Table of Contents

MTH 351 LAB 5 John Waczak

```
clear all;
format long;
```

1. Trapezoidal Rule

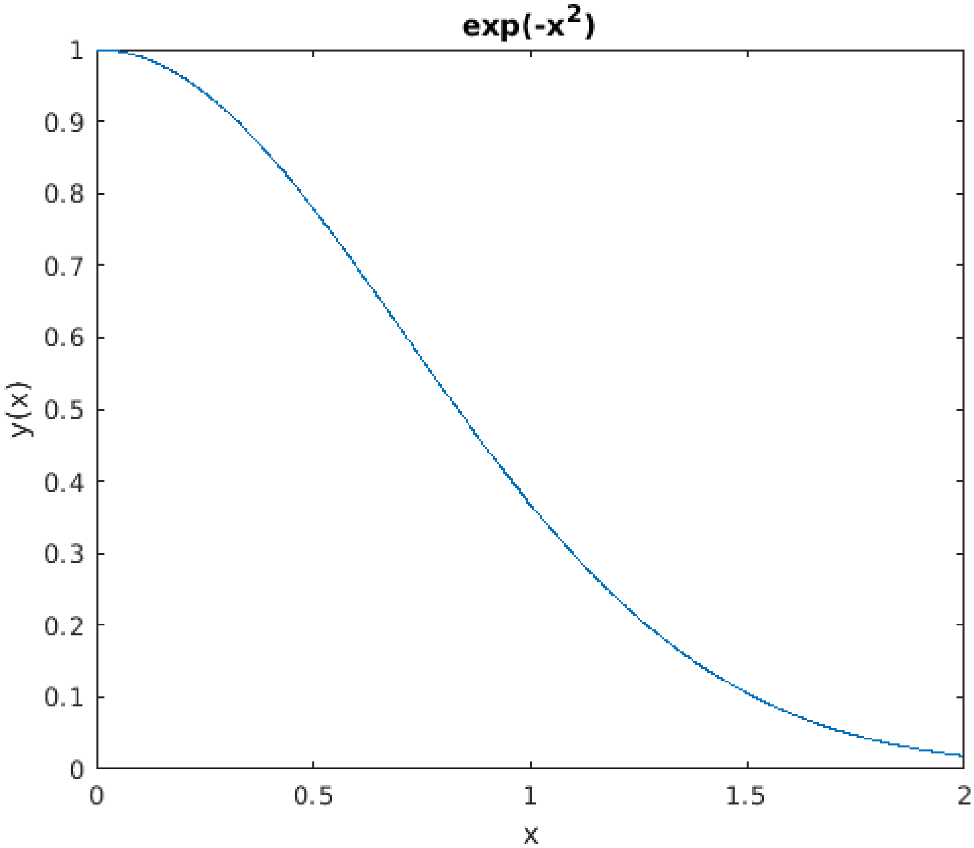
i. Integral of e^(-x^2) from 0 to 1

```
a = 0;
b = 2;
n0 = 2; % This will get us up to n=512 points in our grid
f = 'exp(-x^2)' ;
[inT, diT, raT] = trapezoidal(a,b,n0,f);
i = [1:1:9];
n = 2.^i;
n = transpose(n);
fprintf('n \t\t\tIntegral \t\t\t\t\t\tError \t\t\t\t\t\tRatio\n')
fprintf('%03d \t%0.12f \t%0.5e \t%0.5g\n', [n, inT, diT, raT]')



figure()
x = linspace(0,2,1000);
y = exp(-x.^2);
plot(x,y);
title("exp(-x^2)");
xlabel("x");
ylabel("y(x)");
```

```
n    Integral        Error        Ratio
002  0.877037260616  0.00000e+00  0
004  0.880618634125  3.58137e-03  0
008  0.881703791332  1.08516e-03  3.3003
016  0.881986245266  2.82454e-04  3.8419
032  0.882057557801  7.13125e-05  3.9608
064  0.882075429611  1.78718e-05  3.9902
128  0.882079900293  4.47068e-06  3.9976
256  0.882081018134  1.11784e-06  3.9994
512  0.882081297605  2.79471e-07  3.9998
```



ii. Integral of 1/(1+x^2) from 0 to 4

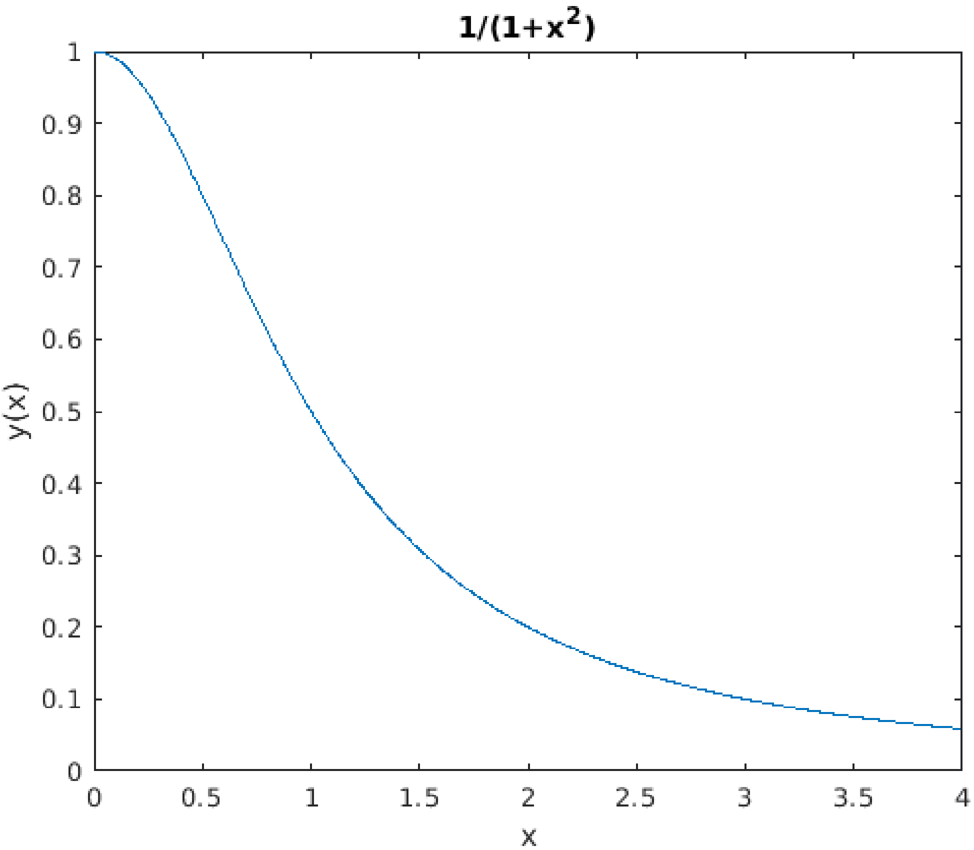
```
clear all;
a = 0;
b = 4;
n0 = 2; % This will get us up to n=512 points in our grid
f = '1/(1+x^2)' ;
[inT, diT, raT] = trapezoidal(a,b,n0,f);

i = [1:1:9];
n = 2.^i;
n = transpose(n);
fprintf('n \t\t\tIntegral \t\t\t\t\t\t\tError \t\t\t\t\t\t\tRatio\n')
fprintf('%03d \t%0.12f \t%0.5e \t%0.5g\n', [n, inT, diT, raT]')

figure()
x = linspace(0,4,1000);
y = 1./(1+x.^2);
plot(x,y);
title("1/(1+x^2)");
xlabel("x");
ylabel("y(x)");
```

```
n     Integral         Error          Ratio
002   1.458823529412   0.00000e+00    0
```

```
004  1.329411764706  1.29412e-01  0
008  1.325253402497  4.15836e-03  31.121
016  1.325673581733  4.20179e-04  9.8966
032  1.325781625682  1.08044e-04  3.889
064  1.325808653076  2.70274e-05  3.9976
128  1.325815410952  6.75788e-06  3.9994
256  1.325817100485  1.68953e-06  3.9998
512  1.325817522872  4.22387e-07  4
```



iii. Integral of 1/(2+sin(x)) from 0 to 2pi

```
clear all;
a = 0;
b = 2*pi;
n0 = 2; % This will get us up to n=512 points in our grid
f = '1/(2+sin(x))' ;
[inT, diT, raT] = trapezoidal(a,b,n0,f);

i = [1:1:9];
n = 2.^i;
n = transpose(n);
fprintf('n \t\t\tIntegral \t\t\t\t\t\tError \t\t\t\t\t\tRatio\n')
fprintf('%03d \t%0.12f \t%0.5e \t%0.5g\n', [n, inT, diT, raT]')

figure()
```

```
x = linspace(0,2*pi,1000);
y = 1./(2+sin(x));
plot(x,y);
title("1/(2+sin(x))");
xlabel("x");
ylabel("y(x)");
```

```
n    Integral        Error        Ratio
002  3.141592653590  0.00000e+00  0
004  3.665191429188  5.23599e-01  0
008  3.627791516645  3.73999e-02  14
016  3.627598733591  1.92783e-04  194
032  3.627598728468  5.12258e-09  37634
064  3.627598728468  0.00000e+00  Inf
128  3.627598728468  8.88178e-16  0
256  3.627598728468  8.88178e-16  1
512  3.627598728468  2.22045e-15  0.4
```

iv. Integral of sqrt(x) from 0 to 1

```
clear all;
a = 0;
b = 1;
n0 = 2; % This will get us up to n=512 points in our grid
f = 'sqrt(x)' ;
[inT, diT, raT] = trapezoidal(a,b,n0,f);

i = [1:1:9];
n = 2.^i;
n = transpose(n);
fprintf('n \t\t\tIntegral \t\t\t\t\t\t\tError \t\t\t\t\t\t\tRatio\n')
fprintf('%03d \t%0.12f \t%0.5e \t%0.5g\n', [n, inT, diT, raT]')

figure()
x = linspace(0,2*pi,1000);
y = sqrt(x);
plot(x,y);
title("sqrt(x)");
xlabel("x");
ylabel("y(x)");
```

b) Comment if the trapzoidal rule performed worse or better than expected for each integral. Explain what might be the cause.

For the first function, the trapezoidal rule appeared to work as expected. We should see error that goes like order 2 i.e. if we halve the grid spaceing, the error should quarter. Looking at the error column for that integral we see that this is roughly what happens for each iteration.

For the second funciton. The error begins quite large and then slowly trickles down to the 10e-7 precision for the final iteration of 512 points (same as first function). This is likely performing differently because of the dramatic change in slope over the interval. In the beginning, the slop is large and negative. Then after around x=2 the function is very shallow so any error over here will be small. The ratio column is almost 4 for all numbers of points though which suggests we still have order 2 convergence.

The third functon converges very quickly to an error on the order of 10^-15. In class we said that periodic functions behave really well with the trapezoid rule so this is probably an effect due to the periodic nature of the sine function that is composed within f. The ratio column is all over the place.

The final square root function appears to be converging linearly if we look at the error column. The ratio column confirms this with roughly 2 for all number of points.

2. Repeat 1 using Composite Simpson's rule. Compare to trapezoid rule

```
clear all;
a = 0;
b = 2;
n0 = 2; % This will get us up to n=512 points in our grid
f = 'exp(-x^2)' ;
[inS, diS, raS] = simpson(a,b,n0,f);
i = [1:1:9];
n = 2.^i;
n = transpose(n);
fprintf('n \t\t\tIntegral \t\t\t\t\t\t\tError \t\t\t\t\t\t\tRatio\n')
fprintf('%03d \t%0.12f \t%0.5e \t%0.5g\n', [n, inS, diS, raS]')
```

```
n    Integral        Error        Ratio
002  0.829944467858  0.00000e+00  0
004  0.881812425294  5.18680e-02  0
008  0.882065510401  2.53085e-04  204.94
016  0.882080396577  1.48862e-05  17.001
032  0.882081328646  9.32069e-07  15.971
064  0.882081386881  5.82343e-08  16.006
128  0.882081390520  3.63916e-09  16.002
256  0.882081390747  2.27440e-10  16.001
512  0.882081390761  1.42157e-11  15.999
```

Here we can see that for this function Simpson's rule converges much faster! It goes 4 orders of magnitude lower in the same number of points. Here we also verify that Simpson's rule is order 4 as the ratio column is consistantly right around 16 (i.e. 2^4)

```
clear all;
a = 0;
b = 4;
n0 = 2; % This will get us up to n=512 points in our grid
f = '1/(1+x^2)' ;
[inS, diS, raS] = simpson(a,b,n0,f);

i = [1:1:9];
n = 2.^i;
n = transpose(n);
fprintf('n \t\t\tIntegral \t\t\t\t\t\t\tError \t\t\t\t\t\t\tRatio\n')
fprintf('%03d \t%0.12f \t%0.5e \t%0.5g\n', [n, inS, diS, raS]')
```

```
n    Integral        Error        Ratio
002  1.239215686275  0.00000e+00  0
004  1.286274509804  4.70588e-02  0
```

```
008  1.323867281761  3.75928e-02  1.2518
016  1.325813641478  1.94636e-03  19.314
032  1.325817640332  3.99885e-06  486.73
064  1.325817662207  2.18759e-08  182.8
128  1.325817663577  1.36926e-09  15.976
256  1.325817663662  8.56231e-11  15.992
512  1.325817663668  5.35216e-12  15.998
```

We can see that this method is also much better than the trapezoid rule for this function. Again the ratio column indicates that we have 4th order convergence. It does take more points than for the first function to reach this rate-- just like with the trapezoid rule.

```
clear all;
a = 0;
b = 2*pi;
n0 = 2; % This will get us up to n=512 points in our grid
f = '1/(2+sin(x))' ;
[inS, diS, raS] = simpson(a,b,n0,f);

i = [1:1:9];
n = 2.^i;
n = transpose(n);
fprintf('n \t\t\tIntegral \t\t\t\t\t\t\tError \t\t\t\t\t\t\tRatio\n')
fprintf('%03d \t%0.12f \t%0.5e \t%0.5g\n', [n, inS, diS, raS]')
```

```
n    Integral        Error        Ratio
002  3.141592653590  0.00000e+00  0
004  3.839724354388  6.98132e-01  0
008  3.615324879131  2.24399e-01  3.1111
016  3.627534472573  1.22096e-02  18.379
032  3.627598726761  6.42542e-05  190.02
064  3.627598728468  1.70753e-09  37630
128  3.627598728468  8.88178e-16  1.9225e+06
256  3.627598728468  0.00000e+00  Inf
512  3.627598728468  2.66454e-15  0
```

For this function we again have the odd behavior that it is converging to 10e-15 within 2^9 points. Once again we cant really specify what order the convergence is as the error and ratio columns are all over the place but I suspect something about the sine function is making this converge rapidly.

```
clear all;
a = 0;
b = 1;
n0 = 2; % This will get us up to n=512 points in our grid
f = 'sqrt(x)' ;
[inS, diS, raS] = trapezoidal(a,b,n0,f);

i = [1:1:9];
n = 2.^i;
n = transpose(n);
fprintf('n \t\t\tIntegral \t\t\t\t\t\t\tError \t\t\t\t\t\t\tRatio\n')
fprintf('%03d \t%0.12f \t%0.5e \t%0.5g\n', [n, inS, diS, raS]')
```

```
n    Integral        Error        Ratio
002  0.603553390593  0.00000e+00  0
```

```
004   0.643283046243   3.97297e-02   0
008   0.658130221624   1.48472e-02   2.6759
016   0.663581196877   5.45098e-03   2.7238
032   0.665558936279   1.97774e-03   2.7562
064   0.666270811379   7.11875e-04   2.7782
128   0.666525657297   2.54846e-04   2.7934
256   0.666616548977   9.08917e-05   2.8038
512   0.666648881550   3.23326e-05   2.8111
```

For this final function we have that the convergence is first oder again! Thus Simpson's method isn't any more efficient than the Trapezoid method for this problem.

3. Asymptotic Error Formula

By looking at the table for Simpson's rule we have that n= 128 has an error of 10e-9. Then for n = 256 we have error 10e-10. This seems to roughly agree with the statement that we need n=160 for an error of 10e-10. Similarly for the second integral we have that when n=256 the error is 10e-11 and when n=512 the error is 10e-12 so n = 360 seems to rougly agree with the asymptotic error formula.

4. Repeat 1 using the Gaussian Quadrature rule.

```
clear all;
a = 0;
b = 2;
n0 = 2; % This will get us up to n=512 points in our grid
f = 'exp(-x^2)' ;
[inG, diG, raG] = gausstable(a,b,n0,f);
i = [1:1:9];
n = 2.^i;
n = transpose(n);
fprintf('n \t\t\tIntegral \t\t\t\t\t\t\tError \t\t\t\t\t\t\tRatio\n')
fprintf('%03d \t%0.12f \t%0.5e \t%0.5g\n', [n, inG, diG, raG]')
```

```
n     Integral         Error         Ratio
002   0.919486116641   0.00000e+00   0
004   0.882229095933   3.72570e-02   0
008   0.882081390420   1.47706e-04   252.24
016   0.882081390762   3.42522e-10   4.3123e+05
032   0.882081390762   6.66134e-16   5.1419e+05
064   0.882081390762   4.44089e-16   1.5
128   0.882081390762   0.00000e+00   Inf
256   0.882081390762   0.00000e+00   NaN
512   0.882081390762   2.22045e-16   0
```

If we look at the output of the quadrature we see that there is extremely rapid convergence. With just 32 points we already have error at 10e-16 which is better than just about every method we tried with 512 points. As we said in class, this is like magic!

```
clear all;
a = 0;
b = 4;
```

```
n0 = 2; % This will get us up to n=512 points in our grid
f = '1/(1+x^2)' ;
[inG, diG, raG] = gausstable(a,b,n0,f);

i = [1:1:9];
n = 2.^i;
n = transpose(n);
fprintf('n \t\t\tIntegral \t\t\t\t\t\tError \t\t\t\t\t\tRatio\n')
fprintf('%03d \t%0.12f \t%0.5e \t%0.5g\n', [n, inG, diG, raG]')
```

```
n     Integral         Error         Ratio
002   1.349112426036   0.00000e+00   0
004   1.327713222795   2.13992e-02   0
008   1.325838869084   1.87435e-03   11.417
016   1.325817663720   2.12054e-05   88.391
032   1.325817663668   5.23941e-11   4.0473e+05
064   1.325817663668   8.88178e-16   58990
128   1.325817663668   0.00000e+00   Inf
256   1.325817663668   0.00000e+00   NaN
512   1.325817663668   4.44089e-16   0
```

For this functin, the Gaussian quadrature took one factor of 2 more points to reach the 10e-16 precision. This is till better than both the trapezoid and simpson's method.

```
clear all;
a = 0;
b = 2*pi;
n0 = 2; % This will get us up to n=512 points in our grid
f = '1/(2+sin(x))' ;
[inG, diG, raG] = gausstable(a,b,n0,f);

i = [1:1:9];
n = 2.^i;
n = transpose(n);
fprintf('n \t\t\tIntegral \t\t\t\t\t\tError \t\t\t\t\t\tRatio\n')
fprintf('%03d \t%0.12f \t%0.5e \t%0.5g\n', [n, inG, diG, raG]')
```

```
n     Integral         Error         Ratio
002   4.109480962483   0.00000e+00   0
004   3.679381279962   4.30100e-01   0
008   3.628039679738   5.13416e-02   8.3772
016   3.627600902211   4.38778e-04   117.01
032   3.627598728468   2.17374e-06   201.85
064   3.627598728468   6.79012e-13   3.2013e+06
128   3.627598728468   4.44089e-16   1529
256   3.627598728468   0.00000e+00   Inf
512   3.627598728468   1.33227e-15   0
```

Yet again this function is having the same problem. Quadrature is missbehaving and we have the same weird convergence to 10e-15 for 512 points.

```
clear all;
a = 0;
b = 1;
n0 = 2; % This will get us up to n=512 points in our grid
```

```
f = 'sqrt(x)' ;
[inG, diG, raG] = gausstable(a,b,n0,f);

i = [1:1:9];
n = 2.^i;
n = transpose(n);
fprintf('n \t\t\tIntegral \t\t\t\t\t\t\tError \t\t\t\t\t\t\tRatio\n')
fprintf('%03d \t%0.12f \t%0.5e \t%0.5g\n', [n, inG, diG, raG]')
```

```
n    Integral         Error         Ratio
002  0.673887338679  0.00000e+00  0
004  0.667827645375  6.05969e-03  0
008  0.666835580100  9.92065e-04  6.1082
016  0.666689631499  1.45949e-04  6.7974
032  0.666669667368  1.99641e-05  7.3105
064  0.666667050398  2.61697e-06  7.6287
128  0.666666715190  3.35208e-07  7.807
256  0.666666672768  4.24229e-08  7.9016
512  0.666666667432  5.33603e-09  7.9503
```

This one was really surprising to me. For the past two methods, this function converged at order 1. Here we have that the ratio is around 8 each time putting this closer to order 2 convergence. So Gaussian quadrature is still better for integrating this funciton but it isn't reaching machine epsilon as quickly as it did for the first two examples.

Published with MATLAB® R2017b