CFD Assignment

Shantanu Malik

(15AE30024)

Introduction

Three different numerical schemes were tested for a linear convection equation with unit positive speed and a sinusoidal initial condition.

Equation $\frac{\partial u}{\partial t}+\frac{\partial u}{\partial x}=0$ was solved for $u(x,t)$ with $u_0=\sin(x)$ where $x\in(0,2\pi)$

Thereafter, their stability characteristics were studied through numerically simulation using MATLAB and the results were compared with theoretical predictions. The code used has been included as appendix.

Scheme 1

Spacial scheme: 2nd order central shceme with periodic BC

Temoral scheme: Forward Euler scheme

Expected stability: Unstable for all *dt*

Simulation time: 4π

Spacial grid size: $2\pi/100$

As shown in class using the Von Newmann analysis, 2nd order central scheme is always unstable for a linear convection equation. The same was verified in the result i obtained. Figure 1 shows the growth in error and the maximum magnitude of the numerical solution as it progressed in time. As evident, the magnitude exponentially grows with time even for a very small time step of *dx/10*. Thus, implying that the scheme is unconditionally unstable.

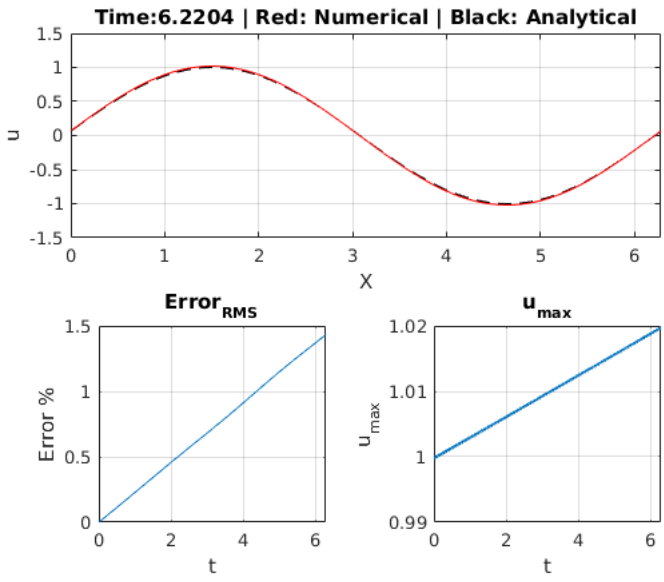


Figure 1: 2nd order central scheme with periodic BC.

Scheme 2

Spacial scheme: 2nd order central interior shceme with 1st order boundary scheme.

Temoral scheme: Forward Euler scheme and RK-4 Scheme

Expected stability: Euler - Unstable for all *dt*, RK-4 stable for $dt < 2.83\,dx$

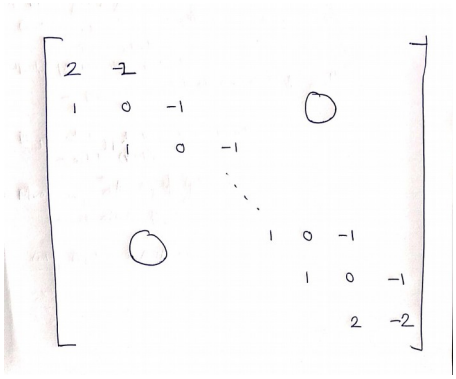
Simulation time: 20π

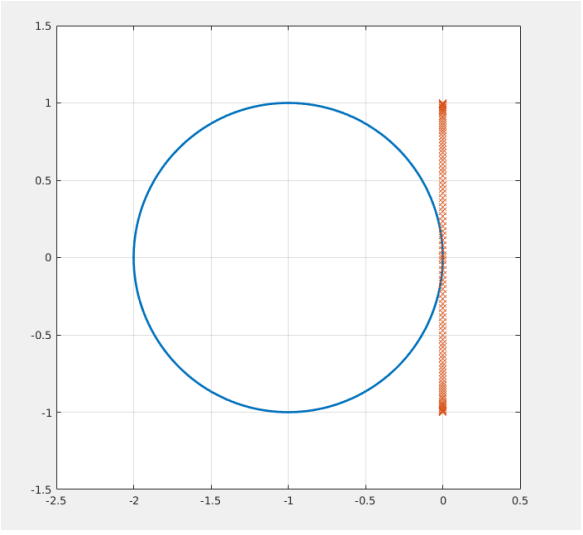
Spacial grid size: $2\pi/100$

For an Euler stability scheme, it is shown using matrix stability analysis, that 2nd order central scheme is unconditionally unstable for a linear convection equation. This is because all the Eigen values of the matrix A are imaginary, and lie outside the stability region of Euler scheme (figure 2a). The same was verified in the result obtained in figure 2b for a very small timestep of *dx/100*. It was however observed that the error did not grow exponentially. Instead, it periodically settled to a constant value and then continued to grow. The reason for this is probably that the Eigen values are nearly purely imaginary, with a very small real part.

$$\frac{d\boldsymbol{u}}{dt} = A\,\boldsymbol{u}$$

Where, matrix `A is`





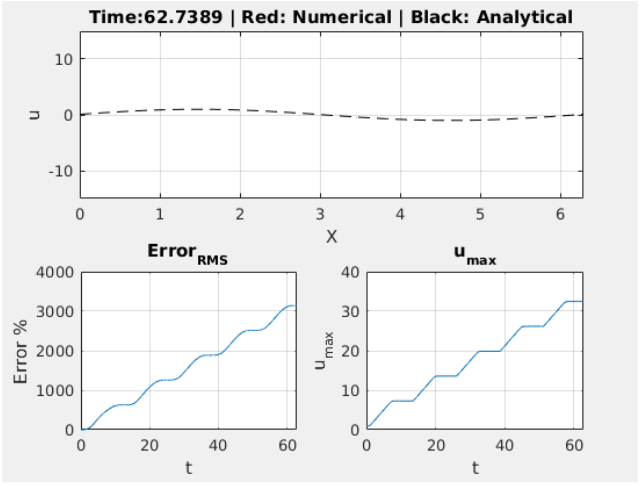


Figure 2: (a) Stability region and Eigen values of A, (b) error and amplitude growth

In the case of RK-4 temporal scheme, the solution should be stable for $dt < \frac{2.83}{\lambda_{max}}$.

Hence, $dt < 2.83\,dx$ for stability. However, the numerical results were not as expected. For $dt > 2.83\,dx$, the magnitude blew to infinity (figure 3a). On the other hand, for smaller timesteps, the maximum magnitude still grew over time, but not monotonically (figure 3b). Instead, as in the case of Euler scheme, it increased in periodic steps. The same was observed for all $dt < 2.83\,dx$. The reason for this unknown. The only thing different from Euler method is that the critical *dt* changed to around *2.83 dx* from around *0.01 dx*.

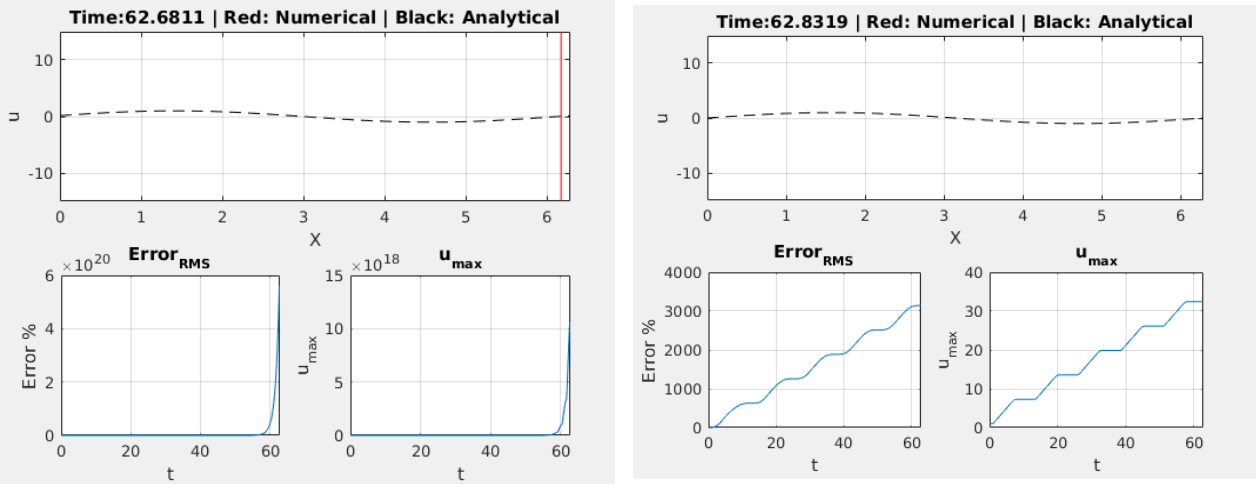


Figure 3: (a) dt = 2.9 dx, (b) dt = 2.8 dx

Scheme 3

Spacial scheme: 1st order backward difference

Temoral scheme: Forward Euler scheme

Expected stability: Stable for *dt < dx*

Simulation time: 10π

Spacial grid size: $2\pi/100$

As per the modified wavenumber analysis done in class, the solution should be stable for *dt <= dx*. The same was observed in the numerical simulation, as shown in figures 4 a-c.

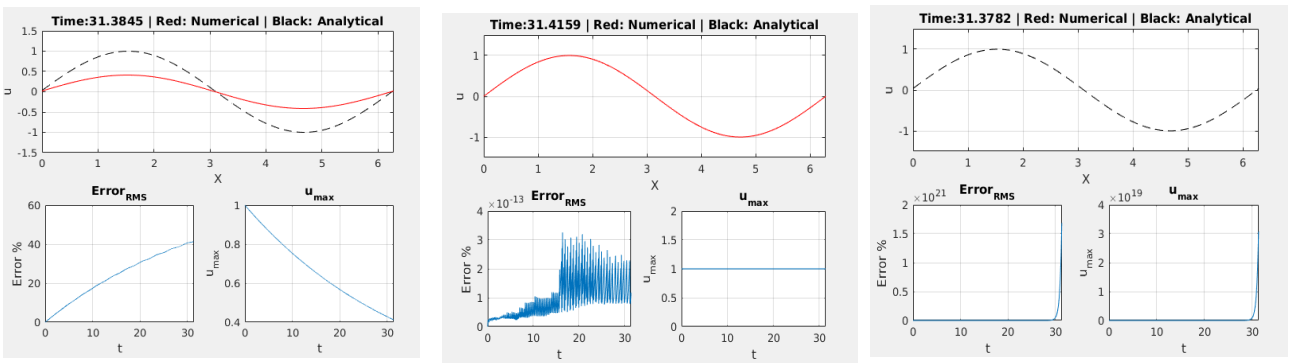


Figure 4: (a) dt = dx/10; (b) dt = dx; (c) dt = 1.1 dx

Appendix

(MATLAB Code)

Stability Finite-differnece Schemes

This script demonstrates the stability of 3 finite difference schemes to solve a convection equation for a sinusoidal function.

Contents

Solver 1: 2nd order central scheme in space with periodic BCs

This function solves the convection equation with c = 1 for a sinusoidal function using finite difference schemes. Spacial scheme: 2nd order central with periodic boundary conditions; Temporal scheme: Forward Euler scheme; Stability: Unstable for all dt.

Solver 1: Initialization

```
T = 2*pi;   % simulation time
dx = 2*pi/100;  % spacial grid size
dt = dx/10; % temporal grid size
x = 0:dx:2*pi;  % spacial domain
t = 0:dt:T; % tempral domain

u = zeros(length(t),length(x));
error = zeros(1,length(t));
u_max = error;
u(1,:) = sin(x);    % initial condition
u_max(1) = max(u(1,:));
```

Solver 1: Solver

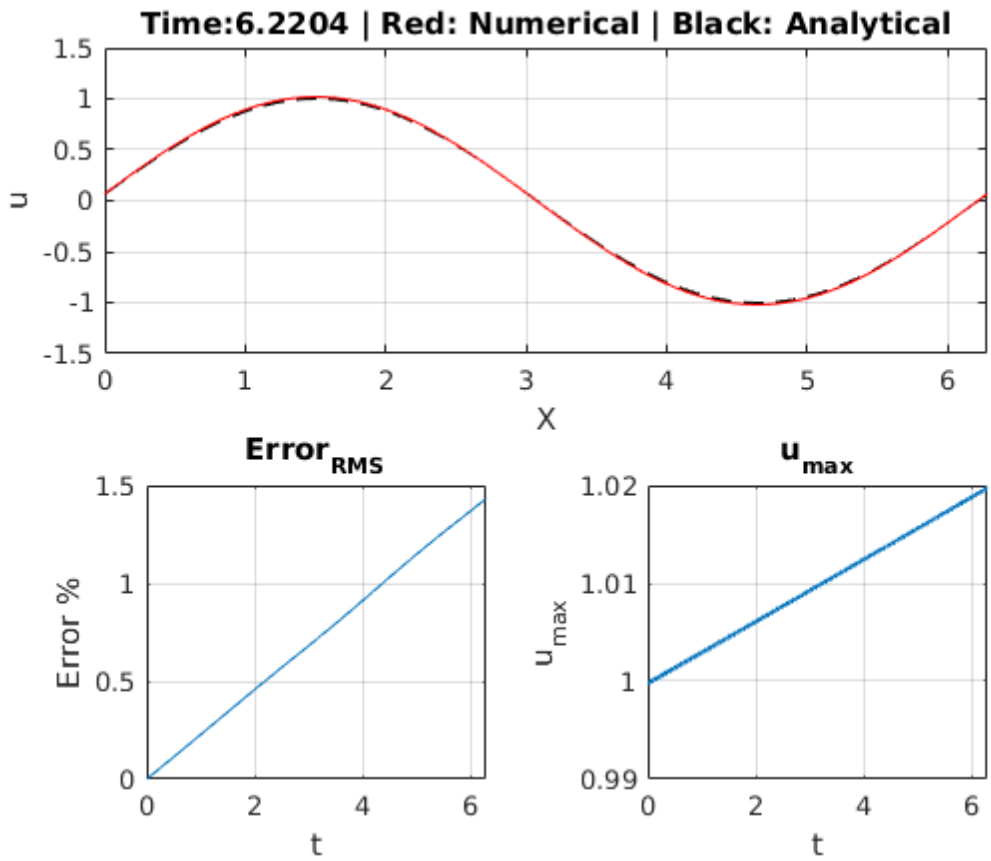
```
for n = 2:size(u,1)
    for j = 1:size(u,2)
        if j == 1
            u_1 = u(n-1,end - 1); u_2 = u(n-1,j+1);
        elseif j == size(u,2)
            u_1 = u(n-1,j-1); u_2 = u(n-1,2);
        else
            u_1 = u(n-1,j-1); u_2 = u(n-1,j+1);
        end
        u_x = (u_2 - u_1)/(2*dx);
        u_t = - u_x;
        u(n,j) = u(n-1,j) + dt*u_t;
    end
    % Error and u_max calculation
    u_max(n) = max(abs(u(n,:)));
```

```
    error(n) = (rms(u(n,:) - sin(x-t(n))))*100/max(sin(x-t(n)));
end
```

Solver 1: Plotting the results

```
% Solution
figure
tic
for n = 1:max(1,floor(0.1/dt)):size(u,1)
    pause((n-1)*dt - toc)
    subplot(2,2,[1,2])
    plot(x,sin(x-t(n)),'--k',x,u(n,:),'r');grid on
    xlim([0,2*pi]);ylim([-1.5,1.5])
    title(['Time:',num2str((n-1)*dt),' | Red: Numerical | Black: Analytical']);
    xlabel('X');ylabel('u')
end

% Error
subplot(2,2,3)
plot(t,error)
grid on;xlim([0,T])
title('Error_{RMS}');xlabel('t');ylabel('Error %')
subplot(2,2,4)
plot(t,u_max)
grid on;xlim([0,T])
title('u_{max}');xlabel('t');ylabel('u_{max}')
```



Solver 2: 2nd order central scheme (interior) with 1st order boundary schemes

This function solves the convection equation with c = 1 for a sinusoidal function using finite difference schemes. Spacial scheme: 2nd order central (interior) with 1st order boundary schemes; Temporal scheme: Forward Euler scheme; Stability: Unstable for all dt.

Solver 2: Initialization

```
T = 4*pi;   % simulation time
dx = 2*pi/100;  % spacial grid size
dt = dx/10; % temporal grid size
x = 0:dx:2*pi;  % spacial domain
t = 0:dt:T; % tempral domain

u = zeros(length(t),length(x));
error = zeros(1,length(t));
u_max = error;
u(1,:) = sin(x);    % initial condition
u_max(1) = max(u(1,:));
```

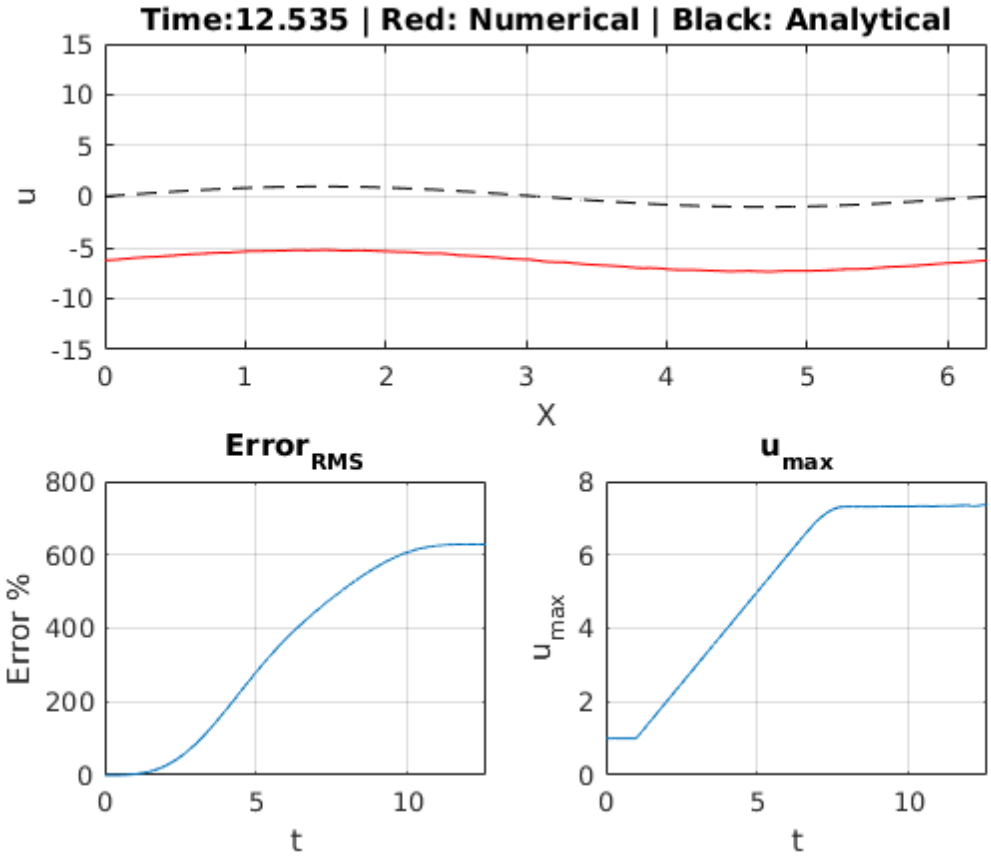
Solver 2: Solver

```
for n = 2:size(u,1)
    for j = 1:size(u,2)
        if j == 1
            u_x = (u(n-1,j+1) - u(n-1,j))/(dx);
        elseif j == size(u,2)
            u_x = (u(n-1,j) - u(n-1,j-1))/(dx);
        else
            u_x = (u(n-1,j+1) - u(n-1,j-1))/(2*dx);
        end
        u_t = - u_x;
        u(n,j) = u(n-1,j) + dt*u_t;
    end
    % Error and u_max calculation
    u_max(n) = max(abs(u(n,:)));
    error(n) = (rms(u(n,:) - sin(x-t(n))))*100/max(sin(x-t(n)));
end
```

Solver 2: Plotting the results

```
% Solution
figure
tic
for n = 1:max(1,floor(0.1/dt)):size(u,1)
    pause((n-1)*dt - toc)
    subplot(2,2,[1,2])
    plot(x,sin(x-t(n)),'--k',x,u(n,:),'r');grid on
    xlim([0,2*pi]);ylim([-15,15])
    title(['Time:',num2str((n-1)*dt),' | Red: Numerical | Black: Analytical']);
    xlabel('X');ylabel('u')
end

% Error
subplot(2,2,3)
plot(t,error)
grid on;xlim([0,T])
title('Error_{RMS}');xlabel('t');ylabel('Error %')
subplot(2,2,4)
plot(t,u_max)
grid on;xlim([0,T])
title('u_{max}');xlabel('t');ylabel('u_{max}')
```



Solver 3: 1st order backward difference scheme in space

This function solves the convection equation with c = 1 for a sinusoidal function using finite difference schemes. Spacial scheme: 1st order backward difference; Temporal scheme: Forward Euler scheme; Stability: Stable for all dt <= dx/c.

Solver 3: Initialization

```
T = 10*pi;   % simulation time
dx = 2*pi/100;  % spacial grid size
dt = dx/100; % temporal grid size
x = 0:dx:2*pi;  % spacial domain
t = 0:dt:T; % tempral domain

u = zeros(length(t),length(x));
error = zeros(1,length(t));
u_max = error;
u(1,:) = sin(x);    % initial condition
u_max(1) = max(u(1,:));
```

Solver 3: Solver

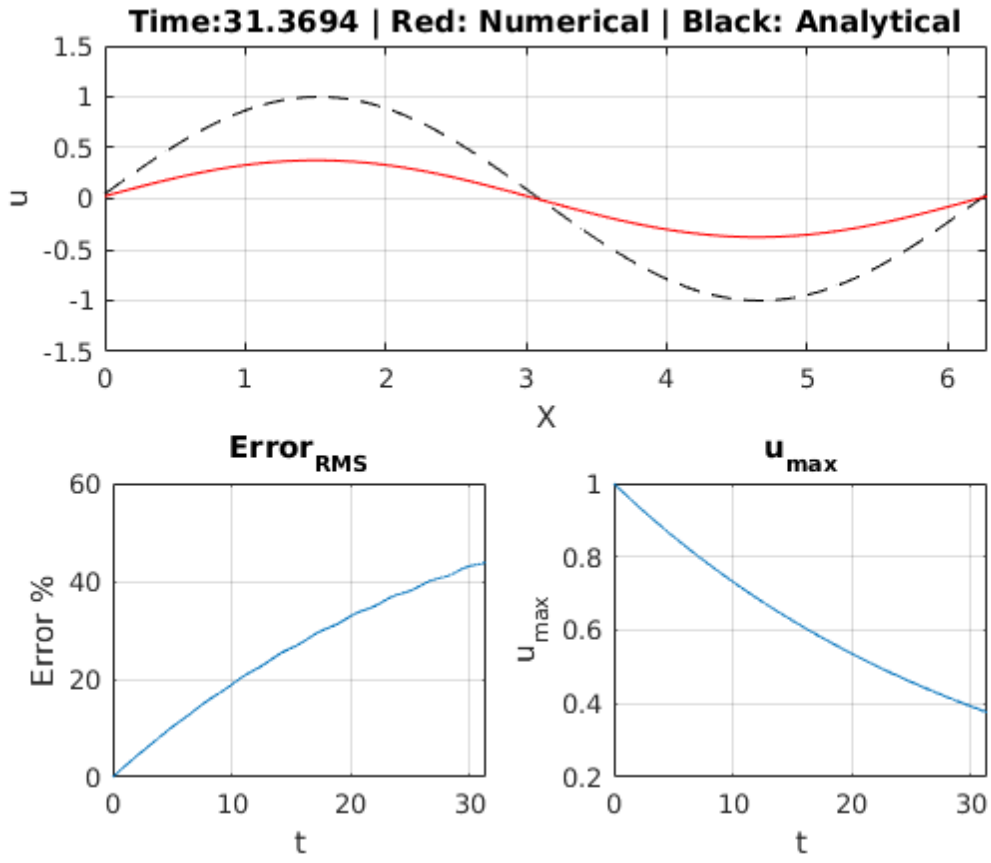
```
for n = 2:size(u,1)
    for j = 1:size(u,2)
        if j == 1
            u_x = (u(n-1,j) - u(n-1,end-1))/(dx);
        elseif j == size(u,2)
            u_x = (u(n-1,j) - u(n-1,j-1))/(dx);
        else
            u_x = (u(n-1,j) - u(n-1,j-1))/(dx);
        end
        u_t = - u_x;
        u(n,j) = u(n-1,j) + dt*u_t;
    end
    % Error and u_max calculation
```

```
    u_max(n) = max(abs(u(n,:)));
    error(n) = (rms(u(n,:) - sin(x-t(n))))*100/max(sin(x-t(n)));
end
```

Solver 3: Plotting the results

```
% Solution
figure
tic
for n = 1:max(1,floor(0.1/dt)):size(u,1)
    pause((n-1)*dt - toc)
    subplot(2,2,[1,2])
    plot(x,sin(x-t(n)),'--k',x,u(n,:),'r');grid on
    xlim([0,2*pi]);ylim([-1.5,1.5])
    title(['Time:',num2str((n-1)*dt),' | Red: Numerical | Black: Analytical']);
    xlabel('X');ylabel('u')
end

% Error
subplot(2,2,3)
plot(t,error)
grid on;xlim([0,T])
title('Error_{RMS}');xlabel('t');ylabel('Error %')
subplot(2,2,4)
plot(t,u_max)
grid on;xlim([0,T])
title('u_{max}');xlabel('t');ylabel('u_{max}')
```



Published with MATLAB® R2016a