МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
имени академика С.П.КОРОЛЁВА»

Институт ракетно-космической техники

Кафедра теоретической механики

Курсовая работа
по дисциплине: « Динамика твердого тела и систем тел»

Выполнил: Студент группы 1135-010403D
Патель Ишан Киранкумар
Проверил: Доцент, Кафедра теоретической механики
Юдинцев Вадим Вячеславович

САМАРА 2019

Abstract

Course work: 19 pages, 9 figures, 1 table, 31 equations, 1 reference.

MULTI-BODY SYSTEMS, SPHERICAL JOINT, CYLINDRICAL JOINT, WITTENBURG’S METHOD, LINEAR VELOCITY, ANGULAR VELOCITY, ANGULAR ACCELERATION, EULER ANGLES, MATLAB, MATHEMATICAL ALGOIRTHM, GENERALIZED COORDINATES, TRANSFORMATION MATRIX, INCIDENCE MATRIX, ORGRAPH, CONSERVATION OF ENERGY, KINEMATICS, BLENDER, 3D ANIMATION.

The goal of the current work is to find numerical solution for the motion of a system consisting of 5 rigid bodies linked to each other sequentially by spherical and cylindrical hinges.

External forces acting on the system are gravitational forces directed towards the ground. There are no torques acting about any hinges in the system.

The system is modelled using Wittenburg’s method for multi-body system. System parameters for modelling are given in the form of mass and length of the rods along with 2 sets of initial conditions defining the orientation of the hinges.

Numerical results are validated by calculating kinetic energy of the system at each step in time within the timespan of integration.

Mathematical algorithm is written in the form of MATLAB code to generate the results and export the data to an external csv file to generate animation of the motion using 3D animation software – Blender.

Table of contents

I. The problem

Using the Wittenburg’s method, build a mathematical model describing the motion of system of five bodies linked to each other by cylindrical and spherical hinges as shown in figure 1. Develop a program to model the motion of this system.

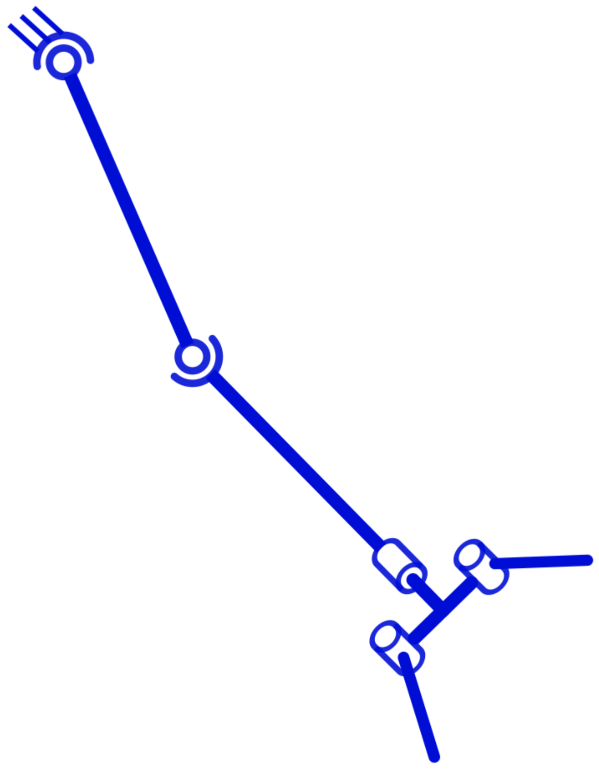


Figure 1 – Multi-body system

Construct a 3D model and animation illustrating the motion of the system on the basis of the results of integration of equations of motion.

II. Description of the system

The system consists of five ideally solid bodies linked by cylindrical and spherical hinges. One of the bodies is fixed to a body in inertial frame, here Earth. The scheme is shown in figure 2.

The system acts under the action of gravitational force of the Earth.

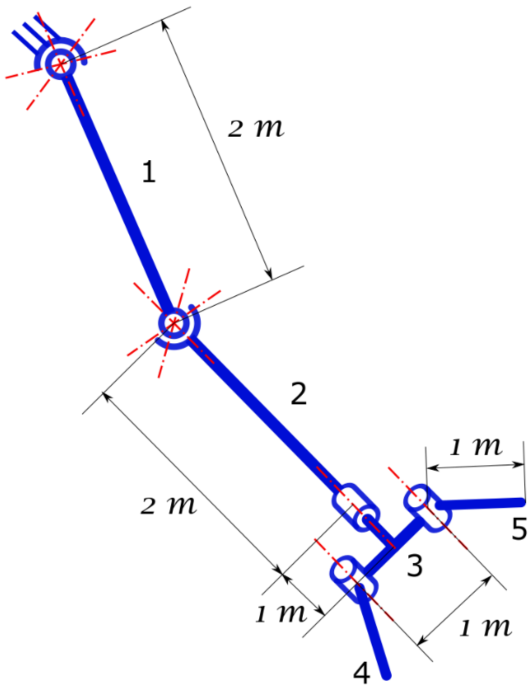


Figure 2 – Description of the system

The solid bodies are in the form of rigid rods of circular cross-section. Hinges are considered weightless.

To construct the mathematical model of this system, Wittenburg's direct method is employed.

A. Parameters of the system

Total mass of the system is 11 kg. With mass distribution as following:

$$m_1 = m_2 = 3\ kg\ ,\ m_3 = 2\ kg\ ,\ m_4 = m_5 = 1.5\ kg$$

Length of rods are as following:

$$l_1 = l_2 = 2\ m\ ,\ l_3(T - shaped) = 1,1\ m\ ,\ l_4 = l_5 = 1\ m.$$

Centre of mass of every individual rod lies at the midpoint of the axial length of the rod.

B. Kinematics of the system

The system consists of 5 rigid bodies linked by hinges with 3 degrees of freedom (spherical hinge) and 1 degree of freedom (cylindrical hinge).

1. Generalized coordinates

For the convenience of computation, angular deviation of individual hinges in local coordinate system are taken as generalized coordinates to determine the state of the system. Additionally, rate of angular deviation $\dot{q}_{\alpha i}$ is also used to determine the state vector q, which comprises these elements: Euler angles – q_{11}, q_{12}, q_{13} for 1st hinge; Euler angles – q_{21}, q_{22}, q_{23} for 2nd hinge; q_{31} for 3rd hinge; q_{41} for 4th hinge; q_{51} for 5th hinge.

The 1st body is fixed to a body *0* whose state is known in an inertial reference frame $x_0y_0z_0$. Successive coordinate systems of each bodies with representation of generalized coordinates q and generalized velocities p as shown in figure 3.

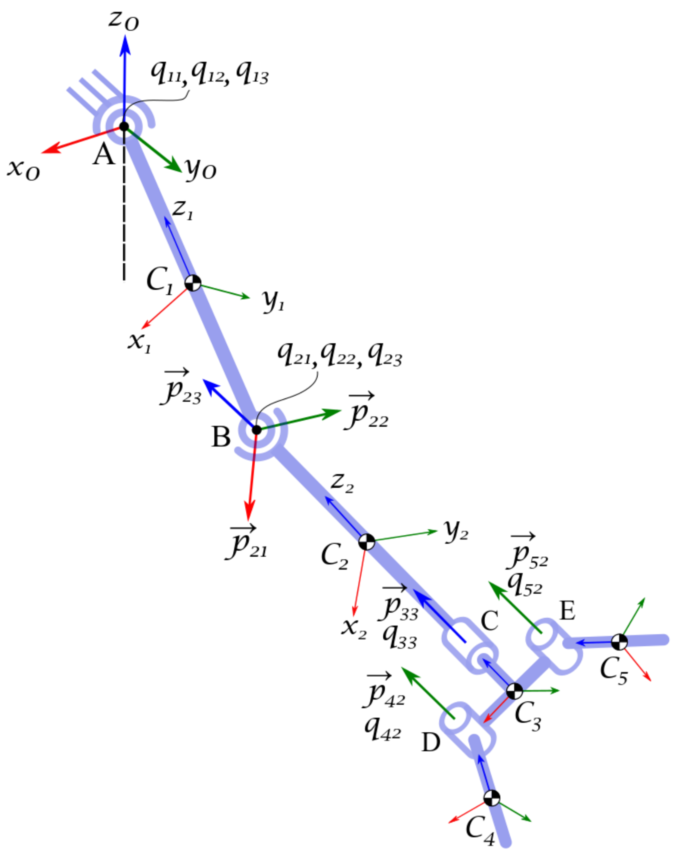


Figure 3 – Mechanical scheme of the system

2. Transformation matrices

Matrix of transformation for transforming the coordinate system from the basis of one body to the basis of another body is expressed in the form of generalized coordinates. For the current problem, the transformation of coordinate system is done from basis of bodies with higher index i to the basis of bodies with lower index $i-1$. For example, transformation matrix for transformation from body *2* to body *1* is written as A^{12}. Consequently, state vector can be transformed from basis *2* to basis *1* as following:

$$r^{(1)} = A^{12} r^{(2)} \tag{1}$$

Transformation matrix for cylindrical hinge with axis of rotation in z-direction is written as,

$$A^{23} = \begin{bmatrix} \cos q_{31} & -\sin q_{31} & 0 \\ \sin q_{31} & \cos q_{31} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

Similarly, transformation matrix for spherical hinges can be written in the form of Euler's angle as following:

$$A^{12} = A_{\psi} A_{\theta} A_{\varphi} \tag{3}$$

Where, $\psi = q_{21}$ $\theta = q_{22}$ and $\varphi = q_{23}$.

$$A_{\psi} = \begin{bmatrix} \cos q_{21} & -\sin q_{21} & 0 \\ \sin q_{21} & \cos q_{21} & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

$$A_{\theta} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos q_{22} & -\sin q_{22} \\ 0 & \sin q_{22} & \cos q_{22} \end{bmatrix}, \text{ and}$$

$$A_{\varphi} = \begin{bmatrix} \cos q_{23} & -\sin q_{23} & 0 \\ \sin q_{23} & \cos q_{23} & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

3. Kinematic expressions

Relative angular velocity Ω_{α} of a body $i^{-}(\alpha)$ with respect to a body $i^{+}(\alpha)$ can be expressed in the form of generalized coordinates as mentioned below[1]:

$$\Omega_\alpha = \sum_{i=1}^{n_\alpha} p_{\alpha i}\dot{\varphi}_{\alpha i}\,, \qquad \alpha = 1, \dots, 5 \tag{4}$$

Where, n_α – Total number of degrees of freedom of the hinge α

$p_{\alpha i}$ – Unit vector directed along the rotation vector

For spherical hinge, $n_\alpha = 3$ thus exists three unit vector $p_{\alpha 1}, p_{\alpha 2}$ and $p_{\alpha 3}$ around which the rotation of adjacent bodies $i^-(\alpha)$ and $i^+(\alpha)$ takes place. Similarly, for cylindrical hinge, $n_\alpha = 1$ with only one unit vector $p_{\alpha 1}$ around which the rotation of adjacent bodies takes place. In any case, coordinates of vector $p_{\alpha i}^{i^-(\alpha)}$ are the function of generalized coordinates $\varphi_{\alpha i}$.

Absolute and relative angular velocity are related in the form,

$$\Omega_\alpha = \omega_{i^-(\alpha)} - \omega_{i^+(\alpha)}\,, \qquad \alpha = 1, \dots, 5 \tag{5}$$

Which can be written as,

$$\Omega_\alpha = -\sum_{i=0}^{5} S_{i\alpha}\omega_i = -S_{0\alpha}\omega_0 - \sum_{i=1}^{5} S_{i\alpha}\omega_i\ , \alpha = 1, \dots, n \tag{6}$$

Above equation (6) can be expressed in the form of matrix as following:

$$\Omega = -\omega_0 {S_0}^T - S^T\omega \tag{7}$$

Where, $\Omega = [\Omega_1 \dots \Omega_5]^T$, $\omega = [\omega_1 \dots \omega_5]^T$ – Column matrices of relative and absolute angular velocities, respectively.

Now, rearranging equation (7) to have only the absolute angular velocity on the left hand side,

$$\omega = -T^T\Omega + \omega_0 1_5 \tag{8}$$

Where, $T = S^{-1}$, $1_5 = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}_5$.

Thus, angular velocity for each body can be written as,

$$\omega_i = -\sum_{a=1}^{5} T_{ai}\Omega_a + \omega_0\,, \qquad a = 1, \dots, 5 \tag{9}$$

III. Algorithm to generate equations of motion

The equation of motion for the multi-body system can be expressed in the following dynamic form,

$$A\ddot{\varphi} = B \tag{10}$$

Where, matrices A and B are expressed in following form [1]:

$$A = (pT) \cdot K \cdot (pT)^T, \tag{11}$$

$$B = -(pT)(K(T^T f - \dot{\omega}_0 1_n) + M' + M) - pY \tag{12}$$

Matrix M' is expressed as,

$$M_i' = -\omega_i \times K_i \omega_i - M \begin{bmatrix} \sum_{j;s_i<s_j} d_{ij} \times \left(\omega_j \times \left(\omega_j \times b_{j0}\right)\right) + \\ +b_{i0} \times \left(\sum_{j;s_j<s_i} \omega_j \times \left(\omega_j \times d_{ji}\right) - \ddot{r}_0 \right) \end{bmatrix} - \tag{13}$$

$$- \sum_{j;s_i<s_j} d_{ij} \times F_j \ , i = 1, \dots, n$$

$$f = \phi + \omega^* , \ \omega^* = \omega_{i^-\alpha} \times \Omega_\alpha \tag{14}$$

ϕ for individual hinge is written as,

$$\phi_1 = \begin{bmatrix} \dot{q}_{12} \sin q_{13} \left(\dot{q}_{11} \cos q_{12} - \dot{q}_{13}\right) + \dot{q}_{11}\dot{q}_{13}(\sin q_{12} \cos q_{13}) \\ \dot{q}_{12} \cos q_{13} \left(\dot{q}_{11} \cos q_{12} - \dot{q}_{13}\right) - \dot{q}_{11}\dot{q}_{13}(\sin q_{12} \sin q_{13}) \\ -\dot{q}_{11}\dot{q}_{12} \sin q_{12} \end{bmatrix} \tag{15}$$

$$\phi_2 = \begin{bmatrix} \dot{q}_{22} \sin q_{23} \left(\dot{q}_{21} \cos q_{22} - \dot{q}_{23}\right) + \dot{q}_{21}\dot{q}_{23}(\sin q_{22} \cos q_{23}) \\ \dot{q}_{22} \cos q_{23} \left(\dot{q}_{21} \cos q_{22} - \dot{q}_{23}\right) - \dot{q}_{21}\dot{q}_{23}(\sin q_{22} \sin q_{23}) \\ -\dot{q}_{21}\dot{q}_{22} \sin q_{22} \end{bmatrix} \tag{16}$$

$$\phi_3 = \phi_4 = \phi_5 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{17}$$

To develop the model further, following steps are to be followed.

First step in the development of mathematical model of the current system is to construct an orientationally numbered graph in accordance with the appearance of bodies and hinges in correct order, which is as shown in figure 4.

Second step is to construct the matrix of incidence S:

$$S_0 = [1 \quad 0 \quad 0 \quad 0 \quad 0]$$

$$S = \begin{bmatrix} -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 1 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 \end{bmatrix} \tag{18}$$

Consequently, matrix T can be constructed as,

$$T = \begin{bmatrix} -1 & -1 & -1 & -1 & -1 \\ 0 & -1 & -1 & -1 & -1 \\ 0 & 0 & -1 & -1 & -1 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 \end{bmatrix} \tag{19}$$

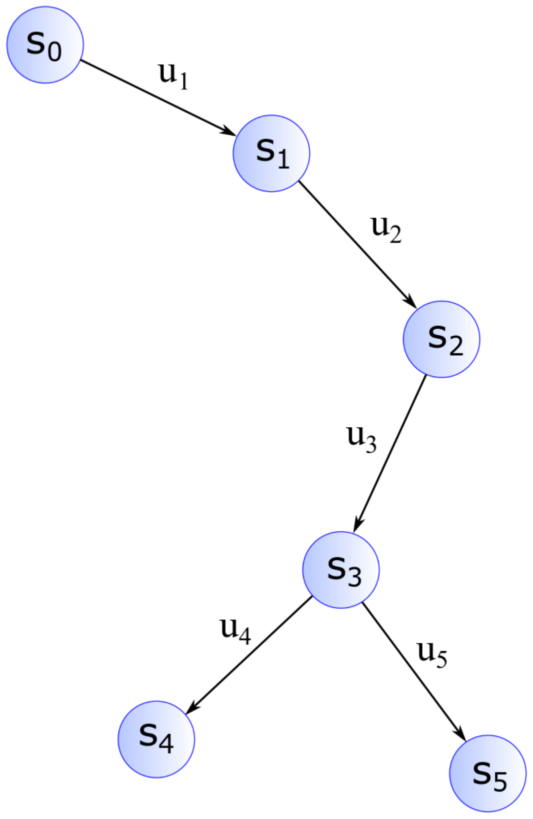


Figure 4 – Orgraph

Additionally, functions $i^+(\alpha)$ and $i^-(\alpha)$ can be written as following:

Table 1 Body index functions $i^+(\alpha)$ and $i^-(\alpha)$

Hinge, α	*1*	*2*	*3*	*4*	*5*
$i^+(\alpha)$	0	1	2	3	3
$i^-(\alpha)$	1	2	3	4	5

Fourth step is the determination of hinge vectors $c_{i\alpha}^{(i)}$ for each body. The hinge vector is directed from centre of mass of the body *i* to the hinge α as shown in figure 5. If the hinge α is not incident on the body *i*, then $c_{i\alpha}^{(i)} = 0$.

$$c_{11}^{(1)} = \begin{bmatrix} 0 & 0 & {}^{l_1}/_2 \end{bmatrix}^T, \qquad c_{12}^{(1)} = \begin{bmatrix} 0 & 0 & -{}^{l_1}/_2 \end{bmatrix}^T$$

$$c_{22}^{(2)} = \begin{bmatrix} 0 & 0 & {}^{l_2}/_2 \end{bmatrix}^T, \qquad c_{23}^{(2)} = \begin{bmatrix} 0 & 0 & -{}^{l_2}/_2 \end{bmatrix}^T$$

$$c_{33}^{(3)} = [0 \quad 0 \quad 1]^T, \qquad c_{34}^{(3)} = [1 \quad 0 \quad 0]^T$$

$$c_{35}^{(3)} = [-1 \quad 0 \quad 0]^T, \qquad c_{44}^{(4)} = \begin{bmatrix} 0 & 0 & -{}^{l_4}/_2 \end{bmatrix}^T$$

$$c_{55}^{(5)} = \begin{bmatrix} 0 & 0 & {}^{l_5}/_2 \end{bmatrix}^T$$

Fifth step is the calculation of vectors d_{ij} in the basis of body *i* as following:

$$d_{ij} = \sum_{a=1}^{5} T_{ai} S_{ja} c_{ja}\ , \qquad i,j = 1, \dots, 5 \qquad (20)$$

The product $T_{ai}S_{ja}$ is non-zero when,

- The arm u_a lies in the path linking the bodies s_0 and s_i ($T_{ai} \neq 0$), and
- Which is incident to s_j ($S_{ja} \neq 0$).

For rest of the cases $d_{ij} = 0$.

Sixth step is the assignment of masses and inertia tensors for every body as following:

$$m_1 = m_2 = 3\ kg\ ,\ m_3 = 2\ kg\ ,\ m_4 = m_5 = 1.5\ kg \qquad (21)$$

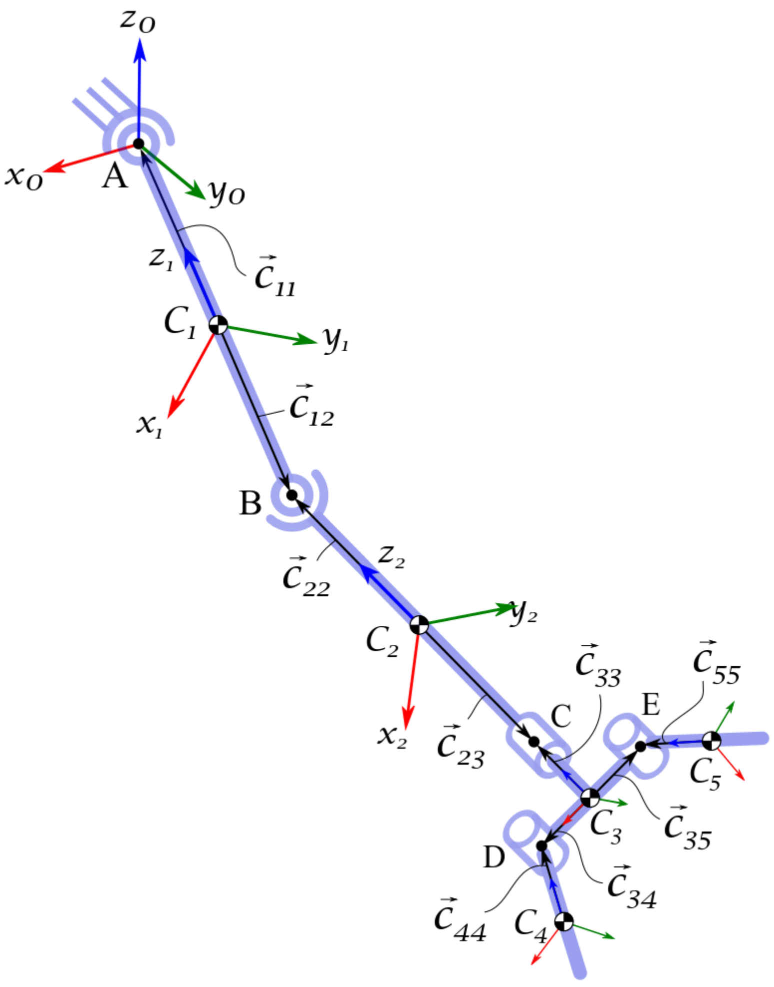


Figure 5 – Hinge vectors $c_{i\alpha}$

Tensor inertia J_i relative to local central axis,

$$J_1^{(1)} = \begin{bmatrix} J_{1xx}^{(1)} & 0 & 0 \\ 0 & J_{1yy}^{(1)} & 0 \\ 0 & 0 & J_{1zz}^{(1)} \end{bmatrix}, \quad J_2^{(2)} = \cdots, \quad J_3^{(3)} = \cdots \tag{22}$$

Sixth step is to determine the barycentre for body i. To do this it is required to determine the vector b_{i0} by using following expression:

$$b_{i0} = \frac{\sum_{j=1}^{5} d_{ij} m_j}{M}\ , where\ M = \sum_j m_j \tag{23}$$

Seventh step is to calculate tensor inertia for augmented bodies relative to the preceding hinge point can be determined as,

$$K_i = J_i + \sum_{k=1}^{n} m_k({d_{ik}}^2 E - d_{ik} d_{ik}) \tag{24}$$

Consequently tensor inertia for each body can be calculated on the basis of following conditions,

$$K_{ij} = \begin{cases} K_i, & i = j, \\ M\big(b_{j0} d_{ij} E - b_{j0} d_{ij}\big), & s_i < s_j, \\ M\big(d_{ji} b_{i0} E - d_{ji} b_{i0}\big), & s_j > s_i, \\ 0, & for\ other\ cases. \end{cases} \tag{25}$$

Eighth step is to construct matrix for $p_{\alpha i}$ vectors as following:

$$p^T = \begin{bmatrix} (p_A^T)_{3\times3} & (0)_{3\times3} & (0)_{3\times1} & (0)_{3\times1} & (0)_{3\times1} \\ (0)_{3\times3} & (p_B^T)_{3\times3} & (0)_{3\times1} & (0)_{3\times1} & (0)_{3\times1} \\ (0)_{3\times3} & (0)_{3\times3} & (p_C^T)_{3\times1} & (0)_{3\times1} & (0)_{3\times1} \\ (0)_{3\times3} & (0)_{3\times3} & (0)_{3\times1} & (p_D^T)_{3\times1} & (0)_{3\times1} \\ (0)_{3\times3} & (0)_{3\times3} & (0)_{3\times1} & (0)_{3\times1} & (p_E^T)_{3\times1} \end{bmatrix} \tag{26}$$

Where, p-vectors for individual hinge can be written as,

$${p_A}^T = \begin{bmatrix} \sin q_{12} \sin q_{13} & \cos q_{13} & 0 \\ \sin q_{12} \cos q_{13} & -\sin q_{13} & 0 \\ \cos q_{12} & 0 & 1 \end{bmatrix} \tag{27}$$

$${p_B}^T = \begin{bmatrix} \sin q_{22} \sin q_{23} & \cos q_{23} & 0 \\ \sin q_{22} \cos q_{23} & -\sin q_{23} & 0 \\ \cos q_{22} & 0 & 1 \end{bmatrix} \tag{28}$$

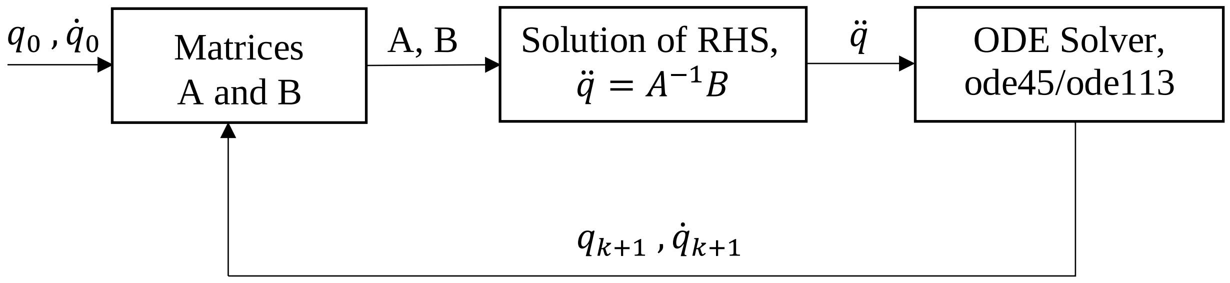
$${p_C}^T = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{29}$$

$${p_D}^T = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \tag{30}$$

$${p_E}^T = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \tag{31}$$

Using equations (18) – (26) matrices A and B can be computed for all five bodies from the equations (11) – (13).

Finally, the above mentioned algorithm is implemented in MATLAB to solve generated ordinary differential equations repeatedly for the given interval of time and initial conditions (See Appendix).



IV. Results

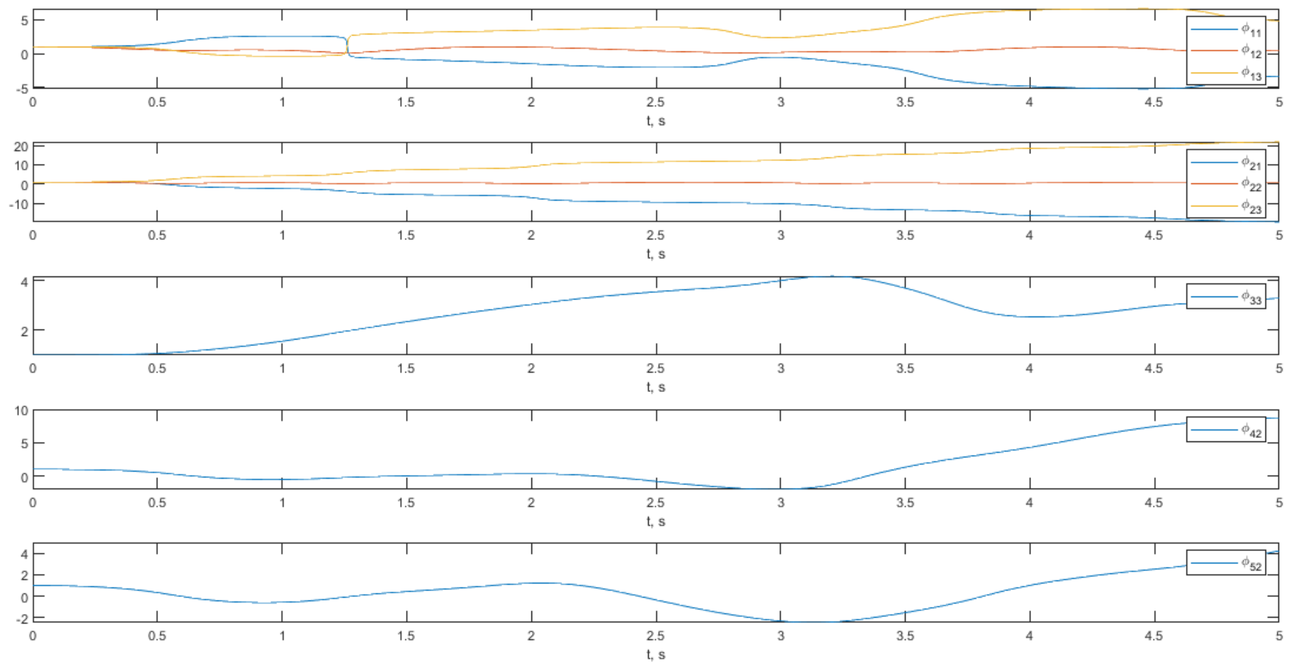


Figure 6 Time plot of generalized coordinates for initial conditions $q_0 = \{1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0\}$

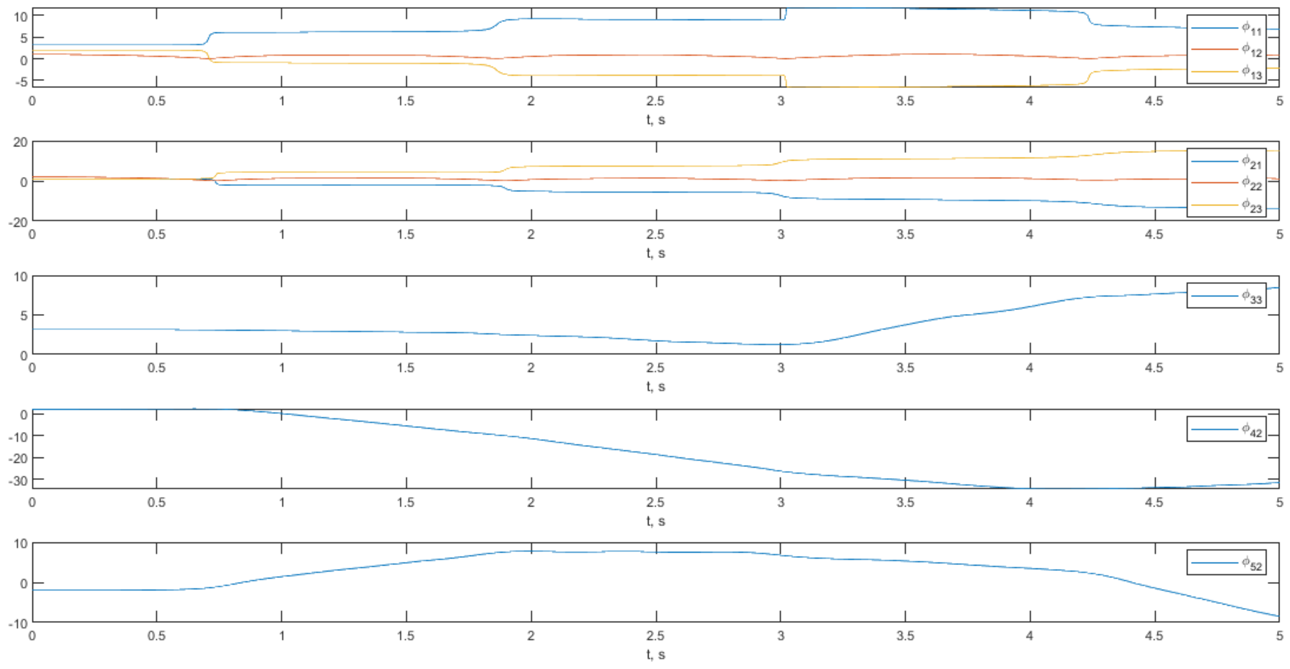


Figure 7 Time plot of generalized coordinates for initial conditions $q_0 = \{\pi, 1,2,1,2,1, \pi, 2, -2,0,0,0,0,0,0,0,0,0\}$

V. Validation of results

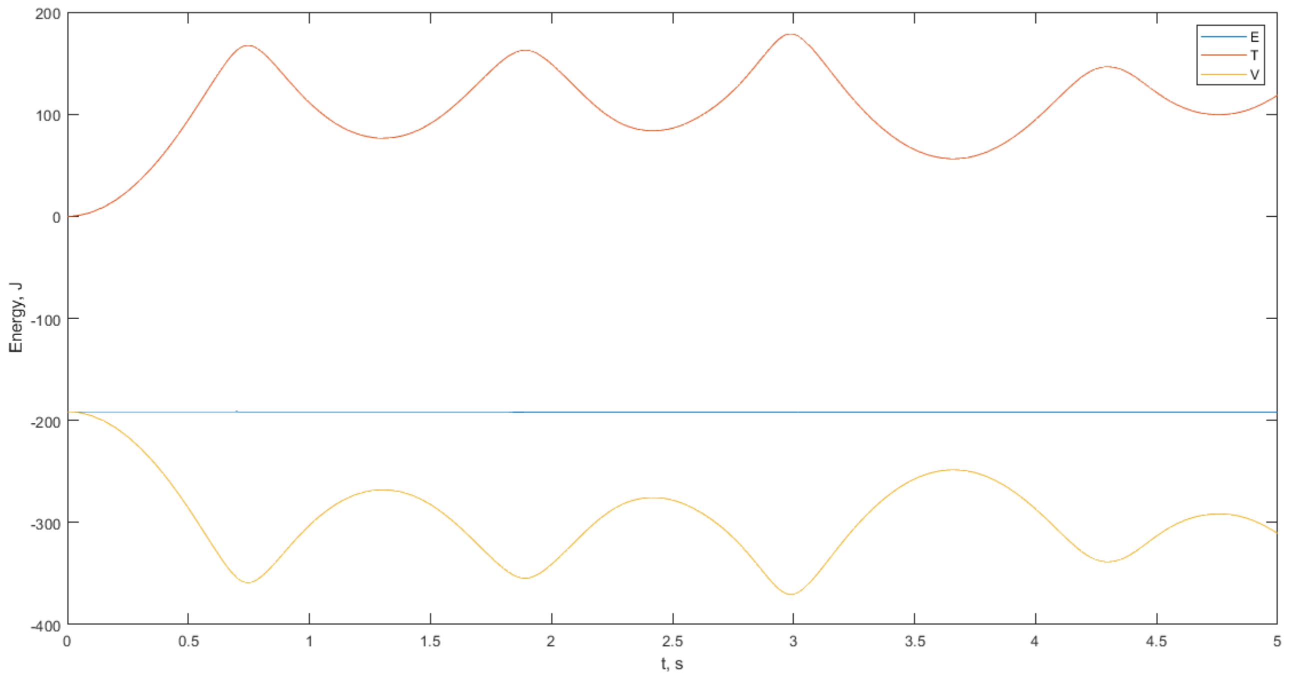


Figure 8 Energy plot – Total energy (E), Kinetic energy (T) and potential energy (V) for initial conditions $q_0 = \{\pi, 1,2,1,2,1, \pi, 2, -2,0,0,0,0,0,0,0,0,0\}$

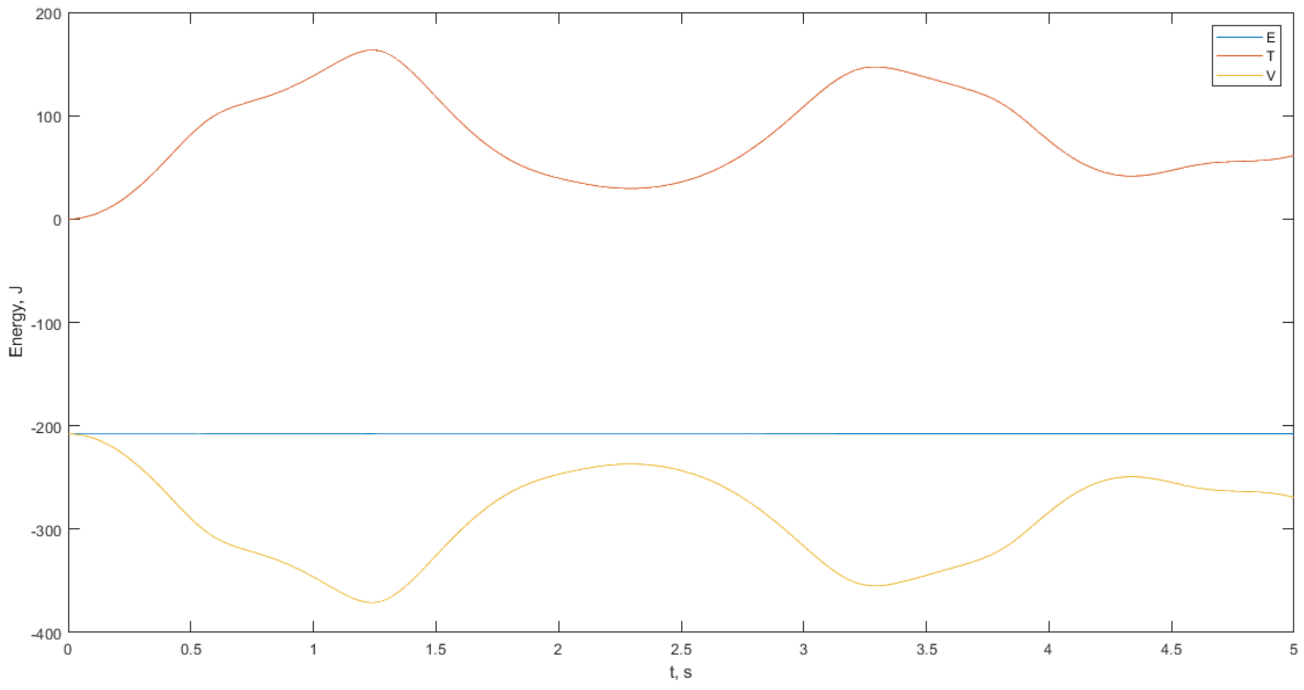


Figure 9 Energy plot – Total energy (E), Kinetic energy (T) and potential energy (V) for initial conditions $q_0 = \{1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0\}$

Appendix

A. Pre-processing / preproc.m

```
%% ==========================================================================
p.N  = sum(p.na);
p.iq = [(cumsum(p.na)-p.na+1) p.na];
p.T  = inv(p.S);

% d_{ji} column vectors in j-CS
p.d  = zeros(3,p.n,p.n);
for i = 1:p.n
    for j = 1:p.n
        for a = 1:p.n
            p.d(:,j,i) = p.d(:,j,i) + p.T(a,i)*p.S(j,a)*p.C(:,j,a);
        end
    end
end
% b_{i0} column vectors in i-CS
p.b  = zeros(3,p.n);
for i = 1:p.n
    for j=1:p.n
        p.b(:,i) = p.b(:,i) + p.d(:,i,j)*p.mass(j);
    end
    p.b(:,i) = p.b(:,i)/sum(p.mass);
end


% Diagonal blocks of K tensor in i-CS
p.Kii = zeros(3,3,p.n);
for i=1:p.n
    p.Kii(:,:,i) = p.I(:,:,i);
    for k=1:p.n
        p.Kii(:,:,i) = p.Kii(:,:,i) +
p.mass(k)*(p.d(:,i,k)'*p.d(:,i,k)*eye(3) - p.d(:,i,k)*p.d(:,i,k)');
    end
end
```

B. Model / model.m

```
% ==========================================================================
% Model description
% ==========================================================================
% Number of bodies
p.n  = 5;

% Numbers DOF for each joint
p.na = [3;3;1;1;1];

% Structure matrix
p.S0 = [1 0 0 0 0];
p.S  = [-1  1  0  0  0
         0 -1  1  0  0
         0  0 -1  1  1
         0  0  0 -1  0
         0  0  0  0 -1];
```

```
% Joint vectors
p.C  = zeros(3,p.n,p.n);
p.C(:,1,1)  = [0;0;+1];
p.C(:,1,2)  = [0;0;-1];
p.C(:,2,3)  = [0;0;-1];
p.C(:,2,2)  = [0;0;+1];
p.C(:,3,3)  = [0;0;+1];
p.C(:,3,4)  = [+0.5;0;0];
p.C(:,3,5)  = [-0.5;0;0];
p.C(:,4,4)  = [0;0;+0.5];
p.C(:,5,5)  = [0;0;+0.5];

% Masses (Critical parameter - choices impart big changes in solution)
p.mass = [3;3;2;1.5;1.5];

% Inertia tensors (Critical parameter - choices impart big changes in
% solution)
p.I    = zeros(3,3,p.n);
p.I(:,:,1) = p.mass(1)*diag([4/3,4/3,0.6*4/2]);     % Iz = 60% of (Ix,Iy)
p.I(:,:,2) = p.mass(2)*diag([4/3,4/3,0.6*4/3]);     % Iz = 60% of (Ix,Iy)
p.I(:,:,3) = p.mass(3)*diag([1/6,5/24,1/24]);
p.I(:,:,4) = p.mass(4)*diag([1/3,1/3,0.6*1/3]);     % Iz = 60% of (Ix,Iy)
p.I(:,:,5) = p.mass(5)*diag([1/3,1/3,0.6*1/3]);     % Iz = 60% of (Ix,Iy)

% Kinematics
p.A  = cell(p.n,1);
p.p = cell(p.n,1);
p.pw = cell(p.n,1);

% Basic rotations
Ax = @(x) [1 0 0; 0 cos(x) -sin(x); 0 sin(x) cos(x)];
Ay = @(x) [cos(x) 0 sin(x); 0 1 0; - sin(x) 0 cos(x)];
Az = @(x) [cos(x) -sin(x) 0; sin(x) cos(x) 0; 0 0 1];

% 0-1 (1 to 0) joint (spherical)
p.A{1} = @(q) Az(q(1))*Ax(q(2))*Az(q(3));

% 1-2 (2 to 1) joint (spherical)
p.A{2} = @(q) Az(q(1))*Ax(q(2))*Az(q(3));

% 2-3 (3 to 2) joint (cylindrical)
p.A{3} = @(q) Az(q(1));

% 3-4 (4 to 3) joint (cylindrical)
p.A{4} = @(q) Ay(q(1));

% 3-5 (5 to 3) joint (cylindrical)
p.A{5} = @(q) Ay(q(1));

% p-vectors
% 0-1 spherical joint: 3 x 3
p.p{1} = @(q) [p.A{1}(q)'*[0;0;1] p.A{1}([0,q(2),q(3)])'*[1;0;0] [0;0;1]];

% 1-2 spherical joint: 3 x 3
p.p{2} = @(q) [p.A{1}(q)'*[0;0;1] p.A{1}([0,q(2),q(3)])'*[1;0;0] [0;0;1]];

% 2-3 cylindrical joint: 3 x 1
p.p{3} = @(q) [0;0;1];
```

```
% 3-4 cylindrical joint: 3 x 1
p.p{4} = @(q) [0;1;0];

% 3-5 cylindrical joint: 3 x 1
p.p{5} = @(q) [0;1;0];

% Relative angular velocity
p.Wr{1} = @(q,dq)  p.p{1}(q)*[dq(1);dq(2);dq(3)];
p.Wr{2} = @(q,dq)  p.p{2}(q)*[dq(1);dq(2);dq(3)];
p.Wr{3} = @(q,dq)  p.p{3}(q)*dq;
p.Wr{4} = @(q,dq)  p.p{4}(q)*dq;
p.Wr{5} = @(q,dq)  p.p{5}(q)*dq;

% Relative angular acceleration
p.pw{1} = @(q,dq) [sin(q(3))*dq(2)*(cos(q(2))*dq(1)-
dq(3))+cos(q(3))*sin(q(2))*dq(1)*dq(3);
                   cos(q(3))*dq(2)*(cos(q(2))*dq(1)-dq(3))-
sin(q(3))*sin(q(2))*dq(1)*dq(3);
                   -sin(q(2))*dq(2)*dq(1)];
p.pw{2} = @(q,dq) [sin(q(3))*dq(2)*(cos(q(2))*dq(1)-
dq(3))+cos(q(3))*sin(q(2))*dq(1)*dq(3);
                   cos(q(3))*dq(2)*(cos(q(2))*dq(1)-dq(3))-
sin(q(3))*sin(q(2))*dq(1)*dq(3);
                   -sin(q(2))*dq(2)*dq(1)];
p.pw{3} = @(q,dq) [0;0;0];
p.pw{4} = @(q,dq) [0;0;0];
p.pw{5} = @(q,dq) [0;0;0];

%% ====================================================================
% Preprocessing
% =====================================================================
preproc;

%% ====================================================================
% Simulation
% =====================================================================
% Initial conditions
q0 = [pi;1;2;1;2;1;pi;2;-2;0;0;0;0;0;0;0;0;0];
% Start integration process
[t, q] = ode113(@(t,q) dqdtcw(t,q,p), [0 5], q0);
fprintf('OK\n');

%% ====================================================================
% Postprocessing
% =====================================================================
figure;
subplot(511);
plot(t, q(:,1:3));legend('\phi_{11}','\phi_{12}','\phi_{13}');xlabel('t,
s');
subplot(512);
plot(t, q(:,4:6));legend('\phi_{21}','\phi_{22}','\phi_{23}');xlabel('t,
s');
subplot(513);
plot(t, q(:,7));legend('\phi_{33}');xlabel('t, s');
subplot(514);
plot(t, q(:,8));legend('\phi_{42}');xlabel('t, s');
subplot(515);
plot(t, q(:,9));legend('\phi_{52}');xlabel('t, s');
```

```
%% ========================================================================
% Validation - Checking conservation of the energy
% ========================================================================
[E, T, V] = Energy(q,p);
figure;
plot(t, E, t, T, t, V); legend('E', 'T', 'V'); xlabel('t, s');
ylabel('Energy, J');

%% ========================================================================
% Export data for animation
% ========================================================================
csvwrite('E:\MATLAB docs\MBD\results.csv',q)
csvwrite('E:\MATLAB docs\MBD\vec_d.csv',p.d)
```

C. Right hand side function for ODE / dqdtcw.m

```
% ODE function
function dq = dqdtcw(t,q,p)
% t - time
% q - state
% p - parameters

% Net force column vectros on each body in 0 frame
Fnet  = getForces(t,q,p);
% Net torques column vectros on each body in body frame
Mnet  = getTorques(t,q,p);

% Slice functions:
% get phi array by hinge index
phi  = @(i) q(p.iq(i,1) : p.iq(i,1)+p.iq(i,2)-1);
% get dphi array by hinge index
dphi = @(i) q(p.iq(i,1)+p.N : p.iq(i,1)+p.iq(i,2)+p.N-1);

% get A0i transform matrices (i body to 0)
A0i = zeros(3,3,p.n);
for i=1:p.n
    A0i(:,:,i) = eye(3);
    for a=i:-1:1
        if p.T(a,i) ~= 0
            A0i(:,:,i) = p.A{a}(phi(a))*A0i(:,:,i);
        end
    end
end

% K matrix
K = zeros(3,3,p.n,p.n);
Mass = sum(p.mass);

% Angular velocity
w      = zeros(3,p.n);

%w_in_0 = zeros(3,p.n);
for i=1:p.n
    for j=1:p.n
        w(:,i) = w(:,i) - A0i(:,:,j)*(p.T(j,i)*p.Wr{j}(phi(j),dphi(j))); %
+ w0 ;
        if i==j
            % Kii in 0 frame
            K(:,:,i,j) = A0i(:,:,i)*p.Kii(:,:,i)*A0i(:,:,i)';
        else
            if p.T(i,j)~=0
               % s_i < s_j
               K(:,:,i,j) =
Mass*((A0i(:,:,j)*p.b(:,j))'*(A0i(:,:,i)*p.d(:,i,j))*eye(3)-
A0i(:,:,j)*p.b(:,j)*(A0i(:,:,i)*p.d(:,i,j))');
            end
            if p.T(j,i)~=0
               % s_j < s_i
               K(:,:,i,j) =
Mass*((A0i(:,:,j)*p.d(:,j,i))'*(A0i(:,:,i)*p.b(:,i))*eye(3)-
A0i(:,:,j)*p.d(:,j,i)*(A0i(:,:,i)*p.b(:,i))');
            end
        end
    end
end

% p*T matrix
pT = zeros(3,p.N,p.n);
i = 1;
for ib=1:p.n
    % to 0 frame
    pblock = A0i(:,:,ib)*p.p{ib}(phi(ib));
    for ia = 1:p.na(ib)
        for k = 1:p.n
            pT(:,i,k) = pblock(:,ia)*p.T(ib,k);
        end
        i = i + 1;
    end
end
% p*T*K matrix
% TODO - Check
pTK = zeros(3,p.N,p.n);
for i=1:p.N
    for j=1:p.n
        for k=1:p.n
            pTK(:,i,j) = pTK(:,i,j) + (pT(:,i,k)'*K(:,:,k,j))';
        end
    end
end

%
% A matrix
%
A = zeros(p.N,p.N);
for i=1:p.N
    for j=1:p.N
        for k=1:p.n
            A(i,j) = A(i,j) + pTK(:,i,k)'*pT(:,j,k);
        end
    end
end

%
% M'
%
Mp = zeros(3,p.n);
for i=1:p.n
    Mp(:,i) = - tilde(w(:,i))*K(:,:,i,i)*w(:,i);
    for j=1:p.n
```

```
        if p.T(i,j)~=0 && i~=j
            % s_i < s_j
            Mp(:,i) = Mp(:,i) -
Mass*cross(A0i(:,:,i)*p.d(:,i,j),cross(w(:,j),cross(w(:,j),A0i(:,:,j)*p.b(:
,j))));
        end
        if p.T(i,j)~=0
            % s_i < =  s_j
            Mp(:,i) = Mp(:,i) - cross(A0i(:,:,i)*p.d(:,i,j),Fnet(:,j));
        end
        if p.T(j,i)~=0 && i~=j
            % s_j < s_i
            Mp(:,i) = Mp(:,i) -
Mass*cross(A0i(:,:,i)*p.b(:,i),cross(w(:,j),cross(w(:,j),A0i(:,:,j)*p.d(:,j
,i))));
        end
    end
end


%
% B matrix
%
pTMpM = zeros(p.N,1);
for i=1:p.N
    for j=1:p.n
        pTMpM(i) = pTMpM(i) - dot(pT(:,i,j),Mp(:,j)+A0i(:,:,j)*Mnet(:,j));
    end
end
% f
f   = zeros(3,p.n);
for i=1:p.n
   f(:,i) = cross(w(:,i),A0i(:,:,i)*p.Wr{i}(phi(i),dphi(i))) +
A0i(:,:,i)*p.pw{i}(phi(i),dphi(i));
end
% TTf
TTf = zeros(3,p.n);
for i=1:p.n
    for j=1:p.n
        TTf(:,i) = TTf(:,i) + p.T(j,i)*f(:,j); % -w0*1n
    end
end
% pTKTTf
pTKTTf = zeros(p.N,1);
for i=1:p.N
    for j=1:p.n
        pTKTTf(i) = pTKTTf(i) - dot(pTK(:,i,j),TTf(:,j));
    end
end
% B
B = pTKTTf + pTMpM;
%
% Solve Ax=B for x
%
d2phi = A\B;

dq = [q(p.N+1:2*p.N);d2phi];

end
```

D. Tilde function / tilde.m

```
function res = tilde(w)

res = [0    -w(3)     w(2);
        w(3)    0    -w(1);
       -w(2)  w(1)      0];

end
```

E. Force function / getForces.m

```
function F = getForces(t,q,p)

% in 0 frame
F = zeros(3,p.n);
for i=1:p.n
   F(:,i) = [0;0;-
p.mass(i)*9.807];
end
end
```

F. Torque function / getToques.m

```
function M = getTorques(t,q,p)
% in body frame

    M = zeros(3,p.n);
end
```

G. Energy function / Energy.m

```
function [E, T, U] = Energy(q, p)
%% ========================================================================
% Angular velocity
w = zeros(3, p.n, size(q,1));

for k=1:size(q,1)
    phi  = @(i) q(k,p.iq(i,1) : p.iq(i,1)+p.iq(i,2)-1);
    dphi = @(i) q(k,p.iq(i,1)+p.N : p.iq(i,1)+p.iq(i,2)+p.N-1);

    A0i = zeros(3,3,p.n);
    for i=1:p.n
        A0i(:,:,i) = eye(3);
        for a=i:-1:1
            if p.T(a,i) ~= 0
                A0i(:,:,i) = p.A{a}(phi(a))*A0i(:,:,i);
            end
        end
    end

    for i=1:p.n
        for j=1:p.n
            w(:,i,k) = w(:,i,k) -
A0i(:,:,j)*(p.T(j,i)*p.Wr{j}(phi(j),dphi(j)));
        end
    end
end

%% ========================================================================
% Linear velocity
v1 = cell(size(q,1),1);
v2 = cell(size(q,1),1);
v3 = cell(size(q,1),1);
```

```
v4 = cell(size(q,1),1);
v5 = cell(size(q,1),1);

for k=1:size(q,1)
    phi  = @(i) q(k,p.iq(i,1) : p.iq(i,1)+p.iq(i,2)-1);
    dphi = @(i) q(k,p.iq(i,1)+p.N : p.iq(i,1)+p.iq(i,2)+p.N-1);
    % transfomraion matrix (body i to 0)
    A0i = zeros(3,3,p.n);
    for j=1:p.n
        A0i(:,:,j) = eye(3);
        for a=j:-1:1
            if p.T(a,j) ~= 0
                A0i(:,:,j) = p.A{a}(phi(a))*A0i(:,:,j);
            end
        end
    end
    % d_{ij} vectors
    d011 = A0i(:,:,1)*p.d(:,1,1);
    d012 = A0i(:,:,1)*p.d(:,1,2);
    d022 = A0i(:,:,2)*p.d(:,2,2);
    d023 = A0i(:,:,2)*p.d(:,2,3);
    d033 = A0i(:,:,3)*p.d(:,3,3);
    d034 = A0i(:,:,3)*p.d(:,3,4);
    d035 = A0i(:,:,3)*p.d(:,3,5);
    d044 = A0i(:,:,4)*p.d(:,4,4);
    d055 = A0i(:,:,5)*p.d(:,5,5);
    % auxiliary angular velocity
    Wr = cell(5,1);
    for i=1:p.n
        Wr{i} = A0i(:,:,i)*p.Wr{i}(phi(i),dphi(i));
    end
    w1 = Wr{1};
    w2 = w1 + Wr{2};
    w3 = w2 + Wr{3};
    w4 = w3 + Wr{4};
    w5 = w3 + Wr{5};
    % linear velocity
    v1{k} = cross(w1,-d011);
    v2{k} = cross(w1,-d012) + cross(w2,-d022);
    v3{k} = cross(w1,-d012) + cross(w2,-d023) + cross(w3,-d033);
    v4{k} = cross(w1,-d012) + cross(w2,-d023) + cross(w3,-d034) +
cross(w4,-d044);
    v5{k} = cross(w1,-d012) + cross(w2,-d023) + cross(w3,-d035) +
cross(w5,-d055);
end

%% ==========================================================================
% Kinetic energy
Tw = zeros(size(q,1),1);    % rotational component
for k=1:size(q,1)
    phi  = @(i) q(k,p.iq(i,1) : p.iq(i,1)+p.iq(i,2)-1);
    % transfomraion matrix (body i to 0)
    A0i = zeros(3,3,p.n);
    for j=1:p.n
        A0i(:,:,j) = eye(3);
        for a=j:-1:1
            if p.T(a,j) ~= 0
                A0i(:,:,j) = p.A{a}(phi(a))*A0i(:,:,j);
            end
        end
    end
    % inertia tensor in body-0 frame
    I01 = A0i(:,:,1)*p.I(:,:,1)*A0i(:,:,1)';
    I02 = A0i(:,:,2)*p.I(:,:,2)*A0i(:,:,2)';
    I03 = A0i(:,:,3)*p.I(:,:,3)*A0i(:,:,3)';
    I04 = A0i(:,:,4)*p.I(:,:,4)*A0i(:,:,4)';
    I05 = A0i(:,:,5)*p.I(:,:,5)*A0i(:,:,5)';
    % individual rotational energies
    Tw_1 = 0.5*w(:,1,k)'*I01*w(:,1,k);
    Tw_2 = 0.5*w(:,2,k)'*I02*w(:,2,k);
    Tw_3 = 0.5*w(:,3,k)'*I03*w(:,3,k);
    Tw_4 = 0.5*w(:,4,k)'*I04*w(:,4,k);
    Tw_5 = 0.5*w(:,5,k)'*I05*w(:,5,k);
    % total rotationl energy
    Tw(k) = Tw_1 + Tw_2 + Tw_3 + Tw_4 + Tw_5;
end

Tl = zeros(size(q,1),1);    % linear component
for k=1:size(q,1)
    v_1 = norm(v1{k});
    v_2 = norm(v2{k});
    v_3 = norm(v3{k});
    v_4 = norm(v4{k});
    v_5 = norm(v5{k});
    Tl(k) = 0.5*p.mass(1)*v_1*v_1 + 0.5*p.mass(2)*v_2*v_2 +
0.5*p.mass(3)*v_3*v_3 + 0.5*p.mass(4)*v_4*v_4 + 0.5*p.mass(5)*v_5*v_5;
end

% total kinetic energy
T = Tw + Tl;

%% =========================================================================
% Potential energy
V = zeros(size(q,1),1);
for k=1:size(q,1)
    phi  = @(i) q(k,p.iq(i,1) : p.iq(i,1)+p.iq(i,2)-1);
    % transfomraion matrix (body i to 0)
    A0i = zeros(3,3,p.n);
    for i=1:p.n
        A0i(:,:,i) = eye(3);
        for a=i:-1:1
            if p.T(a,i) ~= 0
                A0i(:,:,i) = p.A{a}(phi(a))*A0i(:,:,i);
            end
        end
    end
    % d_{ij} vectors
    d011 = A0i(:,:,1)*p.d(:,1,1);
    d012 = A0i(:,:,1)*p.d(:,1,2);
    d022 = A0i(:,:,2)*p.d(:,2,2);
    d023 = A0i(:,:,2)*p.d(:,2,3);
    d033 = A0i(:,:,3)*p.d(:,3,3);
    d034 = A0i(:,:,3)*p.d(:,3,4);
    d035 = A0i(:,:,3)*p.d(:,3,5);
    d044 = A0i(:,:,4)*p.d(:,4,4);
    d055 = A0i(:,:,5)*p.d(:,5,5);
    % distance of body i from body 0
    h1 = - d011;
    h2 = - d012 - d022;
    h3 = - d012 - d023 - d033;
    h4 = - d012 - d023 - d034 - d044;
    h5 = - d012 - d023 - d035 - d055;
    % sum of potential energies of 5 bodies
    V(k) = 9.807*(p.mass(1)*h1(3) + p.mass(2)*h2(3) + p.mass(3)*h3(3) +
p.mass(4)*h4(3) + p.mass(5)*h5(3));
end

% potential energy
U = V;

%% ========================================================================
% Total energy
E = T + U;

end
```

Reference

1. Wittenburg J, “Dynamics of Multibody Systems”, Springer, 2nd edition, 1977.
2. Yudintsev VV, “Lecture notes – Dynamics of rigid body and system of rigid bodies”, 2nd semester, 2018-20, Masters – mechanics and mathematical modelling, 2019, Samara University.