College of Engineering and Computer Science
Mechanical Engineering Department

California State University Northridge

Mechanical Engineering 309
Numerical Analysis of Engineering Systems

Spring 2014 Number: 15237 Instructor: Larry Caretto

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.

```
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;
```

Jacaranda Hall 3314 Mail Code Phone: N/A
Email: lcaretto@csun.edu 8348 Fax: 818.677.7062

```
    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
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 shorte
>> 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)]
r =
     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
```

1.3335e-002	1.5046e-002	1.5046e-002	1.5046e-002	0	0
1.7783e-002	2.0064e-002	2.0064e-002	2.0064e-002	3.4694e-018	0
2.3714e-002	2.6753e-002	2.6753e-002	2.6753e-002	0	3.4694e-018
3.1623e-002	3.5671e-002	3.5671e-002	3.5671e-002	3.4694e-017	6.9389e-018
4.2170e-002	4.7555e-002	4.7555e-002	4.7555e-002	2.6368e-016	7.6328e-017
5.6234e-002	6.3387e-002	6.3387e-002	6.3387e-002	0	5.4123e-016
7.4989e-002	8.4458e-002	8.4458e-002	8.4458e-002	2.7756e-017	4.0523e-015
1.0000e-001	1.1246e-001	1.1246e-001	1.1246e-001	0	3.0032e-014
1.3335e-001	1.4958e-001	1.4958e-001	1.4958e-001	5.5511e-017	2.2193e-013
1.7783e-001	1.9856e-001	1.9856e-001	1.9856e-001	2.7756e-016	1.6220e-012
2.3714e-001	2.6265e-001	2.6265e-001	2.6265e-001	2.9421e-015	1.1608e-011
3.1623e-001	3.4528e-001	3.4528e-001	3.4528e-001	3.5028e-014	7.9868e-011
4.2170e-001	4.4907e-001	4.4907e-001	4.4907e-001	3.6132e-013	5.0838e-010
5.6234e-001	5.7354e-001	5.7354e-001	5.7354e-001	4.4409e-016	2.7198e-009
7.4989e-001	7.1109e-001	7.1109e-001	7.1109e-001	1.1102e-015	8.2381e-009
1.0000e+000	8.4270e-001	8.4270e-001	8.4270e-001	1.0492e-013	5.4065e-008
1.3335e+000	9.4069e-001	9.4069e-001	9.4069e-001	4.9616e-013	3.2926e-008
1.7783e+000	9.8809e-001	9.8809e-001	9.8809e-001	3.2092e-012	8.3471e-008
2.3714e+000	9.9920e-001	9.9920e-001	9.9920e-001	2.4367e-012	1.4893e-008
3.1623e+000	9.9999e-001	9.9999e-001	9.9999e-001	5.1652e-011	1.3827e-008
4.2170e+000	1.0000e+000	1.0000e+000	1.0000e+000	4.2955e-013	1.8090e-007
5.6234e+000	1.0000e+000	1.0000e+000	1.0000e+000	2.4247e-011	5.8552e-008
7.4989e+000	1.0000e+000	1.0000e+000	1.0000e+000	1.6509e-010	1.3784e-008
1.0000e+001	1.0000e+000	1.0000e+000	1.0000e+000	6.7637e-012	9.9166e-007

The MATLAB quad function results use the default accuracy of that function. You can use the options variable in quad to use a more stringent accuracy.

2. Using EXCEL VBA

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

a. Write a function myInt(a, b) that 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.

The VBA code for the Romberg method is shown below.

```
Option Explicit
Option Base 0

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
```

```
    '                   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 improvemetts
        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.

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