1 Problem 1: Gauss's Addition-10 pt

In this problem, you are to write a script inspired by the famous problem by Carl Gauss where his primary school teacher assigned him to add up all numbers from 1 to 100.

A) Use a for loop to add up the numbers 1+2+3+...+n. Plot the solution as a function of n from n=1 to n=100. Hint: the solution for $n = 100$ should be 5,050.

B) Use a for loop to add up the numbers $1^2 + 2^2 + 3^2 + ... + n^2$. Plot the solution as a function of n from $n = 1$ to $n = 100$. Hint: the solution for $n = 100$ should be 338,350

In[ ]:=
```
sum = 0;
For[i = 1, i < 101, i++, sum = sum + i]
Print[sum]
```

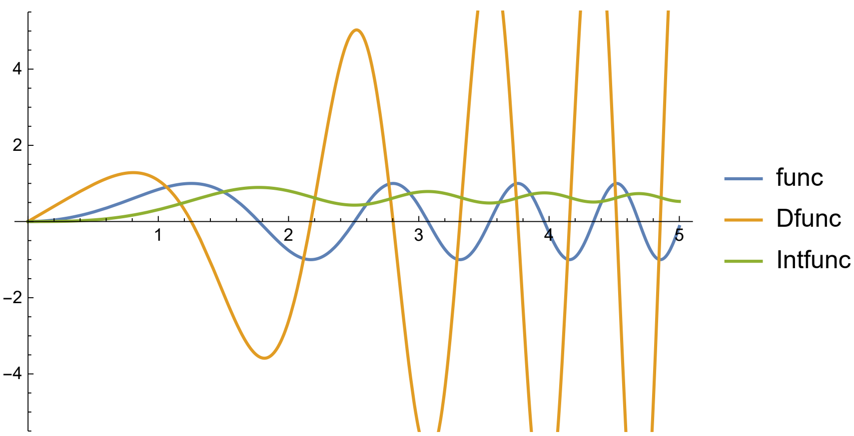
```
5050
```

In[ ]:=
```
sum2 = 0;
For[i = 1, i < 101, i++, sum2 = sum2 + i * i]
Print[sum2]
```

```
338 350
```

In[ ]:=
```
func = Sin[x^2];
Dfunc = D[func, x];
Intfunc = Integrate[func, x];
Plot[{func, Dfunc, Intfunc}, {x, 0, 5},
 PlotRange → {-5.5, 5.5}, PlotLegends -> "Expressions"]
```

Out[ ]=



3 Problem 3: Data Fitting–10 pt

Open the data file **exponentialfit.csv**. Fit the curve to the exponential function below.

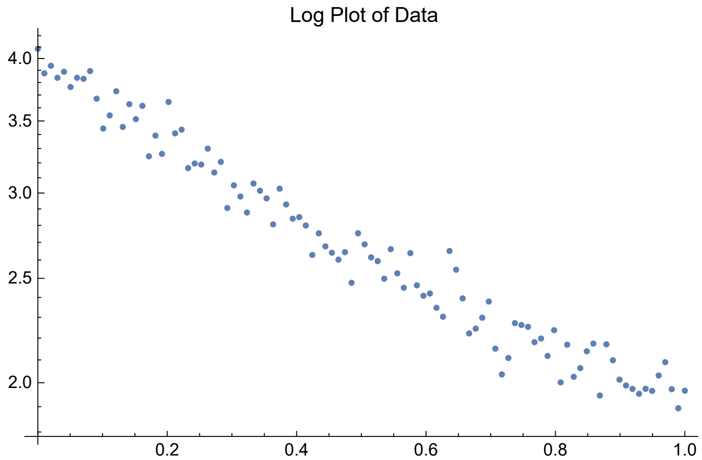
In[ ]:=

$$y = A \exp(-B \cdot x) + C \tag{1}$$

On one graph, plot the data and your fit of the data. Write down your fitted parameters of A, B, and C. Optional: find the error or the 95% confidence interval of each of these parameters.

In[ ]:=
```
data = Import["C:\\Users\\tjtho\\OneDrive\\Documents\\Coursework\\exponential_fit.csv",
    HeaderLines → 1];
ListLogPlot[data, PlotLabel -> " Log Plot of Data"]
```

Out[ ]=

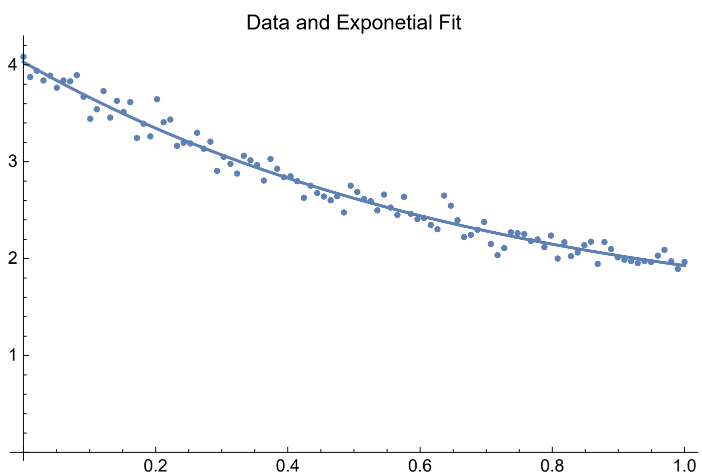


In[ ]:=
```
FindFit[data, a Exp[-b * x] + c, {a, b, c}, x]
```

Out[ ]= {SomeOtherSymbol → 2.77956, b → 1.41012, c → 1.24943}

In[ ]:=
```
Show[ListPlot[data, PlotLabel → "Data and Exponetial Fit"],
 Plot[2.779564708178729 * Exp[-1.410116967426057 * x] + 1.249428196342627,
  {x, 0, 1}, PlotLegends -> "Expressions"]]
```

Out[ ]=



4 Problem 4: Interpolation–10 pt

In[ ]:= Open the data file **current.csv** and **temperature.csv**. The first data file consists of a time-dependent current data taken from an Ammeter (current measurement tool). The second data file consists of the time-dependent temperature data taken from a thermocouple. The two instruments output times in different formats. Using interpolation, plot the current as a function of the temperature.

In[ ]:=
```
current = Import["C:\\Users\\tjtho\\OneDrive\\Documents\\Coursework\\current.csv",
    {"CSV", "Data", All, {1, 2}}, HeaderLines → 1];
tempurature = Import[
    "C:\\Users\\tjtho\\OneDrive\\Documents\\Coursework\\temperature.csv",
    {"CSV", "Data", All, {1, 2}}, HeaderLines → 1];
```

In[◦]:=
```
ItrpltTemp = Interpolation[tempurature, InterpolationOrder → 1]
```

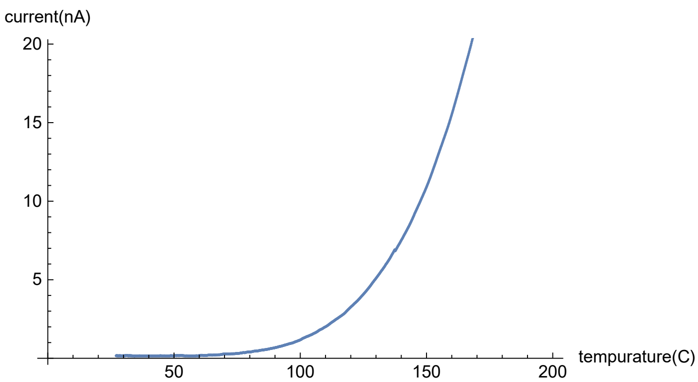
Out[◦]=
```
InterpolatingFunction[ Domain: {{0., 1.92 × 10^3}} Output: scalar ]
```

In[◦]:=
```
ItrpltCurrent = Interpolation[current, InterpolationOrder → 0]
```

Out[◦]=
```
InterpolatingFunction[ Domain: {{0., 1.92 × 10^3}} Output: scalar ]
```

In[◦]:=
```
ListPlot[Table[{ItrpltTemp[i], ItrpltCurrent[i]}, {i, 0, Length[current] - 1}],
 AxesLabel → {"tempurature(C)", "current(nA)"}]
```

Out[◦]=



5 Problem 5: 3D Plotting–10 pt

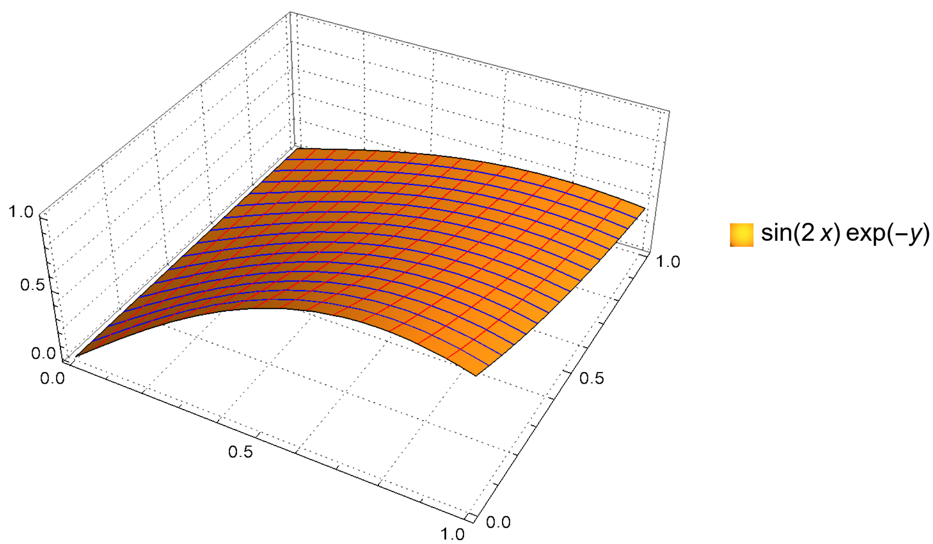
Plot the multivariable function below in the range of x and y from 0 to 1.

In[◦]:=

If you use Matlab, you might want to use the function `pcolor` combined with `shading flat`. Remember to differentiate between * and .*.

$$Z = \sin(2 * x) * \exp(-y) \tag{2}$$

In[]:=
```
Plot3D[Sin[2 x] * Exp[-y], {x, 0, 1}, {y, 0, 1},
  MeshStyle → {Red, Blue}, PlotTheme → "Detailed"]
```

Out[]=


6 Problem 6: Ordinary Differential Equations–15 pt

Solve the ODE:

$$\frac{dy}{dt} = y^{-2} \tag{3}$$

In[]:=

subject to the condition that y = 1 at t=0. Plot the result of the ODE from $t = 0$ to $t = 5$.

Use Euler's method to solve this problem. In Euler's method, we break up time into discrete chunks dt, compute the derivative $\frac{dy}{dt}|_t$ at each time, and then increase time slowly. This can be represented as

$$y(t + \Delta t) = y(t) + \Delta t\frac{dy}{dt}|_t \tag{4}$$

In[]:=
```
fun = NDSolve[{y''[t] == -y[t], y[0] == 1, y'[0] == 0}, y, {t, 0, 1},
   Method → {"TimeIntegration" → "ExplicitEuler"}, InterpolationOrder → All]
```

Out[]=
```
{{y → InterpolatingFunction[ Domain: {{0., 1.}} Output: scalar ]}}
```
Data not in notebook; Store now »

In[ ]:=
```
interpFN = y /. First@sol
Plot[interpFN[x], {x, 0, 5}]
```

Out[ ]= InterpolatingFunction[Domain: {{0., 5.}} Output: scalar]

Data not in notebook; Store now »

Out[ ]=