Graphical Analysis and Animation in an Introductory Electrodynamics Course

Deborah M. Mechtel, Christopher T. Field, and Brian Jenkins

Department of Electrical Engineering
United States Naval Academy
105 Maryland Ave., Annapolis, MD 21402
Mechtel@nadn.navy.mil (email)
(410) 293 - 6165 (tel)
(410) 293 - 3493 (fax)

Abstract

Students frequently experience difficulty visualizing complex multidimensional electrodynamics topics, especially those with time variation. The classic stationary illustrations of inherently dynamic concepts can be greatly enhanced via computer animation. Animation significantly improves student comprehension of time dependent functions, and provides a secondary benefit for students by improving computer programming skills.

Introduction

Computer animation is being used in a junior level electrodynamics class at the United States Naval Academy (USNA), one of the more abstract courses in the undergraduate electrical engineering curriculum. Animation of computer generated graphics provides a powerful learning tool to help students visualize abstract concepts, which is especially useful in electrodynamics since key electromagnetic concepts involve time dependent multidimensional problems.[1] As an additional benefit, modeling this type of problem helps students improve their computer skills. The extensive student use of computers and industry standard software as engineering tools is a USNA electrical engineering department goal.

MATLAB, a popular technical programming tool with inherent graphical analysis features, is used throughout the electrical engineering curriculum at the USNA to (1) develop student proficiency in programming, (2) enhance the students' numerical modeling expertise, and (3) improve the students' comprehension of core subject matter. For the introductory electrodynamics class, a key to achieving a more rapid and complete student grasp of the technical concepts is MATLAB's "handle graphics" feature.[2] The "handle graphics" feature can simulate the evolution of multidimensional graphs over time. As a result, students are able to look at a rapid series of snapshots of a time dependent function as it develops, similar to the changing frames of a movie film.

In the USNA electrodynamics course, the lecture topics are visually supported by computer modeling tasks that may be implemented either as student assignments or as instructor demonstrations. This paper describes three sample lecture demonstrations on wave motion that are tailored to clarify ideas students typically find difficult. The examples include one and three dimensional traveling waves and one dimensional standing waves. These topics may also be implemented as simple student exercises where the equations describing the physical models are graphically displayed and, most importantly, animated.

One and Three Dimensional Traveling Sine Waves

The concept of a transverse constant amplitude traveling wave is a key topic for any introductory electrodynamics course. One common student misconception is that the wave not only propagates through space, but also continuously changes amplitude instead of remaining undistorted. Students are confused about the meaning of the spatial and time variables. The majority of students struggle to distinguish between the time varying wave amplitude at one position and the spatial functionality of the traveling wave at one point in time. The word "one dimensional" in this context confuses students since it refers to position only, ignoring the additional dimension of time.

To aid student understanding, the first demonstration animates a one-dimensional traveling wave. The one dimensional traveling wave is functionally described by:

$$f(x, t) = A \cos(\omega t - kx)$$

where:

A = Amplitude
ω = Angular frequency
k = Propagation constant
x = Position
t = Time

For the animations, A = 1, $\omega = 2\pi / T$ where T is the period of the wave, and $k = 2\pi / \lambda$ where $\lambda = 20$ is the wavelength.

For the purpose of this static paper, the authors attempt to capture the interesting aspects of the dynamically evolving animation by showing time distributed events from the animation. The demonstration begins by illustrating the time dependent amplitude of a sinusoidal wave at a unique spatial position. Figures 1a-c show the changing position of a box as an illustration of the time varying amplitude of $f(x, t)$ at a fixed spatial position, x=0. After displaying the moving box for two cycles, the animation generates the traveling wave that is defined by the changing amplitude denoted by the moving box at this unique spatial position. The second part of this animation (Figures 1d-g) shows the resulting sinusoidal wave progression as it travels away from the moving box. The wave moves in the positive x direction. Figures 1d-g depict snapshots of the traveling wave along the x axis at four unique points in time. The demonstration program uses the "handle graphics" feature of MATLAB, and is listed and documented in Appendix A. The program is written as a MATLAB function. It creates a three dimensional wave but when the argument is "0" it displays a one dimensional graph by viewing along the y axis. While running, clicking the mouse within the area bounded by the axis will freeze the display. Pressing any key will resume animation.

The previous animation depicted a one dimensional traveling wave both spatially and temporally. Typically, students are even more challenged by topics requiring visualization of time varying three dimensional objects. It is important for students to be comfortable with the idea of a transverse traveling wave in both one and three dimensions, since this is the mathematical foundation of classical electromagnetic wave propagation.

Students, newly introduced to one dimensional traveling waves, are again aided by extending that concept to three dimensions in an animated graphical illustration. The second demonstration introduces the concept of a traveling scalar plane wave by changing the view of the previous animation (Appendix A). The scalar plane wave progressing in the positive *x* direction is described by the following equation:

$$f(x, y, z, t) = A(x, y, z) \cos(\omega t - kx)$$

where rectangular coordinates *x*, *y*, and *z* are used and the other variables are as previously defined.

Similar to the one dimensional traveling wave demonstration, a three dimensional "bouncing box" is used to show sine wave motion at stationary spatial positions. A three dimensional traveling wave originating at the moving box is generated as shown in the view in Figure 2a. Next, colored planes are superimposed on the three dimensional traveling wave (Figure 2b) that illustrate planes of constant amplitude and phase in three dimensional space. The program is listed and documented in Appendix A. Setting the function argument to "1" generates a three dimensional plot while setting it to "2" generates a plot which transitions from one to three dimensions.

One Dimensional Standing Waves

In addition, electrical engineering students often find the concept of a standing wave counterintuitive. The idea that two traveling waves can interfere and create a standing wave with null points is difficult to grasp. The one-dimensional standing wave illustrated in this lecture demonstration is based on the following equations where

$$f_1(x,t) = A \cos(\omega t + kx)$$

represents a wave traveling in the negative *x* direction,

$$f_2(x,t) = A \cos(\omega t - kx)$$

represents a wave traveling in the positive *x* direction, and

$$\begin{aligned} f_3(x,t) &= f_1(x,t) + f_2(x,t) \\ &= A \cos(\omega t + kx) + A \cos(\omega t - kx) \\ &= 2 A \cos(kx) \cos(\omega t) \end{aligned}$$

describes a standing wave, where all variables are as previously defined.

The demonstration begins with two differently colored sinusoidal waves traveling in the positive and negative x directions respectively (Figure 3a). In Figures 3b-d the waves interfere and the mathematical model of the standing wave is shown graphically via animation. The superimposed waves produce a differently colored standing wave with obvious null points. The peak amplitude and null points of the standing wave are emphasized with color blocks in Figures 3b-d. The animated graphics describing the standing wave provides a means for students to compare the $f_3(x,t)$ function with the results seen on the computer display. The animation can describe either the time variation of the standing wave in space, or the spatial variations of a standing wave in time. The program is listed and documented in Appendix B. Again, clicking the mouse within the graph area bounded by the axis will freeze the display and typing a key will resume it.

Summary

Animated computer graphics significantly improves student understanding of multidimensional electrodynamics concepts beyond that provided by static figures. Computer animation is key to supporting student understanding of time dependent multidimensional functions that describe physical phenomena. These MATLAB based instructor demonstrations address a class of dynamic problems difficult for students to grasp. Other possible topics include attenuated traveling waves, standing waves created by reflections from different impedances, reflection and transmission of waves at a load impedance and propagating vector plane waves. Many of these topics may also be implemented as student exercises to improve understanding of time dependent concepts while also enhancing computer skills.

This software is freely available for downloading via the www site http://wseweb.ew.usna.edu/ee/LINKS/EE_Links.htm (should the URL be changed, from the Naval Academy home page, select Academics, Academic Divisions and Departments, Electrical Engineering, Links).

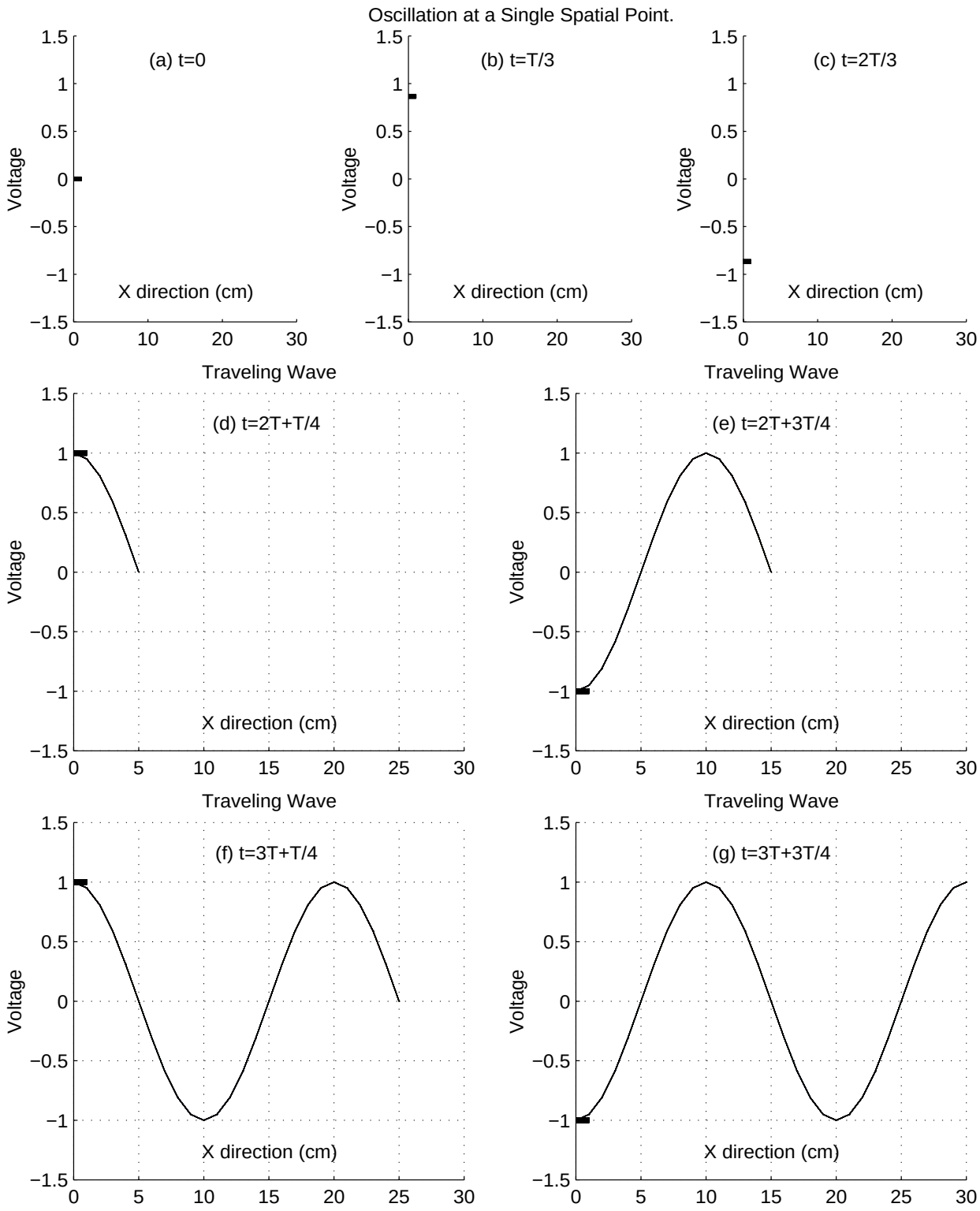

Figure 1. Time distributed events for a one dimensional traveling wave animation.

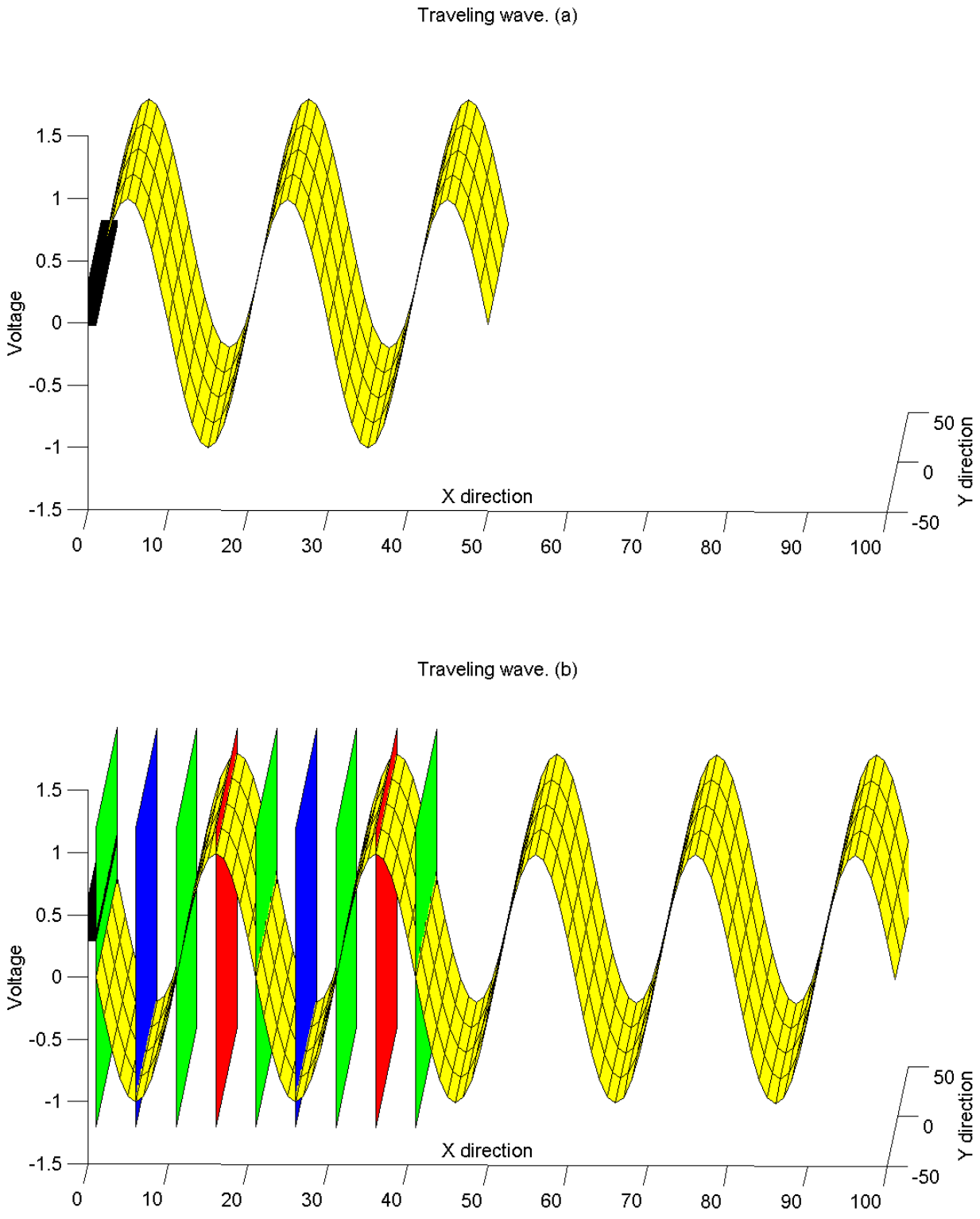

Figure 2. (a) Three dimenstional traveling wave (b) with constant amplitude and phase planes.

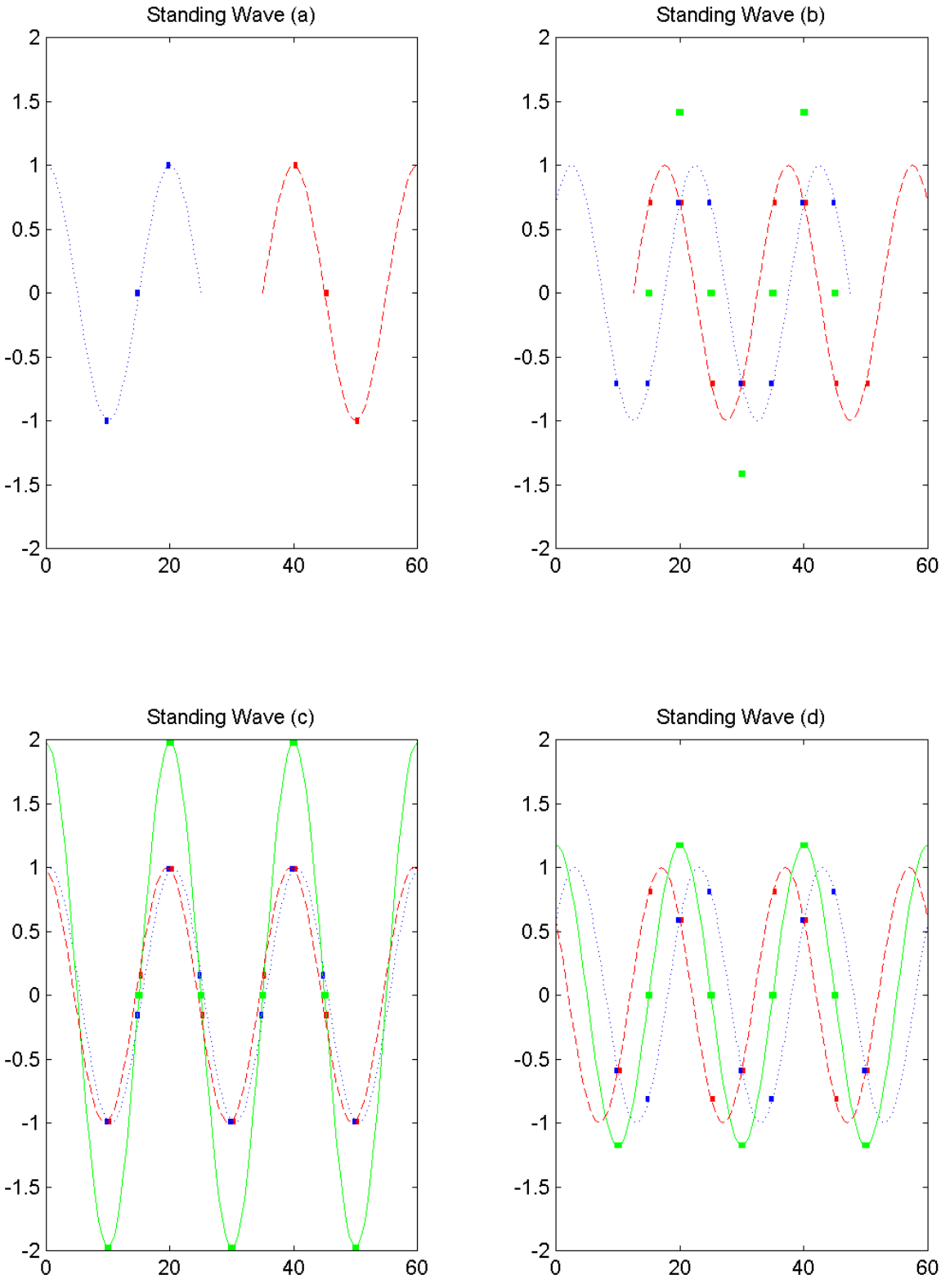

Figure 3. (a) +x and -x direction traveling sine waves.(b-d) Time evolution of the standing wave.

References

1. Jenkins. B, "Simulation and Animation in Optical Fiber Communication", *Computers in Education Journal* (accepted for publication)

2. MATLAB Users's Guide, The MathWorks, Inc., Natwick, MA 01760, 1992.

DEBORAH M. MECHTEL

Deborah M. Mechtel completed requirements for the Ph.D. at Johns Hopkins University in 1994. Dr. Mechtel is currently an assistant professor of Electrical Engineering at the United States Naval Academy, Annapolis, MD. Current research work focuses on electro-optics, high speed circuit testing and advanced packaging development.

CHRISTOPHER T. FIELD

CHRISTOPHER T. FIELD is currently an assistant professor of Electrical Engineering at the US Naval Academy. Previously he was a research scientist at The Johns Hopkins University working at NASA's Goddard Space Flight Center. He received a BS in mathematics and physics from Guilford College in 1980 and an MS and PhD in electrical engineering from The Johns Hopkins University in 1987 and 1993.

BRIAN JENKINS

BRIAN JENKINS received the B.S. and M.S. degrees from The Ohio State University in 1983 and 1991 and the Ph.D degree from the University of Colorado in 1995. From 1983-1989 he worked as a development engineer for IBM Corporation in Endicott, NY. Since 1996, he has been an assistant professor in the Department of Electrical Engineering at the U.S. Naval Academy where he teaches and does research in fiber optic communications.

Appendix A

```
function TravelWave( viewpoint )
%TravelWave.m  displays 1D and 3D traveling waves.
% TravelWave( viewpoint ) animates the motion of 1D and 3D traveling waves.
% First a bouncing box is displayed on the left vertical axis.
% The box undergoes simple harmonic vertical motion.
% After several cycles of the sine wave it starts to generate
% a sine wave which propagates to the right from the box.
% After the wave has moved some distance the view gradually
% changes so that it can be seen that the 1D image has been the
% end view of a 2D plane wave.
% Finally, planes are drawn on the wave to indicate planes of constant phase.
% These planes travel with the sine wave.
%
% The function may be called with one argument, VIEWPOINT.
% If viewpoint = 0, then a 1D display is generated.
% If viewpoint = 1, then a 3D display is generated with a single viewpoint
% If viewpoint = 2, then the display rotates from a 1D to a 3D display
% If called with no arguments, then viewpoint = 0 is assumed.
% Clicking the mouse within the plot area will pause the program. Pressing
% any key will resume it.

% Because this program does both 1D and 3D and transitions between them
% it is more complex than is required to animate a 1D wave.

% This script by Christopher T. Field. USNA
% V1.4    2 March 99.

% If no input arguments, then set viewpoint = 0
if nargin < 1, viewpoint = 0; end

% Define the limits and parameters of the display
Xmin = 0;      Xmax = 100;    % Note that the x axis is labeled z axis.
Ymin = -50;    Ymax = +50;
Zmin = -1.5;   Zmax = +1.5;   % z axis is labeled "voltage"
P = 20;                       % Period in units of X, generally called lambda
Ncycles = 5;                  % Number of cycles of the wave to show on screen
NP = 9;                       % Number of planes to display on the surface.
                              % NP is reset to 0 if viewpoint = 0
% Now start doing things.
clf                           % Clear the current figure of any old images.
figure(gcf)                   % Bring the current figure to the top.
set(gcf, 'Renderer', 'zbuffer' ) % Needed to prevent flicker

colormap( [1, 1, 0] )         % Wave surface is yellow [r g b] = [1 1 0].

if viewpoint == 0,            % Draw no planes on a 1D only plot.
   NP = 0;
end

% The animation is created by placing on the plot one large
% sine wave surface and then changing the x values for the surface.
```

```
% This means that the equations of a traveling wave do not appear in
% the program. A student program would likely use the original equations.

% Create the x and y values for the traveling wave
x2=-Ncycles*P:0;                    % All of the x values start as negative.
y2=linspace(Ymin, Ymax,5 );         % Because this will plot a 2D surface need
                                    % y values.
z2=zeros(length(y2), length(x2) );  % Presize the matrix for the values.

% The matrix 'z2' is indexed z2(y,x).
% For a given x value, the surface has the same height so the same value
% gets stored in all of the values of the first subscript.
for i=1:length(x2),
   z2(:,i) = -sin(2*pi*x2(i)/P);
end;

% The next few lines setup the axis viewpoint. Until the view command
v1 = 0;                                     % there is no axis on the figure
v2 = 0;
if viewpoint == 1,
   v1 = 1.5;
   v2 = 15;
end
view( v1, v2 )                              % Set the desired view position.
axis( [Xmin, Xmax, Ymin, Ymax, Zmin, Zmax])  % Set axis limits and pause if
set(gca, 'ButtonDownFcn', 'pause on; pause' )% mouse is clicked within axis.
hold on

% This part will draw a parallelogram on the left axis which indicates
% the function value at that point.
% ver and fac are the numbers needed by the patch command
% to draw a solid box at the origin.
ver =[-1 Ymin -0.02;    % This array locates each vertex of the box
       1 Ymin -0.02;    % in the form (x y z)
       1 Ymax -0.02;
      -1 Ymax -0.02;
      -1 Ymin  0.02;
       1 Ymin  0.02;
       1 Ymax  0.02;
      -1 Ymax  0.02];
fac = [1 2 6 5;         % This array indicates which vertices define
       2 3 7 6;         % each face of the box.
       3 4 8 7;
       4 1 5 8;
       1 2 3 4;
       5 6 7 8];
% verplane and facplane are the numbers needed to draw the planes
% of constant phase using the patch command.
verplane = ...
  [-3*P Ymin -1.2;
   -3*P Ymax -1.2;
   -3*P Ymax +1.2;
   -3*P Ymin +1.2];
facplane = [1 2 3 4];
```

```
% Draw the bouncing box at the origin
phand = patch('Vertices', ver, 'Faces', fac, 'FaceColor', 'Black' );
zz = get(phand, 'ZData');
zlabel( 'Voltage' )
XlabelHand = xlabel( 'X direction (cm)' );
YlabelHand = ylabel( 'Y direction (cm)' );
title( 'Oscillation at a Single Spatial Point.' )
set(XlabelHand, 'Position', [(Xmin+Xmax)/2 Ymin -1.3], ...
      'HorizontalAlignment', 'center' );
set(YlabelHand, 'Position', [ Xmax*1.07  0 -1.5], ...
      'HorizontalAlignment', 'center', 'Rotation', 90);

% Add the surface plot and phase planes even though they will not yet show.
% This is so there will be a smooth transition to the traveling wave.
shand = surf(x2,y2,z2);                   % Draw the surface and
set(shand, 'Visible', 'off' );            % turn it off so it does not show.

planecolor = str2mat( 'Green', 'Red', 'Green', 'Blue' );
for i = 0:(NP-1),
   vertplane = verplane;
   vertplane(:,1) = vertplane(:,1)-i*P/4;
   pphand(i+1) = patch('Vertices', vertplane, ...
      'Faces', facplane, 'FaceColor', planecolor( mod(i, 4)+1) );
end

% Bounce the box for 2 cycles. 50 updates per cycle
for t = 0:(P/50):2*P,
   vert = ver;
   vert(:,3) = vert(:,3) + sin(2*pi*t/P);
   set(phand, 'ZData', zz+sin(2*pi*t/P) );
   drawnow
end

title( 'Traveling Wave' )                 % Change the plot title.
set(shand, 'Visible', 'on' );             % Turn the surface on so it shows.

% Now start moving the wave in space.

for t=1:(P/20):(length(x2)-1),            % 20 updates per cycle
   set(shand, 'XData', x2+t );            % Move the sine wave
   vert = ver;                            % Move the bouncing box
   vert(:,3) = vert(:,3)+sin(2*pi*t/P);
   delete(phand)
   phand = patch('Vertices', vert, 'Faces', fac, 'FaceColor', 'Black' );

   % Move the constant phase planes by deleting and redrawing them.
   for i=0:(NP-1),
      delete(pphand(i+1));
      vertplane = verplane;
      vertplane(:,1) = vertplane(:,1)-i*P/4+t;
      pphand(i+1) = patch('Vertices', vertplane, ...
         'Faces', facplane, 'FaceColor', planecolor( mod(i, 4)+1) );
   end
```

```
% If more than one quarter the way through the wave movement then introduce
% the second dimension by shifting the viewpoint above the xy plane.
% The movement will continue as the view angle slowly increase to (2, 15)

   if (viewpoint == 2) & (t > length(x2)/4),
      if v1 < 2,
         v1 = v1+.02;
      end;
      if v2 < 15,
         v2 = v2+0.2;
      end;
   end
   view(v1, v2)         % Set the view angle
   set(XlabelHand, 'Position', [(Xmin+Xmax)/2 Ymin -1.3], ...
         'HorizontalAlignment', 'center' );
      set(YlabelHand, 'Position', [ Xmax*1.07  0 -1.5], ...
         'HorizontalAlignment', 'center', 'Rotation', 90);
   drawnow
end
```

Appendix B

```
% WaveSum.m will draw two traveling wave, one moving right
% the other left and show the result of adding them.
% First the waves are made to travel toward each other. A
% few spatial points are highlighted by having a box follow the
% wave height. When the two waves overlap, the sum of heights
% of selected points are plotted.
% Finally the standing wave sum of the two waves at all points is plotted.

% By Christopher T. Field USNA
% V1.2   11 Jan 99

lambda = 20;                  % Wavelength
c = 5;                        % Wave velocity is 5 space units/time unit
Ncycles = 3;                  % Number of cycles to display on axis

AxisXMax = Ncycles * lambda;  % x axis will go from zero to AxisXMax
Tmax = AxisXMax/c;            % Run long enough for waves to cross the axis
Tstop = 2*Tmax;

clf                                 % Clear the current figure
figure( gcf )                       % Bring current figure to the front
set(gcf, 'Renderer', 'zbuffer' )    % Needed to prevent flicker
axis( [0 AxisXMax -2 2] )           % Set axis limits and create axis
hold on
% Set to pause the animation if the mouse is clicked in the graph area.
set(gca, 'ButtonDownFcn', 'pause on; pause' )   % Pau if mouse is clicked.
% set(gcf, 'ButtonDownFcn', 'pause on; pause' ) % Pau if clicked outside axis.

% The traveling waves are actually drawn off the axis and move by changing
% the x values. In other words, the waves actually move as a sheet rather
% than being computed at each time step.
x1 = (-3 * AxisXMax ):0.5:0;          % x values for right moving wave
x2 = 0:0.5:AxisXMax;                  % x values for standing wave
x3 = AxisXMax:0.5:(4 * AxisXMax);     % x values for left moving wave

y = sin(2*pi*x1 / lambda );           % Define traveling wave y values
y2 = cos(2*pi*x2 / lambda );          % Define standing wave amplitudes

whand = plot( x1, -y, x2, y2, x3, y);  % Plot all three waves
set( whand(2), 'Visible', 'off' )      % Not ready to see the sum.

% These patches define the boxes which indicate the height of each wave at
% specific points. 'phand' contains the handles to each patch, one for
% the right traveling wave, the standing wave, and the left traveling wave.
xpatch  = [-1 0 0 -1] * 0.5;
xpatch2 = [-1 1 1 -1] * 0.5;
ypatch  = [-1 -1 1 1] * 0.02;
phand1 = [];                  % Must start with empty handles to the boxes
phand2 = [];
phand3 = [];
xpatchpos = [];
% Now draw each of the boxes. The limits of i define how many boxes are
```

```
% created. The boxes are separated by one quarter of a wavelength.
for i=-4:4,
   x = lambda*Ncycles/2 - i*lambda*(0.25);
   ph = patch( -xpatch + x, ypatch, 'r' );   % Red box
   set( ph, 'EdgeColor', 'r' );              % with red sides.
   set( ph, 'Visible', 'off' );              % Turn box off until later.
   phand1 = [phand1 ph ];                    % Add box handle to handle array

   ph = patch(  xpatch2+ x, ypatch, 'g' );   % Green box
   set( ph, 'EdgeColor', 'g' );              % with green sides.
   set( ph, 'Visible', 'off' );
   phand2 = [phand2 ph ];

   ph = patch(  xpatch + x, ypatch, 'b' );   % Blue box
   set( ph, 'EdgeColor', 'b' );              % with blue sides.
   set( ph, 'Visible', 'off' );
   phand3 = [phand3 ph ];
   xpatchpos = [ xpatchpos x ];
end
drawnow
delX = c;
                        % Motion is introduced by looping through time values.
for t=0:.1:Tstop,
   set(whand(1), 'XData', x1 + c*t );     % Update the right moving wave
                                          % Update the standing wave heights
   set(whand(2), 'YData', y2 * 2*sin(2*pi*t*c/lambda) );
   set(whand(3), 'XData', x3 - c*t );        % Update the left moving wave
   for i=1:length(xpatchpos),
      set(phand1(i), 'YData', ...            % Update the box locations
         ypatch + 1*sin(2*pi*(xpatchpos(i)/lambda + t*c/lambda) ));
      if xpatchpos(i) > (AxisXMax-c*t),      % If the wave has passed the box,
         set(phand1(i), 'Visible', 'on' );   % make it visible.
      end
      set(phand2(i), 'YData', ...
         ypatch + 2*cos(2*pi*xpatchpos(i)/lambda)*sin(2*pi*t*c/lambda) );
      if (xpatchpos(i) > (AxisXMax-c*t)) & (xpatchpos(i) < c*t),
         set(phand2(i), 'Visible', 'on' );
      end
      set(phand3(i), 'YData', ...
         ypatch - 1*sin(2*pi*(xpatchpos(i)/lambda - t*c/lambda) ));
      if xpatchpos(i) < c*t,
         set(phand3(i), 'Visible', 'on' );
      end
   end                                       % Box location updates
   if (c*t> AxisXMax) & (AxisXMax-c*t<0), % If both waves have crossed axis
      set(whand(2), 'Visible', 'on')      % make the standing wave visible
   end
   if (c*t >1.5 * AxisXMax),              % After some time of all three
      set(whand(1), 'Visible', 'off')     % turn off the traveling waves so
      set(whand(3), 'Visible', 'off')     % only the standing wave is visible
   end
   drawnow
end
```