Arrays and Plotting
(By Patrice Harapeti)

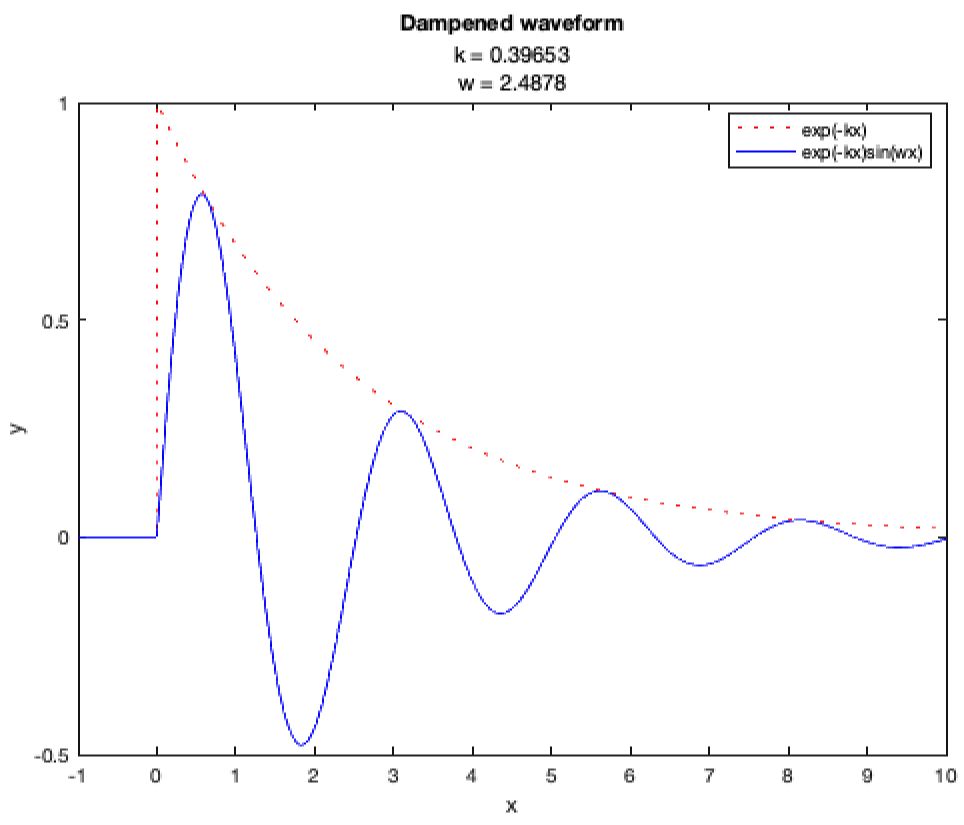
Table of Contents

Background

Demonstrate generating and plotting one and two dimensional arrays.

Part 1

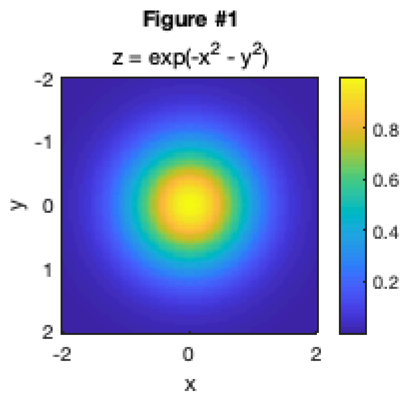
```
clc;
clear;

% Define parameters
k = 0.2 + rand(1) * 0.3;
w = 2 + rand(1) * 3;
x = linspace(-1, 10, 1000);

% Algorithm
y1 = nan(length(x));
y2 = nan(length(x));

y1 = exp(-k .* x);
y2 = exp(-k .* x) .* sin(w .* x);

y1(x <= 0) = 0;
y2(x <= 0) = 0;

% Plot both functions with distinct line-styles
figure(1)
plot(x, y1, "r:", x, y2, "b-");

% Decorate plot
title("Dampened waveform");
subtitle({ "k = " + k, "w = " + w});
xlabel("x");
ylabel("y");
legend('exp(-kx)', 'exp(-kx)sin(wx)')

% Restrict plot domain
xlim([-1, 10]);
```

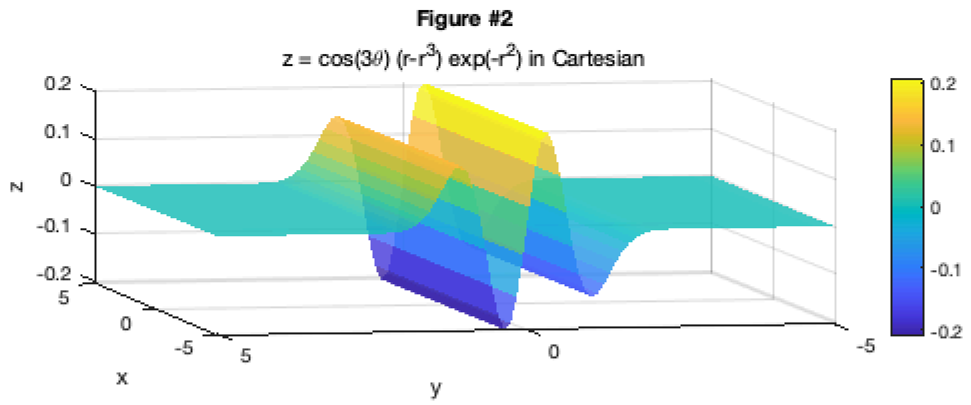


Part 2

```
clc;
clear;

% Broadcasting and top-down plot
x = linspace(-2, 2);
y = linspace(-2, 2).';
z = exp(-x.^2 - y.^2); % broadcasting

% Generate plot
figure(2)
subplot(2,1,1);
imagesc(x, y, z);

% Decorate Figure #1
colorbar
axis('image')
title('Figure #1')
subtitle('z = exp(-x^2 - y^2)')
xlabel('x')
ylabel('y')

% Generate domain in Cartesian coordinates
x2 = linspace(-5, 5);
```

```
y2 = linspace(-5, 5);

% Generate matrix of matching size to store z2 values (for rendering)
z2 = nan([length(x2), length(y2)]);

for i = 1:length(x2)
    for j = 1:length(y2)
        % Grab the corresponding x and y coordinates each loop
        xVal = x2(:, i);
        yVal = y2(:, i);

        % Convert the Cartesian coordinate into a Polar coordinate
        rhoVal = sqrt((xVal .^ 2) + (yVal .^ 2));
        thetaVal = atan2(yVal, xVal);

        % Use Polar coordinate as parameters to compute the given
 function
        % and store result of function into each row of matrix z2
        z2(i, j) = cos(3 .* thetaVal) .* (rhoVal - rhoVal .^ 3)...
                   .* exp(-rhoVal .^ 2);
    end
end

% Plot cartesian coordinates
subplot(2,1,2);
surf(x2, y2, z2, 'EdgeColor', 'none', 'FaceAlpha', 0.8);

% Decorate Figure #2
colorbar
title('Figure #2')
subtitle('z = cos(3\theta) (r-r^3) exp(-r^2) in Cartesian');
xlabel('x');
ylabel('y');
zlabel('z');

% Adjust camera line of sight for Figure #2
view([-15 3 4]);
```





Published with MATLAB® R2020b