SYSTEM IDENTIFICATION OF ROBOTIC SYSTEM

CP 214: Foundations of Robotics

Produced By

Himanshu Sharma
Anujkumar Prajapati
Ritam Chattopadhyay
Kunal Sah

Indian Institute of Science
Bangalore

February 3, 2021

Contents

INTRODUCTION

The mathematical modelling of the real systems, which comprise complex nonlinear dynamics, is a tedious task due to uncertain parameters, nonlinear components, as well as the time-varying nature of the system. Additionally, unlike linear systems, nonlinear equations do not have a specific definition, besides, there is no specific approach or universal methodology in mathematical solutions. Furthermore, closed-form expressions for the solutions of the linear systems are not possible to solve nonlinear systems. Therefore, system identification as well as model discovery are extremely important tasks in the control systems engineering framework.

The main objective of this project is how to obtain an appropriate mathematical model of a dynamic system on the basis of observed time series and prior knowledge of the system.
Basically four problem areas in system identification can be distinguished:
Modelling: A critical step in the application of system theory to a real process is to find a mathematical model which adequately describes the physical situation. First the system boundaries and the system variables have to be specified. Then relation between these variables have to be specified on the basis of prior knowledge and assumption about the uncertainties in the model have to be made. This all together defines the model structure.
Analysis: After modelling comes the analyzation of the system output behaviour by simulation. In addition to this, the stability of the system and the different time scales governed by the system dynamics are important issues to be investigated.
Estimation: A next step, after having obtained an appropriate (un)stable, identifiable, and observable model structure is concerned with the estimation of the unknown variables from a given data set of input-output variables. Basically we distinguish between state estimation and parameter estimation or identification.
In state estimation problems one tries to estimate the states x from the output y under the assumption that the model is perfect and thus the parameters are exactly known. Similarly, parameter estimation focuses on the problem of estimating the model parameters.
Control: The control problem focuses on the calculation of the input u such that the controlled system shows the desired behaviour. Basically one defines two type of control strategies, open-loop and closed-loop controls. The following pages contain information about how we proceeded for the problem statement and the results obtained from the project.

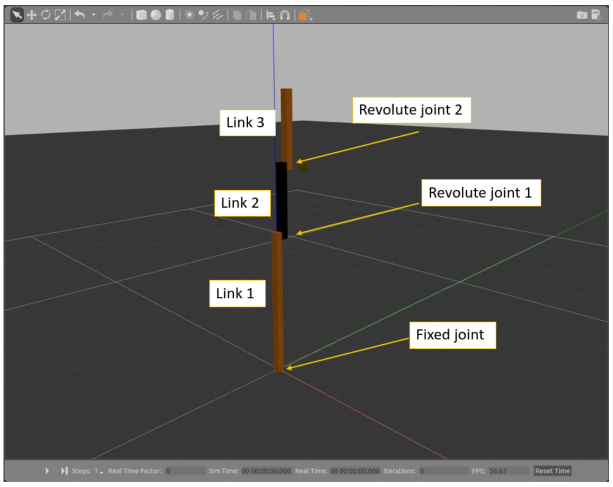
1. DATA COLLECTION

Gazebo
Gazebo is a 3D dynamic simulator with the ability to accurately and efficiently simulate populations of robots in complex indoor and outdoor environments.
ROS The Robot Operating System (ROS) is a flexible framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms.

Since, we did not have access to an actual robotic manipulator, we simulated a simplified version of the robot using Gazebo and using ROS we observed and extracted the data in a usable format.

We followed various tutorials and also referred documentation for Gazebo and ROS to gain a bare minimum understanding of the interface, commands, API, etc.
State : the pose and twist of a rigid body or a link.
Properties : intrinsic properties such as mass, moment of inertia, coefficient of friction, damping factors, etc.
Model : a conglomeration of bodies or links connected by joints.
Joints : connection between two links which may be fixed, continuous (revolute), prismatic, etc.
Topics : channels on which messages are sent and received. It's a connecting link between the publisher and subscriber.
Publishers : platform on which messages are sent.
Subscribers : platform on which messages are received .

Further, we selected a simplified 3 link robot with 2 revolute pairs (commonly known as R-R robot) and prepared a URDF model.



The URDF (Universal Robot Description Format) model is an XML file that describes the physical description of the robot. ROS uses this description to carry out the simulation of situations.
The code for the URDF files has been included in the Appendix.
To open a gazebo instance integrated with ROS.

Start a ROS master using the roscore command

```
$ roscore
```

Launch a Gazebo GUI using

```
$ rosrun gazebo_ros gzclient
```

Launch the rrbot launch file using

```
$ roslaunch rrbot_gazebo rrbot_world.launch
```

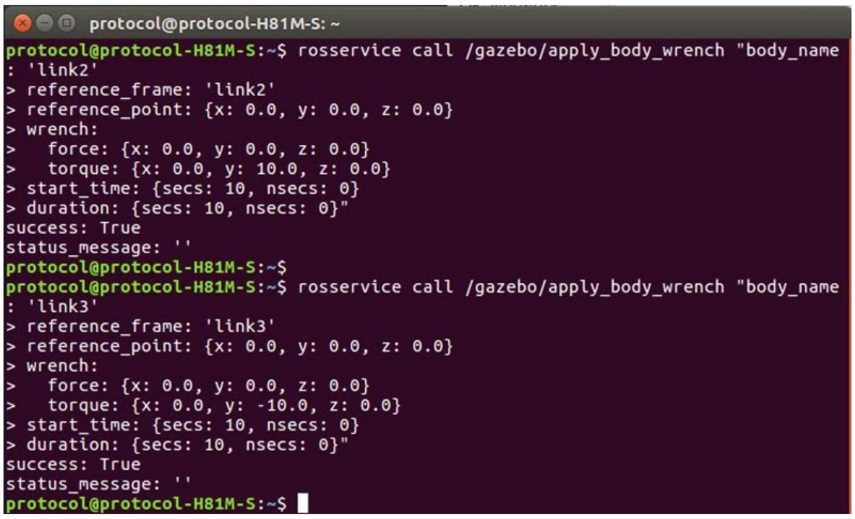
The above steps shall launch a simulation of the required RR robot. The steps 1 and 2 are not required and the roslaunch command itself starts the master, however using the roscore command it was observed that the server didn't crash often.
Next we applied a body wrench on each of the rotating links using the *apply_body_wrench* service. A list of services available is available using the command

```
$ rosservice list
```

Which gives an output similar to this

```
/gazebo/apply_body_wrench
/gazebo/apply_joint_effort
/gazebo/clear_body_wrenches
/gazebo/clear_joint_forces
/gazebo/delete_light
/gazebo/delete_model
/gazebo/get_joint_properties
/gazebo/get_light_properties
/gazebo/get_link_properties
/gazebo/get_link_state
/gazebo/get_loggers
/gazebo/get_model_properties
/gazebo/get_model_state
.
.
.
```

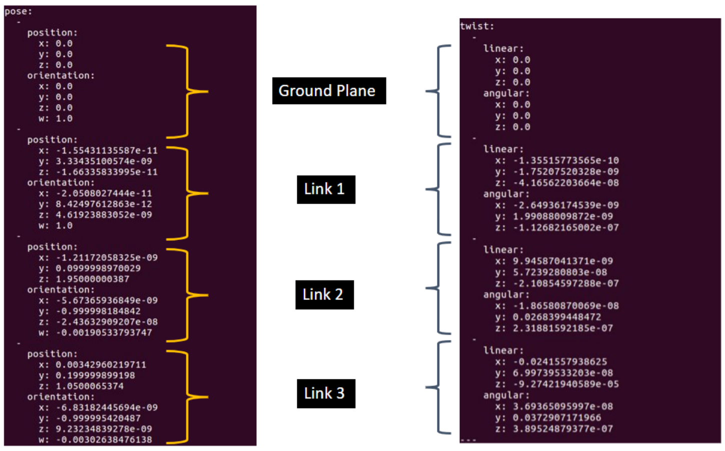
The above output indicates the connection of ROS with Gazebo.
The *apply_body_wrench* service is called by the *rosservice* call command which has the structure:



```
protocol@protocol-H81M-S: ~
protocol@protocol-H81M-S:~$ rosservice call /gazebo/apply_body_wrench "body_name
: 'link2'
> reference_frame: 'link2'
> reference_point: {x: 0.0, y: 0.0, z: 0.0}
> wrench:
>   force: {x: 0.0, y: 0.0, z: 0.0}
>   torque: {x: 0.0, y: 10.0, z: 0.0}
> start_time: {secs: 10, nsecs: 0}
> duration: {secs: 10, nsecs: 0}"
success: True
status_message: ''
protocol@protocol-H81M-S:~$
protocol@protocol-H81M-S:~$ rosservice call /gazebo/apply_body_wrench "body_name
: 'link3'
> reference_frame: 'link3'
> reference_point: {x: 0.0, y: 0.0, z: 0.0}
> wrench:
>   force: {x: 0.0, y: 0.0, z: 0.0}
>   torque: {x: 0.0, y: -10.0, z: 0.0}
> start_time: {secs: 10, nsecs: 0}
> duration: {secs: 10, nsecs: 0}"
success: True
status_message: ''
protocol@protocol-H81M-S:~$
```

The state of the links can be obtained by subscribing to the link_states topic

```
$ rostopic echo /gazebo/link_states
```



```
pose:
  -
    position:
      x: 0.0
      y: 0.0
      z: 0.0
    orientation:
      x: 0.0
      y: 0.0
      z: 0.0
      w: 1.0
  -
    position:
      x: -1.55431135587e-11
      y: 3.33435100574e-09
      z: -1.66335833995e-11
    orientation:
      x: -2.0508027444e-11
      y: 8.42497612863e-12
      z: 4.61923883052e-09
      w: 1.0
  -
    position:
      x: -1.21172058325e-09
      y: 0.0999998970029
      z: 1.95000000387
    orientation:
      x: -5.67365936849e-09
      y: -0.999998184842
      z: -2.43632909207e-08
      w: -0.00190533793747
  -
    position:
      x: 0.00342960219711
      y: 0.199999899198
      z: 1.0500065374
    orientation:
      x: -6.83182445694e-09
      y: -0.999995420487
      z: 9.23234839278e-09
      w: -0.00302638476138
```

```
twist:
  -
    linear:
      x: 0.0
      y: 0.0
      z: 0.0
    angular:
      x: 0.0
      y: 0.0
      z: 0.0
  -
    linear:
      x: -1.35515773565e-10
      y: -1.75207520328e-09
      z: -4.16562203664e-08
    angular:
      x: -2.64936174539e-09
      y: 1.99088009872e-09
      z: -1.12682165002e-07
  -
    linear:
      x: 9.94587041371e-09
      y: 5.7239280803e-08
      z: -2.10854597288e-07
    angular:
      x: -1.86580870069e-08
      y: 0.0268399448472
      z: 2.31881592185e-07
  -
    linear:
      x: -0.0241557938625
      y: 6.99739533203e-08
      z: -9.27421940589e-05
    angular:
      x: 3.69365095997e-08
      y: 0.0372907171966
      z: 3.89524879377e-07
---
```

The above data is recorded in a bag file.

```
$ rosbag record -O filename /topic1 /topic2
```

Thus we obtain 7 pose data and 5 twist data for each of the 3 links as well as the ground plane i.e. 52 data columns for each instant, resulting in a total of 53 columns.
The data required data subset from the bag file is converted into a usable format such as `.csv` format.

```
$ rostopic echo gazebo/link_states -b filename.bag -p > filename.csv
```

The above obtained data is in a crude form and needs to be processed and sorted before feeding into the ML model.
For training purpose, the matrix X = All 53 features (columns) and t = timestamp.
Data collection has been done with the help of Gazebo simulator.
Generated data has 53 columns containing timestamp, positions, orientation, linear velocity, angular velocity (twist coordinates) for all three dimensions of all four links (including the ground plane).

Model is trained with 5000 data point (chosen low due to computation cost).

The main algorithmic framework used for learning in the project is **SINDy** (System Identification for Nonlinear Dynamics).
SINDy aims at finding the closest approximation to the rate of change of state variables in terms of the state variables.

2. BRIEF MATHEMATICAL OVERVIEW OF SINDy

(2.1) Let $x(t) \in \mathbb{R}^n$ denote the state variables of a nonlinear dynamical system.
The aim of SINDy is to approximate $\frac{d}{dt}x(t) = f(x(t))$, where $f(x(t))$ is some function which needs to be explored.

(2.2) Observations of the system variables are collected at time instants $t_1, t_2, ...t_m$.
Matrices to be considered:

$$X = \begin{bmatrix} x^T(t_1) \\ : \\ x^T(t_m) \end{bmatrix}$$

$$\dot{X} = \begin{bmatrix} \dot{x}^T(t_1) \\ : \\ \dot{x}^T(t_m) \end{bmatrix}$$

(2.3) Creation of Library Function $\Theta(X)$.

$$\Theta(X) = \begin{bmatrix} | & | & | & .. & | & | & .. & | \\ 1 & X^{P_1} & X^{P_2} & .. & sin(X) & cos(X) & .. & \Phi(X) \\ | & | & | & .. & | & | & .. & | \end{bmatrix}$$

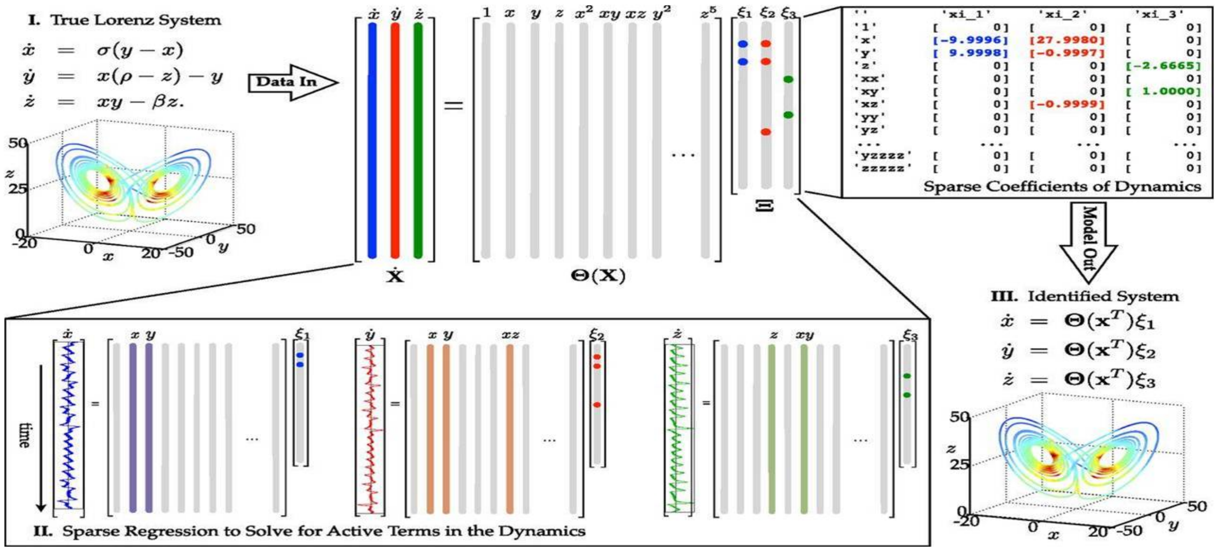
Here X^{P_i} are polynomials of the combinations of state variables with degree i. Notably, $\Phi(X)$ can be any function starting from logarithmic to exponential as per the choice of the algorithm designer.

(2.4) The matrix $\dot{X} = \Theta(X) \cdot \Xi$. Here Ξ represents a matrix such that each column i is ξ_i. So, $\Xi = [\xi_1 \ \xi_2 ... \xi_n]$. Here $\xi_i \in \mathbb{R}^l : l = No.\ of\ columns\ in\ \Theta(X)$.

(2.5) Each ξ_i is obtained by the least squares technique added with the regularised term.
$\xi_i = argmin_{\xi_i} \frac{1}{2} \mid\mid X_k - \dot{\xi_i}\theta(X) \mid\mid_2^2 + \lambda \mid\mid X \mid\mid_2$

(2.6) Now $\dot{x_k} = \Theta(x^T) \cdot \xi_k, \forall k \in [n]$.

SINDy in a nutshell:


I. True Lorenz System
$\dot{x} = \sigma(y - x)$
$\dot{y} = x(\rho - z) - y$
$\dot{z} = xy - \beta z.$
Data In
$\dot{\mathbf{X}}$
$\mathbf{\Theta}(\mathbf{X})$
Ξ
$1\ x\ y\ z\ x^2\ xy\ xz\ y^2\ z^5$
$\xi_1\ \xi_2\ \xi_3$
Model Out
III. Identified System
$\dot{x} = \mathbf{\Theta}(\mathbf{x}^T)\xi_1$
$\dot{y} = \mathbf{\Theta}(\mathbf{x}^T)\xi_2$
$\dot{z} = \mathbf{\Theta}(\mathbf{x}^T)\xi_3$
time
II. Sparse Regression to Solve for Active Terms in the Dynamics
Sparse Coefficients of Dynamics

```
''         'xi_1'      'xi_2'      'xi_3'
'1'      [      0]  [      0]  [      0]
'x'      [-9.9996]  [27.9980]  [      0]
'y'      [ 9.9998]  [-0.9997]  [      0]
'z'      [      0]  [      0]  [-2.6665]
'xx'     [      0]  [      0]  [      0]
'xy'     [      0]  [      0]  [ 1.0000]
'xz'     [      0]  [-0.9999]  [      0]
'yy'     [      0]  [      0]  [      0]
'yz'     [      0]  [      0]  [      0]
...            ...        ...        ...
'yzzzz'  [      0]  [      0]  [      0]
'zzzzz'  [      0]  [      0]  [      0]
```

3. MODEL IMPLEMENTATION DETAILS

(3.1) Implementation of our Machine Learning model has been done using python.

(3.2) Implementation of SINDy was aided by PySINDy library available in python.

(3.3) We have used two optimizer function for PySINDy, namely, Lasso and STLSQ optimizer functions.

(3.4) FourierLibrary and PolynomialLibrary were used for incorporating trigonometric and polynomial feature functions in Library matrix $\Theta(X)$.

(3.5) FourierLibrary uses lasso optimizer and the PolynomialLibrary uses STLSQ optimizer functions.

(3.6) PolynomialLibrary (degree): Using these functions, Polynomial of different columns of X are selected as the feature function. We have used it for up to degree = 3. (Owing to computational difficulty)

(3.7) FourierLibrary (n_dependencies) : Using these functions, combinations of sines and cosines of columns of X matrix is selected as feature function. IN the project, n_dependencies = degree = 3 is chosen.

4. MODEL TESTING

(4.1) Testing is done with two models:
(a) Polynomial feature with STLSQ.
(b) Fourier feature with lasso optimization.

(4.2) The one showing better overall performance is selected.

(4.3) Number of data points for testing = 200 at a time.

(4.4) For testing, an initial reference point is selected (at a certain timestamp) and using simulation 200 points are generated for different time instants.

(4.5) The machine learning model now predicts the respective values of all the features of the system for the same time instants.

(4.6) The predicted and simulated values are tallied to obtain the mean squared error and plot the graphs shown below..

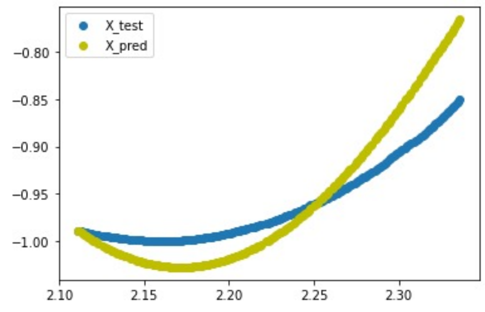
Some of the MSE values for both Fourier and Polynomial feature functions are shown in the table:

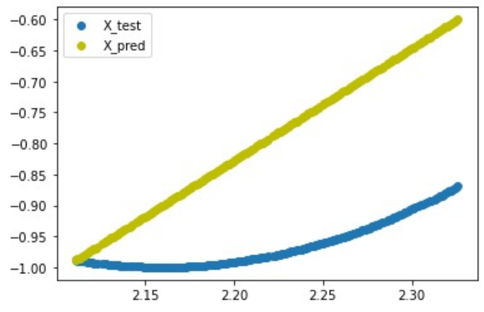
Feature Name	**Fourier with Lasso optimization**	**Polynomial with STLSQ**
Link 2 Orientation.y	0.000838	0.00151
Link 2 Orientation.w	0.000641	0.00381
Link 2 Angular Velocity	0.97651	0.010999
Link 3 Position.x	0.00023	0.009144
Link 3 Position.z	0.0077	0.00191
Link 3 Linear Velocity.x	0.69964	0.978994
Link 3 Linear Velocity.z	0.056568	0.002103
Link 3 Angular Velocity.y	0.052995	2.5916617

5. RESULTING PLOTS

Following are the comparison plots of the various feature columns of the robotic system between the Fourier and Polynomial methods:

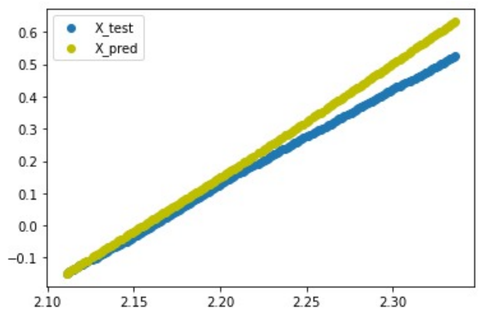
(5.1) Plots for Fourier and Polynomial feature functions for the feature "Orientation" along y-direction for Link-2:

Fourier: 

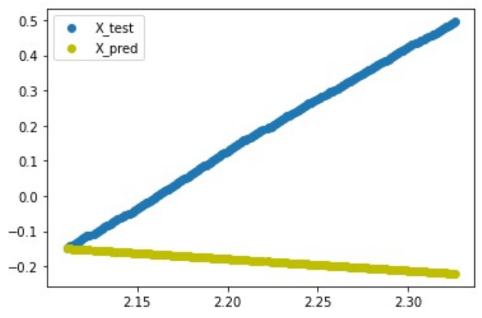
Polynomial: 

(5.2) Plots for Fourier and Polynomial feature functions for the feature "Orientation-w" for Link-2:

Fourier:

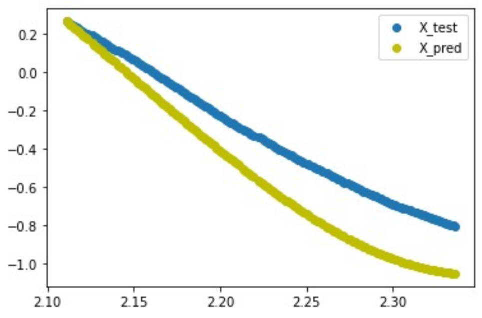


Polynomial:

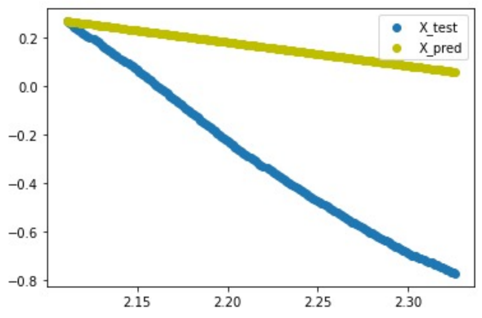


(5.3) Plots for Fourier and Polynomial feature functions for the feature "Position-x" for Link-3:

Fourier:

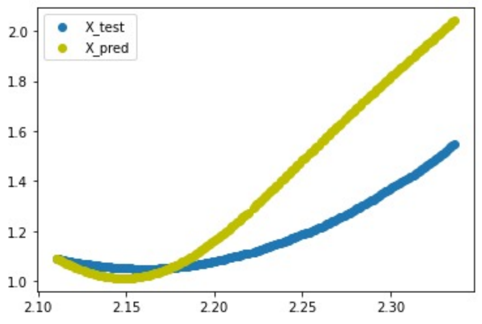


Polynomial:

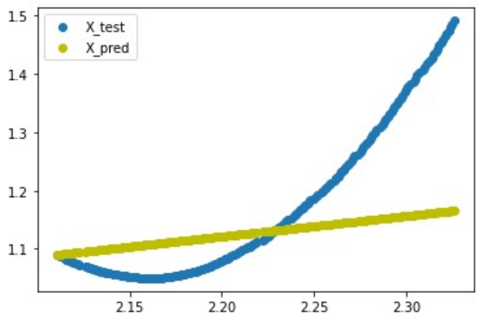


(5.4) Plots for Fourier and Polynomial feature functions for the feature "Position-z" for Link-3:

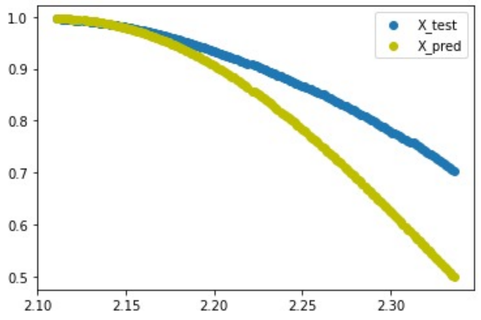
Fourier:

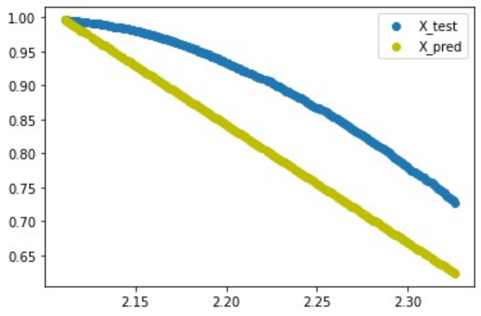


Polynomial:

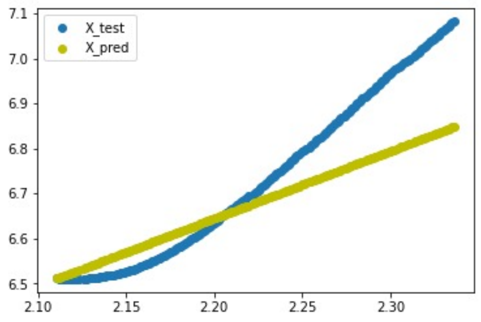


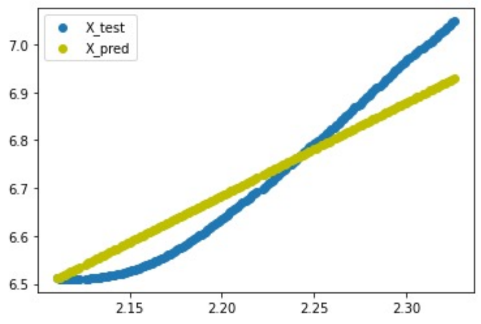
(5.5) Plots for Fourier and Polynomial feature functions for the feature "Orientation-y" for Link-3:

Fourier: 

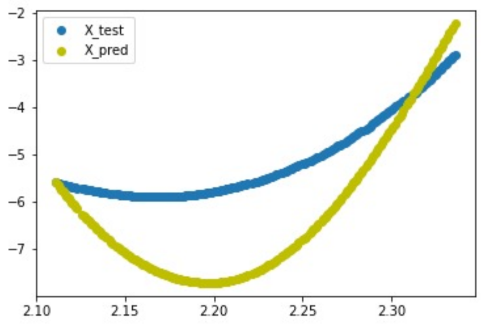
Polynomial: 

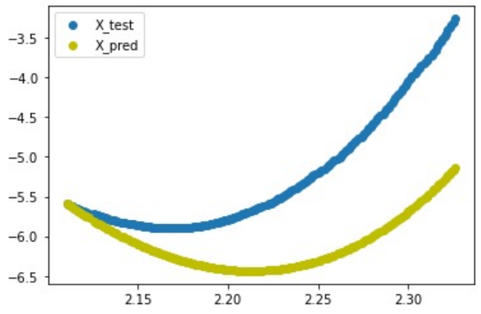
(5.6) Plots for Fourier and Polynomial feature functions for the feature "Angular velocity-y" for Link-2:

Fourier: 

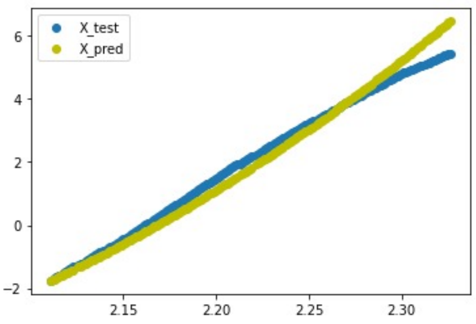
Polynomial: 

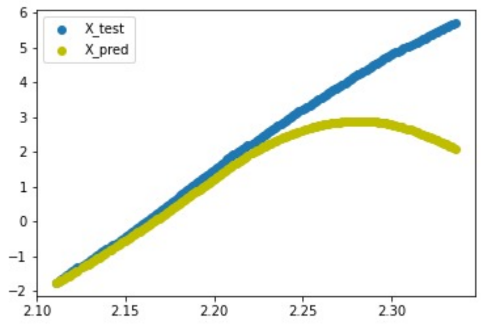
(5.7) Plots for Fourier and Polynomial feature functions for the feature "Linear Velocity along x-axis" for Link-3:

Fourier: 

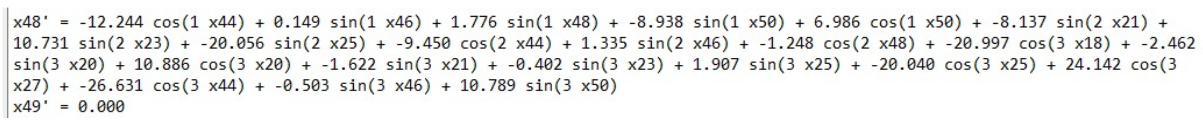
Polynomial: 

(5.8) Plots for Fourier and Polynomial feature functions for the feature "Linear Velocity along z direction" for Link-3:

Fourier: 

Polynomial: 

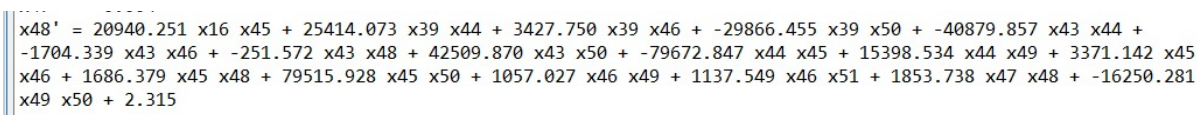
From the above few plots and relative MSE values, we can effectively conclude that using Fourier technique produces better approximation. Hence we go with Fourier feature function as our mathematical model for the robotic system.
Using SINDy we have also generated differential equations which closely captures the dynamics of the system. However the equations are very long. So producing all 53 of them would not be feasible.
An example of equation obtained through Fourier method is shown as:



```
x48' = -12.244 cos(1 x44) + 0.149 sin(1 x46) + 1.776 sin(1 x48) + -8.938 sin(1 x50) + 6.986 cos(1 x50) + -8.137 sin(2 x21) +
10.731 sin(2 x23) + -20.056 sin(2 x25) + -9.450 cos(2 x44) + 1.335 sin(2 x46) + -1.248 cos(2 x48) + -20.997 cos(3 x18) + -2.462
sin(3 x20) + 10.886 cos(3 x20) + -1.622 sin(3 x21) + -0.402 sin(3 x23) + 1.907 sin(3 x25) + -20.040 cos(3 x25) + 24.142 cos(3
x27) + -26.631 cos(3 x44) + -0.503 sin(3 x46) + 10.789 sin(3 x50)
x49' = 0.000
```

Here the terms include combinations of sines and cosines of state variables.

Example of equation obtained through Polynomial; method is as shown:



```
x48' = 20940.251 x16 x45 + 25414.073 x39 x44 + 3427.750 x39 x46 + -29866.455 x39 x50 + -40879.857 x43 x44 +
-1704.339 x43 x46 + -251.572 x43 x48 + 42509.870 x43 x50 + -79672.847 x44 x45 + 15398.534 x44 x49 + 3371.142 x45
x46 + 1686.379 x45 x48 + 79515.928 x45 x50 + 1057.027 x46 x49 + 1137.549 x46 x51 + 1853.738 x47 x48 + -16250.281
x49 x50 + 2.315
```

In this case all the terms are polynomial combinations of state variables.

The above equation (obtained from Fourier method) approximates the rate of change of the feature variable $x48$(Linear Velocity of Link-3 along z-direction) in terms of sines and cosines other feature variables. And empirically this turns out to be a good approximation (from MSE report and plots).

ACKNOWLEDGEMENTS

1. The theory has been derived from the paper by Brunton et al., named by:
"Discovering governing equations from data by sparse identification of nonlinear dynamical systems", published in 2016.

Here is a link to the paper : *Paper*.

2. Following are the references to the GAZEBO/ROS tutorials used for learning:
For gazebo tutorials.
For ROS tutorials.