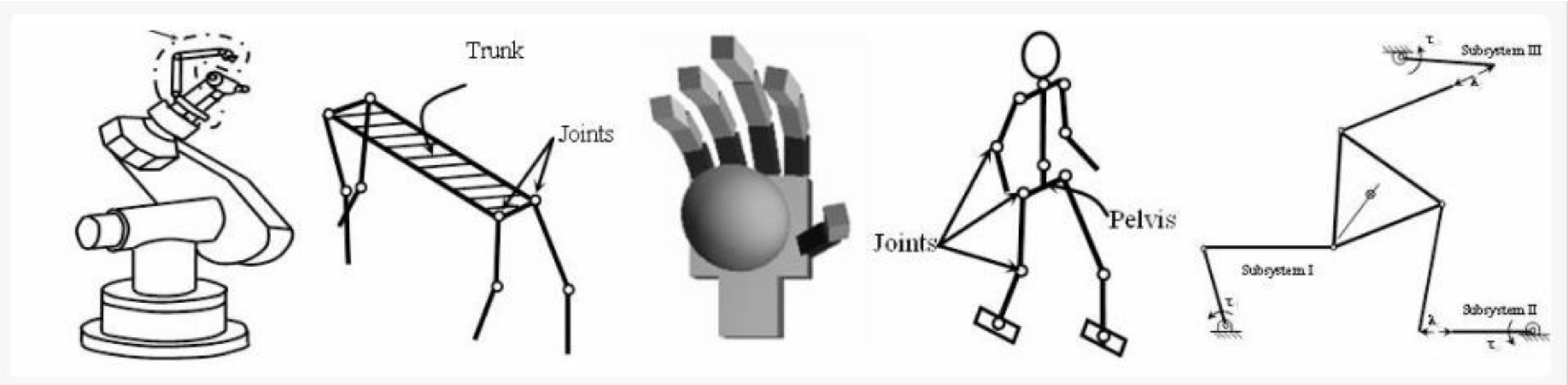
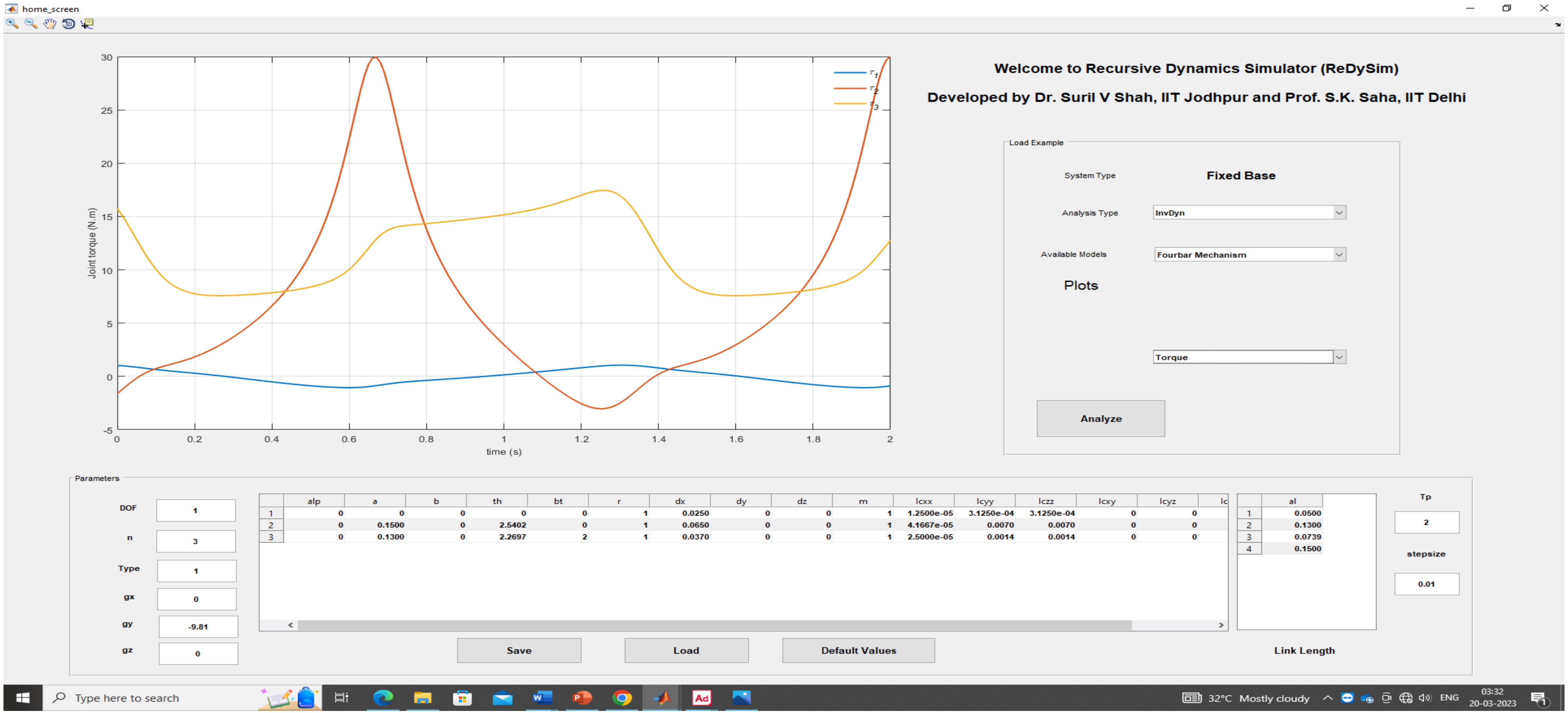
Recursive Dynamics Simulator (ReDySim)

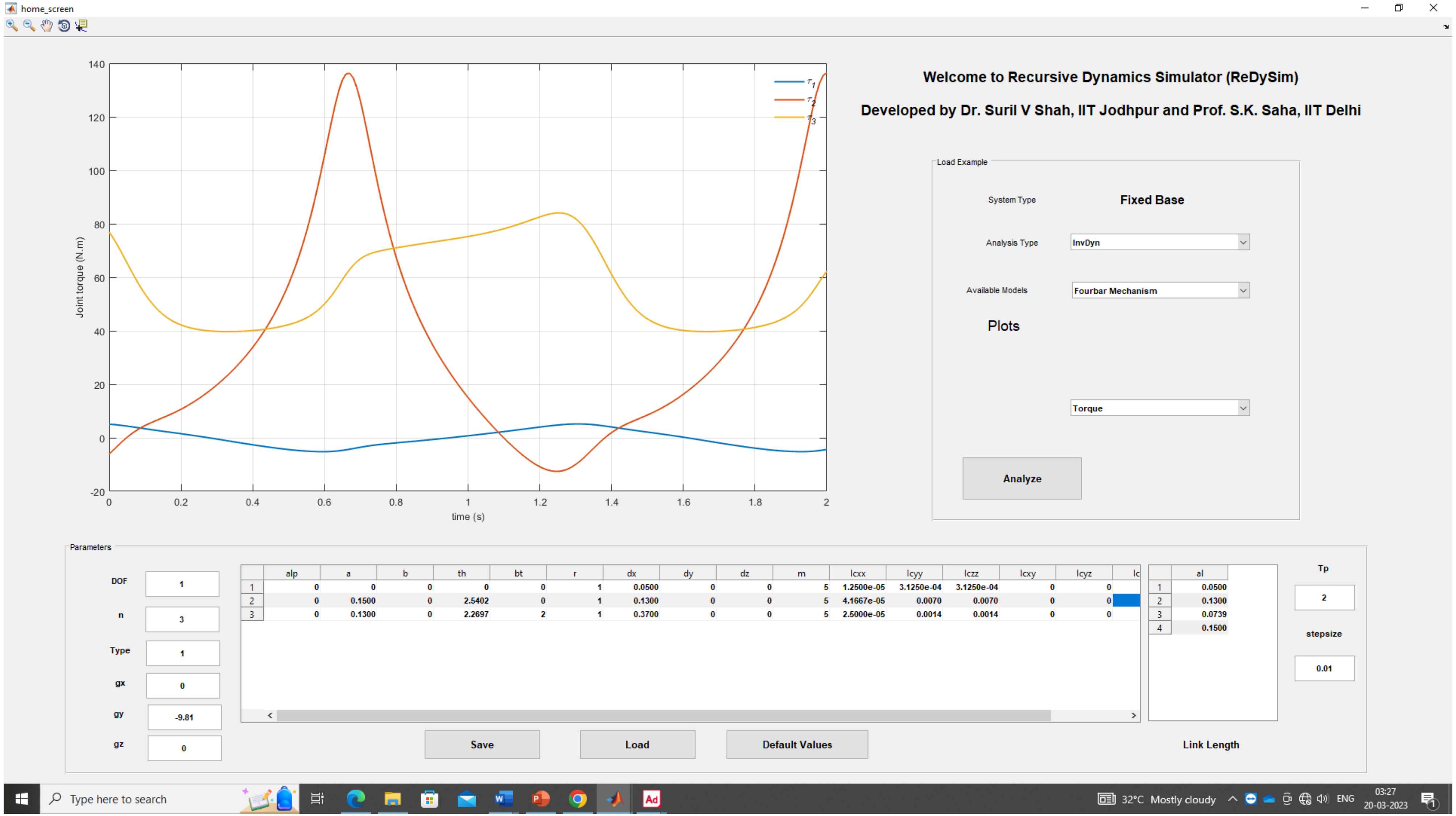
- MATLAB-based recursive solver for dynamic analysis of robotic and multibody systems
- Uses the concept of Decoupled Natural Orthogonal Complement (DeNOC) based approach for dynamic modeling
- Dynamic analysis can be performed without creating solid model
- Considerable improvements in terms of the computational time and numerical accuracy.



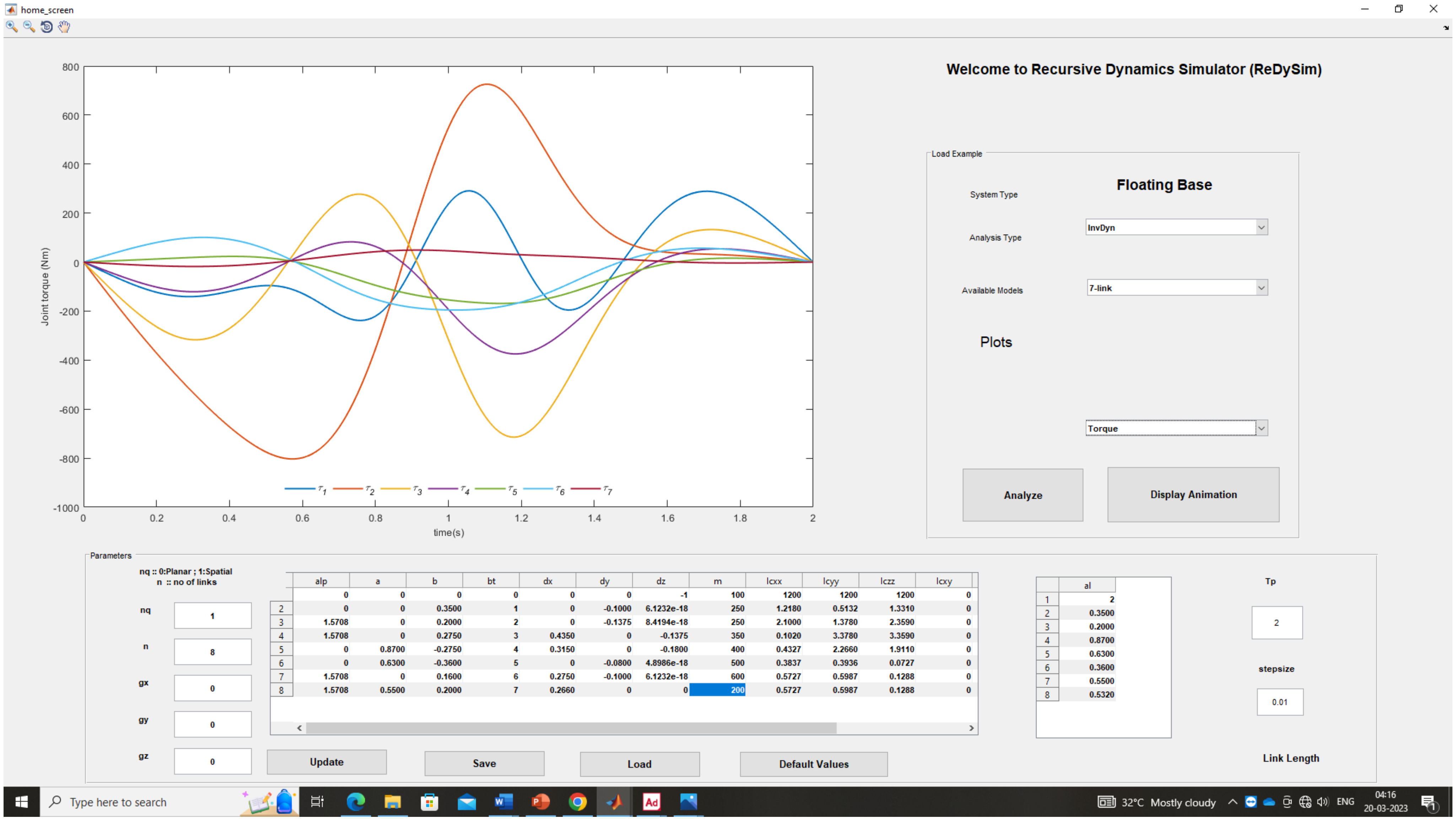
Recursive Dynamics Simulator (ReDySim)



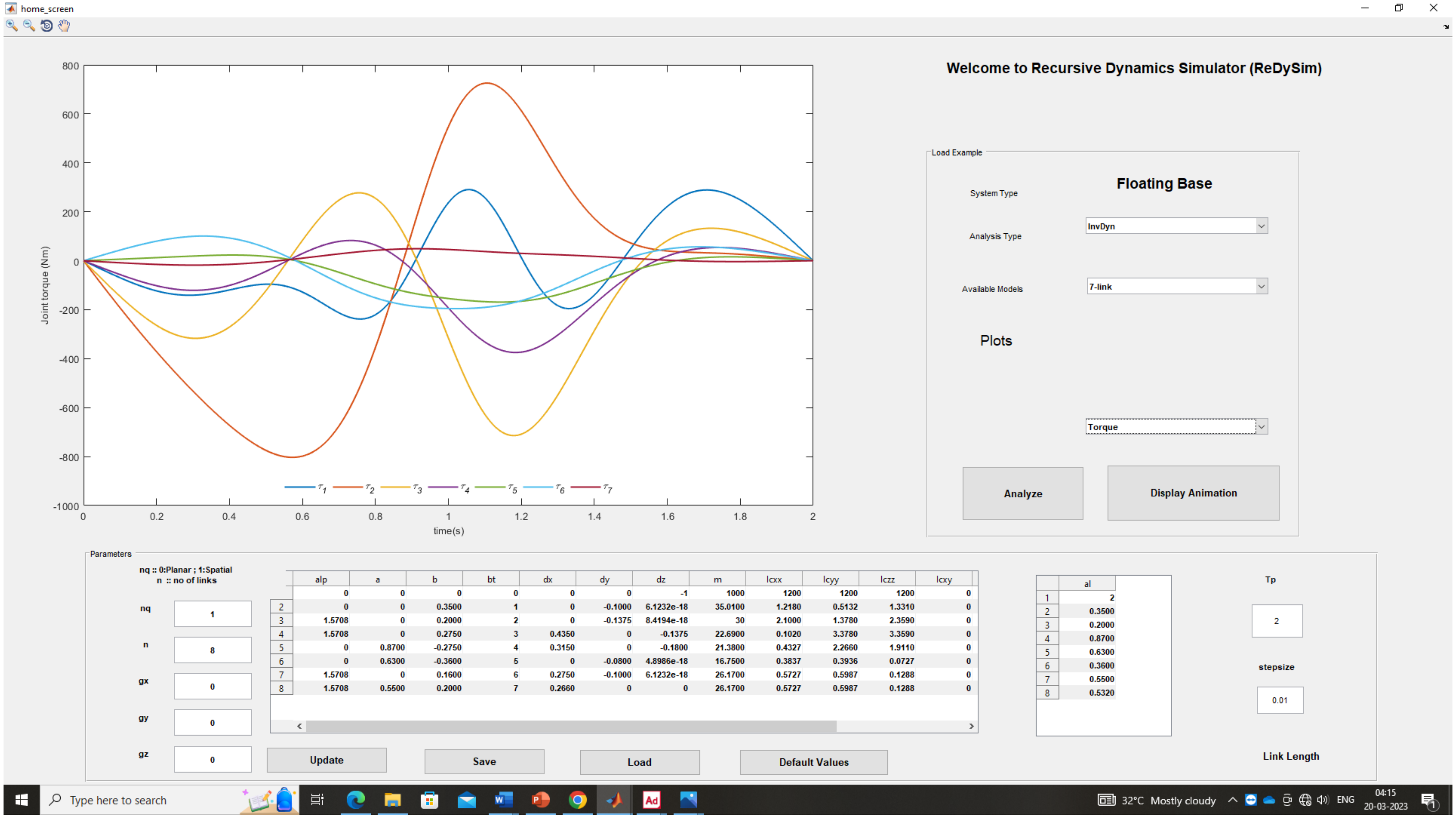
Recursive Dynamics Simulator (ReDySim)



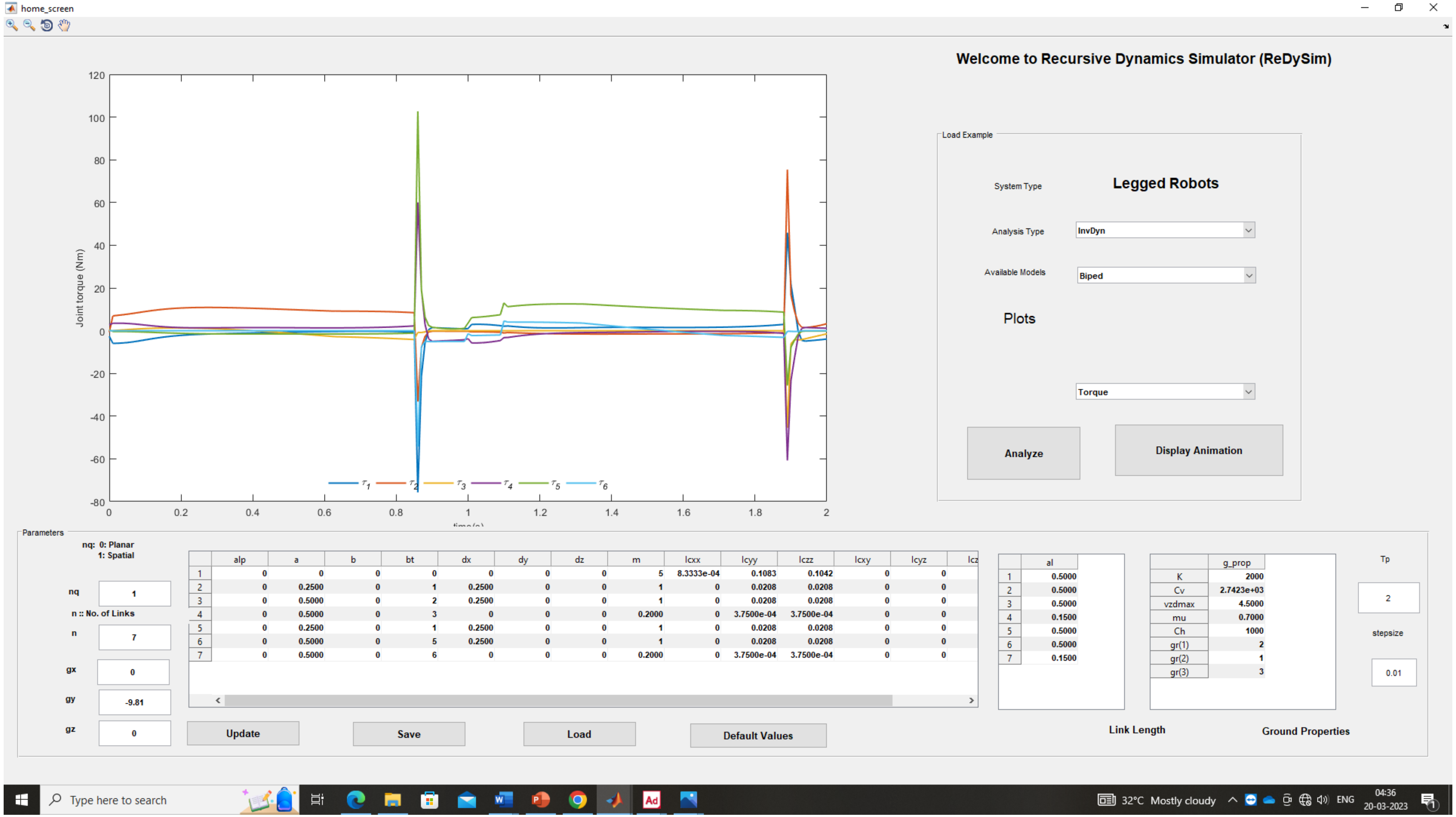
Recursive Dynamics Simulator (ReDySim)



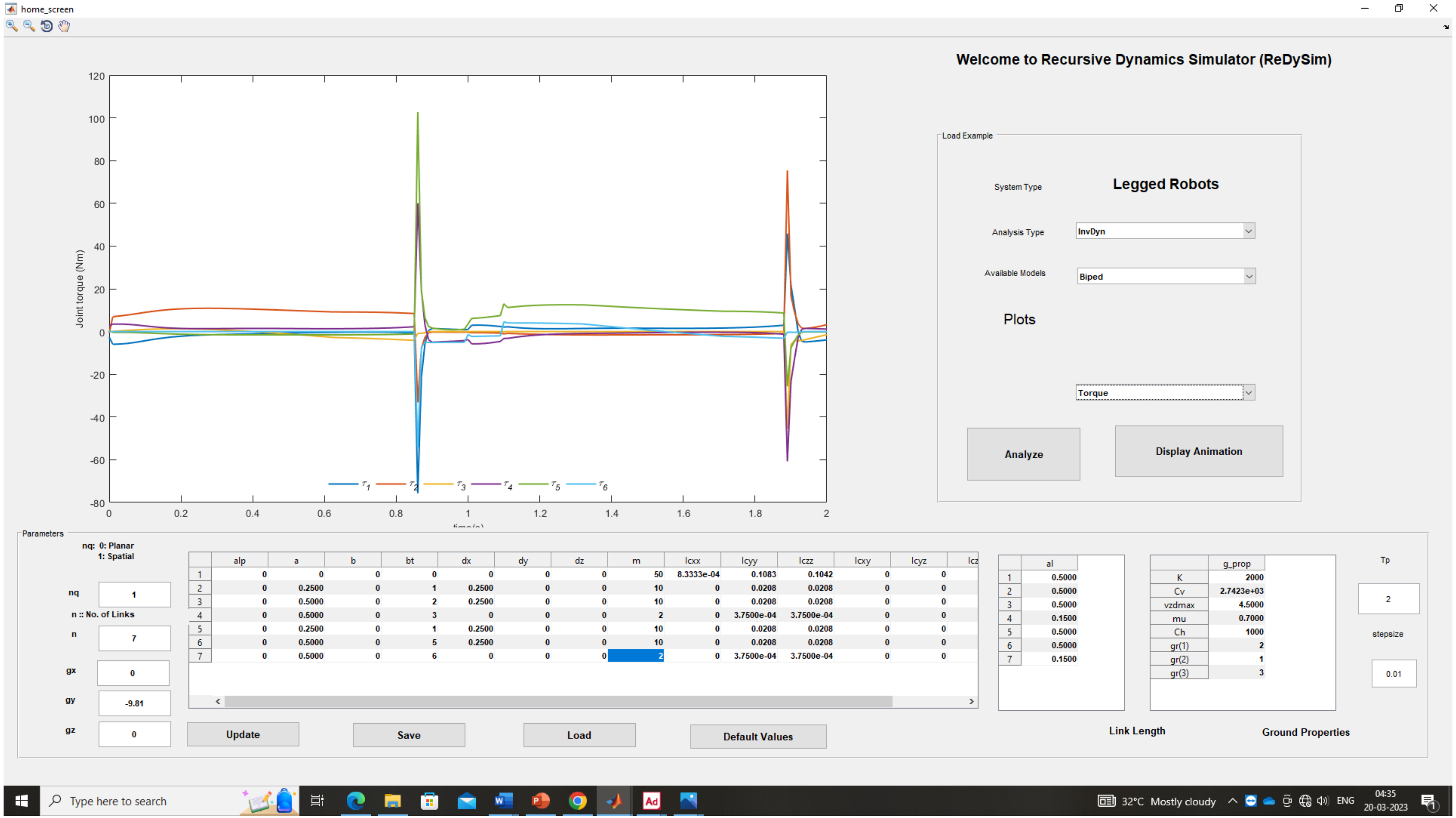
Recursive Dynamics Simulator (ReDySim)



Recursive Dynamics Simulator (ReDySim)



Recursive Dynamics Simulator (ReDySim)



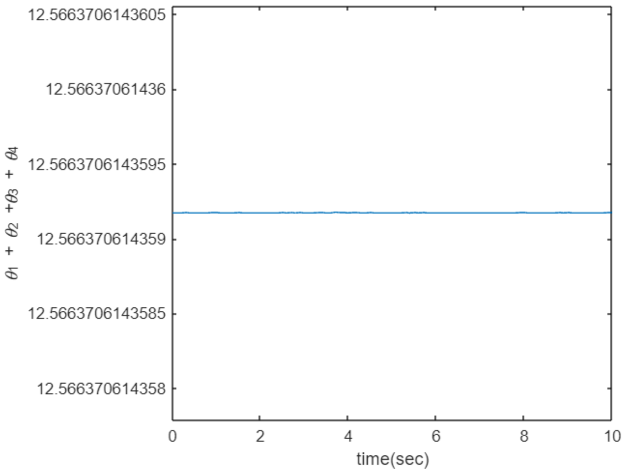
Recursive Dynamics Simulator (ReDySim)

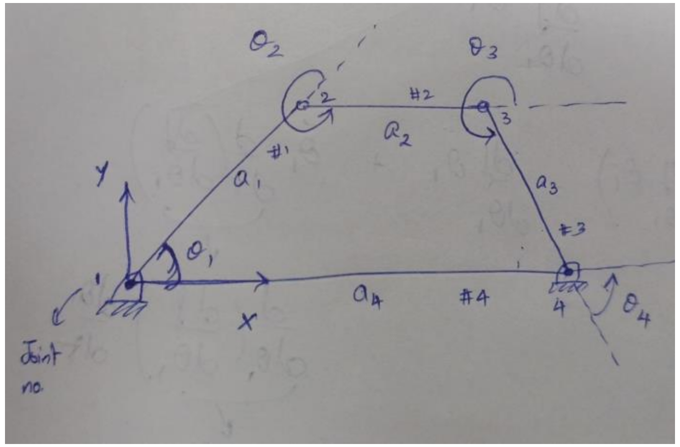
- MATLAB-based recursive solver for dynamic analysis of robotic and multibody systems
- Uses the concept of Decoupled Natural Orthogonal Complement (DeNOC) based approach for dynamic modeling
- Dynamic analyses can be performed without creating solid model
- considerable improvement in terms of the computational time and numerical accuracy.

Inverse Kinematics : 4 Bar Mechanism

- a_1=0.06m
- a_2=0.14m
- a_3=0.16m
- a_4=0.2m

- Initial and Given conditions :
  - $\theta_1 = 0\ and \theta_1 = pi/4$
  - $\dot{\theta}_1 = 1 \frac{rad}{sec}$
  - $\ddot{\theta}_1 = 0 \frac{rad}{sec^2}$

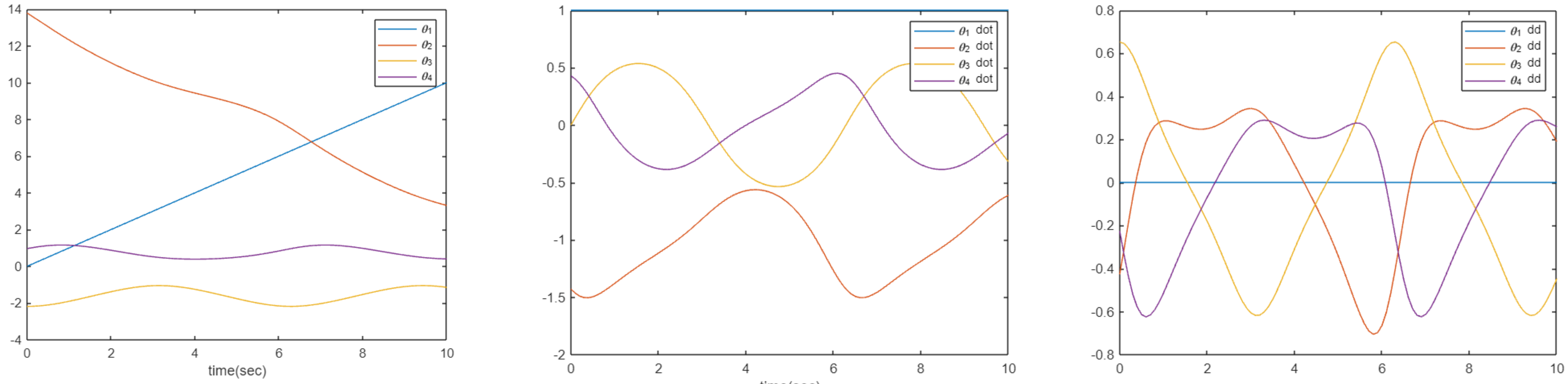




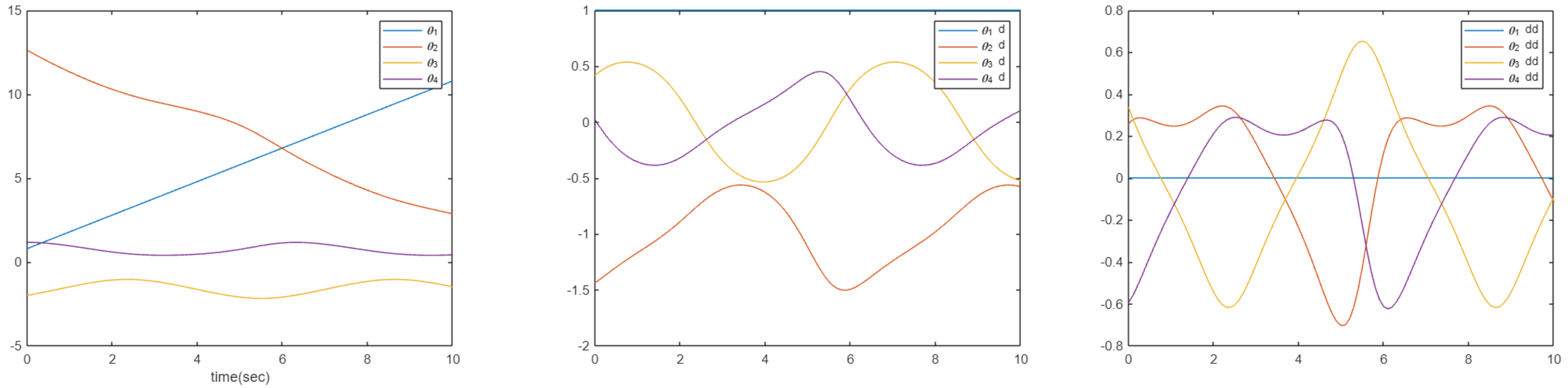

```
%Constraints
eqn1 = (l1*cos(theta_1)) + (l2*cos(theta_1+theta_2)) +
(l3*cos(theta_1+theta_2+theta_3)) - l4;    % x-coord of J-4
eqn2 = (l1*sin(theta_1)) + (l2*sin(theta_1+theta_2)) +
(l3*sin(theta_1+theta_2+theta_3));
eqn3 = theta_1 + theta_2 + theta_3 + theta_4 - (4*pi); % y-coord of J-4

solution = solve([eqn1,eqn2,eqn3],[theta_2,theta_3,theta_4]);
```

Simulation Results

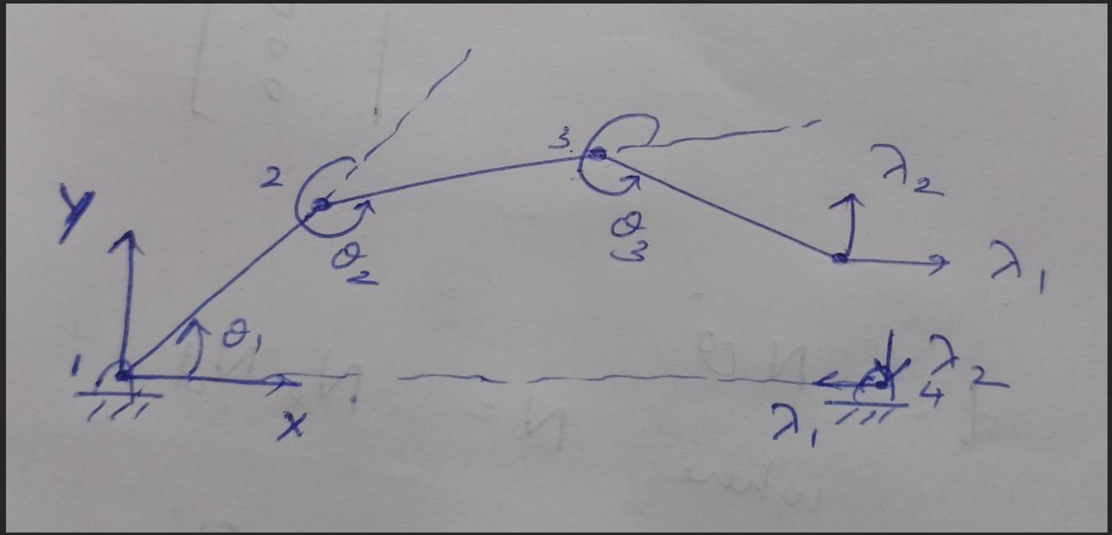
- For 0 initial angle :



- For 45^o initial angle :



Inverse Dynamics Methodology using DeNOC



$A_{i-1,I}$' matrix:

```
A1_2d = [Id vec_cross(-a1_2);ZO Id];
A2_3d = [Id vec_cross(-a2_3);ZO Id];
A3_4d = [Id vec_cross(-a3_4);ZO Id];
```

N_l and N_d matrices:

```
A1_2d_2 = [Id vec_cross(a1_2);ZO Id];
A2_3d_2 = [Id vec_cross(a2_3);ZO Id];
A_d2 = [Id_1 -1*A1_2d_2 ZO_1;ZO_1 Id_1 -1*A2_3d_2;ZO_1 ZO_1 Id_1];
Nu = inv(A_d2);
Nl = transpose(Nu);

Nd = [p1 ZO_c ZO_c;ZO_c p2 ZO_c;ZO_c ZO_c p3];
N = Nl*Nd;
```

Mass matrix:

```
%% COM positions
d1 = [l1*cos(theta_1)/2;l1*sin(theta_1)/2;0];
d2 = [l2*cos(theta_1+theta_2_1)/2;l2*sin(theta_1+theta_2_1)/2;0];
d3 =
[l3*cos(theta_1+theta_2_1+theta_3_1)/2;l3*sin(theta_1+theta_2_1+theta_3_1)/2;0];

M1 = [I1 m1*vec_cross(d1);-m1*vec_cross(d1) m1*Id];
M2 = [I2 m2*vec_cross(d2);-m2*vec_cross(d2) m2*Id];
M3 = [I3 m3*vec_cross(d3);-m3*vec_cross(d3) m3*Id];
```

Wrench Matrix and Final equation

Angular Velocity Cross product matrix:

```
%Finding ang velocities
ang1 = diff([0;0;theta_1],theta_1)*th1_d_i;
ang2 = diff([0;0;theta_1+theta_2_1],theta_1)*th1_d_i;
ang3 = diff([0;0;theta_1+theta_2_1+theta_3_1],theta_1)*th1_d_i;

angC_1 = [vec_cross(ang1) ZO;ZO vec_cross(ang1)];
angC_2 = [vec_cross(ang2) ZO;ZO vec_cross(ang2)];
angC_3 = [vec_cross(ang3) ZO;ZO vec_cross(ang3)];
```

External wrench:

```
w_1_e = [0;0;tau;0;-m1*g;0];
w_2_e = [0;0;0;0;-m2*g;0];
w_3_e = [0;0;0;0;-m3*g;0];
w_e = [w_1_e;w_2_e;w_3_e];
```

Wrench due to Lagrange multipliers:

```
w_1_lam = zeros(6,1);
w_2_lam = zeros(6,1);
w_4_lam = [0;0;0;lambda_1;lambda_2;0];
w_3_lam = A3_4d*w_4_lam;

w_lam = [w_1_lam;w_2_lam;w_3_lam];
```

Twist Matrix:

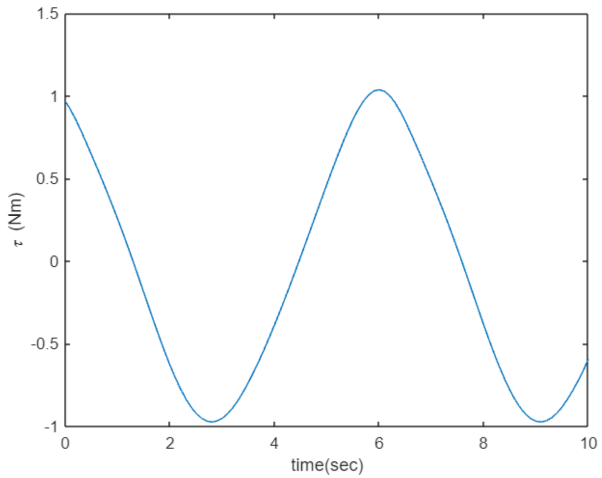
```
twist = Nl*Nd*diff(th_vec,theta_1)*th1_d_i;
```

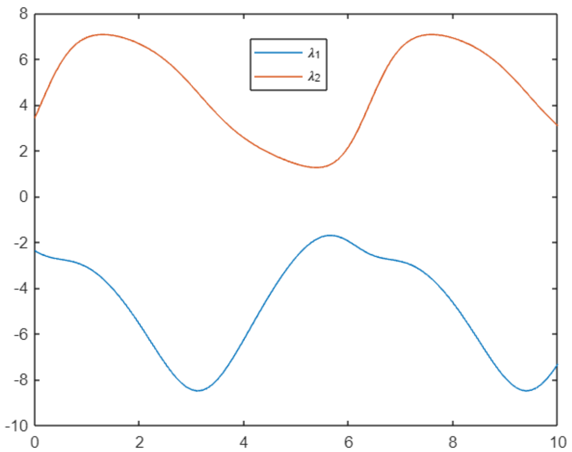
Final equation:

```
f_eqn = transpose(N)*((M_f*twist_d)+(W_f*M_f*E_f*twist)-(w_lam+w_e));
```

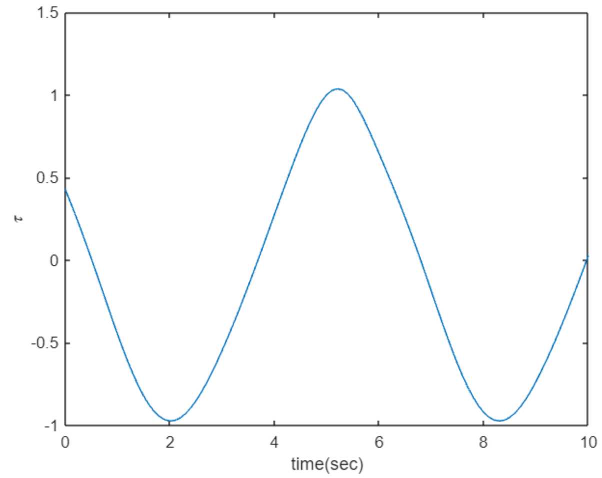
Simulation results

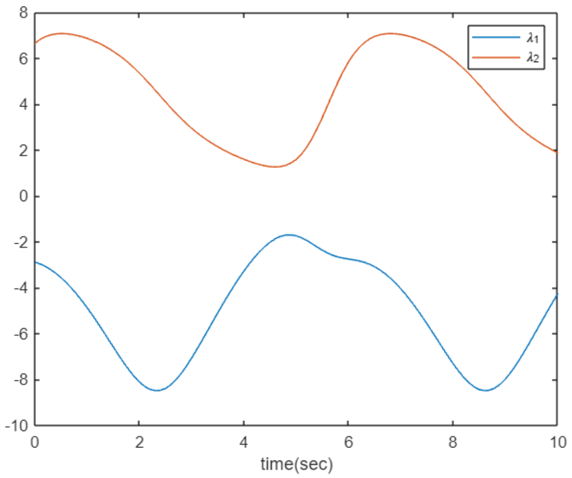
- For initial position of crank = 0 rad



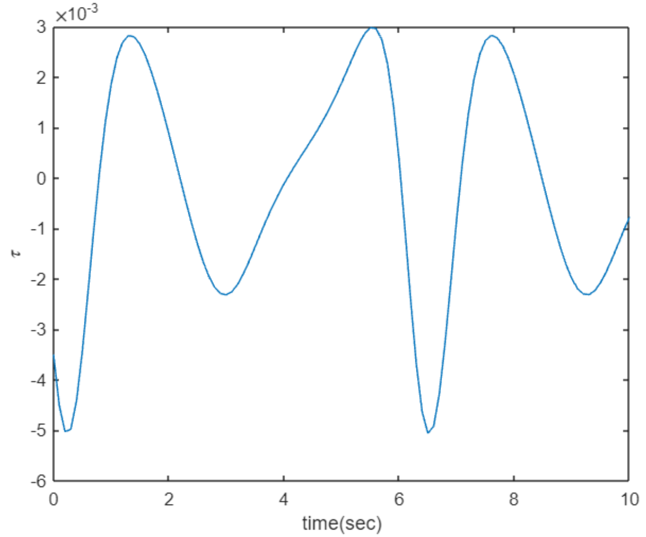


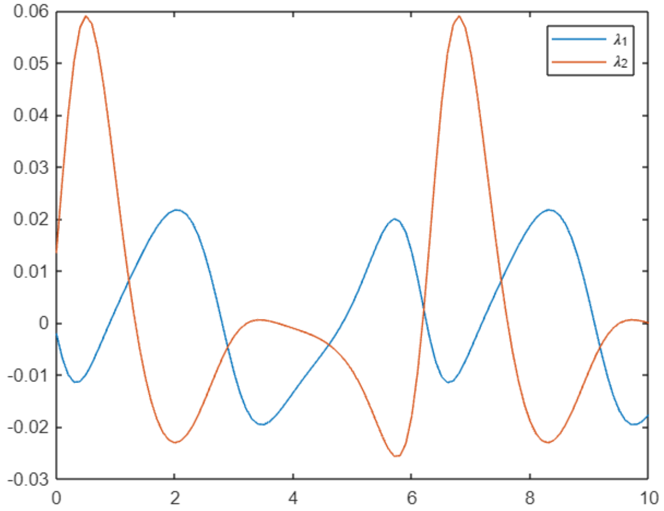
- For initial position of crank = pi/4 rad





- For initial position of crank = 0 rad without gravity





```
f_eqn =
```

$$
\left(\begin{array}{c}
\frac{3\,\mathrm{th}_{\mathrm{fdd}}}{2500} - \sigma_8 + \sigma_{14} + \sigma_{22}\,\sigma_4 + \sigma_7 - \sigma_2 - \sigma_5 + \sigma_1 + \sigma_{18}\,\sigma_3 + \sigma_{10} + \frac{3\cos(\mathrm{th}_f)\left(\frac{7\cos(\sigma_{25})\,\sigma_{12}}{100} - \sigma_{26} + \sigma_{27} - \frac{7\sin(\sigma_{25})\,\sigma_{11}}{100} + \frac{49}{5}\right)}{50} + \sigma_9 + \frac{3\sin(\mathrm{th}_f)\left(\frac{3\,\mathrm{th}_{\mathrm{fd}}^2\cos(\mathrm{th}_f)}{50} + \frac{7\sin(\sigma_{25})\,\sigma_{12}}{100} + \sigma_{24} + \frac{7\cos(\sigma_{25})\,\sigma_{11}}{100}\right)}{50} + \sigma_{13} - \sigma_6 \\
-\sigma_8 + \sigma_{14} + \frac{7\sin(\sigma_{31})\,\sigma_4}{50} + \sigma_7 - \sigma_2 - \sigma_5 + \sigma_1 + \frac{7\cos(\sigma_{31})\,\sigma_3}{50} + \sigma_{10} + \sigma_9 + \sigma_{13} - \sigma_6 \\
\frac{32\,\sigma_{33}}{1875\,\sigma_{49}} - \sigma_2 + \sigma_1 + \sigma_{10} + \sigma_9 + \frac{32\,\sigma_{48}\,\sigma_{47}}{1875\,\sigma_{49}^2}
\end{array}\right)
$$

Forward Dynamics Methodology and problem faced

- Unknown variables : $\theta_1, \theta_2, \theta_3, \theta_4, \lambda_1, \lambda_2$
- We can find $\theta_2, \theta_3, \theta_4$ after we find θ_1 using loop closure equations.
- $\theta_1, \lambda_1, \lambda_2$ can be determined by solving the 3 equations obtained from the DeNOC matrix.
- We need to find λ_1, λ_2 in terms of θ_1 and its derivatives.
- Finally, we must solve the second order differential equation of θ_1.

Forward Dynamics using Euler Lagrange formulation : Methodology

- $[\boldsymbol{M}(\mathbf{q})]\{\ddot{\mathbf{q}}\} + [\boldsymbol{C}(\mathbf{q},\dot{\mathbf{q}})]\{\dot{\mathbf{q}}\} + [\boldsymbol{G}(\mathbf{q})] = \boldsymbol{\tau} + (\boldsymbol{J}_{\eta q})^{\boldsymbol{T}}[\boldsymbol{\lambda}]$
- $\boldsymbol{M}(\mathbf{q}) = \sum \boldsymbol{J}_{viq}{}^{\boldsymbol{T}} m_i \boldsymbol{J}_{viq} + \sum \boldsymbol{J}_{\omega iq}{}^{\boldsymbol{T}} \boldsymbol{I}_i \boldsymbol{J}_{\omega iq}$ $\longrightarrow$

```
%Constructing matrices
syms a b c;
eqn_theta = subs(eqn_theta,[diff(theta_dot,t),diff(phi_dot,t),diff(psi_dot,t)],[a,b,c]);
eqn_phi = subs(eqn_phi,[diff(theta_dot,t),diff(phi_dot,t),diff(psi_dot,t)],[a,b,c]);
eqn_psi = subs(eqn_psi,[diff(theta_dot,t),diff(phi_dot,t),diff(psi_dot,t)],[a,b,c]);
eqn_theta = simplify(subs(eqn_theta,[diff(theta,t),diff(phi,t),diff(psi,t)],[dtht,dphi,dpsi]));
eqn_phi = simplify(subs(eqn_phi,[diff(theta,t),diff(phi,t),diff(psi,t)],[dtht,dphi,dpsi]));
eqn_psi = simplify(subs(eqn_psi,[diff(theta,t),diff(phi,t),diff(psi,t)],[dtht,dphi,dpsi]));

M1 = simplify(formula([(diff(eqn_theta,a)) (diff(eqn_theta,b)) (diff(eqn_theta,c))]));
M2 = simplify(formula([(diff(eqn_phi,a)) (diff(eqn_phi,b)) (diff(eqn_phi,c))]));
M3 = simplify(formula([(diff(eqn_psi,a)) (diff(eqn_psi,b)) (diff(eqn_psi,c))]));
M = [M1;M2;M3];
```

- $[C_{ij}] = 0.5 \times \sum_{k=1}^{n} \left(\frac{\partial M_{ij}}{\partial q_k} + \frac{\partial M_{ik}}{\partial q_j} - \frac{\partial M_{kj}}{\partial q_i} \right) \dot{q}_k$ $\longrightarrow$

```
C_matrix = zeros(3,3);
C_matrix = symmatrix(C_matrix);
C_matrix = symmatrix2sym(C_matrix);

for i=1:3
    for j=1:3
        for k=1:3
            temp = simplify(formula(0.5*((diff(M(i,j),q(k,1)) + ( ...
                diff(M(i,k),q(j,1))) - (diff(M(k,j),q(i,1))))*q_dot(k,1)));
            C_matrix(i,j) = C_matrix(i,j) + temp;
        end
    end
end
```

- $[G_i] = \frac{\partial V}{\partial q_i}$ $\longrightarrow$

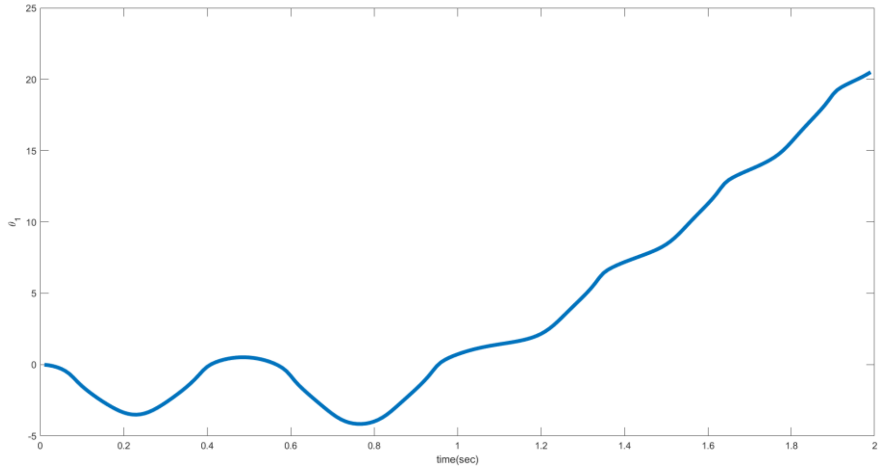
```
G_matrix = zeros(3,1);
G_matrix = symmatrix(G_matrix);
G_matrix = symmatrix2sym(G_matrix);
PE = PE1 + PE2 + PE3;

for i=1:3
    G_matrix(i,1) = diff(PE,q(i,1));
end
```

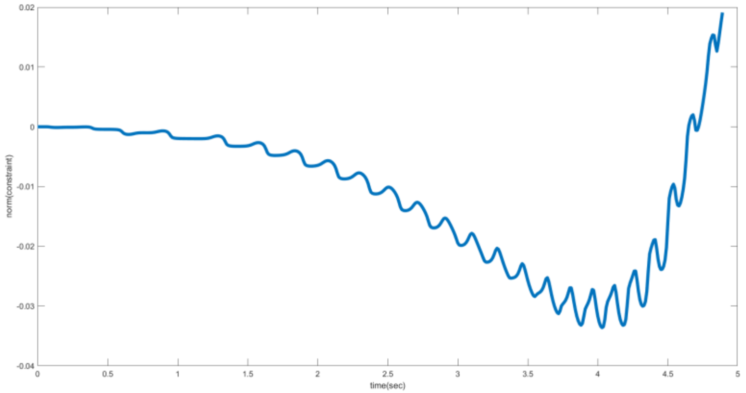
- $\boldsymbol{J}_{\eta q}$ $\longrightarrow$

```
psi_Matrix = formula([diff(cons1,q(1,1)) diff(cons1,q(2,1)) diff(cons1,q(3,1)); ...
    diff(cons2,q(1,1)) diff(cons2,q(2,1)) diff(cons2,q(3,1))]);
psi_Matrix_dot = formula(diff(psi_Matrix,t));
```

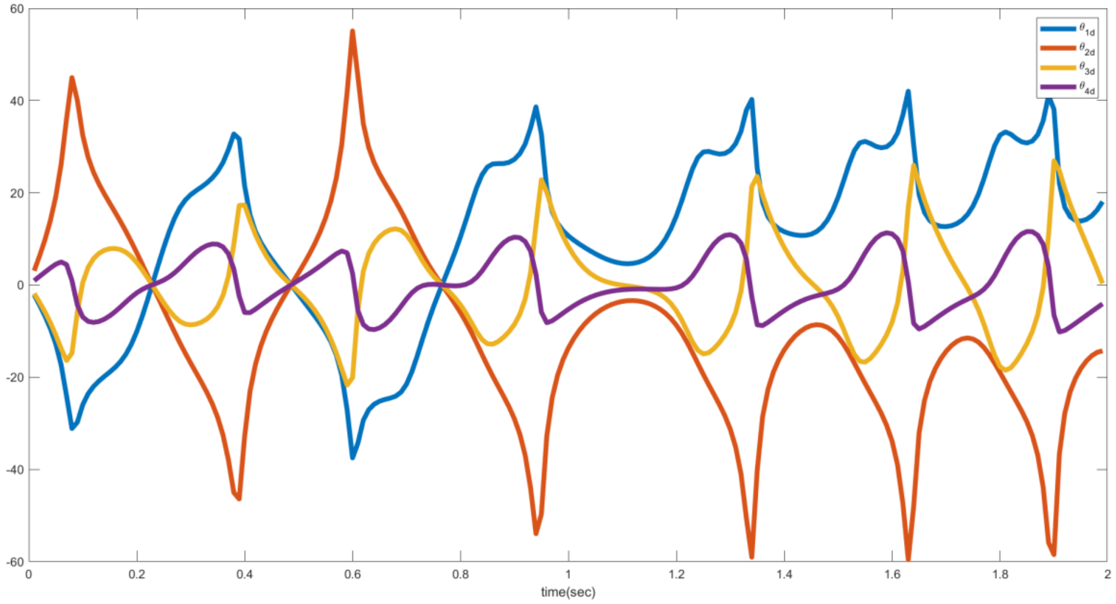
$$\boxed{\boldsymbol{\lambda} = -\left(\boldsymbol{J}_{\eta q}\boldsymbol{M}^{-1}\boldsymbol{J}_{\eta q}{}^{\boldsymbol{T}}\right)^{-1}\left(\dot{\boldsymbol{J}}_{\eta q}\{\dot{\mathbf{q}}\} + \boldsymbol{J}_{\eta q}\boldsymbol{M}^{-1}(\boldsymbol{\tau} - \boldsymbol{C}\{\dot{\mathbf{q}}\} - \boldsymbol{G})\right)}$$

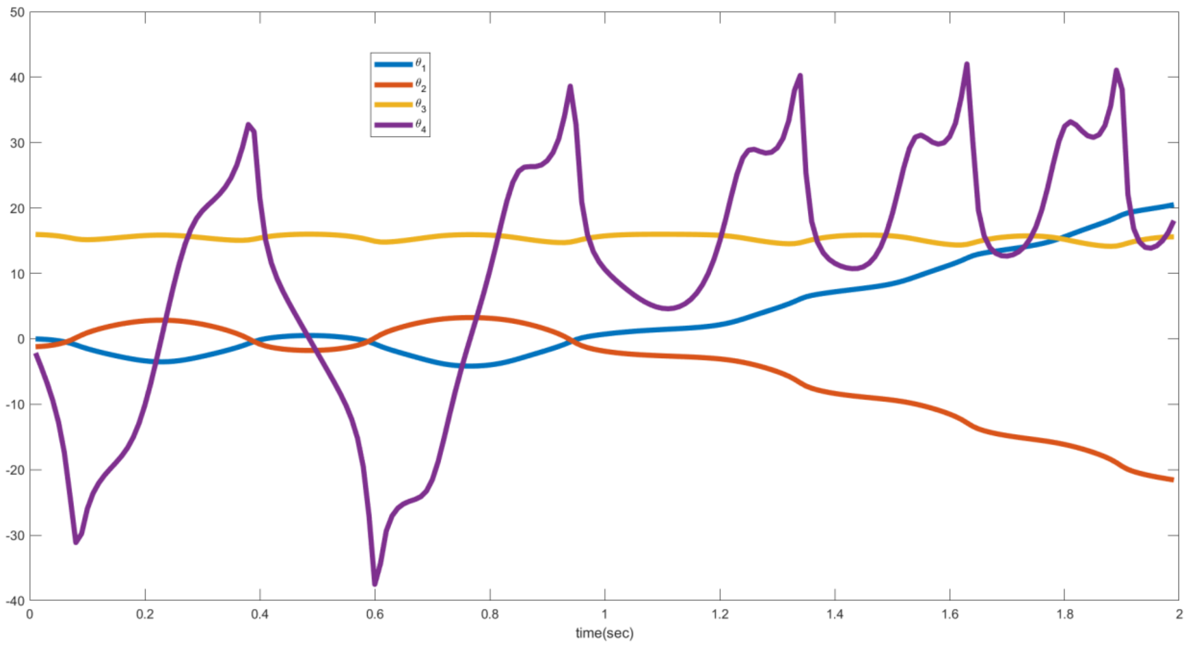
Forward Dynamics using Euler Lagrange formulation



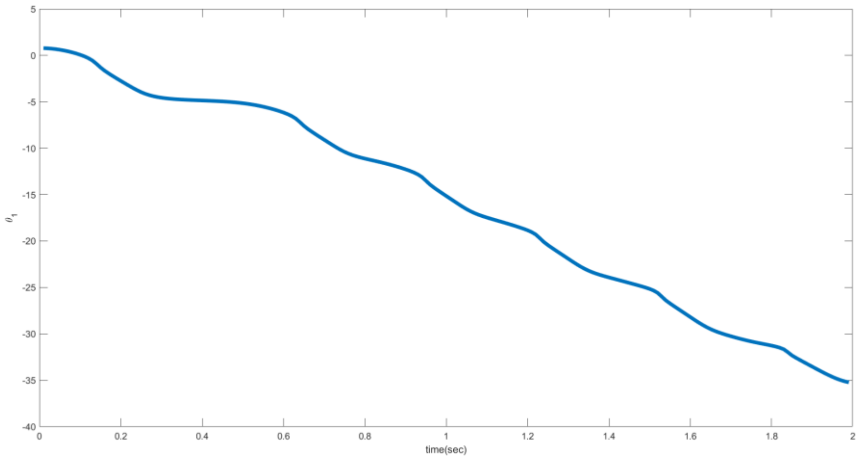


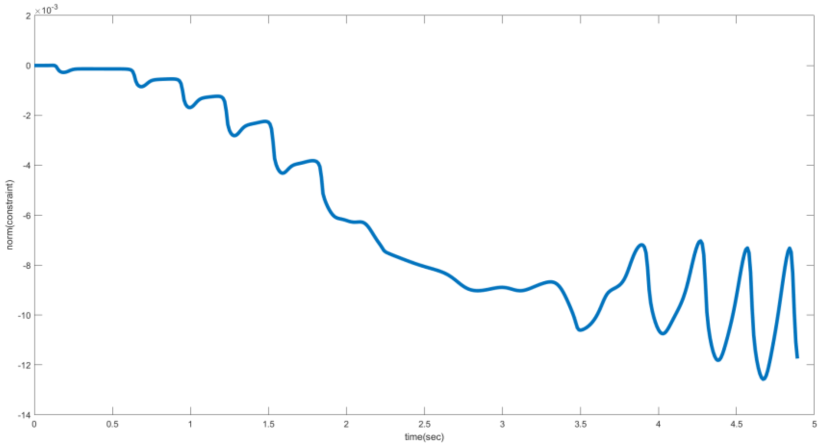
- When the Crank is horizontal :



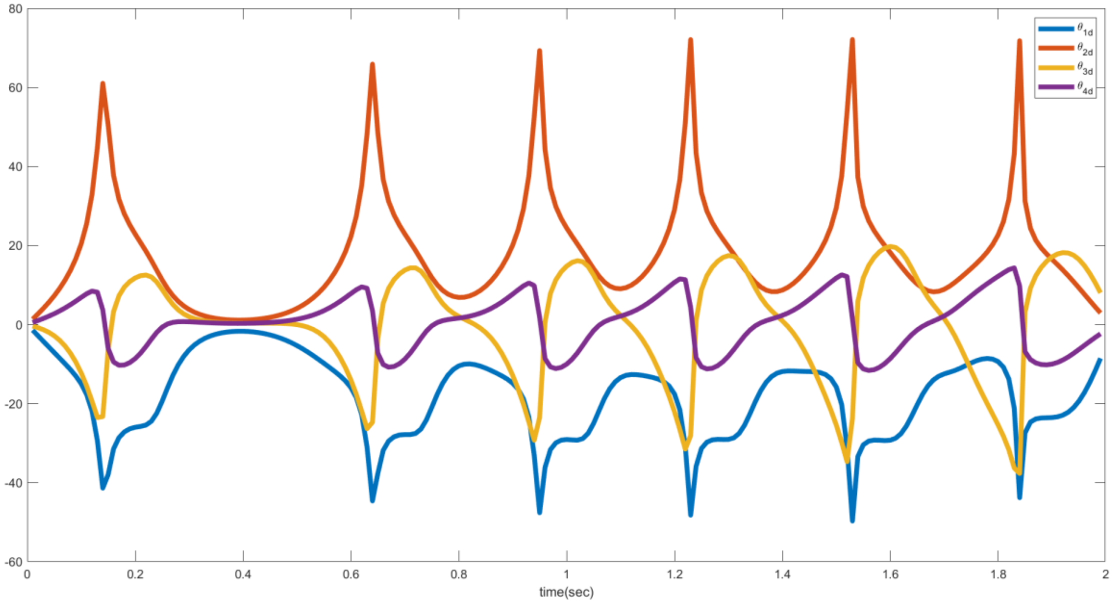


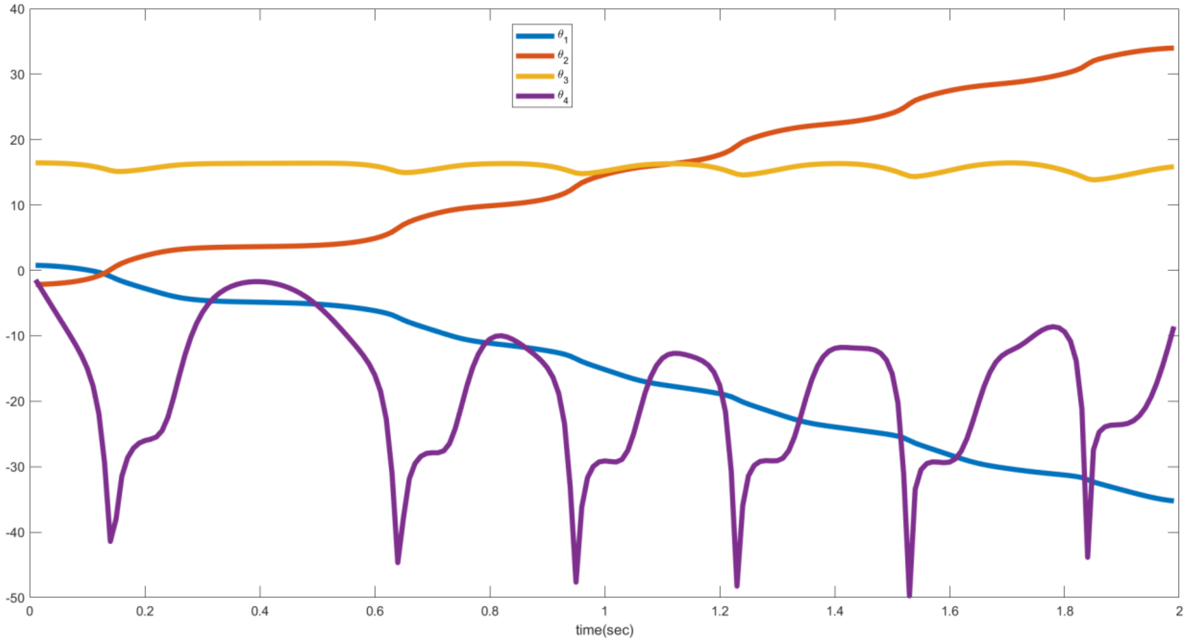
Forward Dynamics using Euler Lagrange formulation





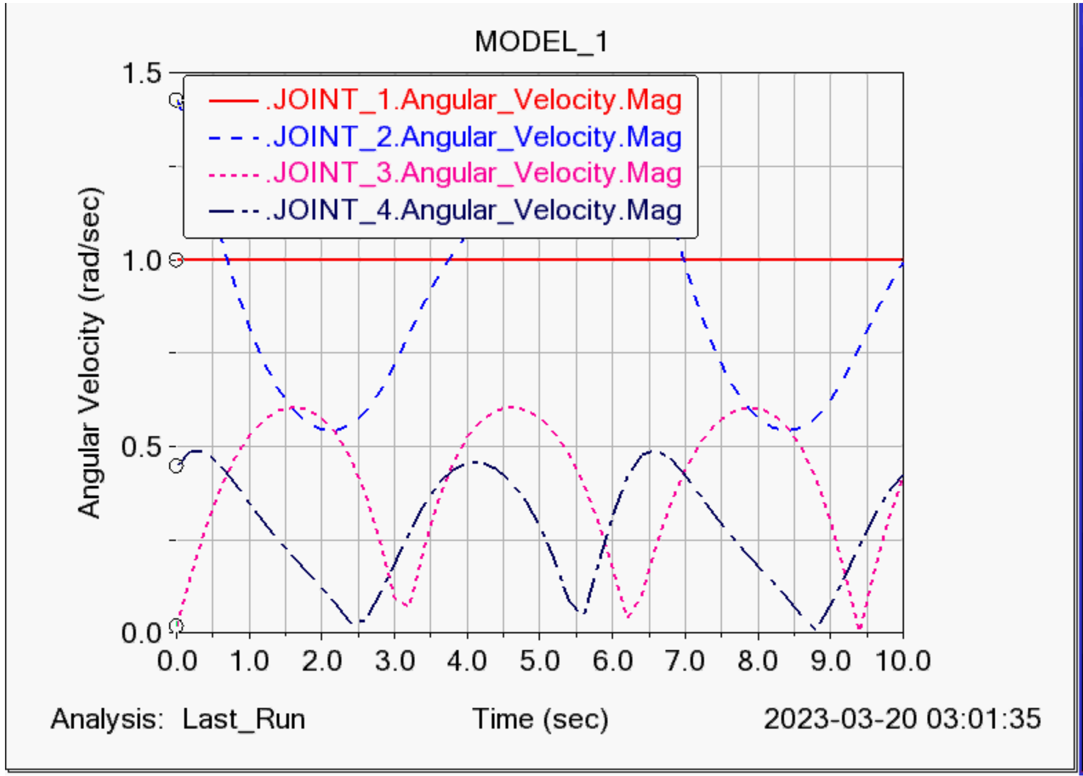
- When the Crank is at 45deg:

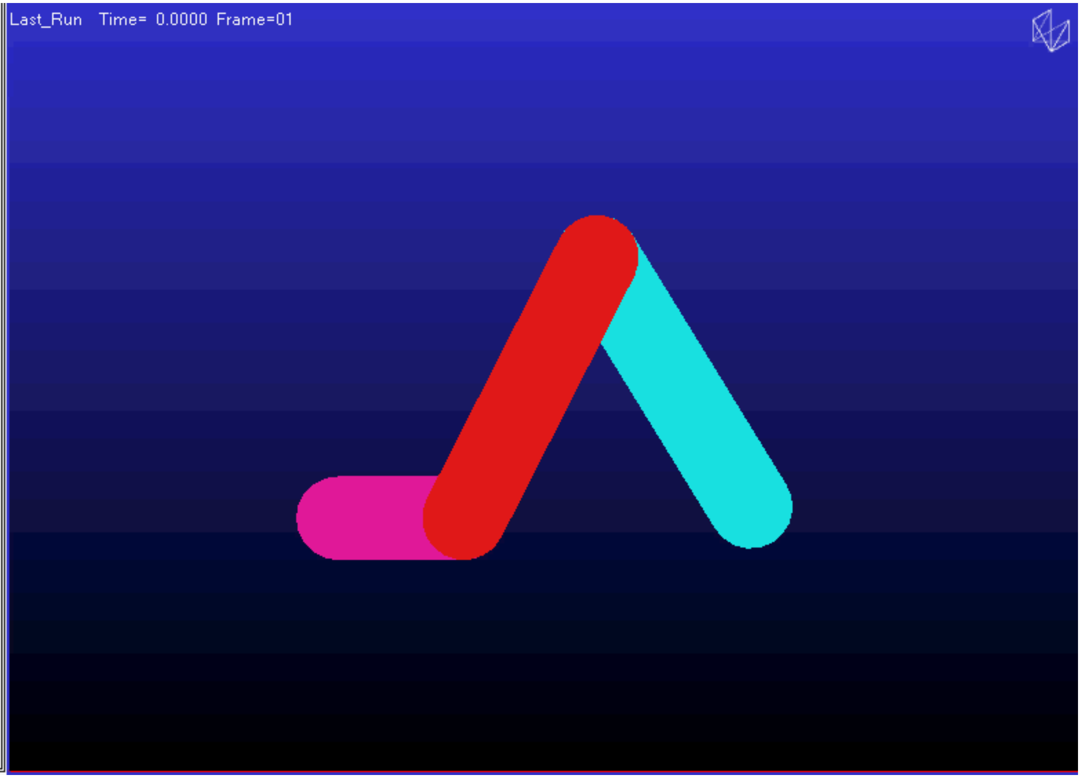




Four bar Using ADAMS View

Crank=0.06m,coupler=0.14m,output link=0.16,base=0.2m

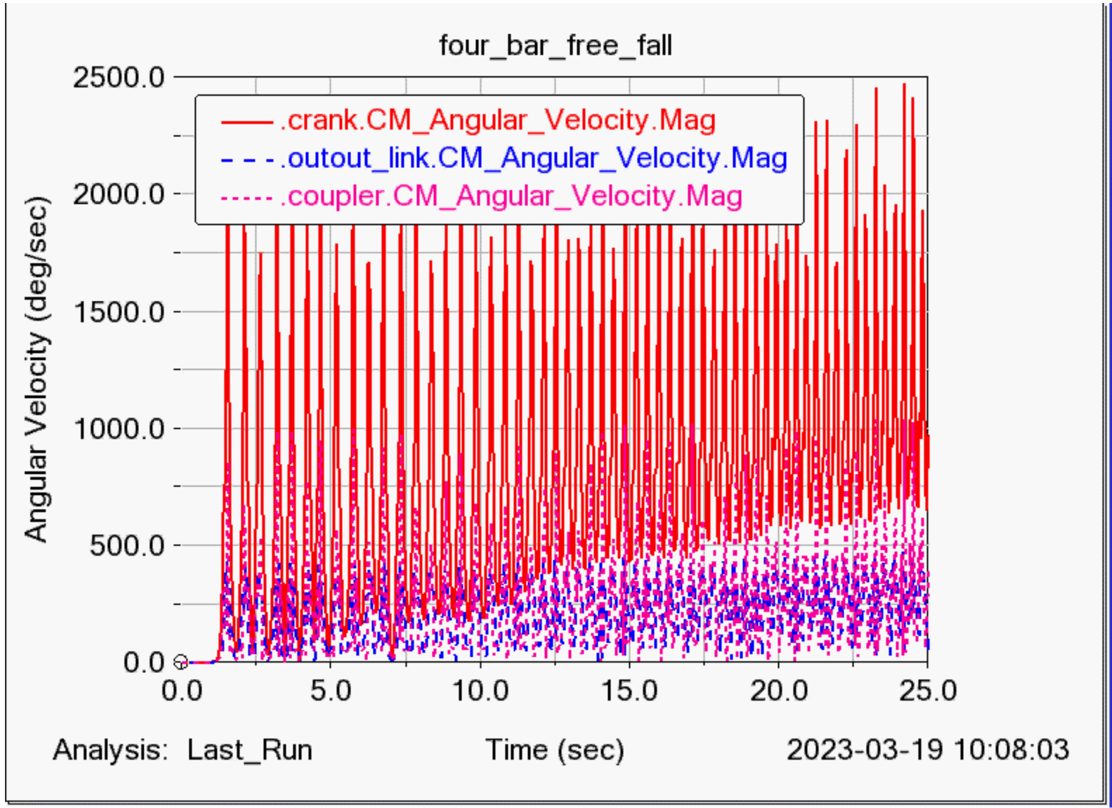


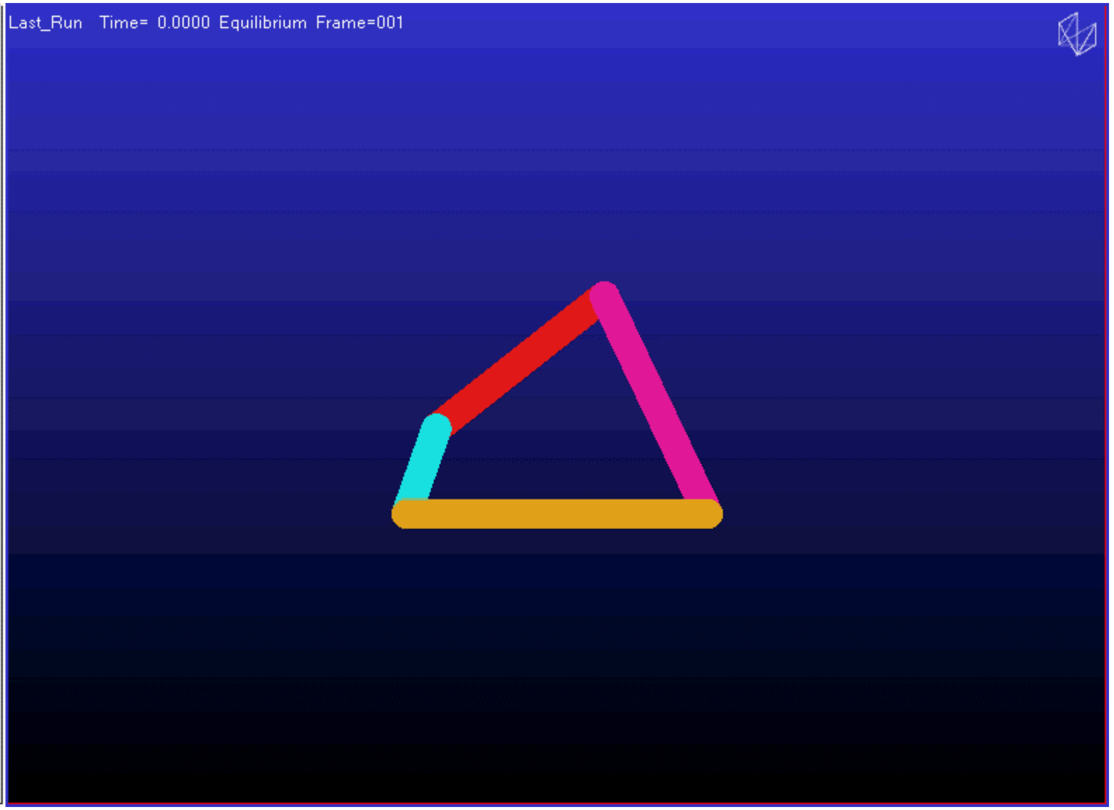


Vid.1:constant angular velocity at crank base (1 rad/sec)

Four bar Using ADAMS View

Crank=0.06m,coupler=0.14m,output link=0.16,base=0.2m





Vid.2: free fall under gravity without any torque at crank-base joint