Computational Methods
Homework 1

Francisco Jose Castillo Carrasco

February 8, 2018

Problem 1

Solution: To obtain the value of the integral for each k I first create a vector where the results will be stored. The individual results are easily calculated by using the function *chebfun* and *integral*.

```
clear all; close all; clc
format long

intf=zeros(21,1);
for k=-10:1:10
    f = chebfun(@(x) exp(-1i*k*x)*exp(sin(x)),[-pi,pi]);
    intf(k+11)=integral(f,-pi,pi);
end

intf

ans =

 -0.000000001729727 - 0.000000000000001i
  0.000000000000000 - 0.000000034673039i
  0.000000625844464 - 0.000000000000001i
  0.000000000000000 + 0.000010048184493i
 -0.000141300427373 + 0.000000000000000i
 -0.000000000000000 - 0.001705653312950i
  0.017197833556866 - 0.000000000000000i
 -0.000000000000000 + 0.139288321767876i
 -0.852927764164121 - 0.000000000000001i
  0.000000000000001 - 3.550999378424362i
  7.954926521012847 + 0.000000000000000i
  0.000000000000000 + 3.550999378424362i
 -0.852927764164121 + 0.000000000000001i
 -0.000000000000000 - 0.139288321767876i
  0.017197833556866 - 0.000000000000000i
 -0.000000000000000 + 0.001705653312950i
 -0.000141300427373 - 0.000000000000000i
  0.000000000000000 - 0.000010048184493i
  0.000000625844464 + 0.000000000000001i
  0.000000000000000 + 0.000000034673039i
 -0.000000001729727 + 0.000000000000001i
```

Problem 2

Solution: We start by expanding $f(x+h)$ and $f(x-h)$ up to eigth order,

$$\begin{aligned} f(x \pm h) =& f(x) \pm h f'(x) + \frac{h^2}{2} f''(x) \pm \frac{h^3}{3!} f'''(x) + \frac{h^4}{4!} f^{(IV)}(x) \pm \frac{h^5}{5!} f^{(V))}(x) + \frac{h^6}{6!} f^{(VI)}(x) \\ & \pm \frac{h^7}{7!} f^{(VII))}(x) + \frac{h^8}{8!} f^{(VIII)}(x) + \mathcal{O}(h^9). \end{aligned}$$

By adding both $f(x+h)$ and $f(x-h)$ together, crossing out many terms,

$$f(x+h) + f(x-h) = 2f(x) + h^2 f''(x) + \frac{2h^4}{4!} f^{(IV)} + \frac{2h^6}{6!} f^{(VI)} + \frac{2h^8}{8!} f^{(VIII)} + \mathcal{O}(h^9),$$

we can now isolate the second derivative

$$f''(x) = \frac{f(x-h) - 2f(x) + f(x+h)}{h^2} - \frac{h^2}{12} f^{(IV)} - \frac{2h^4}{6!} f^{(VI)} - \frac{2h^6}{8!} f^{(VIII)} + \mathcal{O}(h^9).$$

If we want sixth order precision we need to get rid of the firs two leading terms of the error. In order to do so we use more stencil points. We can express the second derivative as

$$f''(x) = \frac{f(x-2h) - 2f(x) + f(x+2h)}{4h^2} - \frac{h^2}{3} f^{(IV)} - \frac{2h^4}{45} f^{(VI)} - \frac{h^6}{315} f^{(VIII)} + \mathcal{O}(h^9),$$

and

$$f''(x) = \frac{f(x-3h) - 2f(x) + f(x+3h)}{9h^2} - \frac{3h^2}{4} f^{(IV)} - \frac{9h^4}{40} f^{(VI)} - \frac{81h^6}{2240} f^{(VIII)} + \mathcal{O}(h^9).$$

Multiplying the three equations by a, b, c, respectively, and adding them together we can impose the condition that the coefficients of the second and fourth order error terms must be zero, obtaining a sixth order approximation. The system of equations to solve is then

$$\begin{aligned} \frac{a}{12} + \frac{b}{3} + \frac{3c}{4} &= 0, \\ \frac{0}{360} + \frac{2b}{45} + \frac{9c}{40} &= 0. \end{aligned}$$

The solutions are $a = 15c$, $b = -6c$, and c will vanish from the system. Once we put this values in the equations and add them together we get the final result

$$\begin{aligned} f''(x) =& \frac{15\left[f(x-h) + f(x+h)\right] - \frac{3}{2}\left[f(x-2h) + f(x+2h)\right] + \frac{1}{9}\left[f(x-3h) + f(x+3h)\right] - \frac{245}{9} f(x)}{10h^2} \\ & - \frac{h^6}{560} f^{(VIII)}(x) + \mathcal{O}(h^7), \end{aligned}$$

a sixth order approximation to the second derivative. In the figure 1 we can see the funtion and its derivatives

$$f(x) = e^x \sin(3x), \qquad f'(x) = e^x\left(\sin(3x) + 3\cos(3x)\right), \qquad f''(x) = e^x\left(6\cos(3x) - 8\sin(3x)\right),$$

as well as the value of the second derivative at the point $x_0 = 0.4$.

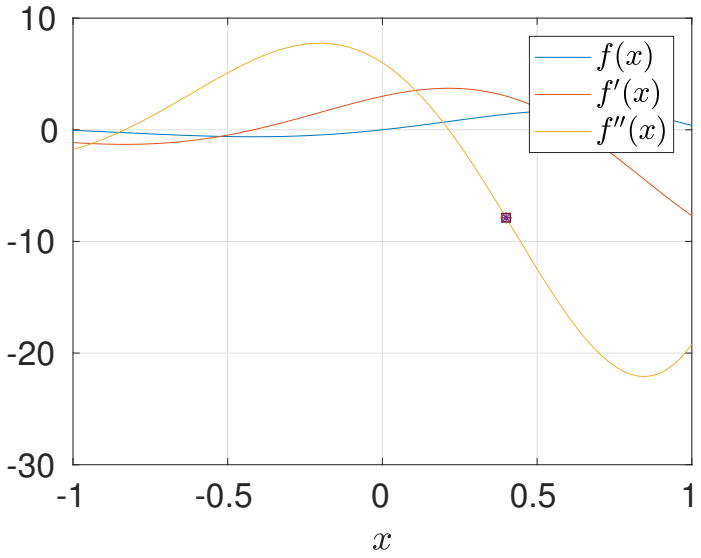


Figure 1: The function $f(x)$ and its two first derivatives. It is also included the value of the second derivative at $x_0 = 0.4$ (asterisk) and the approximated value (square) using the approximation derived above.

We can see that the approximated value for the three values of h is very accurate as we cannot distinguish any square from the astersik. However, in the following figure we can see the error and, in logarithmic scale, seems to have a linear dependency on h.

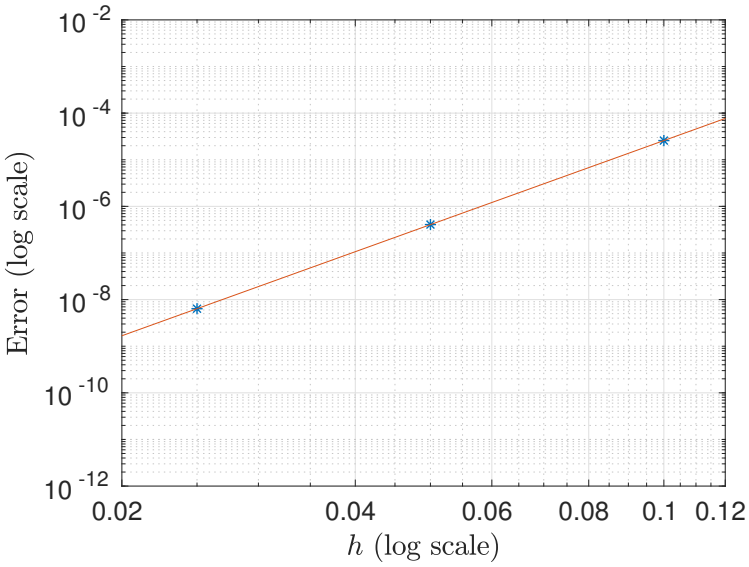


Figure 2: Error of the approximation.

We made a linear fitting and found out that the slope was $m = 5.990378959014029$. This proves that the approximation is, indeed, of sixth order.

Matlab code for this problem

```
x=-1:0.005:1;
x_0=0.4;
f = chebfun(@(x) exp(x)*sin(3*x));
df = chebfun(@(x) exp(x)*(sin(3*x)+3*cos(3*x)));
ddf = chebfun(@(x) exp(x)*(6*cos(3*x)-8*sin(3*x)));

H=[0.1;0.05;0.025];
ddf_approx=zeros(3,1);
for k=1:3
    h=H(k);
    ddf_approx(k)=(15*(f(x_0-h)+f(x_0+h))-1.5*(f(x_0-2*h)+f(x_0+2*h))
    +(1/9)*(f(x_0-3*h)+f(x_0+3*h))-(245/9)*f(x_0))/(10*h^2);
end

figure(1)
plot(x,f(x),x,df(x),x,ddf(x),x_0,ddf(x_0),'*',x_0,ddf_approx(1),'s',
x_0,ddf_approx(2),'s',x_0,ddf_approx(3),'s')
grid on
legend({'$f(x)$','$f''(x)$','$f''''(x)$'},'Interpreter','latex')
xlabel('$x$','Interpreter','latex')
set(gca,'fontsize',18)
saveas(gcf,'IMAGES/problem2_1','epsc')

P=polyfit(log(H),log(abs(ddf(x_0)-ddf_approx)),1);
h=0.01:1e-3:0.15;
figure(2)
loglog(H,abs(ddf(x_0)-ddf_approx),'*')
grid on
xlabel('$h$ (log scale)','fontsize',20,'interpreter','latex')
ylabel('Error (log scale)','fontsize',20,'interpreter','latex')
set(gca,'fontsize',14)
hold on
loglog(h,exp(P(1)*log(h)+P(2)))
grid on
axis([2e-2 1.2e-1 1e-12 1e-2])
saveas(gcf,'IMAGES/problem2_2','epsc')
P =

   5.990378959014029   3.227778361027343
```

Problem 3

Solution: Part a) The error

$$\begin{aligned}
\left|\frac{f(x+h)-f(x)}{h} - \frac{fl(f(x+h))-fl(f(x))}{h}\right| &= \left|\frac{f(x+h)-f(x)}{h} - \frac{f(x+h)(1+\delta_1)-f(x)(1+\delta_2)}{h}\right| \\
&= \left|-\frac{f(x+h)\delta_1 - f(x)\delta_2}{h}\right| \\
&\leq \left|\frac{f(x+h)\delta_1}{h}\right| + \left|\frac{f(x)\delta_2}{h}\right| \\
&< |f(x+h)|\,\frac{\epsilon}{h} + |f(x)|\,\frac{\epsilon}{h} \\
&= \left(\left|f(x)+hf'(x)+\mathcal{O}(h^2)\right| + |f(x)|\right)\frac{\epsilon}{h} \\
&\leq \left(2\,|f(x)| + |hf'(x)| + \left|\mathcal{O}(h^2)\right|\right)\frac{\epsilon}{h} \sim \mathcal{O}\left(\frac{\epsilon}{h}\right),
\end{aligned}$$

is proportional to ϵ/h in the worst case as $h \to 0$.

Part b) We just proved that the rounding error is proportional to ϵ/h, then

$$e_\delta = k_1\frac{\epsilon}{h}.$$

On the other hand, the truncation error is

$$e_T = \left|f'(x) - \frac{f(x+h)-f(x)}{h}\right| = k_2 h,$$

since it is a first order approximation. To minimize the total error

$$e = e_\delta + e_T = k_1\frac{\epsilon}{h} + k_2 h,$$

we just have to solve the optimization problem,

$$\frac{\partial e}{\partial h} = -k_1\frac{\epsilon}{h^2} + k_2 = 0,$$

which gives us $h^* = \sqrt{\frac{k_1}{k_2}\epsilon}$. We check that it is indeed a minimum by checking that

$$\left.\frac{\partial^2 e}{\partial h^2}\right|_{h=h^*} = \frac{2k_1\epsilon}{(h^*)^3} = \frac{2k_1\epsilon}{\left(\frac{k_1}{k_2}\epsilon\right)^{3/2}} > 0.$$

Part c) In equispaced finite difference methods of order p the rounding error is

$$\begin{aligned}
e_\delta &= \left|\sum_{j=0}^{p}\frac{w_j}{h}f_j - \sum_{j=0}^{p}\frac{w_j}{h}f_j(1+\delta_j)\right| \\
&= \left|-\sum_{j=0}^{p}\frac{w_j}{h}f_j\delta_j\right| \\
&< \left|\sum_{j=0}^{p} w_j f_j\right|\frac{\epsilon}{h} = k_1\frac{\epsilon}{h},
\end{aligned}$$

and the truncation error of order p is

$$e_T = k_2 h^p.$$

To minimize the total error

$$e = e_\delta + e_T = k_1 \frac{\epsilon}{h} + k_2 h^p,$$

we just have to solve the optimization problem,

$$\frac{\partial e}{\partial h} = -k_1 \frac{\epsilon}{h^2} + p k_2 h^{p-1} = 0,$$

which gives us $h^* = \left(p \frac{k_1}{k_2} \epsilon\right)^{1/p+1}$. We check that it is indeed a minimum by checking that

$$\left.\frac{\partial^2 e}{\partial h^2}\right|_{h=h^*} = \frac{2k_1\epsilon}{(h^*)^3} + p(p-1)k_2(h^*)^{p-2} = \frac{2k_1\epsilon}{\left(p\frac{k_1}{k_2}\epsilon\right)^{3/p+1}} + p(p-1)k_2 \left(p\frac{k_1}{k_2}\epsilon\right)^{\frac{p-2}{p+1}} > 0.$$

Part d) We are given the function $\tan(x)$ and asked to approximate its derivative at $x_0 = \pi/3$. We can calculate it analytically,

$$f'(x) = \frac{1}{\cos^2 x}, \qquad f'(\pi/3) = 4.$$

With the same procedure as in the previous problem, we have derived the following approximations and the leading error terms:

- First order:

$$f'(x) = \frac{f(x+h) - f(x)}{h} - \frac{h}{2} f''(x) + \mathcal{O}(h^2)$$

- Second order:

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} - \frac{h^2}{6} f'''(x) + \mathcal{O}(h^3)$$

- Fourth order:

$$f'(x) = \frac{2}{3}\frac{f(x+h) - f(x-h)}{h} - \frac{1}{12}\frac{f(x+2h) - f(x-2h)}{h} + \frac{4h^4}{5!} f^{(V)}(x) + \mathcal{O}(h^5)$$

- Sixth order:

$$f'(x) = 15\frac{f(x+h) - f(x-h)}{20h} - 6\frac{f(x+2h) - f(x-2h)}{40h} + \frac{f(x+3h) - f(x-3h)}{60h} - \frac{h^6}{140} f^{(VII)}(x) + \mathcal{O}(h^7)$$

In the next figure we can see, in logarithmic scale, the dependency of the truncation error on h for every approximation. The lines are have slopes $1, 2, 4, 6$ (from top to bottom) mathching its respective data. This proves the order of the truncation error for the four approximations considered. Note that the minimal error is achieved "sooner" (in the sense of not needing to reduce h as much) for higher order approximations. In addition, we can check that once the rounding error is dominant, the dependency is the same for all the approximations. It cannot be any other way if we remember that $e_\delta \sim \frac{\epsilon}{h}$ regardless of the order of our approximation. We can check that this error would in fact have a slope of minus 1 in logarithmic scale, as we see in the figure.

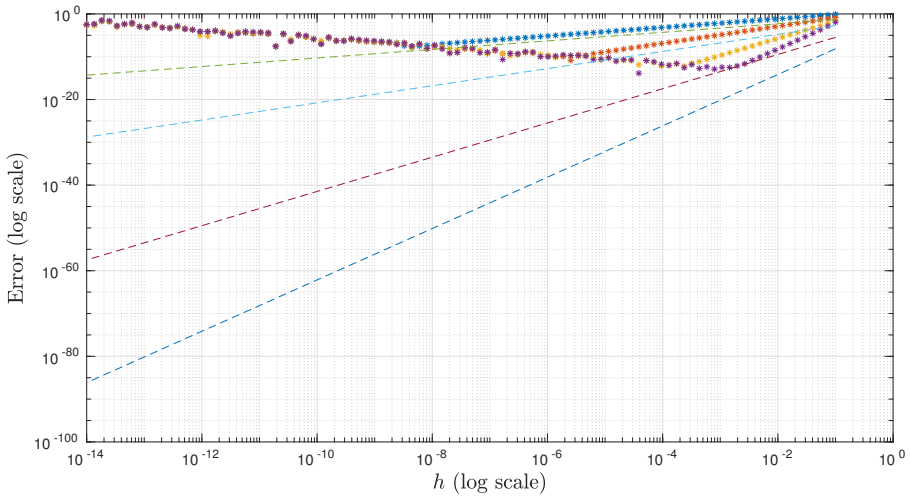


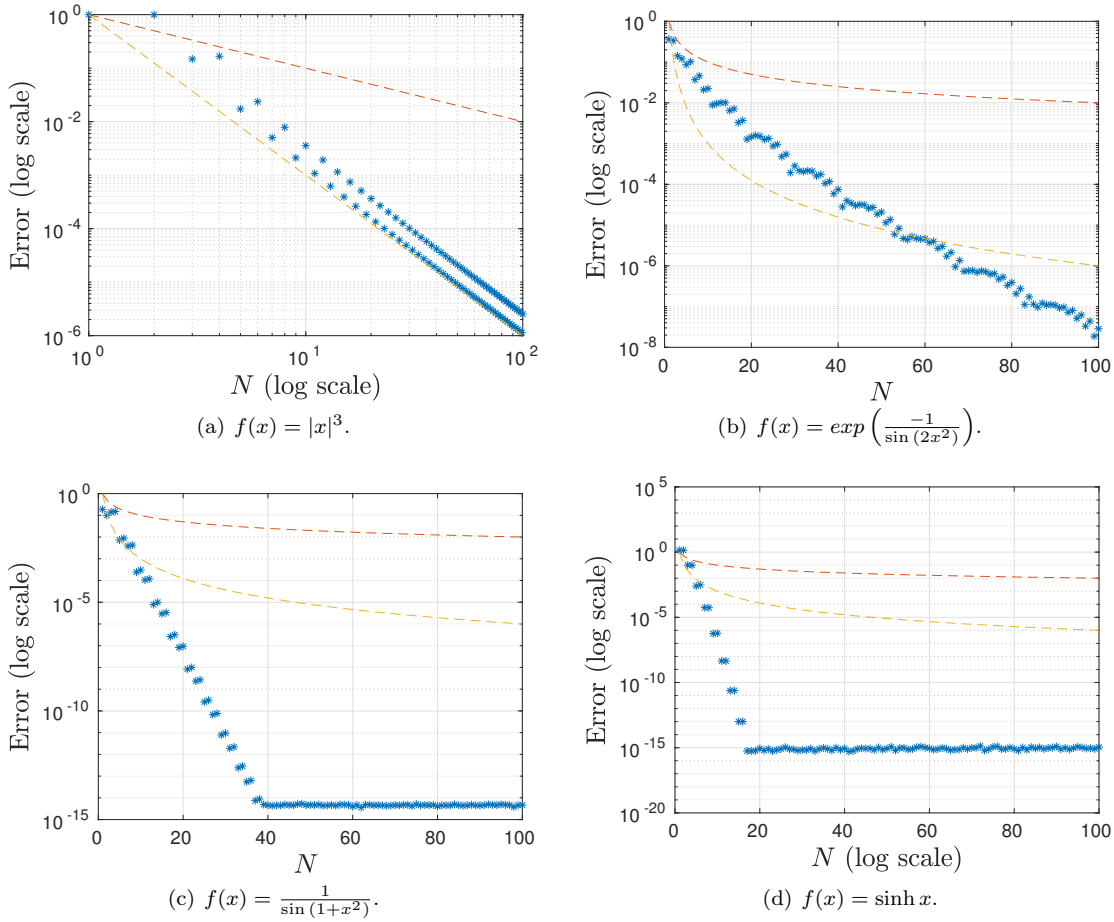
Figure 3: Error dependency on h for the four different approximations.

Matlab code for this problem

```
format long
x_0=pi/3;
f = @(x) tan(x);
fp = @(x) (1/cos(x))^2;
H=logspace(-1,-14,100);
for k = 1:length(H)
    h = H(k);
    df1(k) = (f(x_0+h)-f(x_0))/h;
    df2(k) = (f(x_0+h)-f(x_0-h))/(2*h);
    df4(k) = (-5/60*f(x_0+2*h)+2/3*f(x_0+h)-2/3*f(x_0-h)+5/60*f(x_0-2*h))/h;
    df6(k) = 0.1*(15*(f(x_0+h)-f(x_0-h))/(2*h)-6*(f(x_0+2*h)-f(x_0-2*h))/(4*h)
       +(f(x_0+3*h)-f(x_0-3*h))/(6*h));
end
figure('units','normalized','outerposition',[0 0 1 1])
loglog(H,abs(df1-fp(x_0)),'*',H,abs(df2-fp(x_0)),'*',H,abs(df4-fp(x_0)),'*',
   H,abs(df6-fp(x_0)),'*',H,H/2,'--',H,H.^(2)/6,'--',H,H.^(4)/30,'--',H,H.^6/140,'--')
set(gca,'fontsize',14)
grid on
xlabel('$h$ (log scale)','fontsize',20,'interpreter','latex')
ylabel('Error (log scale)','fontsize',20,'interpreter','latex')
saveas(gcf,'IMAGES/problem3','epsc')
```

Problem 4

Solution: In this problem we interpolate the different functions using polynomial interpolation on Chebishev nodes. The rate of convergence is given in the next figure. Observe that the rate of convergence is the highest for the function of part d,since it is infinitely smooth, followed by the function of part c,a and b. The order of this convergence is because of the smoothness of the different functions. In real numbers the function of part c is infinitely smooth as well, whereas the other two functions arent.



(a) $f(x) = |x|^3$.

(b) $f(x) = exp\left(\frac{-1}{\sin(2x^2)}\right)$.

(c) $f(x) = \frac{1}{\sin(1+x^2)}$.

(d) $f(x) = \sinh x$.

Figure 4: Error of the polynomial interpolation on Chebyshev nodes for the different functions.

Matlab code for this problem

```
% Part a
N = 1:100;
err = 0*N;
ff = @(x) abs(x).^3;
f = chebfun(ff,'splitting','on');
for k = 1:length(N)
    n = N(k);
    g = chebfun(ff,n);
    err(k) = norm(f-g,inf);
```

```
end
figure
loglog(N,err,'*',N,N.^-1,'--',N,N.^-3,'--')
set(gca,'fontsize',14)
grid on
xlabel('$N$ (log scale)','fontsize',20,'interpreter','latex')
ylabel('Error (log scale)','fontsize',20,'interpreter','latex')
saveas(gcf,'IMAGES/problem4a','epsc')
% Part b
ff = @(x) exp(-1/sin(2*x^2));
f = chebfun(ff,'splitting','on');
for k = 1:length(N)
    n = N(k);
    g = chebfun(ff,n);
    err(k) = norm(f-g,inf);
end
figure
semilogy(N,err,'*',N,N.^-1,'--',N,N.^-3,'--')
set(gca,'fontsize',14)
grid on
xlabel('$N$','fontsize',20,'interpreter','latex')
ylabel('Error (log scale)','fontsize',20,'interpreter','latex')
saveas(gcf,'IMAGES/problem4b','epsc')

% Part c
ff = @(x) 1/sin(1+x^2);
f = chebfun(ff,'splitting','on');
for k = 1:length(N)
    n = N(k);
    g = chebfun(ff,n);
    err(k) = norm(f-g,inf);
end
figure
semilogy(N,err,'*',N,N.^-1,'--',N,N.^-3,'--')
set(gca,'fontsize',14)
grid on
xlabel('$N$ ','fontsize',20,'interpreter','latex')
ylabel('Error (log scale)','fontsize',20,'interpreter','latex')
saveas(gcf,'IMAGES/problem4c','epsc')

% Part d
ff = @(x) sinh(x)^2;
f = chebfun(ff,'splitting','on');
for k = 1:length(N)
    n = N(k);
    g = chebfun(ff,n);
    err(k) = norm(f-g,inf);
end
figure
semilogy(N,err,'*',N,N.^-1,'--',N,N.^-3,'--')
set(gca,'fontsize',14)
```

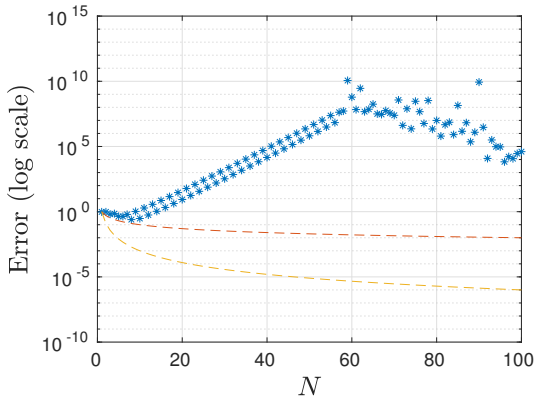
```
grid on
xlabel('$N$ (log scale)','fontsize',20,'interpreter','latex')
ylabel('Error (log scale)','fontsize',20,'interpreter','latex')
saveas(gcf,'IMAGES/problem4d','epsc')
```

Problem 5

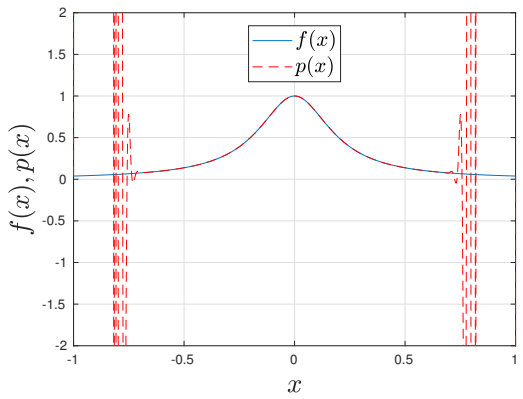
Solution: In this problem we are going to see how inadequate it is to use equispaced nodes instead of, for instance, Chebishev nodes. In the interval $[-1, 1]$ we are going to use polynomial approximation with equispaced nodes to apporximate the function

$$f(x) = \frac{1}{1 + 25x^2},$$

and compute the L_∞ error. In the following figure we ca see how the error grows with the number of nodes used. This clearly telling us that the nodes are not well distributed since using more nodes should give us more precision (at least for not too exotic problems). To understand this, as we saw in class, let's look at the function and the polynomial of degree 100 (highest degree used) together in the figure. It is obvious that the polynomial does not approximate the function at all when we get close to the boundries. This is causing the increasing error with N.



(a) Error dependency on N.



(b) Function and polynomial comparison.

Figure 5: Polynomial interpolation using equispaced nodes for the function $f(x) = \frac{1}{1+25x^2}$.

Matlab code for this problem

```
ff = @(x) 1./(1+25*x.^2);
N = 1:100;
err = 0*N;
xx = linspace(-1,1,1000)';
for k = 1:length(N)
    x = linspace(-1,1,N(k))';
    W = baryWeights(x);
    y = ff(x);
    yy = bary(xx,y,x,W);
```

```
    err(k) = norm(ff(xx)-yy,inf);
end
figure
semilogy(N,err,'*',N,N.^-1,'--',N,N.^-3,'--')
set(gca,'fontsize',14)
grid on
xlabel('$N$','fontsize',20,'interpreter','latex')
ylabel('Error (log scale)','fontsize',20,'interpreter','latex')
saveas(gcf,'IMAGES/problem5_1','epsc')

figure
plot(xx,ff(xx))
hold on
plot(xx,yy,'r--')
xlabel('$x$','fontsize',20,'interpreter','latex')
ylabel('$f(x),p(x)$','fontsize',20,'interpreter','latex')
legend({'$f(x)$','$p(x)$'},'Interpreter','latex','Location','north','fontsize',16)
axis([-1 1 -2 2])
grid on
saveas(gcf,'IMAGES/problem5_2','epsc')
```