Math 4NA3 - Assignment 6

April 14, 2021

Question 1:

a) Using the class notes, we know that the eigenvalues of the coefficient matrix are of the form

$$\lambda_j = \frac{-4}{h^2}\sin^2\left(\frac{j\pi h}{2}\right)$$

Now for Euler stability we require the following

$$k\lambda_{\text{Max}} \leq -2$$

$\iff$

$$k \leq 2\frac{h^2}{4} = \frac{h^2}{2}$$

Thus, the maximum step-size is $k = \frac{h^2}{2}$.

b) To implement the method of lines for the given problem via Euler's method we have

```
in = [0;0.2;0.4;0.6;0.8;1;0.8;0.6;0.4;0.2];
h = 0.1;
JM = 10;
a = -2;
b = 1;
M = diag(a*ones(1,JM)) + diag(b*ones(1,JM-1),1) + diag(b*ones(1,JM-1)
    ,-1);
A = (1/h^2)*M;
[t1,u1] = em(@(u) A*u,0,0.6,0.001,in);
[t2,u2] = em(@(u) A*u,0,0.6,0.005,in);
[t3,u3] = em(@(u) A*u,0,0.6,0.0052,in);
[t4,u4] = em(@(u) A*u,0,0.6,0.0056,in);
function [t, u] = em(f,a,b,k,u0)
t = a:k:b;
u(:,1) = u0;
N = length(t);
for i = 1:N
    u(:,i + 1) = u(:,i) + f(u(:,i))*k;
end
end
```

Then we extract the path at 0.5 for each $k-$value and plot.

```
grid on

plot(t1,u1(5,1:end-1))
plot(t2,u2(5,1:end-1))
plot(t3,u3(5,1:end-1))
plot(t4,u4(5,1:end-1))
```

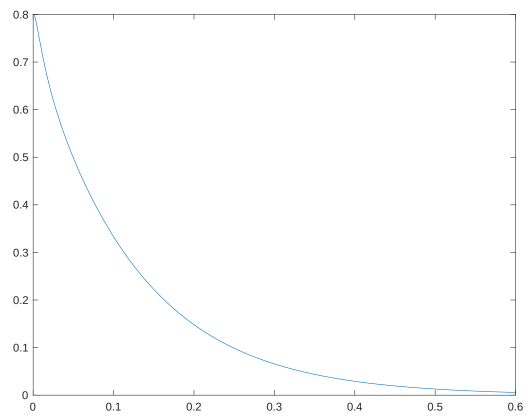

Figure 1: $k = 0.001$

Since $h = 0.1$ we have the requirement $k \leq h^2/2 = 0.005$. For the first two k-values, we see stability and a decay to zero and for the second two we see instability. The last value is also near-zero for most of its path before oscillating.

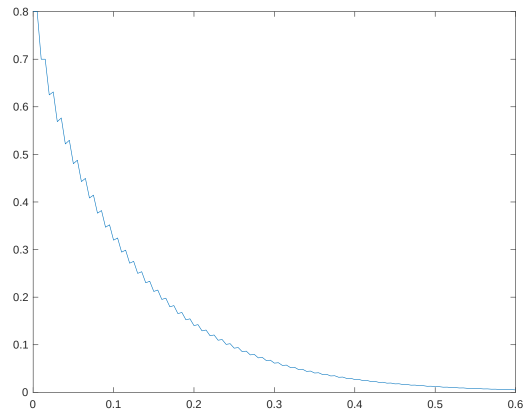

Figure 2: $k = 0.005$

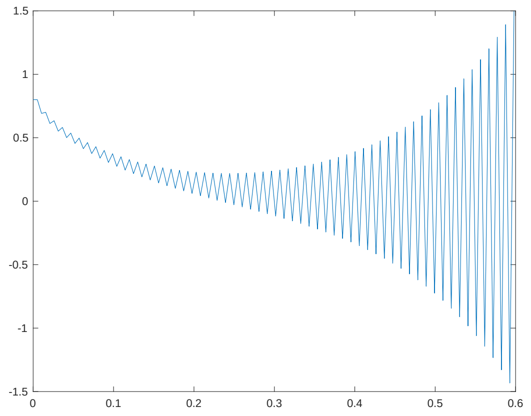

Figure 3: $k = 0.0052$

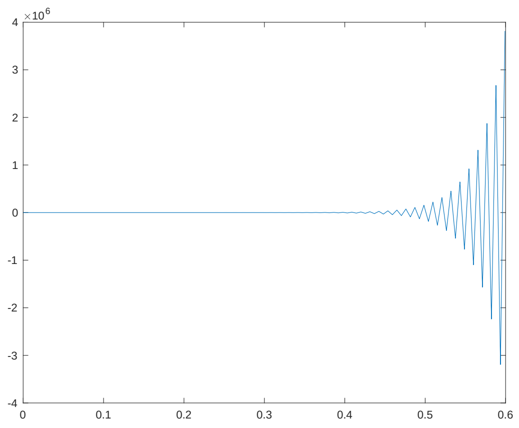

Figure 4: $k = 0.0056$

Question 2:

a) To approximate Burger's equation with second order space first order time finite differences, we use the following.

$$\frac{\partial u}{\partial t} = \frac{u(x_j, t_{n+1}) - u(x_j, t_n)}{k} + \mathcal{O}(k)$$

$$\frac{\partial u}{\partial x} = \frac{u(x_{j+1}, t_n) - u(x_{j-1}, t_n)}{2h} + \mathcal{O}(h^2)$$

$$\frac{\partial^2 u}{\partial x^2} = \frac{u(x_{j+1}, t_n) - 2u(x_j, t_n) + u(x_{j-1}, t_n)}{h^2} + \mathcal{O}(h^2)$$

Substitution gives us

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2}$$

$\approx$

$$\frac{u(x_j, t_{n+1}) - u(x_j, t_n)}{k} + u(x_j, t_n)\frac{u(x_{j+1}, t_n) - u(x_{j-1}, t_n)}{2h} = \nu\frac{u(x_{j+1}, t_n) - 2u(x_j, t_n) + u(x_{j-1}, t_n)}{h^2}$$

$\equiv$

$$\frac{u_j^{n+1} - u_j^n}{k} + \frac{u_j^n}{2h}[u_{j+1}^n - u_{j-1}^n] = \frac{\nu}{h^2}[u_{j+1}^n - 2u_j^n + u_{j-1}^n]$$

$\equiv$

$$u_j^{n+1} = u_{j+1}^n\left[\frac{k\nu}{h^2}\right] + u_j^n\left[1 - \frac{2k\nu}{h^2}\right] + u_{j-1}^n\left[\frac{k\nu}{h^2}\right] - \frac{ku_j^n}{2h}[u_{j+1}^n - u_{j-1}^n]$$

$\equiv$

$$u_j^{n+1} = u_{j+1}^n\,[r] + u_j^n\,[1 - 2r] + u_{j-1}^n\,[r] - \frac{ku_j^n}{2h}[u_{j+1}^n - u_{j-1}^n]$$

b) To implement this method, we have the following code adapted from the textbook. We can see the solution at each time from the entire plots.

```
h = 0.02;
nu = 0.1;
T = 1;
x = 0:h:1;
n = length(x)-1;
k = 0.001;
t = 0:k:T;
m = length(t)-1;
r = nu*k/h^2;
U = zeros(length(x),length(t));
U(:,1) = sin(pi*x);
A1 = (1-2*r)*diag(ones(1,n-1))+r*(diag(ones(1,n-2),1)+diag(ones(1,n-2)
    ,-1));
for i = 1 : (length(t)-1)
```

```
    U(2:n,i+1) = A1*U(2:n,i)-(k/(2*h))*U(2:n,i).*(U(3:n+1,i)-U(1:n-1,i)
        );
end
[T,X] = meshgrid(t,x);
pl = surf(X,T,U);
zlabel('u')
xlabel('x')
ylabel('t')
colorbar
set(pl,'LineStyle','none')
```

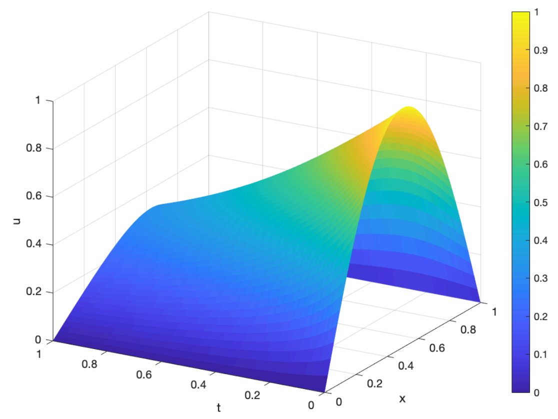

Figure 5: $\nu = 0.1$

From the plots, we see that $\nu = 0.1$ follows a half hyperbolic paraboloid where $t = 0$ is the largest curve and as $t \to 1$ the peaks shrink towards zero. For $\nu = 0.01$, we have similar behaviour until $t = 0.4$ where the curve begins to blow up at the boundary. When $\nu = 0.001$, we see some of the same behaviour in that until $t = 0.3$ there is a very small parabolic peak and then after we have oscillations and a blow up at the boundary. These results occur when $r = 0.25, 0.025, 0.0025$.

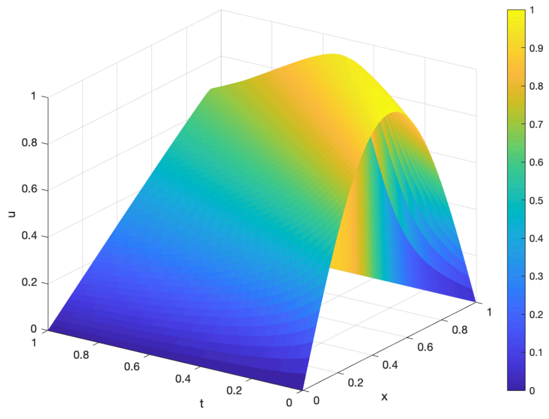

Figure 6: $\nu = 0.01$

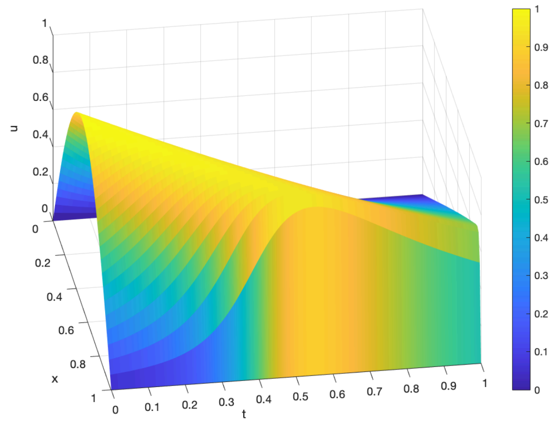

Figure 7: $\nu = 0.01$ view 2

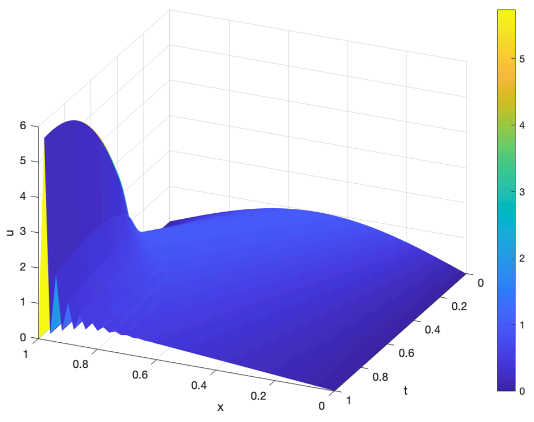

Figure 8: $\nu = 0.001$

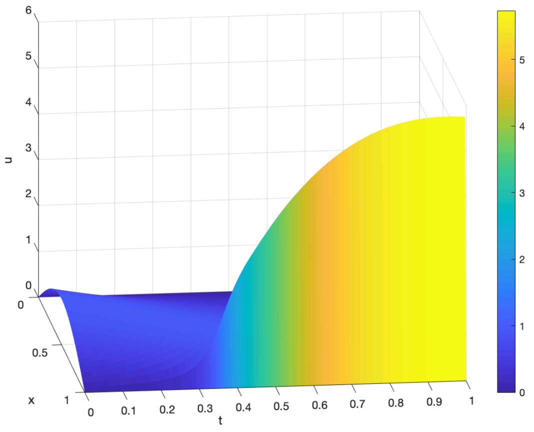

Figure 9: $\nu = 0.001$ view 2

Question 3:

a) Here is a function that solves the 1D wave equation with the given conditions. It takes $f(x)$ and the step-sizes.

```
function [X,T,U] = wave(f, h, k)
x = 0:h:1;
nX = length(x)-2;
x1 = x(2:nX+1);
r = k/h;
t = 0:k:4;
nT = length(t);
U = zeros(nX,nT);
U(:,1) = f(x1');
d1 = diag(ones(1,nX));
d2 = diag(ones(1,nX-1),1)+diag(ones(1,nX-1),-1);
A = 2*(1-r^2)*d1+r^2*d2;
U(:,2) = 0.5*A*U(:,1);
for i = 3 : nT
        U(:,i) = A*U(:,i-1)-U(:,i-2);
end
U = [zeros(1,length(t));U;zeros(1,length(t))];
[X,T] = meshgrid(x,t(1:(nT-1-12*2)/2));
figure(1); surf(X,T,U(:,1:(nT-1-12*2)/2)');
zlabel('u')
xlabel('x')
ylabel('t')
colorbar
end
```

b) We apply the function to $f(x) = \sin(2\pi x)$.

```
wave(@(x) sin(2*pi*x), 0.01, 0.005)
```

The exact solution can be obtained by D'Alembert's formula

$$u(x,t) = \frac{1}{2}[\sin(2\pi(x-t)) + \sin(2\pi(x+t))] + 0$$

We can plot the difference in the exact and numerical solution by using the following.

```
figure(2);surf(X,T,abs(g(X,T) - U(:,1:(nT-1-12*2)/2)'));
zlabel('u')
xlabel('x')
ylabel('t')
colorbar

function exact = g(x,t)
exact = 0.5*(sin(2*pi*(x-t)) + sin(2*pi*(x+t)));
end
```

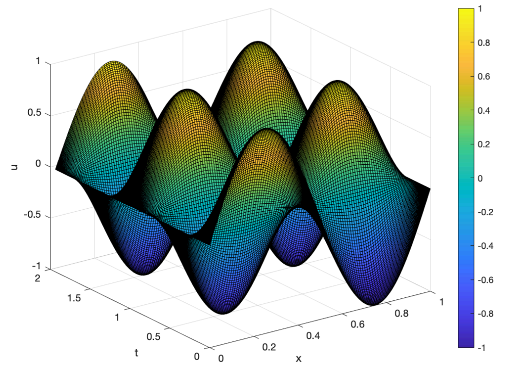

Figure 10: Numerical Solution

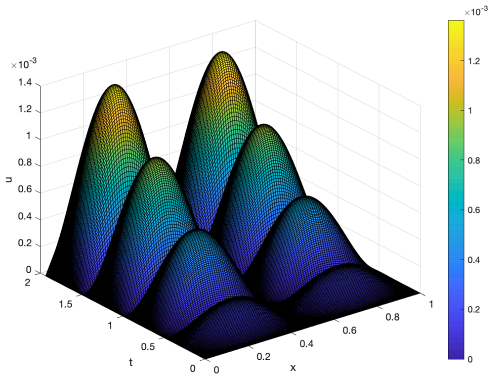

Figure 11: Difference

From the difference plot, we see that the error is accumulating as time increases to $t = 2$ and the difference has magnitude 10^{-3}.

c) For this function we have instability.

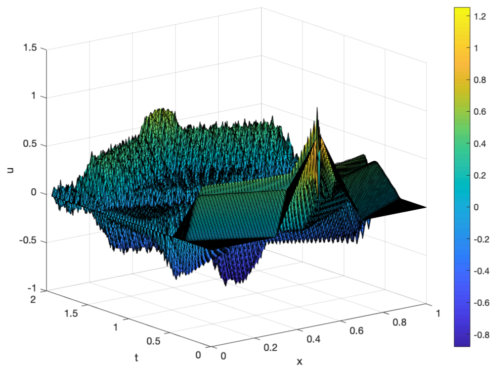

Figure 12: Difference

Question 4:

a) Here is the code adapted from 9.6 in the textbook to implement SOR at the optimal parameter for out specific problem.

```
n = 50;
h = 1/n;
x = 0:h:1;
y = 0:h:1;
omega = 2/(1+sqrt(1-cos(pi/n)^2));
    u = zeros(n+1,n+1);
    l = 0;
    nor = 1;
    while( (nor>10^(-6)) && (l<10000))
        l = l+1;
        uu = u;
        for k = 2:n-1
            for m = 2:n-1
                ut = u(k-1,m)+u(k+1,m)+u(k,m-1)+u(k,m+1) ...
                            -h^2*sin(x(k))^2-h^2*sin(y(k))^2;
                u(k,m) = (1-omega)*u(k,m)+0.25*omega*ut;
            end
        end
        nor = max(max(abs(u-uu)));
    end
    fprintf('Number of iterations for n = %d and omega = %f is %d\n',n,
        omega,l);
[X,Y] = meshgrid(x,y);
mesh(X,Y,u);
```

We can see the optimal parameter and iterations required increasing as the grid size increases. In this cases, it takes twice as many iterations as the grid size at the optimal relaxation.

```
Number of iterations for n = 25 and omega = 1.777251 is 50
Number of iterations for n = 50 and omega = 1.881838 is 100
Number of iterations for n = 100 and omega = 1.939092 is 199
Number of iterations for n = 200 and omega = 1.969071 is 399
```

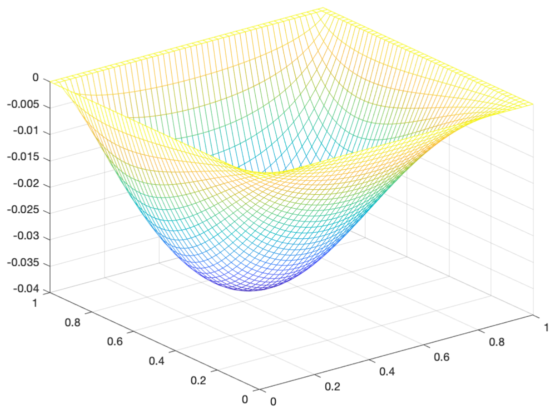

Figure 13: Solution $n = 50$