MATH 3341: Introduction to Scientific Computing Lab

Melissa Butler

University of Wyoming

September 13, 2021

Lab 04: Plotting Data

Basic Plotting

Create a `figure` window

- `figure`: Creates a new figure window, and returns its handle.
  Example:

  ```
  figure
  ```

  or

  ```
  fig = figure
  ```

Create a `figure` window

- `figure`: Creates a new figure window, and returns its handle. Example:

  `figure`

  or

  `fig = figure`

- `figure(handleNumber)`: Makes `handleNumber` the current figure, forces it to become visible, and raises it above all other figures on the screen. If Figure `handleNumber` does not exist, and `handleNumber` is an integer, a new figure is created with handle `handleNumber`. Example:

  `figure(3)`

  or

  `fig = figure(3)`

Scatter plot

- `plot(x, y)`: Plot vector `y` versus vector `x`. Example 1:

```
x = linspace(0, 2 * pi, 100);
y = sin(x);
plot(x, y);
```

Scatter plot

- `plot(x, y)`: Plot vector `y` versus vector `x`. Example 1:

```
x = linspace(0, 2 * pi, 100);
y = sin(x);
plot(x, y);
```

- `plot(y)`: Plot vector `y` versus its index. Example 2:

```
x = linspace(0, 2 * pi, 100);
y = sin(x);
plot(y)  % same as plot(1:length(y), y);
```

Scatter plot

- `plot(x, y)`: Plot vector `y` versus vector `x`. Example 1:

```
x = linspace(0, 2 * pi, 100);
y = sin(x);
plot(x, y);
```

- `plot(y)`: Plot vector `y` versus its index. Example 2:

```
x = linspace(0, 2 * pi, 100);
y = sin(x);
plot(y)  % same as plot(1:length(y), y);
```

- `plot(x, y, style)`: Plot vector `y` versus vector `x` with specified style options in `style`. Example 3:

```
x = linspace(0, 2 * pi, 100);
y = sin(x);
style = 'go-.';
plot(x, y, style)  % same as plot(x, y, 'go-.');
```

Scatter plot: color, marker, and linetype

`style` is a character string made from one element from any or all the following 3 columns:

`b`	`blue`	`.`	`point`	`-`	`solid`
`g`	`green`	`o`	`circle`	`:`	`dotted`
`r`	`red`	`x`	`x-mark`	`-.`	`dashdot`
`c`	`cyan`	`+`	`plus`	`--`	`dashed`
`m`	`magenta`	`*`	`star`	`(none)`	`no line`
`y`	`yellow`	`s`	`square`		
`k`	`black`	`d`	`diamond`		
`w`	`white`	`v`	`triangle (down)`		
		`^`	`triangle (up)`		
		`<`	`triangle (left)`		
		`>`	`triangle (right)`		
		`p`	`pentagram`		
		`h`	`hexagram`		

THE UNIVERSITY OF WYOMING EQUALITY 1886

Scatter plot: Example 1

```
% Example: plot(x, y)
x = linspace(0, 2 * pi, 100);
y = sin(x);
figure(1);
plot(x, y);
```

Scatter plot: Example 1

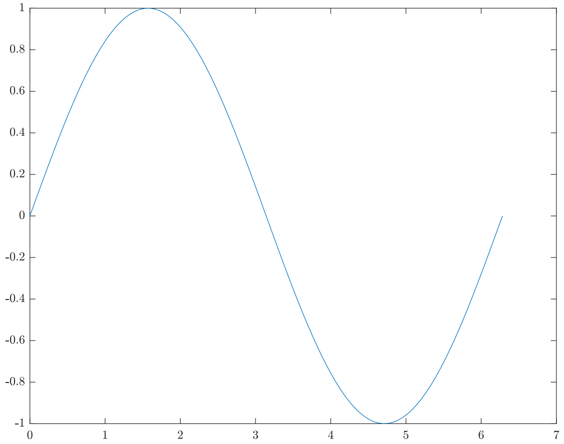

Figure 1:`plot(X,Y)`

Scatter plot: Example 2

```
% Example: plot(y)
x = linspace(0, 2 * pi, 100);
y = sin(x);
figure(2);
plot(y);
```

Scatter `plot`: Example 2

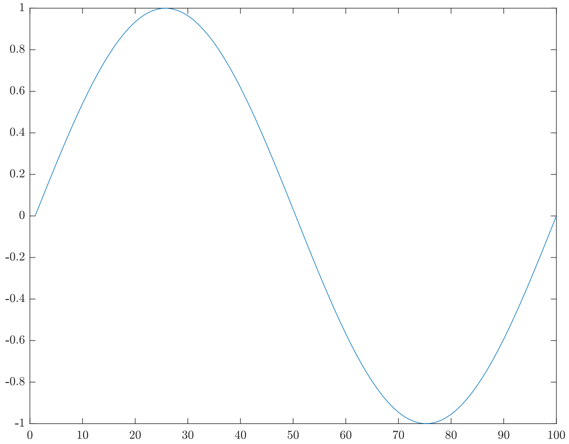

Figure 2:`plot(Y)`

Scatter plot: Example 3

```
% Example: plot(x, y, style)
x = linspace(0, 2 * pi, 100);
y = sin(x);
style1 = 'go-.';  % green, circle, dashdot
style2 = 'r+:';   % red, plus, dotted
style3 = 'm*--';  % magenta, star, dashed
figure(3);
plot(x, y, style1);
figure(4);
plot(x, y, style2);
figure(5);
plot(x, y, style3);
```

Scatter `plot`: Example 3

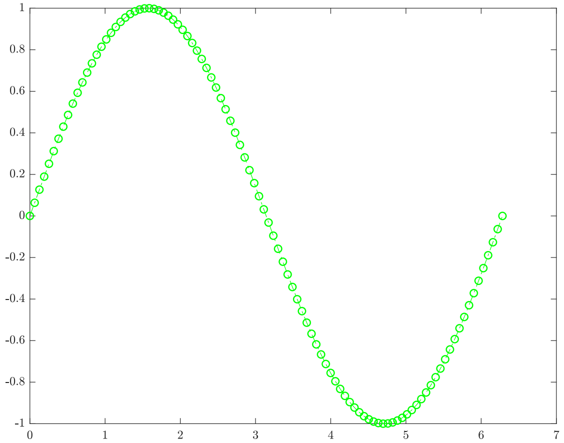

Figure 3:`plot(x, y, 'go-')`

Scatter `plot`: Example 3

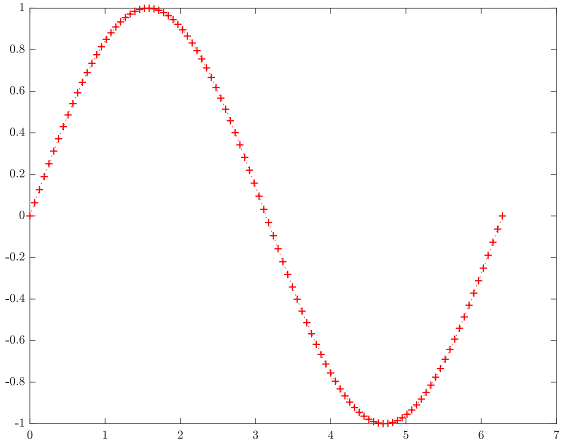

Figure 4:`plot(x, y, 'r+:')`

Scatter `plot`: Example 3

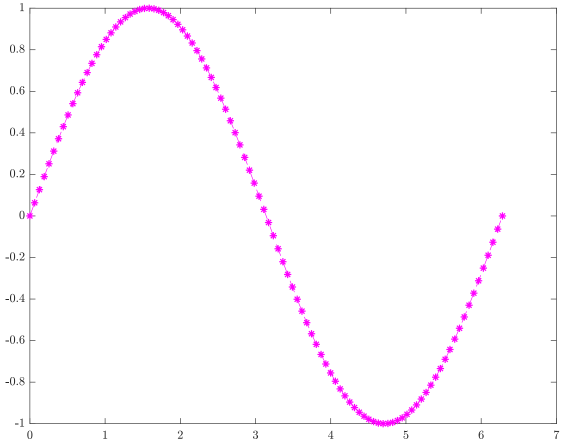

Figure 5:`plot(x, y, 'm*--')`

Scatter plot: Multiple Plots in a Single Figure

- `plot(x1, y1, style1, x2, y2, style2, ...)`: Combines the plots defined by the `(x, y, style)` triples, where `x`'s and `y`'s are vectors and `style`'s are strings. Example:

```
x = linspace(0, 2 * pi, 100)
y1 = sin(x)
y2 = cos(x)
y3 = sin(2 * x)
plot(x, y1, 'go-.', x, y2, 'r+:', x, y3, 'm*--')
```

Scatter plot: Multiple Plots in a Single Figure

- `plot(x1, y1, style1, x2, y2, style2, ...)`: Combines the plots defined by the `(x, y, style)` triples, where `x`'s and `y`'s are vectors and `style`'s are strings. Example:

  ```
  x = linspace(0, 2 * pi, 100)
  y1 = sin(x)
  y2 = cos(x)
  y3 = sin(2 * x)
  plot(x, y1, 'go-.', x, y2, 'r+:', x, y3, 'm*--')
  ```

- `hold on`: holds the current plot and all axis properties, including the current color and linestyle, so that subsequent graphing commands add to the existing graph without resetting the color and linestyle.

Scatter plot: Multiple Plots in a Single Figure

- `plot(x1, y1, style1, x2, y2, style2, ...)`: Combines the plots defined by the `(x, y, style)` triples, where `x`'s and `y`'s are vectors and `style`'s are strings. Example:

```
x = linspace(0, 2 * pi, 100)
y1 = sin(x)
y2 = cos(x)
y3 = sin(2 * x)
plot(x, y1, 'go-.', x, y2, 'r+:', x, y3, 'm*--')
```

- `hold on`: holds the current plot and all axis properties, including the current color and linestyle, so that subsequent graphing commands add to the existing graph without resetting the color and linestyle.
- `hold off`: returns to the default mode whereby `plot` commands erase the previous plots and reset all axis properties before drawing new plots.

Scatter plot: Multiple Plots in a Single Figure

```
% Example: plot(x1, y1, style1, x2, y2,style2,...)
x = linspace(0, 2 * pi, 100);
y1 = sin(x);
y2 = cos(x);
y3 = sin(2 * x);
style1 = 'go-.';
style2 = 'r+:';
style3 = 'm*--';
figure(6);
plot(x, y1, style1, x, y2, style2, x, y3, style3);
```

Scatter `plot`: Multiple Plots in a Single Figure

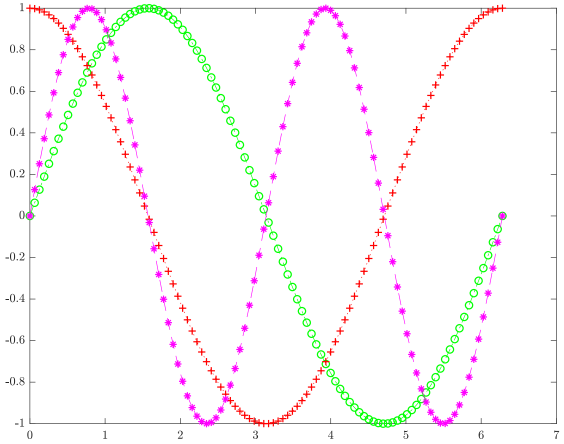

Figure 6:`plot(x, y1, style1, x, y2, style2, x, y3, style3)`

Scatter plot: Multiple Plots in a Single Figure

```
% Example: hold on
x = linspace(0, 2 * pi, 100);
y1 = sin(x);
y2 = cos(x);
y3 = sin(2 * x);
style1 = 'go-.';
style2 = 'r+:';
style3 = 'm*--';
figure(7);
hold on;
plot(x, y1, style1);
plot(x, y2, style2);
plot(x, y3, style3);
```

Scatter `plot`: Multiple Plots in a Single Figure

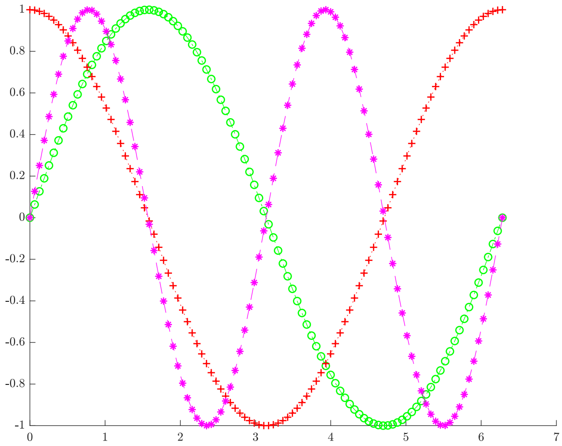

Figure 7:`hold on`

Scatter plot: title, grid, xlabel, ylabel, legend

- grid on/minor/off: Grid lines. Example:

```
grid on       % add grid lines
grid minor    % add fine grid lines
grid off      % Libao Jin lines
```

Scatter plot: `title`, `grid`, `xlabel`, `ylabel`, `legend`

- `grid on/minor/off`: Grid lines. Example:

```
grid on       % add grid lines
grid minor    % add fine grid lines
grid off      % Libao Jin lines
```

- `xlabel('labelText')`: x-axis label.

Scatter plot: `title`, `grid`, `xlabel`, `ylabel`, `legend`

- `grid on/minor/off`: Grid lines. Example:

```
grid on       % add grid lines
grid minor    % add fine grid lines
grid off      % Libao Jin lines
```

- `xlabel('labelText')`: x-axis label.
- `ylabel('labelText')`: y-axis label.

Scatter plot: title, grid, xlabel, ylabel, legend

- `grid on/minor/off`: Grid lines. Example:

```
grid on        % add grid lines
grid minor     % add fine grid lines
grid off       % Libao Jin lines
```

- `xlabel('labelText')`: x-axis label.
- `ylabel('labelText')`: y-axis label.
- `title('titleText')`: Graph title.

Scatter plot: `title`, `grid`, `xlabel`, `ylabel`, `legend`

- `grid on/minor/off`: Grid lines. Example:

```
grid on       % add grid lines
grid minor    % add fine grid lines
grid off      % Libao Jin lines
```

- `xlabel('labelText')`: x-axis label.
- `ylabel('labelText')`: y-axis label.
- `title('titleText')`: Graph title.
- `legend('legend1', 'legend2', ...)`: Display legend.

Scatter plot: title, grid, xlabel, ylabel, legend

- `grid on/minor/off`: Grid lines. Example:

```
grid on        % add grid lines
grid minor     % add fine grid lines
grid off       % Libao Jin lines
```

- `xlabel('labelText')`: x-axis label.
- `ylabel('labelText')`: y-axis label.
- `title('titleText')`: Graph title.
- `legend('legend1', 'legend2', ...)`: Display legend.
- `axis([xmin, xmax, ymin, ymax])`: Control axis scaling and appearance.

Scatter plot: title, grid, xlabel, ylabel, legend

```
% Libao Jint text interpreter to LaTeX
set(groot, 'defaultTextInterpreter','latex');
set(groot, 'defaultAxesTickLabelInterpreter','latex');
set(groot, 'defaultLegendInterpreter','latex')
```

Scatter plot: title, grid, xlabel, ylabel, legend

```
% Example: title, grid, xlabel, ylabel, legend
figure(8); hold on;
plot(x, y1, style1);
plot(x, y2, style2);
plot(x, y3, style3);
title('Trig functions');
grid on;  % grid minor;
xlabel('$x$');
ylabel('$y$');
legend('$\sin(x)$', '$\cos(x)$', '$\sin(2x)$', ...
       'Location', 'best');
axis([0, 2 * pi, -1, 1]);
```

Scatter plot: `title`, `grid`, `xlabel`, `ylabel`, `legend`

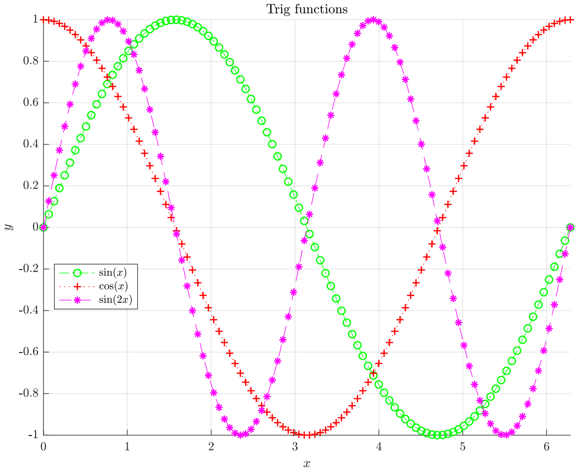

Figure 8:`title`, `grid`, `xlabel`, `ylabel`, `legend`

Advanced Plotting

Get/Set Properties: gcf, gca, get, set

- gcf: Get handle to current figure. Example:

```
fig = gcf
```

Get/Set Properties: `gcf`, `gca`, `get`, `set`

- `gcf`: Get handle to current figure. Example:

  ```
  fig = gcf
  ```

- `gca`: Get handle to current axis. Example:

  ```
  ax = gca
  ```

Get/Set Properties: `gcf`, `gca`, `get`, `set`

- `gcf`: Get handle to current figure. Example:

  ```
  fig = gcf
  ```
- `gca`: Get handle to current axis. Example:

  ```
  ax = gca
  ```
- `get(handle, 'PropertyName')`: Get object properties. Example:

  ```
  get(gcf, 'PaperPositionMode')
  ```

Get/Set Properties: `gcf`, `gca`, `get`, `set`

- `gcf`: Get handle to current figure. Example:

  ```
  fig = gcf
  ```

- `gca`: Get handle to current axis. Example:

  ```
  ax = gca
  ```

- `get(handle, 'PropertyName')`: Get object properties. Example:

  ```
  get(gcf, 'PaperPositionMode')
  ```

- `set(handle, 'PropertyName', PropertyValue)`: Set object properties. Example:

  ```
  set(gcf, 'PaperPositionMode', 'auto')
  ```

Get/Set Properties: gcf, gca, get, set

```
% Example: gcf, gca, get, set
x = linspace(0, 2 * pi, 100); y = sin(x);
figure(9);
plot(x, y);
axis([0, 2 * pi, -1, 1]);
set(get(gca, 'Title'), 'String', '$\sin(x)$');
set(get(gca,'Children'), 'LineWidth', 1.0,...
       'LineStyle', ':',...
       'Marker', 'd',...
       'MarkerSize', 4,...
       'MarkerEdgeColor', 'y',...
       'MarkerFaceColor', 'r');
set(gca, 'XTick', [0, pi / 2, pi, 3 * pi / 2, 2 * pi]);
set(gca, 'XTickLabel', {'0', '$\pi/2$', '$\pi$',...
        '$3 \pi / 2$', '$2\pi$'});
```

Get/Set Properties: gcf, gca, get, set

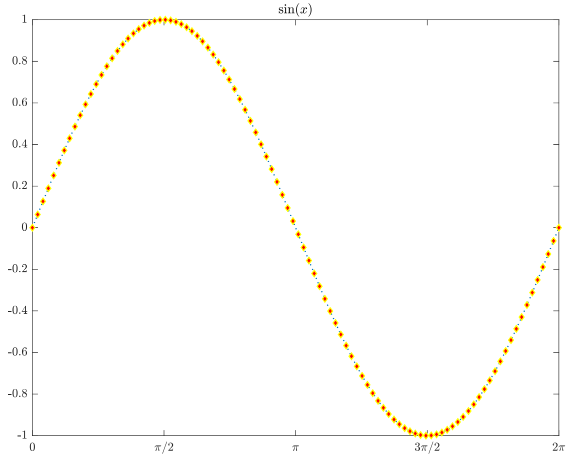

Figure 9:Example: gcf, gca, get, set

`subplot`: Create Tiled Axes

- `subplot(m,n,p)` or `subplot(mnp)`: Breaks the Figure window into an `m`-by-`n` matrix of small axes, selects the p-th axes for the current plot, and returns the axes handle. The axes are counted along the top row of the Figure window, then the second row, etc. Example:

```
figure(2)
subplot(1, 2, 1); plot(x1, y1);
subplot(1, 2, 2); plot(x2, y2);
```

subplot: Create Tiled Axes

```
% Example: subplot
x = linspace(0, 2 * pi, 100);
y1 = sin(x);
y2 = cos(x);
y3 = sin(2 * x);
y4 = cos(2 * x);
figure(10);
subplot(2, 2, 1);
plot(x, y1, 'gd-');  title('$\sin(x)$');
subplot(2, 2, 2);
plot(x, y2, 'ro:');  title('$\cos(x)$');
subplot(2, 2, 3);
plot(x, y3, 'ch-.'); title('$\sin(2x)$');
subplot(2, 2, 4);
plot(x, y4, 'b<--'); title('$\cos(2x)$');
```

subplot: Create Tiled Axes

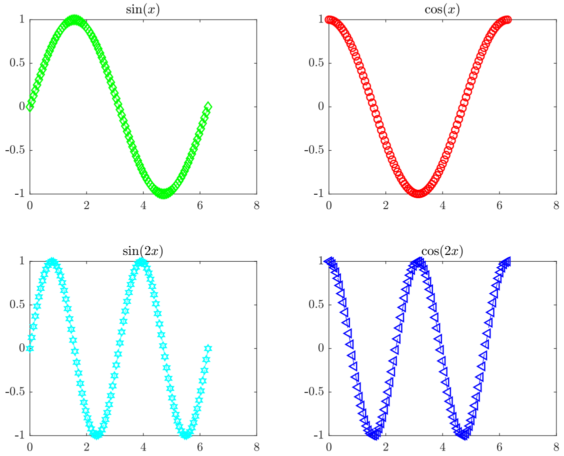

Figure 10:subplot

semilogy, semilogx, loglog, plotyy

- `semilogy`: semilogy Semi-log scale plot, same as `plot`, except a logarithmic (base 10) scale is used for the y-axis

`semilogy`, `semilogx`, `loglog`, `plotyy`

- `semilogy`: semilogy Semi-log scale plot, same as `plot`, except a logarithmic (base 10) scale is used for the y-axis
- `semilogx`: semilogx Semi-log scale plot, same as `plot`, except a logarithmic (base 10) scale is used for the x-axis

semilogy, semilogx, loglog, plotyy

- `semilogy`: semilogy Semi-log scale plot, same as `plot`, except a logarithmic (base 10) scale is used for the y-axis
- `semilogx`: semilogx Semi-log scale plot, same as `plot`, except a logarithmic (base 10) scale is used for the x-axis
- `loglog`: Log-log scale plot, same as `plot`, except logarithmic scales are used for both the x- and y- axes.

`semilogy`, `semilogx`, `loglog`, `plotyy`

- `semilogy`: semilogy Semi-log scale plot, same as `plot`, except a logarithmic (base 10) scale is used for the y-axis
- `semilogx`: semilogx Semi-log scale plot, same as `plot`, except a logarithmic (base 10) scale is used for the x-axis
- `loglog`: Log-log scale plot, same as `plot`, except logarithmic scales are used for both the x- and y- axes.
- `plotyy(x1, y1, x2, y2, 'func1', 'func2')` uses `func1(x1, y1)` to plot the data for the left axes and `func2(x2, y2)` to plot the data for the right axes. Example:

  ```
  plotyy(x1, y1, x2, y2, 'plot', 'semilogy')
  ```

  similar to

  ```
  figure(1); hold on;
  plot(x1, y1)
  semilogy(x2, y2)
  ```

semilogy, semilogx, loglog, plotyy

```
% Example: plotyy
x = 0:0.1:10;
y1 = 200 * exp(-0.05 * x) .* sin(x);
y2 = 0.8 * exp(-0.5 * x) .* sin(10 * x);
figure(11)
[hAx, hLine1, hLine2] = plotyy(x,y1,x,y2,'plot','stem');
set(hLine1, 'LineStyle', '--');
set(hLine2, 'LineStyle', ':');
grid minor;
xlabel('Time ($\mu$s)')
ylabel(hAx(1), 'Slow Decay')
ylabel(hAx(2), 'Fast Decay')
title('Multiple Decay Rates')
```

semilogy, semilogx, loglog, plotyy

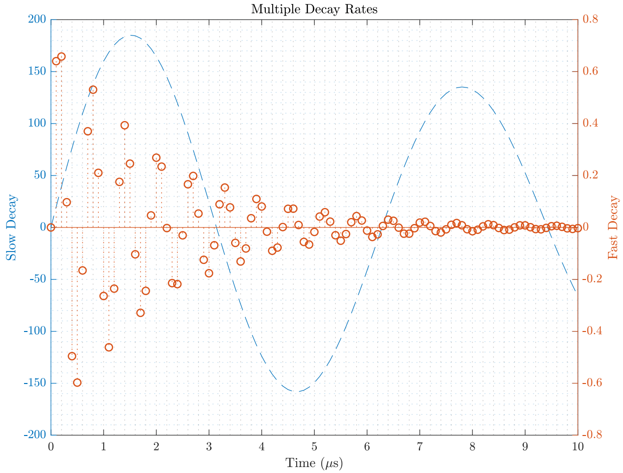

Figure 11:`plotyy`

`print`: Saving Figures

- `num2str(num)`: Convert numbers to character representation. Example:

```
num2str(57)    % returns '57'
```

print: Saving Figures

- `num2str(num)`: Convert numbers to character representation. Example:

  ```
  num2str(57)   % returns '57'
  ```

- `strcat(str1, str2)`: Concatenate str1 and str2 into one single string. Example:

  ```
  strcat('hello ', 'world')  % returns 'hello world'
  ```

print: Saving Figures

- `num2str(num)`: Convert numbers to character representation. Example:

```
num2str(57)   % returns '57'
```

- `strcat(str1, str2)`: Concatenate str1 and str2 into one single string. Example:

```
strcat('hello ', 'world')  % returns 'hello world'
```

- `mkdir newDirName`: Make new directory. Example:

```
mkdir thisIsANewDirectory
ls
```

print: Saving Figures

- `num2str(num)`: Convert numbers to character representation. Example:

  ```
  num2str(57)   % returns '57'
  ```

- `strcat(str1, str2)`: Concatenate str1 and str2 into one single string. Example:

  ```
  strcat('hello ', 'world')  % returns 'hello world'
  ```

- `mkdir newDirName`: Make new directory. Example:

  ```
  mkdir thisIsANewDirectory
  ls
  ```

- `print(handle, '-dformat', 'filename')`: Print or save a figure or model: Example:

  ```
  print(gcf, '-dpng', 'plot1.png')
  print(gcf, '-dpdf', 'plot2.pdf')
  ```

print: Saving Figures

```
% Example: print
mkdir figures
prefix = './figures/figure_';
for i = 1:11
    name = strcat(prefix, num2str(i));
    fig = figure(i);
    set(fig, 'PaperPositionMode', 'auto');
    pos = get(fig, 'PaperPosition');
    set(fig, 'PaperSize', [pos(3) pos(4)]);
    print(fig, '-dpdf', name);
end
```

Summary

- `figure`
- `hold`
- `plot`, `semilogy`, `plotyy`
- `subplot`
- `title`, `xlabel`, `ylabel`, `legend`, `axis`, `grid`
- `gcf`, `gca`, `get`, `set`
- `print`
- `strcat`, `num2str`