FMAT3888 Tutorial Week 4

Group 3

September 1, 2021

Abstract

This tutorial goes over some of the main difference methods for numerically evaluating ODEs.

Q1. a)

Centred difference around $u''(t)$ and forward difference around $u'(t)$.

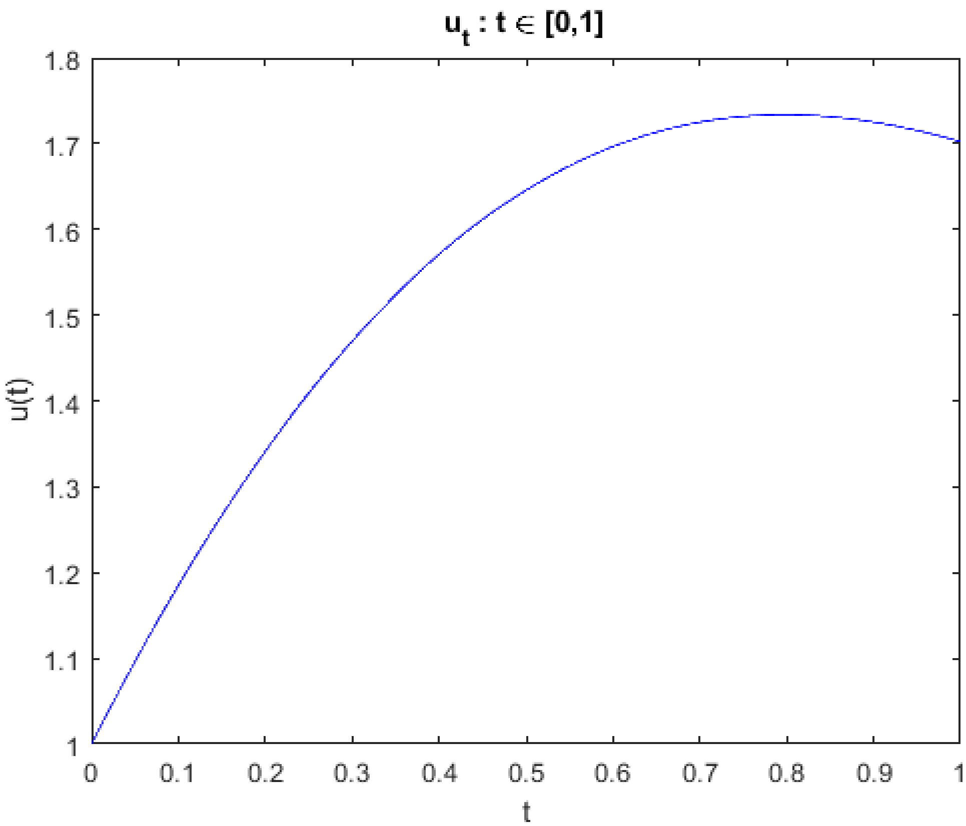
$$u''(t) + (1+t^2)u'(t) + u(t) = 0 \tag{1}$$

$$\frac{u(t+\Delta t) - 2\Delta t + u(t-\Delta t)}{(\Delta t)^2} + (1+t^2)\frac{u(t+\Delta t) - u(t)}{t} + u(t) = 0 \tag{2}$$

$$u(t+\Delta t) = \frac{2u(t) + (1+t^2)\Delta t u(t) - (\Delta t)^2 u(t)}{1+\Delta t(1+t^2)} - \frac{u(t-\Delta t)}{1+\Delta t(1+t^2)} \tag{3}$$

```
T=1;
dt=2^(-10);
t=(0:dt:1);
M=T/dt;
u0=1;
up0=2; % "up" means ‘u-prime’, the first time-derivative of u

% Sparse matrix to be updated
u=zeros(1,M+1);

% IVP conditions
u(1)=u0
u(2)=u0+up0*dt % we need to add the element u(2) since forward
    differencing updates at 3, i.e. i(3) -> i(2) - i(1)

% Loop through grid
for i=3:(M+1)
    u(i) = (2+(1+(i-2)^2*dt^2)*dt-dt^2)/(1+(i-2)^2*dt^2)*dt)*u(i-1)-u(
    i-2)/(1+(1+(i-2)^2*dt^2)*dt);
end

% Create a display table to find the value of u(1)
T=table([u(1)],[u(end)],'VariableNames',{'t=0','t=1'},'RowName',{'u(t)
    '});
display(T)

% Plot u(t)
plot(t,u,'-b');
title('u_t : t \in [0,1]')
xlabel('t')
ylabel('u(t)')
```

t	t=0	t=1
u(t)	1	1.7024



Q1. b)

Centred difference around $u''(t)$ and backward difference around $u'(t)$.

$$u''(t) + (1+t^2)u'(t) + u(t) = 0 \tag{4}$$

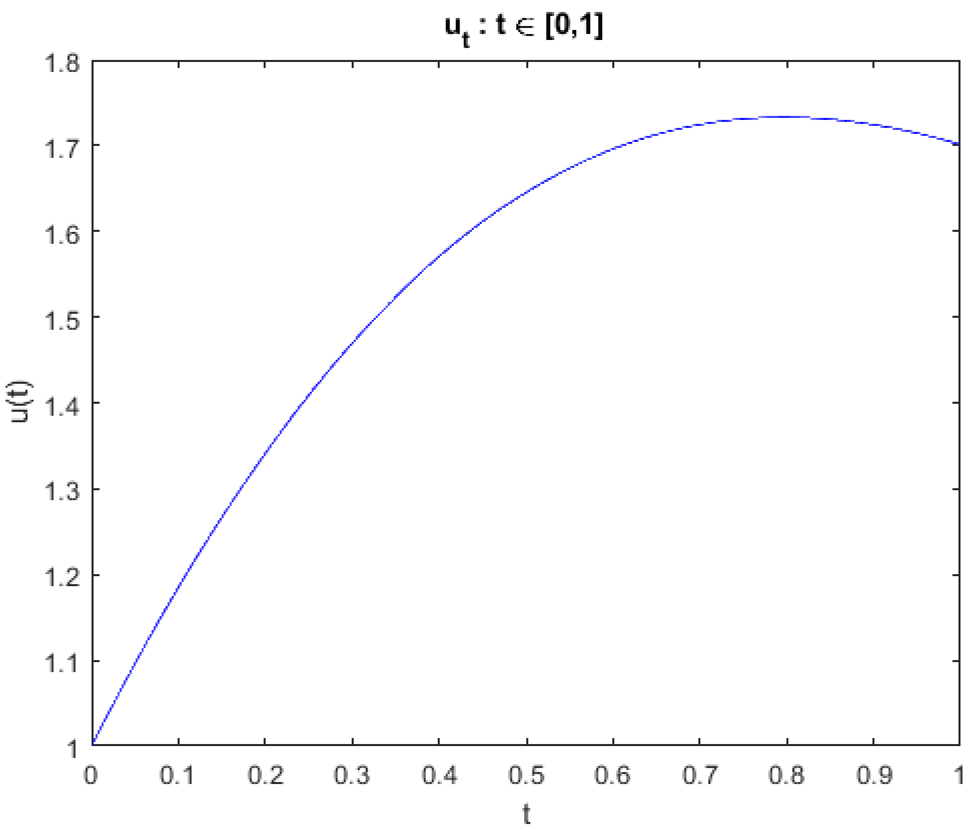
$$\frac{u(t+\Delta t) - 2\Delta t + u(t-\Delta t)}{(\Delta t)^2} + (1+t^2)\frac{u(t) - u(t-\Delta t)}{\Delta t} + u(t) = 0 \tag{5}$$

$$u(t+\Delta t) = 2u(t) - (1+t^2)\Delta t u(t) - (\Delta t)^2 u(t) - u(t-\Delta t)(\Delta t(1+t^2) - 1) \tag{6}$$

```
T=1;
dt=2^(-10);
t=(0:dt:1);
M=T/dt;
u0=1;
up0=2; % "up" means ‘u-prime’, the first time-derivative of u

% Sparse matrix to be updated
u=zeros(1,M+1);

% IVP conditions
u(1)=u0
u(2)=u0+up0*dt % we need to add the element u(2) since forward differencing updates at 3, i.e. i(3) -> i(2) - i(1)

% Loop through grid
```

```
for i=3:(M+1)
    u(i) = (2-(1+(i-2)^2*dt^2)*dt-dt^2)*u(i-1)+((1+(i-2)^2*dt^2)*dt-1)*u(i-2);
end

% Create a display table to find the value of u(1)
T=table([u(1)],[u(end)],'VariableNames',{'t=0','t=1'},'RowName',{'u(t)'});
display(T)

% Plot u(t)
plot(t,u,'-b');
title('u_t : t \in [0,1]')
xlabel('t')
ylabel('u(t)')
```

t	t=0	t=1
u(t)	1	1.7014



Q1. c)

Centred difference around $u''(t)$ and centred difference around $u'(t)$.

$$u''(t) + (1+t^2)u'(t) + u(t) = 0 \tag{7}$$

$$\frac{u(t+\Delta t) - 2\Delta t + u(t-\Delta t)}{(\Delta t)^2} + (1+t^2)\frac{u(t+\Delta t) - u(t-\Delta t)}{2\Delta t} + u(t) = 0 \tag{8}$$

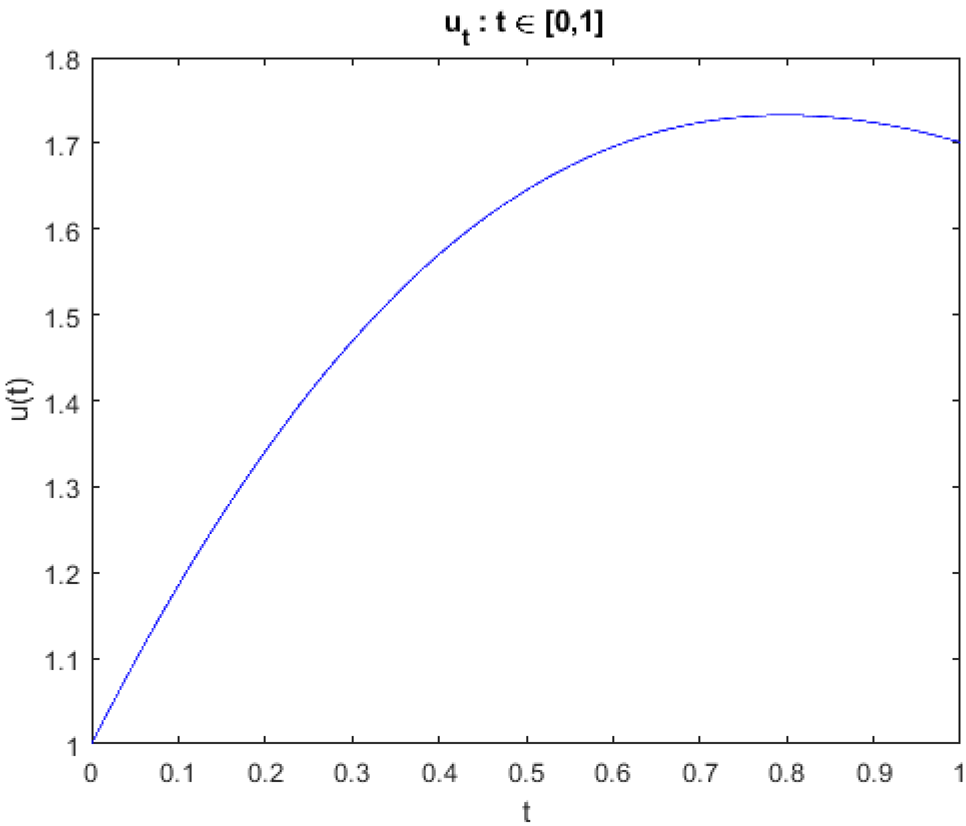
$$u(t+\Delta t) = \frac{2u(t) - (\Delta t)^2 u(t)}{\Delta t\left(\frac{1+t^2}{2}\right) + 1} - u(t-\Delta t)\frac{1 - \Delta t\left(\frac{1+t^2}{2}\right)}{1 + \Delta t\left(\frac{1+t^2}{2}\right)} \tag{9}$$

Initial condition for centred difference:

$$u''(0) = -3 \tag{10}$$

```matlab
T=1;
dt=2^(-10);
t=(0:dt:1);
M=T/dt;
u0=1;
up0=2; % "up" means 'u-prime', the first time-derivative of u
upp0=-3;

% Sparse matrix to be updated
u=zeros(1,M+1);

% IVP conditions
u(1)=u0;
u(2)=u0 + up0*dt + upp0*dt^2/2; % we need to add the element u(2) since forward differencing updates at 3, i.e. i(3) -> i(2) - i(1)

% Loop through grid
for i=3:(M+1)
    u(i) = (2-dt^2)/(1+(1+(i-2)^2*dt^2)*dt/2)*u(i-1)-(1-(1+(i-2)^2*dt^2)*dt/2)/(1+(1+(i-2)^2*dt^2)*dt/2)*u(i-2);
end

% Create a display table to find the value of u(1)
T = table([u(1)],[u(end)],'VariableNames',{'t=0','t=1'},'RowName',{'u(t)'});
display(T)

% Plot u(t)
plot(t,u,'-b');
title('u_t : t \in [0,1]')
xlabel('t')
```

t	t=0	t=1
u(t)	1	1.7012



Q1. d)

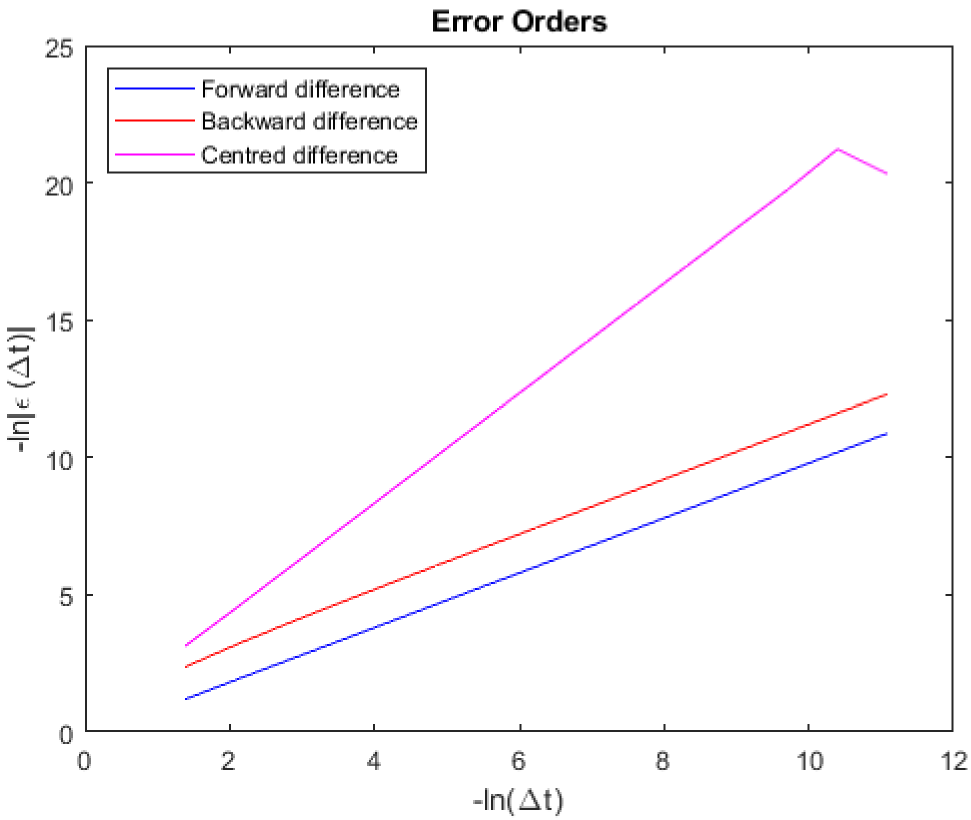
$$\varepsilon(\Delta t) = c\Delta t^{\alpha} - c(2\Delta t)^{\alpha} \tag{11}$$

$$-\ln|\varepsilon(\Delta t)| = -\alpha \ln(\Delta t) + \ln|c(2^{\alpha} - 1)| \tag{12}$$

```
T=1;
u0=1;
up0=2;
upp0=-3;
Nmax=16;
udt=zeros(1,Nmax);

for N=1:Nmax
    dt=2^(-1*N);
    M=T/dt;
    u=zeros(1,M+1);
    u(1)=u0;
    u(2)=u0+up0*dt;

    for i=3:(M+1)
        u(i) = (2+(1+(i-2)^2*dt^2)*dt-dt^2)/(1+(1+(i-2)^2*dt^2)*dt)*u(i-1)-u(i-2)/(1+(1+(i-2)^2*dt^2)*dt);
    end
    udt(N)=u(M+1);
end
X=zeros(1,Nmax-1);
```

```
Y=zeros(1,Nmax-1);
for N=1:(Nmax-1)
    X(N)=log(2^(N+1));
    Y(N)=-log(abs(udt(N+1)-udt(N)));
end
plot(X,Y,'blue')
xlabel('-ln(\Delta{t})')
ylabel('-ln|\epsilon (\Delta{t})|')
hold on;


for N=1:Nmax
    dt=2^(-1*N);
    M=T/dt;
    u=zeros(1,M+1);
    u(1)=u0;
    u(2)=u0+up0*dt;

    for i=3:(M+1)
        u(i) = (2-(1+(i-2)^2*dt^2)*dt-dt^2)*u(i-1)+((1+(i-2)^2*dt^2)*
   dt-1)*u(i-2);
    end
    udt(N)=u(M+1);
end
X=zeros(1,Nmax-1);
Y=zeros(1,Nmax-1);
for N=1:(Nmax-1)
    X(N)=log(2^(N+1));
    Y(N)=-log(abs(udt(N+1)-udt(N)));
end
plot(X,Y,'red')
xlabel('-ln(\Delta{t})')
ylabel('-ln|\epsilon (\Delta{t})|')
hold on;

for N=1:Nmax
    dt=2^(-1*N);
    M=T/dt;
    u=zeros(1,M+1);
    u(1)=u0;
    u(2)=u0+up0*dt+upp0*dt^2/2;

    for i=3:(M+1)
        u(i) = (2-dt^2)/(1+(1+(i-2)^2*dt^2)*dt/2)*u(i-1)-(1-(1+(i-2)
   ^2*dt^2)*dt/2)/(1+(1+(i-2)^2*dt^2)*dt/2)*u(i-2);
    end
    udt(N)=u(M+1);
end
X=zeros(1,Nmax-1);
Y=zeros(1,Nmax-1);
for N=1:(Nmax-1)
    X(N)=log(2^(N+1));
    Y(N)=-log(abs(udt(N+1)-udt(N)));
end
plot(X,Y,'magenta')
xlabel('-ln(\Delta{t})')
```

```
ylabel('-ln|\epsilon (\Delta{t})|')
hold on;

title('Error Orders')
legend({'Forward difference','Backward difference','Centred difference
    '},'Location','northwest')
```



It's clear from the above graph that the error for the centred difference follows the slope $\alpha \approx 2$ while those of the forward and backward differences follow $\alpha \approx 1$. Therefore, we conclude that the error orders for each method are as follows,

i. Centred difference truncation error $\sim \mathcal{O}h^2)$

ii. Forward difference truncation error $\sim \mathcal{O}(h)$

iii. Backward difference truncation error $\sim \mathcal{O}(h)$

Out of these three methods, the <u>centred difference</u> gives us the smallest truncation error.

Q2. a)

Let $\lambda = \frac{\Delta t}{\Delta x^2}$,

$$u(t+\Delta t, x) = \lambda u(t, x+\Delta x) + (1-2\lambda)u(t, x-\Delta x) \tag{13}$$

$$\frac{u(t+\Delta t, x) - u(t,x)}{\Delta t} - \frac{u(t, x+\Delta x) - 2u(t,x) + u(t, x-\Delta x)}{\Delta x^2} = 0 \tag{14}$$

This two-dimensional differential equation has the following boundary conditions,

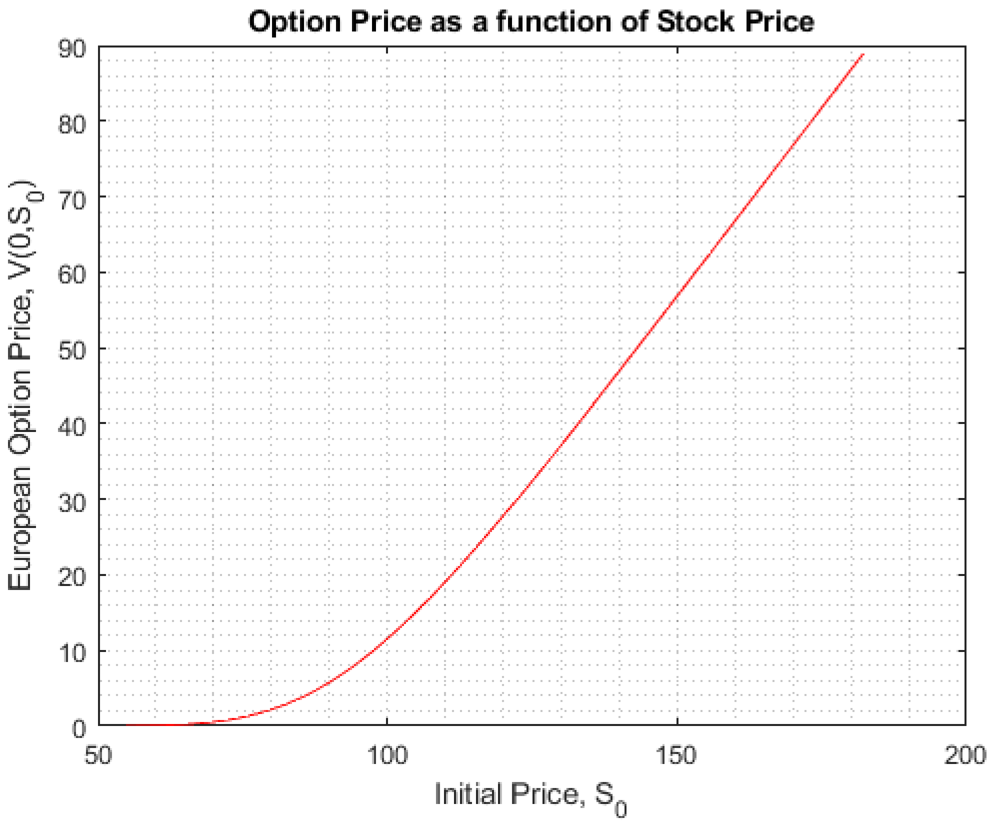
$$u(t, x_{min}) = 0 \tag{15}$$

$$u(t, x_{max}) = e^{\left(\frac{q+1}{2}x_{max} + \frac{(q+1)^2}{4}t\right)} - e^{\left(\frac{q-1}{2}x_{max} + \frac{(q-1)^2}{4}t\right)} \tag{16}$$

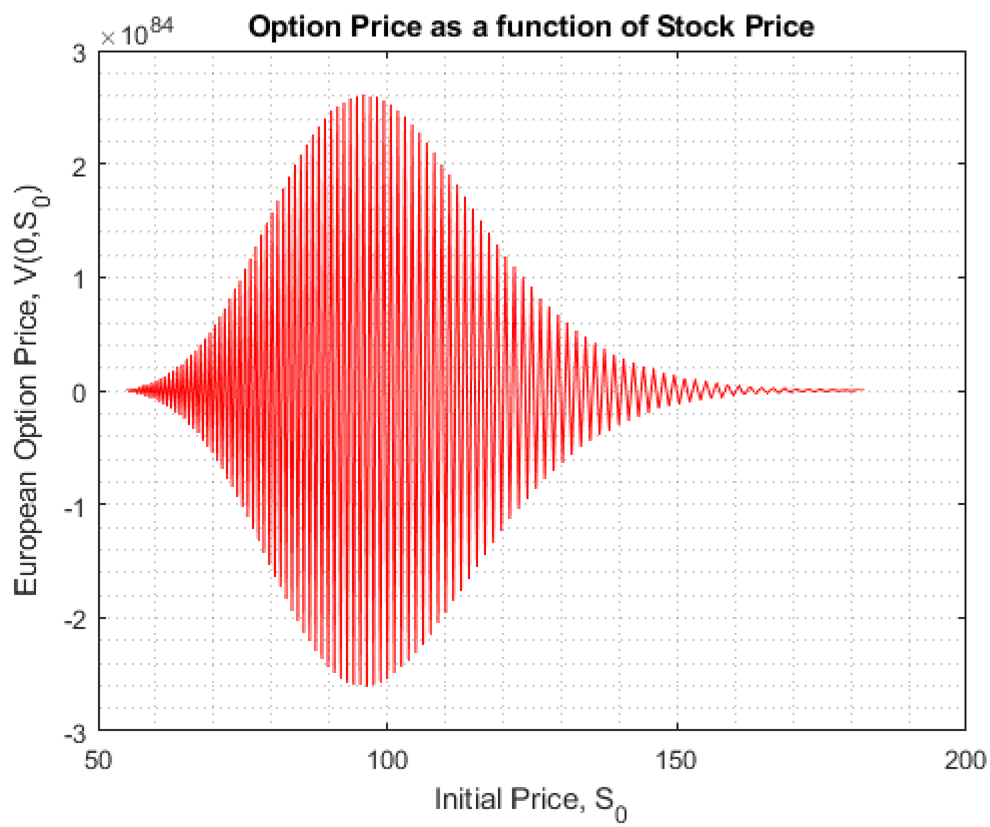
We know for European options that $S_0 = Ke^x$, and so the pricing model for $t = 0$ is,

$$V(0, Ke^x) = u\left(\frac{\sigma^2 T}{2}, x\right) K e^{\left(-\frac{q-1}{2}x - \frac{(q+1)^2}{4}\frac{\sigma^2 T}{2}\right)} \tag{17}$$

```
T=1;
sigma=0.2;
r=0.07;
K=100;
q=2*r/sigma^2;
M=1000;
N=100;
dt=sigma^2*T/(2*M);
dx=6*sigma*sqrt(T)/N;
lambda=dt/dx^2;
x=zeros(1,N+1);
for n=1:N+1
    x(n)=-3*sigma*sqrt(T)+(n-1)*dx;
end
%u(n) is dependant on x(n), we need two new loops
u=zeros(1,N+1);
for n=1:N+1
    u(n)=max(exp((q+1)/2*x(n))-exp((q-1)/2*x(n)),0);
end
%update u(t + delta t)
for m=2:M+1
    v=zeros(1,N+1);%
    v(1)=0;
    v(N+1)=exp((q+1)*x(N+1)/2+(q+1)^2*(m-1)*dt/4)-exp((q-1)*x(N+1)/2+(q-1)^2*(m-1)*dt/4);
    for n=2:N
        v(n)=lambda*u(n-1)+(1-2*lambda)*u(n)+lambda*u(n+1);
    end
    u=v;
end
S=zeros(1,N+1);
V=zeros(1,N+1);
for n=1:N+1
    S(n)=K*exp(x(n));
    V(n)=u(n)*K*exp(-(q-1)/2*x(n)-(q+1)^2/4*sigma^2*T/2);
end
plot(S,V,'red','LineWidth',1,'Color','red');
title('Option Price as a function of Stock Price')
xlabel('Initial Price, S_0')
ylabel('European Option Price, V(0,S_0)')
grid minor
```


Option Price as a function of Stock Price
European Option Price, V(0,S0)
Initial Price, S0
90
80
70
60
50
40
30
20
10
0
50
100
150
200

```
...N=200;...
```


Option Price as a function of Stock Price
×10^84
European Option Price, V(0,S0)
Initial Price, S0
3
2
1
0
-1
-2
-3
50
100
150
200

Now we want to compute the option prices for each element in $S_0 = (60, 100, 150)$, for $N = 100$,

$\mathbf{S_0}$	60	100	150
$\mathbf{V(0, S_0)}$	0.0692	11.5393	56.8311

Repeating this for $N = 200$,

$\mathbf{S_0}$	60	100	150
$\mathbf{V(0, S_0)}$	6.3867e+82	-2.5346e+84	-1.4197e+82

The price of the option when taken as a function of stock price is a positively increasing function, this can be shown clearly when we have M=1000 and N=100 grid points. The price of the European option as a function of stock price when we increase the resolution of the grid to M=1000 and N=200 is aperiodic and unstable.

We can get to the bottom of this by examining λ, since this is the only part of the algorithm where N shows up.

$$\lambda = \frac{\sigma^2 T}{2M} \frac{N}{6\sigma\sqrt{T}} \tag{18}$$

$$\lambda \propto N \tag{19}$$

We require a stability condition to restrict the value of λ so that the algorithm stays stable. This is not at all obvious, but we know that this condition must be [1],

$$\lambda < \frac{1}{2} \tag{20}$$

References

[1] LeVeque, Randall (2002). Finite Volume Methods for Hyperbolic Problems. Cambridge University Press. FTCS Wikipedia