

## **Problem Statement:**

Derive the equations of motions using dynamics methods for a 2R manipulator and verify its simulation on MapleSoft.

## **Theory:**

### ➤ Lagrangian Method:

The Lagrangian method is an energy based approach for deriving equations of motion, defining the Lagrangian (L) as the difference between Kinetic (T) and Potential (V) energies ( $L = T - V$ ). It uses independent generalized coordinates, allowing for simpler derivation of dynamics for complex systems with constraints compared to Newtonian methods.

The code symbolically computes:

- Kinetic Energy (T)
- Potential Energy (V)
- Lagrangian ( $L = T - V$ )
- Equations of Motion ( $\tau_1, \tau_2$ )

What the code does?

- Models a 2-link planar robotic arm
- Uses symbolic differentiation to automatically derive:
  - Mass matrix  $M(q)$
  - Coriolis/centrifugal terms  $C(q, \dot{q})$
  - Gravity vector  $G(q)$

Final result structure:

$$\tau = M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q)$$

### ➤ Newton-Euler Method:

In classical mechanics, the Newton–Euler equations describe the combined translational and rotational dynamics of a rigid body.

Traditionally the Newton–Euler equations is the grouping together of Euler's two laws of motion for a rigid body into a single equation with 6 components, using column vectors and matrices. These laws relate the motion of the center of gravity of a rigid body with the sum of forces and torques (or synonymously moments) acting on the rigid body.

restart;

with(LinearAlgebra) :

t := 't':

#Symbolic Representation of ' $\theta_1(t)$ ' and ' $\theta_2(t)$ ' as ' $q_1$ ' and ' $q_2$ ' is needed as the using generalized coordinates with differentiation of Lagrangian becomes invalid. Being a dependent variable to time 't', it cannot be used as a second argument for differentiation in Maple. For this we have converted these variables to ' $q_1$ ' and ' $q_2$ '. Similar process has also been done with  $dq_1$  and  $dq_2$ .

# Independent symbolic variables

$q_1 := 'q_1'$ ;  $q_2 := 'q_2'$ ;

$dq_1 := 'dq_1'$ ;  $dq_2 := 'dq_2'$ ;

$L := (1/2) * m_1 * dq_1^2$   
 $+ (1/2) * m_2 * (dq_1^2 + dq_2^2 + 2 * dq_1 * dq_2 * \cos(q_2))$   
 $- (m_1 * g * l_1 * \cos(q_1) + m_2 * g * (l_1 * \cos(q_1) + l_2 * \cos(q_1 + q_2)))$ ;

$$L := \frac{m_1 dq_1^2}{2} + \frac{m_2 (dq_1^2 + dq_2^2 + 2 dq_1 dq_2 \cos(q_2))}{2} - m_1 g l_1 \cos(q_1) - m_2 g (l_1 \cos(q_1) + l_2 \cos(q_1 + q_2)) \quad (1)$$

#Simplifying tau1 and tau2 equations:

$dLdq_1 := \text{diff}(L, q_1)$ ;

$dLdq_2 := \text{diff}(L, q_2)$ ;

$dLddq_1 := \text{diff}(L, dq_1)$ ;

$dLddq_2 := \text{diff}(L, dq_2)$ ;

$dLdq_1 := m_1 g l_1 \sin(q_1) - m_2 g (-l_1 \sin(q_1) - l_2 \sin(q_1 + q_2))$

$dLdq_2 := -m_2 dq_1 dq_2 \sin(q_2) + m_2 g l_2 \sin(q_1 + q_2)$

$dLddq_1 := m_1 dq_1 + \frac{m_2 (2 dq_1 + 2 dq_2 \cos(q_2))}{2}$

$$dLddq_2 := \frac{m_2 (2 dq_2 + 2 dq_1 \cos(q_2))}{2} \quad (2)$$

$subs\_map := \{$   
 $q_1 = \theta_1(t),$   
 $q_2 = \theta_2(t),$   
 $dq_1 = \text{diff}(\theta_1(t), t),$   
 $dq_2 = \text{diff}(\theta_2(t), t)$   
 $\};$

$dLddq_1\_t := \text{subs}(subs\_map, dLddq_1)$ ;

$dLddq_2\_t := \text{subs}(subs\_map, dLddq_2)$ ;

$$subs\_map := \left\{ dq1 = \frac{d}{dt} \theta1(t), dq2 = \frac{d}{dt} \theta2(t), q1 = \theta1(t), q2 = \theta2(t) \right\}$$

$$dLddq1\_t := m1 \left( \frac{d}{dt} \theta1(t) \right) + \frac{m2 \left( 2 \frac{d}{dt} \theta1(t) + 2 \left( \frac{d}{dt} \theta2(t) \right) \cos(\theta2(t)) \right)}{2}$$

$$dLddq2\_t := \frac{m2 \left( 2 \frac{d}{dt} \theta2(t) + 2 \left( \frac{d}{dt} \theta1(t) \right) \cos(\theta2(t)) \right)}{2} \quad (3)$$

# -----

# Step 5: Euler-Lagrange

# -----

EL := simplify( Vector([  
diff(dLddq1\_t, t) - dLdq1,  
diff(dLddq2\_t, t) - dLdq2  
]));

EL := (4)

$$\begin{bmatrix} (m1 + m2) \left( \frac{d^2}{dt^2} \theta1(t) \right) + \cos(\theta2(t)) \left( \frac{d^2}{dt^2} \theta2(t) \right) m2 - m2 g l2 \sin(q1 + q2) - \sin(\theta2(t)) \left( \frac{d}{dt} \theta1(t) \right) \left( \frac{d}{dt} \theta2(t) \right) \\ m2 \left( \frac{d^2}{dt^2} \theta2(t) + \left( \frac{d^2}{dt^2} \theta1(t) \right) \cos(\theta2(t)) - \left( \frac{d}{dt} \theta1(t) \right) \left( \frac{d}{dt} \theta2(t) \right) \sin(\theta2(t)) \end{bmatrix}$$

# -----

# Step 6: Extract M, C, G

# -----

ddtheta1 := diff(theta1(t), t\$2) :

ddtheta2 := diff(theta2(t), t\$2) :

EL := collect(EL, [ddtheta1, ddtheta2]) :

# Mass Matrix

M11 := coeff(EL[1], ddtheta1) :

M12 := coeff(EL[1], ddtheta2) :

M21 := coeff(EL[2], ddtheta1) :

M22 := coeff(EL[2], ddtheta2) :

M := Matrix([ [M11, M12],  
[M21, M22] ]) :

# Remaining terms

H := simplify(EL - M . Vector([ddtheta1, ddtheta2])) :

# Gravity (zero velocities)

```
G := simplify(subs( {
diff(theta1(t), t) = 0,
diff(theta2(t), t) = 0
}, H)) :
```

# Coriolis/Centrifugal

```
C := simplify(H - G) :
```

# -----

# Final Outputs

# -----

```
print( `-----Inertia Matrix M-----` ) :
```

M;

```
print( `-----Coriolis Force Vector C-----` ) :
```

C;

```
print( `-----Gravity Vector G-----` ) :
```

G;

-----Inertia Matrix M-----

$$\begin{bmatrix} m1 + m2 & \cos(\theta2(t)) m2 \\ \cos(\theta2(t)) m2 & m2 \end{bmatrix}$$

-----Coriolis Force Vector C-----

$$\begin{bmatrix} -\sin(\theta2(t)) \left( \frac{d}{dt} \theta2(t) \right)^2 m2 \\ -\left( \frac{d}{dt} \theta1(t) \right) \sin(\theta2(t)) \left( \frac{d}{dt} \theta2(t) \right) m2 \end{bmatrix}$$

-----Gravity Vector G-----

$$\begin{bmatrix} -g (l2 m2 \sin(q1 + q2) + l1 (m1 + m2) \sin(q1)) \\ m2 (dq1 dq2 \sin(q2) - g l2 \sin(q1 + q2)) \end{bmatrix}$$

(5)

> restart;

with(LinearAlgebra) :

# 1. SETUP PARAMETERS AND VARIABLES

# Using 'TH' instead of 'theta' because 'theta' is a protected symbol

TH := Vector([theta1(t), theta2(t)]);

omega := Vector([diff(theta1(t), t), diff(theta2(t), t)]);

alpha := Vector([diff(theta1(t), t, t), diff(theta2(t), t, t)]);

# Geometric and Dynamic Constants

L := Vector([L1, L2]);

#Considering The Center of mass at the edge of the Link

Lc := Vector([L1, L2]);

M := Vector([m1, m2]);

I\_val := Vector([I1, I2]);

# Gravity vector (Acting in negative Y direction)

g\_vec := Vector([0, -g, 0]);

# 2. FORWARD RECURSION (Kinematics)

# Base is stationary

w[0] := Vector([0, 0, 0]);

dw[0] := Vector([0, 0, 0]);

dv[0] := Vector([0, 0, 0]);

**for i from 1 to 2 do**

# Rotation axis (Z-axis for planar)

z\_axis := Vector([0, 0, 1]);

# Angular Velocity

w[i] := w[i-1] + z\_axis \* omega[i];

# Angular Acceleration

dw[i] := dw[i-1] + z\_axis \* alpha[i] + CrossProduct(w[i-1], z\_axis \* omega[i]);

# Position vectors relative to previous joint

# Angle is cumulative for a 2R manipulator

current\_angle := add(TH[j], j = 1 .. i);

r\_i := Vector([L[i] \* cos(current\_angle), L[i] \* sin(current\_angle), 0]);

r\_ci := Vector([Lc[i] \* cos(current\_angle), Lc[i] \* sin(current\_angle), 0]);

# Linear Acceleration of the joint i

```
dv[i] := dv[i-1] + CrossProduct(dw[i], r_i) + CrossProduct(w[i], CrossProduct(w[i],  
r_i));
```

```
# Linear Acceleration of the Center of Mass of link i
```

```
dv_c[i] := dv[i-1] + CrossProduct(dw[i], r_ci) + CrossProduct(w[i],  
CrossProduct(w[i], r_ci));
```

```
end do:
```

```
# 3. BACKWARD RECURSION (Dynamics)
```

```
# Forces/torques from end-effector (none)
```

```
f[3] := Vector([0, 0, 0]);
```

```
n[3] := Vector([0, 0, 0]);
```

```
for i from 2 by -1 to 1 do
```

```
# Inertial Force ( $F = m*a$ ) - we subtract gravity to include it as an external force
```

```
F_i := M[i] * (dv_c[i] - g_vec);
```

```
# Inertial Torque (Euler Equation)
```

```
# For planar 2D, inertia is a scalar around the Z axis
```

```
N_i := Vector([0, 0, I_val[i] * alpha[i]]);
```

```
# Force balance
```

```
f[i] := f[i+1] + F_i;
```

```
# Torque balance
```

```
current_angle := add(TH[j], j=1..i);
```

```
r_ci_vec := Vector([Lc[i] * cos(current_angle), Lc[i] * sin(current_angle), 0]);
```

```
r_i_vec := Vector([L[i] * cos(current_angle), L[i] * sin(current_angle), 0]);
```

```
n[i] := n[i+1] + CrossProduct(r_ci_vec, F_i) + CrossProduct(r_i_vec, f[i+1]) + N_i;
```

```
# Final joint torque is the Z-component
```

```
tau[i] := simplify(n[i][3]);
```

```
end do:
```

```
# 4. DERIVE M, C, G MATRICES
```

```
# Torque equation form:  $\tau = M(q)\ddot{q} + V(q, \dot{q}) + G(q)$ 
```

```
# --- Gravity Vector G ---
```

```
# Extract terms containing 'g'
```

```
G1 := coeff(tau[1], g) * g;
```

```
G2 := coeff(tau[2], g) * g;
```

```
G_vec := Vector([G1, G2]);
```

```

# --- Inertia Matrix M ---
# Extract coefficients of the second derivatives (alpha)
M11 := coeff(tau[1], alpha[1]);
M12 := coeff(tau[1], alpha[2]);
M21 := coeff(tau[2], alpha[1]);
M22 := coeff(tau[2], alpha[2]);
M_mat := Matrix([ [M11, M12], [M21, M22] ]);

# --- Coriolis and Centripetal Vector C ---
# C = Total Torque - M*ddq - G
C1 := simplify(tau[1] - (M11 * alpha[1] + M12 * alpha[2]) - G1);
C2 := simplify(tau[2] - (M21 * alpha[1] + M22 * alpha[2]) - G2);
C_vec := Vector([C1, C2]);

# 5. DISPLAY SEPARATED MATRICES
print('--- Inertia Matrix (M) ---');
print('*M11*');
print(simplify(M11));
print('*M12*');
print(simplify(M12));
print('*M21*');
print(simplify(M21));
print('*M22*');
print(simplify(M22));

print('--- Coriolis/Centripetal Vector (C) ---');
print(simplify(C_vec));

print('--- Gravity Vector (G) ---');
print(simplify(G_vec));

#6.Displaying Joint Torques
print('---Joint torque 1---');
print(tau[1]);
print('---Joint Torque 2---');
print(tau[2]);

```

$$TH := \begin{bmatrix} \theta 1(t) \\ \theta 2(t) \end{bmatrix}$$

$$\mathfrak{w}:=\left[\begin{array}{c} \frac{\mathrm{d}}{\mathrm{d} t} \ \theta l(t) \\ \frac{\mathrm{d}}{\mathrm{d} t} \ \theta 2(t) \end{array}\right]$$

$$\alpha:=\left[\begin{array}{c} \frac{\mathrm{d}^2}{\mathrm{d} t^2} \ \theta l(t) \\ \frac{\mathrm{d}^2}{\mathrm{d} t^2} \ \theta 2(t) \end{array}\right]$$

$$L:=\left[\begin{array}{c} L1 \\ L2 \end{array}\right]$$

$$Lc:=\left[\begin{array}{c} L1 \\ L2 \end{array}\right]$$

$$M:=\left[\begin{array}{c} m1 \\ m2 \end{array}\right]$$

$$I\_val:=\left[\begin{array}{c} I1 \\ I2 \end{array}\right]$$

$$g\_vec:=\left[\begin{array}{c} 0 \\ -g \\ 0 \end{array}\right]$$

$$w_0:=\left[\begin{array}{c} 0 \\ 0 \\ 0 \end{array}\right]$$

$$dw_0:=\left[\begin{array}{c} 0 \\ 0 \\ 0 \end{array}\right]$$

$$dv_0:=\left[\begin{array}{c} 0 \\ 0 \\ 0 \end{array}\right]$$



$$f_3 := \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$n_3 := \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$G1 := (L2 \cos(\theta1(t) + \theta2(t)) m2 + \cos(\theta1(t)) (m1 + m2) L1) g$$

$$G2 := L2 \cos(\theta1(t) + \theta2(t)) g m2$$

$$G\_vec := \begin{bmatrix} (L2 \cos(\theta1(t) + \theta2(t)) m2 + \cos(\theta1(t)) (m1 + m2) L1) g \\ L2 \cos(\theta1(t) + \theta2(t)) g m2 \end{bmatrix}$$

$$M11 := 2 \cos(\theta2(t)) L1 L2 m2 + (L1^2 + L2^2) m2 + L1^2 m1 + I1$$

$$M12 := \cos(\theta2(t)) L1 L2 m2 + L2^2 m2 + I2$$

$$M21 := L2 m2 (L1 \cos(\theta2(t)) + L2)$$

$$M22 := L2^2 m2 + I2$$

$$M\_mat :=$$

$$\begin{bmatrix} 2 \cos(\theta2(t)) L1 L2 m2 + (L1^2 + L2^2) m2 + L1^2 m1 + I1 & \cos(\theta2(t)) L1 L2 m2 + L2^2 m2 + I2 \\ L2 m2 (L1 \cos(\theta2(t)) + L2) & L2^2 m2 + I2 \end{bmatrix}$$

$$C1 := -L1 L2 \sin(\theta2(t)) \left( \frac{d}{dt} \theta2(t) \right) m2 \left( \frac{d}{dt} \theta2(t) + 2 \frac{d}{dt} \theta1(t) \right)$$

$$C2 := L1 L2 \left( \frac{d}{dt} \theta1(t) \right)^2 \sin(\theta2(t)) m2$$

$$C\_vec := \begin{bmatrix} -L1 L2 \sin(\theta2(t)) \left( \frac{d}{dt} \theta2(t) \right) m2 \left( \frac{d}{dt} \theta2(t) + 2 \frac{d}{dt} \theta1(t) \right) \\ L1 L2 \left( \frac{d}{dt} \theta1(t) \right)^2 \sin(\theta2(t)) m2 \end{bmatrix}$$

$$--- Inertia Matrix (M) ---$$

$$*M11*$$

$$2 \cos(\theta_2(t)) L_1 L_2 m_2 + (m_1 + m_2) L_1^2 + L_2^2 m_2 + I_1$$

\*M12\*

$$\cos(\theta_2(t)) L_1 L_2 m_2 + L_2^2 m_2 + I_2$$

\*M21\*

$$L_2 m_2 (L_1 \cos(\theta_2(t)) + L_2)$$

\*M22\*

$$L_2^2 m_2 + I_2$$

--- Coriolis/Centripetal Vector (C) ---

$$\begin{bmatrix} -L_1 L_2 \sin(\theta_2(t)) \left( \frac{d}{dt} \theta_2(t) \right) m_2 \left( \frac{d}{dt} \theta_2(t) + 2 \frac{d}{dt} \theta_1(t) \right) \cdots \\ L_1 L_2 \left( \frac{d}{dt} \theta_1(t) \right)^2 \sin(\theta_2(t)) m_2 \cdots \end{bmatrix}$$

--- Gravity Vector (G) ---

$$\begin{bmatrix} (L_2 \cos(\theta_1(t) + \theta_2(t)) m_2 + \cos(\theta_1(t)) (m_1 + m_2) L_1 \cdots \\ L_2 \cos(\theta_1(t) + \theta_2(t)) g m_2 \cdots \end{bmatrix}$$

---Joint toque 1---

$$\begin{aligned} & L_2 \cos(\theta_1(t) + \theta_2(t)) g m_2 + (2 \cos(\theta_2(t)) L_1 L_2 m_2 + (L_1^2 + L_2^2) m_2 + L_1^2 m_1 \\ & + I_1) \left( \frac{d^2}{dt^2} \theta_1(t) \right) + (\cos(\theta_2(t)) L_1 L_2 m_2 + L_2^2 m_2 + I_2) \left( \frac{d^2}{dt^2} \theta_2(t) \right) + \left( \right. \\ & -\sin(\theta_2(t)) L_2 m_2 \left( \frac{d}{dt} \theta_2(t) \right)^2 - 2 \sin(\theta_2(t)) \left( \frac{d}{dt} \theta_1(t) \right) L_2 m_2 \left( \frac{d}{dt} \theta_2(t) \right) \\ & \left. + g \cos(\theta_1(t)) (m_1 + m_2) \right) L_1 \end{aligned}$$

---Joint Torque 2---

$$\begin{aligned} & L_2 \cos(\theta_1(t) + \theta_2(t)) g m_2 + L_2 m_2 (L_1 \cos(\theta_2(t)) + L_2) \left( \frac{d^2}{dt^2} \theta_1(t) \right) + (L_2^2 m_2 \\ & + I_2) \left( \frac{d^2}{dt^2} \theta_2(t) \right) + L_1 L_2 \left( \frac{d}{dt} \theta_1(t) \right)^2 \sin(\theta_2(t)) m_2 \end{aligned} \quad (1)$$

>