```
clc
clear all

%**********************************************************************
%
% INPUTS
%
% Click the run button and refer to the command window
% These are the inputs that can be modified by the user
%
% f(x), the function to integrate

  f= @(x) 2000*log(1400/21./x)-9.8*x ;

% a, the lower limit of integration

  a=8 ;

% b, the upper limit of integration

  b=12 ;

% n, the maximum number of Gauss points, 1-10

  n=3 ;

%**********************************************************************

disp(sprintf('\n\nSimulation of Gaussian Quadrature'))
disp(sprintf('University of South Florida'))
disp(sprintf('United States of America'))
disp(sprintf('kaw@eng.usf.edu\n'))

disp(sprintf↙
('\n*****************************Introduction*********************************'))
disp('The two-point Gauss Quadrature Rule is an extension of the Trapezoidal Rule')
disp('approximation where the arguments of hte function are not predetermined as ')
disp('a and b but as unknowns x(1) and x(2).  In the two-point Gauss Quadrature ')
disp('Rule, the integral is approximated as the sum of c(x)*f(x) evaluated at x(1)')
disp('and x(2).  The four unknowns x(1), x(2), c(1), c(2) are found by assuming ')
disp('that the formula gives exact results for integrating a general third order ')
disp('polynomial. Higher point Gaussian Quadrature formula variable are found in a')
disp('similar manner by assuming expressions of higher order polynomials.')

disp(sprintf('\n\n*******************************Input↙
Data*********************************\n'))
disp(sprintf('     f(x), integrand function'))
disp(sprintf('     a = %g, lower limit of integration ',a))
disp(sprintf('     b = %g, upper limit of integration ',b))
disp(sprintf('     n = %g, number of Gauss Points, 1-10',n))
format short g
```

```
% Setup Gaussian Coefficients and Evaluation Points
x_values = zeros(10,10) ;
c_values = zeros(10,10) ;

i=1 ;
x_values(i,1) = 0.0 ;
c_values(i,1) = 2.0 ;

i=2 ;
x_values(i,1) = -0.5773502691896260 ;
x_values(i,2) = -x_values(i,1) ;
c_values(i,1) = 1.0 ;
c_values(i,2) = 1.0 ;

i=3 ;
x_values(i,1) = -0.7745966692414830 ;
x_values(i,2) = 0.0 ;
x_values(i,3) = -x_values(i,1) ;
c_values(i,1) = 0.5555555555555560 ;
c_values(i,2) = 0.8888888888888890 ;
c_values(i,3) =c_values(i,1) ;

i=4 ;
x_values(i,1) = -0.8611363115940530 ;
x_values(i,2) = -0.3399810435848560 ;
x_values(i,3) = -x_values(i,2) ;
x_values(i,4) = -x_values(i,1) ;
c_values(i,1) = 0.3478548451374540 ;
c_values(i,2) = 0.6521451548625460 ;
c_values(i,3) =c_values(i,2) ;
c_values(i,4) =c_values(i,1) ;

i=5 ;
x_values(i,1) = -0.9061798459386640 ;
x_values(i,2) = -0.5384693101056830 ;
x_values(i,3) = 0.0 ;
x_values(i,4) = -x_values(i,2) ;
x_values(i,5) = -x_values(i,1) ;
c_values(i,1) = 0.2369368850561890 ;
c_values(i,2) = 0.4786386704993660 ;
c_values(i,3) = 0.5688888888888890 ;
c_values(i,4) =c_values(i,2) ;
c_values(i,5) =c_values(i,1) ;

i=6 ;
x_values(i,1) = -.9324695142032520 ;
x_values(i,2) = -.6612093864662650 ;
x_values(i,3) = -.2386191860831970 ;
x_values(i,4) = -x_values(i,3) ;
x_values(i,5) = -x_values(i,2) ;
x_values(i,6) = -x_values(i,1) ;
c_values(i,1) = 0.1713244923791700 ;
```

```
c_values(i,2) = 0.3607615730481390 ;
c_values(i,3) = 0.4679139345726910 ;
c_values(i,4) =c_values(i,3) ;
c_values(i,5) =c_values(i,2) ;
c_values(i,6) =c_values(i,1) ;

i=7 ;
x_values(i,1) = -0.9491079123427590 ;
x_values(i,2) = -0.7415311855993940 ;
x_values(i,3) = -0.4058451513773970 ;
x_values(i,4) = 0.0 ;
x_values(i,5) = -x_values(i,3) ;
x_values(i,6) = -x_values(i,2) ;
x_values(i,7) = -x_values(i,1) ;
c_values(i,1) = 0.1294849661688700 ;
c_values(i,2) = 0.2797053914892770 ;
c_values(i,3) = 0.3818300505051190 ;
c_values(i,4) = 0.4179591836734690 ;
c_values(i,5) =c_values(i,3) ;
c_values(i,6) =c_values(i,2) ;
c_values(i,7) =c_values(i,1) ;

i=8 ;
x_values(i,1) = -0.9602898564975360 ;
x_values(i,2) = -0.7966664774136270 ;
x_values(i,3) = -0.5255324099163290 ;
x_values(i,4) = -0.1834346424956500 ;
x_values(i,5) = -x_values(i,4) ;
x_values(i,6) = -x_values(i,3) ;
x_values(i,7) = -x_values(i,2) ;
x_values(i,8) = -x_values(i,1) ;
c_values(i,1) = 0.1012285362903760 ;
c_values(i,2) = 0.2223810344533740 ;
c_values(i,3) = 0.3137066458778870 ;
c_values(i,4) = 0.3626837833783620 ;
c_values(i,5) =c_values(i,4) ;
c_values(i,6) =c_values(i,3) ;
c_values(i,7) =c_values(i,2) ;
c_values(i,8) =c_values(i,1) ;

i=9 ;
x_values(i,1) = -0.9681602395076260 ;
x_values(i,2) = -0.8360311073266360 ;
x_values(i,4) = -0.6133714327005900 ;
x_values(i,4) = -0.3242534234038090 ;
x_values(i,5) = 0.0 ;
x_values(i,6) = -x_values(i,4) ;
x_values(i,7) = -x_values(i,3) ;
x_values(i,8) = -x_values(i,2) ;
x_values(i,9) = -x_values(i,1) ;
c_values(i,1) = 0.0812743883615740 ;
c_values(i,2) = 0.1806481606948570 ;
```

```
c_values(i,3) = 0.2606106964029350 ;
c_values(i,4) = 0.3123470770400030 ;
c_values(i,5) = 0.3302393550012600 ;
c_values(i,6) =c_values(i,4) ;
c_values(i,7) =c_values(i,3) ;
c_values(i,8) =c_values(i,2) ;
c_values(i,9) =c_values(i,1) ;

i=10 ;
x_values(i,1)  = -0.9739065285171720 ;
x_values(i,2)  = -0.8650633666889850 ;
x_values(i,3)  = -0.6794095682990240 ;
x_values(i,4)  = -0.4333953941292470 ;
x_values(i,5)  = -0.1488743389816310 ;
x_values(i,6)  = -x_values(i,5) ;
x_values(i,7)  = -x_values(i,4) ;
x_values(i,8)  = -x_values(i,3) ;
x_values(i,9)  = -x_values(i,2) ;
x_values(i,10) = -x_values(i,1) ;
c_values(i,1)  = 0.0666713443086880 ;
c_values(i,2)  = 0.1494513491505810 ;
c_values(i,3)  = 0.2190863625159820 ;
c_values(i,4)  = 0.2692667193099960 ;
c_values(i,5)  = 0.2955242247147530 ;
c_values(i,6)  = c_values(i,5) ;
c_values(i,7)  = c_values(i,4) ;
c_values(i,8)  = c_values(i,3) ;
c_values(i,9)  = c_values(i,2) ;
c_values(i,10) = c_values(i,1) ;


disp(sprintf↙
('\n*******************************Simulation*********************************\n'))
disp('1) Lookup Gauss point constants and evaluation points from the table.')
disp(' ')
disp(sprintf('          Evaluation Locations'))

% Used to display the gauss points and their weights
for i=1:n
  disp(sprintf('            x%i=%1.15f',i,x_values(n,i)))
end
disp(' ')
disp(sprintf('          Evaluation Constants'))
for i=1:n
  disp(sprintf('            c%i=%1.15f',i,c_values(n,i)))
end

disp(' ')
disp('2) The evaluation points are definted for the interval [-1,1]. Transform them')
disp('   to fit [a,b]. This is done by the following formula')
disp(' ')
disp('          xnew = (b-a)/2 * x + (b+a)/2')
```

```
disp(' ')
disp(sprintf('          Transformed Evaluation Locations'))
% Transform the locations
for i=1:n
  xv(i)=(b-a)/2*x_values(n,i)+(b+a)/2 ;
  disp(sprintf('            x%i=%1.15f',i,xv(i)))
end

disp(' ')
disp('3) The approximation is given as')
disp(' ')
disp('          approx = sum(i=1,n) c(i)*f(x(i))')

approx = 0 ;
disp(' ')
% approximation via Gaussian integration
for i=1:n-1
  approx = approx + c_values(n,i)*f(xv(i)) ;
  disp(sprintf('          %1.15f * %g',c_values(n,i),f(xv(i))))
end
approx = approx + c_values(n,n)*f(xv(n)) ;
disp(sprintf('        + %1.15f * %g',c_values(n,n),f(xv(n))))
disp(sprintf('         ----------------------------'))
disp(sprintf('           %g',approx))


disp(' ')
disp('4) The approximation must also be transformed to fit the [a,b] interval.')
disp(' ')
disp('         approx = (b-a)/2 * approx')
disp(sprintf('                = %g * %g',(b-a)/2,approx))
% also transform the approximation
approx = (b-a)/2 * approx ;
disp(sprintf('                = %g',approx))

disp(sprintf('\n\n************************Results*****************************'))

% The following finds what is called the 'Exact' solution
exact = quad(f,a,b) ;

disp(sprintf('\n   Approximate = %g',approx))
disp(sprintf('   Exact       = %g',exact))
disp(sprintf('\n   True Error = Exact - Approximate'))
disp(sprintf('              = %g - %g',exact,approx))
disp(sprintf('              = %g',exact-approx))
disp(sprintf('\n   Absolute Relative True Error Percentage'))
disp(sprintf('              = | ( Exact - Approximate ) / Exact | * 100'))
disp(sprintf('              = | %g / %g | * 100',exact-approx,exact))
disp(sprintf('              = %g\n\n',abs( (exact-approx)/exact )*100))
```