Rigid MBS Program
for MBFB Courses only

Munich University of Applied Scienses, FK06, 2008-2010

1- Setup the MBS Data and Equations

Note: schript RMBS_init must run before!

Created Prof. Dr. O.Wallrapp - 30.09.2008
Updated Prof. Dr. O.Wallrapp, 10.05.2009-v31
Updated Prof. Dr. O.Wallrapp, 04.06.2009-v32 - plotMBS
Updated Prof. Dr. O.Wallrapp, 01.03.2010-v33 - force element extended
Updated Prof. Dr. O.Wallrapp - 01.03.2010 v40 - New Design

```
> Version;
```
40

```
> keyPrint := 1;     # 0 = minimum, 1 = medium, 2 = all
```
keyPrint := 1

```
> unassign(t):
```

Model Description

Note: You can use variables (check their names with those useed in the MBS Equ.) or values

Set up Name of Model

```
> MBS_Name := "Pendulum Ex-1.5.1";
```
MBS_Name := "Pendulum Ex-1.5.1"

Set up Body Data

BodyData(i, 12), i = 1..NBody
Note: type: r = rigid body, e = elastic
Each body i: iID, type(r,e), mass mi , cix,y,z; IOixx, yy, zz, xy, xz, yz , w.r.t. body fixed frame !

1 2 3 4,5,6 7,8,9 10,11,12

```
> BodyData := Matrix([
     [ b1  , r,  m1,     c1,0,0,  Ixx1,Iyy1,Izz1, 0,0,0]
   ]):   printBodyData(BodyData);
```
"Nr. of bodies: NBody = ", 1

"BodyData",

0	1	2	3	4	5	6	7	8	9	10	11	12
i	*iID*	*type*	*mi*	*cix*	*ciy*	*ciz*	*IOixx*	*IOiyy*	*IOizz*	*IOixy*	*IOixz*	*IOiyz*
1	*b1*	*r*	*m1*	*b*	0	0	*Ixx1*	*Iyy1*	*Izz1*	0	0	0

Gravitational field gI[3], w.r.t. Inertial frame

```
> gI := Vector([0, -9.81, 0]);
```
$$gI := \begin{bmatrix} 0 \\ -9.81 \\ 0 \end{bmatrix}$$

Set up Marker Data

MarkerData(k, 9), k = 1..NMarker
Each marker k: kID, bodyID, free, rk(1,2,3), theta_k(1,2,3), w.r.t. body fixed frame !

1 2 3 4, 5, 6 7, 8, 9

```
> MarkerData := Matrix([
     [ m1 , 0, 0,   0,0,0,   0,0,0],
     [ m2 , 0, 0,   a,0,0,   0,0,0],
     [ m3 , 1, 0,   0,0,0,   0,0,0],
     [ m4 , 1, 0,   1*b,0,0,   0,0,0],
     [ m5 , 1, 0,   2*b,0,0,   0,0,0]
   ]):   printMarkerData(MarkerData);
```
"Nr. of markers: NMarker = ", 5

"MarkerData",

0	1	2	3	4	5	6	7	8	9
k	*kID*	*iNr*	*free*	*rkx*	*rky*	*rkz*	*thetakx*	*thetaky*	*thetakz*
1	*m1*	0	0	0	0	0	0	0	0
2	*m2*	0	0	*a*	0	0	0	0	0
3	*m3*	1	0	0	0	0	0	0	0
4	*m4*	1	0	*b*	0	0	0	0	0
5	*m5*	1	0	2 *b*	0	0	0	0	0

Set up Joint Data

JointData (s, 13), s = 1..NJoint
Note : tree = OL joints listed first!!! CL = closed loop joints at the end. FROM-markerID < TO-markerID ALWAYS! Motions w.r.t From-frame
Types of OL-joints :
0 = bracket,
1 = hinge(D) about z, used as DOF or βz = f(t)
2 = slider(Sx) about x, used as DO or dx = f(t)
3 = hinge(Dz)/slider(Sx), <<<< not yet !!!
4 = double slider(Sx,Sy) about x and y, <<<< not yet !!!
5 = free 2D in x-y-plane, used as DOF, f(t) is not allowed!
Types aof CL-joints : 99 = constraints are given by the 6 keys: free = 0, locked = 1: stored in 6 .. 11.

Each joint s: sID, FromMarkerNr, ToMarkerNr, Tree/CL, Type Fcts(t) for dx, dy, dz, βx, βy, βz, || nsf =No. of joint DOF keyft:0=no,1=yes

1 2 3 4 5 6, 7 , 8, 9, 10, 11 | 12 13

```
> JointData := Matrix([
     [ j1 , 2, 3,   Tree, 1,   0,0,0, 0,0,0,       0,0]
   ]):  printJointData(JointData);
```
"Nr. of joints: NJoint = ", 1

```
"JointData", [[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13],
   [s, sID, kFrom, kTo, Tree/CL, type, fct_dx, dy, dz, betax,
   betay, betaz, nsf, keyft],
   [1, j1, 2, 3, Tree, 1, 0, 0, 0, 0, 0, 0, 0, 0]]
```

Set up Force Element Data

ForceData (a , 11): a = 1..NForce
Note : all element works between From -> To markers, action on marker From, in the case of axial element, joint ID =0,
Types of force elements :
0 = nothing,
1 = axial linear spring-damper between marker From and To, as fct of ls,lsdot,
2 = Cartesian spring-damper between From and To of From frame, as fct in d1,2,3; V1,2,3; b1,2,3; O1,2,3;
11 = rot. spring-damper at hinge JointNr as fct of hinge DOF bz and bzdot, for ds=0
12 = lin. spring-damper at slider JointNr as fct of slider DOF dx and dxdot
13 = lin. and rot. spring-damper at hinge/slider JointNr as fct of hinge/slider DOF dx, bz and dxdot, bzdot

Each force element a: fID, FromMarkerID, ToMarkerID, Type JointNr, Fcts(t) of DOF (max=6)
1 2 3 4 5 6, 7 , 8, 9, 10, 11

```
> ForceData := Matrix([
     [ f1 , 2, 3,   11,  1,   dT1*bzdot, 0, 0, 0,0,0]
   ]):   printForceData(ForceData);
              "Nr. of Force Elements: NForce = ", 1
"ForceData",
   [[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11],
   [f, fID, kFrom, kTo, type, JointNr, fct1, fct2, fct2, fct3,
   fct4, fct6],
   [1, f1, 2, 3, 11, 1, dT1 bzdot, 0, 0, 0, 0, 0]]
```

set parameters

```
> par := [m1 = 10.8, c1 = 1, Izz1 = 14.4, Iyy1 = 14.4, Ixx1 =
  0.036,
          a = 1, b = 1, dT1 = 10];
par := [m1 = 10.8, c1 = 1, Izz1 = 14.4, Iyy1 = 14.4, Ixx1 = 0.036, a = 1, b = 1,
   dT1 = 10]
```

Computations - Part 1: Kinematics of markers and joints

**** Compute marker matrices 1:
# rk(k)[3], k=1..NMarker, position of k
# Dk(k)[3,3], orientation of k, ek = Dk . ei
# BodyMarker[NBody,10]: each body i: 1 = nn, 2.. = marker Nrs

```
> Marker_Matrices_1(NMarker, MarkerData, keyPrint);
"k=", 1, " at body=", 0, ": rk=", [0, 0, 0], "Dk=", [1 0 0; 0 1 0; 0 0 1],
   "w.r.t. body frame"
"k=", 2, " at body=", 0, ": rk=", [a, 0, 0], "Dk=", [1 0 0; 0 1 0; 0 0 1],
   "w.r.t. body frame"
"k=", 3, " at body=", 1, ": rk=", [0, 0, 0], "Dk=", [1 0 0; 0 1 0; 0 0 1],
   "w.r.t. body frame"
"k=", 4, " at body=", 1, ": rk=", [b, 0, 0], "Dk=", [1 0 0; 0 1 0; 0 0 1],
   "w.r.t. body frame"
"k=", 5, " at body=", 1, ": rk=", [2 b, 0, 0], "Dk=", [1 0 0; 0 1 0; 0 0 1],
   "w.r.t. body frame"
"Each body i: (nn, marker Nrs): BodyMarker[i,k]=",
   [3 3 4 5 0 0 0 0 0 0]
```

**** Compute joint kinematic matrices 1
Dimensions: OL = joints open look, CL = joint to close a open loop
NJoint_OL = number joint in OL; NJoint_CL = number joint in CL
nz = number Cartesian coord. = 6 NBody; nf_OL = number DOF in OL;
nc_OL = number constraints in OL; nc_CL = number constraints in CL
nf_MBS = number of DOF of MBS
Info_yOL [nf_OL, 4]: 1=sNr, 2=yID, 3=vel-fct/0, 4=DOF-name
Info_yOL_bar[nc_OL, 4]: 1=sNr, 2=yID, 3=vel-fct/0, 4=DOF-name
yIOL[nf_OL], yIIOL[nf_OL] = joint state variables in OL on position and velocity
yIIOL_bar[nc_OL] = joint kin. driven fct in OL on velocity
dyIIOL_bar[nc_OL] = joint kin. driven fct in OL on acceleration = d(yIIOL_bar)/dt
YOL[nf_OL, nf_OL] = matrix of kin. DE in OL
Matrices Bs(s), ds(s), Vs(s), Omega_s(s), phi_s(s), phi_sbar(s), s = 1..NJoint_OL (only for OL joints)

```
> Joint_Matrices_1(NJoint, JointData, keyPrint);
j1, Tree, ":s=", 1, "type=", 1, "bw markers", 2, 3, ": ds=", [0, 0, 0], "Vs=",
```

$$\begin{bmatrix}0\\0\\0\end{bmatrix}, \text{"Bs="}, \begin{bmatrix}\cos(yI1) & \sin(yI1) & 0\\ -\sin(yI1) & \cos(yI1) & 0\\ 0 & 0 & 1\end{bmatrix}, \text{"Omega_s="}, \begin{bmatrix}0\\0\\yIII\end{bmatrix}, \text{"phi_s="},$$

$$\begin{bmatrix}0\\0\\0\\0\\0\\1\end{bmatrix}, \text{"phi_sbar="}, \begin{bmatrix}1&0&0&0&0\\0&1&0&0&0\\0&0&1&0&0\\0&0&0&1&0\\0&0&0&0&1\\0&0&0&0&0\end{bmatrix}$$

```
"=== MBS Body and Joint Summary ====="
"The MBS has    ", 1, " body"
"Dim. nz = ", 6, "Card.coord. zII", "  and  nf_MBS = ", 1, "DOF"
"------------ OL MBS -----------------"
"NJoint_OL=", 1, "; nf_OL=", 1, "; nc_OL=", 5
```

$$\text{"Info_yOL="}, \begin{bmatrix}1&6&0&bz1\end{bmatrix}, \text{" yIOL[nf_OL]="}, \begin{bmatrix}yI1\end{bmatrix}, \text{" yIIOL="}, \begin{bmatrix}yIII\end{bmatrix}$$

$$\text{"Info_yOL_bar="}, \begin{bmatrix}1&1&0&dx1\\1&2&0&dy1\\1&3&0&dz1\\1&4&0&bx1\\1&5&0&by1\end{bmatrix}, \text{"yIIOL_bar[nc_OL]="}, \begin{bmatrix}0\\0\\0\\0\\0\end{bmatrix},$$

$$\text{" dyIIOL_bar="}, \begin{bmatrix}0\\0\\0\\0\\0\end{bmatrix}$$

```
"The kin. DGL of OL is dyI_OL = YOL . yII_OL , where YOL = I"
"------------ CL MBS -----------------"
"NJoint_CL=", 0
"nc_CL=", 0
"------------ show JointData again -----------------"
"Nr. of joints: NJoint = ", 1
```

$$\text{"JointData"}, \left[\left[0,\ 1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8,\ 9,\ 10,\ 11,\ 12,\ 13\right],\right.$$

$$\left[s,\ sID,\ kFrom,\ kTo,\ \frac{Tree}{CL},\ type,\ fct\_dx,\ dy,\ dz,\ betax,\right.$$

$$\left.betay,\ betaz,\ nsf,\ keyft\right],$$

$$\left.\left[1,\ j1,\ 2,\ 3,\ Tree,\ 1,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 1,\ 0\right]\right]$$

```
# Compute abs. body kinematics of the OL-MBS in terms of yIOL, yIIOL or joint fct.
# A(i)     = orientation matrix of ei = A . eI,  i = 0 .. NBody
# alpha(i) = Cardan angles of A
# omega(i) = angular velocity of i w.r.t. I, given body ref frame
# rho(i)   = position of Oi given in body ref frame
# rhoI(i)  = position of Oi given in inertial frame I
# v(i)     = lin. velocity of Oi w.r.t OI, given body ref frame

> Body_Kin_Matrices_1(NJoint_OL, JointData, keyPrint);
```


$$\text{"body i="}, 1, \text{": A="}, \begin{bmatrix}\cos(yI1) & \sin(yI1) & 0\\ -\sin(yI1) & \cos(yI1) & 0\\ 0 & 0 & 1\end{bmatrix}, \text{"alpha="},$$

$$\begin{bmatrix}0\\0\\\arctan(\sin(yI1), \cos(yI1))\end{bmatrix}, \text{"rho="}, \begin{bmatrix}\cos(yI1)\ a\\ -\sin(yI1)\ a\\ 0\end{bmatrix}, \text{"rhoI="}, \begin{bmatrix}a\\0\\0\end{bmatrix},$$

$$\text{"v="}, \begin{bmatrix}0\\0\\0\end{bmatrix}, \text{"omega="}, \begin{bmatrix}0\\0\\yIII\end{bmatrix}$$

```
# **** Compute marker matrices 2:
# Tkit(k)[3,6], k=1..NMarker, Jacobean matrix of velocity - translational, Eq.(2.31)
# Tkir(k)[3,6], k=1..NMarker, Jacobean matrix of velocity - rotational
> Marker_Matrices_2(NMarker, MarkerData, keyPrint);
# Compute abs. marker kinematics in terms of yIOL, yIIOL or joint fct.
# Ak(k)     = orientation matrix of ek = Ak . eI,  k = 1 .. NMarker
# alphak(k) = Cardan angles of Ak
# omegak(k) = angular velocity of  k w.r.t. I, given body ref frame
# rhok(k)   = position of Ok given in body ref frame
# rhokI(k)  = position of Ok given in inertial frame I
# vk(k)     = lin. velocity of Ok w.r.t OI, given body ref frame
# vkI(k)    = lin. velocity of Ok w.r.t OI, given inertial frame

> Marker_Matrices_3(NMarker, MarkerData, keyPrint);
```

$$\text{"k,i="}, 1, 0, \text{":Ak="}, \begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}, \text{"omegak="}, \begin{bmatrix}0\\0\\0\end{bmatrix}, \text{"rhokI="}, \begin{bmatrix}0\\0\\0\end{bmatrix}, \text{"vkI="},$$

$$\begin{bmatrix}0\\0\\0\end{bmatrix}$$

$$\text{"k,i="}, 2, 0, \text{":Ak="}, \begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}, \text{"omegak="}, \begin{bmatrix}0\\0\\0\end{bmatrix}, \text{"rhokI="}, \begin{bmatrix}a\\0\\0\end{bmatrix}, \text{"vkI="},$$

$$\begin{bmatrix}0\\0\\0\end{bmatrix}$$

$$\text{"k,i="}, 3, 1, \text{":Ak="}, \begin{bmatrix}\cos(yI1) & \sin(yI1) & 0\\ -\sin(yI1) & \cos(yI1) & 0\\ 0 & 0 & 1\end{bmatrix}, \text{"omegak="}, \begin{bmatrix}0\\0\\yIII\end{bmatrix},$$

$$\text{"rhokI="}, \begin{bmatrix}a\\0\\0\end{bmatrix}, \text{"vkI="}, \begin{bmatrix}0\\0\\0\end{bmatrix}$$

$$\text{"k,i="}, 4, 1, \text{":Ak="}, \begin{bmatrix}\cos(yI1) & \sin(yI1) & 0\\ -\sin(yI1) & \cos(yI1) & 0\\ 0 & 0 & 1\end{bmatrix}, \text{"omegak="}, \begin{bmatrix}0\\0\\yIII\end{bmatrix},$$

$$\text{"rhokI="}, \begin{bmatrix} \cos(yI1)\ b + a \\ \sin(yI1)\ b \\ 0 \end{bmatrix}, \text{"vkI="}, \begin{bmatrix} -\sin(yI1)\ yII1\ b \\ \cos(yI1)\ yII1\ b \\ 0 \end{bmatrix}$$

$$\text{"k,i="}, 5, 1, \text{":Ak="}, \begin{bmatrix} \cos(yI1) & \sin(yI1) & 0 \\ -\sin(yI1) & \cos(yI1) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \text{"omegak="}, \begin{bmatrix} 0 \\ 0 \\ yII1 \end{bmatrix},$$

$$\text{"rhokI="}, \begin{bmatrix} 2\cos(yI1)\ b + a \\ 2\sin(yI1)\ b \\ 0 \end{bmatrix}, \text{"vkI="}, \begin{bmatrix} -2\sin(yI1)\ yII1\ b \\ 2\cos(yI1)\ yII1\ b \\ 0 \end{bmatrix}$$

Computations - Part 2: Find explicit constraint Eqs. of OL and closed loop constraint Eqs. of CL

Compute explicit contraint Eqs. of OL-MBS, see Sect. 3.7.2 !
fyOL[nz] = vector of zI = fyOL(yOLI, t)
Tzx [nz,nz] = matrix of zII = Tzx . xII_OL, where xII_OL = (Vs,Omega_s) of OL with s = i = 1 .. NBody
Then: from Tzx find the Jacobian of explicit OL body constraint Eq. zII = JyOL . yOLII + JyOL_bar . yOLII_bar
JyOL [nz,nf_OL] = Jacobean of free motion
JyOL_bar[nz,nc_OL] = Jacobean of locked motion

```
> Body_Kin_Matrices_OL( NJoint_OL, NBody, nf_OL, keyPrint):
">>>> Matrices of explicite kin. Eq. of OL: zI = fyOL(yI,t);  zII
   = JyOL . yII_OL + JyOL_bar . yII_OL_bar,
                        with nf_OL=", 1, " and nc_OL=", 5
```

$$\text{"fyOL[nz]="}, \begin{bmatrix} \cos(yI1)\ a \\ -\sin(yI1)\ a \\ 0 \\ 0 \\ 0 \\ \arctan(\sin(yI1), \cos(yI1)) \end{bmatrix}, \text{" JyOL[nz,nf_OL]="}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix},$$

$$\text{" JyOL_bar[nz,nc_OL]="}, \begin{bmatrix} \cos(yI1) & \sin(yI1) & 0 & 0 & 0 \\ -\sin(yI1) & \cos(yI1) & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & \cos(yI1) & \sin(yI1) \\ 0 & 0 & 0 & -\sin(yI1) & \cos(yI1) \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Compute implicit contraint Eqs. GOL . zII = kappaCL of OL-MBS, see Eq. (2.41) - for tests
GOL[nc_OL, nz] = implicit contraint matrix

```
> Body_Kin_Matrices_GOL(NJoint, NJoint_OL, NJoint_CL, NBody,
  nc_CL, nc_OL, keyPrint);
```


$$\text{"GOL[nc_OL,nz]="}, \begin{bmatrix} \cos(yI1) & -\sin(yI1) & 0 & 0 & 0 & 0 \\ \sin(yI1) & \cos(yI1) & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \cos(yI1) & -\sin(yI1) & 0 \\ 0 & 0 & 0 & \sin(yI1) & \cos(yI1) & 0 \end{bmatrix}$$

```
> # Tests:
  Transpose(JyOL) . Transpose(GOL); simplify(Transpose
(JyOL_bar) . Transpose(GOL));
```

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

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Computations - Part 3: Set up the dynamical Eqs M . zII + Q = h, where h = hg + hf + G_T . λ

Compute mass matrix M of all bodies i = 1 .. nBody, see Eq.(3.15)
M[nz,nz] = (Mitt Mitr)
(Mirt Mirr)
Q[nz] = (Qit)
(Qir)

```
> Mass_Matrices(NBody, BodyData, keyPrint);
```

$$\text{"M[nz,nz]="}, \begin{bmatrix} m1 & 0 & 0 & 0 & 0 & 0 \\ 0 & m1 & 0 & 0 & 0 & m1\ b \\ 0 & 0 & m1 & 0 & -m1\ b & 0 \\ 0 & 0 & 0 & Ixx1 & 0 & 0 \\ 0 & 0 & -m1\ b & 0 & Iyy1 & 0 \\ 0 & m1\ b & 0 & 0 & 0 & Izz1 \end{bmatrix}, \text{"for par+t=0:M="},$$

$$\begin{bmatrix} 10.8 & 0. & 0. & 0. & 0. & 0. \\ 0. & 10.8 & 0. & 0. & 0. & 10.8 \\ 0. & 0. & 10.8 & 0. & -10.8 & 0. \\ 0. & 0. & 0. & 0.036 & 0. & 0. \\ 0. & 0. & -10.8 & 0. & 14.4 & 0. \\ 0. & 10.8 & 0. & 0. & 0. & 14.4 \end{bmatrix}, \text{" Q[nz]="}, \begin{bmatrix} -m1\ yII1^2\ b \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix},$$

$$\text{"for par+t=0:Q="}, \begin{bmatrix} -10.8\ yII1^2 \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \end{bmatrix}$$

Compute gen. forces due to gravity
#
hg[nz] = gen. forces of all bodies

```
> Gravity_Forces(NBody, gI, BodyData, keyPrint);
```

"hg[nz]=", $\begin{bmatrix} -9.81\ m1\ \sin(yI1) \\ -9.81\ m1\ \cos(yI1) \\ 0. \\ 0. \\ 0. \\ -9.81\ m1\ b\ \cos(yI1) \end{bmatrix}$, "for par+t=0:hg=",

$\begin{bmatrix} -105.948\ \sin(yI1) \\ -105.948\ \cos(yI1) \\ 0. \\ 0. \\ 0. \\ -105.948\ \cos(yI1) \end{bmatrix}$

```
> ForceElement_Forces(NForce, ForceData, keyPrint);
```

"NfctForce=", 1, "fctForce=", [Lsa1 = dT1 yII1]

"hf[nz]=", $\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -Lsa1 \end{bmatrix}$, "for par+t=0:hf=", $\begin{bmatrix} 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ -10.\ yII1 \end{bmatrix}$

```
> ha := evalf(subs(par, t=0, hf + hg));
```

$$ha := \begin{bmatrix} -105.948\ \sin(yI1) \\ -105.948\ \cos(yI1) \\ 0. \\ 0. \\ 0. \\ -10.\ yII1 - 105.948\ \cos(yI1) \end{bmatrix}$$

```
> # Compute zeta_OL of the acceleration in Eq.(3.39): zIIOL_dot
  = Jy_OL . yIIOL_dot + zeta_OL
  #
  # zeta_OL[nz] = acceleration due to time derivations and
  driven joints and

  Body_Kin_Matrices_OL_dot(nf_OL, nc_OL, keyPrint);
```

Find the Solution for nf_MBS >= 0 but nc_CL = 0 : ODE - System of dimension 2*nf_MBS

```
> nf_MBS;
```

1

```
> par;
```

[m1 = 10.8, c1 = 1, Izz1 = 14.4, Iyy1 = 14.4, Ixx1 = 0.036, a = 1, b = 1, dT1 = 10]

```
>
  tEnd := 20;     # Simulation time
```

tEnd := 20

1 - Solution for nf_MBS = 0:

2 - Solutions for nf_MBS > 0.

 # solve the ODE-system: ODEk = dyI/dt = yII; ODEd = My . dyII/dt - (hay - Qy) = 0

```
> if (nf_MBS > 0 and nc_CL = 0) then    # solve the ODE-
  system

    yIOLt   := seq(yIOL[i] = yIOL[i](t), i=1..nf_MBS);
    yIIOLt  := seq(yIIOL[i] = yIIOL[i](t), i=1..nf_MBS);
    dyIIOLt := Vector([seq(diff(yIIOL[i](t),t), i=1..nf_MBS)
  ]);
  fi;
```

$$yIOLt := yI1 = yI1(t)$$

$$yIIOLt := yII1 = yII1(t)$$

$$dyIIOLt := \left[\frac{d}{dt}\ yII1(t) \right]$$

```
> if (nf_MBS > 0 and nc_CL = 0) then
  # setup the minimal set of dyn Eq. ODEd:  My . zII_dot +
  Qy = hay  and  ODEk:  yI_dot = Y . yII;

    M := subs(par, M):  Q := subs(par, Q):
    zeta_OL := subs(par, zeta_OL): JyOL := subs(par,JyOL):
  JyOL_bar := subs(par,JyOL_bar):
    My  := simplify(subs(par, yIOLt,          Transpose
  (JyOL) . M . JyOL )):
    Qy  := simplify(subs(par, yIOLt, yIIOLt,( Transpose
  (JyOL) . (Q + M . zeta_OL) ) )):
    hay := simplify(subs(par, yIOLt, yIIOLt,  Transpose
  (JyOL) . ha )):

    h1  := simplify(hay - Qy):
    h2  := My . dyIIOLt - h1:

    zerovec := ZeroVector(nf_MBS):
    ODEd := seq(h2[i] = zerovec[i], i=1..nf_MBS):
    ODEk := seq(diff(yIOL[i](t),t) = yIIOL[i](t), i=1..
  nf_MBS):
  fi:
```

```
> ODEd;
  accel0 := MatrixInverse(My).h1:       #  Intial
  acceleration
  subs(initcond, subs(t=0, accel0));
```

$$14.40000000 \left(\frac{d}{dt}\ yII1(t) \right) + 10.\ yII1(t) + 105.9480000\ \cos(yI1(t)) = 0$$

$$\left[-7.357500000 \right]$$

```
> if (nf_MBS > 0 and nc_CL = 0) then

  # use dsolve !!! results are stored in res2[t, yI1(t),  ..
  .  yInf(t),  yII1(t),  ...  yIInf(t) ]

    initcond := seq(yIOL[i](0) = 0, i=1..nf_MBS), seq(yIIOL
  [i](0) = 0, i=1..nf_MBS) ; # all are zero

    vars := [ seq(yIOL[i](t), i=1..nf_MBS), seq(yIIOL[i](t),
  i=1..nf_MBS) ];
    res2 := dsolve({ODEk, ODEd, initcond}, vars, numeric);
  # finds the solution numericaly
```

```
  res2(1);     # Test of result for t
  res2(tEnd);  # Test of result for t

  odeplot(res2,[[t,yI1(t)], [t,yII1(t)]], 0..tEnd,
numpoints= 300,gridlines=true);
  # odeplot(res2,[[t,yI2(t)], [t,yII2(t)]], 0..tEnd,
numpoints= 300,gridlines=true);
  # odeplot(res2,[[t,yI3(t)], [t,yII3(t)]], 0..tEnd,
numpoints= 300,gridlines=true);

# Take the result and store it in a variable ;  check the
outputs of res2!!!
  y1  := t -> rhs( op(2,res2(t)) ) ; y1(1) ;
  y1d := t -> rhs( op(3,res2(t)) ) ; y1d(1);

  # y2  := t -> rhs( op(3,res2(t)) ) ; y2(1);
  # y2d := t -> rhs( op(5,res2(t)) ) ; y2d(1);

  plot([y1], 0..tEnd, title = "yI1", gridlines=true);
# it works

fi;
```

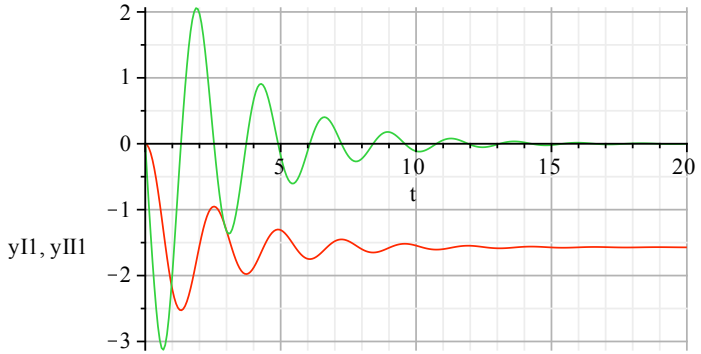
$$initcond := yI1(0) = 0,\ yII1(0) = 0$$

$$vars := [yI1(t),\ yII1(t)]$$

$$res2 := \mathbf{proc}(x\_rkf45)\ \ldots\ \mathbf{end\ proc}$$

$$[t = 1.,\ yI1(t) = -2.21226179915895971,\ yII1(t) = -1.97512815434640765]$$

$$[t = 20.,\ yI1(t) = -1.57213148333943420,\ yII1(t) = -0.00097643056595371100 7]$$

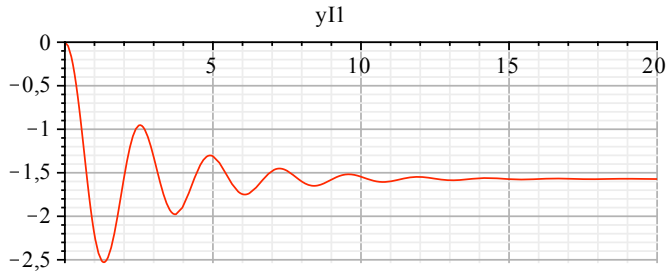


$$y1 := t \rightarrow rhs(op(2,\ res2(t)))$$

$$-2.21226179915895971$$

$$y1d := t \rightarrow rhs(op(3,\ res2(t)))$$

$$-1.97512815434640765$$

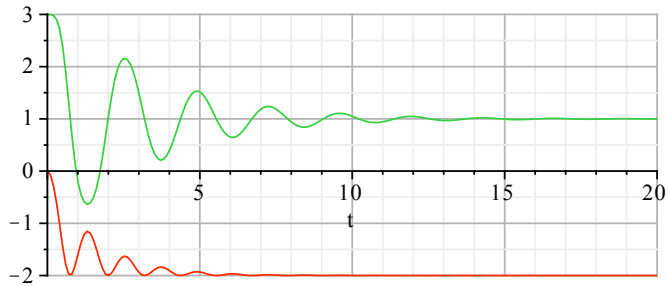



```
> if (nf_MBS > 0 and nc_CL = 0) then
# First: set yIOL1 = yIOL1(t), - now we can plot any
marker coordinates
# Now, use unapply to get values of rk and give values for
t, y1, y2 ... ,
  for k from 1 to NMarker do rkt(k) := evalf(subs(par,
yIOLt, rhokI(k))): od;
  rx := rkt(NMarker)[2]: ry := rkt(NMarker)[1]:   # plot
of marker motions
  odeplot( res2, [[t,rx],[t,ry]], t=0..tEnd, numpoints=
200, title="rk(NMarker)",gridlines=true );

fi;
```

$$rx := 2.\ \sin(yI1(t))$$

$$ry := 2.\ \cos(yI1(t)) + 1.$$




```
> if (nf_MBS > 0 and nc_CL = 0) then
# animation of the MBS
 No_picts:= 40:
 deltat := tEnd/No_picts; tt := 0;
 data:='data':  data:=[]:
 for i from 0 to No_picts+1 do
  for k from 1 to NMarker do
   rkx(k) := evalf(unapply(rkt(k)[1], t, yI1, yI2, yI3)(
tt, y1, y2, y3)):
   rky(k) := evalf(unapply(rkt(k)[2], t, yI1, yI2, yI3)(
tt, y1, y2, y3)):
  od:
  data:=[op(data),  plotMBS(MBS_Name, NBody, NMarker, -3,
3, tt)  ]:
  tt := tt + deltat:
 od :
 display(data,insequence=true, scaling=constrained, axes=
none);
```

```
fi;
```

$$No\_picts := 40$$

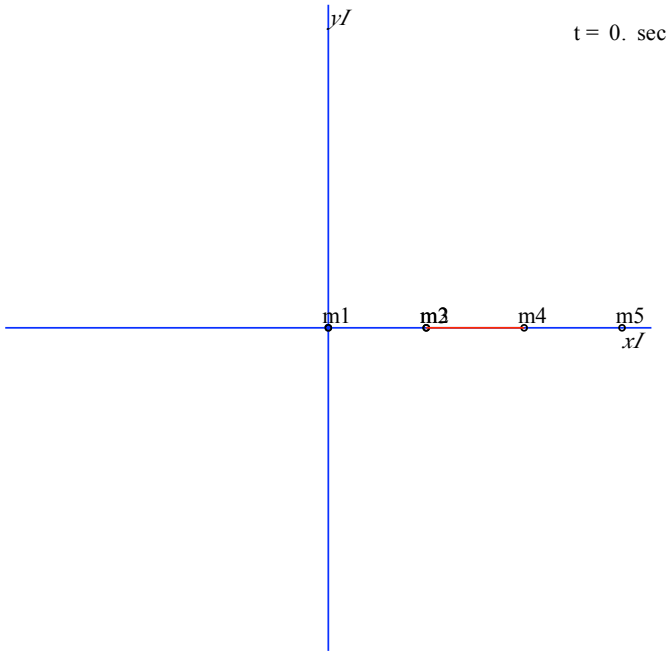
$$deltat := \frac{1}{2}$$

$$tt := 0$$

$$data := data$$

$$data := []$$

Pendulum Ex-1.5.1



End of computations / O.Wallrapp