**Name:** Alam Ali (NetID: aa8007)

**Course Name:** Deep Learning (ECE-GY-9123) (Course Project)

**Semester:** Spring 2021 semester

**Course Teacher:** Prof. Chinmay Hegde

**Submission Date:** 18-May-2021 (Tuesday)

**Team Member:** Alam Ali (Team member=1)

***Deliverable 4: Final Project report***

# Title: Comparison of Traditional Optimization and Deep Learning (ML) techniques in ACOPF solvers for obtaining ACOPF Optimal Solutions.

## Problem Description:

Use of machine learning to obtain solutions to AC optimal power flow has recently been a very active area of research due to the astounding speedups that result from bypassing traditional optimization techniques. In this project, we train a neural network to emulate an iterative solver in order to approximately iterate towards the optimum and obtain an overall AC-feasible solution. Results shown for networks up to IEEE500 bus network indicates that the proposed method is capable of finding feasible, near-optimal solutions to ACOPF problems in milliseconds on a laptop computer [1].

## Literature Survey:

ACOPF is a canonical power systems operation problem that is at the heart of optimizing large-scale power networks. Solving this problem quickly and efficiently has been the subject of decades of research. One particularly interesting development in solving these problems is the use of machine learning techniques, such as deep learning to obtain ACOPF solutions [2] [3]. We will propose a deep learning model which aims to emulate an ACOPF solver, and we observe positive results on the chosen networks in terms of optimality gap, speed, and convergence success. Deep Learning approaches have been proposed to recover the power systems states and to enhance solving optimal power flow problems [4].

**Figure 1** below shows the architecture of proposed NN method in operation phase. By using this framework, we can determine optimal power generation set-points on timescales that are appropriate for balancing fluctuations in renewable generation and load. For this NN model, we will have a NN with three (3) hidden layers with ReLU, tanh and linear activation functions.

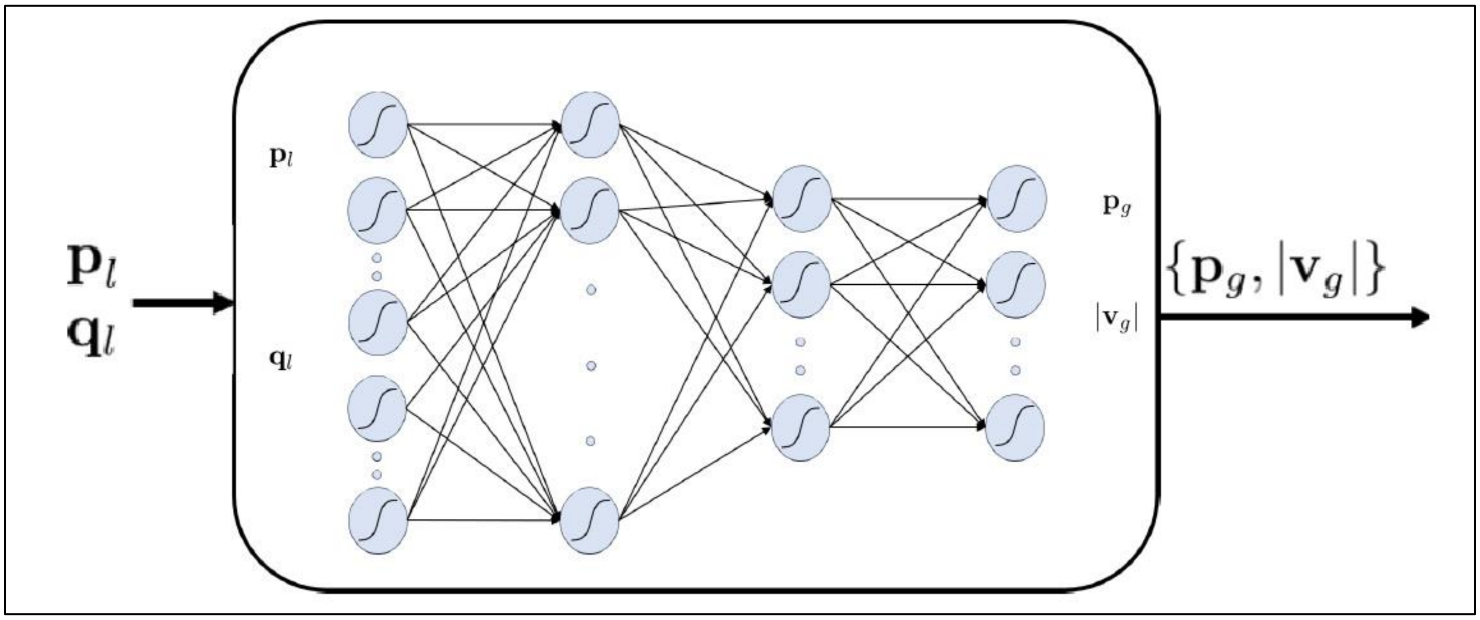


**Fig 1**: Proposed Deep Neural Network Machine Learning model.

## Motivation:

We present a machine learning based method to obtain feasible solutions to ACOPF problem with negligible optimality gaps with fast timescales bypassing solving an ACOPF altogether. This is motivated by the fact that as power grids experiences increasing amounts of renewable power generation, controllable loads, and other inverter interfaced devices, fast system dynamics and quick fluctuations in power supply are likely to occur [5].

Many efforts have focused on linearizing the problem in order to solve ACOPF on faster timescales. We leverage historical data to learn a mapping between system loads and optimal generation values, enabling us to find near-optimal and feasible ACOPF solutions on faster timescales without actually solving an optimization problem.

## ACOPF Optimization Problem:

ACOPF optimization problem can be formulated as follows:

$$\begin{aligned} & min_{\,v,\{p_g,n,q_g,n\},n\in G} && c(p_g) = \sum_{n\in G} c_n(p_{g,n}) \\ & subject\ to: && \underline{p_{g,n}} \le p_{g,n} \le \overline{p_{g,n}} \\ & && \underline{q_{g,n}} \le q_{g,n} \le \overline{q_{g,n}} \\ & && \underline{|v|} \le \mathrm{v} \le \overline{|v|} \\ & && \mathrm{h(v,p,q)} = 0 \end{aligned}$$

$where, variables\ \boldsymbol{p_{g,n}}\ \boldsymbol{and}\ \boldsymbol{q_{g,n}}\ denote\ the\ active\ and\ reactive\ power\ injections\ from\ generator\ installed$

$at\ bus\ \boldsymbol{n} \in \boldsymbol{G}, and\ \boldsymbol{v}\ represents\ voltage\ magnitude.\ \boldsymbol{p_g}\ is\ a\ vector\ which\ collects\ active\ power$

$injections\ from\ all\ generators\ and\ cost\ of\ power\ generation\ from\ generator\ at\ \boldsymbol{bus\ n\ is\ c_n(p_{g,n})}.$

$All\ non-linear\ power\ flow\ equations\ are\ collected\ in\ \boldsymbol{h(v,p,q)}.$

The objective of this proposed data-driven approach is to learn an underlying ACOPF mapping between the load demands at all load buses and all generators active power and voltage magnitude set points.

**Approximate mapping between Loads and the Optimal Generation Solution:**

Because the solution of ACOPF is on boundary of feasibility set, we aim at ensuring that NN outputs are in the interior of ACOPF feasibility set [6]. We generate training by solving the following restricted R-ACOPF problem.

$$\begin{aligned} & min_{\, v,\{p_g,n,q_g,n\},n\in G} && c(p_g) = \sum_{n\in G} c_n(p_{g,n}) \\ & subject\ to: && \underline{p_{g,n}} \le p_{g,n} \le \overline{p_{g,n}} \\ & && \underline{q_{g,n}} \le q_{g,n} \le \overline{q_{g,n}} \\ & && \underline{|v|} + \lambda \le \mathrm{v} \le \overline{|v|} - \lambda \\ & && \mathrm{h(v,p,q)} = 0 \end{aligned}$$

Training samples generated by solving above problem are strictly in the interior of the voltage magnitudes feasibility set. Any bounded deviations in the learned mapping are expected to remain within the voltage limits. Parameter $\boldsymbol{\lambda}$ is an algorithmic tuning parameter that addresses the optimality and feasibility trade-off.

**Description of Dataset:**

For this project, we have generated IEEE 14-bus data, IEEE 30-bus data and IEEE-500 bus data (big data files) in excel sheet using MATPOWER MIPS solver. We loaded these excel sheets in Jupiter Notebook to test our neural network (NN) algorithm, and generate results and average problem solve times for ACOPF problem. Dataset (accessed by MATPOWER .mpc struct) comprising huge amount of data of more than 16,000 data points is attached in final report.

MATPOWER Interior Point Solver (MIPS) was used to generate the data and was used as the baseline for comparison with NN model. A single training sample consists of the pair $[\boldsymbol{x^k}, \boldsymbol{x^{k+1}}]$ obtained from solver.

## Description of Neural Network (NN) Model:

**Figure 2** below shows a Neural Network model layout and an overview of the testing phase of algorithm.

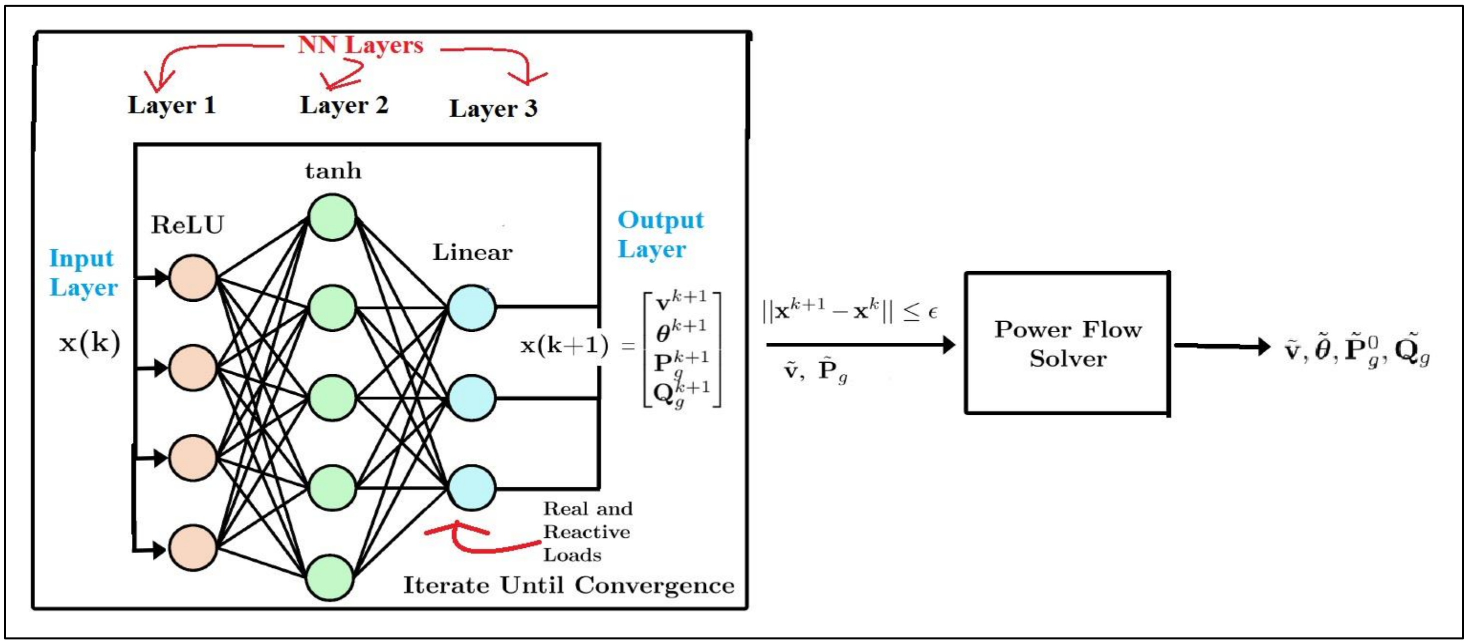

NN Layers
Layer 1
Layer 2
Layer 3
tanh
ReLU
Linear
Input Layer
Output Layer
x(k)
$x(k+1) = \begin{bmatrix} v^{k+1} \\ \theta^{k+1} \\ P_g^{k+1} \\ Q_g^{k+1} \end{bmatrix}$
Real and Reactive Loads
Iterate Until Convergence
$\|x^{k+1} - x^k\| \leq \epsilon$
$\tilde{v}, \tilde{P}_g$
Power Flow Solver
$\tilde{v}, \tilde{\theta}, \tilde{P}_g^0, \tilde{Q}_g$

**Fig 2:** Model of Neural Network to approximate fast iterations towards the optimum power flow solution.

We will use Deep Learning based model $\boldsymbol{F_R}(.)$ that takes $\boldsymbol{x^k}$ as an input and returns $\boldsymbol{x^{k+1}}$ as an output, such that

$$\boldsymbol{x^{k+1} = F_R(x^k),}$$

$$\boldsymbol{where, \qquad vector\ x^k = [v^k, \theta^k, P_g^k, Q_g^k]^T}$$

## Description of Loss Function:

For our model, we will train and test IEEE-14, IEEE-30 and IEEE 500 power bus data using two optimizers and the loss function used will be Mean square error (mse). One optimizer is Adam and second one is RMSprop.

## Training details:

For the final report, Deep learning (NN) based method to solve OPF problem will be compared with conventional DCOPF. DCOPF does not produce an AC feasible solution, but it gives a good approximation for ACOPF and is used in many LMP-based markets to calculate prices and hence provides an interesting comparison between learning methods and traditional optimization techniques [1]. A single training sample consists of the pair $\boldsymbol{x^k\ and\ x^{k+1}}$ obtained from the solver. Tolerance $\boldsymbol{\varepsilon}$ of learning-based solver was set to $\mathbf{10^{-4}}$, where convergence is reached when $\left|\left|\boldsymbol{x^{k+1} - x^k}\right|\right| \leq \boldsymbol{\epsilon}.$

## Neural Network Problem formulation:

This problem is a general nonconvex optimization problem with n-dimensional optimization variable vector $\boldsymbol{x}$, cost function $\boldsymbol{f}(.): \boldsymbol{R}^{\boldsymbol{n}} \rightarrow \boldsymbol{R}$, M equality constraints $\boldsymbol{g}_{\boldsymbol{i}}(\boldsymbol{x}) = \boldsymbol{0}$ and P inequality constraints $\boldsymbol{h}_{\boldsymbol{j}}(\boldsymbol{x}) \leq \boldsymbol{0}$ as:

$$\boldsymbol{min}_{\boldsymbol{x}} \;\; \boldsymbol{f}(\boldsymbol{x})$$

$$\boldsymbol{s.t} \qquad \boldsymbol{g}_{\boldsymbol{i}}(\boldsymbol{x}) = \boldsymbol{0}, \qquad \boldsymbol{for\ i} = \boldsymbol{1}, \boldsymbol{2}, \ldots, \boldsymbol{M}$$

$$\boldsymbol{h}_{\boldsymbol{j}}(\boldsymbol{x}) \leq \boldsymbol{0}, \qquad \boldsymbol{for\ j} = \boldsymbol{1}, \boldsymbol{2}, \ldots, \boldsymbol{P}$$

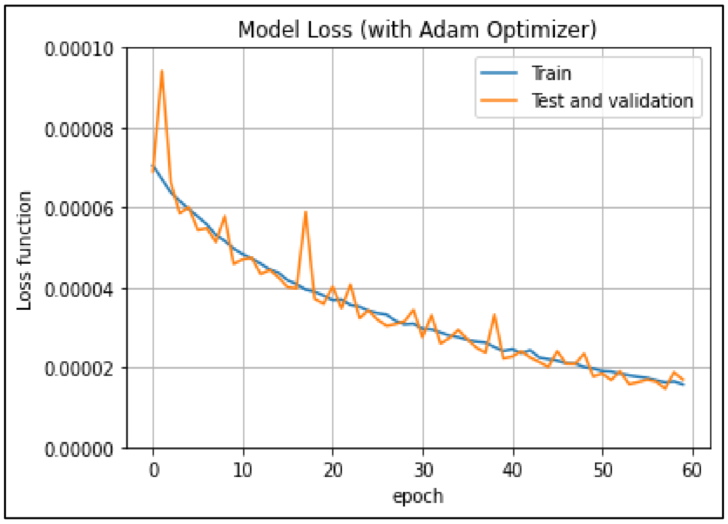
Instead of using Lagrangian functions or forming Hessian matrices, learning-based method uses a deep learning model $\boldsymbol{F}_{\boldsymbol{R}}(.): \boldsymbol{R}^{\boldsymbol{n}} \rightarrow \boldsymbol{R}^{\boldsymbol{n}}$ that takes in $\boldsymbol{x}^{\boldsymbol{k}}$ as an input and provides $\boldsymbol{x}^{\boldsymbol{k+1}}$ as an output, written below as:

$$\boldsymbol{x}^{\boldsymbol{k+1}} = \boldsymbol{F}(\boldsymbol{x}^{\boldsymbol{k}})$$

Proposed model in this project is comprised of a fully connected three-layer neural network (NN) with feedback, where input $\boldsymbol{x}^{\boldsymbol{k}}$ is the candidate optimal solution vector at iteration $k$. This model iteratively uses feedback from the output layer $\boldsymbol{x}^{\boldsymbol{k+1}}$ to the input layer $\boldsymbol{x}^{\boldsymbol{k}}$ until convergence occurs $\left|\left|\boldsymbol{x}^{\boldsymbol{k+1}} - \boldsymbol{x}^{\boldsymbol{k}}\right|\right| \leq \boldsymbol{\varepsilon}$.

## Preliminary Results (Initial Testing):

I have tested this data with IEEE 14-bus and IEEE 30-bus power system. Model Loss functions with Adam Optimizer and RMSprop Optimizer have been found, and are plotted below. Model loss is lesser with Adam Optimizer, keeping in mind that same hyperparamters have been selected for tuning both networks.



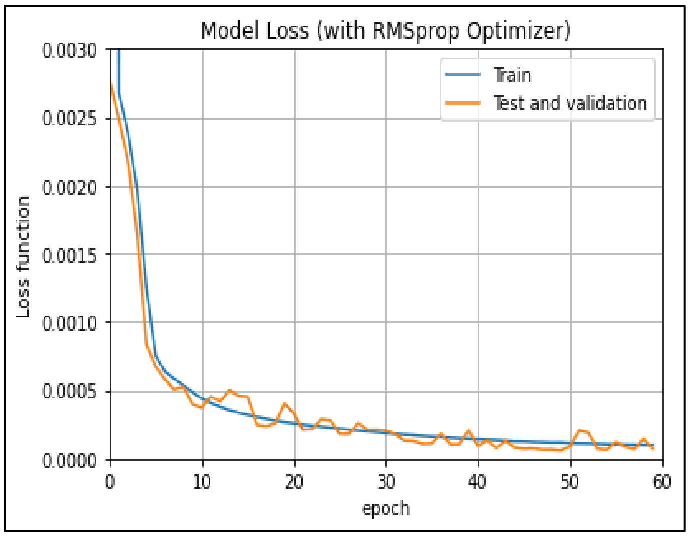


**Fig 3**: Model Loss with Adam Optimizer. **Fig 4**: Model Loss with RMSprop Optimizer.

**Simulation Results:**

**Table 1** below shows the comparison of average solve time of Deep Learning-based method ('NN') with two other cases (ACOPF flat started and DCOPC) across the 500-sample training set.

**Table 1**: Deep Learning solution finds ACOPF solutions faster than traditional OPF methods.

| Power network type | OPF-type | Avg problem solve time (sec) |
|---|---|---|
| IEEE 14-bus | **ACOPF with flat start** (traditional method) | **0.41s** |
| | **DCOPF** (traditional method) | **0.29s** |
| | **NN with PF** (deep learning-based) | **0.08s** |
| IEEE 30-bus | **ACOPF with flat start** (traditional method) | **0.68s** |
| | **DCOPF** (traditional method) | **0.51s** |
| | **NN with PF** (deep learning-based) | **0.12s** |
| IEEE 500-bus | **ACOPF with flat start** (traditional method) | **3.24s** |
| | **DCOPF** (traditional method) | **1.61s** |
| | **NN with PF** (deep learning-based) | **0.24s** |

**Table 2** below shows the optimality gap (difference in cost function value) for the deep-learning ("NN") based OPF solution and traditional DCOPF solution.

**Table 2**: Optimality Gaps for deep-learning based method and DCOPF model.

| Power network type | Average Gap for Neural Network case | Average Gap for DCOPF case |
|---|---|---|
| IEEE 14-bus | **0.12%** | **1.53%** |
| IEEE 30-bus | **0.23%** | **2.08%** |
| IEEE 500-bus | **2.61%** | **10.05%** |

## Project Outcomes:

Results show extremely fast convergence times for NN based method as compared to traditional DCOPC method.

Table comparison is done that shows learning-based method ("NN") is compared with 2 other cases across the 500-sample training set. We showed that for larger networks, the learning-based solution finds approximate solutions to ACOPF faster than it takes to find a solution to DCOPF, with a smaller optimality gap than DCOPF provides, and without the AC infeasibility of DCOPF.

## Discussion about possible challenges:

Generating the code in Tensor flow was quite a challenge, because we had to learn OPF data modeling in python to generate this optimization for NN and DCOPF power flow models. It is observed that generating a diverse and representative dataset is an important thrust of research within learning-based OPF methods.

ACOPF is one of the most commonly solved problems in power systems. The current emphasis on integration of renewable sources and power electronics into modern grids, has increased the complexity of power flow calculations. Obtaining a feasible ACOPF solution in large-scale power systems is a very big challenge. Existence of multiple solutions causes certain initial points to lead to non-feasible solutions in which some operating network constraints are violated, hence NN network solves this problem.

## Reflections on Results:

We learn a model that can predict an accurate solution over a fixed grid topology and constraints set, which further gives some measure of consistency in the solution space, meaning similar load distributions should correspond to similar generator settings. Neural networks have demonstrated the ability to model extremely complicated non-convex functions, making them highly attractive for this setting. A model can be trained off-line on historic data and used in real-time to make predictions on an optimal power setting [6].

Only a subset of variables was sent to the power flow solver because of huge amount of training data, but the ML model utilized information about the entire OPF solution for better informing the neural network model for iterating it towards the optimum solution [1]. Directions of future work include development of datasets or dataset generation methods for learning-based OPF, inclusion of additional constraints such as line flow limits, and speed or accuracy comparisons with other relaxations or convexifications.

## Conclusions:

This research work provides a learning-based approximation for solving ACOPF problem. Initial results show that this method can achieve very fast convergence speeds with minimal optimality gaps, even converging faster and with more accuracy than DCOPF on large neural networks. Hence, a larger neural network with more training data and more hyperparamters tuning may improve these results even further. Black-box models (such as NN) do not offer grid operators much insight in decision-making. A combination of greater understanding of these models and increased failsafes can help expedite the potential use of these NN models in actual operation.

## References:

[1] Baker, K., “Emulating AC OPF solvers for Obtaining Sub-second Feasible, Near-Optimal Solutions”, https://arxiv.org/abs/2012.10031, 2020arXiv201210031B, 2020.

[2] A. Zamzam and K. Baker, “Learning optimal solutions for extremely fast AC optimal power flow,” in IEEE SmartGridComm, December 2020, available at: https://arxiv.org/abs/2861719.

[3] X. Pan, M. Chen, T. Zhao, and S. Low, “Deep OPF: A feasibility optimized deep neural network approach for AC optimal power flow problems,” arXiv preprint arXiv:2007.0100, July 2020.

[4] A. S. Zamzam, X. Fu, and N. D. Sidiropoulos, “Data-driven learning based optimization for distribution system state estimation,” IEEE Transactions on Power Systems (to appear), 2019.

[5] Y. Tang, K. Dvijotham, and S. Low, “Real-time optimal power flow,” IEEE Transactions on Smart Grid, vol. 8, no. 6, pp. 2963–2973, Nov 2017.

[6] D. K. Molzahn, S. S. Baghsorkhi, and I. A. Hiskens, “Semidefinite relaxations of equivalent optimal power flow problems: An illustrative example,” in IEEE International Symposium on Circuits and Systems (ISCAS), May 2015, pp. 1887–1890.

## Code:

Code on Jupyter notebook and Dataset (first two pages) follows on next pages.

```
# Project code for AC OPF solvers

from pandas import read_csv
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import MinMaxScaler
import tensorflow as tf
from tensorflow.python.keras.models import Sequential
from tensorflow.python.keras.layers import Dense
from tensorflow.python.keras.layers import LeakyReLU
from tensorflow.python.keras.layers import BatchNormalization
from tensorflow.python.keras.wrappers.scikit_learn import KerasRegressor
from keras.constraints import max_norm

# load dataset

system = "14"

dataframeX = read_csv("/content/drive/MyDrive/14result_overall_in.csv", delimit
X = dataframeX.values[0:]
dataframeY = read_csv("/content/drive/MyDrive/14result_overall_out.csv", delimi
Y = dataframeY.values

# split into input (X) and output (Y) variables
print (len(X))
print (X.shape)

print (len(Y))
print (Y.shape)

nsamples = X.shape[0]
npredictors = X.shape[1]
noutvars = Y.shape[1]
print (npredictors)
print (noutvars)
#print (X[1])
```

```
70677
(70677, 60)
70677
(70677, 38)
```

```
60
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, c
```

```
model = Sequential() # do this every time to reset the model!
model.add(Dense(npredictors, input_dim=npredictors, kernel_initializer='normal'
model.add(Dense(10000, activation='relu'))
model.add(Dense(noutvars, activation='linear'))#,kernel_constraint=max_norm(3),

model.summary()
opt = tf.keras.optimizers.Adam(learning_rate=0.00007)
model.compile(loss='mse', optimizer=opt, metrics=['mse','mae'])
```

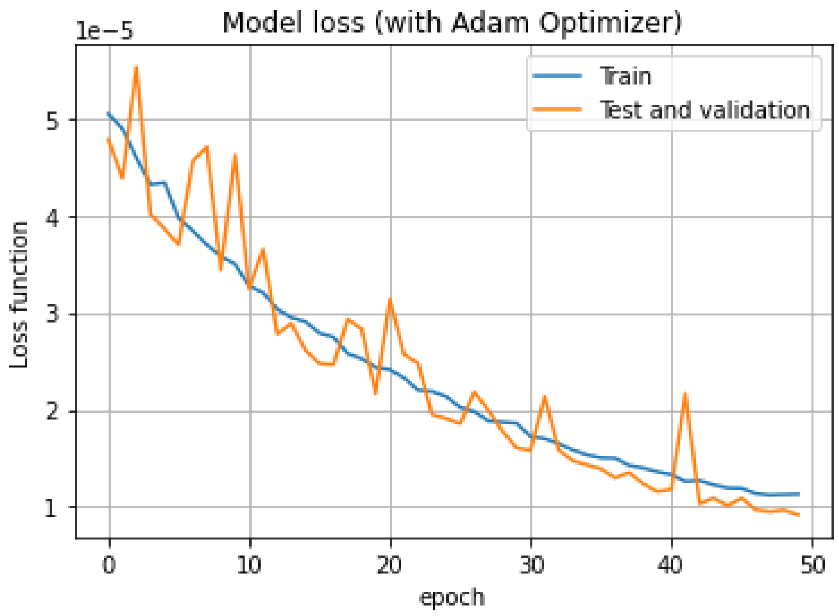
```
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_16 (Dense)             (None, 60)                3660
_________________________________________________________________
dense_17 (Dense)             (None, 10000)             610000
_________________________________________________________________
dense_18 (Dense)             (None, 38)                380038
=================================================================
Total params: 993,698
Trainable params: 993,698
Non-trainable params: 0
_________________________________________________________________
```

```
history = model.fit(X, Y, epochs=50, batch_size=64,  verbose=1, validation_spli
```

```
model.save(system+'busNN.h5')
```

```
print(history.history.keys())
# "Loss"
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss (with Adam Optimizer)')
plt.ylabel('Loss function')
plt.xlabel('epoch')
plt.legend(['Train', 'Test and validation'], loc='upper right')
plt.grid()
plt.show()
```

```
dict_keys(['loss', 'mse', 'mae', 'val_loss', 'val_mse', 'val_mae'])
```



```
model = Sequential() # do this every time to reset the model!
model.add(Dense(npredictors, input_dim=npredictors, kernel_initializer='normal'
model.add(Dense(10000, activation='relu'))
model.add(Dense(noutvars, activation='linear'))#,kernel_constraint=max_norm(3),

model.summary()
opt = tf.keras.optimizers.RMSprop(learning_rate=0.00007)
model.compile(loss='mse', optimizer=opt, metrics=['mse','mae'])
```

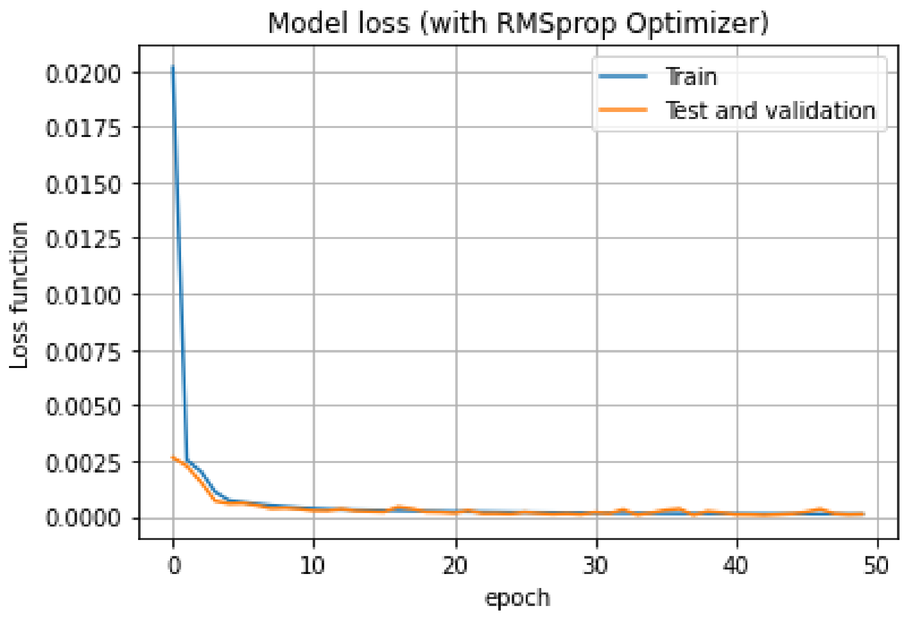
```
Model: "sequential_8"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_22 (Dense)             (None, 60)                3660
_________________________________________________________________
dense_23 (Dense)             (None, 10000)             610000
_________________________________________________________________
dense_24 (Dense)             (None, 38)                380038
=================================================================
Total params: 993,698
Trainable params: 993,698
Non-trainable params: 0
_________________________________________________________________
```

```
history = model.fit(X, Y, epochs=50, batch_size=64,  verbose=1, validation_spli
```

```
model.save(system+'busNN.h5')
```

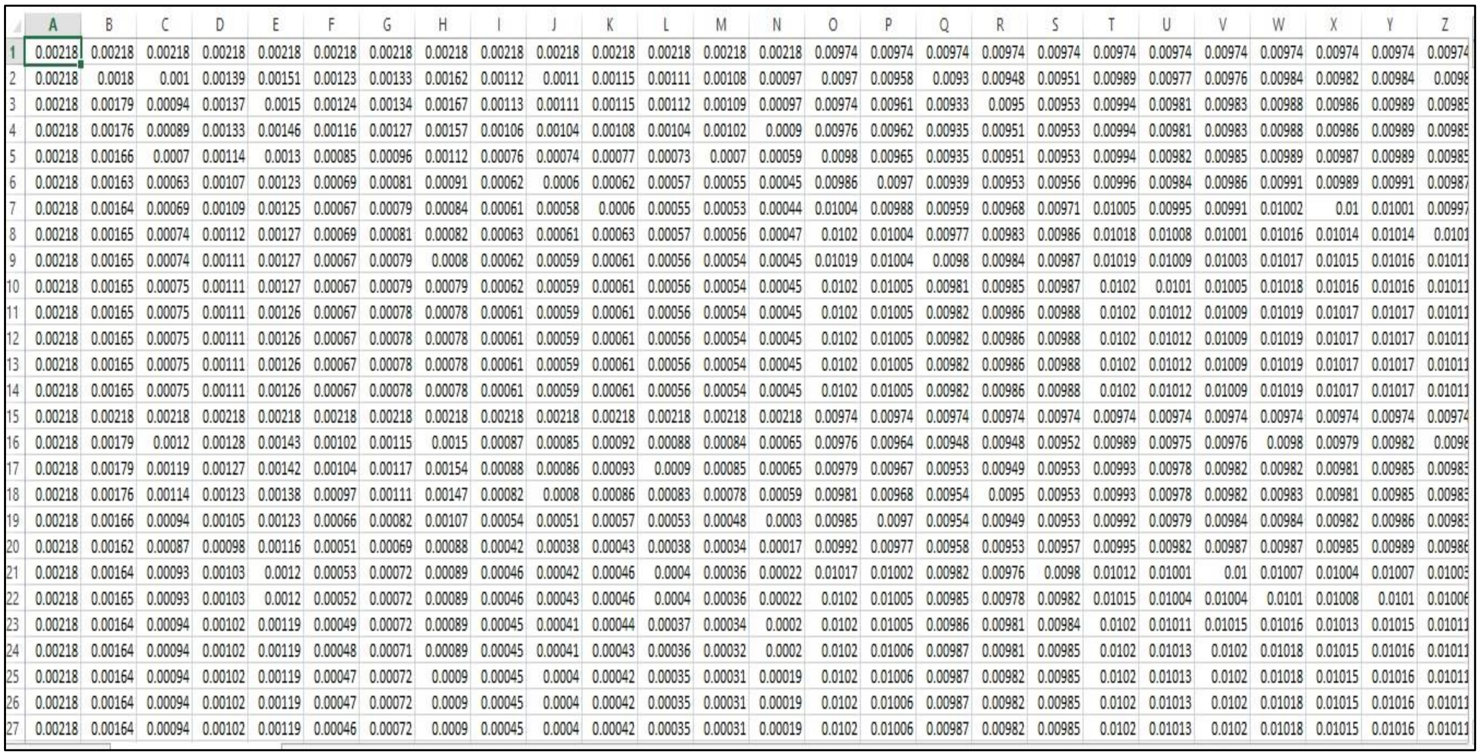
```
print(history.history.keys())
# "Loss"
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss (with RMSprop Optimizer)')
plt.ylabel('Loss function')
plt.xlabel('epoch')
plt.legend(['Train', 'Test and validation'], loc='upper right')
plt.grid()
plt.show()
```

```
dict_keys(['loss', 'mse', 'mae', 'val_loss', 'val_mse', 'val_mae'])
```

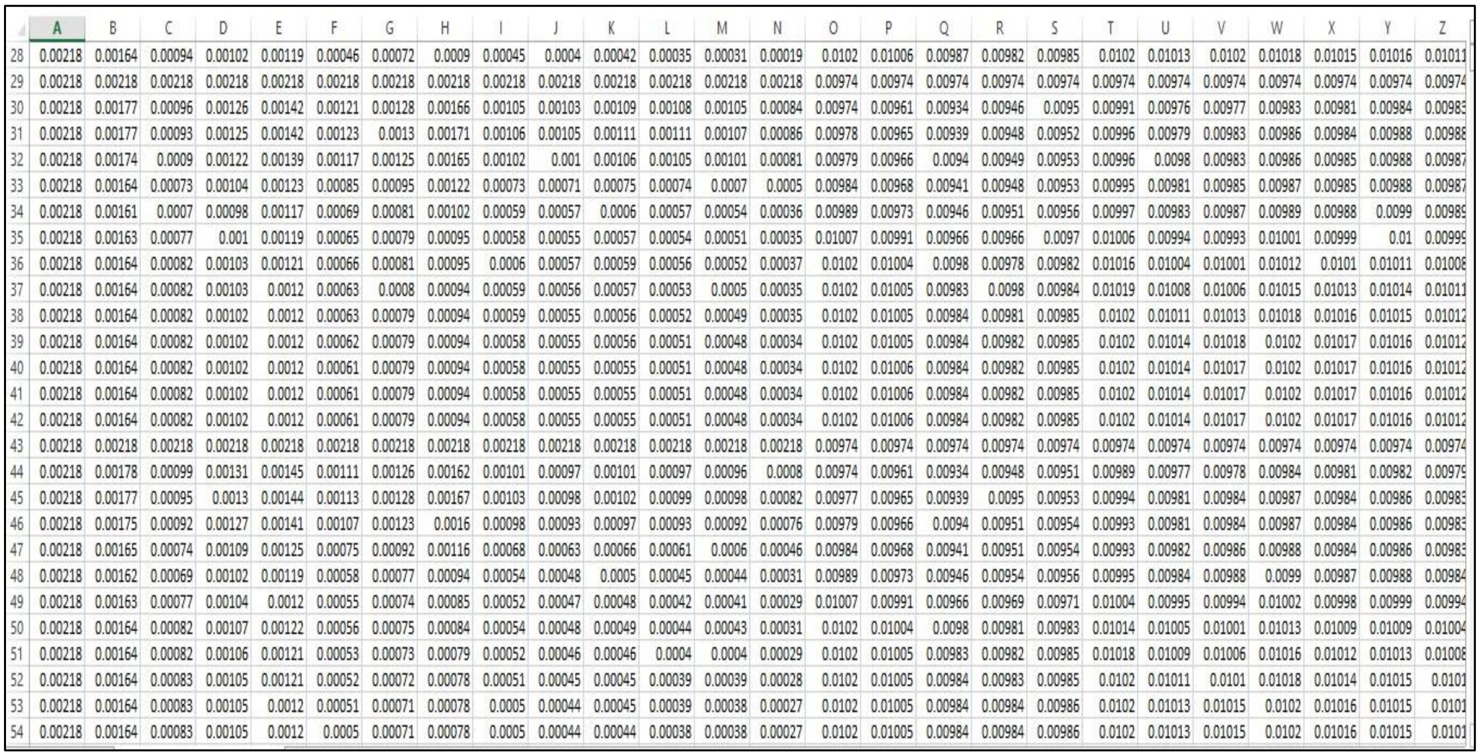


Name: Alam Ali, Net ID: aa8007, DL course project (Final report)
DL dataset (x(k) and x(k+1)) is given s.t x(k+1)=F(x(k))
This is a huge amount of data (more than 16,000 data points). So, I have just attached first two pages of data.

(1) x(k) is an input data to Deep Learning program.

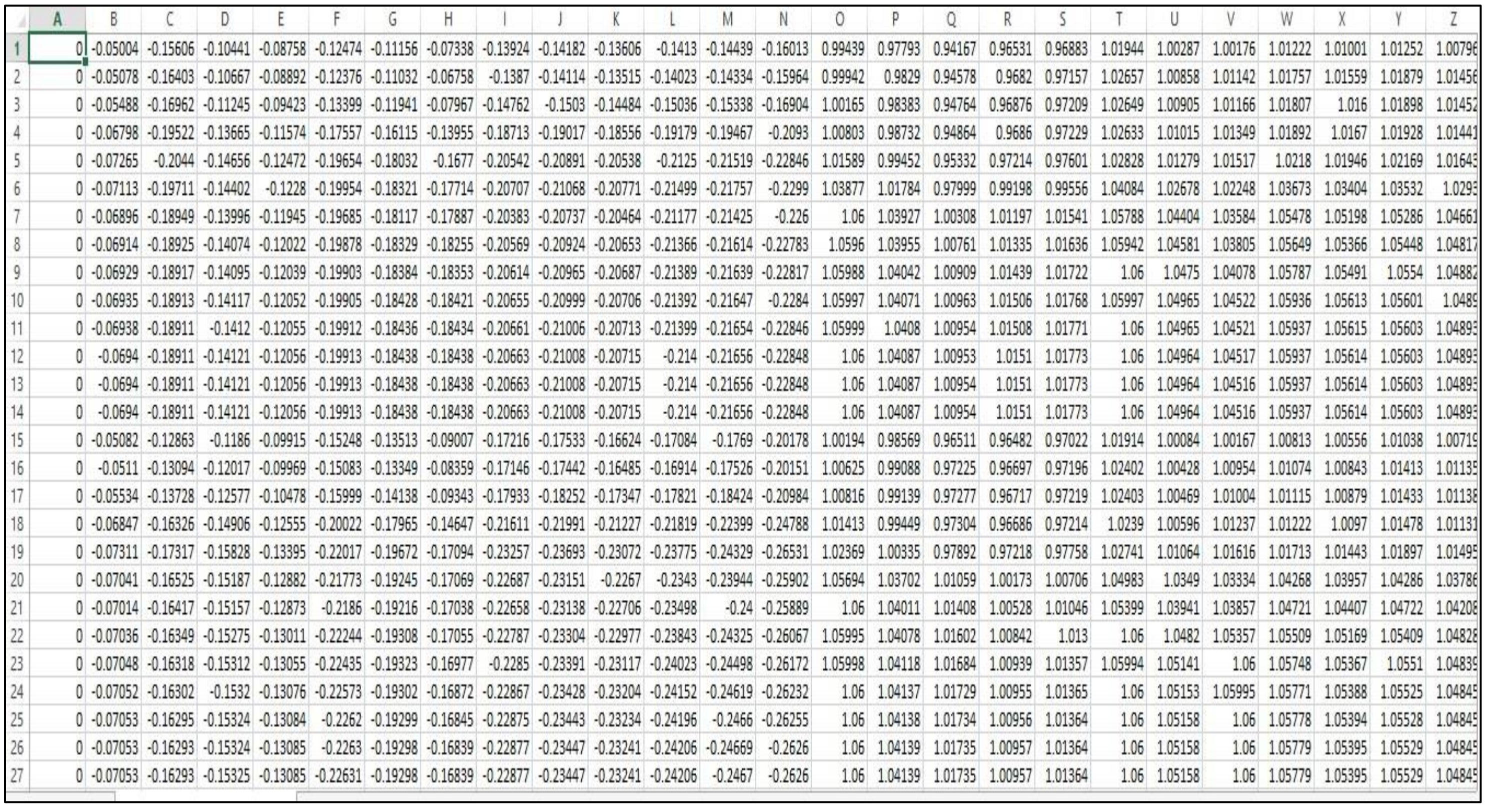


| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.00218 | 0.00218 | 0.00218 | 0.00218 | 0.00218 | 0.00218 | 0.00218 | 0.00218 | 0.00218 | 0.00218 | 0.00218 | 0.00218 | 0.00218 | 0.00218 | 0.00974 | 0.00974 | 0.00974 | 0.00974 | 0.00974 | 0.00974 | 0.00974 | 0.00974 | 0.00974 | 0.00974 | 0.00974 | 0.00974 |
| 2 | 0.00218 | 0.0018 | 0.001 | 0.00139 | 0.00151 | 0.00123 | 0.00133 | 0.00162 | 0.00112 | 0.0011 | 0.00115 | 0.00111 | 0.00108 | 0.00097 | 0.0097 | 0.00958 | 0.0093 | 0.00948 | 0.00951 | 0.00989 | 0.00977 | 0.00976 | 0.00984 | 0.00982 | 0.00984 | 0.0098 |
| 3 | 0.00218 | 0.00179 | 0.00094 | 0.00137 | 0.0015 | 0.00124 | 0.00134 | 0.00167 | 0.00113 | 0.00111 | 0.00115 | 0.00112 | 0.00109 | 0.00097 | 0.00974 | 0.00961 | 0.00933 | 0.0095 | 0.00953 | 0.00994 | 0.00981 | 0.00983 | 0.00988 | 0.00986 | 0.00989 | 0.00985 |
| 4 | 0.00218 | 0.00176 | 0.00089 | 0.00133 | 0.00146 | 0.00116 | 0.00127 | 0.00157 | 0.00106 | 0.00104 | 0.00108 | 0.00104 | 0.00102 | 0.0009 | 0.00976 | 0.00962 | 0.00935 | 0.00951 | 0.00953 | 0.00994 | 0.00981 | 0.00983 | 0.00988 | 0.00986 | 0.00989 | 0.00985 |
| 5 | 0.00218 | 0.00166 | 0.0007 | 0.00114 | 0.0013 | 0.00085 | 0.00096 | 0.00112 | 0.00076 | 0.00074 | 0.00077 | 0.00073 | 0.0007 | 0.00059 | 0.0098 | 0.00965 | 0.00935 | 0.00951 | 0.00953 | 0.00994 | 0.00982 | 0.00985 | 0.00989 | 0.00987 | 0.00989 | 0.00985 |
| 6 | 0.00218 | 0.00163 | 0.00063 | 0.00107 | 0.00123 | 0.00069 | 0.00081 | 0.00091 | 0.00062 | 0.0006 | 0.00062 | 0.00057 | 0.00055 | 0.00045 | 0.00986 | 0.0097 | 0.00939 | 0.00953 | 0.00956 | 0.00996 | 0.00984 | 0.00986 | 0.00991 | 0.00989 | 0.00991 | 0.00987 |
| 7 | 0.00218 | 0.00164 | 0.00069 | 0.00109 | 0.00125 | 0.00067 | 0.00079 | 0.00084 | 0.00061 | 0.00058 | 0.0006 | 0.00055 | 0.00053 | 0.00044 | 0.01004 | 0.00988 | 0.00959 | 0.00968 | 0.00971 | 0.01005 | 0.00995 | 0.00991 | 0.01002 | 0.01 | 0.01001 | 0.00997 |
| 8 | 0.00218 | 0.00165 | 0.00074 | 0.00112 | 0.00127 | 0.00069 | 0.00081 | 0.00082 | 0.00063 | 0.00061 | 0.00063 | 0.00057 | 0.00056 | 0.00047 | 0.0102 | 0.01004 | 0.00977 | 0.00983 | 0.00986 | 0.01018 | 0.01008 | 0.01001 | 0.01016 | 0.01014 | 0.01014 | 0.0101 |
| 9 | 0.00218 | 0.00165 | 0.00074 | 0.00111 | 0.00127 | 0.00067 | 0.00079 | 0.0008 | 0.00062 | 0.00059 | 0.00061 | 0.00056 | 0.00054 | 0.00045 | 0.01019 | 0.01004 | 0.0098 | 0.00984 | 0.00987 | 0.01019 | 0.01009 | 0.01003 | 0.01017 | 0.01015 | 0.01016 | 0.01011 |
| 10 | 0.00218 | 0.00165 | 0.00075 | 0.00111 | 0.00127 | 0.00067 | 0.00079 | 0.00079 | 0.00062 | 0.00059 | 0.00061 | 0.00056 | 0.00054 | 0.00045 | 0.0102 | 0.01005 | 0.00981 | 0.00985 | 0.00987 | 0.0102 | 0.0101 | 0.01005 | 0.01018 | 0.01016 | 0.01016 | 0.01011 |
| 11 | 0.00218 | 0.00165 | 0.00075 | 0.00111 | 0.00126 | 0.00067 | 0.00078 | 0.00078 | 0.00061 | 0.00059 | 0.00061 | 0.00056 | 0.00054 | 0.00045 | 0.0102 | 0.01005 | 0.00982 | 0.00986 | 0.00988 | 0.0102 | 0.01012 | 0.01009 | 0.01019 | 0.01017 | 0.01017 | 0.01011 |
| 12 | 0.00218 | 0.00165 | 0.00075 | 0.00111 | 0.00126 | 0.00067 | 0.00078 | 0.00078 | 0.00061 | 0.00059 | 0.00061 | 0.00056 | 0.00054 | 0.00045 | 0.0102 | 0.01005 | 0.00982 | 0.00986 | 0.00988 | 0.0102 | 0.01012 | 0.01009 | 0.01019 | 0.01017 | 0.01017 | 0.01011 |
| 13 | 0.00218 | 0.00165 | 0.00075 | 0.00111 | 0.00126 | 0.00067 | 0.00078 | 0.00078 | 0.00061 | 0.00059 | 0.00061 | 0.00056 | 0.00054 | 0.00045 | 0.0102 | 0.01005 | 0.00982 | 0.00986 | 0.00988 | 0.0102 | 0.01012 | 0.01009 | 0.01019 | 0.01017 | 0.01017 | 0.01011 |
| 14 | 0.00218 | 0.00165 | 0.00075 | 0.00111 | 0.00126 | 0.00067 | 0.00078 | 0.00078 | 0.00061 | 0.00059 | 0.00061 | 0.00056 | 0.00054 | 0.00045 | 0.0102 | 0.01005 | 0.00982 | 0.00986 | 0.00988 | 0.0102 | 0.01012 | 0.01009 | 0.01019 | 0.01017 | 0.01017 | 0.01011 |
| 15 | 0.00218 | 0.00218 | 0.00218 | 0.00218 | 0.00218 | 0.00218 | 0.00218 | 0.00218 | 0.00218 | 0.00218 | 0.00218 | 0.00218 | 0.00218 | 0.00218 | 0.00974 | 0.00974 | 0.00974 | 0.00974 | 0.00974 | 0.00974 | 0.00974 | 0.00974 | 0.00974 | 0.00974 | 0.00974 | 0.00974 |
| 16 | 0.00218 | 0.00179 | 0.0012 | 0.00128 | 0.00143 | 0.00102 | 0.00115 | 0.0015 | 0.00087 | 0.00085 | 0.00092 | 0.00088 | 0.00084 | 0.00065 | 0.00976 | 0.00964 | 0.00948 | 0.00948 | 0.00952 | 0.00989 | 0.00975 | 0.00976 | 0.0098 | 0.00979 | 0.00982 | 0.0098 |
| 17 | 0.00218 | 0.00179 | 0.00119 | 0.00127 | 0.00142 | 0.00104 | 0.00117 | 0.00154 | 0.00088 | 0.00086 | 0.00093 | 0.0009 | 0.00085 | 0.00065 | 0.00979 | 0.00967 | 0.00953 | 0.00949 | 0.00953 | 0.00993 | 0.00978 | 0.00982 | 0.00982 | 0.00981 | 0.00985 | 0.00983 |
| 18 | 0.00218 | 0.00176 | 0.00114 | 0.00123 | 0.00138 | 0.00097 | 0.00111 | 0.00147 | 0.00082 | 0.0008 | 0.00086 | 0.00083 | 0.00078 | 0.00059 | 0.00981 | 0.00968 | 0.00954 | 0.0095 | 0.00953 | 0.00993 | 0.00978 | 0.00982 | 0.00983 | 0.00981 | 0.00985 | 0.00983 |
| 19 | 0.00218 | 0.00166 | 0.00094 | 0.00105 | 0.00123 | 0.00066 | 0.00082 | 0.00107 | 0.00054 | 0.00051 | 0.00057 | 0.00053 | 0.00048 | 0.0003 | 0.00985 | 0.0097 | 0.00954 | 0.00949 | 0.00953 | 0.00992 | 0.00979 | 0.00984 | 0.00984 | 0.00982 | 0.00986 | 0.00983 |
| 20 | 0.00218 | 0.00162 | 0.00087 | 0.00098 | 0.00116 | 0.00051 | 0.00069 | 0.00088 | 0.00042 | 0.00038 | 0.00043 | 0.00038 | 0.00034 | 0.00017 | 0.00992 | 0.00977 | 0.00958 | 0.00953 | 0.00957 | 0.00995 | 0.00982 | 0.00987 | 0.00987 | 0.00985 | 0.00989 | 0.00986 |
| 21 | 0.00218 | 0.00164 | 0.00093 | 0.00103 | 0.0012 | 0.00053 | 0.00072 | 0.00089 | 0.00046 | 0.00042 | 0.00046 | 0.0004 | 0.00036 | 0.00022 | 0.01017 | 0.01002 | 0.00982 | 0.00976 | 0.0098 | 0.01012 | 0.01001 | 0.01 | 0.01007 | 0.01004 | 0.01007 | 0.01003 |
| 22 | 0.00218 | 0.00165 | 0.00093 | 0.00103 | 0.0012 | 0.00052 | 0.00072 | 0.00089 | 0.00046 | 0.00043 | 0.00046 | 0.0004 | 0.00036 | 0.00022 | 0.0102 | 0.01005 | 0.00985 | 0.00978 | 0.00982 | 0.01015 | 0.01004 | 0.01004 | 0.0101 | 0.01008 | 0.0101 | 0.01006 |
| 23 | 0.00218 | 0.00164 | 0.00094 | 0.00102 | 0.00119 | 0.00049 | 0.00072 | 0.00089 | 0.00045 | 0.00041 | 0.00044 | 0.00037 | 0.00034 | 0.0002 | 0.0102 | 0.01005 | 0.00986 | 0.00981 | 0.00984 | 0.0102 | 0.01011 | 0.01015 | 0.01016 | 0.01013 | 0.01015 | 0.01011 |
| 24 | 0.00218 | 0.00164 | 0.00094 | 0.00102 | 0.00119 | 0.00048 | 0.00071 | 0.00089 | 0.00045 | 0.00041 | 0.00043 | 0.00036 | 0.00032 | 0.0002 | 0.0102 | 0.01006 | 0.00987 | 0.00981 | 0.00985 | 0.0102 | 0.01013 | 0.0102 | 0.01018 | 0.01015 | 0.01016 | 0.01011 |
| 25 | 0.00218 | 0.00164 | 0.00094 | 0.00102 | 0.00119 | 0.00047 | 0.00072 | 0.0009 | 0.00045 | 0.0004 | 0.00042 | 0.00035 | 0.00031 | 0.00019 | 0.0102 | 0.01006 | 0.00987 | 0.00982 | 0.00985 | 0.0102 | 0.01013 | 0.0102 | 0.01018 | 0.01015 | 0.01016 | 0.01011 |
| 26 | 0.00218 | 0.00164 | 0.00094 | 0.00102 | 0.00119 | 0.00047 | 0.00072 | 0.0009 | 0.00045 | 0.0004 | 0.00042 | 0.00035 | 0.00031 | 0.00019 | 0.0102 | 0.01006 | 0.00987 | 0.00982 | 0.00985 | 0.0102 | 0.01013 | 0.0102 | 0.01018 | 0.01015 | 0.01016 | 0.01011 |
| 27 | 0.00218 | 0.00164 | 0.00094 | 0.00102 | 0.00119 | 0.00046 | 0.00072 | 0.0009 | 0.00045 | 0.0004 | 0.00042 | 0.00035 | 0.00031 | 0.00019 | 0.0102 | 0.01006 | 0.00987 | 0.00982 | 0.00985 | 0.0102 | 0.01013 | 0.0102 | 0.01018 | 0.01015 | 0.01016 | 0.01011 |

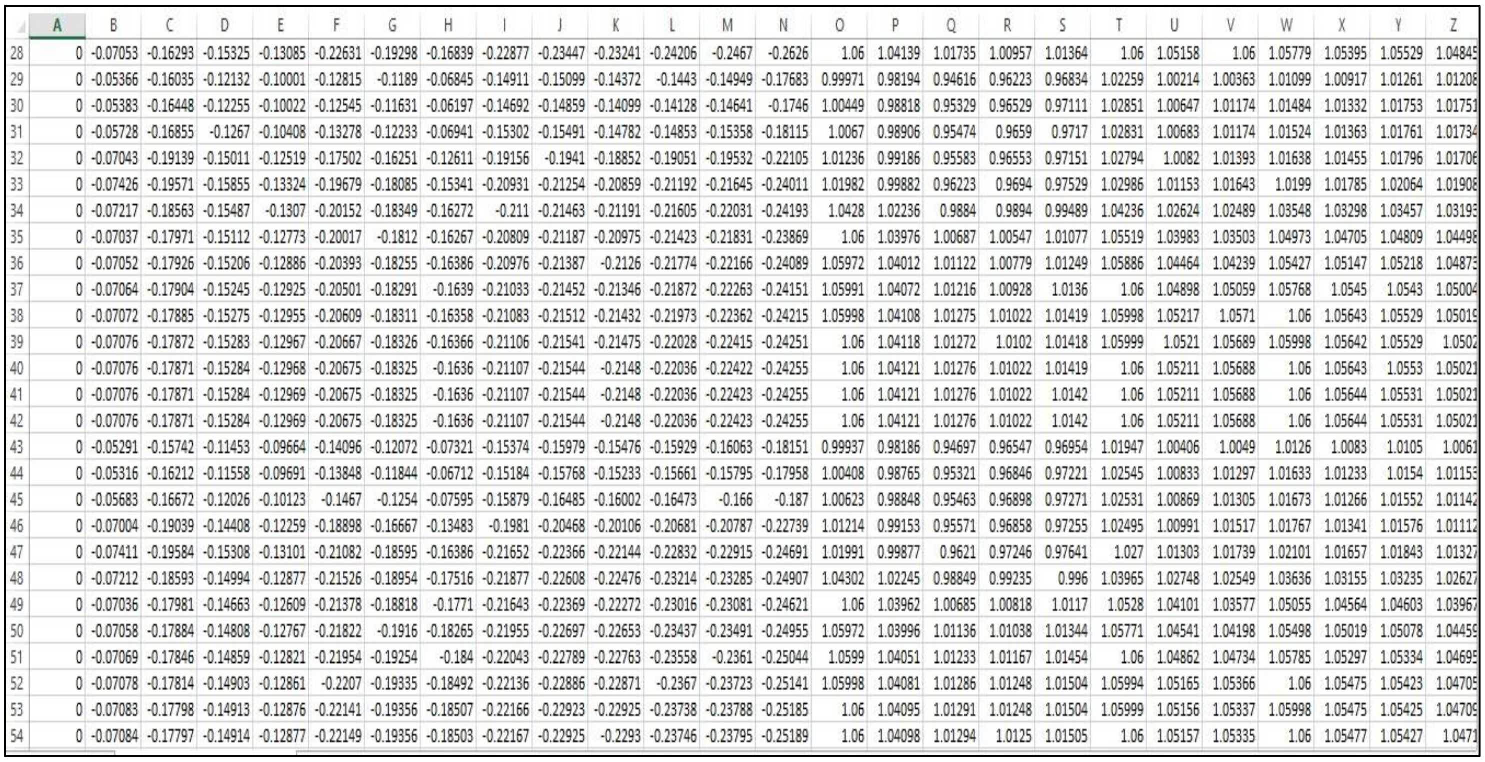


| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 28 | 0.00218 | 0.00164 | 0.00094 | 0.00102 | 0.00119 | 0.00046 | 0.00072 | 0.0009 | 0.00045 | 0.0004 | 0.00042 | 0.00035 | 0.00031 | 0.00019 | 0.0102 | 0.01006 | 0.00987 | 0.00982 | 0.00985 | 0.0102 | 0.01013 | 0.0102 | 0.01018 | 0.01015 | 0.01016 | 0.01011 |
| 29 | 0.00218 | 0.00218 | 0.00218 | 0.00218 | 0.00218 | 0.00218 | 0.00218 | 0.00218 | 0.00218 | 0.00218 | 0.00218 | 0.00218 | 0.00218 | 0.00218 | 0.00974 | 0.00974 | 0.00974 | 0.00974 | 0.00974 | 0.00974 | 0.00974 | 0.00974 | 0.00974 | 0.00974 | 0.00974 | 0.00974 |
| 30 | 0.00218 | 0.00177 | 0.00096 | 0.00126 | 0.00142 | 0.00121 | 0.00128 | 0.00166 | 0.00105 | 0.00103 | 0.00109 | 0.00108 | 0.00105 | 0.00084 | 0.00974 | 0.00961 | 0.00934 | 0.00946 | 0.0095 | 0.00991 | 0.00976 | 0.00977 | 0.00983 | 0.00981 | 0.00984 | 0.00983 |
| 31 | 0.00218 | 0.00177 | 0.00093 | 0.00125 | 0.00142 | 0.00123 | 0.0013 | 0.00171 | 0.00106 | 0.00105 | 0.00111 | 0.00111 | 0.00107 | 0.00086 | 0.00978 | 0.00965 | 0.00939 | 0.00948 | 0.00952 | 0.00996 | 0.00979 | 0.00983 | 0.00986 | 0.00984 | 0.00988 | 0.00988 |
| 32 | 0.00218 | 0.00174 | 0.0009 | 0.00122 | 0.00139 | 0.00117 | 0.00125 | 0.00165 | 0.00102 | 0.001 | 0.00106 | 0.00105 | 0.00101 | 0.00081 | 0.00979 | 0.00966 | 0.0094 | 0.00949 | 0.00953 | 0.00996 | 0.0098 | 0.00983 | 0.00986 | 0.00985 | 0.00988 | 0.00987 |
| 33 | 0.00218 | 0.00164 | 0.00073 | 0.00104 | 0.00123 | 0.00085 | 0.00095 | 0.00122 | 0.00073 | 0.00071 | 0.00075 | 0.00074 | 0.0007 | 0.0005 | 0.00984 | 0.00968 | 0.00941 | 0.00948 | 0.00953 | 0.00995 | 0.00981 | 0.00985 | 0.00987 | 0.00985 | 0.00988 | 0.00987 |
| 34 | 0.00218 | 0.00161 | 0.0007 | 0.00098 | 0.00117 | 0.00069 | 0.00081 | 0.00102 | 0.00059 | 0.00057 | 0.0006 | 0.00057 | 0.00054 | 0.00036 | 0.00989 | 0.00973 | 0.00946 | 0.00951 | 0.00956 | 0.00997 | 0.00983 | 0.00987 | 0.00989 | 0.00988 | 0.0099 | 0.00989 |
| 35 | 0.00218 | 0.00163 | 0.00077 | 0.001 | 0.00119 | 0.00065 | 0.00079 | 0.00095 | 0.00058 | 0.00055 | 0.00057 | 0.00054 | 0.00051 | 0.00035 | 0.01007 | 0.00991 | 0.00966 | 0.00966 | 0.0097 | 0.01006 | 0.00994 | 0.00993 | 0.01001 | 0.00999 | 0.01 | 0.00999 |
| 36 | 0.00218 | 0.00164 | 0.00082 | 0.00103 | 0.00121 | 0.00066 | 0.00081 | 0.00095 | 0.0006 | 0.00057 | 0.00059 | 0.00056 | 0.00052 | 0.00037 | 0.0102 | 0.01004 | 0.0098 | 0.00978 | 0.00982 | 0.01016 | 0.01004 | 0.01001 | 0.01012 | 0.0101 | 0.01011 | 0.01008 |
| 37 | 0.00218 | 0.00164 | 0.00082 | 0.00103 | 0.0012 | 0.00063 | 0.0008 | 0.00094 | 0.00059 | 0.00056 | 0.00057 | 0.00053 | 0.0005 | 0.00035 | 0.0102 | 0.01005 | 0.00983 | 0.0098 | 0.00984 | 0.01019 | 0.01008 | 0.01006 | 0.01015 | 0.01013 | 0.01014 | 0.01011 |
| 38 | 0.00218 | 0.00164 | 0.00082 | 0.00102 | 0.0012 | 0.00063 | 0.00079 | 0.00094 | 0.00059 | 0.00055 | 0.00056 | 0.00052 | 0.00049 | 0.00035 | 0.0102 | 0.01005 | 0.00984 | 0.00981 | 0.00985 | 0.0102 | 0.01011 | 0.01013 | 0.01018 | 0.01016 | 0.01015 | 0.01012 |
| 39 | 0.00218 | 0.00164 | 0.00082 | 0.00102 | 0.0012 | 0.00062 | 0.00079 | 0.00094 | 0.00058 | 0.00055 | 0.00056 | 0.00051 | 0.00048 | 0.00034 | 0.0102 | 0.01005 | 0.00984 | 0.00982 | 0.00985 | 0.0102 | 0.01014 | 0.01018 | 0.0102 | 0.01017 | 0.01016 | 0.01012 |
| 40 | 0.00218 | 0.00164 | 0.00082 | 0.00102 | 0.0012 | 0.00061 | 0.00079 | 0.00094 | 0.00058 | 0.00055 | 0.00055 | 0.00051 | 0.00048 | 0.00034 | 0.0102 | 0.01006 | 0.00984 | 0.00982 | 0.00985 | 0.0102 | 0.01014 | 0.01017 | 0.0102 | 0.01017 | 0.01016 | 0.01012 |
| 41 | 0.00218 | 0.00164 | 0.00082 | 0.00102 | 0.0012 | 0.00061 | 0.00079 | 0.00094 | 0.00058 | 0.00055 | 0.00055 | 0.00051 | 0.00048 | 0.00034 | 0.0102 | 0.01006 | 0.00984 | 0.00982 | 0.00985 | 0.0102 | 0.01014 | 0.01017 | 0.0102 | 0.01017 | 0.01016 | 0.01012 |
| 42 | 0.00218 | 0.00164 | 0.00082 | 0.00102 | 0.0012 | 0.00061 | 0.00079 | 0.00094 | 0.00058 | 0.00055 | 0.00055 | 0.00051 | 0.00048 | 0.00034 | 0.0102 | 0.01006 | 0.00984 | 0.00982 | 0.00985 | 0.0102 | 0.01014 | 0.01017 | 0.0102 | 0.01017 | 0.01016 | 0.01012 |
| 43 | 0.00218 | 0.00218 | 0.00218 | 0.00218 | 0.00218 | 0.00218 | 0.00218 | 0.00218 | 0.00218 | 0.00218 | 0.00218 | 0.00218 | 0.00218 | 0.00218 | 0.00974 | 0.00974 | 0.00974 | 0.00974 | 0.00974 | 0.00974 | 0.00974 | 0.00974 | 0.00974 | 0.00974 | 0.00974 | 0.00974 |
| 44 | 0.00218 | 0.00178 | 0.00099 | 0.00131 | 0.00145 | 0.00111 | 0.00126 | 0.00162 | 0.00101 | 0.00097 | 0.00101 | 0.00097 | 0.00096 | 0.0008 | 0.00974 | 0.00961 | 0.00934 | 0.00948 | 0.00951 | 0.00989 | 0.00977 | 0.00978 | 0.00984 | 0.00981 | 0.00982 | 0.00979 |
| 45 | 0.00218 | 0.00177 | 0.00095 | 0.0013 | 0.00144 | 0.00113 | 0.00128 | 0.00167 | 0.00103 | 0.00098 | 0.00102 | 0.00099 | 0.00098 | 0.00082 | 0.00977 | 0.00965 | 0.00939 | 0.0095 | 0.00953 | 0.00994 | 0.00981 | 0.00984 | 0.00987 | 0.00984 | 0.00986 | 0.00983 |
| 46 | 0.00218 | 0.00175 | 0.00092 | 0.00127 | 0.00141 | 0.00107 | 0.00123 | 0.0016 | 0.00098 | 0.00093 | 0.00097 | 0.00093 | 0.00092 | 0.00076 | 0.00979 | 0.00966 | 0.0094 | 0.00951 | 0.00954 | 0.00993 | 0.00981 | 0.00984 | 0.00987 | 0.00984 | 0.00986 | 0.00983 |
| 47 | 0.00218 | 0.00165 | 0.00074 | 0.00109 | 0.00125 | 0.00075 | 0.00092 | 0.00116 | 0.00068 | 0.00063 | 0.00066 | 0.00061 | 0.0006 | 0.00046 | 0.00984 | 0.00968 | 0.00941 | 0.00951 | 0.00954 | 0.00993 | 0.00982 | 0.00986 | 0.00988 | 0.00984 | 0.00986 | 0.00983 |
| 48 | 0.00218 | 0.00162 | 0.00069 | 0.00102 | 0.00119 | 0.00058 | 0.00077 | 0.00094 | 0.00054 | 0.00048 | 0.0005 | 0.00045 | 0.00044 | 0.00031 | 0.00989 | 0.00973 | 0.00946 | 0.00954 | 0.00956 | 0.00995 | 0.00984 | 0.00988 | 0.0099 | 0.00987 | 0.00988 | 0.00984 |
| 49 | 0.00218 | 0.00163 | 0.00077 | 0.00104 | 0.0012 | 0.00055 | 0.00074 | 0.00085 | 0.00052 | 0.00047 | 0.00048 | 0.00042 | 0.00041 | 0.00029 | 0.01007 | 0.00991 | 0.00966 | 0.00969 | 0.00971 | 0.01004 | 0.00995 | 0.00994 | 0.01002 | 0.00998 | 0.00999 | 0.00994 |
| 50 | 0.00218 | 0.00164 | 0.00082 | 0.00107 | 0.00122 | 0.00056 | 0.00075 | 0.00084 | 0.00054 | 0.00048 | 0.00049 | 0.00044 | 0.00043 | 0.00031 | 0.0102 | 0.01004 | 0.0098 | 0.00981 | 0.00983 | 0.01014 | 0.01005 | 0.01001 | 0.01013 | 0.01009 | 0.01009 | 0.01004 |
| 51 | 0.00218 | 0.00164 | 0.00082 | 0.00106 | 0.00121 | 0.00053 | 0.00073 | 0.00079 | 0.00052 | 0.00046 | 0.00046 | 0.0004 | 0.0004 | 0.00029 | 0.0102 | 0.01005 | 0.00983 | 0.00982 | 0.00985 | 0.01018 | 0.01009 | 0.01006 | 0.01016 | 0.01012 | 0.01013 | 0.01008 |
| 52 | 0.00218 | 0.00164 | 0.00083 | 0.00105 | 0.00121 | 0.00052 | 0.00072 | 0.00078 | 0.00051 | 0.00045 | 0.00045 | 0.00039 | 0.00039 | 0.00028 | 0.0102 | 0.01005 | 0.00984 | 0.00983 | 0.00985 | 0.0102 | 0.01011 | 0.0101 | 0.01018 | 0.01014 | 0.01015 | 0.0101 |
| 53 | 0.00218 | 0.00164 | 0.00083 | 0.00105 | 0.0012 | 0.00051 | 0.00071 | 0.00078 | 0.0005 | 0.00044 | 0.00045 | 0.00039 | 0.00038 | 0.00027 | 0.0102 | 0.01005 | 0.00984 | 0.00984 | 0.00986 | 0.0102 | 0.01013 | 0.01015 | 0.0102 | 0.01016 | 0.01015 | 0.0101 |
| 54 | 0.00218 | 0.00164 | 0.00083 | 0.00105 | 0.0012 | 0.0005 | 0.00071 | 0.00078 | 0.0005 | 0.00044 | 0.00044 | 0.00038 | 0.00038 | 0.00027 | 0.0102 | 0.01005 | 0.00984 | 0.00984 | 0.00986 | 0.0102 | 0.01013 | 0.01015 | 0.0102 | 0.01016 | 0.01015 | 0.0101 |

Name: Alam Ali, Net ID: aa8007, DL course project (Final report)
DL dataset (x(k) and x(k+1)) is given s.t x(k+1)=F(x(k))
(1) x(k+1) is an output data to Deep Learning program.



| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | -0.05004 | -0.15606 | -0.10441 | -0.08758 | -0.12474 | -0.11156 | -0.07338 | -0.13924 | -0.14182 | -0.13606 | -0.1413 | -0.14439 | -0.16013 | 0.99439 | 0.97793 | 0.94167 | 0.96531 | 0.96883 | 1.01944 | 1.00287 | 1.00176 | 1.01222 | 1.01001 | 1.01252 | 1.00796 |
| 2 | 0 | -0.05078 | -0.16403 | -0.10667 | -0.08892 | -0.12376 | -0.11032 | -0.06758 | -0.1387 | -0.14114 | -0.13515 | -0.14023 | -0.14334 | -0.15964 | 0.99942 | 0.9829 | 0.94578 | 0.9682 | 0.97157 | 1.02657 | 1.00858 | 1.01142 | 1.01757 | 1.01559 | 1.01879 | 1.01456 |
| 3 | 0 | -0.05488 | -0.16962 | -0.11245 | -0.09423 | -0.13399 | -0.11941 | -0.07967 | -0.14762 | -0.1503 | -0.14484 | -0.15036 | -0.15338 | -0.16904 | 1.00165 | 0.98383 | 0.94764 | 0.96876 | 0.97209 | 1.02649 | 1.00905 | 1.01166 | 1.01807 | 1.016 | 1.01898 | 1.01452 |
| 4 | 0 | -0.06798 | -0.19522 | -0.13665 | -0.11574 | -0.17557 | -0.16115 | -0.13955 | -0.18713 | -0.19017 | -0.18556 | -0.19179 | -0.19467 | -0.2093 | 1.00803 | 0.98732 | 0.94864 | 0.9686 | 0.97229 | 1.02633 | 1.01015 | 1.01349 | 1.01892 | 1.0167 | 1.01928 | 1.01441 |
| 5 | 0 | -0.07265 | -0.2044 | -0.14656 | -0.12472 | -0.19654 | -0.18032 | -0.1677 | -0.20542 | -0.20891 | -0.20538 | -0.2125 | -0.21519 | -0.22846 | 1.01589 | 0.99452 | 0.95332 | 0.97214 | 0.97601 | 1.02828 | 1.01279 | 1.01517 | 1.0218 | 1.01946 | 1.02169 | 1.01643 |
| 6 | 0 | -0.07113 | -0.19711 | -0.14402 | -0.1228 | -0.19954 | -0.18321 | -0.17714 | -0.20707 | -0.21068 | -0.20771 | -0.21499 | -0.21757 | -0.2299 | 1.03877 | 1.01784 | 0.97999 | 0.99198 | 0.99556 | 1.04084 | 1.02678 | 1.02248 | 1.03673 | 1.03404 | 1.03532 | 1.0293 |
| 7 | 0 | -0.06896 | -0.18949 | -0.13996 | -0.11945 | -0.19685 | -0.18117 | -0.17887 | -0.20383 | -0.20737 | -0.20464 | -0.21177 | -0.21425 | -0.226 | 1.06 | 1.03927 | 1.00308 | 1.01197 | 1.01541 | 1.05788 | 1.04404 | 1.03584 | 1.05478 | 1.05198 | 1.05286 | 1.04661 |
| 8 | 0 | -0.06914 | -0.18925 | -0.14074 | -0.12022 | -0.19878 | -0.18329 | -0.18255 | -0.20569 | -0.20924 | -0.20653 | -0.21366 | -0.21614 | -0.22783 | 1.0596 | 1.03955 | 1.00761 | 1.01335 | 1.01636 | 1.05942 | 1.04581 | 1.03805 | 1.05649 | 1.05366 | 1.05448 | 1.04817 |
| 9 | 0 | -0.06929 | -0.18917 | -0.14095 | -0.12039 | -0.19903 | -0.18384 | -0.18353 | -0.20614 | -0.20965 | -0.20687 | -0.21389 | -0.21639 | -0.22817 | 1.05988 | 1.04042 | 1.00909 | 1.01439 | 1.01722 | 1.06 | 1.0475 | 1.04078 | 1.05787 | 1.05491 | 1.0554 | 1.04882 |
| 10 | 0 | -0.06935 | -0.18913 | -0.14117 | -0.12052 | -0.19905 | -0.18428 | -0.18421 | -0.20655 | -0.20999 | -0.20706 | -0.21392 | -0.21647 | -0.2284 | 1.05997 | 1.04071 | 1.00963 | 1.01506 | 1.01768 | 1.05997 | 1.04965 | 1.04522 | 1.05936 | 1.05613 | 1.05601 | 1.0489 |
| 11 | 0 | -0.06938 | -0.18911 | -0.1412 | -0.12055 | -0.19912 | -0.18436 | -0.18434 | -0.20661 | -0.21006 | -0.20713 | -0.21399 | -0.21654 | -0.22846 | 1.05999 | 1.0408 | 1.00954 | 1.01508 | 1.01771 | 1.06 | 1.04965 | 1.04521 | 1.05937 | 1.05615 | 1.05603 | 1.04893 |
| 12 | 0 | -0.0694 | -0.18911 | -0.14121 | -0.12056 | -0.19913 | -0.18438 | -0.18438 | -0.20663 | -0.21008 | -0.20715 | -0.214 | -0.21656 | -0.22848 | 1.06 | 1.04087 | 1.00953 | 1.0151 | 1.01773 | 1.06 | 1.04964 | 1.04517 | 1.05937 | 1.05614 | 1.05603 | 1.04893 |
| 13 | 0 | -0.0694 | -0.18911 | -0.14121 | -0.12056 | -0.19913 | -0.18438 | -0.18438 | -0.20663 | -0.21008 | -0.20715 | -0.214 | -0.21656 | -0.22848 | 1.06 | 1.04087 | 1.00954 | 1.0151 | 1.01773 | 1.06 | 1.04964 | 1.04516 | 1.05937 | 1.05614 | 1.05603 | 1.04893 |
| 14 | 0 | -0.0694 | -0.18911 | -0.14121 | -0.12056 | -0.19913 | -0.18438 | -0.18438 | -0.20663 | -0.21008 | -0.20715 | -0.214 | -0.21656 | -0.22848 | 1.06 | 1.04087 | 1.00954 | 1.0151 | 1.01773 | 1.06 | 1.04964 | 1.04516 | 1.05937 | 1.05614 | 1.05603 | 1.04893 |
| 15 | 0 | -0.05082 | -0.12863 | -0.1186 | -0.09915 | -0.15248 | -0.13513 | -0.09007 | -0.17216 | -0.17533 | -0.16624 | -0.17084 | -0.1769 | -0.20178 | 1.00194 | 0.98569 | 0.96511 | 0.96482 | 0.97022 | 1.01914 | 1.00084 | 1.00167 | 1.00813 | 1.00556 | 1.01038 | 1.00719 |
| 16 | 0 | -0.0511 | -0.13094 | -0.12017 | -0.09969 | -0.15083 | -0.13349 | -0.08359 | -0.17146 | -0.17442 | -0.16485 | -0.16914 | -0.17526 | -0.20151 | 1.00625 | 0.99088 | 0.97225 | 0.96697 | 0.97196 | 1.02402 | 1.00428 | 1.00954 | 1.01074 | 1.00843 | 1.01413 | 1.01135 |
| 17 | 0 | -0.05534 | -0.13728 | -0.12577 | -0.10478 | -0.15999 | -0.14138 | -0.09343 | -0.17933 | -0.18252 | -0.17347 | -0.17821 | -0.18424 | -0.20984 | 1.00816 | 0.99139 | 0.97277 | 0.96717 | 0.97219 | 1.02403 | 1.00469 | 1.01004 | 1.01115 | 1.00879 | 1.01433 | 1.01138 |
| 18 | 0 | -0.06847 | -0.16326 | -0.14906 | -0.12555 | -0.20022 | -0.17965 | -0.14647 | -0.21611 | -0.21991 | -0.21227 | -0.21819 | -0.22399 | -0.24788 | 1.01413 | 0.99449 | 0.97304 | 0.96686 | 0.97214 | 1.0239 | 1.00596 | 1.01237 | 1.01222 | 1.0097 | 1.01478 | 1.01131 |
| 19 | 0 | -0.07311 | -0.17317 | -0.15828 | -0.13395 | -0.22017 | -0.19672 | -0.17094 | -0.23257 | -0.23693 | -0.23072 | -0.23775 | -0.24329 | -0.26531 | 1.02369 | 1.00335 | 0.97892 | 0.97218 | 0.97758 | 1.02741 | 1.01064 | 1.01616 | 1.01713 | 1.01443 | 1.01897 | 1.01495 |
| 20 | 0 | -0.07041 | -0.16525 | -0.15187 | -0.12882 | -0.21773 | -0.19245 | -0.17069 | -0.22687 | -0.23151 | -0.2267 | -0.2343 | -0.23944 | -0.25902 | 1.05694 | 1.03702 | 1.01059 | 1.00173 | 1.00706 | 1.04983 | 1.0349 | 1.03334 | 1.04268 | 1.03957 | 1.04286 | 1.03786 |
| 21 | 0 | -0.07014 | -0.16417 | -0.15157 | -0.12873 | -0.2186 | -0.19216 | -0.17038 | -0.22658 | -0.23138 | -0.22706 | -0.23498 | -0.24 | -0.25889 | 1.06 | 1.04011 | 1.01408 | 1.00528 | 1.01046 | 1.05399 | 1.03941 | 1.03857 | 1.04721 | 1.04407 | 1.04722 | 1.04208 |
| 22 | 0 | -0.07036 | -0.16349 | -0.15275 | -0.13011 | -0.22244 | -0.19308 | -0.17055 | -0.22787 | -0.23304 | -0.22977 | -0.23843 | -0.24325 | -0.26067 | 1.05995 | 1.04078 | 1.01602 | 1.00842 | 1.013 | 1.06 | 1.0482 | 1.05357 | 1.05509 | 1.05169 | 1.05409 | 1.04828 |
| 23 | 0 | -0.07048 | -0.16318 | -0.15312 | -0.13055 | -0.22435 | -0.19323 | -0.16977 | -0.2285 | -0.23391 | -0.23117 | -0.24023 | -0.24498 | -0.26172 | 1.05998 | 1.04118 | 1.01684 | 1.00939 | 1.01357 | 1.05994 | 1.05141 | 1.06 | 1.05748 | 1.05367 | 1.0551 | 1.04839 |
| 24 | 0 | -0.07052 | -0.16302 | -0.1532 | -0.13076 | -0.22573 | -0.19302 | -0.16872 | -0.22867 | -0.23428 | -0.23204 | -0.24152 | -0.24619 | -0.26232 | 1.06 | 1.04137 | 1.01729 | 1.00955 | 1.01365 | 1.06 | 1.05153 | 1.05995 | 1.05771 | 1.05388 | 1.05525 | 1.04845 |
| 25 | 0 | -0.07053 | -0.16295 | -0.15324 | -0.13084 | -0.2262 | -0.19299 | -0.16845 | -0.22875 | -0.23443 | -0.23234 | -0.24196 | -0.2466 | -0.26255 | 1.06 | 1.04138 | 1.01734 | 1.00956 | 1.01364 | 1.06 | 1.05158 | 1.06 | 1.05778 | 1.05394 | 1.05528 | 1.04845 |
| 26 | 0 | -0.07053 | -0.16293 | -0.15324 | -0.13085 | -0.2263 | -0.19298 | -0.16839 | -0.22877 | -0.23447 | -0.23241 | -0.24206 | -0.24669 | -0.2626 | 1.06 | 1.04139 | 1.01735 | 1.00957 | 1.01364 | 1.06 | 1.05158 | 1.06 | 1.05779 | 1.05395 | 1.05529 | 1.04845 |
| 27 | 0 | -0.07053 | -0.16293 | -0.15325 | -0.13085 | -0.22631 | -0.19298 | -0.16839 | -0.22877 | -0.23447 | -0.23241 | -0.24206 | -0.2467 | -0.2626 | 1.06 | 1.04139 | 1.01735 | 1.00957 | 1.01364 | 1.06 | 1.05158 | 1.06 | 1.05779 | 1.05395 | 1.05529 | 1.04845 |



| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 28 | 0 | -0.07053 | -0.16293 | -0.15325 | -0.13085 | -0.22631 | -0.19298 | -0.16839 | -0.22877 | -0.23447 | -0.23241 | -0.24206 | -0.2467 | -0.2626 | 1.06 | 1.04139 | 1.01735 | 1.00957 | 1.01364 | 1.06 | 1.05158 | 1.06 | 1.05779 | 1.05395 | 1.05529 | 1.04845 |
| 29 | 0 | -0.05366 | -0.16035 | -0.12132 | -0.10001 | -0.12815 | -0.1189 | -0.06845 | -0.14911 | -0.15099 | -0.14372 | -0.1443 | -0.14949 | -0.17683 | 0.99971 | 0.98194 | 0.94616 | 0.96223 | 0.96834 | 1.02259 | 1.00214 | 1.00363 | 1.01099 | 1.00917 | 1.01261 | 1.01208 |
| 30 | 0 | -0.05383 | -0.16448 | -0.12255 | -0.10022 | -0.12545 | -0.11631 | -0.06197 | -0.14692 | -0.14859 | -0.14099 | -0.14128 | -0.14641 | -0.1746 | 1.00449 | 0.98818 | 0.95329 | 0.96529 | 0.97111 | 1.02851 | 1.00647 | 1.01174 | 1.01484 | 1.01332 | 1.01753 | 1.01751 |
| 31 | 0 | -0.05728 | -0.16855 | -0.1267 | -0.10408 | -0.13278 | -0.12233 | -0.06941 | -0.15302 | -0.15491 | -0.14782 | -0.14853 | -0.15358 | -0.18115 | 1.0067 | 0.98906 | 0.95474 | 0.9659 | 0.9717 | 1.02831 | 1.00683 | 1.01174 | 1.01524 | 1.01363 | 1.01761 | 1.01734 |
| 32 | 0 | -0.07043 | -0.19139 | -0.15011 | -0.12519 | -0.17502 | -0.16251 | -0.12611 | -0.19156 | -0.1941 | -0.18852 | -0.19051 | -0.19532 | -0.22105 | 1.01236 | 0.99186 | 0.95583 | 0.96553 | 0.97151 | 1.02794 | 1.0082 | 1.01393 | 1.01638 | 1.01455 | 1.01796 | 1.01706 |
| 33 | 0 | -0.07426 | -0.19571 | -0.15855 | -0.13324 | -0.19679 | -0.18085 | -0.15341 | -0.20931 | -0.21254 | -0.20859 | -0.21192 | -0.21645 | -0.24011 | 1.01982 | 0.99882 | 0.96223 | 0.9694 | 0.97529 | 1.02986 | 1.01153 | 1.01643 | 1.0199 | 1.01785 | 1.02064 | 1.01908 |
| 34 | 0 | -0.07217 | -0.18563 | -0.15487 | -0.1307 | -0.20152 | -0.18349 | -0.16272 | -0.211 | -0.21463 | -0.21191 | -0.21605 | -0.22031 | -0.24193 | 1.0428 | 1.02236 | 0.9884 | 0.9894 | 0.99489 | 1.04236 | 1.02624 | 1.02489 | 1.03548 | 1.03298 | 1.03457 | 1.03193 |
| 35 | 0 | -0.07037 | -0.17971 | -0.15112 | -0.12773 | -0.20017 | -0.1812 | -0.16267 | -0.20809 | -0.21187 | -0.20975 | -0.21423 | -0.21831 | -0.23869 | 1.06 | 1.03976 | 1.00687 | 1.00547 | 1.01077 | 1.05519 | 1.03983 | 1.03503 | 1.04973 | 1.04705 | 1.04809 | 1.04498 |
| 36 | 0 | -0.07052 | -0.17926 | -0.15206 | -0.12886 | -0.20393 | -0.18255 | -0.16386 | -0.20976 | -0.21387 | -0.2126 | -0.21774 | -0.22166 | -0.24089 | 1.05972 | 1.04012 | 1.01122 | 1.00779 | 1.01249 | 1.05886 | 1.04464 | 1.04239 | 1.05427 | 1.05147 | 1.05218 | 1.04873 |
| 37 | 0 | -0.07064 | -0.17904 | -0.15245 | -0.12925 | -0.20501 | -0.18291 | -0.1639 | -0.21033 | -0.21452 | -0.21346 | -0.21872 | -0.22263 | -0.24151 | 1.05991 | 1.04072 | 1.01216 | 1.00928 | 1.0136 | 1.06 | 1.04898 | 1.05059 | 1.05768 | 1.0545 | 1.0543 | 1.05004 |
| 38 | 0 | -0.07072 | -0.17885 | -0.15275 | -0.12955 | -0.20609 | -0.18311 | -0.16358 | -0.21083 | -0.21512 | -0.21432 | -0.21973 | -0.22362 | -0.24215 | 1.05998 | 1.04108 | 1.01275 | 1.01022 | 1.01419 | 1.05998 | 1.05217 | 1.0571 | 1.06 | 1.05643 | 1.05529 | 1.05019 |
| 39 | 0 | -0.07076 | -0.17872 | -0.15283 | -0.12967 | -0.20667 | -0.18326 | -0.16366 | -0.21106 | -0.21541 | -0.21475 | -0.22028 | -0.22415 | -0.24251 | 1.06 | 1.04118 | 1.01272 | 1.0102 | 1.01418 | 1.05999 | 1.0521 | 1.05689 | 1.05998 | 1.05642 | 1.05529 | 1.0502 |
| 40 | 0 | -0.07076 | -0.17871 | -0.15284 | -0.12968 | -0.20675 | -0.18325 | -0.1636 | -0.21107 | -0.21544 | -0.2148 | -0.22036 | -0.22422 | -0.24255 | 1.06 | 1.04121 | 1.01276 | 1.01022 | 1.01419 | 1.06 | 1.05211 | 1.05688 | 1.06 | 1.05643 | 1.0553 | 1.05021 |
| 41 | 0 | -0.07076 | -0.17871 | -0.15284 | -0.12969 | -0.20675 | -0.18325 | -0.1636 | -0.21107 | -0.21544 | -0.2148 | -0.22036 | -0.22423 | -0.24255 | 1.06 | 1.04121 | 1.01276 | 1.01022 | 1.0142 | 1.06 | 1.05211 | 1.05688 | 1.06 | 1.05644 | 1.05531 | 1.05021 |
| 42 | 0 | -0.07076 | -0.17871 | -0.15284 | -0.12969 | -0.20675 | -0.18325 | -0.1636 | -0.21107 | -0.21544 | -0.2148 | -0.22036 | -0.22423 | -0.24255 | 1.06 | 1.04121 | 1.01276 | 1.01022 | 1.0142 | 1.06 | 1.05211 | 1.05688 | 1.06 | 1.05644 | 1.05531 | 1.05021 |
| 43 | 0 | -0.05291 | -0.15742 | -0.11453 | -0.09664 | -0.14096 | -0.12072 | -0.07321 | -0.15374 | -0.15979 | -0.15476 | -0.15929 | -0.16063 | -0.18151 | 0.99937 | 0.98186 | 0.94697 | 0.96547 | 0.96954 | 1.01947 | 1.00406 | 1.0049 | 1.0126 | 1.0083 | 1.0105 | 1.0061 |
| 44 | 0 | -0.05316 | -0.16212 | -0.11558 | -0.09691 | -0.13848 | -0.11844 | -0.06712 | -0.15184 | -0.15768 | -0.15233 | -0.15661 | -0.15795 | -0.17958 | 1.00408 | 0.98765 | 0.95321 | 0.96846 | 0.97221 | 1.02545 | 1.00833 | 1.01297 | 1.01633 | 1.01233 | 1.0154 | 1.01153 |
| 45 | 0 | -0.05683 | -0.16672 | -0.12026 | -0.10123 | -0.1467 | -0.1254 | -0.07595 | -0.15879 | -0.16485 | -0.16002 | -0.16473 | -0.166 | -0.187 | 1.00623 | 0.98848 | 0.95463 | 0.96898 | 0.97271 | 1.02531 | 1.00869 | 1.01305 | 1.01673 | 1.01266 | 1.01552 | 1.01142 |
| 46 | 0 | -0.07004 | -0.19039 | -0.14408 | -0.12259 | -0.18898 | -0.16667 | -0.13483 | -0.1981 | -0.20468 | -0.20106 | -0.20681 | -0.20787 | -0.22739 | 1.01214 | 0.99153 | 0.95571 | 0.96858 | 0.97255 | 1.02495 | 1.00991 | 1.01517 | 1.01767 | 1.01341 | 1.01576 | 1.01112 |
| 47 | 0 | -0.07411 | -0.19584 | -0.15308 | -0.13101 | -0.21082 | -0.18595 | -0.16386 | -0.21652 | -0.22366 | -0.22144 | -0.22832 | -0.22915 | -0.24691 | 1.01991 | 0.99877 | 0.9621 | 0.97246 | 0.97641 | 1.027 | 1.01303 | 1.01739 | 1.02101 | 1.01657 | 1.01843 | 1.01327 |
| 48 | 0 | -0.07212 | -0.18593 | -0.14994 | -0.12877 | -0.21526 | -0.18954 | -0.17516 | -0.21877 | -0.22608 | -0.22476 | -0.23214 | -0.23285 | -0.24907 | 1.04302 | 1.02245 | 0.98849 | 0.99235 | 0.996 | 1.03965 | 1.02748 | 1.02549 | 1.03636 | 1.03155 | 1.03235 | 1.02627 |
| 49 | 0 | -0.07036 | -0.17981 | -0.14663 | -0.12609 | -0.21378 | -0.18818 | -0.1771 | -0.21643 | -0.22369 | -0.22272 | -0.23016 | -0.23081 | -0.24621 | 1.06 | 1.03962 | 1.00685 | 1.00818 | 1.0117 | 1.0528 | 1.04101 | 1.03577 | 1.05055 | 1.04564 | 1.04603 | 1.03967 |
| 50 | 0 | -0.07058 | -0.17884 | -0.14808 | -0.12767 | -0.21822 | -0.1916 | -0.18265 | -0.21955 | -0.22697 | -0.22653 | -0.23437 | -0.23491 | -0.24955 | 1.05972 | 1.03996 | 1.01136 | 1.01038 | 1.01344 | 1.05771 | 1.04541 | 1.04198 | 1.05498 | 1.05019 | 1.05078 | 1.04459 |
| 51 | 0 | -0.07069 | -0.17846 | -0.14859 | -0.12821 | -0.21954 | -0.19254 | -0.184 | -0.22043 | -0.22789 | -0.22763 | -0.23558 | -0.2361 | -0.25044 | 1.0599 | 1.04051 | 1.01233 | 1.01167 | 1.01454 | 1.06 | 1.04862 | 1.04734 | 1.05785 | 1.05297 | 1.05334 | 1.04695 |
| 52 | 0 | -0.07078 | -0.17814 | -0.14903 | -0.12861 | -0.2207 | -0.19335 | -0.18492 | -0.22136 | -0.22886 | -0.22871 | -0.2367 | -0.23723 | -0.25141 | 1.05998 | 1.04081 | 1.01286 | 1.01248 | 1.01504 | 1.05994 | 1.05165 | 1.05366 | 1.06 | 1.05475 | 1.05423 | 1.04705 |
| 53 | 0 | -0.07083 | -0.17798 | -0.14913 | -0.12876 | -0.22141 | -0.19356 | -0.18507 | -0.22166 | -0.22923 | -0.22925 | -0.23738 | -0.23788 | -0.25185 | 1.06 | 1.04095 | 1.01291 | 1.01248 | 1.01504 | 1.05999 | 1.05156 | 1.05337 | 1.05998 | 1.05475 | 1.05425 | 1.04709 |
| 54 | 0 | -0.07084 | -0.17797 | -0.14914 | -0.12877 | -0.22149 | -0.19356 | -0.18503 | -0.22167 | -0.22925 | -0.2293 | -0.23746 | -0.23795 | -0.25189 | 1.06 | 1.04098 | 1.01294 | 1.0125 | 1.01505 | 1.06 | 1.05157 | 1.05335 | 1.06 | 1.05477 | 1.05427 | 1.0471 |