Spectral Methods

Francisco Castillo
Homework 4

April 11, 2019

Problem 1

Solve the boundary value problem $u_{xx} + 4u_x + e^x u = \sin(8x)$ numerically on $[-1, 1]$ with boundary conditions $u(\pm 1) = 0$. To 10 digits of accuracy, what is $u(0)$?

We can express the BVP as

$$\begin{aligned} D^2u + 4Du + \text{diag}(e^x)u &= f, \\ \left[D^2 + 4D + \text{diag}(e^x)\right] u &= f, \\ Mu &= f. \end{aligned}$$

where D is the Chebyshev differentiation matrix and f is the right hand side of the equation. Hence, we can calculate the solution to our BVP as

$$u = M^{-1} f.$$

The solution, using $N + 1 = 51$ points (odd number of points to have a node exactly at $x = 0$), is plotted in the figure 1. In the figure 2 we plot the value of $u(x = 0)$ for different values of N. We see how with just 10+ nodes we achieve high accuracy. The value of $u(x = 0)$ obtained is

$$u(0) = 0.0095978572.$$

The number of nodes needed to achieve 10 digits of accuracy was $N + 1 = 23$.

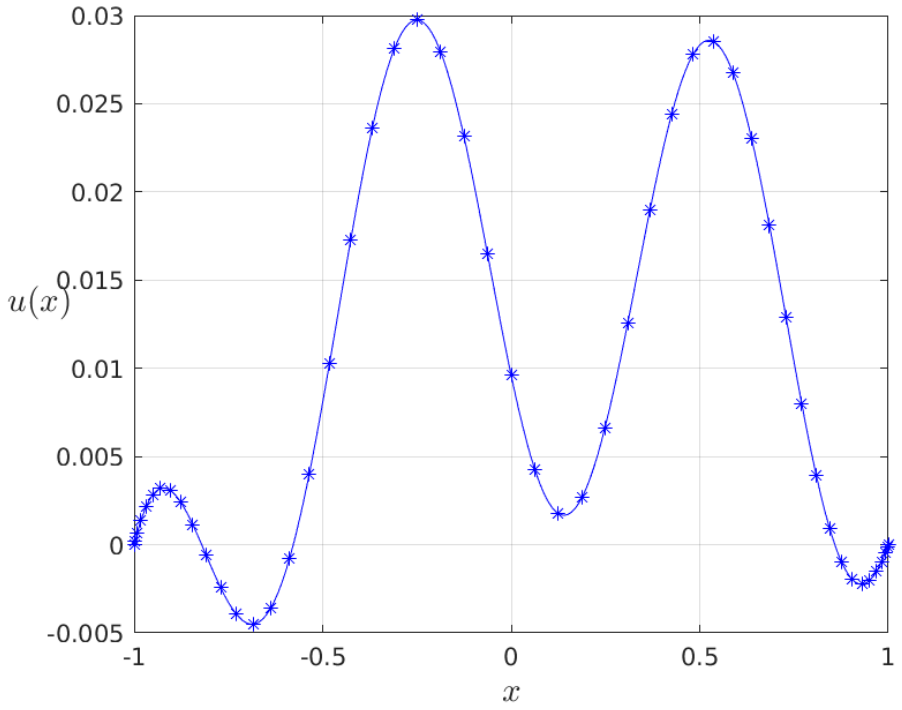

Figure 1: Solution to the BVP using Chebyshev matrices and 51 nodes.

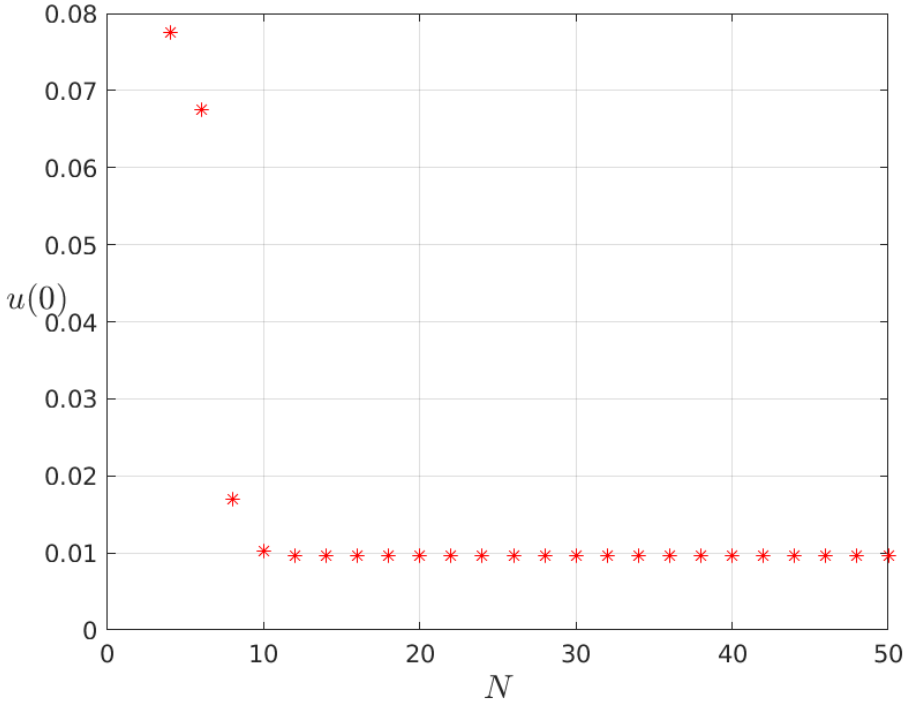

Figure 2: Value of $u(0)$ for different number of nodes.

Matlab code for this problem

```
%% Homework 4, Problem 1 - Francisco Castillo
clear all; close all; clc;
labelfontsize = 14;

NN = 4:2:50;
for j=1:length(NN)
    N=NN(j);
    [D,x] = cheb(N); % D:(N+1)x(N+1), x:(N+1)x1
    D2 = D^2;
    D = D(2:N,2:N);
    D2 = D2(2:N,2:N);       % Zero Dirichlet BCs
    f = sin(8*x(2:N));
    E = diag(exp(x(2:N)));
    u = (D2+4*D+E)\f;
    u = [0;u;0];
    u0(j) = u(N/2+1);
end

figure
plot(NN,u0,'r*')
grid on
xlabel('$N$','fontsize',labelfontsize,'interpreter','latex')
ylabel('$u(0)$','fontsize',labelfontsize,'interpreter','latex')
set(get(gca,'ylabel'),'rotation',0)
txt = 'Latex/FIGURES/P1_a';
saveas(gcf,txt,'png')

xx = [x(end):.01:x(1)]';
uu = polyval(polyfit(x,u,N),xx);

figure
plot(x,u,'b*')
hold on
plot(xx,uu,'b')
grid on
xlabel('$x$','fontsize',labelfontsize,'interpreter','latex')
ylabel('$u(x)$','fontsize',labelfontsize,'interpreter','latex')
set(get(gca,'ylabel'),'rotation',0)
txt = 'Latex/FIGURES/P1_b';
saveas(gcf,txt,'png')
```

Problem 2

Consider the first order linear initial boundary value problem

$$u_t = u_x, \qquad x \in [-1,1], \quad t > 0, \quad u(\pm 1, t) = u(x,0) = 0,$$

with initial data $u(x,0) = \exp(-60(x-1/2)^2)$**. Write a program to solve this problem by a matrix-based Chebyshev spectral discretization in** x **coupled with the third order Adams-Bashforth formula in** t**,**

$$u^{(n+3)} = u^{(n+2)} + \frac{1}{12}\Delta t \left(23 f^{(n+2)} - 16 f^{(n+1)} + 5 f^{(n)}\right).$$

Initial values can be supplied from the exact solution. Take $N = 50$ **and** $\Delta t = \nu N^{-2}$**, where** ν **is a parameter. For each of the two choices** $\nu = 7$ **and** $\nu = 8$**, produce one plot of the computed solution at** $t = 1$ **and another that superimposes the stability region in the** $\lambda \Delta t$**-plane, the eigenvalues of the spatial discretization matrix, and its** ϵ**-pseudospectra fpr** $\epsilon = 10^{-2}, 10^{-3}, \ldots, 10^{-6}$**. Comment on the results.**

We start by finding the exact solution using characteristics,

$$u(x,t) = \begin{cases} e^{-60(x+t-1/2)^2}, & x \geq -t, \\ e^{-60(x+t+1/2)^2}, & x \leq -t. \end{cases}$$

Further, we use Chebyshev differentiation matrices,

$$\begin{aligned} u_t &= u_x, \\ u_t &= Du = f, \end{aligned}$$

and the AB time-stepping method:

$$\begin{aligned} u^{(n+3)} &= u^{(n+2)} + \frac{1}{12}\Delta t \left(23 f^{(n+2)} - 16 f^{(n+1)} + 5 f^{(n)}\right), \\ &= u^{(n+2)} + \frac{1}{12}\Delta t D \left(23 u^{(n+2)} - 16 u^{(n+1)} + 5 u^{(n)}\right). \end{aligned}$$

To start the simulation we use the exact solution. The results obtained are shown in the following figure. In the top two figures we see the exact solution (red) superposed with the numberical solution (blue). As we can see, for $\nu = 7$ (left) we have a correct numerical solution while for $\nu = 8$ (right) the numerical solution blows up. This can be further understood looking at the bottom figures. As we can see in the bottom left graph, the eigenvalues near the border of the stability region are susceptible to perturbations. Hence, by just increasing ν to 8, those eigenvalues are now outside of the stability region (bottom-right figure), giving us an understanding of why the numerical simulation is unstable.

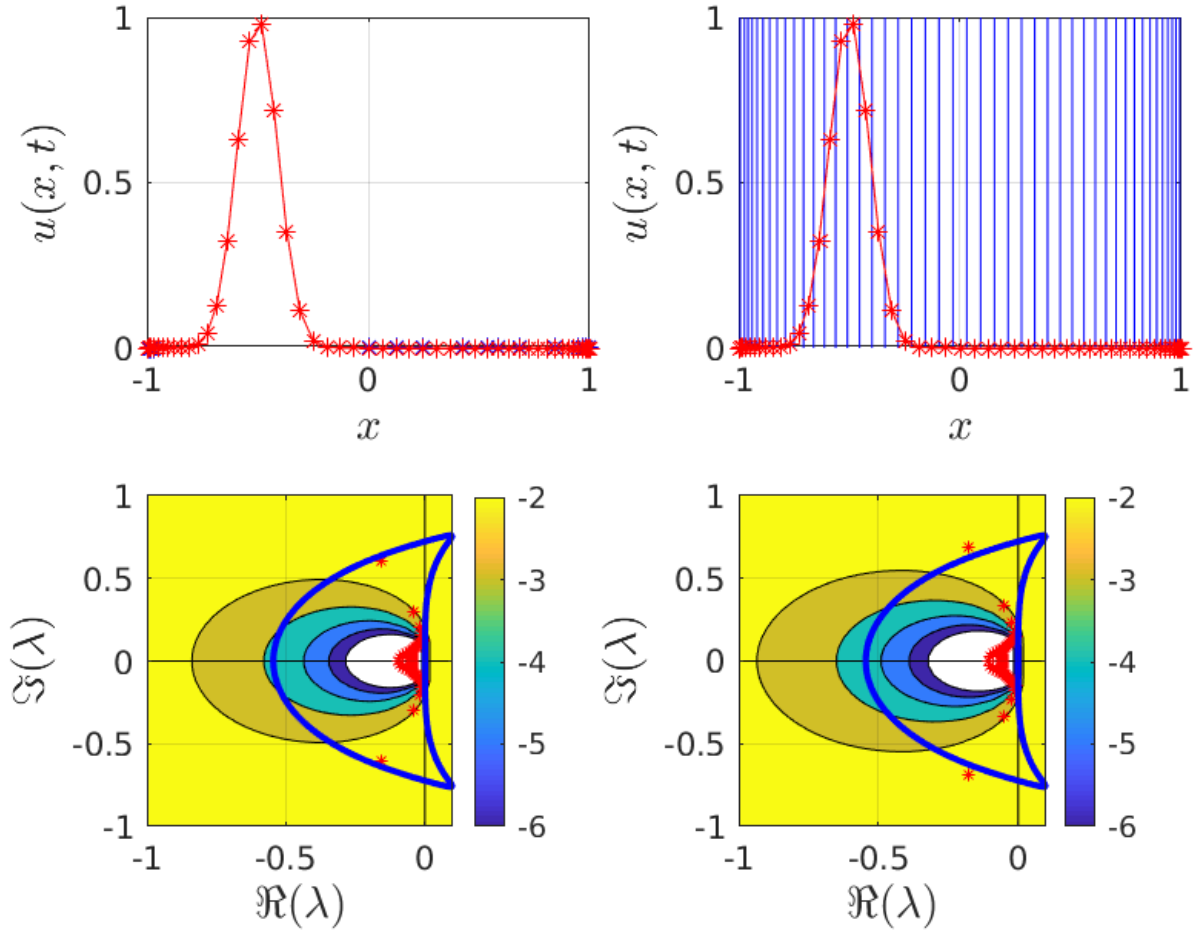

Figure 3: Numerical and analytical solution of the PDE for $\nu = 7$ and $\nu = 8$ (top) and pseudo-spectra with stability regions and eigenvalues in the $\lambda\Delta t$-plane for those two values of ν (bottom).

Matlab code for this problem

```
%% Homework 4, Problem 2 - Francisco Castillo'
clear all; close all; clc;
labelfontsize = 14;
markersize = 4;
linewidth = 2;

N = 50;
V = [7,8];
levels = log10([1e-2 1e-3 1e-4 1e-5 1e-6]);
% figure('units','normalized','outerposition',[0 0 1 1])
for j=1:length(V)
```

```
v = V(j);
dt = v/N^2;
tf = 1;

[D,x] = cheb(N); % D:(N+1)x(N+1), x:(N+1)x1
D = D (2:end,2:end);
u_n = analytic_sol(x(2:end),0);
u_n1 = analytic_sol(x(2:end),dt);
u_n2 = analytic_sol(x(2:end),2*dt);

t = 2*dt;
while t<tf
    if (t+dt>tf)
        dt=tf-t;
        t=t+dt;
    else
        t=t+dt;
    end

    u_n3 = u_n2+dt*D*(23*u_n2-16*u_n1+5*u_n)/12;
    u = [analytic_sol(1,t);u_n3];

    u_n = u_n1;
    u_n1 = u_n2;
    u_n2 = u_n3;

    res = analytic_sol(x,t);

    subplot(2,2,j)
    plot(x,u,'b*')
    hold on
    plot(x,u,'b')
    plot(x,res,'r*')
    plot(x,res,'r')
    grid on
    axis([-1 1 0 1])
    xlabel('$x$','interpreter','latex','fontsize',labelfontsize)
    ylabel('$u(x,t)$','interpreter','latex','fontsize',labelfontsize)
    hold off
    shg
end
%%
% Adams-Bashforth stability region and pseudo-spectra of this problem
dt = v/N^2;
ee = eig(D);
```

```
    dtee = dt*ee;
    xr = 6.5*linspace(min(real(dtee)),0.1,50);
    yr = 2*linspace(min(imag(dtee)),max(imag(dtee)),50);
    [xx,yy] = meshgrid(xr,yr);
    zz = xx+1i*yy;
    ps = 0*zz;
    for k=1:numel(zz)
        ps(k) = min(svd(eye(size(D))*zz(k)-dt*D));
    end

    z = exp(1i*pi*(0:200)/100); r = z-1;
    s = (23-16./z+5./z.^2)/12;


    subplot(2,2,j+2)
    contourf(xx,yy,log10(ps),levels)
    c = colorbar;
    hold on
    plot([-8,8],[0,0],'k')
    plot([0,0],[-8,8],'k')
    plot(real(dtee),imag(dtee),'r*','markersize',markersize)
    plot(r./s,'b','linewidth',linewidth)
    xlabel('$\Re(\lambda)$','interpreter','latex','fontsize',labelfontsize)
    ylabel('$\Im(\lambda)$','interpreter','latex','fontsize',labelfontsize)
    axis([-1 0.1 -1 1])
    grid on
end

saveas(gcf,'Latex/FIGURES/P2','png')

function res = analytic_sol(x,t)
    res = zeros(size(x));
    res(x>=-t) = exp(-60*(x(x>=-t)+t-0.5).^2);
    res(x<-t) = exp(-60*(x(x<-t)+t+0.5).^2);
end
```

Problem 3

Consider the nonlinear initial value problem

$$u_t = u_{xx} + e^u, \qquad x \in [-1, 1], \quad t > 0, \quad u(\pm 1, t) = u(x, 0) = 0,$$

for the unknown function $u(x, t)$. To at least eight digits of accuracy, what is $u(0, 3.5)$, and what is the time t_5 such that $u(0, t_5) = 5$.

The numerical scheme used in this problem is the following,

$$u^{n+1} = u^n + \Delta t \left[D u^n + e^{u^n} \right].$$

We have obtained the solutions shown in figure 4 and the following results,

$$t_5 = 3.53594879,$$
$$u(0, 3.5) = 3.53878310.$$

It is evident given the results that the solutions starts developing slowly but, because of the exponential non-linear term, it accelerates quickly. This can be appreciated by realizing that it takes the solution $t = 3.5$ to reach the red curve, and only $t_5 - 3.5 = 0.03594879$ to get to the blue curve.

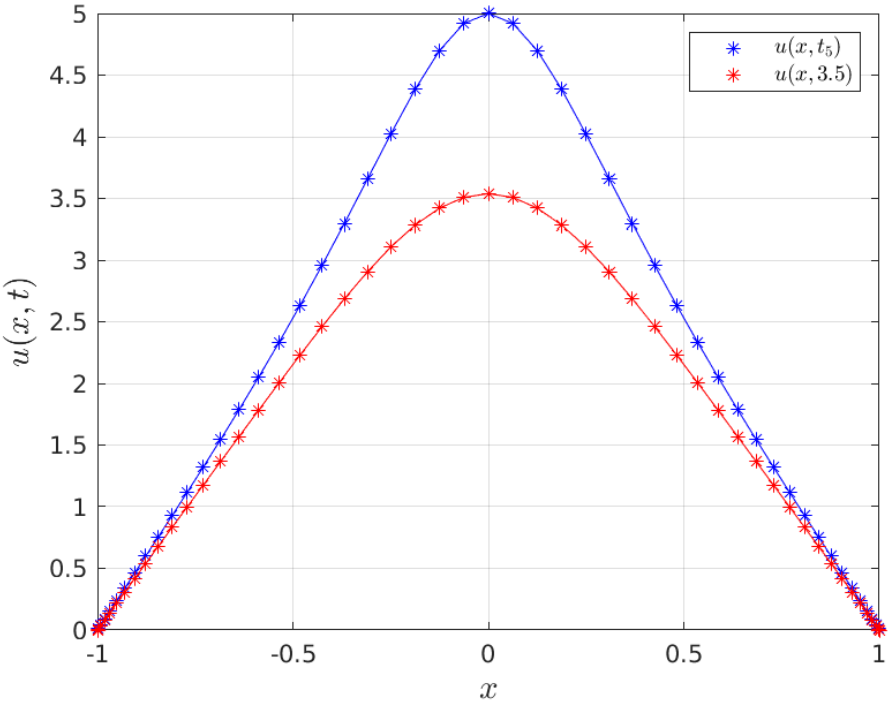

Figure 4: Numerical solution of the non-linear PDE at $t = 3.5$ and at t_5, time when $u(0) = 5$.

Matlab code for this problem

```
%% Homework 4, Problem 3 - Francisco Castillo'
clear all; close all; clc;
labelfontsize = 14;
linewidth = 2;

N = 50;
dt = 8e-1/N^3;
[D,x] = cheb(N); % D:(N+1)x(N+1), x:(N+1)x1
x = x(2:N);
D2=D^2;
D2 = D2(2:N,2:N);
u = zeros(size(x));
u35 = u;
u0 = u(N/2);
tobs = [3.5 100];
t = 0;
k=1;
igraph = 1000;
tstep = 0;

while u0<5
    if (t+dt>tobs(k))
        dt=tobs(k)-t;
        t=t+dt;
        k=k+1;
    else
        t=t+dt;
        dt = 8e-1/N^3;
    end

    u = u +dt*(D2*u+exp(u));
    u0 = u(N/2);
    tstep = tstep+1;
    if t == tobs(1)
        u35 = u;
    end
    if (mod(tstep,igraph)==0 || round(u0,10)>=5)
        h1 = plot([1;x;-1],[0;u;0],'b*');
        hold on
        plot([1;x;-1],[0;u;0],'b')
        if (u35~=0)
            h2 = plot([1;x;-1],[0;u35;0],'r*');
            plot([1;x;-1],[0;u35;0],'r')
```

```
        end
        grid on
        axis([-1 1 0 5])
        xlabel('$x$','interpreter','latex','fontsize',labelfontsize)
        ylabel('$u(x,t)$','interpreter','latex','fontsize',labelfontsize)
        hold off
        shg
    end
end
legend([h1 h2],'$u(x,t_5)$', '$u(x,3.5)$','interpreter','latex')
saveas(gcf,'Latex/FIGURES/P3','png')
```

Problem 4

Download the program wave2D_leap_frog.m. This code solves the wave equation,

$$u_{tt} = u_{xx} + u_{yy}, \qquad (x, y) \in [-1, 1] \times [-1, 1], \quad t > 0,$$

with homogeneous Dirichlet boundary conditions and initial conditions

$$u(0, x, y) = exp(-40((x0.2)^2 + y^2)), \quad u_t(0, x, y) = 0.$$

Modify this code to solve the same problem but with homogeneous Neumann boundary instead of Dirichlet. Plot your solution at time $t = 10$. How accurate is your solution? (number of digits)

To impose homogeneous Neumann boundary conditions we can simply ignore some rows and colums on our matrix multiplications. Taking u_{yy}:

$$u_{yy} = DDu = Du_y,$$

where D is the Chebyshev differentiation matrix. We know that u_y has zero components in the first and last rows. This allows us to ignore the first and last rows of D when taking the first multiplication. In addition, we can also ignore the first and last column of the second multiplication. Hence, we can simply impose Neumann boundary conditions by making,

$$u_{yy} = D(:, 2 : N)D(2 : N, :)u,$$

following MATLAB notation. Analogous discussion can be made about u_{xx}, obtaining

$$u_{yy} = uD(2 : N, :)'D(:, 2 : N)',$$

where the D' represents the transpose of D. The solution at $t = 10$ is shown in the next figure.

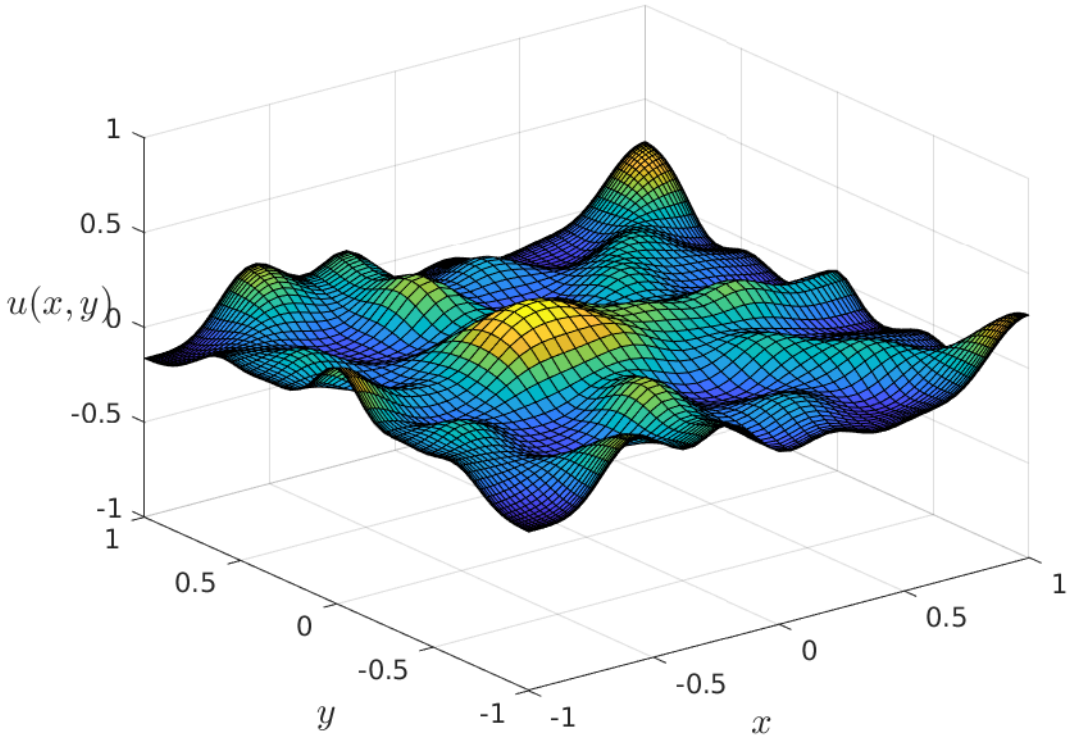

Figure 5: Numerical solution to the wave equation with homogeneous Neumann boundary conditions at $t = 10$.

Matlab code for this problem

```
%% Homework 4, Problem 4 -  Francisco Castillo
clear all; close all; clc
labelfontsize = 14;
%% 2D wave equation Chebyshev+Leap-frog, ZERO Neumann BC'S
N = 64;
[D,x] = cheb(N);
% x = x(2:end-1);
% D2 = D^2;
% D2 = D2(2:end-1,2:end-1);

% 2D grid
[X,Y] = meshgrid(x);

% Initial condition
u0 = exp(-40*((X-0.2).^2+Y.^2));
u = u0;
```

```
h = 1-x(2);
dt = h/2;
t = 0;
tf = 10;
count=0;

while t<tf
    if t+dt>tf
        dt = tf-t;
    else
        dt = h/2;
    end
    uyy = D(:,2:N)*D(2:N,:)*u;
    uxx = u*D(2:N,:)'*D(:,2:N)';
    u2 = 2*u- u0 + dt^2*(uxx+uyy);
    u0 = u;
    u = u2;

    if count == 10
        surf(X,Y,u)
        zlim([-1 1])
        drawnow
        shg
        count = 0;
    end

    count = count+1;
    t = t+dt;

    if t == tf
        surf(X,Y,u)
        zlim([-1 1])
        drawnow
        shg
        xlabel('$x$','interpreter','latex','fontsize',labelfontsize)
        ylabel('$y$','interpreter','latex','fontsize',labelfontsize)
        zlabel('$u(x,y)$','interpreter','latex','fontsize',labelfontsize)
        set(get(gca,'ZLabel'),'Rotation',0)
        saveas(gcf,'Latex/FIGURES/P4','png')
    end
end
```

Problem 5

Modify the code fluidflow.m to solve the equations

$$\begin{aligned} \omega_t + \psi_y \omega_x - \psi_x \omega_y &= Pr\Delta\omega + RaPrT_x, \\ T_t + \psi_y T_x - \psi_x T_y &= \Delta T, \\ \Delta\psi &= -\omega, \end{aligned}$$

where $(x, y) \in (0, 1) \times (0, 1)$ and $t > 0$. Here ω is the fluid vorticity, ψ the stream function and T the temperature. The independent non-dimensional constants are the Rayleigh number Ra, which reflects the buoyant contribution, and the Prandtl number Pr, which is the ratio of viscous to thermal diffusion. For this exercise, set to $Ra = 2 \cdot 10^5$ and $Pr = 0.71$ (air). The fluid is at rest at $t = 0$, with $T = \psi = \omega = 0$. The boundary condition for the stream function is $\psi|_\Gamma = 0$, which implies that there is no mass transfer through the boundary Γ. The value of the vorticity at the walls is expressed as $\omega_\Gamma = -\Delta\psi|_\Gamma$ and the temperature at Γ is defined by

$$T(t, x, y) = \begin{cases} 2^9 \tanh^4(100t) x^5 (x-1)^4, & y = 0, x \in [0, 1], t > 0, \\ 0, & (x, y) \in \Gamma, y \neq 0, t > 0. \end{cases}$$

We first compute the vorticity from the velocity field,

$$\omega = vD_p - Du,$$

where D_p is the transpose of D, the Chebyshev differentiation matrix. Then we advance the vorticity and the temperature,

$$\begin{aligned} \omega &= \omega + \Delta t \left[-v.D\omega - u.\omega D_p + Pr\left(wD_{2p} + D_2\omega\right) + RaPrTD_p \right], \\ T &= T + \Delta t \left[-v.DT - u.TD_p + TD_{2p} + D_2T \right]. \end{aligned}$$

Further, we recompute the stream function in the interior (the exterior keeps being zero). To solve the Poisson equation we are going to use the Sylvester equation and algorithm instead of LU factorization (see figures 6 and 7 for comparison between the two methods).

$$\begin{aligned} D_2\psi + \psi D_{2p} &= -w, \\ A\psi + \psi B &= C. \end{aligned}$$

To finish, we recalculate the velocities from the streamfunction,

$$\begin{aligned} u &= D\psi, \\ v &= -\psi D_p, \end{aligned}$$

and impose boundary conditions on the velocities and the temperature,

$$\begin{aligned} u_\Gamma &= 0, \\ v_\Gamma &= 0, \\ T_\Gamma &= 0, \\ T_\Gamma(t, x, y = 0) &= 2^9 \tanh^4(100t) x^5 (x-1)^4. \end{aligned}$$

In the following figure we show the ratio between the time spent in solving the Poisson equation using LU factorization and Sylvester's algorithm versus the number of iterations. The red lines indicate the average and the standard deviation of the date. We can see that using Sylvester's equation leads to an improvement of a factor greater than 3. In figure 7 we can see the difference in the solutions is of the order of 10^{-11}, concluding that it is advisable for this problem to use the Sylvester's approach to solve the Poisson equation.

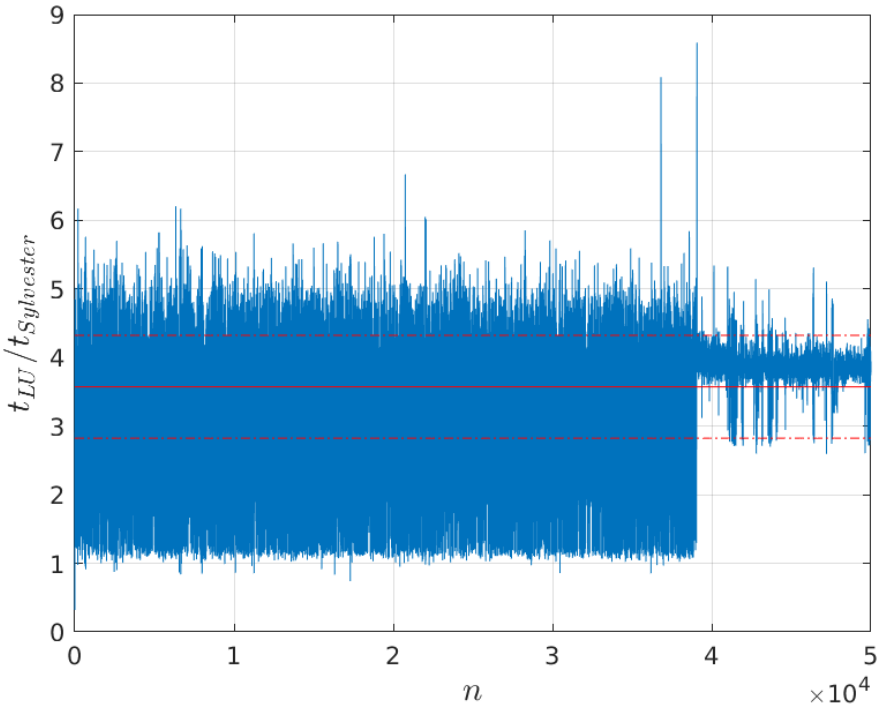

Figure 6: Performance comparison between LU and Sylvester's algorithm for the Poisson equation.

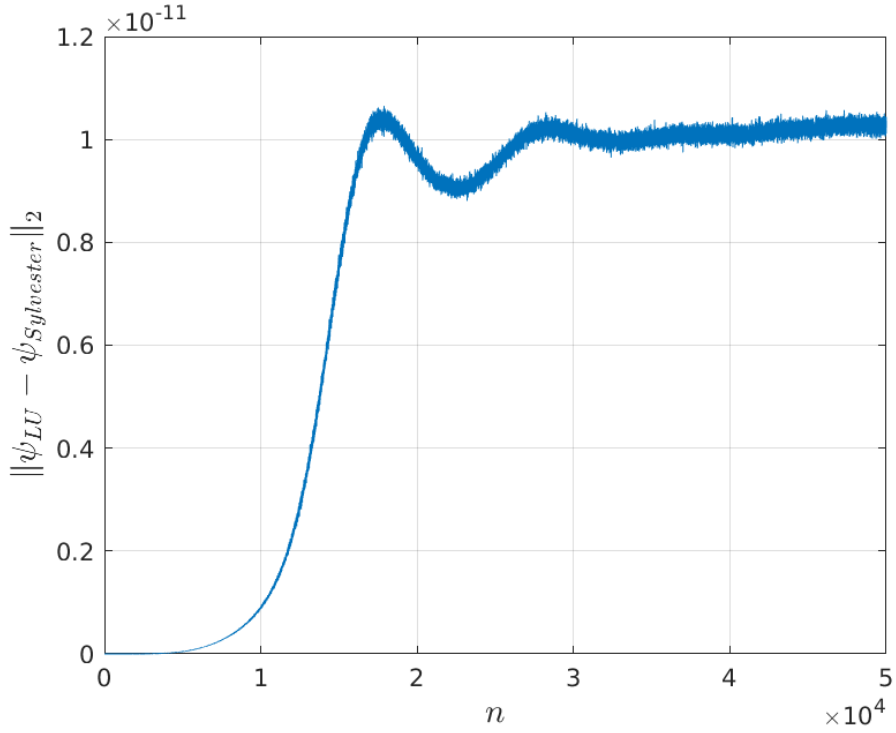

Figure 7: Differnce between the ψ obtained using LU and Sylvester's algorithm.

In the next figure we see how the natural convection makes the fluid move. The hot floor makes the fluid ascend and given the no divergence condition and the non-uniformity of the temperature at the base, it is forced to rotate as we can see from the streamlines, vorcitity and velocity fields. We can see a very different situation if the hot surface is placed on top. Given the blocked convection the fluid would not move as much, as we see in figure 9. We can still see that the fluid moves, but its movement is confined to the upper part of the domain. This results are at $t = 0.1$.

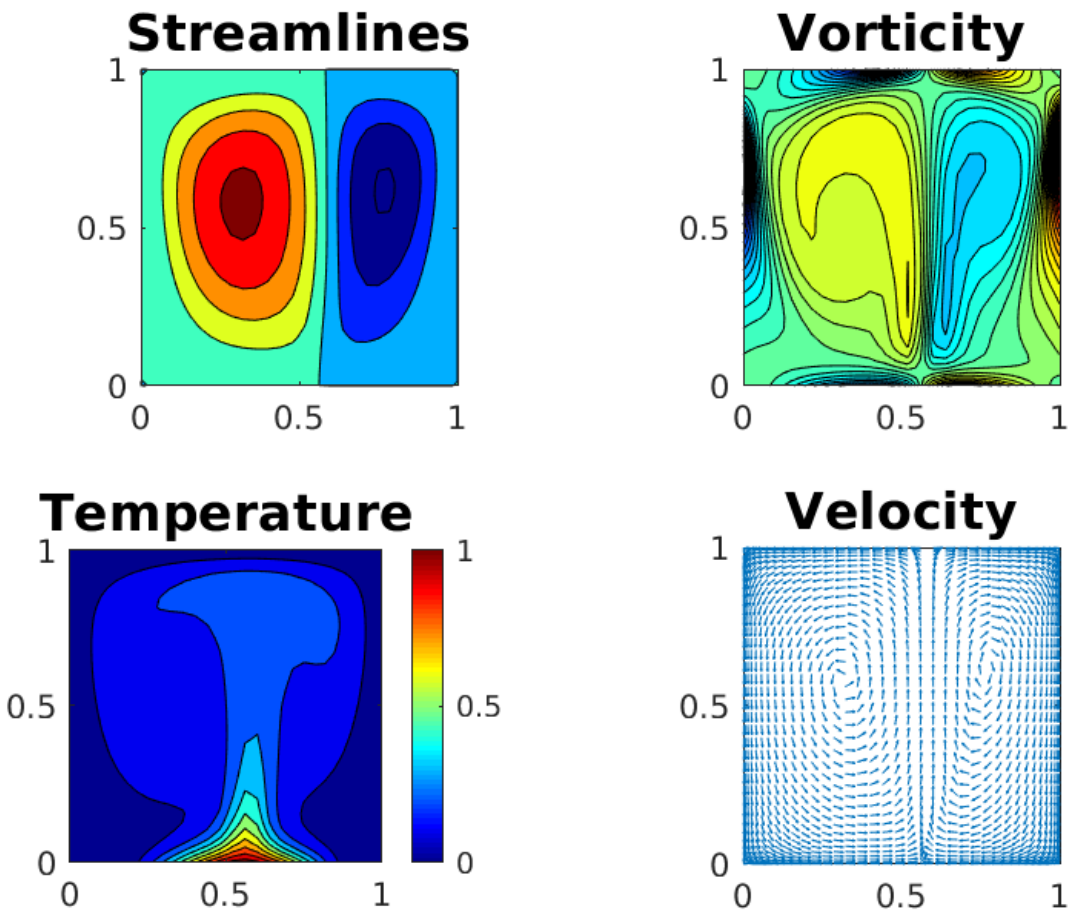

Figure 8: Results obtained for natural convection at $t = 0.1$.

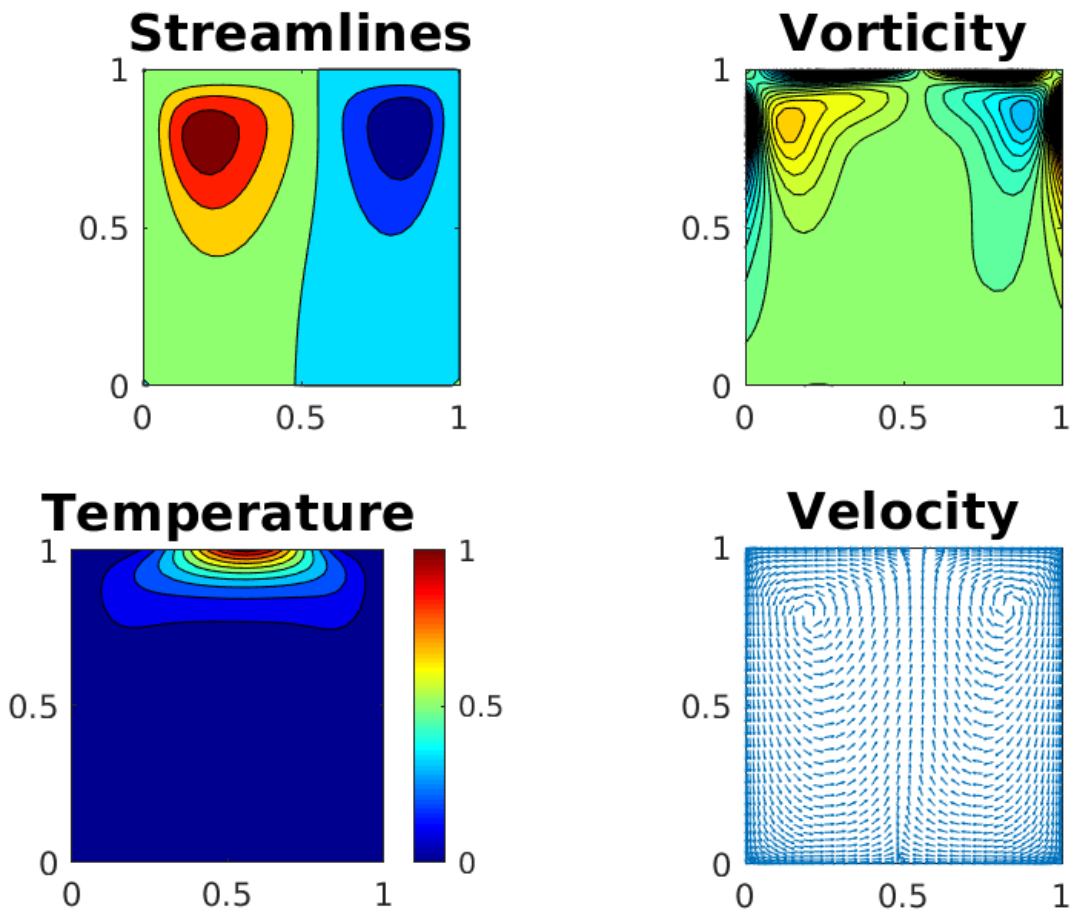

Figure 9: Results obtained for blocked convection at $t = 0.1$.

Matlab code for this problem

```
%% Homework 4, Problem 5 -  Francisco Castillo
clear all; close all; clc;

% Parameters
N = 40;  %40
Pr = 0.71;
Ra = 2e5;
dt = 2e-6;

% Grid and diff matrices
[D,xch] = cheb(N-1);
x = (-xch+1)/2; D = -2*D;   % To trasnlate the domain to [0,1]
[xx,yy] = meshgrid(x);
```

```
Dp = D';
D2 = D^2; D2p = D2';
indb = find(xx==0|xx==1|yy==0|yy==1); % For what?? Impose BCs in velocities


% Laplacian of the interior
L = kron(eye(N-2),D2(2:end-1,2:end-1))+kron(D2(2:end-1,2:end-1),eye(N-2));
%Linv = inv(L);
[lo,up,per] = lu(L,'vector');  % LU factorization

% Initial velocity & pre-allocate memory
T = 0*xx;
psi = T;
psi2 = T;
w = T;
u = T;
v = T;
% Everything initialized to zero
count = 0;
t = 0;

% boundary condition
Topt = 'blocked convection';
if (strcmp(Topt,'natural convection'))
    T(1,:) = TempBC(t,x);
elseif (strcmp(Topt,'blocked convection'))
T(end,:) = TempBC(t,x);
end

% main loop
i=0;
while t<.1%200
    i=i+1;
    % vorticity
    w = v*Dp-D*u;  % w = dvdx-dudy

    % Advance Vorticity
    w =  w + dt*(-v.*(D*w)-u.*(w*Dp)+Pr*(w*D2p+D2*w)+Ra*Pr*T*Dp);

    % Advance Temperature
    T = T + dt*(-v.*(D*T)-u.*(T*Dp)+T*D2p+D2*T);

    % compute stream function
    %    tic
```

```
%    wi = w(2:end-1,2:end-1); wi=wi(:);
%    psi(2:end-1,2:end-1) = reshape(up\(lo\(-wi(per))),N-2,N-2);
%    time1(i) = toc;
%    tic
psi(2:end-1,2:end-1) = sylvester(D2(2:end-1,2:end-1),D2p(2:end-1,2:end-1),-w(2:end-1
%    time2(i) = toc;
%    diff(i) = norm(psi-psi2);

% Update Velocity
u = D*psi;
v = -psi*Dp;

% BC's for u,v,T. Vorticity is calculated from u,v. Stream-function is
% obtained from w.
u(indb) = 0;
v(indb) = 0;
T(:,1) = 0;
T(:,end) = 0;
if (strcmp(Topt,'natural convection'))
    T(1,:) = TempBC(t,x);
    T(end,:) = 0;
elseif (strcmp(Topt,'blocked convection'))
    T(1,:) = 0;
    T(end,:) = TempBC(t,x);
end

% Advance time
t = t+dt;

count = count + 1;
if count == 200

    subplot(2,2,1)
    contourf(xx,yy,psi)
    axis([0 1 0 1]), axis square
    colormap(hot)
    title('Streamlines','fontsize',16)

    subplot(2,2,2)
    contourf(xx,yy,w,30)
    axis([0 1 0 1]), axis square
    title('Vorticity','fontsize',16)

    subplot(2,2,3)
    contourf(xx,yy,T)
```

```
        axis([0 1 0 1]), axis square
        title('Temperature','fontsize',16)
        colormap(jet)
        colorbar
        caxis([0 1])

        speed = sqrt(u.^2+v.^2);

        subplot(2,2,4)
        quiver(xx,yy,u./speed,v./speed)
        axis([0 1 0 1]), axis square
        title('Velocity','fontsize',16)

        drawnow

        count = 0;
    end

end
if (strcmp(Topt,'natural convection'))
    saveas(gcf,'Latex/FIGURES/P5','png')
elseif (strcmp(Topt,'blocked convection'))
    saveas(gcf,'Latex/FIGURES/P5_blocked','png')
end

%%
% n = 1:i;
% figure
% plot(n,time1./time2)
% hold on
% plot(n,mean(time1./time2)*ones(size(n)),'r')
% plot(n,(mean(time1./time2)+std(time1./time2))*ones(size(n)),'r-.')
% plot(n,(mean(time1./time2)-std(time1./time2))*ones(size(n)),'r-.')
% grid on
% xlim([0 n(end)])
% xlabel('$n$','interpreter','latex','fontsize',14)
% ylabel('$time_1/time_2$','interpreter','latex','fontsize',14)
% ylabel('$t_{LU}/t_{Sylvester}$','interpreter','latex','fontsize',14)
% saveas(gcf,'Latex/FIGURES/P5_comparison','png')
%
% figure
% plot(n,diff)
% grid on
% xlabel('$n$','interpreter','latex','fontsize',14)
% ylabel('$\|\psi_{LU}-\psi_{Sylvester}\|$','interpreter','latex','fontsize',14)
```

```
% ylabel('$\|\psi_{LU}-\psi_{Sylvester}\|_2$','interpreter','latex','fontsize',14)
% xlim([0 n(end)])
% saveas(gcf,'Latex/FIGURES/P5_diff','png')


function Ty0 = TempBC(t,x)
    Ty0 = 2^9*(tanh(100*t))^4*x.^5.*(x-1).^4;
end
```