Inverse Dynamics Control with Sliding Mode Neural Network compensator for Space Manipulators
Motivation

- Mathematical modeling of Robotic Manipulator.
- Design of high precision control system subject to model uncertainties and external disturbances.
Objectives
OBJECTIVES

• Development of Two-Link Manipulator Simulation Model in MATLAB/Simulink®.

• Design of Inverse dynamic control explicitly based on mathematical model

• Design of compensator using Adaptive Sliding mode based Radial Basis Function Neural Network.

• Efficacy proof of designed controller by simulation results and its comparison with Computed Torque Control.
Modeling of Two Link Planar Robotic Arm
Mathematical Modeling
The mathematical model of Two-Link Robotic Manipulator consists of two types of modeling

A. Kinematic Modeling
   a. Forward Kinematics
   b. Inverse Kinematics

B. Dynamic Modeling
   a. Newton–Euler
   b. Euler-Lagrange formulations
Kinematic Modeling

*Forward Kinematics:* It gives position of its end effector with the knowledge of all the joint angles.

\[ \xi = K(q) \]

*Inverse Kinematics:* It gives all the Joint angles with the knowledge of position of its end effector.

\[ \xi = K(q) \]

For computing Forward and Inverse Kinematics, Denavit–Hartenberg (DH) is frequently used technique for Kinematics modeling.

<table>
<thead>
<tr>
<th>j</th>
<th>Theta</th>
<th>d</th>
<th>a</th>
<th>Alpha</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>q1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>q2</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
**Dynamic Modeling**

The dynamic equation of n-link manipulator is written as

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + F(\dot{q}) + G(q) + T_d = Q$$

$$M(q) = \begin{bmatrix}
\frac{1}{4} m_1 + m_2 & l_1^2 + \frac{1}{4} m_2 l_2^2 + \frac{1}{2} m_2 l_1 l_2 \cos q_2 & \frac{1}{4} m_2 l_2^2 + \frac{1}{2} m_2 l_1 l_2 \cos q_2 \\
\frac{1}{4} m_2 l_2^2 + \frac{1}{2} m_2 l_1 l_2 \cos q_2 & \frac{1}{4} m_2 l_2^2 & \frac{1}{4} m_2 l_2^2 \\
\end{bmatrix}$$

$$G(q) = \begin{bmatrix}
\frac{1}{2} m_1 + m_2 & g l_1 \cos q_1 + \frac{1}{2} m_2 g l_2 \cos (q_1 + q_2) \\
\frac{1}{2} m_2 g l_2 \cos (q_1 + q_2) \\
\end{bmatrix}$$

$$C(q, \dot{q})\dot{q} = \begin{bmatrix}
-\frac{1}{2} m_2 l_1 l_2 (2\dot{q}_1 \dot{q}_2 + \dot{q}_2^2) \sin q_2 \\
\frac{1}{4} m_2 l_1 l_2 \dot{q}_1^2 \sin q_2
\end{bmatrix}$$

$$F(\dot{q}) = \begin{bmatrix}
f_1 \text{sign}(\dot{q}_1) \\
f_2 \text{sign}(\dot{q}_2)
\end{bmatrix}$$
Autopilot Design
Challenges

➢ Highly nonlinearity
➢ High degree of coupling
➢ Model uncertainty and time-variant
**IDSNN Control Algorithm**

To precisely follow the desired trajectory, the proposed control architecture consists of:

*Feed-forward term*: based on Inverse Dynamics of manipulator which computes the control moment or required torque which is explicitly based on the modeled dynamics of the system.

*Feedback term*: based on sliding mode based neural network compensator is employed in order to compensate for unmodeled dynamics and to guarantee desired closed-loop behavior.
IDSNN Control Algorithm

Inverse Dynamic Control

- The dynamics equation of manipulator is written as:

\[
M(q)\ddot{q} + C(q, \dot{q})\dot{q} + F(\dot{q}) + G(q) + T_d = Q
\]

- Recursive Newton-Euler algorithm is applied to derive the equation of inverse dynamics of manipulator
IDSNN Control Algorithm

Shortcoming! (Inverse Dynamic Control)

Exact Dynamic Model

- Structured uncertainties
- Unstructured uncertainties.

It is necessary to devise a control strategy that can handle these nonlinear factors and uncertainties.
IDSNN Control Algorithm

Sliding Mode based NN Compensator

Sliding Mode based Radial Basis Neural Network compensator is used to compensate the errors due to unmodeled dynamics and disturbances.
IDSNN Control Algorithm

Sliding Mode based NN Compensator

The input to the RBF NN compensator is the sliding surface, based on joint error and its rate denoted by:

\[ S_{qn} = Ce + \dot{e} \]

The neuron in hidden layer is defined by Radial basis function i.e.

\[ h_i(S) = \exp \left( \frac{\|s - c_i\|^2}{2\sigma_i^2} \right), \quad i = 1, 2, \ldots 5. \]

The output of Sliding mode based RBF NN compensator is written as

\[ U_{nj} = \hat{w}_i^* h_i \]
IDSNN Control Algorithm

Sliding Mode based NN Compensator

- Lyapunov synthesis approach is used to develop an adaptive control algorithm in which the weights of the RBF neural network are adjusted adaptively online.
IDSNN Control Algorithm

Lyapunov Stability

- Lyapunov function is defined as:

\[
L = \frac{1}{2} \dot{s}^2 + \frac{1}{2} \gamma \tilde{w}^T \tilde{w}
\]

- Stability Criteria

\[
\dot{L} \leq 0
\]

- The derivation is written as:

\[
\dot{L} = s \dot{s} + \gamma \tilde{w}^T \dot{\tilde{w}}
\]
IDSNN Control Algorithm

Problem formulation

- Let the nonlinear system is defined as:

\[ \ddot{x} = f(x, \dot{x}) + gu + dt \]

- We know that

\[ s = ce + \dot{e} \]

\[ \dot{s} = \dot{e} + c\dot{e} = \ddot{x}_d - \dot{x} + c\dot{e} = \ddot{x}_d - fx - gu - dt + c\dot{e} = -k\text{sign}(s) \]

\[ u = \frac{1}{g} (-\ddot{x}_d - fx - dt + c\dot{e} + k\text{sign}(s)) \]

- Placing u we can write:

\[ \dot{s} = -fx - dt - k\text{sign}(s) \]
IDSNN Control Algorithm

Lyapunov Stability

- The derivation is written as:

\[ \dot{L} = ss + \gamma \tilde{w}^T \dot{\tilde{w}} \]

\[ \dot{L} = -\tilde{w}^T (sh(s) + \gamma \dot{\tilde{w}}) - s(\varepsilon + dt + k\text{sign}(s)) \]

- The adaptive law is selected as

\[ \dot{\tilde{w}} = -\frac{1}{\gamma} sh(s) \]
IDSNN Control Algorithm

Lyapunov Stability

- Now we can write:
  \[
  \dot{L} = -s(\varepsilon + dt + k\text{sign}(s)) = -s(\varepsilon + dt) - k|s|
  \]

- Since approximation error is sufficiently small and \( k \geq \varepsilon + dt \)

We get

\[ L\dot{L} \leq 0 \]
IDSNN Control Algorithm

Block Diagram
Simulink Model

INVERSE DYNAMIC WITH ADAPTIVE SLIDING MODE RBF NEURAL NETWORK COMPENSATOR OF TWO LINK MANIPULATOR

MODELING AND CONTROL OF QUADROTOR
Simulation Results
Simulation results

To evaluate the performance of proposed controller, a reference trajectory is generated in XY co-ordinate.

Initial pose: (1.5m, -0.5m)
Final pose: (-1.5m, 0.8m)
Simulation results

Trajectory tracking using Inverse Dynamic

Joint 1 trajectory

Joint 2 trajectory
Simulation results

- To evaluate the robustness, 20%-30% variations in system dynamics are included.
- The following slides will show the performance of Inverse Dynamics control.
Simulation results

Trajectory tracking using Inverse Dynamic

Joint 1 trajectory

Joint 2 trajectory
Simulation results
Comparison of IDSNN with ID

Joint 1 trajectory

Time [Sec]

q1 [Deg]

Reference
IDSNN
ID

Joint 1 Torque

Time [Sec]

Q [Nm]

IDSNN
ID
Simulation results

Comparison of IDSNN with ID

Joint 2 trajectory

Joint 2 Torque
Simulation results

Comparison of IDSNN with CTC

Joint 1 trajectory

Joint 1 Error
SIMULATION RESULTS

Manipulator Dynamic Dynamic Simulator
Conclusion

• The Mathematical modeling of two-link planar robotic manipulator and its control design using IDSNN is successfully implemented in Simulink/MATLAB.

• Sliding Mode based Neural Network feedback term is added to compensate for unmodeled dynamics and to guarantee stability.

• The weights of the neural networks adjusts adaptively based on Lyapunov principle to make controller robust against parametric variations.

• The results also depict that the feedback control using SNN effectively deals with the imperfections remaining in calculating the inverse dynamics of the manipulator.

• A comparison of the proposed algorithm has also been carried out with conventional computed torque controller that indicates the superior performance of the proposed design.
Future Work

• Development of mathematical model of free-floating system using Virtual Manipulator technique in order to conserve fuel which permits the spacecraft to move freely in response to manipulator motions

• The development of image based visual servoing algorithm
Questions?