Problem 1

```
% A Fredholm integral equation of the second kind is an equation of
 the
%   form:
%       u(x) = f(x) + Integral(a,b, K(x,t)u(t)dt)
%   where a and b and the functions f and K are given. To approximate
 the
%   function u on the interval [a, b], a partition:
%       x0 = a < x1 < ... < x_(m-1) < x_m = b
%   is selected and the equations
%       u(xi) = f(xi) + Integral(a,b, K(xi,t)u(t)dt) i in 0..m
%   are solved for u(x0), u(x1),...,u(xm). The integrals are
 approximated
%   using quadrature formulas based on the nodes x0, x1,..., xm. In
 out
%   problem:
%       a = 0, b = 1, f(x) = x^2, and K(x,t) = exp(|x-t|)
% (a) Set up and solve the linear system that results when the
 Composite
%       Trapezoidal rule is used with n = 4.
% (b) Repeat part (a) using the Composite Simpson's rule.

% After some calculation (which can be reproduced upon asking but is
 too
% long to include in this file) it can be shown that this problem can
 be
% reduced to the set of linear equations given by:
%   (I - kC)u = b
% which is of the form Ax = B where A = (I-kC)
% where
%   I = (n+1)x(n+1) identity matrix
%   k = { h/2 for Comp Trapezoidal rule, h/3 for Comp Simpson's Rule
%   C = coefficient matrix depending on quadrature rule (see code
 below)
%   u = u(xi) for i in 0..n
%   b = f(xi) for i in 0..n

a = 0;                          % initial value of integral
b = 1;                          % final value of integral
f = @(x) x.^2;                  % defined in the lab
K = @(x,t) exp(abs(x-t));       % defined in the lab
n = 4;                          % number of intervals between data points

xVals = linspace(a,b,n+1);      % data points
tVals = xVals;                  % for every k in 0..n, tk = xk

h = ((b-a)/n);                  % step size

fOfxVec = f(xVals)';            % b in above description

% Start building coefficient matrix
```

```
%   The first and last colums will be the same for both Comp
 Trapezoidal
%   Rule and Comp Simpson's Rule because both give a weight of 1 to
 f(a)
%   and f(b) where a = x0 and b = xn.
col1 = K(xVals, tVals(1))';
coln = K(xVals, tVals(end))';

% The entries in KMatrix are exp(|xi - tj|) i in 0..n, j in 1..n-1
[X, T] = meshgrid(xVals, tVals(2:end-1));
KMatrix = (K(X',T'));

eyeNPlus1 = eye(n+1); % Used in both Comp Trapezoidal and Simpson's
 Rule

% Coefficient matrix for Composite Trapezoidal Rule
% Comp Trap Rule:
%   Integral(a,b, f(x)dx) = (h/2)(f(x0) + 2*sum(f(xi),i in 1..n-1) +
 f(xn))
% middleCols represents the 2*sum(xi, i in 1..n-1) entries for each u
middleCols = 2*KMatrix;
compTrapCoeffMat = [col1 middleCols coln];

% A = I - kC
A = eyeNPlus1 - (h/2)*compTrapCoeffMat;
% Solve for unknowns
uTrap = A \ fOfxVec;

% Coefficient matrix for Composite Trapezoidal Rule
% Comp Trap Rule:
%   Integral(a,b, f(x)dx) = (h/3)(f(x0)
%                          | + 2*sum(f(xi), i is even)
%                          | + 4*sum(f(xj), j is odd)
%                          | + f(xn))
% middleCols represents the 2*sum(even) + 4*sum(odd) part where the
 even
% and odd entries have been put in then correct order.

middleCols = zeros(size(KMatrix));      % reinitialize middleCols
[rowNum, colNum] = size(middleCols);    % get size so we have colNum

% apply a weight of 2 to even columns and a weight of 4 to odd columns
for i = 1: colNum
    if mod(i,2) == 0
        middleCols(:,i) = 2*KMatrix(:,i);
    else
        middleCols(:,i) = 4*KMatrix(:,i);
    end
end
compSimpCoeffMat = [col1 middleCols coln];

% A = I - kC
A = eyeNPlus1 - (h/3)*compSimpCoeffMat;
% Solve for unknowns
```

```
uSimp = A \ fOfxVec;

% Print Results
nVec = 0:n;
ansMatrix = [ nVec; xVals; uTrap'; uSimp'];
% Table good for n < 100, dispalying answers to 4 decimal places
fprintf('%2s |\t %-6s |\t %-7s |\t %-7s
\n', 'n', 'x', 'uTrap', 'uSimps');
fprintf('-----------------------------------------\n')
fprintf('%2d |\t %.4f |\t %1.4f |\t %1.4f \n', ansMatrix);
```

```
 n |  x      |  uTrap   |  uSimps
-----------------------------------------
 0 |  0.0000 |  -1.1543 |  -1.2343
 1 |  0.2500 |  -0.9093 |  -0.9507
 2 |  0.5000 |  -0.7153 |  -0.7659
 3 |  0.7500 |  -0.5473 |  -0.5845
 4 |  1.0000 |  -0.3931 |  -0.4485
```

Published with MATLAB® R2018b