Solution to Programming Assignment Six – Numerical Integration

1. Using MATLAB

Summarize your MATLAB results from parts a and b below as a table with six columns: (1) upper limit, z, (2) actual error function, (3) error function from your Romberg function,(4) error function from MATLAB quad function, (5) absolute value of difference between columns 2 and 3, (6) absolute difference between columns 2 and 4. Set an appropriate format to show the errors.

a. Write a function myInt(f, a, b) that accepts a function handle, f, for the integrand, f(x), and the upper and lower limits of the integral and returns a value for the integral using Romberg integration. Your function should allow the value of N to be doubled no more than 20 times and should exit when two approximations to the integral agree to within a value of $10^{-10}$. Use this function to compute error function integral, erf(z) for a range of z values from 0.01 to 10. Use the MATLAB logspace function to generate these values. Note that some MATLAB functions for integrands must be able to process array operations.

The romberg function and the integrand function are shown below.

```matlab
function I = myInt( f,a,b )
% ROMBERG for Romberg integration of definite integrals
% This program computes a numerical estimate of the integral of f(x)dx
% between x = a and x = b

% Before calling this program the user must write a function that
% computes the integrand f(x) for array arguments. For example, the
% function for finding the integral of x^2 would have the following form:
% function y = testIntegrand(x)
%       y = x.^2;
% end
% Note the use of the semicolon after x.^2 to avoid printed output.

% To use the romberg function with testIntegrand(x) the following call is
% used: romberg(@testIntegrand, 0, 1). This gives the integral of x^2
% between x = 0 and x = 1.

% Initialization of program variables
maxIter = 20;
maxRelErr = 1e-10;
converged = 0;
k = 1;
T = zeros(maxIter,maxIter);

% Compute initial trapezoid rule, T(1,1), for h = (b-a)/2

N = 2;
h = (b-a)/2;
T(1,1) = h * (f(a) + 2 * f(a+h) + f(b))/2;
```

College of Engineering and Computer Science
Mechanical Engineering Department

Mechanical Engineering 309
Numerical Analysis of Engineering Systems
Spring 2014  Number: 15237  Instructor: Larry Caretto
while ~converged && k <= maxIter;
    %Start Romberg loop by doubling N and getting new Trapezoid value
    k = k + 1;
    N = 2 * N;
    h = (b - a)/N;
    sumOdd = 0;
    for m = 1:2:(N-1);
        sumOdd = sumOdd + f(a + m*h);
    end
    T(k,1) = T(k-1,1)/2 + h * sumOdd;
    %Romberg algorithm using Richardson expansion
    for m = 2:k;
        T(k,m) = (4^(m-1) * T(k,m-1) - T(k-1,m-1)) / (4^(m-1) - 1);
    end
    converged = abs(T(k,k) - T(k,k-1)) <= maxRelErr * abs(T(k,k));
end
%See if loop exit converged or exceed allowed iterations
if converged
    I = T(k,k);
else
    I = 'Error: no convergence in Romberg';
end

function f = expx2(x)
    f = exp(-x.*x)
end

b. Review the help file for the MATLAB function quad. Use this function to compute the error, erf(z) function integral for the same range of z values that you used in part a.

The MATLAB commands to compute the error function integral for a range of upper limit values taken from the logspace function as shown below. This set of commands obtains all the column data requested at the start of the assignment. (The table headings were added after the output from the command that combined all the column arrays into a single matrix.)

```
>> z = logspace(-2,1,25);
>> matlabERF = erf(z);
>> quadERF = 2/sqrt(pi)*quad(@errFintegral,0,z(1));
>> for k = 2:25 ; quadERF = [quadERF; 2/sqrt(pi)*quad(@errFintegral,0,z(k))]; end
>> format short e
>> rombergERF = 2/sqrt(pi)*romberg(@errFintegral,0,z(1));
>> for k = 2:25 ; rombergERF = [rombergERF; 2/sqrt(pi)*romberg(@errFintegral,0,z(k))]; end
>> r=[z' matlabERF rombergERF quadERF abs(matlabERF - rombergERF) abs(matlabERF - quadERF)]
>> z    MATLAB   Romberg   quad      MATLAB ERF - MATLAB ERF
   erf(z)    erf(z)    erf(z)  romberg ERF - quad ERF
1.0000e-002 1.1283e-002 1.1283e-002 1.1283e-002 1.7347e-018 1.7347e-018
Function Romberg(a As Double, b As Double, Optional maxRelErr As Double = 0.000000000001, Optional maxDoublings As Integer = 21) As Variant

'Function for numerical evaluation of integral from a to b of f(x)dx using simpson 's rule
'Rule is applied repeatedly with number of panels doubled with each repetition until two successive values have desired relative error
'Value of integral is returned in function name
'User has to code integrand as part of function f, discussed further below.
'If desired accuracy is not achieved an error message is returned in functio name
'Input variables
' a = lower limit of integral
' b = upper limit of integral
' maxRelError = relative error 1e-8 unless other value supplied by user;
' Calculation is converged when relative difference between two

Option Explicit
Option Base 0

Option Explicit
'successive evaluations (with N and 2N) panels is less than this
'    maxDoublings = 32 unless other value supplied by user - sets limit on number
' of times the number of panels is doubled

'The function below is an example of an integrand function required for this code.
Function f(x As Double) As Double
    f = Exp(-x * x)
End Function

'This function is used to find the error function (after multiplication by.
' 2/sqrt(pi). This function must be rewritten for each new integral

Dim h As Double            'Step size for integration
Dim N As Long              'Current number of steps
Dim i As Long              'Loop index for initial trapezoid rule
Dim doubling As Integer    'Loop index for doubling
Dim x As Double            'Argument of integrand
Dim T() As Double          'Array for Romberg integration (redimensioned below)
Dim p As Variant           'Parameter array to pass to integrand (if present)

ReDim T(0 To maxDoublings, 0 To maxDoublings) As Double

'Compute initial step size and trapezoid approximation for two steps
N = 2
h = (b - a) / N
T(0, 0) = (f(a) + 2 * f(a + h) + f(b)) * h / 2

'Start step number doubling for Romberg algorithm
For doubling = 1 To maxDoublings
    'Compute new step size and trapezoid rule approximation
    N = 2 * N
    h = (b - a) / N
    T(doubling, 0) = 0
    For i = 1 To N - 1 Step 2
        T(doubling, 0) = T(doubling, 0) + f(a + i * h)
    Next i
    T(doubling, 0) = (T(doubling - 1, 0) + 2 * h * T(doubling, 0)) / 2

    'Get Romberg improvements
    For i = 1 To doubling
        T(doubling, i) = (4 ^ i * T(doubling, i - 1) - T(doubling - 1, i - 1)) / (4 ^ i - 1)
    Next i

    'Check convergence
    If (Abs(T(doubling, doubling) - T(doubling, doubling - 1)) <= 
        maxRelErr * Abs(T(doubling, doubling))) Then
        Romberg = T(doubling, doubling)
        Exit Function
    End If
Next doubling

'Return error message if no convergence
Romberg = "No convergence in Romberg iteration"

End Function
Function f(x As Double) As Double
    f = Exp(-x * x)
End Function

The workbook table shown below was used to call the Excel error function and the Romberg function
(with the integrand function to compute exp(-x^2)) for different values of the upper limit z. In the plot of
these results, there is no difference between the Romberg results and the Excel worksheet erf function.
(The plot appears to show only the Romberg result, because it is the second one calculated.)
The relative error in the Romberg calculation, as opposed to the Excel calculation shows a wide scatter. This is probably due to the fact that the Romberg integration results will differ in accuracy depending on how close the error in the final converged Romberg integral is to the error tolerance. If the next-to-last iteration has an error that is only slightly greater than the error tolerance; the final iteration will be more accurate than a solution where the error tolerance is just barely met on the final Romberg iteration.

<table>
<thead>
<tr>
<th>z</th>
<th>Excel erf(z)</th>
<th>Romberg erf(z)</th>
<th>Relative Error</th>
<th>z</th>
<th>Excel erf(z)</th>
<th>Romberg erf(z)</th>
<th>Relative Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>0.011283</td>
<td>0.011283</td>
<td>3.07482E-16</td>
<td>0.316628</td>
<td>0.345279</td>
<td>0.345279</td>
<td>1.60772E-16</td>
</tr>
<tr>
<td>0.013335</td>
<td>0.015046</td>
<td>0.015046</td>
<td>3.45877E-16</td>
<td>0.421697</td>
<td>0.449071</td>
<td>0.449071</td>
<td>3.7084E-16</td>
</tr>
<tr>
<td>0.017783</td>
<td>0.020064</td>
<td>0.020064</td>
<td>3.45845E-16</td>
<td>0.562341</td>
<td>0.573544</td>
<td>0.573544</td>
<td>1.93572E-16</td>
</tr>
<tr>
<td>0.023714</td>
<td>0.026753</td>
<td>0.026753</td>
<td>1.00E-16</td>
<td>0.749894</td>
<td>0.711088</td>
<td>0.711088</td>
<td>1.5613E-15</td>
</tr>
<tr>
<td>0.031623</td>
<td>0.035671</td>
<td>0.035671</td>
<td>1.94527E-16</td>
<td>1.00E-16</td>
<td>0.842701</td>
<td>0.842701</td>
<td>1.245E-13</td>
</tr>
<tr>
<td>0.04217</td>
<td>0.047555</td>
<td>0.047555</td>
<td>1.45913E-16</td>
<td>1.333521</td>
<td>0.940689</td>
<td>0.940689</td>
<td>2.36045E-16</td>
</tr>
<tr>
<td>0.056234</td>
<td>0.063387</td>
<td>0.063387</td>
<td>2.18939E-16</td>
<td>1.778279</td>
<td>0.988092</td>
<td>0.988092</td>
<td>7.86522E-15</td>
</tr>
<tr>
<td>0.074989</td>
<td>0.084458</td>
<td>0.084458</td>
<td>4.92947E-16</td>
<td>2.371374</td>
<td>0.999202</td>
<td>0.999202</td>
<td>7.77776E-16</td>
</tr>
<tr>
<td>0.1</td>
<td>0.112463</td>
<td>0.112463</td>
<td>1.23399E-16</td>
<td>3.162278</td>
<td>0.999992</td>
<td>0.999992</td>
<td>1.21015E-14</td>
</tr>
<tr>
<td>0.133352</td>
<td>0.149585</td>
<td>0.149585</td>
<td>3.71102E-16</td>
<td>4.216965</td>
<td>1</td>
<td>1</td>
<td>4.29434E-13</td>
</tr>
<tr>
<td>0.177828</td>
<td>0.198562</td>
<td>0.198562</td>
<td>1.11826E-15</td>
<td>5.623413</td>
<td>1</td>
<td>1</td>
<td>5.77316E-15</td>
</tr>
<tr>
<td>0.237137</td>
<td>0.262649</td>
<td>0.262649</td>
<td>1.00E-16</td>
<td>7.498942</td>
<td>1</td>
<td>1</td>
<td>2.22045E-15</td>
</tr>
<tr>
<td>0.316228</td>
<td>0.345279</td>
<td>0.345279</td>
<td>1.60772E-16</td>
<td>10</td>
<td>1</td>
<td>1</td>
<td>6.7637E-12</td>
</tr>
</tbody>
</table>