const Matrix &ydata,
                int start_y, int end_y);

void set_plot2d(const char *title_graph, const char *x_axis_title,
                const char *y_axis_title, const std::vector<char *> label,
```

Reading and writing

yn

```
short read_conf();
short write_conf(const string filename, const string file_title,const int space_betwe
```

De c on

The member function read

_conf reads a configuration file (specified by con-structor). The membeonsu

select

yn

```
short select(const string section, const string parameter,
                    string & value) const;
short select(const string section, const string parameter,
                  bool & value) const;
short select(const string section, const string parameter,
                   short & value) const;
short select(const string section, const string parameter,
                 int & value) const;
short select(const string section, const string parameter,
                    double & value) const;
short select(const string section, const string parameter,
                    float & value) const;
```

De c on

These member functions are use to assign to the variable `value` **the value of the parameter** `parameter` **from section section** `section`**.**

e n e

Status, as a short int.

- **0 successful**
- **SECTION_OR_PARAMETER_DOES_NOT_EXIST**

add

yn{

```
void add(const string section, const string parameter,
                const string value);
void add(const string section, const string parameter,
              const bool value);
void add(const string section, const string parameter,
               const short value);
void add(const string section, const string parameter,
             const int value);
void add(const string section, const string parameter,
                const double value);
void add(const string sectvoid add(const string sectt dcot
```

pinv

x_prod_matrix

Syntax

```
ReturnMatrix x_prod_matrix(const ColumnVector & x);
```

Description

This function computes the cross product matrix $S(x)$ of `x` such that $S(x)y = x \times y$.

Table 2.3: Quaternion class member functions

e n on	
+, -, *, /, =	

Table 2.16: `Robot` **(and** `mRobot`**) class member functions**

Table 2.17: Miscellaneous

SAMPLE CODE THAT MAKE THE BUG APPARENT:

Chapter 4

Credits and

Chapter 5

Bibliography

[1] Jack C. K. Chou, “Quaternion kinematic and dynamic di�erential eq0.21036742(a)0.0492351(-)]TJ

[11] M. W. Walker and D. E. Orin, "E□cient dynamic computer simulation of robotic mechanisms", ASME Jour. of Dynamic Systems, Measurement, and Control, vol. 104, pp. 205–211, 1982.

Appendix A

Recursive Newton-Euler algorithms, DH notation

In order to apply the RNE as presented in [13], let us define the following variables (referenced in the i^{th} coordinate frame if applicable):

Initialize: $\mathbf{f}_{n+1} = \mathbf{n}_{n+1} = 0.$

$$\dot{\mathbf{v}}_{ci} = \bar{\bar{\mathbf{v}}}$$

i

$$\delta \mathbf{F}_i = m_i \delta \mathbf{v}_{ci} \tag{A.16}$$
$$\delta \mathbf{N}_i = \mathbf{I}_{ci} \delta \dot{\boldsymbol{\omega}}_i +$$

Appendix B

Recursive Newton-Euler

- Backward initializations for i

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have

those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another lan-

though third parties are not compelled to copy the source along with the object code.

system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that

Also add information on how to contact you by electronic cd p .0492351(c)-per
m.0492503541(i)-0.248413(l)-0.247294(.)-0.248413]TJ 16.9289-13.5539Td [(Y)83.4109(o)0.0492351(u)-2