AAE 694: Computational Aeroacoustics

Project 1

Michael Crawley

2/15/2011

Introduction

In this project, the one dimensional wave equation is solved for a given initial wave to numerically simulate the propagation of user defined waveforms. This numerical solution is accomplished using the finite difference method, with coefficients for the spatial derivative derived from Taylor series approximations of various orders of accuracy, as well as from Tam's Dispersion-Relation-Preserving optimized scheme. Discretization of the temporal derivative is accomplished using a third order, optimized, explicit scheme. The governing equation for this simulation is therefore:

$$\frac{\partial u}{\partial t} = -\frac{\partial u}{\partial x}$$

which is discretized as

$$u_{x_o}^{n+1} = u_{x_o}^{n} - \Delta t \sum_{j=0}^{3} b_j \left[\frac{1}{\Delta x} \sum_{i=-N}^{M} a_i u_{x_o+i} \right]^{n-j}$$

Where the superscript refers to the temporal index and the subscript refers to the spatial index. The coefficients a_i are found using either the standard Taylor series expansions, or Tam's fourth order dispersion-relation-preserving scheme for spatial derivatives. The coefficients b_j are found from Tam's third order optimized scheme for temporal derivatives. The propagation of the wave is simulated for 400 seconds.

Boundary Conditions

For the given problem, no boundary conditions have been specified; it has merely been assumed that the amplitude of the wave goes to zero (and hence, the spatial derivative goes to zero) at the boundaries. The treatment of the boundaries in the numerical simulation is therefore not immediately clear. Figure 1 shows sample results computed for a Gaussian wave using second and fourth order central difference schemes in which no special treatment has been made at the boundaries. While the lower order scheme shows no disconcerting behavior at the bounds, the higher order scheme clearly exhibits erroneous behavior at the left boundary (this behavior was also observed in the sixth order central difference scheme and the fourth order DRP optimized scheme, not shown). The spurious oscillations at the upstream side of the wave are due to the central difference schemes used, not the boundary conditions.

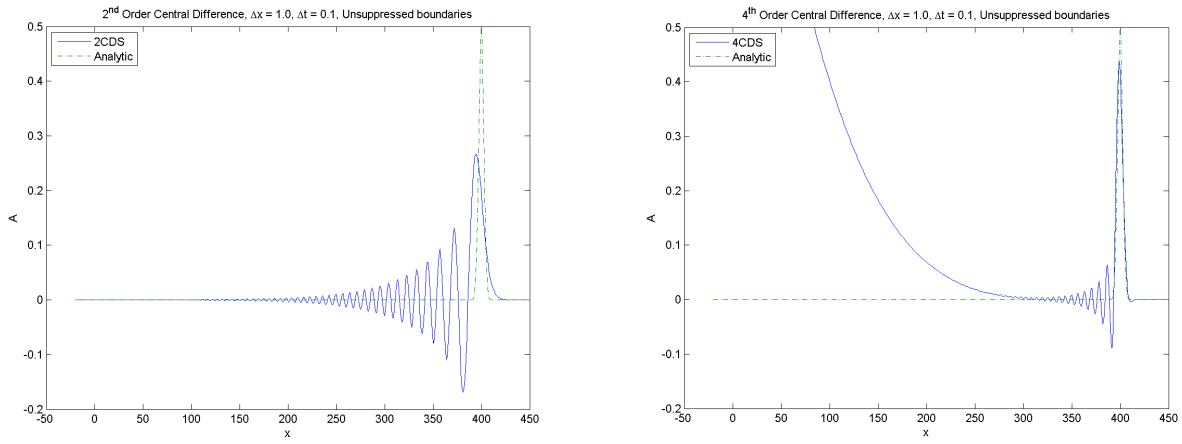

Figure 1: Effects of boundary conditions: fixed boundary conditions. Computational domain is -20 ≤ x ≤ 450. Left figure utilizes second order central difference scheme, whereas right figure utilizes fourth order central difference scheme. Solid lines represent the numeric solution while dashed lines represent the analytic solution.

In order to rectify this behavior at the left boundary, two methods were proposed: suppressing the spatial derivative at the boundaries (by setting them to zero after each computation), and extending the computational domain in the negative axial direction. Figure 2 shows the effects of suppressing the spatial derivate at the left boundary. Comparing the results against those in Figure 1, it is seen that boundary suppression is quite effective at removing the specious rise that developed previously at the boundary. Additionally, boundary suppression does not appear to affect the accuracy of the solution near the wave: the wave location and amplitude are unchanged between the two sets of figures, and the spurious oscillations on the upstream side of the wave are unaffected as well.

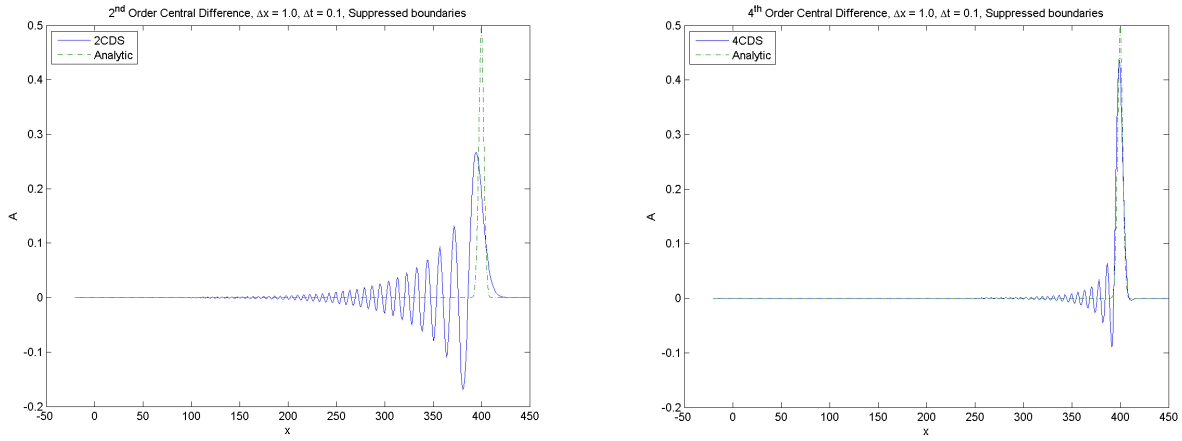

Figure 2: Effects of boundary conditions: suppressed boundary conditions. Computational domain is -20 ≤ x ≤ 450. Left figure utilizes second order central difference scheme, whereas right figure utilizes fourth order central difference scheme. Solid lines represent the numeric solution while dashed lines represent the analytic solution.

Figure 3 shows the effects of extending the left boundary from -20 to -100 (note that the plots only show 0 ≤ x ≤ 450). For the fourth order scheme, extending the left boundary has resulted in inhibiting the rise previously seen at the left boundary. However, for the sixth order scheme, the

erroneous behavior remains, though it is delayed. It was found that in order to prevent the left boundary rise from occurring in the highest order schemes, it was necessary to extend the left boundary to -400. As doing so resulted in a nontrivial increase in computational cost, all subsequent computations in this paper were made using suppressed boundaries, as opposed to extended boundaries.

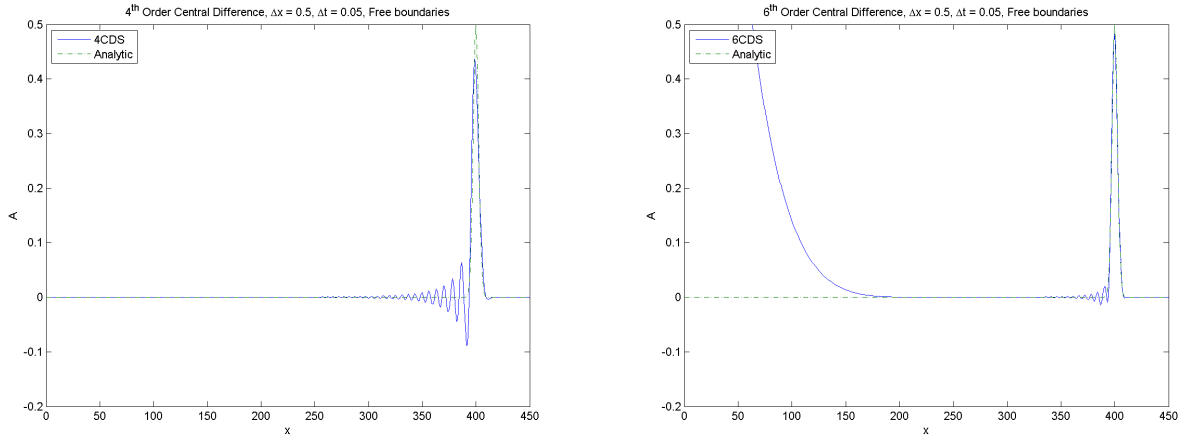

Figure 3: Effects of boundary conditions: extended boundary. Computational domain is -100 ≤ x ≤ 450. Left figure utilizes second order central difference scheme, whereas right figure utilizes fourth order central difference scheme. Solid lines represent the numeric solution while dashed lines represent the analytic solution.

Grid Resolution

The results of the simulation have been plotted in Figure 4 for two different grid resolutions and two different order central differencing schemes. From these plots, it is apparent that the accuracy of the numeric solution in the near wave region is highly dependent on the spatial and temporal step size, as one would expect. In addition to reducing the phase and amplitude inaccuracy between the analytic wave and the numeric wave, reducing the step sizes damped the spurious oscillations on the upstream side of the wave. Clearly, the step sizes used in a simulation should be determined by the order of accuracy of the differencing equations. For the second order case with the coarse grid, by the end of the simulation the initial wave is nearly indistinguishable from the spurious oscillations that accompany it; obviously, for this case a finer resolution grid is necessary. However, for the fourth order optimized scheme, increasing the grid resolution does little to improve the accuracy of the simulation, as the simulation is already fairly accurate on the coarse grid. For this case, the increased computational cost associated with the higher resolution grid may out weight the benefits of the increased accuracy. Note that the fourth order optimized scheme better approximates the analytic solution than the second order scheme, even when the second order scheme utilizes a step size one half that of the fourth order optimized scheme; thus is the power of higher order optimized schemes.

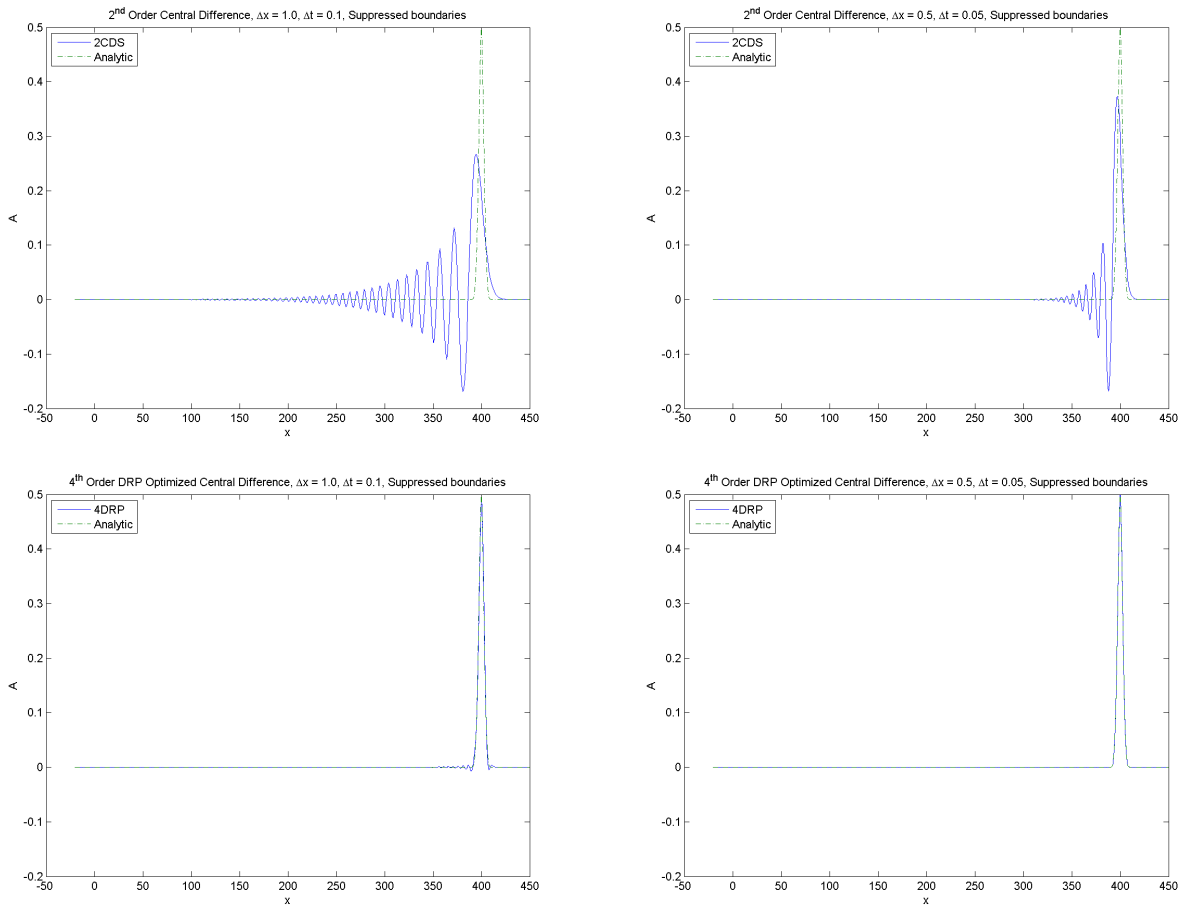

Figure 4: Effects of grid resolution. Left column uses dx of 1.0 and dt of 0.1, right column uses dx of 0.5 and dt of 0.05. Top row utilizes second order scheme, bottom row utilizes Tam's fourth order optimized scheme. Solid lines represent the numeric solution while dashed lines represent the analytic solution.

Results

Gaussian Wave

The following plots were computed to simulate the propagation of a Gaussian wave; the wave was initially at x = 0, and the propagation was simulated for 400 seconds. Suppressed boundary conditions were utilized on the left boundary, and the coarse grid was used on a domain of $-20 \leq x \leq 450$.

Figure 5 shows the simulation results using the second order central finite difference scheme. As was previously mentioned, the initial wave is practically indistinguishable from the spurious oscillations, and significant dispersion and dissipation errors are present between the numeric and analytic solution. The greatest error in amplitude is observed at the locations of greatest gradient in the analytic solution. As the central difference schemes do not result in dissipative errors, the error in amplitude is likely due to a transfer of energy from the initial wave to the spurious oscillations.

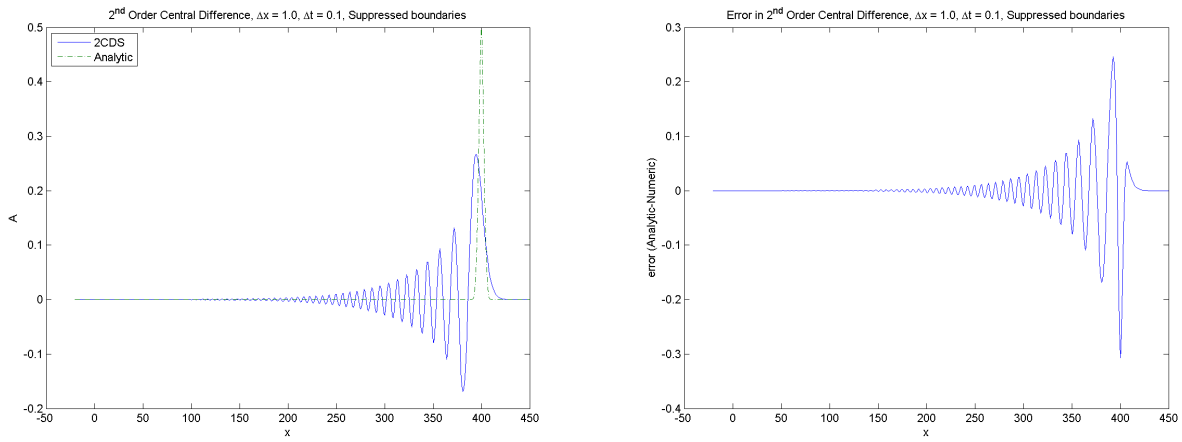

Figure 5: Results for second order central difference scheme solution to Gaussian wave. Figure on left shows numeric versus analytic solution; figure on right shows error between the solutions (analytic-numeric).

The results for the fourth order central difference scheme are shown in Figure 6. Compared to the second order scheme, the fourth order scheme results in far less significant dispersive and dissipative errors, however they are still non-negligible. The amplitudes of the spurious oscillations have been greatly diminished, while the frequency has doubled.

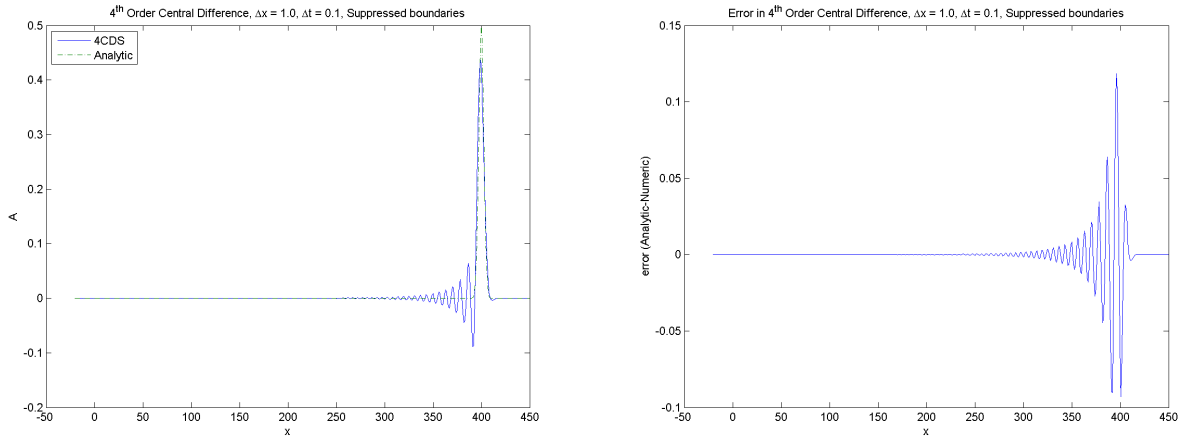

Figure 6: Results for fourth order central difference scheme solution to Gaussian wave. Figure on left shows numeric versus analytic solution; figure on right shows error between the solutions (analytic-numeric).

The results for the sixth order central difference scheme are shown in Figure 7. Again, increasing the order of accuracy has resulted in lower dissipative and dispersive errors, as well as lower amplitude, higher frequency spurious oscillations.

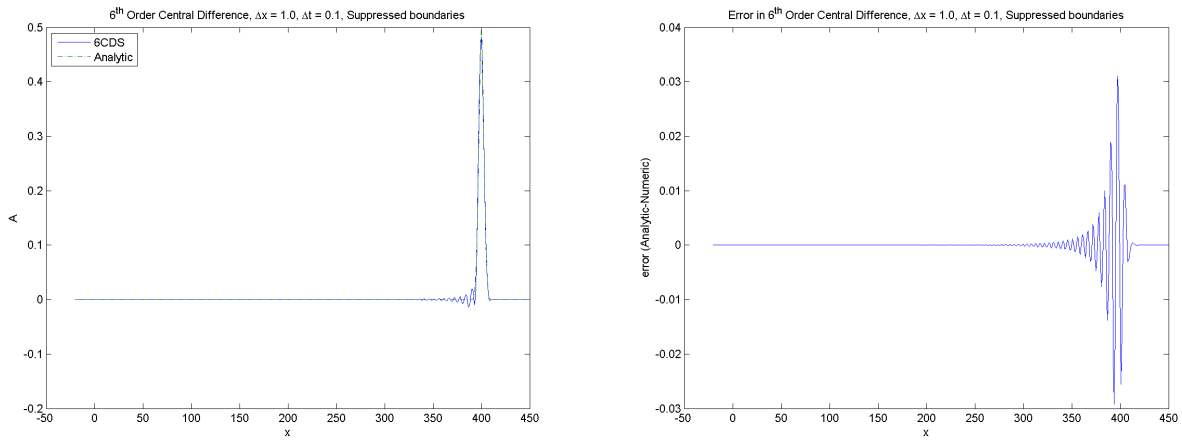

Figure 7: Results for sixth order central difference scheme solution to Gaussian wave. Figure on left shows numeric versus analytic solution; figure on right shows error between the solutions (analytic-numeric).

Figure 8 shows the results from Tam's fourth order dispersion-relation-preserving optimized scheme. Though this scheme utilizes a lower order of accuracy than the previous sixth order scheme, the optimization procedure has allowed it to better simulate the wave propagation. The dispersion error observed with the standard difference schemes is negligible, as is the dissipation error. Additionally, the spurious oscillations have been greatly damped over even the standard sixth order scheme. As this scheme utilizes the same number of terms as the standard sixth order scheme, the frequency of the spurious oscillations is the same.

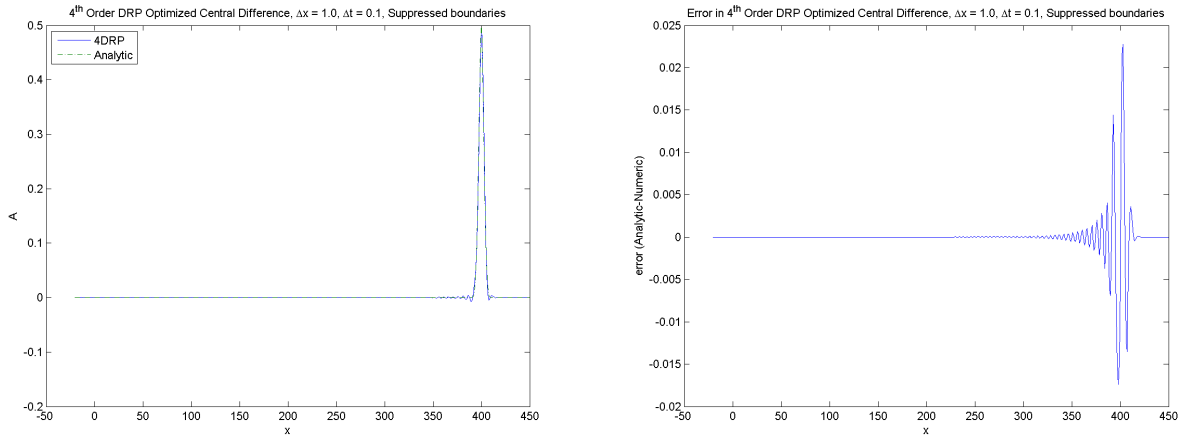

Figure 8: Results for fourth order DRP optimized central difference scheme solution to Gaussian wave. Figure on left shows numeric versus analytic solution; figure on right shows error between the solutions (analytic-numeric).

Square Wave

The following plots were computed to simulate the propagation of a square wave of wavelength 200; the wave was initially at x = 0, and the propagation was simulated for 400 seconds. Suppressed boundary conditions were utilized, and the coarse grid was used on a domain of $-100 \leq x \leq 300$.

Figure 9 shows the simulation results using the second order central finite difference scheme. High frequency, high amplitude noise is observed upstream of the square wave at the end of the

simulation. These oscillations are due to upstream propagating waves generated by the numerical scheme reflecting at the boundaries, as outflow boundary conditions have not been introduced. Organized oscillations with increasing amplitude occur at the wave's peak. Compared to the numeric solution to the Gaussian wave, the numeric solution to the square wave exhibits far greater error in dissipation and dispersion, due to the discontinuous spatial derivative. This is reinforced by the error plot, where the greatest error is observed to be located at the rising and falling edge of the wave, and is of much greater amplitude than that of the Gaussian wave error.

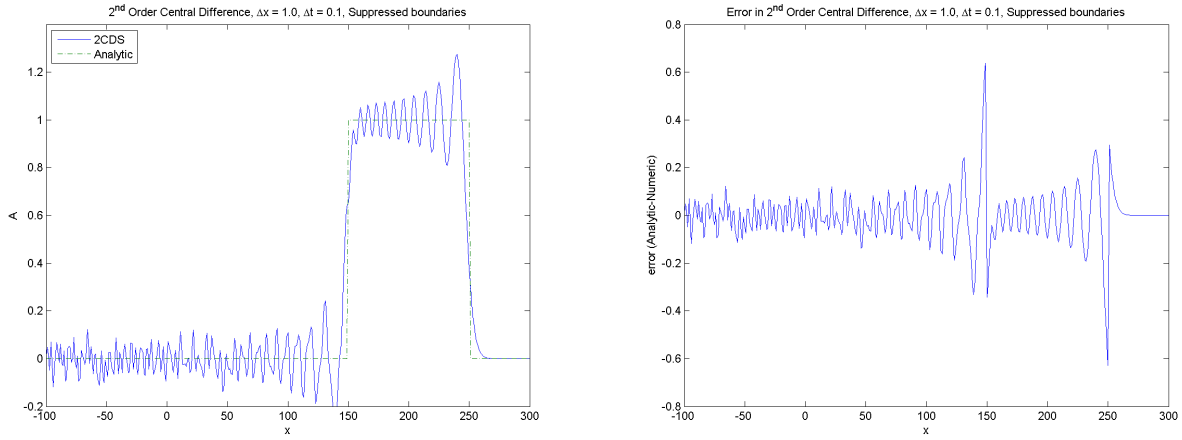

Figure 9: Results for second order central difference scheme solution to square wave. Figure on left shows numeric versus analytic solution; figure on right shows error between the solutions (analytic-numeric).

The results for the fourth order central difference scheme are shown in Figure 10. As expected, the amplitude of the noise upstream of the wave and the oscillations about the wave peak has been diminished by increasing the order of accuracy of the differencing equations. The frequency of the upstream noise remains unchanged from the second order difference case, while the frequency of the oscillations about the peak has increased.

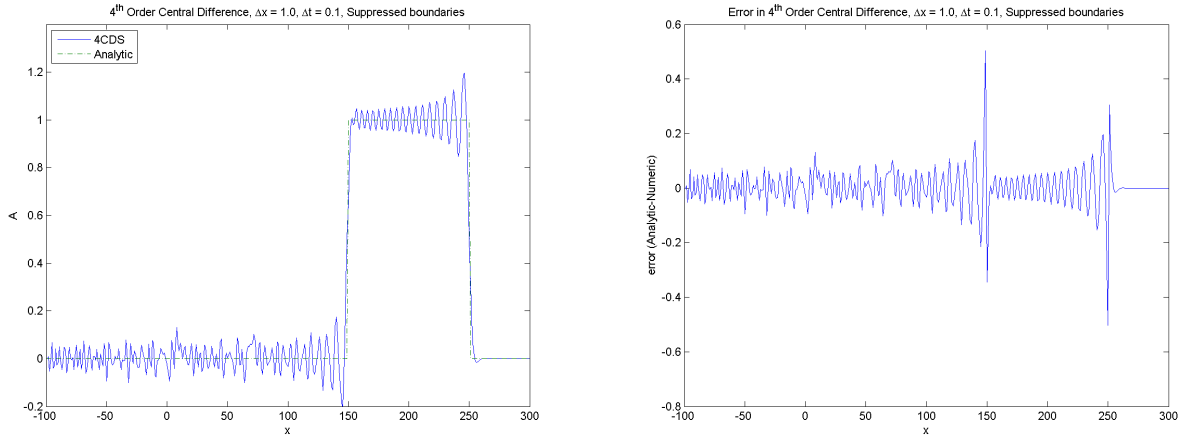

Figure 10: Results for fourth order central difference scheme solution to square wave. Figure on left shows numeric versus analytic solution; figure on right shows error between the solutions (analytic-numeric).

Figure 11 shows the results for the sixth order central difference scheme. Similar trends in terms of noise amplitude, oscillation frequency and amplitude, and dispersion errors, are observed when increasing the differencing order of accuracy from fourth to sixth as from increasing from second to fourth.

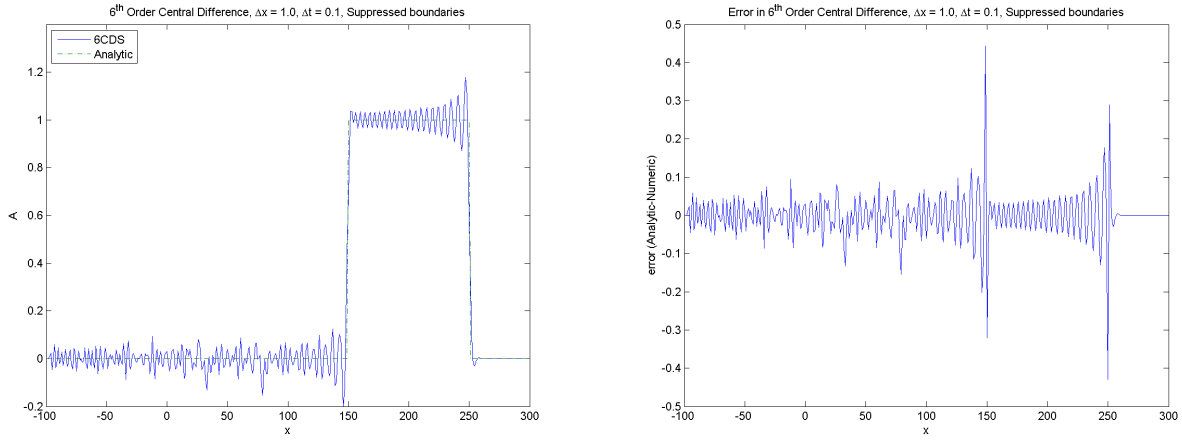

Figure 11: Results for sixth order central difference scheme solution to square wave. Figure on left shows numeric versus analytic solution; figure on right shows error between the solutions (analytic-numeric).

The results for the fourth order DRP optimized differencing scheme are shown for the square wave in Figure 12. Again, the fourth order dispersion-relation-preserving scheme results in better agreement with the analytical solution than even the standard sixth order scheme; the amplitude of the upstream noise and the oscillations about the wave peak has been decreased. Surprisingly, however, the fourth order optimized scheme exhibits greater overshoot at the rising and falling edge of the wave than the standard differencing schemes. The origin of this error is not entirely clear.

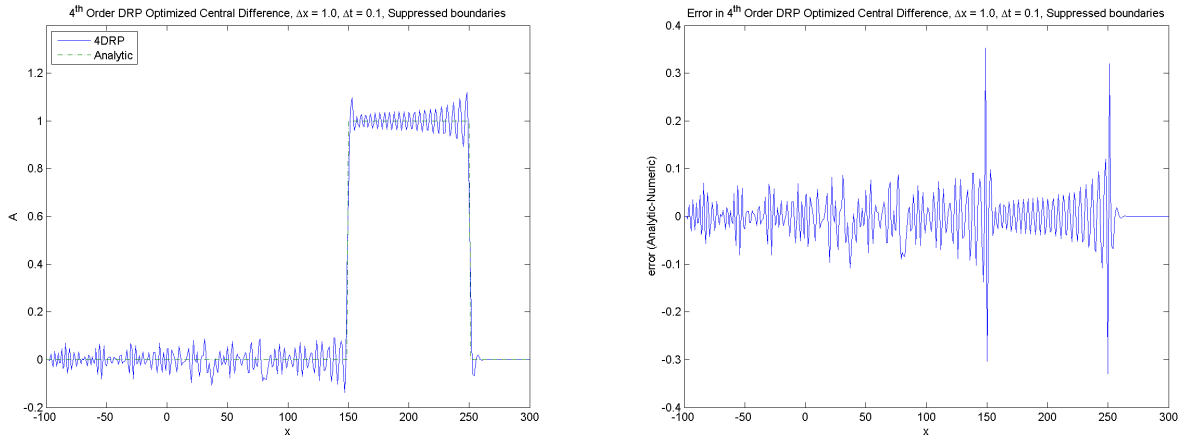

Figure 12: Results for fourth order DRP optimized central difference scheme solution to square wave. Figure on left shows numeric versus analytic solution; figure on right shows error between the solutions (analytic-numeric).

Conclusions

In this study, the propagation of Gaussian and square waves has been studied in one dimension. Numerical discretization of the spatial derivative has been accomplish via second, fourth and sixth order differencing schemes, as well as Tam's fourth order dispersion-relation-preserving optimized scheme. The temporal derivative has been discretized using an explicit third order optimized scheme. The spatial derivative at the left bound was suppressed throughout the simulations, in order to prevent the amplification of numerical errors at the boundary, as well as to reduce computational costs.

Simulations of a Gaussian wave found that the central differencing schemes employed in this study resulted in spurious oscillations just upstream of the initial wave which grew in amplitude as it propagated downstream. These oscillations resulted in a dissipation error in the wave, in addition to the dispersion error inherent in central differencing schemes. Utilizing higher order schemes, particularly Tam's fourth order dispersion-relation-preserving scheme resulting in significantly lower error levels.

Simulations of a square wave demonstrated the importance of non-reflective boundary conditions. As no special care was made for the boundary conditions in this study, the large numerical errors due to the discontinuous spatial derivative of a square wave produced upstream travelling waves which reflected off of the boundary and resulted in high frequency, high amplitude noise on the upstream side of the wave. Additionally, the weakness of central differencing schemes for simulating square waves was obvious: the numeric solution produced large amplitude oscillations about the wave peak. While increasing the order of the differencing scheme decreased the amplitude of these oscillations, higher order schemes produced increased overshoot at the rising and falling edges of the square wave.

Appendix

Appendix A: Main Program

```
% Project 1
% Completed by Michael Crawley for AAE 694
clear all;

%3rd order optimized time discretization coefficients (b0, b1, b2, b3)
b = [2.3025580883830 -2.4910075998482 1.5743409331815 -
0.38589142217162];
dx = 1.0;
dt = 0.1;

%Case 1
x1 = -20:dx:450;
t1 = [0 0 0 0:dt:400];

%Initialize solutions
u2CDS1 = zeros(length(t1),length(x1)); %2nd Order Central Difference
u2CDS1(1,:) = 0.5*exp(-log(2)*(x1/3).^2);
u2CDS1(2,:) = u2CDS1(1,:);
u2CDS1(3,:) = u2CDS1(1,:);
u2CDS1(4,:) = u2CDS1(1,:);
u4CDS1 = u2CDS1; %4th Order Central Difference
u6CDS1 = u2CDS1; %6th Order Central Difference
u4DRP1 = u2CDS1; %4th Order DRP
ua1 = 0.5*exp(-log(2)*((x1-400)/3).^2); %analytic solution

for j = 4:length(t1)-1
    u2CDS1(j+1,:) = u2CDS1(j,:)-
dt*b*NumericalDerivative(1,2,dx,u2CDS1(j:-1:j-3,:));
    u2CDS1(j+1,1) = 0;   %suppress left boundary fluctuation

    u4CDS1(j+1,:) = u4CDS1(j,:)-
dt*b*NumericalDerivative(1,4,dx,u4CDS1(j:-1:j-3,:));
    u4CDS1(j+1,1:2) = 0;   %suppress left boundary fluctuation

    u6CDS1(j+1,:) = u6CDS1(j,:)-
dt*b*NumericalDerivative(1,6,dx,u6CDS1(j:-1:j-3,:));
    u6CDS1(j+1,1:3) = 0;   %suppress left boundary fluctuation

    u4DRP1(j+1,:) = u4DRP1(j,:)-
dt*b*FakeNumericalDerivative(dx,u4DRP1(j:-1:j-3,:));
    u4DRP1(j+1,1:3) = 0;   %suppress left boundary fluctuation
end

%calc rms error
u2CDS1error = sqrt(sum((ua1-u2CDS1(end,:)).^2));
u4CDS1error = sqrt(sum((ua1-u4CDS1(end,:)).^2));
u6CDS1error = sqrt(sum((ua1-u6CDS1(end,:)).^2));
u4DRP1error = sqrt(sum((ua1-u4DRP1(end,:)).^2));
```

```
%Case 2
x2 = -100:dx:300;
t2 = [0 0 0 0:dt:200];

u2CDS2 = zeros(length(t2),length(x2)); %2nd Order Central Difference
u2CDS2(1,:) = x2>=-50 & x2<=50;
u2CDS2(2,:) = u2CDS2(1,:);
u2CDS2(3,:) = u2CDS2(1,:);
u2CDS2(4,:) = u2CDS2(1,:);
u4CDS2 = u2CDS2; %4th Order Central Difference
u6CDS2 = u2CDS2; %6th Order Central Difference
u4DRP2 = u2CDS2; %4th Order DRP
ua2 = double(x2>=150 & x2<=250); %analytic solution

for j = 4:length(t2)-1
    u2CDS2(j+1,:) = u2CDS2(j,:)-
dt*b*NumericalDerivative(1,2,dx,u2CDS2(j:-1:j-3,:));
    u2CDS2(j+1,1) = 0;   %suppress left boundary fluctuation

    u4CDS2(j+1,:) = u4CDS2(j,:)-
dt*b*NumericalDerivative(1,4,dx,u4CDS2(j:-1:j-3,:));
    u4CDS2(j+1,1:2) = 0;   %suppress left boundary fluctuation

    u6CDS2(j+1,:) = u6CDS2(j,:)-
dt*b*NumericalDerivative(1,6,dx,u6CDS2(j:-1:j-3,:));
    u6CDS2(j+1,1:3) = 0;   %suppress left boundary fluctuation

    u4DRP2(j+1,:) = u4DRP2(j,:)-
dt*b*FakeNumericalDerivative(dx,u4DRP2(j:-1:j-3,:));
    u4DRP2(j+1,1:3) = 0;   %suppress left boundary fluctuation
end

%calc rms error
u2CDS2error = sqrt(sum((ua2-u2CDS2(end,:)).^2));
u4CDS2error = sqrt(sum((ua2-u4CDS2(end,:)).^2));
u6CDS2error = sqrt(sum((ua2-u6CDS2(end,:)).^2));
u4DRP2error = sqrt(sum((ua2-u4DRP2(end,:)).^2));
```

Appendix B: Numerical Derivative Routine

```
function [deriv] = NumericalDerivative(Dorder, Horder, h,
phi,varargin)
    %Numerically derivate matrix given constant step size
    %Code Version: 3.0 @ 2011-02-14
    %Inputs:
    %   Dorder: Derivative Order
    %   Horder: Number of terms to use
    %   h: step size
    %   phi: function to derivate
    %   options:    '-center' for central difference only
    %               '-upstream' for backward difference only
    %               '-downstream' for forward difference only
    %Outputs:
    %   deriv: Dorder derivative of phi with Horder accuracy

    [M N] = size(phi);
    deriv = zeros(M,N);

    %set default scheme (if not given by user)
    if isempty(varargin)
       varargin{1} = '-center';
    end

    %perform differencing operations
    if strcmp(varargin,'-center')
        if mod(Horder,2)
            error('Given accuracy order cannot be accomplished with
central finite difference method; please provide even accuracy
order');
        end
        %central differencing
        coefs = TSE(Horder/2+floor((Dorder-
1)/2),Horder/2+floor((Dorder-1)/2),h,Dorder);
        for i = 1+Horder/2+floor((Dorder-1)/2):N-
Horder/2+floor((Dorder-1)/2)
            deriv(:,i) = phi(:,i-Horder/2+floor((Dorder-
1)/2):i+Horder/2+floor((Dorder-1)/2))*coefs;
        end

        %left bound
        for i = 1:Horder/2+floor((Dorder-1)/2)
           coefs = TSE(i-1,Horder+Dorder-i,h,Dorder);
           deriv(:,i) = phi(:,1:Horder+Dorder)*coefs;
        end

        %right bound
        for i = N-Horder/2+floor((Dorder-1)/2)+1:N
           coefs = TSE(-N+Horder+Dorder-1+i,N-i,h,Dorder);
           deriv(:,i) = phi(:,N-Horder-Dorder+1:end)*coefs;
        end
```

```
    elseif strcmp(varargin,'-upstream');
        %Upstream differencing
        coefs = TSE(Horder+Dorder-1,0,h,Dorder);
        for i = Horder+Dorder:N
          deriv(:,i) = phi(:,i-Horder-Dorder+1:i)*coefs;
        end

        %Left bound
        for i = 1:Horder+Dorder-1
           coefs = TSE(i-1,Horder+Dorder-i,h,Dorder);
           deriv(:,i) = phi(:,1:Horder+Dorder)*coefs;
        end
    elseif strcmp(varargin,'-downstream');
        %Downstream differencing
        coefs = TSE(0,Horder+Dorder-1,h,Dorder);
        for i = 1:N-Horder-Dorder+1
          deriv(:,i) = phi(:,i:i+Horder+Dorder-1)*coefs;
        end

        %Right bound
        for i = N-Horder-Dorder:N
           coefs = TSE(-N+Horder+Dorder-1+i,N-i,h,Dorder);
           deriv(:,i) = phi(:,N-Horder-Dorder+1:end)*coefs;
        end
    end
end
```

Appendix C: Taylor Series Expansion Coefficients Routine

```
function [coefs] = TSE(n,m,h,deriv)
    %Calculate coefficients for use in finite difference scheme by
Taylor series expansions.
    %Code Version: 1.0 @ 2011-02-14
    %Inputs:
    %   n: number of backwards terms
    %   m: number of forwards terms (stencil goes from -n:m
    %   h: step size
    %   deriv: derivative to be approximated (1, 2, ...)

    l = n+m+1;
    A = zeros(l,l);
    F = zeros(l,1);
    for i = 0:l-1
        A(:,i+1) = ((-n+i)*h).^(0:l-1);
    end
    F(deriv+1) = factorial(deriv);
    coefs = (A\F);
end
```

Appendix D: DRP Numerical Derivative Routine

```
function [deriv] = FakeNumericalDerivative(h, phi)
    %Numerically derivate matrix given constant step size, uses 4th
order
    %DRP scheme for first derivative only
    %Inputs:
    %   h: step size
    %   phi: function to derivate
    %Outputs:
    %   deriv: Dorder derivative of phi with Horder accuracy

    [M N] = size(phi);
    deriv = zeros(M,N);

    %perform central differencing operations
    coefs = FakeDRP(3,3,h);
    for i = 4:N-3
        deriv(:,i) = phi(:,i-3:i+3)*coefs;
    end

    %left bound
    for i = 1:3
       coefs = FakeDRP(i-1,7-i,h);
       deriv(:,i) = phi(:,1:7)*coefs;
    end

    %right bound
    for i = 1:3
       coefs = FakeDRP(3+i,3-i,h);
       deriv(:,i+N-3) = phi(:,N-6:N)*coefs;
    end
end
```

Appendix E: DRP Coefficients Routine

```
function [coefs] = FakeDRP(n,m,h)
%Determine coefficients for 4th order DRP scheme via lookup table
    a{33} = [-0.02084314277031176 ...
                0.166705904414580469 ...
                -0.77088238051822552 ...
                0 ...
                0.77088238051822552 ...
                -0.166705904414580469 ...
                0.02084314277031176]';

    a{42} = [0.02636943100 ...
                -0.16613853300 ...
                0.518484526d0 ...
                -1.27327473700 ...
                0.47476091400 ...
                0.46884035700 ...
                -0.049041958d0]';

    a{51} = [-0.048230454 ...
                0.281814650 ...
                -0.768949766 ...
                1.388928322 ...
                -2.147776050 ...
                1.084875676 ...
                0.209337622]';

    a{60} = [0.203876371 ...
                -1.128328861 ...
                2.833498741 ...
                -4.461567104 ...
                5.108851915 ...
                -4.748611401 ...
                2.192280339]';

    a{24} = -flipud(a{42});
    a{15} = -flipud(a{51});
    a{06} = -flipud(a{60});

    coefs = a{str2double(strcat([num2str(n),num2str(m)]))}/h;
end
```