Problem 2

To solve this problem we made a number of functions in the numericalIntegrals.py file.

First we made the generateX function which takes as arguments the number of points you want and both a start and finish point. It then returns a list the length specified in the arguments of evenly spaced numbers staring at the start specified and ending at the ending specified in the arguments.

Next we made the trapezoidal function which takes an x list and y list and returns the result of the trapezoidal rule applied the (x[i],y[i]) points.

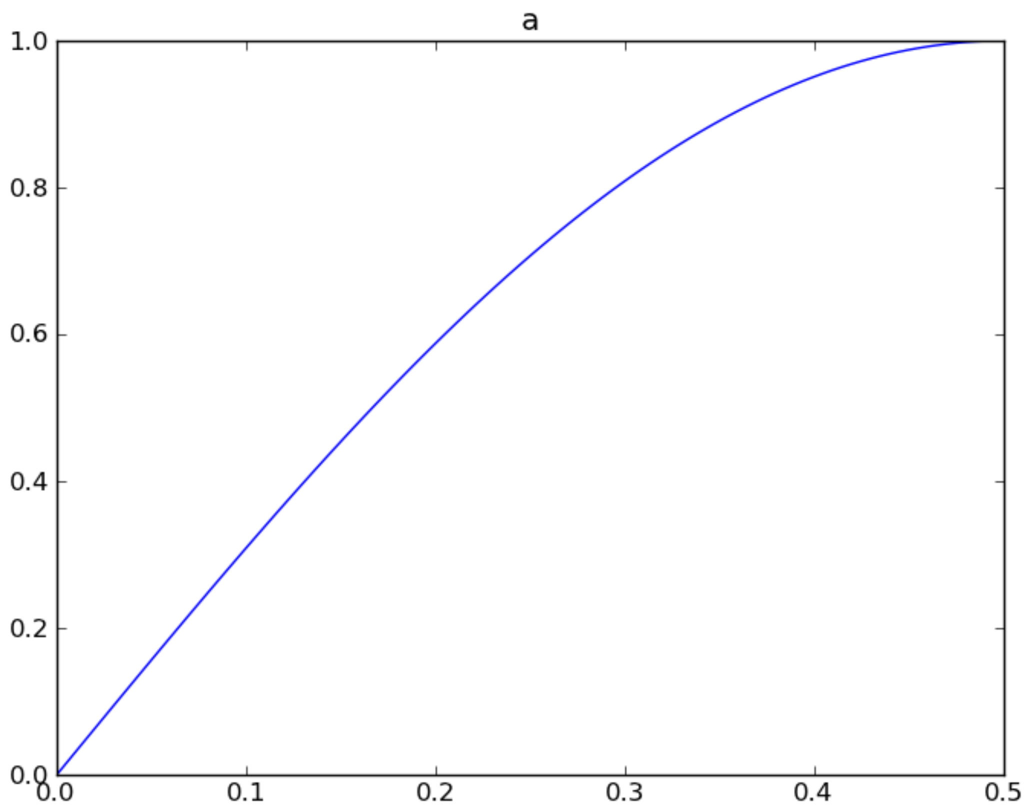
Then we made the romberg function which takes as parameters a start pointe, end point, the number of points we have, and the function whose integral we are approximating. Using these arguments and the generateX function we produce the necessary x and y lists and applied the trapezoidal function to them and return the result of romberg integration.

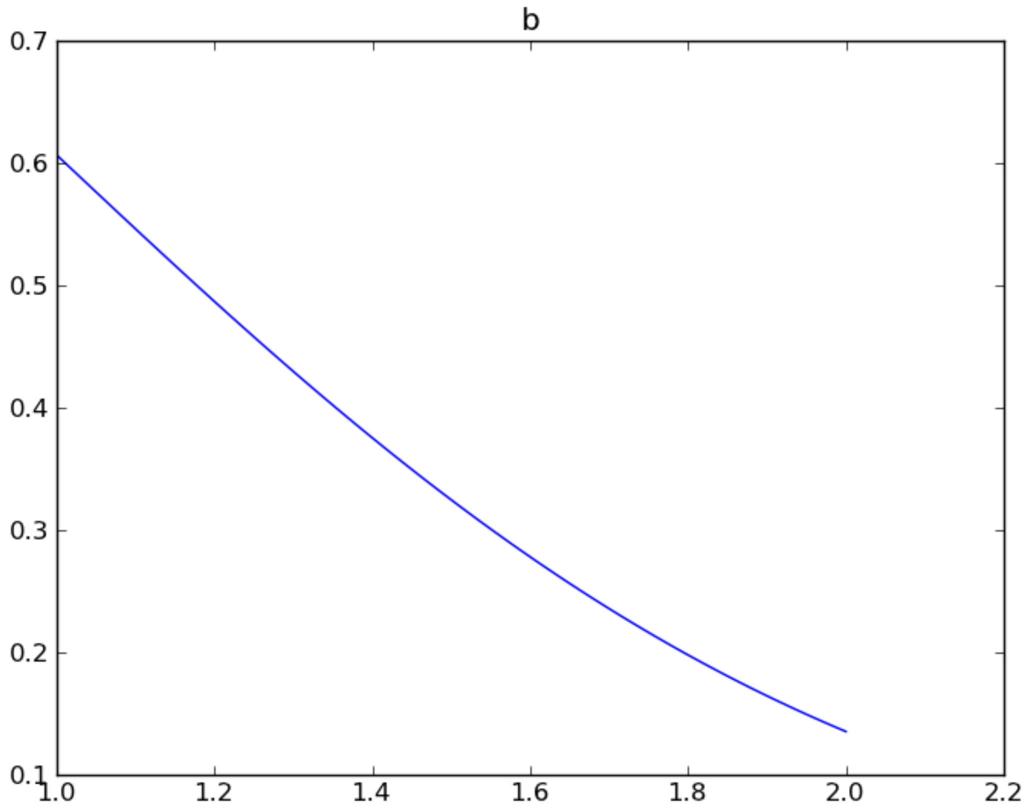
Finally we made the gaussQuadrature function which takes as parameters a and b representing the area we are integrating over, the function whose integral we are approximating, and the number of points to use. Within this function we define the t function which transforms x so we can apply the integral from -1,1 instead of from a,b. We then populate the x and w list with the specified number of gauss points and weights, create a list of the terms in the gauss quadrature function and return the sum of these terms.

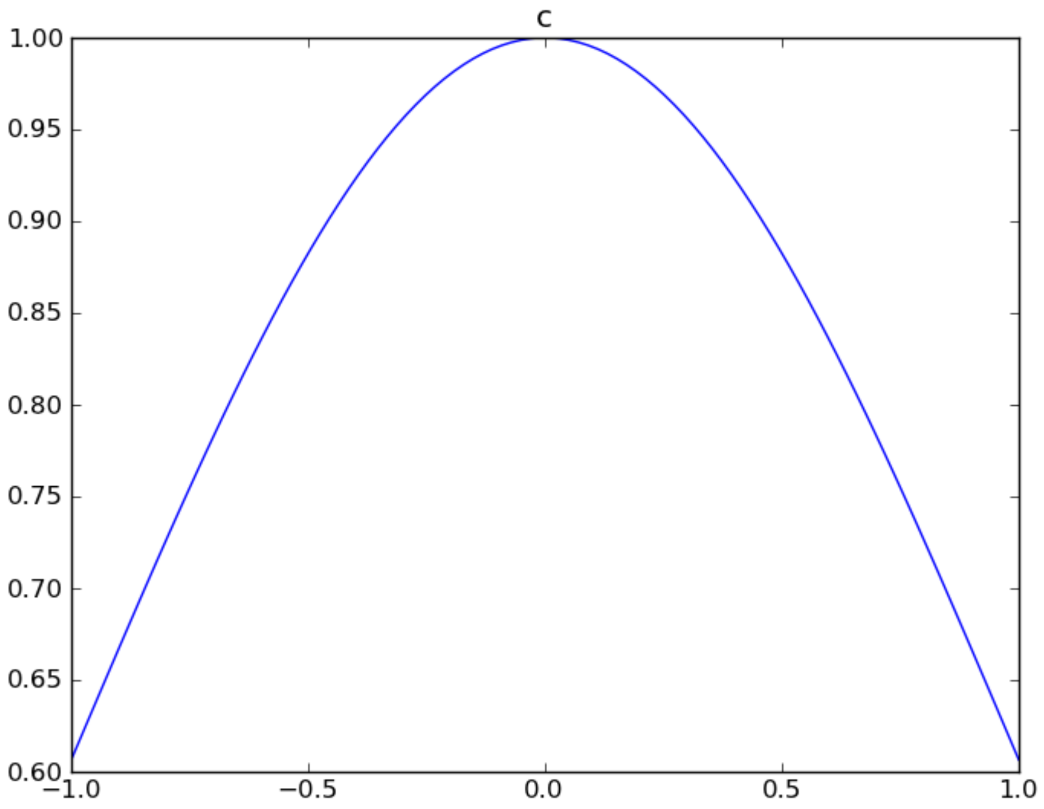
With these functions made we then used wolfram alpha to find the result of each integral to many decimal places and subtracted our approximations from this to see how accurate they where.

It was really surprising to see how many points were actually needed to get the trapezoidal and romberg approximations accurate to 7 decimal places. For part (a) needed over 5 million points, which was far from my initial test of 100. Romberg took over 1.5 million points which is a lot better but still huge. Given this data it was impressive to see Gaussian Quadrature achieving the same accuracy(actually better accuracy) with just 4 points. For part (b) the results where similar though Trapezoidal and Romberg only needed 1.5 million and 500 thousand respectively. In part (c) Trapezoidal and Romberg did considerably better and Gaussian Quadrature did a bit worse requiring 7 thousand, 310, and 9 points respectively. But even so Gaussian Quadratures 9 points vs. Rombergs 310 is a clear winner.

Though it was initially surprising that Trapezoidal and Romberg required so many points to be that accurate, after thinking more it does make sense. The integral is supposed to be the sum of the areas of infinitesimally thin trapezoids so a large number of points meaning thinner trapezoids would be necessary for accurate approximations. But I am still very impressed with how accurate Gaussian Quadrature can be with so few points.


a
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.1
0.2
0.3
0.4
0.5


b
0.7
0.6
0.5
0.4
0.3
0.2
0.1
1.0
1.2
1.4
1.6
1.8
2.0
2.2


c
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
−1.0
−0.5
0.0
0.5
1.0