Appendix B

MATLAB Codes for Kinematic and Dynamic Models of the Backhoe Excavator

This appendix presents the MATLAB codes for the complete mathematical kinematic model including direct kinematic model, inverse kinematic model, determination of bucket velocity, and piston velocities of hydraulic actuators, inverse Jacobian matrix determination, backhoe excavator static model (description is given in chapter 3). This appendix also presents the MATLAB code for the proposed dynamic model based on L-E formulation as described in chapter 4 of section 4.4 and for the dynamic model given by (A. J. Koivo et al., 1996) based on N-E formulation.

B.1 MATLAB code for the proposed mathematical kinematic model

In this section MATLAB codes for the complete kinematic model are presented. Firstly MATLAB code for the direct kinematics is presented, secondly MATLAB code for the inverse kinematic model is presented, thirdly MATLAB code for the determination of the bucket velocity is presented, fourthly MATLAB code for the determination of an inverse of a Jacobian matrix is presented, and finally MATLAB code for the backhoe excavator static model is presented.

B.1.1 MATLAB code for the direct kinematics

If the lengths of the piston rods in hydraulic actuators and the $\angle A_{12}A_{10}A_{11} = \zeta_3$ can be determined by the sensors, and the backhoe geometrical constants are known then the position of the bucket hinge point ($A_3$) and point ($A_4$) can be determined by the following MATLAB code.

```matlab
clear all
clc
%Enter the values of link lengths of the backhoe excavator
a1 = __;
a2 = __;
a3 = __;
a4 = __;
%Enter the values of geometry constants (distances and angles)
alpha = (__)*pi/180;
XS = __;
OX = __;
OS = __;
OT = __;
```
\[
\begin{align*}
\theta_1 &= \alpha - \pi - \arctan2(XS, OX) + \arctan2\left((4*(OS^2)*(OT^2)) - ((OS^2) + (OT^2) - (ST^2))^2\right)^{1/2}, ((OS^2) + (OT^2) - (ST^2)); \\
\theta_2 &= \pi - \gamma_1 - \gamma_2 - \arctan2\left((4*(AE^2)*(AF^2)) - ((AE^2) + (AF^2) - (EF^2))^2\right)^{1/2}, ((AE^2) + (AF^2) - (EF^2)); \\
\theta_3 &= 3\pi - \delta_1 - \delta_2 - \arctan2\left((4*(BG^2)*(BH^2)) - ((BG^2) + (BH^2) - (GH^2))^2\right)^{1/2}, ((BG^2) + (BH^2) - (GH^2)); \\
\zeta_1 &= 2\pi - \epsilon_1 - \arctan2\left((4*(IL^2)*(JL^2)) - ((IL^2) + (JL^2) - (IJ^2))^2\right)^{1/2}, ((IL^2) + (JL^2) - (IJ^2)); \\
\zeta_2 &= \arctan2\left((4*(CK^2)*(JK^2)) - ((CK^2) + (JK^2) - (CL^2) - (JL^2) + (2*CL*JL*cos(zeta_1))^2\right)^{1/2}, ((CK^2) + (JK^2) - (CL^2) - (JL^2) + (2*CL*JL*cos(zeta_1)))); \\
\theta_4 &= \zeta_1 + \zeta_2 + \pi - \eta_1 - \eta_2 + \zeta_3 \\
PA_3 &= [\cos(\theta_1)*(a1+a2*cos(\theta2)+a3*cos(\theta2+\theta3));\sin(\theta1)*(a1+a2*cos(\theta2)+a3*cos(\theta2+\theta3));a2*sin(\theta2)+a3*sin(\theta2+\theta3)]; \\
PA_4 &= [\cos(\theta1)*(a1+a2*cos(\theta2)+a3*cos(\theta2+\theta3)+a4*cos(\theta2+\theta4));\sin(\theta1)*(a1+a2*cos(\theta2)+a3*cos(\theta2+\theta3)+a4*cos(\theta2+\theta3+\theta4));a2*sin(\theta2)+a3*sin(\theta2+\theta3)+a4*sin(\theta2+\theta3+\theta4)]; \\
\end{align*}
\]

The blank spaces in the program require input data. Note that in all MATLAB codes presented the angles are required to be entered in degrees. The terms used in the program can be compared with the notations used in chapter 3 as: \(\alpha = \alpha, \gamma_1 = \gamma_1, \gamma_2 = \gamma_2, \delta_1 = \delta_1, \delta_2 = \delta_2, \epsilon_1 = \epsilon_1, \zeta_1 = \zeta_1, \zeta_2 = \zeta_2, \zeta_3 = \zeta_3, \zeta_4 = \zeta_4, AE = \text{distance} (A_1A_5), AF = \text{distance} (A_1A_6), EF = \text{distance} (A_5A_6), BG = \text{distance} (A_2A_7), BH = \text{distance} (A_2A_8), GH = \text{distance} (A_7A_8), IL = \text{distance} (A_9A_{12}), JL = \text{distance} (A_9A_{12})\).
= distance (A_{10}A_{12}), IJ = distance (A_{9}A_{10}), CL = distance (A_{3}A_{12}), CK = distance (A_{3}A_{11}), JK = distance (A_{10}A_{11}), \theta_1, \theta_2, \theta_3, \theta_4. This MATLAB code when run in the blank M file will return the coordinates of points pA3 = 0P_A, and pA4 = 0P_A, with respect to the base frame \{0\}.

**B.1.2 MATLAB code for the inverse kinematics**

If the link lengths, backhoe geometry constants, and the coordinates of point A_{3} in the base coordinate frame are known i.e. \( ^0P_A = [0x_P A, 0y_P A, 0z_P A, 1]^T \) then the lengths of piston rods of all actuators, and joint angles to make the bucket hinge point reach the required position and orientation can be determined from the following MATLAB code.

```matlab
clear all
clc

% Enter the values of link lengths of the backhoe excavator
a1 = __;
a2 = __;
a3 = __;
a4 = __;

% Enter the values of geometry constants (distances and angles)
alpha = (__)*pi/180;
XS = __;
OX = __;
OS = __;
OT = __;
XU = __;
OU = __;
OV = __;
gamma1 = (__)*pi/180;
gamma2 = (__)*pi/180;
delta1 = (__)*pi/180;
delta2 = (__)*pi/180;
epsilon1 = (__)*pi/180;
AE = __;
AF = __;
BG = __;
BH = __;
IL = __;
JL = __;
CL = __;
CK = __;
JK = __;
zeta3 = (__)*pi/180;
et1 = (__)*pi/180;
et2 = (__)*pi/180;

% Enter the bucket hinge points coordinates with respect to the base frame
xPA3 = __;
yPA3 = __;
zPA3 = __;

% Enter the bucket orientation angles while digging: rho, and lambda
rho = (__)*pi/180;
lambda = (__)*pi/180;

% Determination of the required joint angles and piston rod lengths to make the bucket hinge
```
The blank spaces in the program require inputs. The new terms used in the program can be compared with the notations used in chapter 3 as: \( \eta_1 \), \( \eta_2 \), \( \rho \), \( \lambda \), and \( \zeta_5 \). The descriptions of the other used terms are same as given in section B.1.1.

### B.1.3 MATLAB code for bucket velocity

Three linear and one angular bucket velocities can be determined from the following MATLAB code, if the link lengths, backhoe geometry constants, joint angles, and joint speeds are known.

```matlab
clear all
clc
%Enter the values of link lengths of the backhoe excavator
a1 = ___;
a2 = ___;
a3 = ___;
a4 = ___;
%Enter the values of geometry constants (distances and angles)
alpha = (___)*pi/180;
XS = ___;
OX = ___;
OS = ___;
OT = ___;
XU = ___;
OU = ___;
OV = ___;
gamma1 = (___)*pi/180;
gamma2 = (___)*pi/180;
delta1 = (___)*pi/180;
delta2 = (___)*pi/180;
epsilon1 = (___)*pi/180;
AE = ___;
AF = ___;
BG = ___;
```
BH = __;
IL = __;
JL = __;
CL = __;
CK = __;
JK = __;
zeta3 = (___)*pi/180;
etal1 = (___)*pi/180;
etal2 = (___)*pi/180;
%Enter joint angles and joint speeds
theta1 = __;
theta2 = __;
theta3 = __;
theta4 = __;
theta1d = __;
theta2d = __;
theta3d = __;
theta4d = __;
zeta3d = __;
%Determination of the required bucket velocity vector, and
%piston velocities of hydraulic actuators
J11 = - sin(theta1)*(a4*cos(theta2+theta3+theta4)+a3*cos(theta2+theta3)+a2*cos(theta2)+a1);
J12 = - cos(theta1)*(a4*sin(theta2+theta3+theta4)+a3*sin(theta2+theta3)+a2*sin(theta2));
J13 = -cos(theta1)*(a4*sin(theta2+theta3+theta4)+a3*sin(theta2+theta3)+a2*sin(theta2)+a1);
J14 = -cos(theta1)*a4*sin(theta2+theta3+theta4);
J21 = cos(theta1)*(a4*cos(theta2+theta3+theta4)+a3*cos(theta2+theta3)+a2*cos(theta2)+a1);
J22 = -sin(theta1)*(a4*sin(theta2+theta3+theta4)+a3*sin(theta2+theta3)+a2*sin(theta2));
J23 = -sin(theta1)*(a4*sin(theta2+theta3+theta4)+a3*sin(theta2+theta3)+a2*sin(theta2)+a1);
J24 = -sin(theta1)*a4*sin(theta2+theta3+theta4);
J31 = 0;
J32 = (a4*cos(theta2+theta3+theta4)+a3*cos(theta2+theta3)+a2*cos(theta2)+a1);
J33 = a4*cos(theta2+theta3+theta4)+a3*cos(theta2+theta3);
J34 = a4*cos(theta2+theta3+theta4);
J41 = 0;
J42 = 1;
J43 = 1;
J44 = 1;
J = [J11 J12 J13 J14; J21 J22 J23 J24; J31 J32 J33 J34; J41 J42 J43 J44];
Thetad = [theta1d; theta2d; theta3d; theta4d];
Vb = J*Thetad
Vst = ((OT*((OT*cos(alpha-theta1)+OX)*sin(alpha-theta1)+(-OT*sin(alpha-theta1)+XS)*cos(alpha-theta1)))/ST)*theta1d
Vuv = ((OV*((-OV*cos(alpha-theta1)+OX)*sin(alpha-theta1)+(OV*sin(alpha-theta1)-XU)*cos(alpha-theta1)))/UV)*theta1d
Vef = ((-AE*AP*sin(pi-gamma1-gamma2-theta2))/EF)*theta2d
zetab1 = 2*pi-epsilon1-atan2(((4*(IL^2)*(JL^2))-(IL^2)+(JL^2)-(IJ^2))^2^(1/2),((IL^2)+(JL^2)-(IJ^2)));
zetab2 = atan2((4*(CK^2)*(JK^2))-(CK^2)+(JK^2)-(CL^2)-(JL^2)+(2*CL*JL*cos(zeta1)))^2^(1/2),((CK^2)+(JK^2)-(CL^2)-(JL^2)+(2*CL*JL*cos(zeta1)));
Vgh = ((-BG*BH*sin(3*pi-delta1-delta2-theta3))/GH)*theta3d
zeta1d = ((theta4d-zeta3d)/(1+((CL*JL*sin(zeta1))/(CK*JK*sin(zeta2)))));
Vij = ((-IL*JL*sin(2*pi-epsilon1-zeta2))/IJ)*zeta1d

The blank spaces in the program require inputs. The new terms used in the program can be compared with the notations used in chapter 3 as: theta1d = \( \dot{\theta}_1 \), theta2d = \( \dot{\theta}_2 \), theta3d = \( \dot{\theta}_3 \), theta4d = \( \dot{\theta}_4 \), \( J_{ii} \) = the elements of the Jacobian matrix \( J \), \( \text{Thetad} \) = the vector of four joint speeds \( \mathbf{\dot{\theta}} \), \( V_b \) = \( V_b \), \( V_{st} \) = \( V_{ST} \), \( V_{uv} \) = \( V_{UV} \), \( V_{ef} \) = \( V_{A_5A_6} \), \( V_{gh} \) = \( V_{A_7A_8} \), and \( V_{ij} \) = \( V_{A_9A_{10}} \). The descriptions of the other used terms are same as given in earlier sections.

### B.1.4 MATLAB code for the determination of an inverse of the Jacobian

The MATLAB code to find the Jacobian inverse for given values of link lengths \( a_i \) for \( i = 1, 2, 3, 4 \) and joint angles \( \theta_i \) for \( i = 1, 2, 3, 4 \) can be given as follows:

```matlab
clear all
clc
% Enter the values of link lengths of the backhoe excavator
a1 = ___;
a2 = ___;
a3 = ___;
a4 = ___;
% Enter the values of geometry constants (distances and angles)
alpha =(__)*pi/180;
XS = ___;
OX = ___;
OS = ___;
OT = ___;
ST = ___;
XU = ___;
OU = ___;
OV = ___;
UV = ___;
gamma1 =(__)*pi/180;
gamma2 =(__)*pi/180;
delta1 =(__)*pi/180;
delta2 =(__)*pi/180;
epsilon1 =(__)*pi/180;
AE = ___;
AF = ___;
EF = ___;
BG = ___;
BH = ___;
GH = ___;
IL = ___;
JL = ___;
IJ = ___;
CL = ___;
CK = ___;
JK = ___;
zeta3 =(__)*pi/180;
eta1 =(__)*pi/180;
eta2 =(__)*pi/180;
zeta3d = ___;
% Enter the values of the joint angles
theta1 = ___;
```

237
\theta_2 = \_\_;
\theta_3 = \_\_;
\theta_4 = \_\_;

%Enter the bucket velocity vector \( \mathbf{V}_b \)
\( \mathbf{V}_b = [\_\_; \_\_; \_\_; \_\_]; \)

%Determination of the required joint speed vector \( \mathbf{\theta}_d \) to achieve the required bucket velocity \( \mathbf{V}_b \)
\( J_{11} = -\sin(\theta_1) \times (a_4 \cos(\theta_2 + \theta_3 + \theta_4) + a_3 \cos(\theta_2 + \theta_3) + a_2 \cos(\theta_2) + a_1); \)
\( J_{12} = -\cos(\theta_1) \times (a_4 \sin(\theta_2 + \theta_3 + \theta_4) + a_3 \sin(\theta_2 + \theta_3) + a_2 \sin(\theta_2)); \)
\( J_{13} = -\cos(\theta_1) \times (a_4 \sin(\theta_2 + \theta_3 + \theta_4) + a_3 \sin(\theta_2 + \theta_3)); \)
\( J_{14} = -\cos(\theta_1) \times a_4 \sin(\theta_2 + \theta_3 + \theta_4); \)
\( J_{21} = \cos(\theta_1) \times (a_4 \cos(\theta_2 + \theta_3 + \theta_4) + a_3 \cos(\theta_2 + \theta_3) + a_2 \cos(\theta_2) + a_1); \)
\( J_{22} = -\sin(\theta_1) \times (a_4 \sin(\theta_2 + \theta_3 + \theta_4) + a_3 \sin(\theta_2 + \theta_3) + a_2 \sin(\theta_2)); \)
\( J_{23} = -\sin(\theta_1) \times (a_4 \sin(\theta_2 + \theta_3 + \theta_4) + a_3 \sin(\theta_2 + \theta_3)); \)
\( J_{24} = -\sin(\theta_1) \times a_4 \sin(\theta_2 + \theta_3 + \theta_4); \)
\( J_{31} = 0; \)
\( J_{32} = (a_4 \cos(\theta_2 + \theta_3 + \theta_4) + a_3 \cos(\theta_2 + \theta_3) + a_2 \cos(\theta_2) + a_1); \)
\( J_{33} = a_4 \cos(\theta_2 + \theta_3 + \theta_4) + a_3 \cos(\theta_2 + \theta_3); \)
\( J_{34} = a_4 \cos(\theta_2 + \theta_3 + \theta_4); \)
\( J_{41} = 0; \)
\( J_{42} = 1; \)
\( J_{43} = 1; \)
\( J_{44} = 1; \)
\( J = [J_{11} \ J_{12} \ J_{13} \ J_{14}; J_{21} \ J_{22} \ J_{23} \ J_{24}; J_{31} \ J_{32} \ J_{33} \ J_{34}; J_{41} \ J_{42} \ J_{43} \ J_{44}]; \)
\( \det(J); \)
\( J_1 = \text{inv}(J); \)
\( \mathbf{\theta}_d = \text{inv}(J) \times \mathbf{V}_b \)
\( \theta_1d = \_\_; \)
\( \theta_2d = \_\_; \)
\( \theta_3d = \_\_; \)
\( \theta_4d = \_\_; \)

%Determination of the required piston velocities to achieve the desired bucket velocity
\( \mathbf{V}_{st} = ((\mathbf{O}_T \times ((\mathbf{O}_T \times \cos(\alpha - \theta_1) + \mathbf{O}_X) \times \sin(\alpha - \theta_1)) + (\mathbf{O}_T \times \sin(\alpha - \theta_1) + \mathbf{X}_S) \times \cos(\alpha - \theta_1)))) / \mathbf{S}_T \times \theta_1d \)
\( \mathbf{V}_{uv} = ((\mathbf{O}_V \times ((\mathbf{O}_V \times \cos(\alpha + \theta_1) + \mathbf{O}_X) \times \sin(\alpha + \theta_1)) + (\mathbf{O}_V \times \sin(\alpha + \theta_1) - \mathbf{X}_U) \times \cos(\alpha + \theta_1)))) / \mathbf{U}_V \times \theta_1d \)
\( \mathbf{V}_{ef} = ((-\mathbf{A}_E \times \mathbf{A}_F \times \sin(\pi - \gamma_1 - \gamma_2 - \theta_2)) / \mathbf{E}_F \times \theta_2d \)
\( \mathbf{V}_{gh} = ((-\mathbf{B}_G \times \mathbf{B}_H \times \sin(3 \pi - \delta_1 - \delta_2 - \theta_3)) / \mathbf{G}_H \times \theta_3d \)
\( \mathbf{V}_{ij} = ((\mathbf{I}_L \times \mathbf{J}_L \times \sin(2 \pi - \epsilon_1 - zeta_2)) / (1 + ((\mathbf{C}_L \times \mathbf{J}_L \times \sin(zeta_1)) / (\mathbf{C}_K \times \mathbf{J}_K \times \sin(zeta_2)))); \)

Note that this MATLAB code when written in the blank M file of MATLAB will require these inputs: four link lengths \( a_i \) for \( i = 1, 2, 3, \) and \( 4, \) and four joint angles \( \theta_i \) for \( i \)
= 1, 2, 3, and 4. Once the values of link lengths are fixed with other geometry constants and joint angles then required joint speeds can be found, and from that the required piston velocities for the desired bucket velocity can be determined. If \( \det(J) \) shows the zero value then \( \text{inv}(J) \) returns error, this means that for that particular set of joint angles the inverse of the Jacobian matrix does not exist and thus the required piston velocities cannot be determined (i.e. reverse mapping of the bucket velocity is not possible). The descriptions of the terms used in this MATLAB code are same as given in earlier sections.

### B.1.5 MATLAB code for backhoe excavator static model

If the link lengths, joint angles, and the resulting resistive forces, and moments due to the ground-bucket interaction in the static equilibrium condition (as explained in section 3.5 of chapter 3) are known then the resulting joint torques due to the ground-bucket interaction can be determined from the following MATLAB code.

```matlab
clear all
clc

% Enter the values of link lengths of the backhoe excavator
a1 = __;
a2 = __;
a3 = __;
a4 = __;

% Enter the values of the joint angles
theta1 = __*pi/180;
theta2 = __*pi/180;
theta3 = __*pi/180;
theta4 = __*pi/180;

% Enter the values of the resolved resistive forces and moments
Fx = __;
Fy = __;
Fz = __;
Nx = __;
Ny = __;
Nz = __;

% Determination of the resulting joint torques due to ground-bucket interaction in static equilibrium condition

J11 = -sin(theta1)*(a4*cos(theta2+theta3+theta4)+a3*cos(theta2+theta3)+a2*cos(theta2)+a1);
J12 = -cos(theta1)*(a4*sin(theta2+theta3+theta4)+a3*sin(theta2+theta3)+a2*sin(theta2));
J13 = -cos(theta1)*(a4*sin(theta2+theta3+theta4)+a3*sin(theta2+theta3));
J14 = -cos(theta1)*a4*sin(theta2+theta3+theta4);
J21 = -cos(theta1)*(a4*cos(theta2+theta3+theta4)+a3*cos(theta2+theta3)+a2*cos(theta2)+a1);
J22 = -sin(theta1)*(a4*sin(theta2+theta3+theta4)+a3*sin(theta2+theta3)+a2*sin(theta2));
J23 = -sin(theta1)*(a4*sin(theta2+theta3+theta4)+a3*sin(theta2+theta3));
J24 = -sin(theta1)*a4*sin(theta2+theta3+theta4);
```

239
\[ J_{31} = 0; \]
\[ J_{32} = (a4 \cos(\theta_2 + \theta_3 + \theta_4) + a3 \cos(\theta_2 + \theta_3) + a2 \cos(\theta_2) + a1); \]
\[ J_{33} = a4 \cos(\theta_2 + \theta_3 + \theta_4) + a3 \cos(\theta_2 + \theta_3); \]
\[ J_{34} = a4 \cos(\theta_2 + \theta_3 + \theta_4); \]
\[ J_{41} = 0; \]
\[ J_{42} = 1; \]
\[ J_{43} = 1; \]
\[ J_{44} = 1; \]
\[ J = [J_{11} \ J_{12} \ J_{13} \ 0 \ 0 \ J_{41}; \ J_{12} \ J_{13} \ J_{31} \ J_{32} \ J_41 \ J_42; \ J_{13} \ J_{33} \ J_{11} \ J_{12} \ J_43 \ J_42; \]
\[ J_{14} \ J_{24} \ J_{34} \ J_{22} \ J_{32} \ J_43 \ J_44]; \]
\[ F = [F_x; F_y; F_z; N_x; N_y; N_z]; \]
\[ T = J \times F \]

The blank spaces in the program require inputs. The new terms used in the program can be compared with the notations used in chapter 3 as: \( F_x, F_y, F_z \) = the components of the resistive force resolved in X, Y and Z directions of the bucket frame respectively, \( N_x, N_y, N_z \) = the components of the moments at the bucket teeth due to the resistive forces about X, Y, and Z directions of the bucket frame respectively. \( F \) = the vector of resistive force and moments, \( T \) = the vector of resulting joint torques. The descriptions of the other used terms are same as given in earlier sections. The static joint torque can be found by considering bucket cylinder active and arm cylinder active. In the developed MATLAB code, only made changes in the value of \( N_z \) when bucket cylinder and arm cylinder active respectively. It provides two different values of static joint torques for both the cases.

**B.2 MATLAB codes for the proposed L-E dynamic model and A. J. Koivo et al. dynamic model**

To determine the joint torques, the MATLAB code is now presented for the proposed L-E dynamic model and N-E dynamic model (A. J. Koivo et al., 1996).

- The MATLAB code for the proposed L-E dynamic model is given as:

```
clear all
clc
% Enter the moment of inertia for boom link (Izz2), arm link (Izz3), bucket link (Izz4) in Kg*m^2(2) with respect to coordinate zi axis of reference frame (2),(3), and (4) respectively;
Izz2 = __;
Izz3 = __;
Izz4 = __;
% Enter the mass of boom (m2), arm (m3), and bucket (m4) in Kg;
m2 = __;
m3 = __;
m4 = __;
% Enter the center of mass (centroid) vector of body from body's coordinate frame origin: r2 for boom, r3 for arm in m, and r4 for bucket;
```
\[x_2 = \_\_;\]
\[y_2 = \_\_;\]
\[z_2 = \_\_;\]
\[x_3 = \_\_;\]
\[y_3 = \_\_;\]
\[z_3 = \_\_;\]
\[x_4 = \_\_;\]
\[y_4 = \_\_;\]
\[z_4 = \_\_;\]
\[r_2 = [x_2; y_2; z_2; 1];\]
\[r_3 = [x_3; y_3; z_3; 1];\]
\[r_4 = [x_4; y_4; z_4; 1];\]

% Enter the link lengths for boom, arm and bucket as \(a_2\), \(a_3\), and \(a_4\) in mm respectively;
\[a_2 = \_\_;\]
\[a_3 = \_\_;\]
\[a_4 = \_\_;\]

% Enter the joint displacement, joint velocity, and joint acceleration profiles as a function of time;
\[\theta_2 = (\_\_)*\pi/180;\]
\[\theta_3 = (\_\_)*\pi/180;\]
\[\theta_4 = (\_\_)*\pi/180;\]
\[\theta_2d = \_\_;\]
\[\theta_3d = \_\_;\]
\[\theta_4d = \_\_;\]
\[\theta_2dd = \_\_;\]
\[\theta_3dd = \_\_;\]
\[\theta_4dd = \_\_;\]

% Enter the value of acceleration due to gravity of earth \(g\);
\[g = 9.81;\]

% Enter the required values for load vector constants tangential and resistive

% Enter the two components of the resistive force: tangential force \(F_t\), normal resistive force \(F_n\), and angles: \(\rho\), and \(\lambda\);
\[F_t = \_\_;\]
\[F_n = \_\_;\]
\[\rho = (\_\_)*\pi/180;\]
\[\lambda = (\_\_)*\pi/180;\]

% Determination of the elements of the inertia matrix;
\[M_{22} = I_{zz2} + I_{zz3} + I_{zz4} + (2*m_2*x_2*a_2) + (m_2*((a_2)^2)) + (m_3*x_3*(2*a_3+2*a_2*cos(\theta_3))) -
(2*m_3*y_3*a_2*sin(\theta_3)) + (m_3*(((a_2)^2+(a_3)^2+2*a_2*a_3*cos(\theta_3))) + (m_4*x_4 * (2*a_4+2*a_2*cos(\theta_3+a_4)) + 2*a_3*cos(\theta_4))) + (m_4*y_4*(2*a_2*sin(\theta_3+a_4)+2*a_3*sin(\theta_4)))) + m_4*((a_2)^2+(a_3)^2+(2*a_4) + 2*a_3*cos(\theta_3+a_4)) + 2*a_2*a_3*cos(\theta_4) + 2*a_3*cos(\theta_4)+2*a_2*a_3*cos(\theta_3+a_4)) + 2*a_2*a_3*cos(\theta_3+a_4)) + 2*a_2*a_3*cos(\theta_3+a_4)) + 2*a_2*a_3*cos(\theta_3+a_4)) + 2*a_2*a_3*cos(\theta_3+a_4));\]
\[M_{23} = I_{zz3} + I_{zz4} + m_3*x_3*(2*a_3+a_2*cos(\theta_3)) -
m_3*y_3*a_2*sin(\theta_3) + m_3*(a_3)^2 + m_4*x_4*(2*a_4+2*a_3*cos(\theta_3)+2*a_3*cos(\theta_4))*a_2*cos(\theta_3+a_4));\]
\[M_{24} = I_{zz4} + m_4*x_4*(2*a_4+a_3*cos(\theta_4)+a_2*cos(\theta_3+a_4)) -
m_4*y_4*(a_3*sin(\theta_4)+a_2*sin(\theta_3+a_4)) + m_4*((a_3)^2+(a_4)^2+2*a_3*a_4*cos(\theta_4)+a_2*a_3*cos(\theta_3+a_4))) + m_4*(a_3)^2+2*a_3*a_4*cos(\theta_4)+a_2*a_3*cos(\theta_3+a_4));\]
\[M_{32} = M_{23};\]
\[M_{33} = I_{zz3} + I_{zz4} + 2*m_3*x_3*a_3 + m_3*(a_3)^2 + 2*m_4*x_4*(2*a_4+2*a_3*cos(\theta_4)) -
m_4*y_4*(2*a_3*sin(\theta_4)+a_2*sin(\theta_3+a_4))+m_4*((a_3)^2+(a_4)^2+2*a_3*a_4*cos(\theta_4)+a_2*a_3*cos(\theta_3+a_4)));\]
\[M_{34} = I_{zz4} + m_4*x_4*(2*a_3+a_3*cos(\theta_4)) -
m_4*y_4*(2*a_3*sin(\theta_4)+a_2*sin(\theta_3+a_4))+m_4*((a_3)^2+2*a_3*a_4*cos(\theta_4));\]
\[M_{42} = M_{24};\]
\[M_{43} = M_{34};\]

241
M44 = Izz4 + 2*m4*x4*a4+m4*(a4)^2;

% Determination of the elements of the velocity induced torque matrix;
H22 = 0*theta2d-
( m3^2*x3*a2*sin(theta3)+m3*y3*a2*cos(theta3)+m3*a2*a3*sin(theta3)+m4*x4*a2
* sin(theta3+theta4)+m4*y4*a2*cos(theta3+theta4)+m4*(a2*a3*sin(theta3)+a2
* a4*sin(theta3+theta4))) * theta2d-
( m4*x4*(a2*sin(theta3+theta4)+a3*sin(theta4))+m4*y4*(a2*cos(theta3+theta4)
+a3*cos(theta4))+m4*(a3*a4*sin(theta4)+a2*a4*sin(theta3+theta4))) * theta4d;
H23 = -
( m3*x3*a2*sin(theta3)+m3*y3*a2*cos(theta3)+m3*a2*a3*sin(theta3)+m4*x4*a2
* sin(theta3+theta4)+m4*y4*a2*cos(theta3+theta4)+m4*(a2*a3*sin(theta3)+a2
* a4*sin(theta3+theta4))) * theta2d-
( m3*x3*a2*sin(theta3)+m3*y3*a2*cos(theta3)+m3*a2*a3*sin(theta3)+m4*x4*a2
* sin(theta3+theta4)+m4*y4*a2*cos(theta3+theta4)+m4*(a2*a3*sin(theta3)+a2
* a4*sin(theta3+theta4))) * theta2d-
( m4*x4*a3*sin(theta4)+m4*y4*a3*cos(theta4)+m4*a3*a4*sin(theta4)) * theta4d;
H24 = -
( m4*x4*(a2*sin(theta3+theta4)+a3*sin(theta4))+m4*y4*(a2*cos(theta3+theta4)
+a3*cos(theta4)+m4*(a3*a4*sin(theta4)+a2*a4*sin(theta3+theta4))) * theta2d-
( m4*x4*(a2*sin(theta3+theta4)+a3*sin(theta4))+m4*y4*(a2*cos(theta3+theta4)
+a3*cos(theta4)+m4*(a3*a4*sin(theta4)+a2*a4*sin(theta3+theta4))) * theta3d-
( m4*x4*a3*sin(theta4)+m4*y4*a3*cos(theta4)+m4*a3*a4*sin(theta4)) * theta4d;
H32 =
( m3*x3*a2*sin(theta3)+m3*y3*a2*cos(theta3)+m3*a2*a3*sin(theta3)+m4*x4*a2
* sin(theta3+theta4)+m4*y4*a2*cos(theta3+theta4)+m4*(a2*a3*sin(theta3)+a2
* a4*sin(theta3+theta4))) * theta2d-
( m3*x3*a2*sin(theta3)+m3*y3*a2*cos(theta3)+m3*a2*a3*sin(theta3)+m4*x4*a2
* sin(theta3+theta4)+m4*y4*a2*cos(theta3+theta4)+m4*(a2*a3*sin(theta3)+a2
* a4*sin(theta3+theta4))) * theta3d-
( m4*x4*a3*sin(theta4)+m4*y4*a3*cos(theta4)+m4*a3*a4*sin(theta4)) * theta4d;
H33 = 0*theta2d+0*theta3d-
( m4*x4*a3*sin(theta4)+m4*y4*a3*cos(theta4)+m4*a3*a4*sin(theta4)) * theta4d;
H34 = -
( m4*x4*a3*sin(theta4)+m4*y4*a3*cos(theta4)+m4*a3*a4*sin(theta4)) * theta2d-
( m4*x4*a3*sin(theta4)+m4*y4*a3*cos(theta4)+m4*a3*a4*sin(theta4)) * theta3d-
( m4*x4*a3*sin(theta4)+m4*y4*a3*cos(theta4)+m4*a3*a4*sin(theta4)) * theta4d;
H42 =
( m4*x4*(a2*sin(theta3+theta4)+a3*sin(theta4))+m4*y4*(a2*cos(theta3+theta4)
+a3*cos(theta4)+m4*(a3*a4*sin(theta4)+a2*a4*sin(theta3+theta4))) * theta2d-
( m4*x4*(a2*sin(theta3+theta4)+a3*sin(theta4))+m4*y4*(a2*cos(theta3+theta4)
+a3*cos(theta4)+m4*(a3*a4*sin(theta4)+a2*a4*sin(theta3+theta4))) * theta3d-
( m4*x4*a3*sin(theta4)+m4*y4*a3*cos(theta4)+m4*a3*a4*sin(theta4)) * theta4d;
H43 =
( m4*x4*a3*sin(theta4)+m4*y4*a3*cos(theta4)+m4*a3*a4*sin(theta4)) * theta2d-
( m4*x4*a3*sin(theta4)+m4*y4*a3*cos(theta4)+m4*a3*a4*sin(theta4)) * theta3d-
( m4*x4*a3*sin(theta4)+m4*y4*a3*cos(theta4)+m4*a3*a4*sin(theta4)) * theta4d;
H44 = 0*theta2d+0*theta3d+0*theta4d;

% Determination of the elements of the gravity loading vector;
G2 = -m2*g*(x2*cos(theta2)-y2*sin(theta2)+a2*cos(theta2))-
( m3*g*(x3*cos(theta2+theta3)-
y3*sin(theta2+theta3)+a3*cos(theta2+theta3)+a2*cos(theta2))-
( m4*g*(x4*cos(theta2+theta3+theta4)-
( m4*x4*a3*sin(theta4)+m4*y4*a3*cos(theta4)+m4*a3*a4*sin(theta4)) * theta4d-
( m4*x4*a3*sin(theta4)+m4*y4*a3*cos(theta4)+m4*a3*a4*sin(theta4)) * theta4d);
\[ y_4 \sin(\theta_2 + \theta_3 + \theta_4) + a_4 \cos(\theta_2 + \theta_3 + \theta_4) + a_3 \cos(\theta_2 + \theta_3) + a_2 \cos(\theta_2); \]
\[ G_3 = -m_3 g (x_3 \cos(\theta_2 + \theta_3) - y_3 \sin(\theta_2 + \theta_3) + a_3 \cos(\theta_2 + \theta_3)); \]
\[ y_4 \sin(\theta_2 + \theta_3 + \theta_4) + a_4 \cos(\theta_2 + \theta_3 + \theta_4) + a_3 \cos(\theta_2 + \theta_3 + \theta_4); \]
\[ G_4 = -m_4 g (x_4 \cos(\theta_2 + \theta_3 + \theta_4) - y_4 \sin(\theta_2 + \theta_3 + \theta_4) + a_4 \cos(\theta_2 + \theta_3 + \theta_4)); \]

% Determination of the elements of the Load vector;
\[ F_2 = a_2 (F_t \sin(\theta_2 - \rho) - F_n \cos(\theta_2 - \rho)); \]
\[ F_3 = a_3 (F_t \sin(\theta_2 + \theta_3 - \rho) - F_n \cos(\theta_2 + \theta_3 - \rho)); \]
\[ F_4 = a_4 (-F_t \sin(\lambda) + F_n \cos(\lambda)); \]

% Determination of joint torques by multiplication of matrices;
\[ M_{ij} = [M_{22} \ M_{23} \ M_{24}; M_{32} \ M_{33} \ M_{34}; M_{42} \ M_{43} \ M_{44}]; \]
\[ H_{ij} = [H_{22} \ H_{23} \ H_{24}; H_{32} \ H_{33} \ H_{34}; H_{42} \ H_{43} \ H_{44}]; \]
\[ G_i = [G_2; G_3; G_4]; \]
\[ F_{load} = [F_2; F_3; F_4]; \]
\[ \Theta_{add} = [\theta_2d; \theta_3d; \theta_4d]; \]
\[ \Theta_{add} = [\theta_2dd; \theta_3dd; \theta_4dd]; \]
\[ M_{ij} \Theta_{add} + H_{ij} \Theta_{add} + G_i + F_{load} \]

The blank spaces in the programs suggest the inputs, when inputs are given to the program, it will return the values of joint torques. Now the MATLAB code for the N-E dynamic model (A. J. Koivo et al., 1996) is presented as follows:

- The MATLAB code for Koivo’s N-E dynamic model is given as:

```matlab
clear all
clc
% Enter the moment of inertia for boom link (Izz2), arm link (Izz3),
% bucket link (Izz4) in Kg*m^2(2) with respect to coordinate zi axis of
% reference frame {2},{3}, and {4} respectively;
Izz2 = ___;
Izz3 = ___;
Izz4 = ___;
% Enter the mass of boom (m2), arm (m3), and bucket (m4) in Kg;
m2 = ___;
m3 = ___;
m4 = ___;
% Enter the distances of center of mass (centroid) from their respective
% origins of the frames: x2 (from centroid of boom to origin of frame 2),
% x3, and x4 in m.
x2 = ___;
x3 = ___;
x4 = ___;
sigma1 = (___)*pi/180;
sigma2 = (___)*pi/180;
sigma3 = (___)*pi/180;
% Enter the link lengths for boom, arm and bucket as a2, a3, and a4 in m
% respectively;
a2 = ___;
a3 = ___;
a4 = ___;
```
% Enter the joint displacement, joint velocity, and joint acceleration
profiles as a function of time;
theta2 = (__)*pi/180;
theta3 = (__)*pi/180;
theta4 = (__)*pi/180;
theta2d = __;
theta3d = __;
theta4d = __;
theta2dd = __;
theta3dd = __;
theta4dd = __;
% Enter the value of acceleration due to gravity of earth g;
g = 9.81;
% Enter the required values for load vector constants tangential
% resistive
% Enter the two components of the resistive force: tangential force Ft, normal resistive force Fn, and angles: rho, and lambda;
Fr = __;
Ft = Fr*cos(0.1);
Fn = -Fr*sin(0.1);
rho = (__)*pi/180;
lambda = (__)*pi/180;
% Determination of the elements of the inertia matrix;
M22 = Izz2+Izz3+Izz4+m2*(x2)^2+m3*(x3)^2+m4*(x4)^2+2*m3*a2*x3*cos(theta3+sigma2)+2*m4*a2*a3*cos(theta3)+2*m4*a3*x4*cos(theta4+sigma1)*theta4d+2*m4*a2*x4*cos(theta3+theta4+sigma1);
M23 = Izz3+Izz4+m3*(x3)^2+m4*(a4)^2+m4*a3*x3*cos(theta3+sigma2)+2*m4*a2*a3*cos(theta3)+2*m4*a3*x4*cos(theta4+sigma1)+2*m4*a2*x4*cos(theta3+theta4+sigma1);
M32 = M23;
M24 = Izz4+m4*(x4)^2+2*m4*a3*x4*cos(theta4+sigma1)+m4*a2*x4*cos(theta3+theta4+sigma1);
M42 = M24;
M33 = Izz3+Izz4+m3*(x3)^2+2*m4*(a3)^2+2*m4*a3*x4*cos(theta4+sigma1)+m4*a3*x4*cos(theta4+sigma1);
M34 = Izz4+m4*(x4)^2+m4*a3*x4*cos(theta4+sigma1);
M43 = M34;
M44 = Izz4+m4*(x4)^2;
% Determination of the elements of the velocity induced torque matrix;
H22 = 0*theta2d-
2*(m3*a2*x3*sin(theta3+sigma2)+m4*a2*a3*sin(theta3)+m4*a2*x4*sin(theta3+theta4+sigma1))*theta2d-
2*(m4*a2*x4*sin(theta4+sigma1))*theta4d;
H23 = 0*theta2d-
(m3*a2*x3*sin(theta3+sigma2)+m4*a2*a3*sin(theta3)+m4*a2*x4*sin(theta3+theta4+sigma1))*theta2d-
2*(m4*a2*x4*sin(theta4+sigma1))*theta4d;
H24 = 0*theta2d+0*theta3d-
(m4*a2*x4*sin(theta3+theta4+sigma1)+m4*a3*x4*sin(theta4+sigma1))*theta4d;
H32 = (m3*a2*x3*sin(theta3+sigma2)+m4*a2*a3*sin(theta3)+m4*a2*x4*sin(theta3+theta4+sigma1))*theta2d+0*theta3d-
(m4*a3*x4*sin(theta4+sigma1))*theta3d;
H33 = 0*theta2d+0*theta3d-
(m4*a3*x4*sin(theta4+sigma1))/2;
H42 = 
(m4*a2*x4*sin(theta3+theta4+sigma1)+m4*a3*x4*sin(theta4+sigma1))*theta2d 
-(m4*a3*x4*sin(theta4+sigma1))*theta3d+0*theta4d;
H43 = 
(m4*a3*x4*sin(theta4+sigma1))*theta2d+(m4*a3*x4*sin(theta4+sigma1))*theta3d+0*theta4d;
H44 = 0*theta2d+0*theta3d+0*theta4d;

% Determination of the elements of the gravity loading vector;
G2 = -
m4*g*(a2*cos(theta2)+a3*cos(theta2+theta3)+x4*cos(theta2+theta3+theta4+sigma1))-
m3*g*(a2*cos(theta2)+x3*cos(theta2+theta3+sigma2))-
m2*g*x2*cos(theta2+sigma3);
G3 = -m4*g*(a3*cos(theta2+theta3)+x4*cos(theta2+theta3+theta4+sigma1))-
m3*x3*cos(theta2+theta3+sigma2);
G4 = -m4*g*x4*cos(theta2+theta3+theta4+sigma1);

% Determination of the Load vector;
F2 = a2*(Ft*sin(theta2-rho)-Fn*cos(theta2-rho));
F3 = a3*(Ft*sin(theta2+theta3-rho)-Fn*cos(theta2+theta3-rho));
F4 = a4*(-Ft*sin(lambda)+Fn*cos(lambda));

% Determination of joint torques by multiplication of matrices;
Mij = [M22 M23 M24; M32 M33 M34; M42 M43 M44];
Hij = [H22 H23 H24; H32 H33 H34; H42 H43 H44];
Gi = [G2; G3; G4];
Fload = [F2; F3; F4];
Thetad = [theta2d; theta3d; theta4d];
Thetadd = [theta2dd; theta3dd; theta4dd];
Mij*Thetadd;
Hij*Thetad;
Gi;
Fload;
Ti = Mij*Thetadd+Hij*Thetad+Gi+Fload

All the required variables or inputs should be in proper units. For this program all moments of inertias $l_{zz}$ should be in (Kg $\cdot$ m$^2$), mass of the links $m_i$ in (Kg), the $x_i$, $y_i$, and $z_i$ describing the centre of mass of the body $i$ should me in (m), link lengths $a_i$ should be in (m), joint angles $\theta_i$ (in program these are written as theta2, etc) should be in (rad), joint velocities $\dot{\theta}_i$ (in program these are written as theta2d, etc.) should be in (rad/sec), joint accelerations $\ddot{\theta}_i$ (in program these are written as theta2dd, etc.) should be in (rad/sec$^2$), the other forces should be in (N).

Note that for N-E dynamic model (A. J. Koivo et al., 1996), the values of $x_i$ for $i = 2$, 3, and 4 in program indicates the distance of the centre of mass of boom, arm, and bucket from the origin of frames {2}, {3}, and {4}.

B.3 MATLAB code for determination of the joint displacements, speeds and accelerations

It should be noted that for the determination of the joint torques from the dynamic model presented in section 4.4 of chapter 4, the joint displacements, joint velocities, and joint
accelerations must be known, and these are the functions of time. This means the joint displacement, velocity, and acceleration profiles should be predefined for the determination of the joint torques.

For determination of these joint displacement, velocities and accelerations the cubic polynomial trajectory (point to point motion without via points) for the joint displacements has been assumed as follows:

\[ \theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \]  \hspace{1cm} \ldots (B.1)

The derivation of the equation (B.1) with respect time leads to the parabolic velocity profile as:

\[ \dot{\theta}(t) = a_1 + 2a_2 t + 3a_3 t^2 \]  \hspace{1cm} \ldots (B.2)

The derivation of equation (B.2) with respect to time gives the linear acceleration profile as:

\[ \ddot{\theta}(t) = 2a_2 + 6a_3 t \]  \hspace{1cm} \ldots (B.3)

Let \( \theta^s \), and \( \theta^g \) be the start and goal point values for the joint variables respectively, and \( t_s \) and \( t_g \) be the start and the goal time (time to reach the goal) respectively. For the smooth motion between initial and goal points, let us define two constraints each on the joint position and velocity functions as:

\[ \theta(t = 0) = \theta^s \]
\[ \theta(t = t_g) = \theta^g \]
\[ \dot{\theta}(t = 0) = 0 \]
\[ \dot{\theta}(t = t_g) = 0 \]

This means at time \( t = 0 = t_s \), the joint displacement \( \theta = \theta^s \) and joint velocity \( \dot{\theta} = 0 \). At the goal time \( t = t_g \), the joint displacement \( \theta = \theta^g \) and joint velocity \( \dot{\theta} = 0 \). Applying these constraints into equations (B.1), (B.2), and (B.3) leads to the following set of four equations for four unknowns:

\[ a_0 = \theta^s \]
\[ a_1 = 0 \]
\[ a_2 = \frac{3}{t_g^2} (\theta^g - \theta^s) \]
\[ a_3 = -\frac{2}{t_g^3} (\theta^g - \theta^s) \]
Application of these four constants \(a_1, a_2, a_3, a_4\) in equations (B.1), (B.2), and (B.3) will determine the joint displacement, velocity and acceleration as a function of time only.

For the determination of joint velocities and accelerations for different set of joint displacements with user defined constraints the following MATLAB code should be applied.

```matlab
clear all
clc
qs = (__)*pi/180;
a0 = qs;
ts = 0;
tg = __;
a1 = 0;
qg = (__)*pi/180;
a2 = 3*(qg-a0)/(tg)^2;
a3 = -2*(qg-a0)/(tg)^3;
t = __;
qi = a0+a1*t+a2*t^2+a3*t^3;
qid = a1+2*a2*t+3*a3*t^2;
qidd = 2*a2+6*a3*t;
```

Where, \(qs\) represents the starting value of the joint angle \(\theta_s\), and \(qg\) represents the final or goal value of the joint angle \(\theta_g\) in terms of radians, \(ts\) represents the time to reach the goal \(ts\), \(tg\) represents the time to reach the goal \(tg\). \(q\), \(qd\), \(qdd\) represent the joint displacement \(\theta\), joint speed \(\dot{\theta}\) and the joint acceleration \(\ddot{\theta}\) and \(i\) represents the link number. Using the above MATLAB code the joint displacement, velocity and acceleration can be finding out by changing the inputs for the boom, arm, bucket and intermediate links one by one.

Joint accelerations also find out using the following MATLAB code based on kinematics of the backhoe excavator:

```matlab
clear all
clc
%Joint acceleration vector thetadd can be determined from the following
%equations
AE = __;
AF = __;
EF = __;
BG = __;
BH = __;
GH = __;
IL = __;
JL = __;
IJ = __;
CL = __;
CK = __;
JK = __;
zeta3 = (__)*pi/180;
```
zeta3d = ___;
zeta3dd = ___;
et1 = (__) * pi/180;
et2 = (__) * pi/180;
theta1 = 0;
gamma1 = (__) * pi/180;
gamma2 = (__) * pi/180;
delta1 = (__) * pi/180;
delta2 = (__) * pi/180;
epsilon1 = (__) * pi/180;
Vef = ___;
Vgh = ___;
Vij = ___;
Aef = ___;
Agh = ___;
Aij = ___;
theta2 = pi - gamma1 - gamma2 - atan2(((4*(AE^2)*(AF^2)) - ((AE^2)+(AF^2) - (EF^2))^2)^(1/2),((AE^2)+(AF^2) - (EF^2)))/2);  
theta3 = 3*pi - delta1 - delta2 - atan2(((4*(BG^2)*(BH^2)) - ((BG^2)+(BH^2) - (GH^2))^2)^(1/2),((BG^2)+(BH^2) - (GH^2)))/2);  
zeta1 = 2*pi - epsilon1 - atan2(((4*(IL^2)*(JL^2)) - ((IL^2)+(JL^2) - (IJ^2))^2)^(1/2),((IL^2)+(JL^2) - (IJ^2)))/2);  
zeta2 = atan2(((4*(CK^2)*(JK^2)) - ((CK^2)+(JK^2) - (CL^2) - (JL^2)+(2*CL*JL*cos(zeta1)))/2)^2,((CK^2)+(JK^2) - (CL^2) - (JL^2)+(2*CL*JL*cos(zeta1)))/2);  
theta4 = zeta1 + zeta2 + pi - etal1 - etal2 + zeta3d;  
theta2d = (EF*Vef)/(AE*EF*sin(pi - gamma1 - gamma2 - theta2));  
theta3d = (GH*Vgh)/(BG*BH*sin((3*pi) - delta1 - delta2 - theta3));  
zeta1d = -(IJ*Vij)/(IL*JL*sin((2*pi) - epsilon1 - zeta1));  
theta4d = zeta1d*1 + ((CL*JL*sin(zeta1))/CK*JK*sin(zeta2)) + zeta3d;  
theta2dd = (EF*cos(pi-gamma1-gamma2-theta2) * theta2d * Vef - sin(pi-gamma1-gamma2-theta2) * Aef)/(AE*AF*sin(pi-gamma1-gamma2-theta2))^2);  
theta3dd = (GH*cos((3*pi) - delta1 - delta2 - theta3) * theta3d * Vgh - sin((3*pi) - delta1 - delta2 - theta3) * Agh)/(BG*BH*sin((3*pi) - delta1 - delta2 - theta3));  
zeta2d = (zeta1d*CL*JL*sin(zeta1))/CK*JK*sin(zeta2);  
zeta1dd = -(IJ*cos((2*pi) - epsilon1 - zeta1)*zeta1d*Vij - IJ*sin((2*pi) - epsilon1 - zeta1)*Aij)/(IL*JL*sin((2*pi) - epsilon1 - zeta1));  
theta4dd = zeta1d*1 + ((CL*JL*sin(zeta2)*cos(zeta1)*zeta1d*Vij - IJ*JL*sin((2*pi) - epsilon1 - zeta1)*Aij)/(IL*JL*sin((2*pi) - epsilon1 - zeta1)));  
theta2ddd = (EF*cos(pi-gamma1-gamma2-theta2) * theta2d * Vef - sin(pi-gamma1-gamma2-theta2) * Aef)/(AE*AF*sin(pi-gamma1-gamma2-theta2))^3;  
theta3ddd = (GH*cos((3*pi) - delta1 - delta2 - theta3) * theta3d * Vgh - sin((3*pi) - delta1 - delta2 - theta3) * Agh)/(BG*BH*sin((3*pi) - delta1 - delta2 - theta3));  
theta4ddd = zeta1d*1 + ((CL*JL*sin(zeta2)*cos(zeta1)*zeta1d*Vij - IJ*JL*sin((2*pi) - epsilon1 - zeta1)*Aij)/(IL*JL*sin((2*pi) - epsilon1 - zeta1)));  
zeta2dd = (zeta1d*CL*JL*sin(zeta1))/CK*JK*sin(zeta2);  
zeta1ddd = -(IJ*cos((2*pi) - epsilon1 - zeta1)*zeta1d*Vij - IJ*sin((2*pi) - epsilon1 - zeta1)*Aij)/(IL*JL*sin((2*pi) - epsilon1 - zeta1));  
theta4ddd = zeta1d*1 + ((CL*JL*sin(zeta2)*cos(zeta1)*zeta1d*Vij - IJ*JL*sin((2*pi) - epsilon1 - zeta1)*Aij)/(IL*JL*sin((2*pi) - epsilon1 - zeta1)));  
zeta2dd = (zeta1d*CL*JL*sin(zeta1))/CK*JK*sin(zeta2);  
zeta1ddd = -(IJ*cos((2*pi) - epsilon1 - zeta1)*zeta1d*Vij - IJ*sin((2*pi) - epsilon1 - zeta1)*Aij)/(IL*JL*sin((2*pi) - epsilon1 - zeta1));  
theta4ddd = zeta1d*1 + ((CL*JL*sin(zeta2)*cos(zeta1)*zeta1d*Vij - IJ*JL*sin((2*pi) - epsilon1 - zeta1)*Aij)/(IL*JL*sin((2*pi) - epsilon1 - zeta1)));  

B.4 MATLAB code for determination of the link accelerations

Link accelerations can be found out using the following MATLAB code based on kinematics of the backhoe excavator:

clear all
clc

%Link acceleration vector thetaadd can be determined from the following equations
AE = ___;
AF = ___;
EF = ___;
BG = ___;
BH = ___;
GH = ___;
IL = ___;

248
\[ \text{zeta3} = (\_\_*)\pi/180; \]
\[ \text{zeta3d} = \_\_; \]
\[ \text{zeta3dd} = \_\_; \]
\[ \text{eta1} = (\_\_*)\pi/180; \]
\[ \text{eta2} = (\_\_*)\pi/180; \]
\[ \text{theta1} = 0; \]
\[ \text{gamma1} = (\_\_*)\pi/180; \]
\[ \text{gamma2} = (\_\_*)\pi/180; \]
\[ \text{delta1} = (\_\_*)\pi/180; \]
\[ \text{delta2} = (\_\_*)\pi/180; \]
\[ \text{epsilon1} = (\_\_*)\pi/180; \]
\[ \text{theta2} = \_\_; \]
\[ \text{theta2d} = \_\_; \]
\[ \text{theta2dd} = \_\_; \]
\[ \text{theta3} = \_\_; \]
\[ \text{theta3d} = \_\_; \]
\[ \text{theta3dd} = \_\_; \]
\[ \text{Vef} = \_\_; \]
\[ \text{Vgh} = \_\_; \]
\[ \text{zeta1} = (\_\_*)\pi/180; \]
\[ \text{zeta1d} = \_\_; \]
\[ \text{Vij} = \_\_; \]
\[ \text{theta4dd} = \_\_; \]
\[ \text{zeta2} = (\_\_*)\pi/180; \]
\[ \text{zeta3d} = \_\_; \]
\[ \text{zeta2d} = \text{zeta1d}^2 \text{((CL*JL* \sin(zeta1)))/(CK*JK* \sin(zeta2))}; \]
\[ \text{Aef} = \text{(EF*cos(pi-gamma1-gamma2-theta2)*theta2d*Vef-(AE*AF*(sin(pi-gamma1-gamma2-theta2))}^2)\text{theta2dd}/(\text{sin(pi-gamma1-gamma2-theta2)*EF}) \]
\[ \text{Agh} = \text{(GH*cos((3*pi)-delta1-delta2-theta3)*theta3d*Vgh-} \text{(BG*BH*theta3dd*(sin((3*pi)-delta1-delta2-theta3)}^2)}/(\text{GH*sin((3*pi)-delta1-delta2-theta3}) \]
\[ \text{zeta1dd} = \text{(theta4dd-zeta1d*(CL*JL* \sin(zeta2)*cos(zeta1)*zeta1d)}- \text{(cos(zeta1)*sin(zeta2)*zeta2d))/}(\text{CK*JK* \sin(zeta2)}^2)\text{zeta3d}/(\text{(CL*JL* \sin(zeta1)))/(CK*JK* \sin(zeta2)}) \]
\[ \text{Aij} = \text{((IL*JL* \sin(2*pi-epsilon1-zeta1)}^2)\text{zeta1dd}+(\text{IJ*cos(2*pi-epsilon1-zeta1)}*zeta1d*Vij))/\text{(IJ* \sin(2*pi-epsilon1-zeta1))} \]

**B.5 MATLAB code for validation of direct kinematic model**

The following MATLAB code is used for validation of direct kinematic model of backhoe excavator attachment.

```matlab
% Validation of direct kinematic model
% Joint angle variation within -180 to +180 degree is as below
% First joint angle theta2 = ___ to ___;
% Second joint angle theta3 = ___ to ___;
% Third joint angle theta4 = ___ to ___;

clear all
clc
% Enter the values of link length of backhoe parts
a2 = ___;
```
% Enter the initial position of each link of backhoe attachment
theta2 = linspace(__, __, __)
theta3 = __
theta4 = __

% Validation for angle theta2
p = theta2;
q = theta2+theta3;
r = theta2+theta3+theta4;
XE1 = a2*cos(pi*p/180)+a3*cos(pi*q/180)+a4*cos(pi*r/180)+430
ZE2 = (a2*sin(pi*p/180)+a3*sin(pi*q/180)+a4*sin(pi*r/180)+500)

% Validation for angle theta3
theta3 = linspace(__, __, __)
theta4 = __
theta2 = __
p = theta2;
q = theta2+theta3;
r = theta2+theta3+theta4;
XC1 = a2*cos(pi*p/180)+a3*cos(pi*q/180)+a4*cos(pi*r/180)+430
ZC2 = (a2*sin(pi*p/180)+a3*sin(pi*q/180)+a4*sin(pi*r/180)+500)

% Validation for angle theta4
theta4 = linspace(__, __, __)
theta2 = __
theta3 = __
p = theta2;
q = theta2+theta3;
r = theta2+theta3+theta4;
XAB1 = a2*cos(pi*p/180)+a3*cos(pi*q/180)+a4*cos(pi*r/180)+430
ZAB2 = (a2*sin(pi*p/180)+a3*sin(pi*q/180)+a4*sin(pi*r/180)+500)
plot(XAB1, ZAB2, 'mo', 'LineWidth', 2, 'MarkerEdgeColor', 'k', 'MarkerFaceColor', [.49 1 .63], 'MarkerSize', 4)
hold on
plot(XE1, ZE2, 'mo', 'LineWidth', 2, 'MarkerEdgeColor', 'k', 'MarkerFaceColor', [.49 1 .63], 'MarkerSize', 4)
plot(XC1, ZC2, 'mo', 'LineWidth', 2, 'MarkerEdgeColor', 'k', 'MarkerFaceColor', [.49 1 .63], 'MarkerSize', 4)
hold off
line([-3100 3100], [0 0]);
line([0 0], [-3100 3100]);
set(gca, 'XDir', 'reverse')
axis equal
grid on

B.6 MATLAB code for validation of inverse kinematic model

The following MATLAB code is used for validation of inverse kinematic model of backhoe excavator attachment.

% Joint angle verification within -180 TO +180 degree are as below
% Validation of inverse kinematic for angle theta2
% First joint theta2 = __ to __;
% Second joint theta3 = __ to __;
% Third joint theta4 = __ to __;

clear all
clc
a2 = __;
\[ a_3 = \_ ; \]
\[ a_4 = \_ ; \]
\[ \text{theta}_2 = \text{linspace}(\_ , \_ , \_ ) ; \]
\[ \text{theta}_3 = \_ ; \]
\[ \text{theta}_4 = \_ ; \]
\[ p = \text{theta}_2 ; \]
\[ q = \text{theta}_2 + \text{theta}_3 ; \]
\[ r = \text{theta}_2 + \text{theta}_3 + \text{theta}_4 ; \]
\[ x = a_2 \cdot \cos(p_1 \cdot \text{p}/180) + a_3 \cdot \cos(p_1 \cdot q/180) + a_4 \cdot \cos(p_1 \cdot r/180) \]
\[ z = a_2 \cdot \sin(p_1 \cdot \text{p}/180) + a_3 \cdot \sin(p_1 \cdot q/180) + a_4 \cdot \sin(p_1 \cdot r/180) \]
\[ \phi_1 = \text{theta}_2 + \text{theta}_3 + \text{theta}_4 \]
\[ \text{for } i = \text{linspace}(1, 10, 10) \]
\[ \text{x1} (i) = x(i) - a_4 \cdot \cos(p_1 \cdot \text{phi}(i)/180) ; \]
\[ \text{z1} (i) = z(i) - a_4 \cdot \sin(p_1 \cdot \text{phi}(i)/180) ; \]
\[ \text{d} (i) = -\text{z1} (i) / \sqrt{\text{x1} (i) \cdot \text{x1} (i) + \text{z1} (i) \cdot \text{z1} (i)} ; \]
\[ \text{f} (i) = -\text{x1} (i) / \sqrt{\text{x1} (i) \cdot \text{x1} (i) + \text{z1} (i) \cdot \text{z1} (i)} ; \]
\[ \text{g} (i) = -(\text{x1} (i) \cdot \text{x1} (i) + \text{z1} (i) \cdot \text{z1} (i) + a_2 \cdot a_2 - a_3 \cdot a_3) / (2 \cdot a_2 \cdot \sqrt{\text{x1} (i) \cdot \text{x1} (i) + \text{z1} (i) \cdot \text{z1} (i)}) ; \]
\[ \text{theta}_22 (i) = \text{real}(\text{atan2} (\text{d}(i), \text{f}(i)) + \cos(\text{g}(i))) \cdot 180 / \pi ; \]
\[ \text{if } \text{theta}_22 > 180 \]
\[ \text{theta}_22 (i) = \text{theta}_22 (i) - 360 ; \]
\[ \text{end} \]
\[ \text{theta}_223 (i) = (\text{atan2}((\text{z1} (i) - a_2 \cdot \sin(p_1 \cdot \text{theta}_22 (i)/180)) / a_3, (\text{x1} (i) - a_2 \cdot \cos(p_1 \cdot \text{theta}_22 (i)/180)) / a_3)) \cdot 180 / \pi - \text{theta}_22 (i) ; \]
\[ \text{theta}_244 (i) = \phi_1 (i) - \text{theta}_22 (i) - \text{theta}_23 (i) ; \]
\[ \text{end} \]
\[ \text{invtheta}_2 = \text{theta}_22 ; \]
\[ \text{invtheta}_3 = \text{theta}_23 ; \]
\[ \text{invtheta}_4 = \text{theta}_24 ; \]
\[ \% \text{Validation of inverse kinematic for angle theta}_3 \]
\[ \text{clear all} \]
\[ \text{clc} \]
\[ a_2 = \_ ; \]
\[ a_3 = \_ ; \]
\[ a_4 = \_ ; \]
\[ \text{theta}_3 = \text{linspace}(\_ , \_ , \_ ) ; \]
\[ \text{theta}_4 = \_ ; \]
\[ \text{theta}_2 = \_ ; \]
\[ p = \text{theta}_2 ; \]
\[ q = \text{theta}_2 + \text{theta}_3 ; \]
\[ r = \text{theta}_2 + \text{theta}_3 + \text{theta}_4 ; \]
\[ x = a_2 \cdot \cos(p_1 \cdot \text{p}/180) + a_3 \cdot \cos(p_1 \cdot q/180) + a_4 \cdot \cos(p_1 \cdot r/180) \]
\[ z = a_2 \cdot \sin(p_1 \cdot \text{p}/180) + a_3 \cdot \sin(p_1 \cdot q/180) + a_4 \cdot \sin(p_1 \cdot r/180) \]
\[ \phi_1 = \text{theta}_2 + \text{theta}_3 + \text{theta}_4 \]
\[ \text{for } i = \text{linspace}(1, 10, 10) \]
\[ \text{x1} (i) = x(i) - a_4 \cdot \cos(p_1 \cdot \text{phi}(i)/180) ; \]
\[ \text{z1} (i) = z(i) - a_4 \cdot \sin(p_1 \cdot \text{phi}(i)/180) ; \]
\[ \text{d} (i) = -\text{z1} (i) / \sqrt{\text{x1} (i) \cdot \text{x1} (i) + \text{z1} (i) \cdot \text{z1} (i)} ; \]
\[ \text{f} (i) = -\text{x1} (i) / \sqrt{\text{x1} (i) \cdot \text{x1} (i) + \text{z1} (i) \cdot \text{z1} (i)} ; \]
\[ \text{g} (i) = -(\text{x1} (i) \cdot \text{x1} (i) + \text{z1} (i) \cdot \text{z1} (i) + a_2 \cdot a_2 - a_3 \cdot a_3) / (2 \cdot a_2 \cdot \sqrt{\text{x1} (i) \cdot \text{x1} (i) + \text{z1} (i) \cdot \text{z1} (i)}) ; \]
\[ \text{theta}_22 (i) = \_ ; \]
\[ \text{theta}_233 (i) = (\text{atan2}((\text{z1} (i) - a_2 \cdot \sin(p_1 \cdot \text{theta}_22 (i)/180)) / a_3, (\text{x1} (i) - a_2 \cdot \cos(p_1 \cdot \text{theta}_22 (i)/180)) / a_3)) \cdot 180 / \pi - \text{theta}_22 (i) ; \]
\[ \text{theta}_244 (i) = \phi_1 (i) - \text{theta}_22 (i) - \text{theta}_233 (i) ; \]
\[ \text{end} \]
\[ \text{invtheta}_2 = \text{theta}_22 \]
invtheta3 = theta33
invtheta4 = theta44

%Validation of inverse kinematic for angle theta4
clear all
clc
a2 = __;
a3 = __;
a4 = __;
theta4 = linspace(__, __, __);
theta2 = __;
theta3 = 0;
p = theta2;
q = theta2+theta3;
r = theta2+theta3+theta4;
x = a2.*cos(pi.*p/180)+a3.*cos(pi.*q/180)+a4.*cos(pi.*r/180)
z = a2.*sin(pi.*p/180)+a3.*sin(pi.*q/180)+a4.*sin(pi.*r/180)
phi = theta2+theta3+theta4
for i = linspace(1,10,10)
x1(i) = x(i)-a4*cos(pi*phi(i)/180);
z1(i) = z(i)-a4*sin(pi*phi(i)/180);
d(i) = -z1(i)/sqrt(x1(i)*x1(i)+z1(i)*z1(i));
f(i) = -x1(i)/sqrt(x1(i)*x1(i)+z1(i)*z1(i));
g(i) = -(x1(i)*x1(i)+z1(i)*z1(i)+a2*a2-a3*a3)/(2*a2*sqrt(x1(i)*x1(i)+z1(i)*z1(i)));
theta22(i) = theta2;
theta33(i) = atan2((z1(i)-a2.*sin(pi*theta22(i)/180))/a3,(x1(i)-a2.*cos(pi*theta22(i)/180))/a3)).*180/pi-
theta22(i);
theta44(i) = phi(i)-theta22(i)-theta33(i);
end
invtheta2 = theta22
invtheta3 = theta33
invtheta4 = theta44