APPENDICES

Appendix – A

(Python programming for artificial neural network to validate the model)

```
import numpy as np
#Input array
X=np.array([ ])
X1 = np.array([ ])
#Output
y=np.array([ ])
#Sigmoid Function
def sigmoid (x):
      return 1/(1 + np.exp(-x))
#Derivative of Sigmoid Function
def derivatives_sigmoid(x):
      return x * (1 - x)
np.random.seed(1)
      #Variable initialization
epoch= 1000000 #Setting training iterations
lr=0.001 #Setting learning rate
inputlayer_neurons = X.shape[1] #number of features in data set
hiddenlayer_neurons = 5 #number of hidden layers neurons
output_neurons = 1 #number of neurons at output layer
#weight and bias initialization
wh=np.random.uniform(size=(inputlayer_neurons,hiddenlayer_neuron
```

```
s))

bh=np.random.uniform(size=(1,hiddenlayer_neurons))

wout=np.random.uniform(size=(hiddenlayer_neurons,output_neurons)
)

bout=np.random.uniform(size=(1,output_neurons))

for i in range(epoch):


      #Forward Propogation

      print(i)

      print("\n")

      hidden_layer_input1=np.dot(X,wh)

      hidden_layer_input=hidden_layer_input1 + bh

      hiddenlayer_activations = sigmoid(hidden_layer_input)

      output_layer_input1=np.dot(hiddenlayer_activations,wout)

      output_layer_input= output_layer_input1+ bout

      output = sigmoid(output_layer_input)

      #print(output)

      #print("\n")


      #Backpropagation

      E = y-output

      print(E)

      print("\n")

      slope_output_layer = derivatives_sigmoid(output)

      slope_hidden_layer =
derivatives_sigmoid(hiddenlayer_activations)

      d_output = E * slope_output_layer

      Error_at_hidden_layer = d_output.dot(wout.T)
```

```
        d_hiddenlayer = Error_at_hidden_layer * slope_hidden_layer
        wout += hiddenlayer_activations.T.dot(d_output) *lr
        bout += np.sum(d_output, axis=0,keepdims=True) *lr
        wh += X.T.dot(d_hiddenlayer) *lr
        bh += np.sum(d_hiddenlayer, axis=0,keepdims=True) *lr


print(output*18.095)
hidden_layer_input1=np.dot(X1,wh)
hidden_layer_input=hidden_layer_input1 + bh
hiddenlayer_activations = sigmoid(hidden_layer_input)
output_layer_input1=np.dot(hiddenlayer_activations,wout)
output_layer_input= output_layer_input1+ bout
output = sigmoid(output_layer_input)
print(output* )
input("enter to exit")
```

Appendix – B

(Solid particle erosion test data for pre-hot corroded surface on Type 446 stainless steel)

↓ Test duration	Erosion Rate* (gm/gm X10^{-4})		
Test Temperatures →	550°C	650°C	750°C
5	5.12	6.24	14.81
10	5.34	6.59	15.32
15	5.52	7.47	16.9
20	5.41	7.73	17.2
25	5.23	7.68	16.84
30	5.19	7.62	16.57

(Solid particle erosion test data for USSPed and pre-hot corroded surface on Type 446 stainless steel)

↓ Test duration	Erosion Rate* (gm/gm X10^{-4})		
Test Temperatures →	550°C	650°C	750°C
5	4.86	6.42	11.42
10	4.79	6.36	11.63
15	4.60	6.83	11.38
20	4.73	6.72	11.27
25	4.59	6.67	11.76
30	4.63	6.80	11.98

* Average of five samples

Appendix – C

(List of Publications)

International SCI Journal Publications

1. **A. Mishra,** Dhananjay Pradhan, C.K. Behera, S. Mohan and A. Mohan, Effect of pre-hot corrosion on erosion behavior of high chromium ferritic steel for heat exchangers, Journal of Tribology, DOI:10.1115/1.4042391 (SCIE indexed, Impact factor – 1.78).
2. **A. Mishra**, Dhananjay Pradhan, C.K. Behera, S. Mohan and A. Mohan, Modeling and optimization of parameters for high-temperature solid particle erosion of the AISI 446SS using RSM and ANN, Materials Research Express, Volume 6 (2), 026513, 2019 (SCIE indexed, Impact factor – 1.45).
3. **A. Mishra**, C. K. Behera, S. Mohan, and A. Mohan. "Erosive wear of 446SS ferritic steel: a potential material for heat exchangers application." Materials Research Express 5, no. 10 (2018): 106522. (SCIE indexed, Impact factor – 1.45).
4. **A. Mishra,** C.K. Behera, S. Mohan and A. Mohan, High temperature erosion behavior of Type 446 stainless steel under the combined effect of surface modification and pre-hot corrosion, Tribology International (Under review).