Assignment V

M693B
DR. BLOMGREN
APRIL 19, 2018

MATTEO POLIMENO

1 Problem 1.3.4 from Strikverda

In this problem we have to numerically solve the PDE given by

$$u_t + u_x = -sin^2(u),$$

whose analytical solution is given by

$$u(t,x) = tan^{-1}\left(\frac{tan[u_0(x-t)]}{1+t\tan[u_0(x-t)]}\right).$$

Initial data is given to be

$$u(0,x) = \begin{cases} \cos^2 \pi x & \text{if } |x| \leq \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}$$

The x-interval is given by $[-1,3]$ while for the time t interval it holds $t \in [0, 2.4]$.

The PDE will be solved using the Leapfrog scheme for the value of $\lambda = 0.5$ and for $h = 1/10,\ 1/20,\ 1/40$. Boundary condition at $x = -1$ is $u(t,-1) = 0$.

1.1 Function

Here is the function called

```
function J = G(x)

if abs(x) <= .5
    J = cos(pi*x)^2;
else
    J = 0;
end
end
```

and here is the numerical scheme

```
clear all


lambda = .5;
h = 1/10; %change value of h when needed
x = -1:h:3;
k = lambda*h;
p = length(x);
t = 0:k:2.4;
q = length(t);


for m = 1:p
```

```
    u(1,m) = G(x(m));
end


for i = 1:2 %FTCS scheme
    u(i,1) = 0;
    for j = 1:p-2
        u(i+1,j+1) = -lambda*((u(i,j+2) - u(i,j)))/2 + u(i,j+1);
    end
end


for i = 1:q-1 %leap frog

    for j = 2:p-2
        u(i+2,j+1) = -lambda*(u(i+1,j+2) - u(i+1,j)) + u(i,j+1) - 2*k*(
            sin(u(i,j+1)));
        u(i+1,p) = u(i,p-1);
    end
end

for i = 1:q
    v(i,1) = 0;
end


for i = 1:q
    for j = 1:p
        v(i,j) = atan((tan(G((x(j) -  t(i))))))/(1+t(i)*(tan(G((x(j) -
            t(i)))))));
    end
end

for i = 1:q
    plot(x,u(i,:),'b-o',x,v(i,:),'k-*');
    ylim([-0.1,1.5])
    xlim([-1,3])
    title(['Wave Function: a = 1, h = ' num2str(h) ',\lambda = '
        num2str(lambda) ', Time = ' num2str(t(i))])
    xlabel('x')
    ylabel('u')
    grid on
    M(i) = getframe;
end

supnorm = max(abs(v(q,:)-u(q,:)))
```

```
57
58 L2norm = norm(v(q,:)-u(q,:))
```

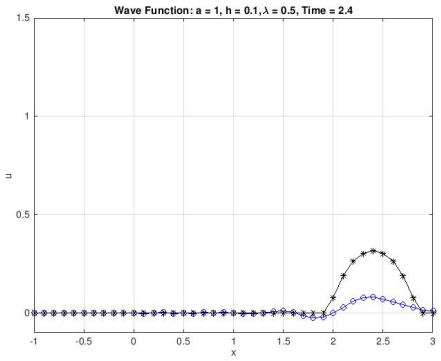

Figure 1: Analytical solution vs. scheme for h=1/10

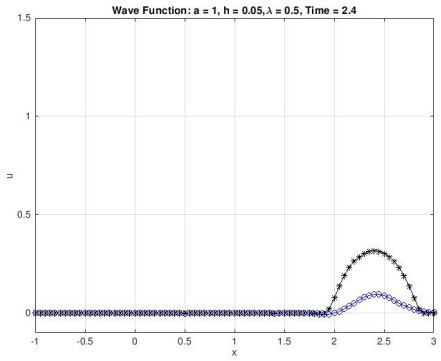

Figure 2: Analytical solution vs. scheme for h=1/20

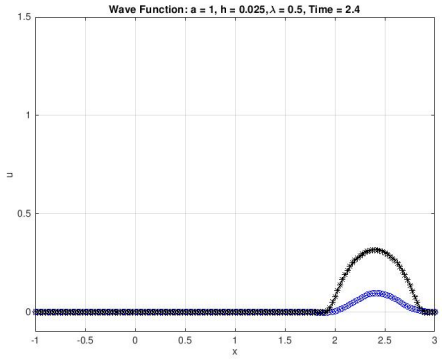

Figure 3: Analytical solution vs. scheme for h=1/40

The scheme was run for the listed values of h and we plotted the results

From the plots above we can see that, while the scheme seems much smoother for $h = 1/40$, the overall accuracy does not seem to terribly improve by reducing h. Although, it is worth noting that we do not lose information at either end of the peak for $h = 1/20$ and $1/40$, while we do lose it for $h = 1/10$. However, the amount of information lost at the peak is virtually the same for the scheme run for the entirety of the time interval. We make a table to illustrate the supremum norm and the L2 norm for the different cases and therefore have a quantitative picture of the error and accuracy of the scheme.

L2 Norm and Supremum Norm		
h value	Supremum Norm	L2 Norm
1/10	0.2366	0.5514
1/20	0.2244	0.7661
1/40	0.2221	1.0732

2 Problem 3.1.2 from Strikwerda

In this problem, we solve

$$u_t + u_x = 0$$

in the interval $-1 \leq x \leq 1$ for $t \in [0, 1.2]$ with $u(0, x) = \sin(2\pi x)$ and $u(t, 1) = u(t, -1)$. The values of h will be chosen to be $1/10$, $1/20$, $1/40$ and $1/80$.

Here is the function called

```
function J = G8(x)
J = sin(2*pi*x);
end
```

2.1 Part a

In this part we will numerically solve the equation using the Forward-time backward space scheme with $\lambda = 0.8$,

Here is the numerical scheme implemented

```
clear all

lambda = .8;
h = 1/10; %change the value of h as needed
xd = -1:h:1;
k = lambda*h;
p = length(xd);
td = 0:k:1.2;
q = length(td);


for i = 1:p
    u(1,i) = G(xd(i)); %set initial data
```

```
14 end
15
16
17 for i = 1:q %forward-time backward-space scheme
18     for j = 1:p-1
19         u(i,1) = u(i,p); %set boundary data
20         u(i+1,j+1) = (1-lambda)*u(i,j+1) + lambda*u(i,j);
21     end
22 end
23
24 u(:,p) = u(:,1);
25
26 for i = 1:p %boundary data for exact solution
27     v(i,1) = 0;
28 end
29
30
31 for i = 1:q %calculates exact solution from initial data
32     for j = 1:p
33         v(i,j) = G((xd(j) -  td(i)));
34     end
35 end
36
37 for i = 1:q
38     plot(xd,u(i,:),'b-o',xd,v(i,:),'k-*');
39     ylim([-1,1])
40     xlim([-1,1])
41     title(['Wave Function: a = 1, h = ' num2str(h) ',\lambda = '
          num2str(lambda) ', Time = ' num2str(td(i))])
42     xlabel('x')
43     ylabel('u')
44     grid on
45    M(i) = getframe;
46 end
47
48 supnorm = max(abs(v(q,:)-u(q,:)))
49
50 L2norm = norm(v(q,:)-u(q,:))
51
52
53 for i = 1:q %calculates Sum of Squared Errors
54      e(i,:) = abs((v(i,:)-u(i,:)));
55      err(i) = sum(e(i,:)).^2;
56 end
57
58 figure()
```

```
59 plot(td,err,'b-') %plots SSEs at all calculated times
60 title(['SSE vs time: a = 1, h = ' num2str(h) ',\lambda = ' num2str(
       lambda)])
61 xlabel('Time')
62 ylabel('SSE')
63 grid on
```

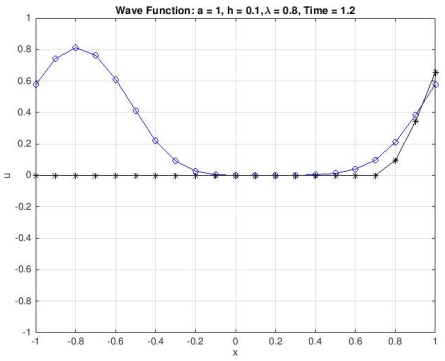

Figure 4: Analytical solution vs. scheme for h=1/10

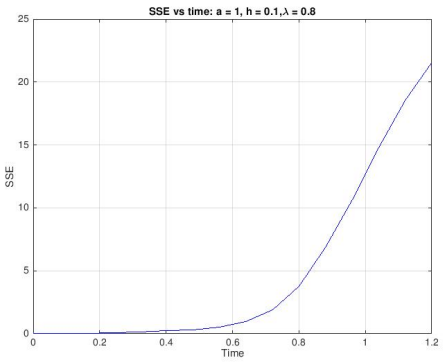

Figure 5: Sum of Squared Errors for h=1/10

From the plots above we can see how the scheme does not do a good job in reproducing the dynamics of the solution for $h = 1/10$, as it completely fails to do so after $t = 0.5$, with the SSE the increases virtually exponentially from that point on, although it seems to go toward an asymptote. While the mapping at the right boundary is at least reasonable, the scheme fails completely at the left boundary. The scheme is useless.

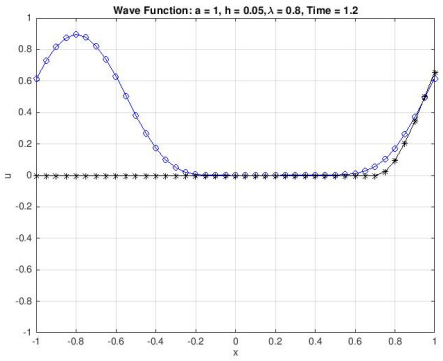

Figure 6: Analytical solution vs. scheme for h=1/20

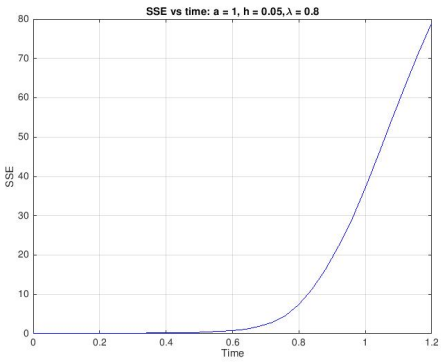

Figure 7: Sum of Squared Errors for h=1/20

From the plots above we can see how the scheme does not do a good job in reproducing the dynamics of the solution for $h = 1/20$, as it completely fails to do so after $t = 0.5$, with the SSE the increases virtually exponentially from that point on. While the mapping at the right boundary is at least reasonable, the scheme fails completely at the left boundary. So we see that reducing the value of h by one half still does not cut it and the scheme is useless in this case as well.

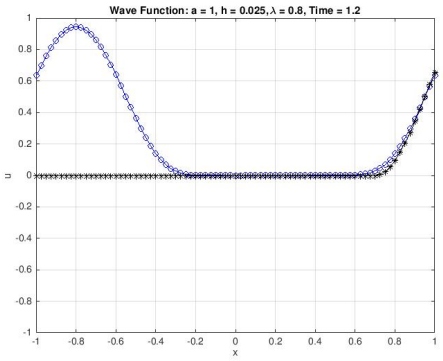

Figure 8: Analytical solution vs. scheme for h=1/40

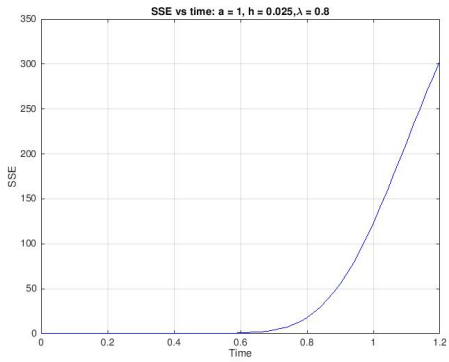

Figure 9: Sum of Squared Errors for h=1/40

From the plots above it seems that reducing the value of h to $1/40$, increases the stability of the scheme, but only for very little time, as it basically blows up after $t = 0.7$. While the mapping at the right boundary has improved, the scheme fails completely at the left boundary. The scheme is useless in this case as well.

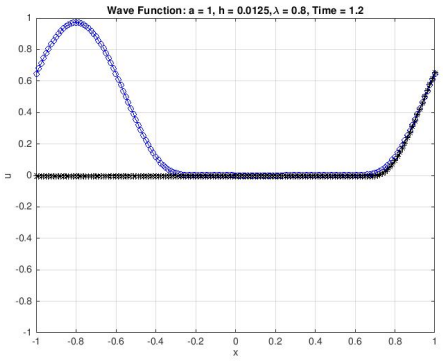

Figure 10: Analytical solution vs. scheme for h=1/80

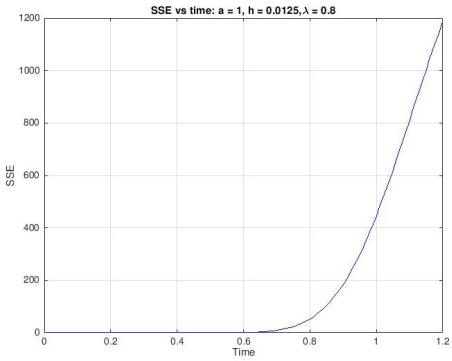

Figure 11: Sum of Squared Errors for h=1/80

From the plots above it seems that reducing the value of h to $1/80$, increases the accuracy of the scheme, it still basically blows up after $t = 0.7$. While the mapping at the right boundary has improved sensibly, the scheme fails completely at the left boundary. The scheme is once again useless.

We then calculate the L2 norm and the Supremum Norm for each case and display our results

L2 Norm and Supremum Norm		
h value	L2 Norm	Supremum Norm
1/10	1.6591	0.8109
1/20	2.4607	0.8945
1/40	3.5841	0.9442
1/80	5.1531	0.9713

2.2 Part b

In this part we will numerically solve the equation using the Lax-Wendroff scheme with $\lambda = 0.8$,
Here is the numerical scheme implemented

```
clear all

lambda = .8;
h = 1/10; %change the value of h as needed
x = -1:h:1;
k = lambda*h;
p = length(x);
t = 0:k:1.2;
q = length(t);


for k = 1:p
```

```
    u(1,k) = G8(x(k)); %set initial data
end


for i = 1:q-1 %lax-wendroff scheme
    for j = 1:p-2
        u(i,1) = G8((x(j) -  t(i))); %set boundary data1
        u(i+1,j+1) = u(i,j+1) - (lambda/2)*(u(i,j+2)-u(i,j)) + ((lambda
            ^2)/2)*(u(i,j+2) - 2*u(i,j+1) + u(i,j));
    end
end
u(q,1) = G8((x(1) -  t(q)));
u(:,p) = u(:,1);

for i = 1:p %boundary data for exact solution
    v(i,1) = 0;
end


for i = 1:q %calculates exact solution from initial data
    for j = 1:p
        v(i,j) = G8((x(j) -  t(i)));
    end
end

for i = 1:q
    plot(x,u(i,:),'b-o',x,v(i,:),'k-*');
    ylim([-1,1])
    xlim([-1,1])
    title(['Wave Function: a = 1, h = ' num2str(h) ',\lambda = '
        num2str(lambda) ', Time = ' num2str(t(i))])
    xlabel('x')
    ylabel('u')
    grid on
   M(i) = getframe;
end


for i = 1:q %calculates Sum of Squared Errors
    e(i,:) = abs((v(i,:)-u(i,:)));
    err(i) = sum(e(i,:)).^2;
end

supnorm = max(abs(v(q,:)-u(q,:)))

L2norm = norm(v(q,:)-u(q,:))
```

```
57
58 figure()
59 plot(td,err,'b-') %plots SSEs at all calculated times
60 title(['SSE vs time: a = 1, h = ' num2str(h) ',\lambda = ' num2str(
       lambda)])
61 xlabel('Time')
62 ylabel('SSE')
63 grid on
```

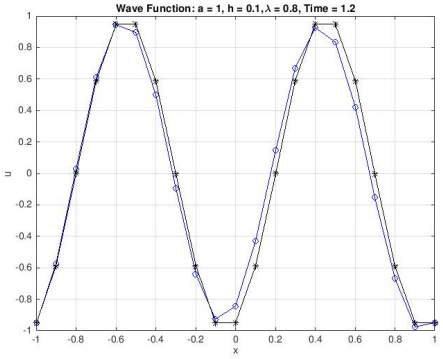

Figure 12: Analytical solution vs. scheme for h=1/10

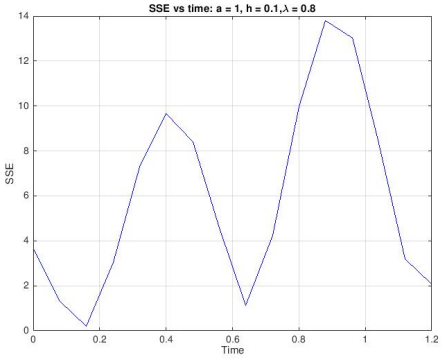

Figure 13: Sum of Squared Errors for h=1/10

From the plots above, it can readily be seen that, for $h = 1/10$ the scheme is unstable. While it does follow the dynamics of the solution, it still fails to reproduce it accurately and this can be inferred by the plot of the Sum of Squared Errors as well. The error literally goes sharply up and down as time goes on. That being said, it can readily be seen that the Lax-Wendroff scheme does a much better job than the FTCS analyzed in part a, even for "big" values of h.

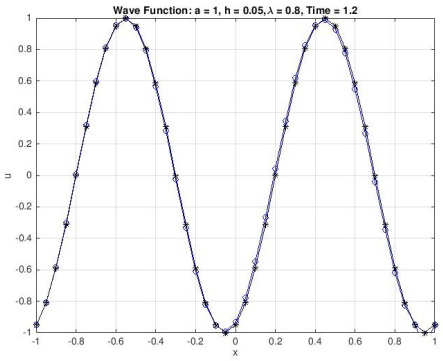

Figure 14: Analytical solution vs. scheme for h=1/20

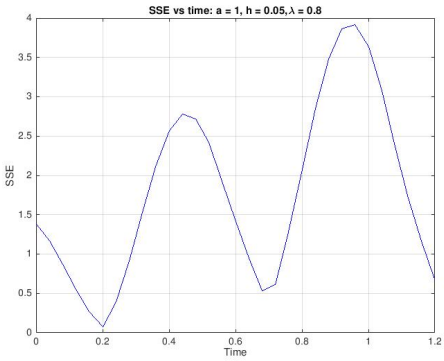

Figure 15: Sum of Squared Errors for h=1/20

From the plots above, it can readily be seen that, for $h = 1/20$ that the scheme has improved its accuracy sharply. It reproduces the dynamics of the solution over time, even though some error still remains.

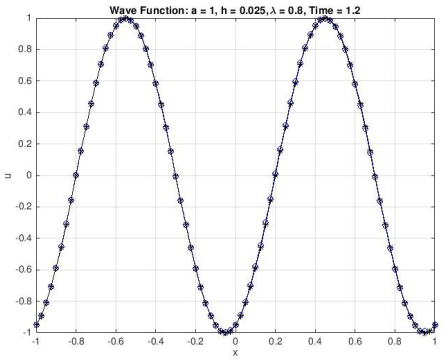

Figure 16: Analytical solution vs. scheme for h=1/40

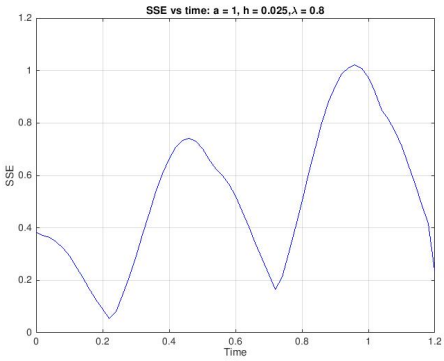

Figure 17: Sum of Squared Errors for h=1/40

From the plots above it seems that reducing the value of h to $1/40$ basically insures that the scheme will properly describe the dynamics of the solution, as it is almost indistinguishable from our analytical results.

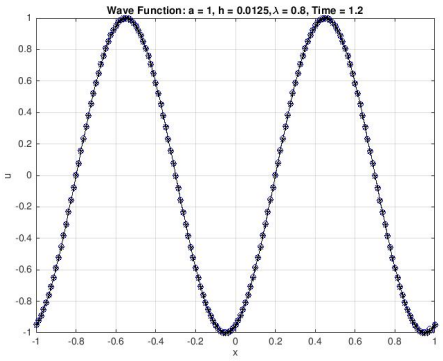

Figure 18: Analytical solution vs. scheme for h=1/80

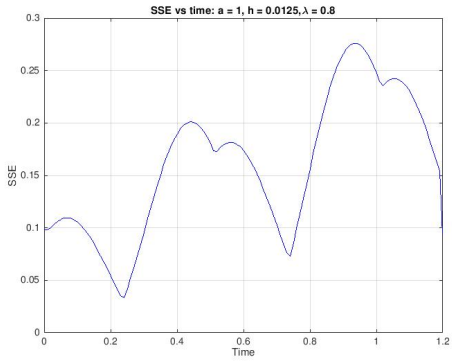

Figure 19: Sum of Squared Errors for h=1/80

From the plots above it can readily be seen that the scheme is stable for $h = 1/80$.
We then calculate the L2 norm and the Supremum Norm for each case and display our results

L2 Norm and Supremum Norm		
h value	L2 Norm	Supremum Norm
1/10	0.4018	0.1665
1/20	0.1678	0.0771
1/40	0.1100	0.0955
1/80	0.1041	0.1017