Lab 5

Table of Contents

David Merrick

1.

```
    fprintf('Trapezoidal Rule\n\n');
% The Trapezoidal Rule tends to converge quickly for periodic functions,
% because when integrating over one period, there are roughly as many
% sections of the graph that are concave up as concave down, so the errors
% cancel.
%       i.
        a=0;
        b=2;
        n0=2;
        f='exp(-x^2)';
        [inT,diT,raT]=trapezoidal(a,b,n0,f);
        fprintf('i. Function: %s\n', f);
        fprintf('n \tIntegral \tError \t\tRatio\n');
        for i=1:length(inT),
            fprintf('%d\t%0.12f\t%0.5e\t%g\n',n0*2^(i-1),inT(i),diT(i),raT(i))
        end
        fprintf('\n\n');

        % This is not a periodic function. The error converged roughly linearly,
        % as expected, with a ratio between 3 and 4.

%       ii.
        a=0;
        b=4;
        n0=2;
        f = '1/(1 + x.^2)';
        [inT,diT,raT]=trapezoidal(a,b,n0,f);
        fprintf('ii. Function: %s\n', f);
        fprintf('n \tIntegral \tError \t\tRatio\n');
        for i=1:length(inT),
            fprintf('%d\t%0.12f\t%0.5e\t%g\n',n0*2^(i-1),inT(i),diT(i),raT(i))
        end
        fprintf('\n\n');

        % The Trapezoidal Rule performed much better than expected for this
        % interval. This is not a periodic function, so for the error to
        % shrink by a ratio of 31 initially was unexpected. Eventually this
```

```
        % ratio did decrease to a roughly linear convergence rate of about
        % 4.


%       iii.
        a = 0;
        b = (2*pi);
        n0 = 2;
        f = '1/(2 + sin(x))';
        [inT,diT,raT]=trapezoidal(a,b,n0,f);
        fprintf('iii. Function: %s\n', f);
        fprintf('n \tIntegral \tError \t\tRatio\n');
        for i=1:length(inT),
            fprintf('%d\t%0.12f\t%0.5e\t%g\n',n0*2^(i-1),inT(i),diT(i),raT(i))
        end
        fprintf('\n\n');

        % This is a periodic function, so it converged very quickly, as
        % expected.

%       iv.
        a = 0;
        b = 1;
        n0 = 2;
        f = 'sqrt(x)';
        [inT,diT,raT]=trapezoidal(a,b,n0,f);
        fprintf('iv. Function: %s\n', f);
        fprintf('n \tIntegral \tError \t\tRatio\n');
        for i=1:length(inT),
            fprintf('%d\t%0.12f\t%0.5e\t%g\n',n0*2^(i-1),inT(i),diT(i),raT(i))
        end
        fprintf('\n\n');

        % This is not a periodic function. The error converged roughly linearly,
        % as expected, with a ratio of about 1.75
```

```
Trapezoidal Rule

i. Function: exp(-x^2)
n  Integral  Error   Ratio
2 0.877037260616 0.00000e+00 0
4 0.880618634125 3.58137e-03 0
8 0.881703791332 1.08516e-03 3.30033
16 0.881986245266 2.82454e-04 3.84189
32 0.882057557801 7.13125e-05 3.96079
64 0.882075429611 1.78718e-05 3.99022
128 0.882079900293 4.47068e-06 3.99756
256 0.882081018134 1.11784e-06 3.99939
512 0.882081297605 2.79471e-07 3.99985


ii. Function: 1/(1 + x.^2)
n  Integral  Error   Ratio
2 1.458823529412 0.00000e+00 0
```

```
4 1.329411764706 1.29412e-01 0
8 1.325253402497 4.15836e-03 31.1208
16 1.325673581733 4.20179e-04 9.89664
32 1.325781625682 1.08044e-04 3.88897
64 1.325808653076 2.70274e-05 3.99757
128 1.325815410952 6.75788e-06 3.99939
256 1.325817100485 1.68953e-06 3.99985
512 1.325817522872 4.22387e-07 3.99996


iii. Function: 1/(2 + sin(x))
n  Integral  Error   Ratio
2 3.141592653590 0.00000e+00 0
4 3.665191429188 5.23599e-01 0
8 3.627791516645 3.73999e-02 14
16 3.627598733591 1.92783e-04 194
32 3.627598728468 5.12258e-09 37634
64 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. Function: sqrt(x)
n  Integral  Error   Ratio
2 0.603553390593 0.00000e+00 0
4 0.643283046243 3.97297e-02 0
8 0.658130221624 1.48472e-02 2.67591
16 0.663581196877 5.45098e-03 2.72376
32 0.665558936279 1.97774e-03 2.75616
64 0.666270811379 7.11875e-04 2.77821
128 0.666525657297 2.54846e-04 2.79335
256 0.666616548977 9.08917e-05 2.80384
512 0.666648881550 3.23326e-05 2.81115
```

2.

Simpson's Rule is precise for polynomials of degree 3 or less.

```
        fprintf('Simpsons Rule\n\n');
%       i.
        a=0;
        b=2;
        n0=2;
        f='exp(-x^2)';
        fprintf('i. Function: %s\n', f);
        [inS,diS,raS]=simpson(a,b,n0,f);
        fprintf('n \tIntegral \tError \t\tRatio\n')
        for i=1:length(inS),
            fprintf('%d\t%0.12f\t%0.5e\t%g\n',n0*2^(i-1),inS(i),diS(i),raS(i))
        end
```

```
        fprintf('\n\n');

        % The error after an n of 512 using Trapezoidal was 2.79471e-07.
        % Using Simpson's, it was 1.42157e-11. This means Simpson's was
        % much more accurate on approximating this integral than
        % Trapezoidal. Simpson's was also much more efficient in converging
        % to this error value; initially it had a ratio of almost 205 and
        % continued with a ratio of between 15 and 17, as compared with
        % Trapezoidal on the same function with a steady ratio of about 4.

%       ii.
        a=0;
        b=4;
        n0=2;
        f = '1/(1 + x.^2)';
        fprintf('ii. Function: %s\n', f);
        [inS,diS,raS]=simpson(a,b,n0,f);
        fprintf('n \tIntegral \tError \t\tRatio\n')
        for i=1:length(inS),
            fprintf('%d\t%0.12f\t%0.5e\t%g\n',n0*2^(i-1),inS(i),diS(i),raS(i))
        end
        fprintf('\n\n');

        % The error after an n of 512 using Trapezoidal was 4.22387e-07.
        % Using Simpson's, it was 5.35216e-12. This means Simpson's was
        % much more accurate on approximating this integral than
        % Trapezoidal. Simpson's was also much more efficient in converging
        % to this error value; It had ratios of 486.729, 182.797, and
        % 19.3144, as compared with the highest ratio generated by
        % Trapezoidal of 31.1208.

%       iii.
        a = 0;
        b = (2*pi);
        n0 = 2;
        f = '1/(2 + sin(x))';
        fprintf('iii. Function: %s\n', f);
        [inS,diS,raS]=simpson(a,b,n0,f);
        fprintf('n \tIntegral \tError \t\tRatio\n')
        for i=1:length(inS),
            fprintf('%d\t%0.12f\t%0.5e\t%g\n',n0*2^(i-1),inS(i),diS(i),raS(i))
        end
        fprintf('\n\n');

        % The error after an n of 512 using Trapezoidal was 2.22045e-15.
        % Using Simpson's, it was 2.66454e-15. This means that the two
        % approximation methods performed about the same on this integral.
        % In terms of efficiency, they were also both about the same. They
        % both had errors of 0 after a certain n. Trapezoidal had a ratio
        % of 37634 at an n of 32 while Simpson's had a ratio of 37630 at an
        % n of 64.

%       iv.
        a = 0;
```

```
b = 1;
n0 = 2;
f = 'sqrt(x)';
fprintf('iv. Function: %s\n', f);
[inS,diS,raS]=simpson(a,b,n0,f);
fprintf('n \tIntegral \tError \t\tRatio\n')
for i=1:length(inS),
    fprintf('%d\t%0.12f\t%0.5e\t%g\n',n0*2^(i-1),inS(i),diS(i),raS(i))
end
fprintf('\n\n');

% The error after an n of 512 using Trapezoidal was 3.23326e-05.
% Using Simpson's, it was 1.28129e-05. This means Simpson's was
% slightly more accurate on approximating this integral than
% Trapezoidal. In terms of efficiency, the two methods both converged at a
% rate. The ratio of both tended to be about 2.8.
```

```
Simpsons Rule

i. Function: exp(-x^2)
n  Integral  Error   Ratio
2 0.829944467858 0.00000e+00 0
4 0.881812425294 5.18680e-02 0
8 0.882065510401 2.53085e-04 204.943
16 0.882080396577 1.48862e-05 17.0014
32 0.882081328646 9.32069e-07 15.9711
64 0.882081386881 5.82343e-08 16.0055
128 0.882081390520 3.63916e-09 16.0021
256 0.882081390747 2.27440e-10 16.0006
512 0.882081390761 1.42157e-11 15.9992


ii. Function: 1/(1 + x.^2)
n  Integral  Error   Ratio
2 1.239215686275 0.00000e+00 0
4 1.286274509804 4.70588e-02 0
8 1.323867281761 3.75928e-02 1.25181
16 1.325813641478 1.94636e-03 19.3144
32 1.325817640332 3.99885e-06 486.729
64 1.325817662207 2.18759e-08 182.797
128 1.325817663577 1.36926e-09 15.9764
256 1.325817663662 8.56231e-11 15.9918
512 1.325817663668 5.35216e-12 15.9978


iii. Function: 1/(2 + sin(x))
n  Integral  Error   Ratio
2 3.141592653590 0.00000e+00 0
4 3.839724354388 6.98132e-01 0
8 3.615324879131 2.24399e-01 3.11111
16 3.627534472573 1.22096e-02 18.3789
32 3.627598726761 6.42542e-05 190.02
64 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

iv. Function: sqrt(x)
n Integral Error Ratio
2 0.638071187458 0.00000e+00 0
4 0.656526264793 1.84551e-02 0
8 0.663079280085 6.55302e-03 2.81627
16 0.665398188628 2.31891e-03 2.82591
32 0.666218182746 8.19994e-04 2.82796
64 0.666508103078 2.89920e-04 2.82834
128 0.666610605936 1.02503e-04 2.82841
256 0.666646846203 3.62403e-05 2.82842
512 0.666659659074 1.28129e-05 2.82843

3.

For integral i, yes, my results agree with the asymptotic error formula. At an n of 256, the error is 2.27440e-10. The asymptotic error formula for Simpson's predicts that an accuracy of 10e-10 requires at least 160 subdivisions for this integral. As the error and n are both greater than the prediction, my results agree.

For integral ii, my results also agree with the asymptotic error formula. At 512 subdivisions, the error of this integral is 5.35216e-12. The asymptotic error formula predicts that at least 396 subdivisions are required for an error of 10e-12. As the error and n are both greater than this prediction, my results agree with it.

4.

The degree of precision of the Quadrature Rule is the largest integer m such that the rule exactly integrates all polynomials of degree less than or equal to m. There is no error estimate formula for the Gaussian Quadrature Rule, but the error convergence is generally better than O(h^n).

```
fprintf('Gaussian Quadrature Rule\n\n');
%       i.
    a=0;
    b=2;
    n0=2;
    f='exp(-x^2)';
    fprintf('i. Function: %s\n', f);
    [inG,diG,raG]=gausstable(a,b,n0,f);
    fprintf('n \tIntegral \tError \t\tRatio\n')
    for i=1:length(inG),
        fprintf('%d\t%0.12f\t%0.5e\t%g\n',n0*2^(i-1),inG(i),diG(i),raG(i))
    end
    fprintf('\n\n');

    % The error after an n of 512 using Trapezoidal was 2.79471e-07.
    % Using Simpson's, it was 1.42157e-11. Using Gaussian Quadrature, it was 1.221
    % This means Gaussian Quadrature was the most accurate of the three
    % methods on approximating this integral by several orders of magnitude. It wa
```

```
    % efficient at converging to this error value. After an n of just 16,
    % it had a ratio of 431229 followed by a ratio of 205678 at an n of 32.
    % Compare this to the Simpson ratio of about 205 at an n of 8, followed by a s
    % between 15 and 17 and the Trapezoidal with a steady ratio of about 4.

%       ii.
    a=0;
    b=4;
    n0=2;
    f = '1/(1 + x.^2)';
    fprintf('ii. Function: %s\n', f);
    [inG,diG,raG]=gausstable(a,b,n0,f);
    fprintf('n \tIntegral \tError \t\tRatio\n')
    for i=1:length(inG),
        fprintf('%d\t%0.12f\t%0.5e\t%g\n',n0*2^(i-1),inG(i),diG(i),raG(i))
    end
    fprintf('\n\n');

    % The error after an n of 512 using Trapezoidal was 4.22387e-07.
    % Using Simpson's, it was 5.35216e-12. Using Gaussian Quadrature, it was 1.998
    % This means Gaussian Quadrature was the most accurate of the three methods on
    % Gaussian Quadrature was also much more efficient in converging
    % to this error value; It had a ratio of 404721 at an n of 32, followed by a r
    % at an n of 64. Compare this with the highest ratios achieved by Simpson's of
    % 19.3144, and  the highest ratio generated by Trapezoidal of 31.1208.

%       iii.
    a = 0;
    b = (2*pi);
    n0 = 2;
    f = '1/(2 + sin(x))';
    fprintf('iii. Function: %s\n', f);
    [inG,diG,raG]=gausstable(a,b,n0,f);
    fprintf('n \tIntegral \tError \t\tRatio\n')
    for i=1:length(inG),
        fprintf('%d\t%0.12f\t%0.5e\t%g\n',n0*2^(i-1),inG(i),diG(i),raG(i))
    end
    fprintf('\n\n');

    % The error after an n of 512 using Trapezoidal was 2.22045e-15.
    % Using Simpson's, it was 2.66454e-15. Using Gaussian Quadrature, it was 6.661
    % This means that Gaussian Quadrature was slighly less efficient than
    % Simpson's and Trapezoidal on approximating this integral.
    % In terms of efficiency, all three methods performed about the same.

%       iv.
    a = 0;
    b = 1;
    n0 = 2;
    f = 'sqrt(x)';
    fprintf('iv. Function: %s\n', f);
    [inG,diG,raG]=gausstable(a,b,n0,f);
    fprintf('n \tIntegral \tError \t\tRatio\n')
    for i=1:length(inG),
```

```
    fprintf('%d\t%0.12f\t%0.5e\t%g\n',n0*2^(i-1),inG(i),diG(i),raG(i))
end
fprintf('\n\n');


% The error after an n of 512 using Trapezoidal was 3.23326e-05.
% Using Simpson's, it was 1.28129e-05. Using Gaussian Quadrature, it was 5.336
% This means Gaussian Quadrature was the most accurate of the three at approxi
% by several orders of magnitude.
% In terms of efficiency, they all converged linearly. The ratio of Simpson's
% while Gaussian Quadrature averaged between 6 and 8.
```

```
Gaussian Quadrature Rule

i. Function: exp(-x^2)
n  Integral  Error   Ratio
2 0.919486116641 0.00000e+00 0
4 0.882229095933 3.72570e-02 0
8 0.882081390420 1.47706e-04 252.239
16 0.882081390762 3.42522e-10 431229
32 0.882081390762 1.66533e-15 205678
64 0.882081390762 1.11022e-15 1.5
128 0.882081390762 1.11022e-16 10
256 0.882081390762 7.77156e-16 0.142857
512 0.882081390762 1.22125e-15 0.636364


ii. Function: 1/(1 + x.^2)
n  Integral  Error   Ratio
2 1.349112426036 0.00000e+00 0
4 1.327713222795 2.13992e-02 0
8 1.325838869084 1.87435e-03 11.4168
16 1.325817663720 2.12054e-05 88.3905
32 1.325817663668 5.23950e-11 404721
64 1.325817663668 1.33227e-15 39327.7
128 1.325817663668 4.44089e-16 3
256 1.325817663668 4.44089e-16 1
512 1.325817663668 1.99840e-15 0.222222


iii. Function: 1/(2 + sin(x))
n  Integral  Error   Ratio
2 4.109480962483 0.00000e+00 0
4 3.679381279962 4.30100e-01 0
8 3.628039679738 5.13416e-02 8.37722
16 3.627600902211 4.38778e-04 117.011
32 3.627598728468 2.17374e-06 201.853
64 3.627598728468 6.85674e-13 3.17023e+06
128 3.627598728468 5.32907e-15 128.667
256 3.627598728468 1.33227e-15 4
512 3.627598728468 6.66134e-15 0.2


iv. Function: sqrt(x)
```

```
n  Integral  Error   Ratio
2 0.673887338679 0.00000e+00 0
4 0.667827645375 6.05969e-03 0
8 0.666835580100 9.92065e-04 6.10816
16 0.666689631499 1.45949e-04 6.79736
32 0.666669667368 1.99641e-05 7.31054
64 0.666667050398 2.61697e-06 7.62872
128 0.666666715190 3.35208e-07 7.80701
256 0.666666672768 4.24229e-08 7.90157
512 0.666666667432 5.33603e-09 7.95028
```

Published with MATLAB® 8.0