AAE 694 - Project2 Martin Kearney

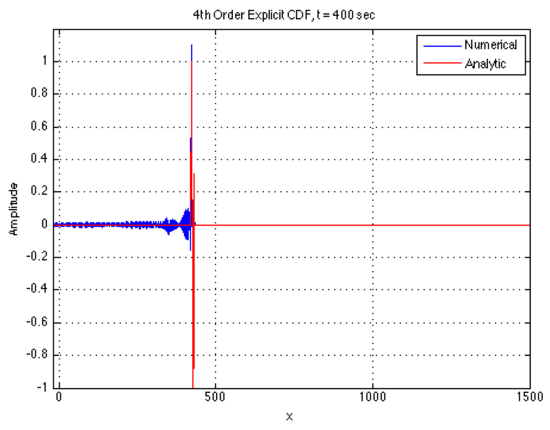
The purpose of this project is to implement a numerical solution to the first order wave equation.

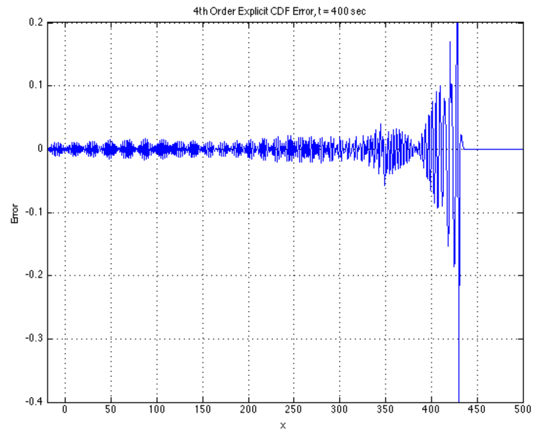
$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = 0$$

We are given two sets of initial conditions and asked to implement and observe the behavior of three different solvers: explicit fourth-order central difference, explicit sixth-order central difference, and Tam's fourth-order dispersion relation preserver. The implementations are appended to the bottom of the document.

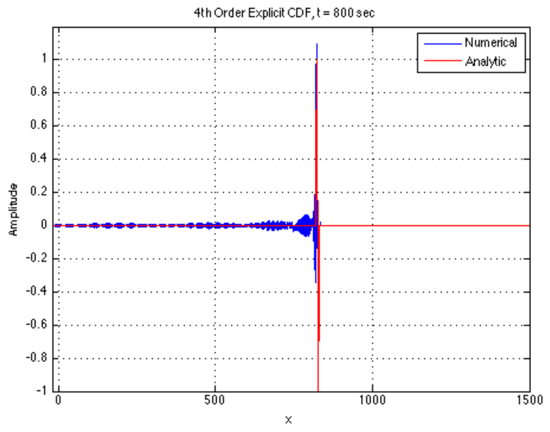
The results will be discussed below.

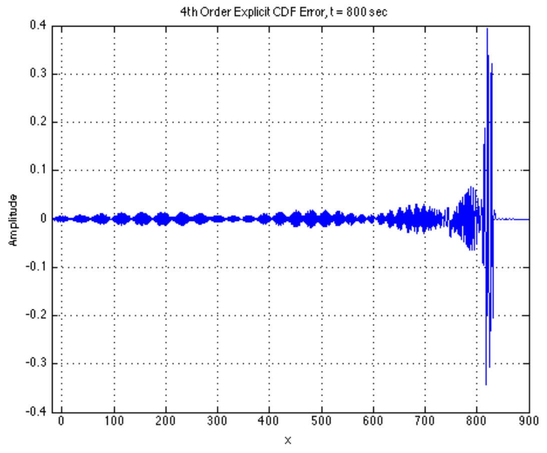
1. Propagating Sine wave
    a. Explicit Fourth-Order Central Difference Scheme



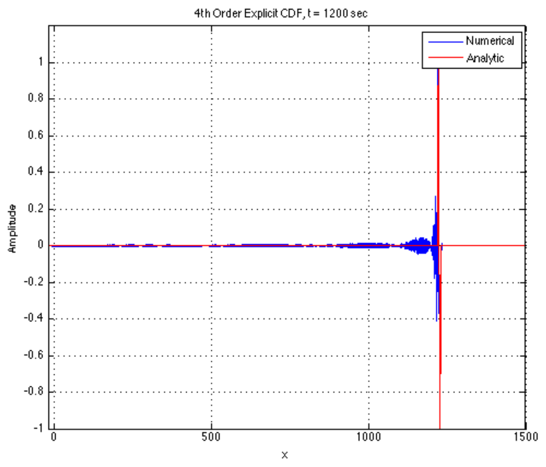


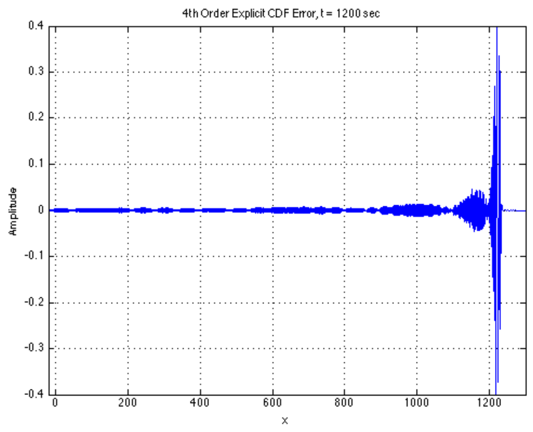
At 400 seconds, the wave is in the correct location, but its amplitude has drifted slightly positive. The error is fairly low amplitude for most of the domain with a fundamental frequency of 0.5 Hz and a beat frequency of about 0.025 Hz. The RMS error over the whole domain is 0.0171.





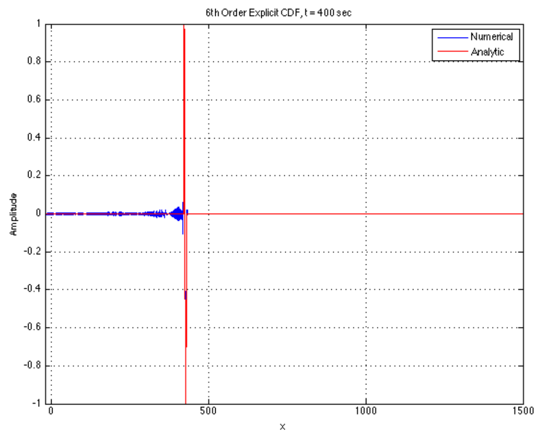
The same behavior continues at 800 seconds. The fundamental error frequency is the same and the beat frequency of the error has been halved (0.0125 Hz). The RMS error over the whole domain is 0.0254.

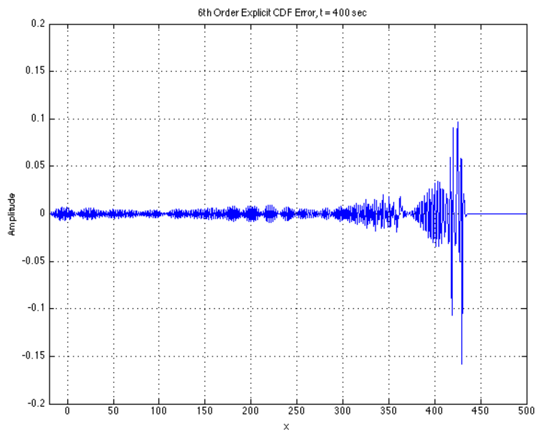




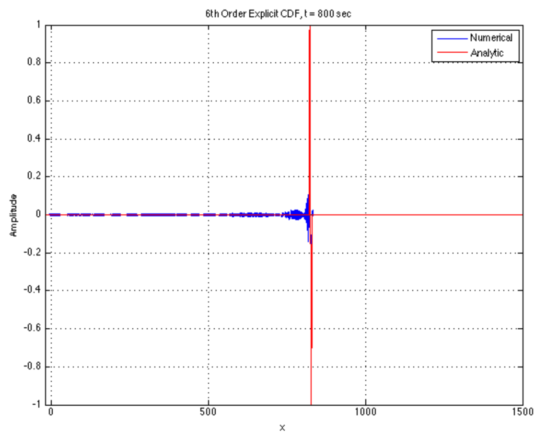
Finally, at 1200 seconds, the same patterns continue. As expected, the fundamental error frequency is 0.5 Hz, the beat frequency is (0.0083 Hz). The RMS error over the whole domain is 0.0317.

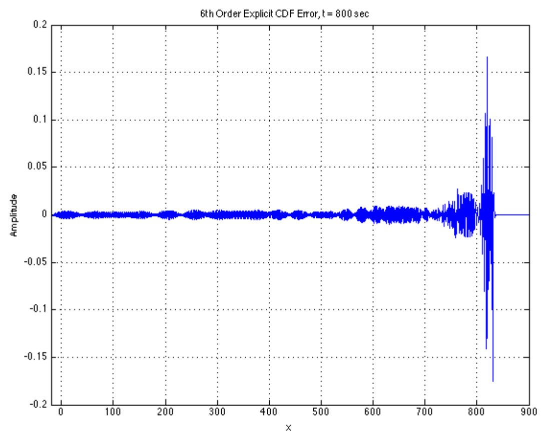
b. Explicit Sixth-Order Central Difference Scheme



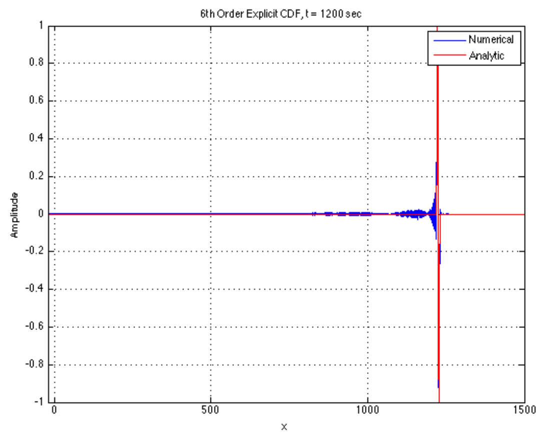


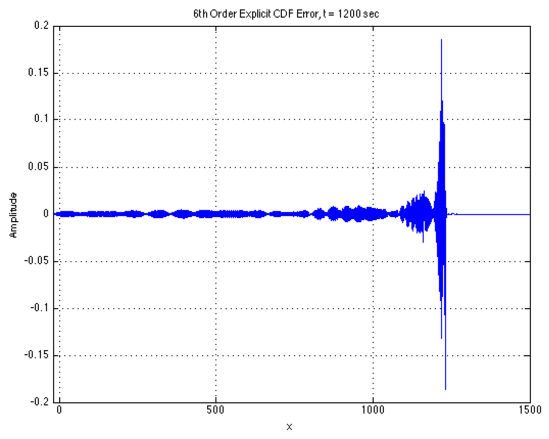
At 400 seconds, the wave is in the correct location, but its amplitude has shrunk slightly. The error is fairly low amplitude for most of the domain with a fundamental frequency of 0.5 Hz and a beat frequency of about 0.025 Hz. The RMS error over the whole domain is 0.0069.





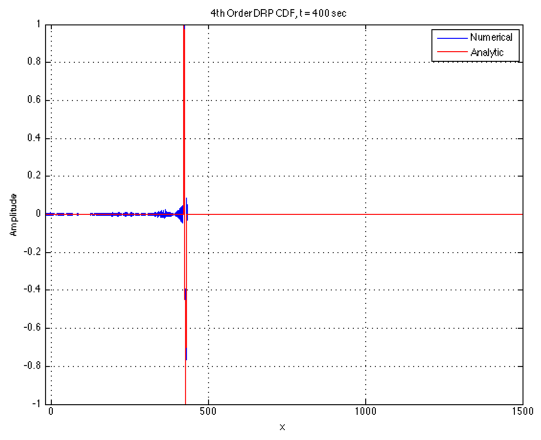
The same behavior continues at 800 seconds. The fundamental error frequency is the same and the beat frequency of the error has been halved (0.0125 Hz). The RMS error over the whole domain is 0.0092.

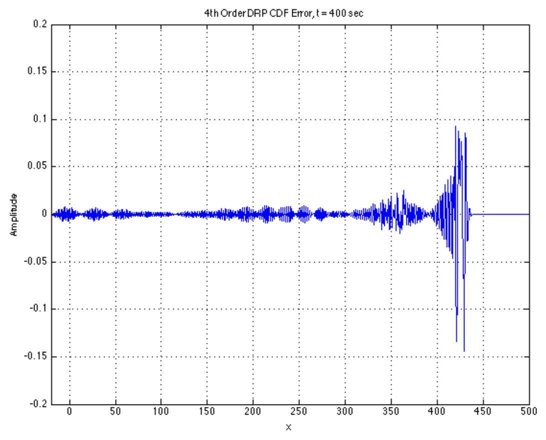




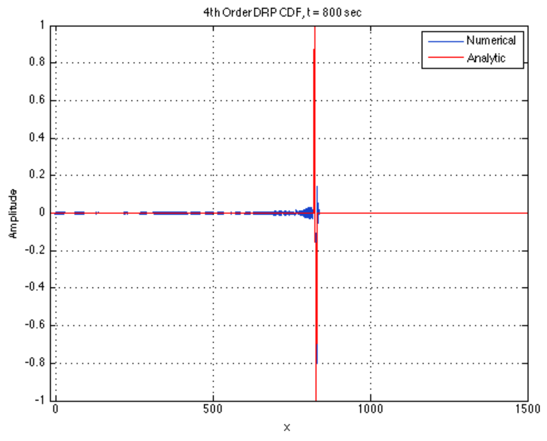
Finally, at 1200 seconds, the same patterns continue. As expected, the fundamental error frequency is 0.5 Hz, the beat frequency is (0.0083 Hz). The RMS error over the whole domain is 0.010.

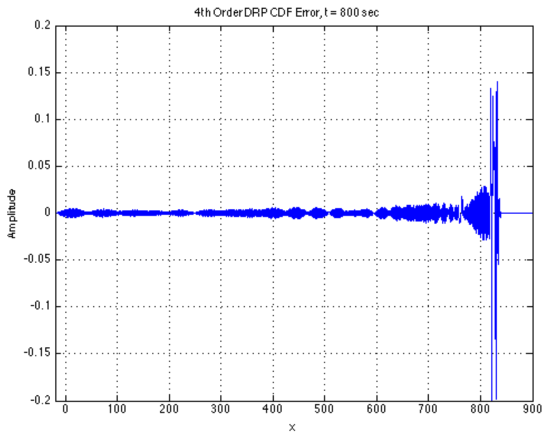
c. Tam’s fourth-Order DRP Scheme



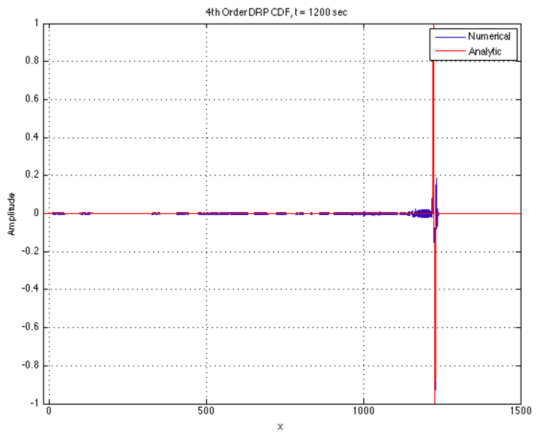


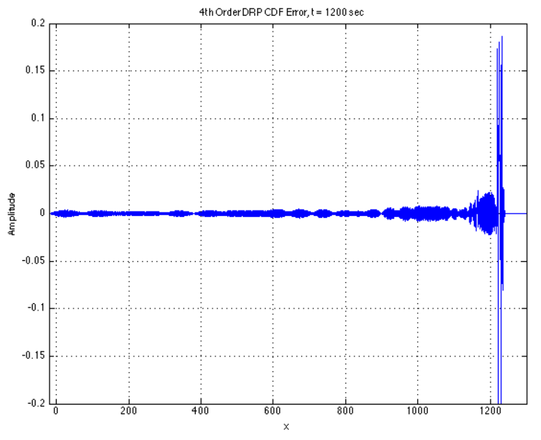
At 400 seconds, the wave is in the correct location, but its amplitude has shrunk slightly. The error is fairly low amplitude for most of the domain with a fundamental frequency of 0.5 Hz and a beat frequency of about 0.025 Hz. A second, ultra low, beat frequency can also be observed (~0.003 Hz). The RMS error over the whole domain is 0.0078.





The same behavior continues at 800 seconds. The fundamental error frequency is the same and the beat frequency of the error has been halved (0.0125 Hz). The ultra low beat frequency has become hard to detect. The RMS error over the whole domain is 0.011.

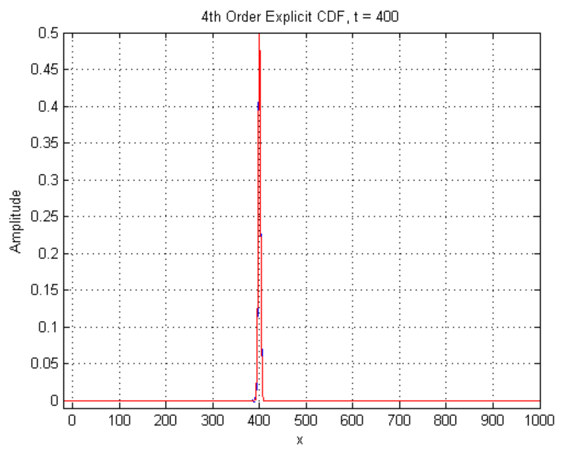


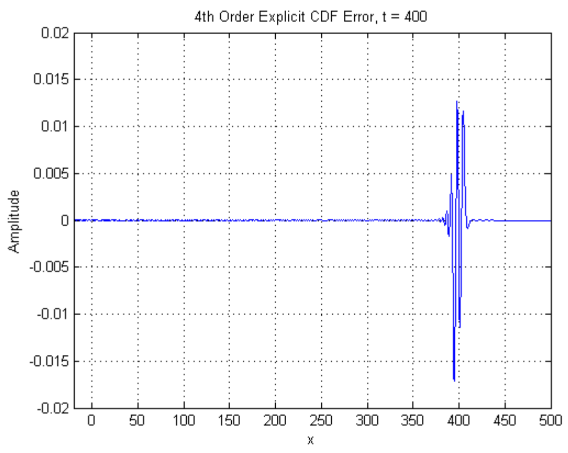


Finally, at 1200 seconds, the same patterns continue. As expected, the fundamental error frequency is 0.5 Hz, the beat frequency is (0.0083 Hz). The RMS error over the whole domain is 0.014.

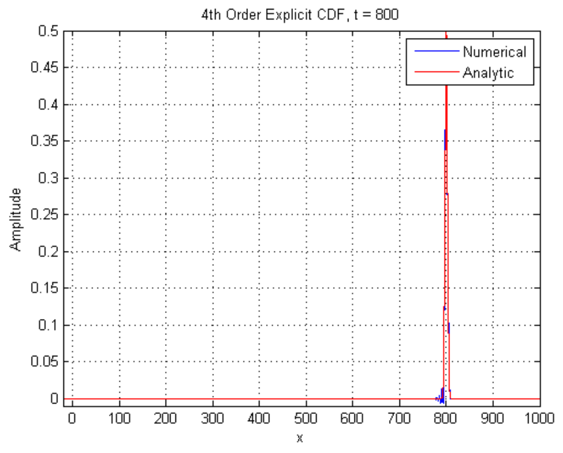
Conclusions for Sine wave: All three methods have similar error characteristics. The 6th order minimizes the most error and is almost indistinguishable from Tam's method. There may be some evidence of differences in the ultra low beat frequency, but it is hard to say.

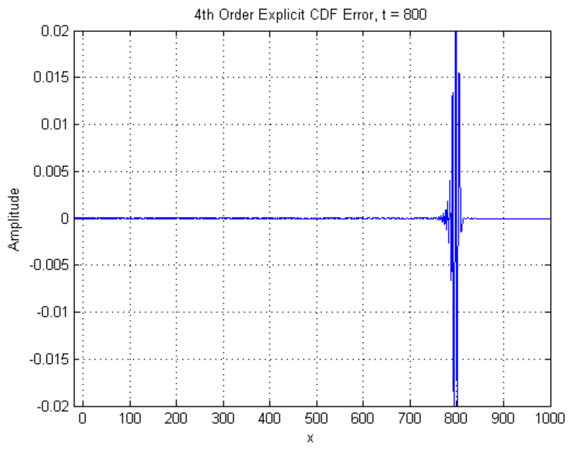
2. Gaussian Wave Packet
    a. Explicit fourth-Order Central Difference Scheme





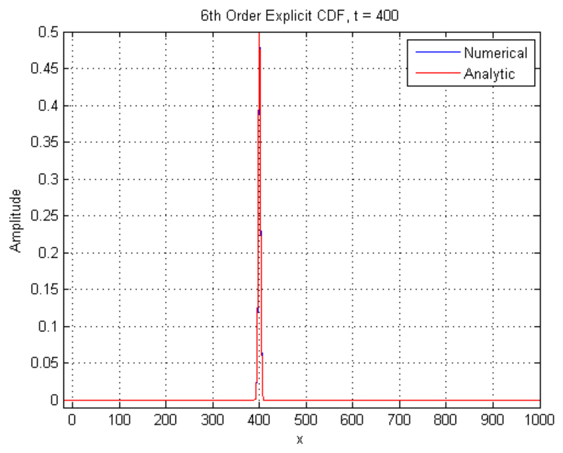
At 400 seconds, the wave is in the correct location, and errors are localized to the large gradient regions on the side of the waveform. The error is very low amplitude for most of the domain. The RMS error over the whole domain is 9.3945e-004.

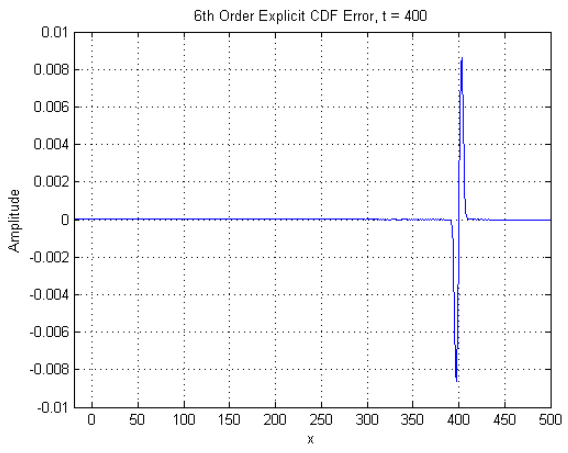




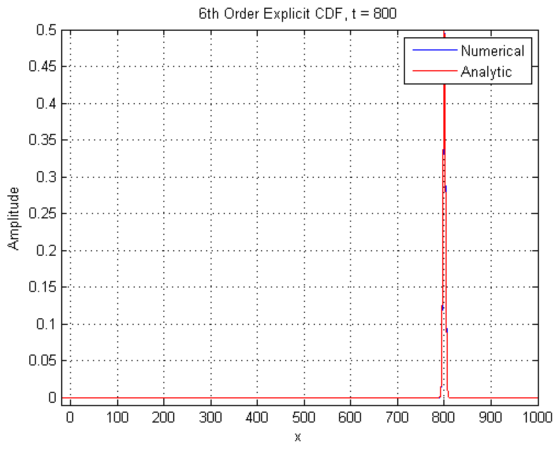
The same behavior continues at 800 seconds. There is a small increase in error amplitude on the trailing edge of the waveform. The RMS error over the whole domain is 1.8093e-003.

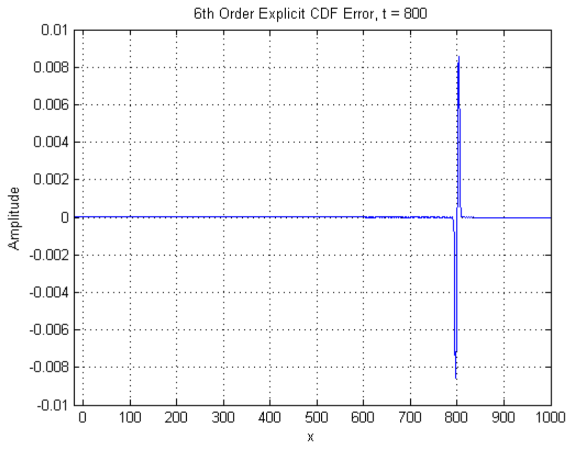
b. Explicit Sixth-Order Central Difference Scheme





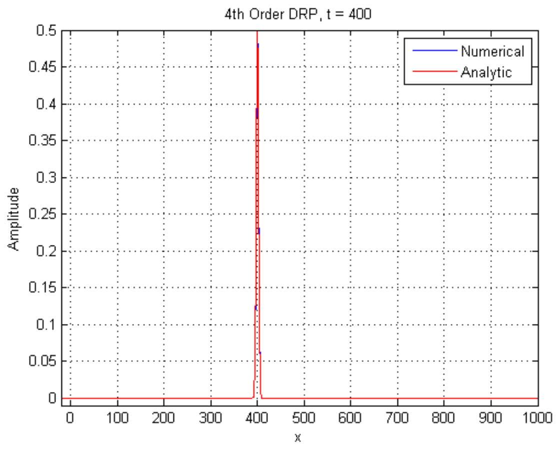
At 400 seconds, the wave is in the correct location, and errors are highly localized to the large gradient regions on the side of the waveform. The error is very low amplitude for most of the domain. The RMS error over the whole domain is 5.5082e-004.

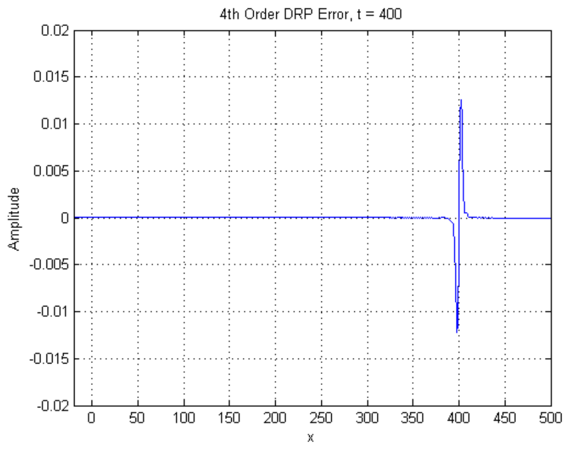




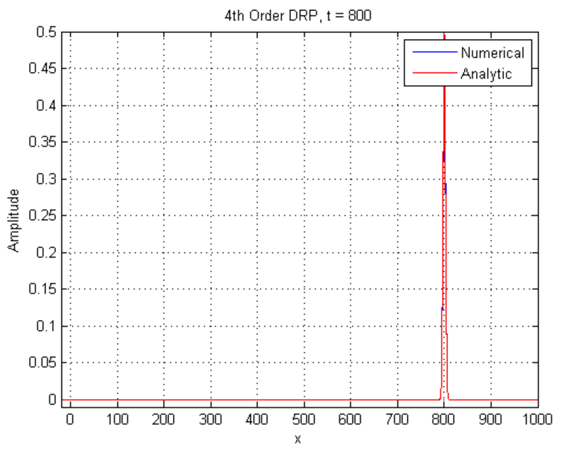
The same behavior continues at 800 seconds. There is no appreciable difference in error. The RMS error over the whole domain is 5.397e-004.

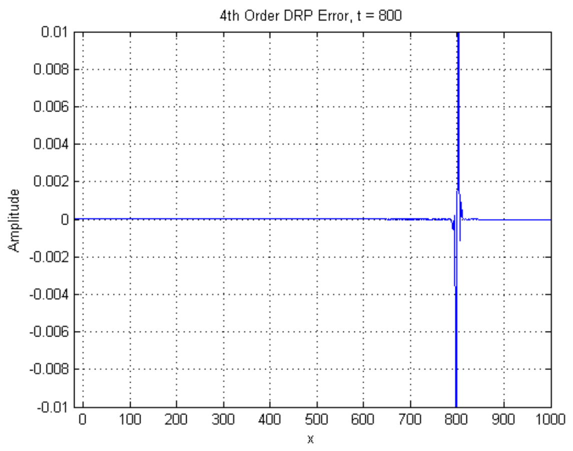
c. Tam’s Fourth-Order DRP Scheme





At 400 seconds, the wave is in the correct location, and errors are highly localized to the large gradient regions on the side of the waveform. The error is very low amplitude for most of the domain. The RMS error over the whole domain is 7.0704e-004.





The same behavior continues at 800 seconds. There is no appreciable difference in error. The RMS error over the whole domain is 8.7827e-004.

Conclusions for Gaussian wave packet: There is almost no observable difference between 6th order and DRP. The only distinguishing difference is a slightly higher error amplitude in DRP. Also, the error is highly confined in all cases to the region immediately near the waveform.

Generally, the differences between these two initial conditions are caused by the discontinuities in the gradient of the sine wave. This discontinuity sets off and error wave which propagates backward with considerable amplitude.

```
function u = proj2()

bc = 2;
mthd = 3;

b(1) = 2.3025580883830;     %n-0
b(2) = -2.4910075998482;    %n-1
b(3) = 1.57434009331815;    %n-2
b(4) = -0.38589142217162;   %n-3

N = length(b);  %number of temporal back-steps

dx = 0.5;   %spatial step-size
dt = 0.05;  %temporal step-size
Lx = -20;   %Left edge of spatial domain
Rx = 1500;   %Right edge of spatial domain

x = (Lx:dx:Rx)';
T = (0:dt:800)';

u = zeros(length(x),N+1);
dUdx = u;

%Initial condition
switch bc
    case 1
        st = 20;

        Li = min(find(x > st));
        Ri = min(find(x > st+9));
        u(Li:Ri,1) = sin(pi*(x(Li:Ri)-20)/5);
        u(Li:Ri,2) = u(Li:Ri,1);
        u(Li:Ri,3) = u(Li:Ri,1);
        u(Li:Ri,4) = u(Li:Ri,1);
    otherwise
        u(:,1) = 0.5*exp(-log(2)*x.^2/9);
        u(:,2) = u(:,1);
        u(:,3) = u(:,1);
        u(:,4) = u(:,1);
end

%Set up Weights
[idx,W] = getWeights(mthd);

%Iteration
plot(x,u(:,1),'b')
hold on
for t = 1:length(T)
    u(:,2:end) = u(:,1:end-1);
    dUdx(:,2:end) = dUdx(:,1:end-1);

    dUdx(:,2) = dUcalc(u(:,2),idx,W)/dx;
    u(:,1) = u(:,1) - dt*dUdx(:,2:end)*b';

    u(1:2,1) = 0; u(end-1:end,1) = 0;   %Boundary conditions

    if ~mod(T(t),100)
%     if abs(u(2,1)) > eps
        plot(x,u(:,1),'Color',rand(3,1))
        title(num2str(T(t)))
        pause(0.5)
    end
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [idx,W] = getWeights(mthd)

switch mthd
    case 1  %4th order Central difference
        idx(:,3) = [-2 -1 0 1 2]';
        W(:,3) = fdweights_arb(idx(:,3),1);

        idx(:,2) = idx(:,3)+1;
        W(:,2) = fdweights_arb(idx(:,2),1);
        idx(:,1) = idx(:,3)+2;
        W(:,1) = fdweights_arb(idx(:,1),1);

        idx(:,4) = idx(:,3)-1;
        W(:,4) = fdweights_arb(idx(:,4),1);
        idx(:,5) = idx(:,3)-2;
        W(:,5) = fdweights_arb(idx(:,5),1);
    case 2  %6th order Central difference
        idx(:,4) = [-3 -2 -1 0 1 2 3]';
        W(:,4) = fdweights_arb(idx(:,4),1);

        idx(:,3) = idx(:,4)+1;
        W(:,3) = fdweights_arb(idx(:,3),1);
        idx(:,2) = idx(:,4)+2;
        W(:,2) = fdweights_arb(idx(:,2),1);
        idx(:,1) = idx(:,4)+3;
        W(:,1) = fdweights_arb(idx(:,1),1);

        idx(:,5) = idx(:,4)-1;
        W(:,5) = fdweights_arb(idx(:,5),1);
        idx(:,6) = idx(:,4)-2;
        W(:,6) = fdweights_arb(idx(:,6),1);
        idx(:,7) = idx(:,4)-3;
        W(:,7) = fdweights_arb(idx(:,7),1);
    otherwise   %4th order DRP
        idx(:,4) = [-3 -2 -1 0 1 2 3]';
        W(1,4) = -0.02084314277031176;    %i-3
        W(2,4) = 0.166705904414580469;    %i-2
        W(3,4) = -0.77088238051822552;    %i-1
        W(4,4) = 0;                       %i
        W(5,4) = 0.77088238051822552;     %i+1
        W(6,4) = -0.166705904414580469;   %i+2
        W(7,4) = 0.02084314277031176;     %i+3

        idx(:,5) = idx(:,4)-1;
        W(1,5) = 0.026369431;   %i-4
        W(2,5) = -0.166138533;  %i-3
        W(3,5) = 0.518484526;   %i-2
        W(4,5) = -1.273274737;  %i-1
        W(5,5) = 0.474760914;   %i
        W(6,5) = 0.468840357;   %i+1
        W(7,5) = -0.049041958;  %i+2

        idx(:,6) = idx(:,4)-2;
        W(1,6) = -0.048230454;  %i-5
        W(2,6) = 0.281814650;   %i-4
        W(3,6) = -0.768949766;  %i-3
        W(4,6) = 1.388928322;   %i-2
        W(5,6) = -2.147776050;  %i-1
        W(6,6) = 1.084875676;   %i
        W(7,6) = 0.209337622;   %i+1

        idx(:,7) = idx(:,4)-3;
        W(1,7) = 0.203876371;    %i-6
        W(2,7) = -1.128328861;   %i-5
        W(3,7) = 2.833498741;    %i-4
        W(4,7) = -4.461567104;   %i-3
        W(5,7) = 5.108851915;    %i-2
        W(6,7) = -4.748611401;   %i-1
        W(7,7) = 2.192280339;    %i

        idx(:,1) = idx(:,4)+3;
        W(:,1) = -flipud(W(:,7));   %i,i+1,i+2...
        idx(:,2) = idx(:,4)+2;
        W(:,2) = -flipud(W(:,6));  %i-1,i,i+1...
        idx(:,3) = idx(:,4)+1;
        W(:,3) = -flipud(W(:,5));  %i-2,i-1,i...
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function dU = dUcalc(u,idx,W)

SW = size(W); SW = SW(1);
S = (SW-1)/2;

dU = zeros(size(u));
for m = 1:SW
    dU(S+1:end-S) = dU(S+1:end-S) + W(m,S+1)*u(S+1+idx(m,S+1):end-S+idx(m,S+1));  %Bulk
        %Left boundary
```

```
%     for n = 1:S
%         dU(n) = dU(n) + W(m,n)*u(S+idx(m,n));     %This is unnecessary since we set↙
boundaries to zero
%     end
%         %Right boundary
%     for n = S+2:SW
%         dU(end-SW+n) = dU(end-SW+n) + W(m,n)*u(end-SW+n+idx(m,n));
%     end
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function W = fdweights_arb(idx,der)

A = zeros(length(idx)); %The spacing matrix
for i = 1:length(idx)
    A(:,i) =  idx(i).^(0:length(idx)-1); %Calculates exponents of spacings
end

B = zeros(length(idx),1);   %Supplementary column vector
B(der+1) = 1;   %sets proper order of derivative to have coefficient of one
W = A\B*factorial(der); %calculates weights for each index
```