```
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=10 ;

% n, the maximum number of segments

  n=2^3 ;

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

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

disp(sprintf↙
('\n*****************************Introduction*********************************'))
disp('Romberg Integration is an extrapolation formula of the Trapezoidal Rule')
disp('for integration. It provides a better approximation of the integral by ')
disp('reducing the True Error')

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 subdivisions, needs to be in powers of 2',n))
format short g

disp(sprintf↙
('\n*****************************Simulation*********************************\n'))
sum=0 ;
disp('The true error of the approximation can be improved by the following formula')
disp(' ')
```

```
disp('     I(k,j) = I(k-1,j+1) + [I(k-1,j+1) - I(k-1,j)] / [4^(k-1) - 1]')
disp(' ')

nstep = floor(log2(n))+1 ;

disp('1) Begin by obtaining the trapezoidal rule approximations')
disp(' ')
% The following in the trapezoidal rule for varying number of segments
for i=1:nstep

  I(1,i)=0.5*(f(a)+f(b)) ;
  h=(b-a)/(2^(i-1)) ;
  for j=1:2^(i-1)-1
    I(1,i)=I(1,i)+f(a+j*h) ;
  end
  I(1,i)=h*I(1,i) ;
  disp(sprintf('   %i Segment Trap, I(1,%i) = %g',2^(i-1),i,I(1,i)))

end

disp(' ')
disp('2) Use the error improvement formula recursively to reduce error.')
disp(' ')

% Iterative Improvements using Romberg Method
index = nstep-1;
for k = [2:nstep]
  for j = [1:index]

    I(k,j)=I(k-1,j+1) + (I(k-1,j+1)-I(k-1,j))/(4^(k-1)-1) ;
    disp(sprintf('    I(%i,%i) = I(%i,%i) + (I(%i,%i) - I(%i,%i))/%g',k,j,k-1,j+1,k-1,j+1,↙
k-1,j,4^(k-1)-1))
    disp(sprintf('           = %g + (%g - %g)/%g',I(k-1,j+1),I(k-1,j+1),I(k-1,j),4^(k-1)↙
-1))
    disp(sprintf('           = %g',I(k,j)))
    disp(' ')

  end
  index = index - 1 ;
end

approx = I(nstep,1) ;

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))
```