Spectral Methods

Francisco Castillo
Homework 3

April 3, 2019

Problem 1

Based on the theorems of chapter 8 of Approximation Theory and Approximation Practice by Trefethen, what can you say about the convergence as $n \to \infty$ of the Chebyshev interpolants to the following functions? Which is the case that converges much faster than the theorems predict? Can you speculate why?

The convergence, predicted by THEOREM 8.2, is

$$\|f - p_n\| \leq \frac{4M\rho^{-n}}{\rho - 1},$$

with

$$\rho = \alpha + \sqrt{\alpha^2 - 1} \quad \text{(real singularity at } x = \pm\alpha\text{)},$$

or

$$\rho = \beta + \sqrt{\beta^2 + 1} \quad \text{(imaginary singularity at } x = \pm i\beta\text{)}.$$

Applying this to the following functions we obtain:

a) $f(x) = \tan(x)$. This function has a real singularity at $x = \pm\pi/2$. Hence,

$$\rho = \frac{\pi}{2} + \sqrt{\frac{\pi^2}{4} - 1},$$

and we have obtained the rate of convergence showed in the next figure.

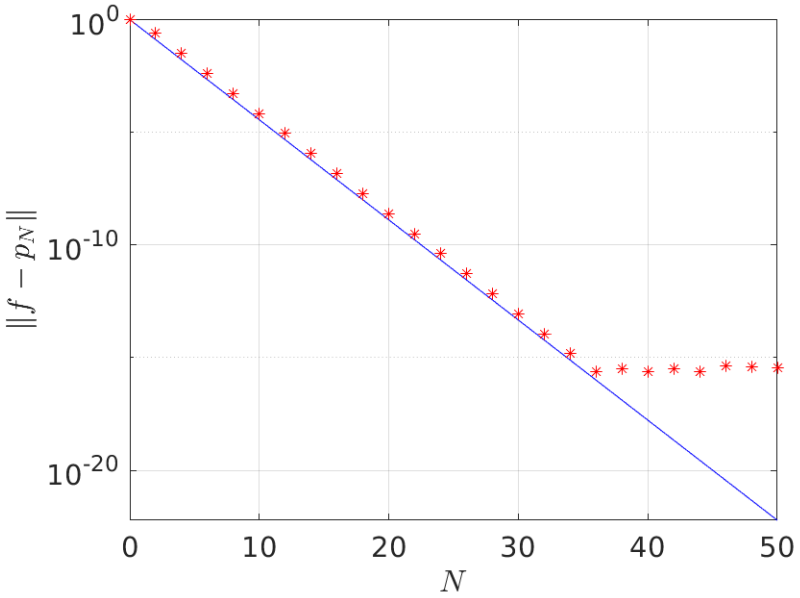

Figure 1: Convergence as $n \to \infty$ of the Chebyshev interpolant to $f(x) = \tan(x)$.

b) $f(x) = \tanh(x)$. This function has a imaginary singularity at $x = \pm i\pi/2$. Hence,

$$\rho = \frac{\pi}{2} + \sqrt{\frac{\pi^2}{4} + 1},$$

and we have obtained the rate of convergence showed in the next figure.

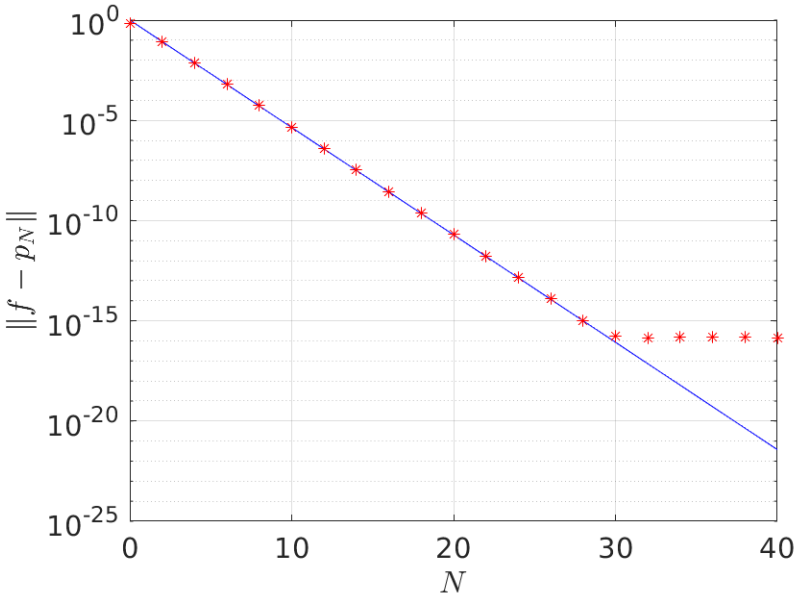

Figure 2: Convergence as $n \to \infty$ of the Chebyshev interpolant to $f(x) = \tanh(x)$.

c) $f(x) = \frac{\log(\frac{x+3}{4})}{x-1}$. This function has a real singularity at $x = -3$. Hence,

$$\rho = 3 + \sqrt{8},$$

and we have obtained the rate of convergence showed in the next figure.

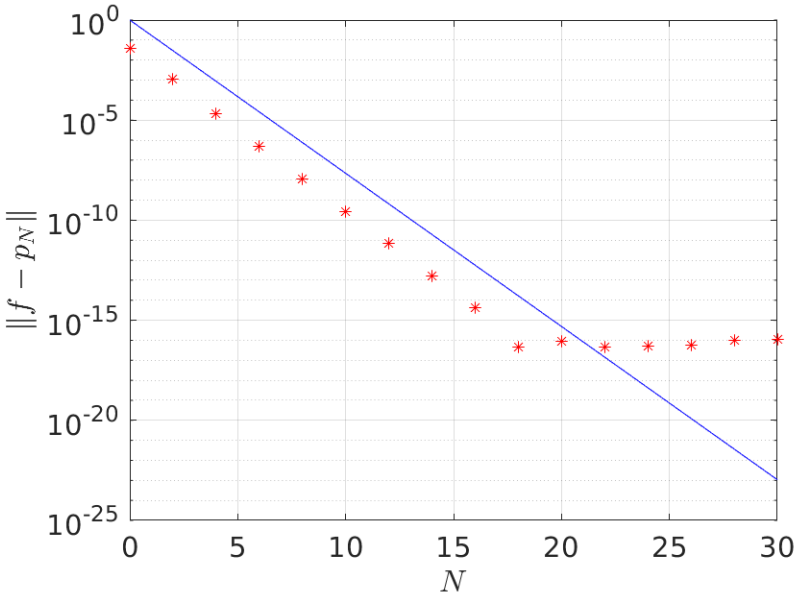

Figure 3: Convergence as $n \to \infty$ of the Chebyshev interpolant to $f(x) = \frac{\log(\frac{x+3}{4})}{x-1}$.

d) $f(x) = \int_{-1}^{x} \cos(t^2)dt$. This function is entire, analytic at all finite points of the complex plane. Hence its convergence is much faster as is shown in the following figure.

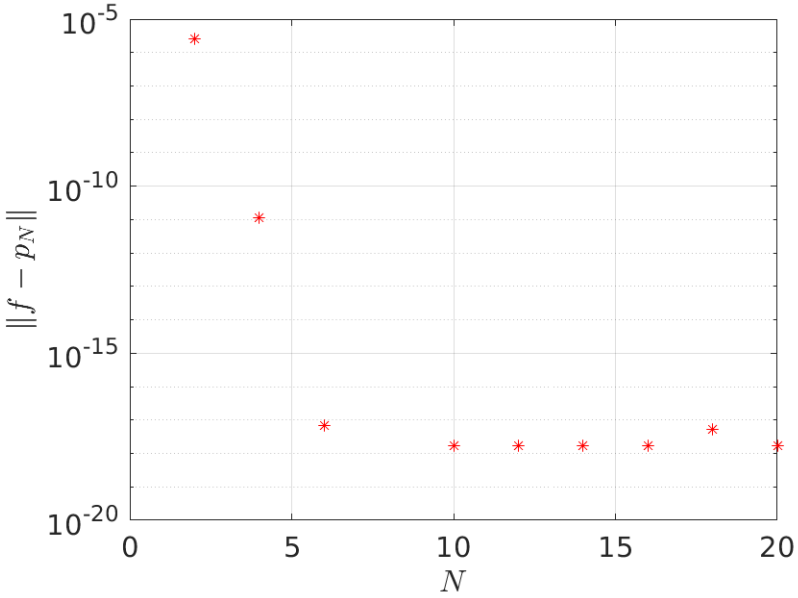

Figure 4: Convergence as $n \to \infty$ of the Chebyshev interpolant to $f(x) = \frac{\log(\frac{x+3}{4})}{x-1}$.

e) $f(x) = \tan(\tan(x))$. This function has a real singularity at $x = \pm \arctan(\pi/2) = \pm k$. Hence,

$$\rho = k + \sqrt{k^2 - 1},$$

and we have obtained the rate of convergence showed in the next figure.

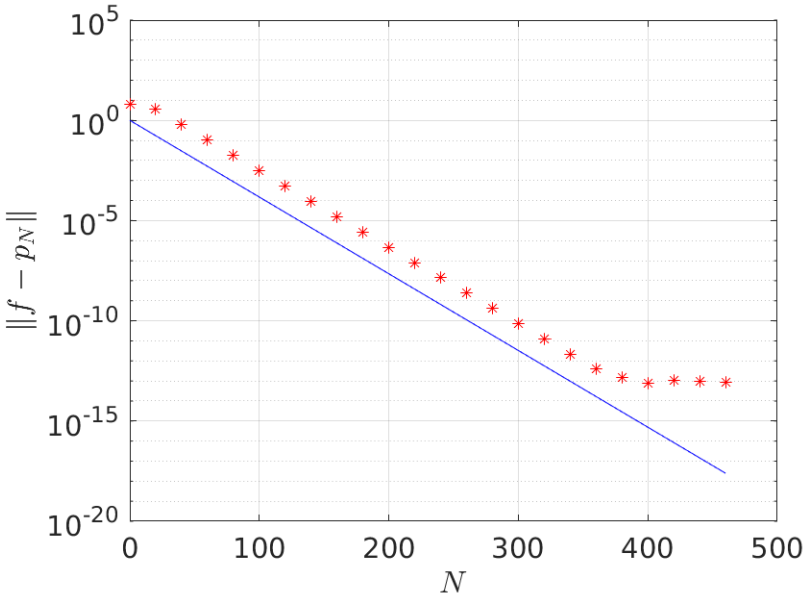

Figure 5: Convergence as $n \to \infty$ of the Chebyshev interpolant to $f(x) = \tan(\tan(x))$.

f) $f(x) = (1+x)\log(1+x)$. This function has a real singularity at $x = -1$. Hence,

$$\rho = -1,$$

and we have obtained the rate of convergence showed in the next figure.

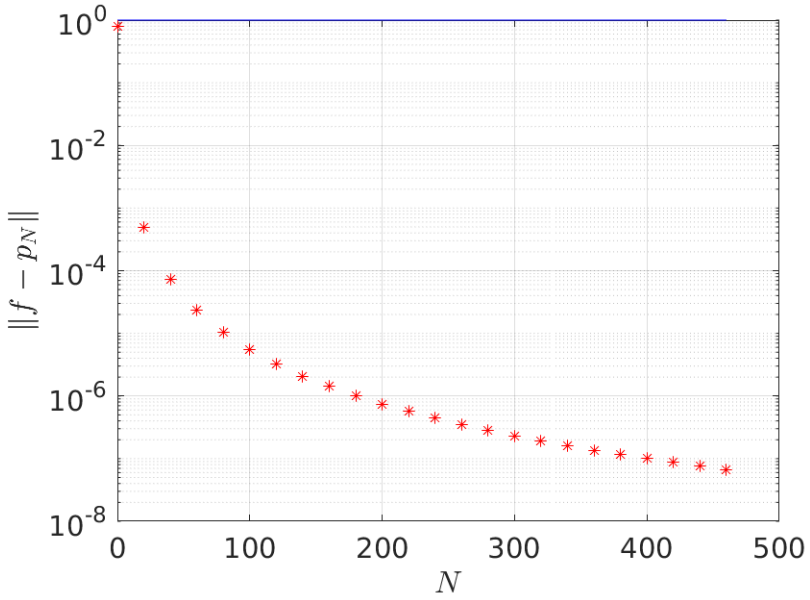

Figure 6: Convergence as $n \to \infty$ of the Chebyshev interpolant to $f(x) = (1+x)\log(1+x)$.

This figure is not right since the function does not satisfies the assumptions of THEOREM 8.1, it is not analytic in $[-1, 1]$. The rate of convergence found is algebraic (much worse than the previous cases), as shown in the following figure.

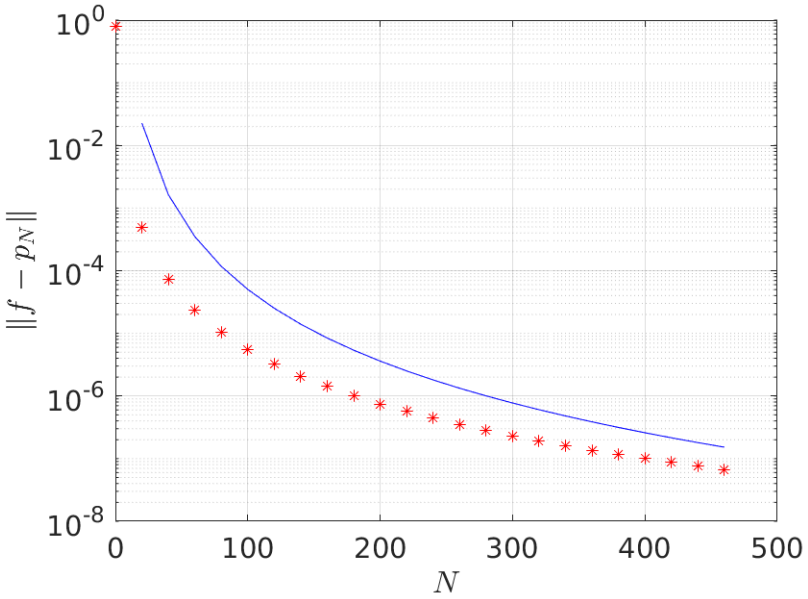

Figure 7: Algebraic convergence as $n \to \infty$ of the Chebyshev interpolant to $f(x) = (1+x)\log(1+x)$.

Overall, it has been tested that the larger the ellipse with foci $\{-1, 1\}$ within which the function is analytic, the faster the convergence.

Matlab code for this problem

```
rho = 0.5*pi+sqrt(0.25*pi^2-1);
orderAcuracy('tan(x)',50,2,rho)

rho = (0.5*pi+sqrt(0.25*pi^2+1));
orderAcuracy('tanh(x)',40,2,rho)

rho = (3+sqrt(8));
orderAcuracy('log((x+3)/4)/(x-1)',30,2,rho,'log1')

k = atan(pi/2);
rho = k+sqrt(k^2-1);
orderAcuracy('tan(tan(x))',460,20,rho)

rho = -1;
orderAcuracy('(1+x)*log(1+x)',460,20,rho,'(1+x)log(1+x)wrong')
orderAcuracy('(1+x)*log(1+x)',460,20,rho,'(1+x)log(1+x)right')

function orderAcuracy(func,Nmax,Nstep,rho,namefig)
    labelfontsize = 14;
```

```
    figformat = 'png';
    if nargin < 5
        namefig = func;
    end
    f = chebfun(func);
    nn = 0:Nstep:Nmax; ee = 0*nn;
    for j=1:length(nn)
        n = nn(j);
        fn = chebfun(f,n+1);
        ee(j) = norm(f-fn);
    end
    figure
    if strcmp(namefig,'(1+x)log(1+x)right')
        semilogy(nn,2000*nn.^(-3.8),'-b')
    else
        semilogy(nn,rho.^(-nn),'-b')
    end
    hold on
    semilogy(nn,ee,'r*')
    grid on
    xlabel('$N$','interpreter','latex')
    ylabel('$\|f-p_N\|$','interpreter','latex')
    set(gca,'fontsize',labelfontsize)
    txt=['Latex/FIGURES\' namefig];
    saveas(gcf,txt,figformat)
end

% For the integral function
nn = 2:2:20;
err = 0*nn;
for k = 1:length(nn)
    N = nn(k);
    x = -.98:0.02:1;
    F = 0*x;
    FN = F;
    for j = 1:length(x)
        f = chebfun('cos(t^2)', [-1 x(j)]);
        fN = chebfun('cos(t^2)', [-1 x(j)],N);
        F(j) = sum(f);
        FN(j) = sum(fN);
        F = [0 F]; FN = [0 FN]; x = [-1 x];
    end
    err(k) = norm(F-FN);
end
figure
```

```
semilogy(nn,err,'r*')
grid on
xlabel('$N$','interpreter','latex')
ylabel('$\|f-p_N\|$','interpreter','latex')
set(gca,'fontsize',labelfontsize)
txt='Latex/FIGURES\integral';
saveas(gcf,txt,figformat)
```

Problem 2

The function $|x-i|$ is analytic for $x \in [-1,1]$. This means it can be analytically continued to an analytic function $f(z)$ in a neighborhood of $[-1,1]$ in the complex z-plane. The formula $|z-i|$ itself does not define an analytic function in any complex neighborhood. Find another formula for f that does, and use it to explain what singularities f has in the complex plane.

For a complex function $f(z) = u+vi$ to be analytic, it has to satisfy the Cauchy-Riemann Equations:

$$\begin{aligned}\frac{\partial u}{\partial x} &= \frac{\partial v}{\partial y},\\ \frac{\partial u}{\partial y} &= -\frac{\partial v}{\partial x}.\end{aligned}$$

In this case we have that

$$\begin{aligned}f(z) = |z-i| &= |x+yi-i|\\ &= |x+i(y-1)|\\ &= \sqrt{x^2+(y-1)^2},\end{aligned}$$

where we have used that $z = x+yi$. Hence we observe that this function is not analytic anywhere since $v = 0$ and u_x,u_y are not. Therefore we can define

$$f(z) = f(x) = |x-i| = \sqrt{x^2+1},$$

which is analytic within a neighborhood of $x \in [-1,1]$ in the complex plane and has singularities at $z = \pm i$. Hence, like in the previous problem, its Bernstein ellipse is defined by

$$\rho = 1+\sqrt{2},$$

and the exponential convergence is shown in the next figure.

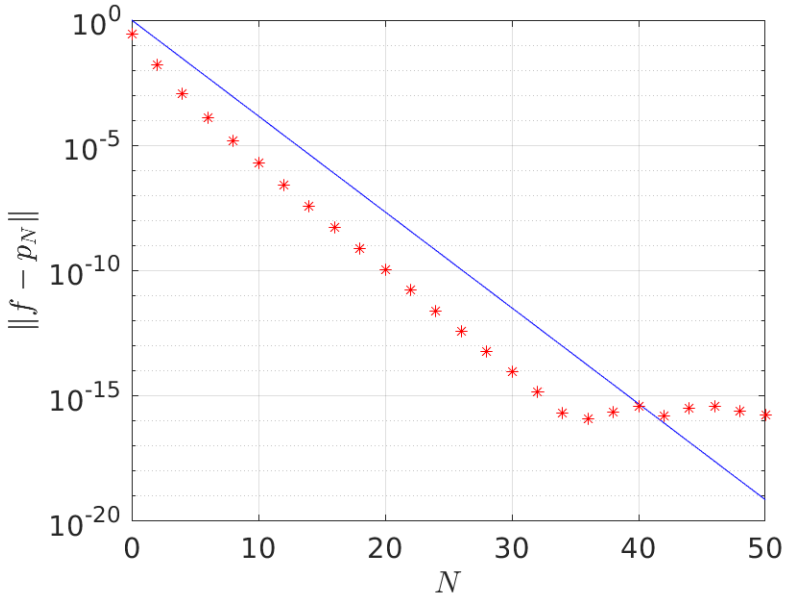

Figure 8: Convergence as $n \to \infty$ of the Chebyshev interpolant to $f(x) = |x - i|$.

Matlab code for this section

```
%% Problem 2 - 8.7 ATAP

rho=1+sqrt(2);
orderAcuracy('sqrt(x^2+1)',50,2,rho)
```

Problem 3

Theorem 6 makes a prediction about the geometric rate of convergence in the third pane of Output 12. Exactly what is this prediction? How well does it match the observed rate of convergence?

Since $f(x) = \frac{1}{1+x^2}$ is analytic everywhere exept $x = \pm i$, where it has singularities, it is analytic on and inside the ellipse with foci $\{-1, 1\}$ and semiminor axis $b < 1$ on which the Chebishev potential takes the value

$$\phi_f = \log\left(\frac{1}{2}(a+b)\right) \approx \log\left(\frac{1}{2}(\sqrt{2}+1)\right),$$

where we have obtained a from the foci identity $f = a^2 - b^2$. Hence, by THEOREM 6, the convergence should be

$$|w_j - u^{(v)}(x_j)| = \mathcal{O}\left(e^{-N(\phi_f + \log 2)}\right) = \mathcal{O}\left(e^{-(\sqrt{2}+1)N}\right),$$

which is proved to be right by the next figure.

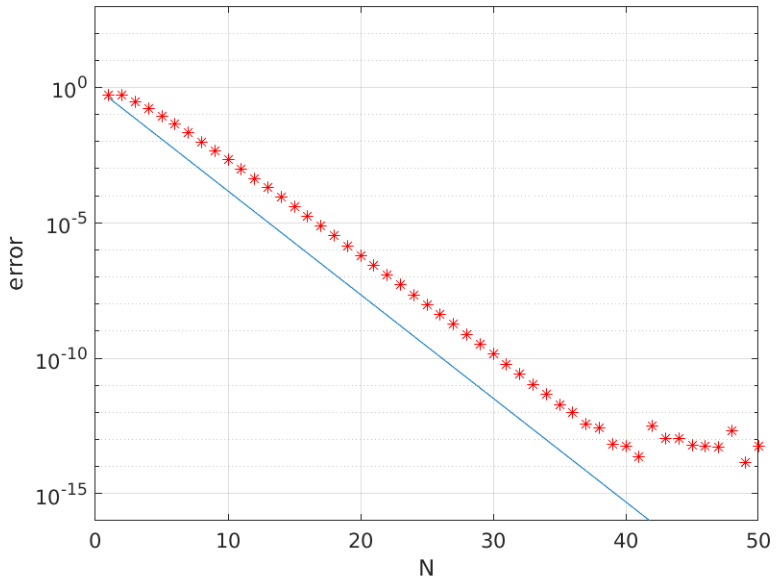

Figure 9: Acuracy of the Chebishev spectral derivative for $f(x) = \frac{1}{1+x^2}$.

Matlab code for this problem

```
%% Problem 3 - 6.7 Trefethen
Nmax = 50; E = zeros(Nmax,1);
for N = 1:Nmax
    [D,x] = cheb(N);
```

```
    v = 1./(1+x.^2); vprime = -2*x.*v.^2;    % analytic in [-1,1]
    E(N) = norm(D*v-vprime,inf);
end
% Define ellipse and potential on it.
a = sqrt(2) ; b = 1;
phif = log(0.5*(a+b));
% Plot results:
figure
semilogy(1:Nmax,E(:),'r*')
hold on
semilogy(1:Nmax,exp(-(phif+log(2))*(1:Nmax)))
axis([0 Nmax 1e-16 1e3]), grid on
set(gca,'xtick',0:10:Nmax,'ytick',(10).^(-15:5:0))
xlabel N, ylabel error
txt='Latex/FIGURES\P6_7';
saveas(gcf,txt,figformat)
```

Problem 4

Modify Program 20 to make use of matricves instead of FFT. Make sure to do this elegantly, using matrix-matrix multiplications rather than explicit loops. You will find that the code gets much shorter, and faster, too. How much faster is it? Does increasing N from 24 to 48 tip the balance

In the following table we have the times spent by the different codes to compute the first derivative (the plotting section was commented out). It can be seen that as N gets large using matrices is more expensive.

N	FFT	Matrices
24	0.1420280	0.0113390
48	1.0236130	3.7638280

Table 1: Time spent in computing the first derivative of f using FFTs and matrices.

Matlab code for this problem

```
format long
% Grid and initial data:
Nvector = [24,48];
time = zeros(length(Nvector),1);
for k = 1:length(Nvector)
    N=Nvector(k);
    [D,x] = cheb(N);  y = x;
    D2 = D^2;
    D2 = D2(2:end-1,2:end-1);
    L = kron(eye(N-1),D2)+kron(D2,eye(N-1));
    dt = 6/N^2;
    [xx,yy] = meshgrid(x(2:N),y(2:N));
    x = xx(:); y = yy(:);

    plotgap = round((1/3)/dt); dt = (1/3)/plotgap;
    u = exp(-40*((x-.4).^2 + y.^2));
    uold = u;

    % Time-stepping by leap frog formula:
    [ay,ax] = meshgrid([.56 .06],[.1 .55]); clf
    tic
    for n = 0:3*plotgap
        t = n*dt;
%         if rem(n+.5,plotgap)<1     % plots at multiples of t=1/3
%           uu = reshape(u,N-1,N-1);
```

```
%           i = n/plotgap+1;
%           subplot('position',[ax(i) ay(i) .36 .36])
%           [xxx,yyy] = meshgrid(-1:1/16:1,-1:1/16:1);
%           uuu = interp2(xx,yy,uu,xxx,yyy,'cubic');
%           mesh(xxx,yyy,uuu), axis([-1 1 -1 1 -0.15 1])
%           colormap(1e-6*[1 1 1]); title(['t = ' num2str(t)]), drawnow
%         end
    % -------------------- %
    % -- Using matrices -- %
    % -------------------- %
        unew = 2*u - uold + dt^2*L*u;
        uold = u; u = unew;
    % --------------- %
    % -- Using fft -- % Done in Program 20
    % --------------- %
    %     uxx = zeros(N+1,N+1); uyy = zeros(N+1,N+1);
    %     ii = 2:N;
    %     for i = 2:N                % 2nd derivs wrt x in each row
    %       v = vv(i,:); V = [v fliplr(v(ii))];
    %       U = real(fft(V));
    %       W1 = real(ifft(1i*[0:N-1 0 1-N:-1].*U)); % diff wrt theta
    %       W2 = real(ifft(-[0:N 1-N:-1].^2.*U));    % diff^2 wrt theta
    %       uxx(i,ii) = W2(ii)./(1-x(ii).^2) - x(ii).* ...
    %                       W1(ii)./(1-x(ii).^2).^(3/2);
    %     end
    %     for j = 2:N                % 2nd derivs wrt y in each column
    %       v = vv(:,j); V = [v; flipud(v(ii))];
    %       U = real(fft(V));
    %       W1 = real(ifft(1i*[0:N-1 0 1-N:-1]'.*U));% diff wrt theta
    %       W2 = real(ifft(-[0:N 1-N:-1]'.^2.*U));   % diff^2 wrt theta
    %       uyy(ii,j) = W2(ii)./(1-y(ii).^2) - y(ii).* ...
    %                       W1(ii)./(1-y(ii).^2).^(3/2);
    %     end
    end
    time(k) = toc;
end
```

Problem 5

Find a way to modify your program of Exercise 8.4, as in Exercise 3.8 but now for a second order problem, to make use of matrix exponentials rather than time discratization. What effect does this have on the computation time?

In this case, just for $N = 24$ it took $t = 25.3899480\ s$ to complete the simulation, a severe increase for such a small number of grid points.

Matlab code for this problem

```
%% Problem 5 - 8.5 Trefethen

% Grid and initial data:

Nvector = [24];
time = zeros(length(Nvector),1);
for k = 1:length(Nvector)
    N=Nvector(k);
    [D,x] = cheb(N);  y = x;
    D2 = D^2;
    D2 = D2(2:end-1,2:end-1);
    L = kron(eye(N-1),D2)+kron(D2,eye(N-1));
    A = [zeros(size(L)) eye(size(L)); L zeros(size(L))];
    dt = 6/N^2;
    [xx,yy] = meshgrid(x(2:N),y(2:N));
    x = xx(:); y = yy(:);

    plotgap = round((1/3)/dt); dt = (1/3)/plotgap;
    u0 = exp(-40*((x-.4).^2 + y.^2));
    u = u0;
    % Time-stepping by leap frog formula:
    [ay,ax] = meshgrid([.56 .06],[.1 .55]); clf
    tic
    for n = 0:3*plotgap
        t = n*dt;
%         if rem(n+.5,plotgap)<1     % plots at multiples of t=1/3
%           uu = reshape(u,N-1,N-1);
%           i = n/plotgap+1;
%           subplot('position',[ax(i) ay(i) .36 .36])
%           [xxx,yyy] = meshgrid(-1:1/16:1,-1:1/16:1);
%           uuu = interp2(xx,yy,uu,xxx,yyy,'cubic');
%           mesh(xxx,yyy,uuu), axis([-1 1 -1 1 -0.15 1])
%           colormap(1e-6*[1 1 1]); title(['t = ' num2str(t)]), drawnow
%         end
```

```
    % ------------------------------ %
    % -- Using matrix exponential -- %
    % ------------------------------ %
        v = expm(A*t)*[u0;zeros(size(u0))];
        u = v(1:length(u0));
    end
    time(k) = toc;
end
```

Problem 6

Explain how the entries of the Chebishev differentiation matrix D_N could be computed by suitable calls to chebfft rather than by explicit formulas as in Theorem 7. What is the asymptotic operation count for this method as $N \to \infty$?
We can express the Chebishev interpolant of f as

$$p_N(x) = \sum_{i=0}^{N} l_j(x) f_j.$$

Thus, the derivative can be expressed as,

$$p'_N(x) = \sum_{i=0}^{N} l'_j(x) f_j.$$

We could call chebfft to compute each derivative $l'_j(x)$, and introducing the results as columns in a matrix. The matrix formed would be the Chebishev differentiation matrix. Since the interpolant requires N flops and the fft requires $N \log N$ assuming that we take full advantage of the algorightm, i.e., having N as a power of 2. In addition, we have to call chebfft N times, so the total number of flops grows as $N^3 \log N$ as $N \to \infty$.

Problem 7

Consider again the approximation of $f(x) = 3/(54cos(x))$, $x \in [-\pi, \pi]$. Let N be the number of nodes in Fourier and polynomial interpolation of this function.

a) **Plot the error as a function of N (on the same figure) for both Chebyshev and Fourier. Notice that Fourier converges at a faster rate in this case.**

b) **Now consider that maximum spacing between nodes: $h = max|x_{i+1} - x_i|$. Plot the error for polynomial and Fourier approximations as a function of h and notice that the rates of convergence are now nearly the same.**

c) **Show that the ratio $h_{cheb}/h_{Fourier}$ is about $\pi/2$.**

For the first part we look at the next figure. We can see that, in fact, Fourier converges at a faster rate in this function.

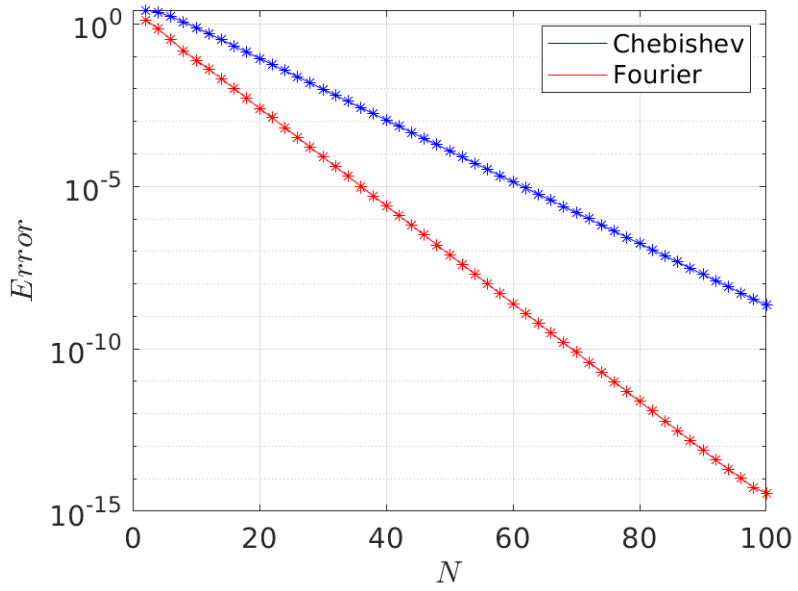

Figure 10: Convergence of the Chebyshev and Fourier interpolants to $f(x) = \frac{3}{54\cos x}$.

We continue by scaling the Chebyshev points to be withing $[-\pi, \pi]$ and calculate h for both Chebishev and Fourier, for each N. We obtain the following figure, which shows that the rates of convergence are nearly the same.

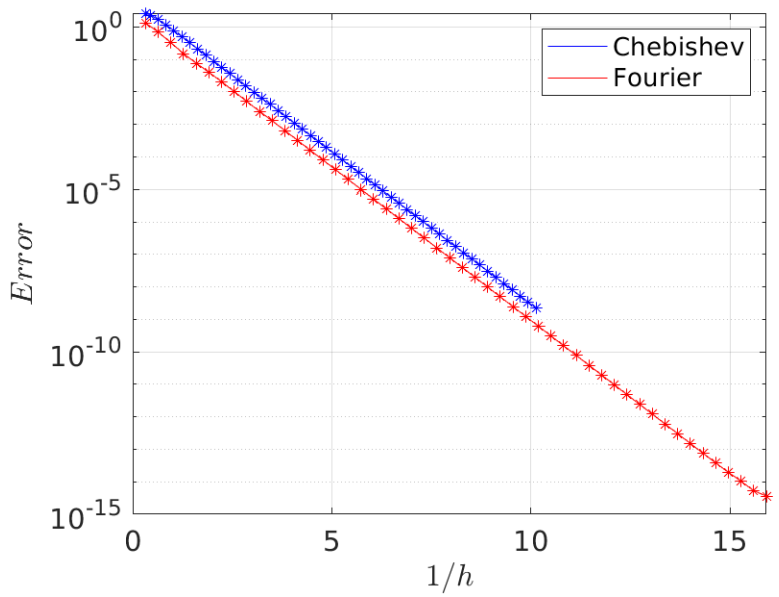

Figure 11: Convergence of the Chebyshev and Fourier interpolants to $f(x) = \frac{3}{54\cos x}$.

Lastly, in the following figure we see that, once is large enough, $h_{Cheb}/h_{Fourier} \approx \pi/2$.

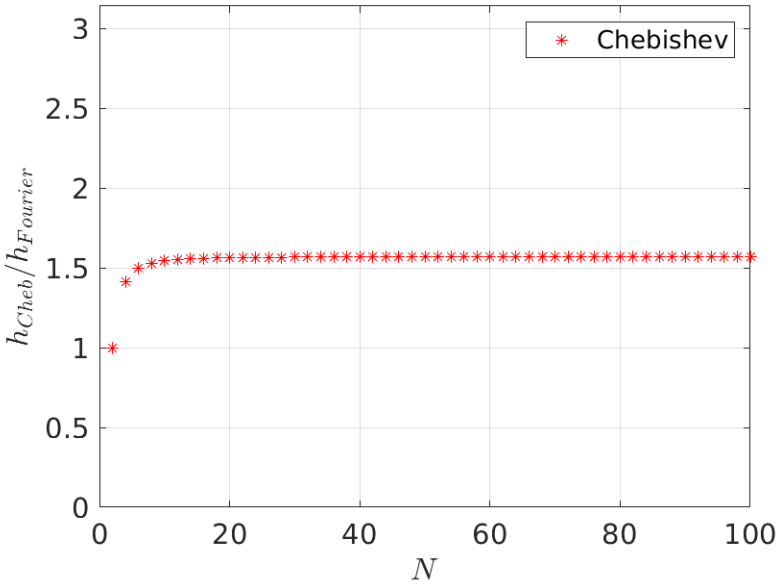

Figure 12: $h_{Cheb}/h_{Fourier}$ for $f(x) = \frac{3}{54\cos x}$.

Matlab code for this problem

```
%% Problem 7
close all
```

```
f = chebfun('3/(5-4*cos(x))',[-pi,pi]);
plot(f)
grid on
N = 2:2:50;
for k = 1:length(N)
    fcheb = chebfun('3/(5-4*cos(x))',[-pi,pi],N(k));
    ffour = chebfun('3/(5-4*cos(x))',[-pi,pi],N(k),"trig");
    errcheb(k) = norm(f-fcheb,inf);
    errfour(k) = norm(f-ffour,inf);
    % b
    [~,x] = cheb(N(k)); x = pi*x;
    hcheb(k) = max(abs(x(2:end)-x(1:end-1)));
    hfour(k) = 2*pi/(N(k));
end
% a
figure
semilogy(N,errcheb,'b',N,errfour,'r')
hold on
semilogy(N,errcheb,'b*',N,errfour,'r*')
grid on
xlabel('$N$','interpreter','latex')
ylabel('$Error$','interpreter','latex')
set(gca,'fontsize',labelfontsize)
legend('Chebishev', 'Fourier')
txt='Latex/FIGURES/P7_a';
saveas(gcf,txt,figformat)
% b
figure
semilogy(hcheb.^(-1),errcheb,'b',hfour.^(-1),errfour,'r')
hold on
semilogy(hcheb.^(-1),errcheb,'b*',hfour.^(-1),errfour,'r*')
grid on
xlabel('$1/h$','interpreter','latex')
ylabel('$Error$','interpreter','latex')
set(gca,'fontsize',labelfontsize)
legend('Chebishev', 'Fourier')
txt='Latex/FIGURES/P7_b';
saveas(gcf,txt,figformat)
% c
figure
plot(N,hcheb./hfour,'r*')
grid on
axis([0 50 0 pi])
xlabel('$N$','interpreter','latex')
ylabel('$h_{Cheb}/h_{Fourier}$','interpreter','latex')
```

```
set(gca,'fontsize',labelfontsize)
legend('Chebishev', 'Fourier')
txt='Latex/FIGURES/P7_c';
saveas(gcf,txt,figformat)
```