Problem 1:

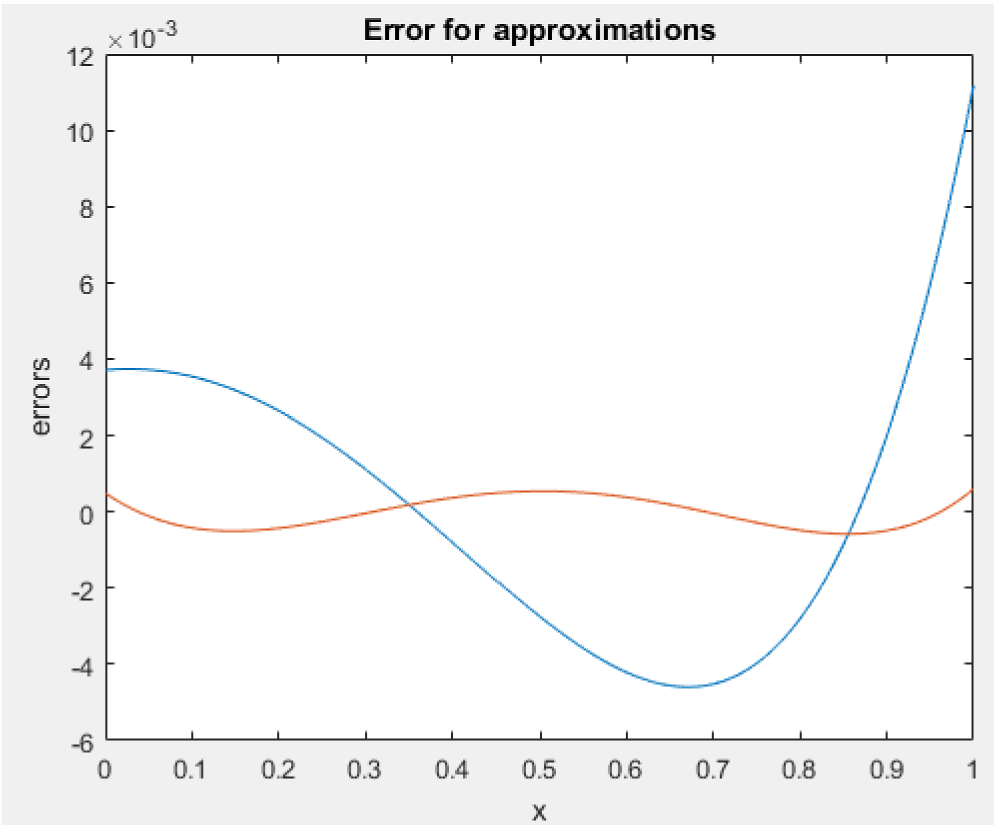
hw7_1.m hw6_4.m

```
n = 3;
xlist = 0:0.01:1;

xs = 0.5*(1-cos(((0:n)+0.5)*pi./(n+1)));  %chebyshev
fs = exp(xs);
fvals = exp(xlist);
dd = divdif(xs,fs);
ls = 0.996294+0.997955*xlist+0.536722*xlist.^2+0.176139*.
ps = dd_interp(xs,dd,xlist);

error1 = fvals - ls; %least squares
error2 = fvals - ps; %chebyshev

plot(xlist,error1)
hold on
plot(xlist,error2);
hold off
title('Error for approximations');
xlabel('x')
ylabel('errors')
```

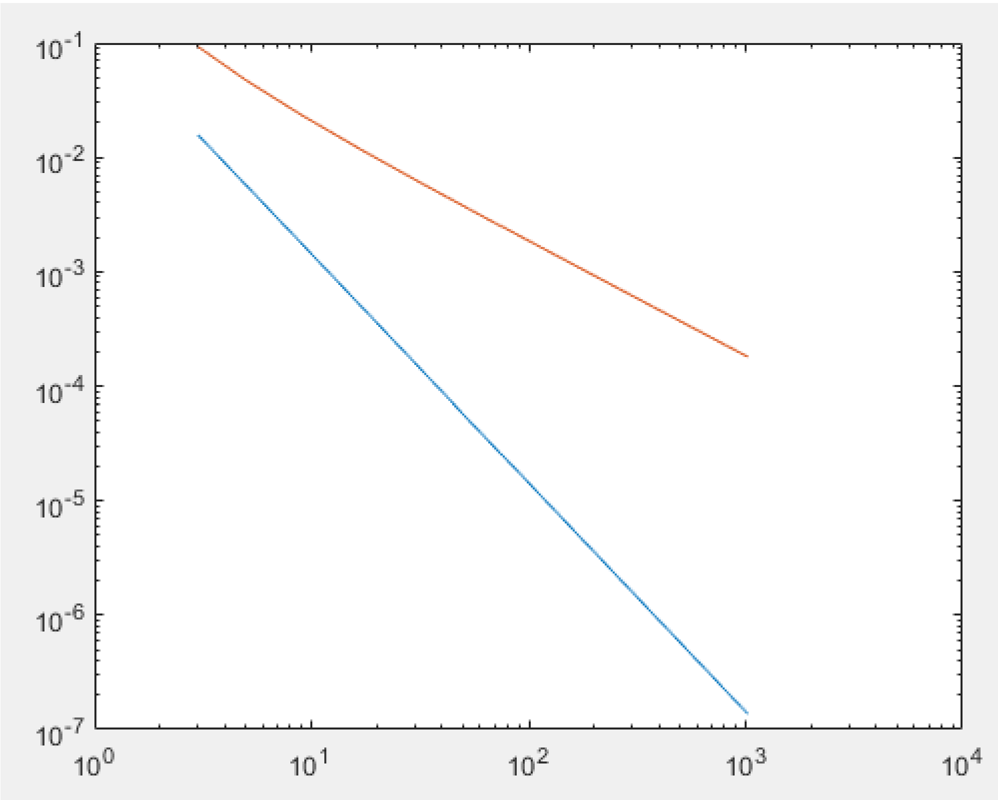


Problem 2:

```
f = @(x)exp(-x)./(1+x)
k = 1:10;
nlist = 2.^k+1;
n = length(nlist);
exact = simpson(f,0,1,10^4/2)

for i = 1:length(nlist)
    trapint = trapezoidal(f,0,1,nlist(i));
    simpint = simpson(f,0,1,nlist(i)/2);
    traperr(i) = trapint - exact;
    simperr(i) = simpint - exact;
end

traperr
simperr
loglog(nlist,abs(traperr),nlist,abs(simperr));
```



The orange line is the error for Simpon’s rule and the blue line is the error for Trapezoidal rule. The plot confirms the theory.

Problem 3:

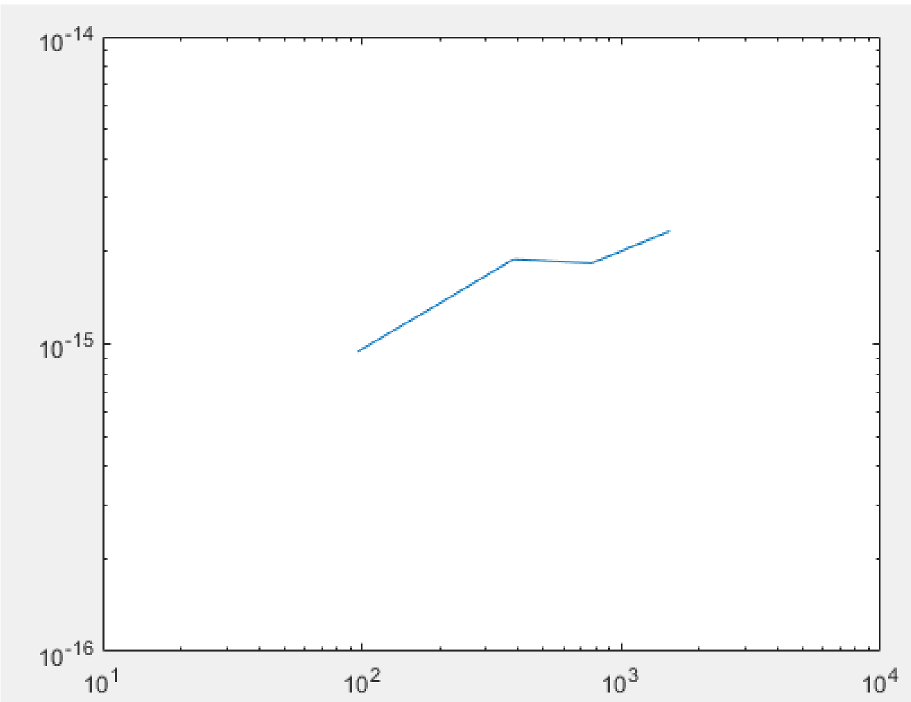
```
function intval = compGauss3(f,a,b,n)
% function intval = compGauss3(f,a,b,n)
% Composite 3-point Gaussian integration rule for computing
% integral of f(x).dx from x = a to x = b with the simple
% 3-point Gauss rule applied to each of n pieces:
% integral f(x).dx from x_i to x_{i+1}.
intval = 0;
h = (b-a)/n; % width of each piece
x = linspace(a,b,n+1);
for i = 0:n-1
  intval = intval + 5/9 * feval(f,x(i+1)+h/2-sqrt(3/5)*h/2);
  intval = intval + 8/9 * feval(f,x(i+1)+h/2);
  intval = intval + 5/9 * feval(f,x(i+1)+h/2+sqrt(3/5)*h/2);
end
intval = h*intval/2;
```

```
f = @(x) exp(-x)./(1+x);
k = 1:9;
nlist = 3 * 2.^k;
n = length(nlist);
exact = simpson(f,0,1,10^4/2)

for i = 1:length(nlist)
    gauint = compGauss3(f,0,1,nlist(i));
    gauerr(i) = gauint - exact;
end

loglog(nlist,gauerr);
```

(Not absolute value of error.)



Problem 4:

```
A = [1 3 -2 1;
     0 -3 2 1;
     2 1 -1 1;
     1 -1 1 1];

[L,U] = mylu(A)
```

```
>> hw7_4

L =

    1.0000         0         0         0
         0    1.0000         0         0
    2.0000    1.6667    1.0000         0
    1.0000    1.3333   -1.0000    1.0000


U =

    1.0000    3.0000   -2.0000    1.0000
         0   -3.0000    2.0000    1.0000
         0         0   -0.3333   -2.6667
         0         0         0   -4.0000
```

```
>> L*U

ans =

     1     3    -2     1
     0    -3     2     1
     2     1    -1     1
     1    -1     1     1
```

(The Matlab outcomes are just for double-checking. Please see the handwritten version of problem 4.)