Christopher Lum
lum@uw.edu

Lecture 07c
Animation

The YouTube videos that cover this lecture are located at:

-'Creating Movies and Animations in Mathematica' at https://youtu.be/S03e6dwM100.

-'Creating Movies and Animations in Matlab' at https://youtu.be/3I1_ 5M7Okqo.

Outline

-Simple Animation in Mathematica
-Animation Theory
-Animation in Matlab
-Animation in Mathematica

Simple Animation in Mathematica

Mathematica provides the 'Manipulate' function to make quick and simple animations. 'Manipulate' allows you to generate a plot and then change one of more parameters of the plot. For example, consider the simple sinusoidal function of

$$f(t) = A \sin(\omega t)$$

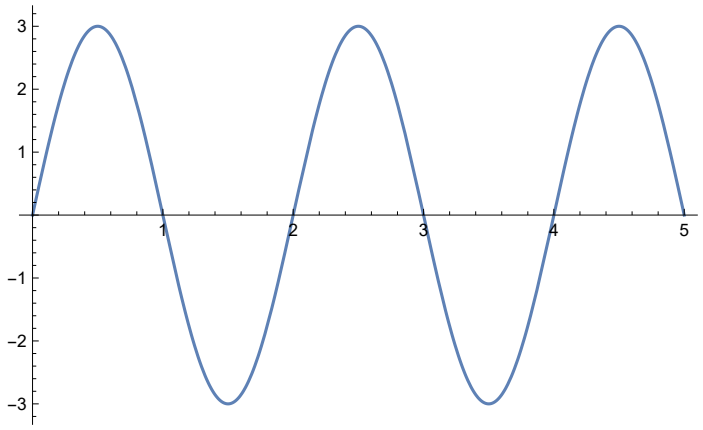
In[•]:=
```
f[t_, A_, ω_] = A Sin[ω t];
```

We can plot this if we define constants A and ω and plot over the time range of $t \in [0, t_{final}]$

In[•]:=
```
AExample = 3;
ωExample = π;
tFinal = 5;

Plot[f[t, AExample, ωExample], {t, 0, tFinal}]
```

Out[•]=

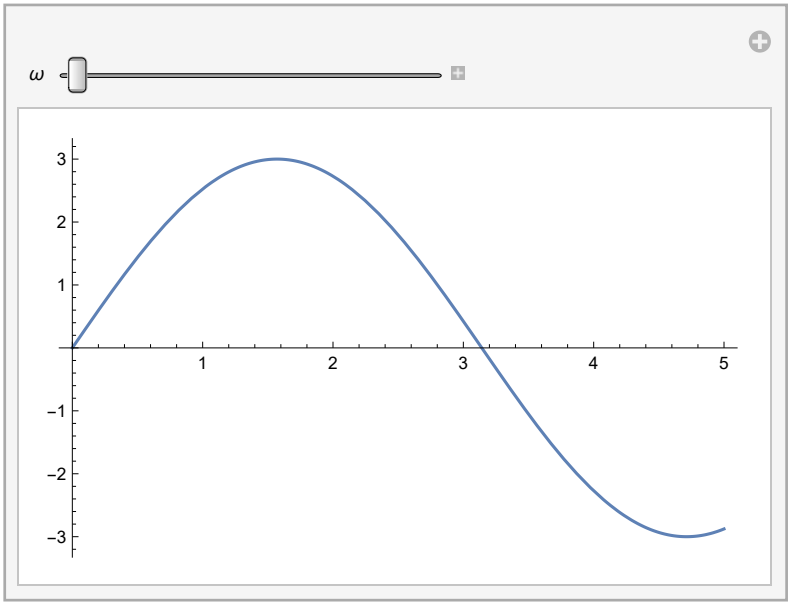


Suppose we want to animate the plot and dynamically vary ω in the function. To do this, we wrap the previous 'Plot' command with a 'Manipulate' command and then insert a symbol for ω instead of specifying a concrete value.

Note that you can hit the '+' button in the figure below and the push the play button to animate the scenario.

In[•]:=
```
Manipulate[
 Plot[f[t, AExample, ω], {t, 0, tFinal}],
 {ω, 1, 5}]
```

Out[•]=

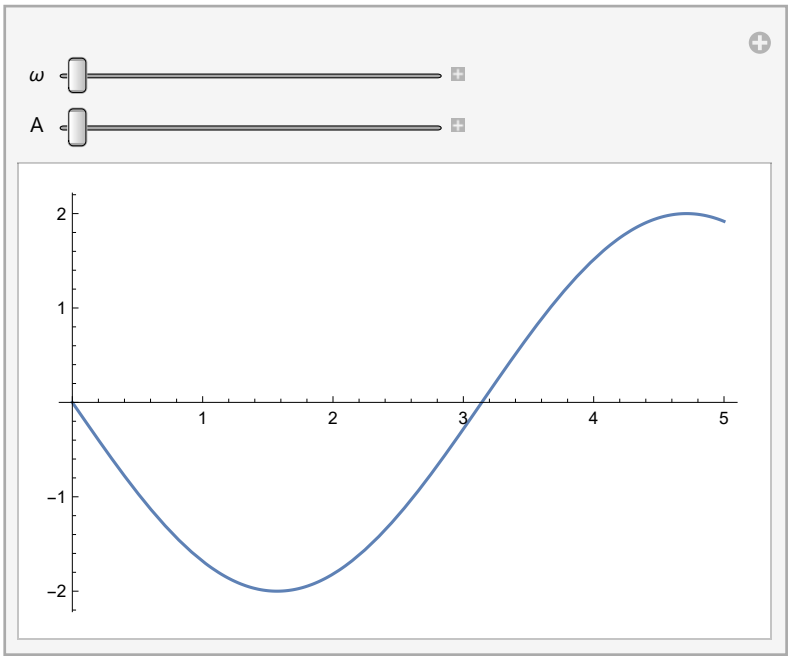


We can even manipulate multiple variables simultaneously. For example, suppose we wanted to see

how the function changed if we vary both A and ω. To do this, we simply "manipulate" both the A and ω variable.

In[ ]:=
```
Manipulate[
 Plot[f[t, A, ω], {t, 0, tFinal}],
 {ω, 1, 5},
 {A, -2, 3}]
```

Out[ ]=



Animation Theory

Animation and movies is simply a series of still frames that are show in rapid succession.

Example: Flip Book

Go to YouTube and search for "Amazing Gangnam Style Animation (Flipbook)" (link)

We already know how to draw static pictures/figures in Matlab and Mathematica so we can build on this to create animations and movies.

Animation in Matlab

Animation in Matlab consists of simply plotting a static picture, taking a snapshot of that picture, and then repeating

1. Run simulation or procedure to generate data
2. Draw/render the scenario at a time $t = t_k$
3. Take a snapshot of this scenario

4. Advance time to $t = t_{k+1}$ and repeat step 2
5. Save movie once all frames are drawn

Matlab provides several functions to help with this process.

Step 1: Run Simulation

If your data is generated from a Simulink model, you can easily use the 'sim' function to automatically run your model from a script. If you have setup appropriate 'To Workspace' blocks to save appropriate signals, you can then extract data from the resulting data structures.

Step 2: Draw/Render Scenario

We would like to step through the data and plot the scenario at each time. Therefore, it makes sense to use a large 'for' loop to step through all the data. We can then use simple Matlab 'plot' commands to draw the scenario as appropriate.

Step 3: Take a Snapshot

Once the still frame is generated, we would like to take a snapshot of the scene. Matlab provides a function 'getframe' to take a snapshot. You can then assign this snapshot to an element of a vector.

Step 4: Advance Time

Your 'for' loop will automatically increment time for you. If desired, you may want to jump ahead in time (or skip several samples) if your time steps are small. Matlab (and many other languages) provide a function 'continue' to skip a given iteration of a loop.

Step 5: Save Movie

Once you have generated a large vector of still images, you can have Matlab put them together into an avi movie using the 'VideoWriter' class.

Older versions of Matlab provide the 'movie2avi' function.

Example: Animating Motion Along 3D Parametric Curve

Suppose that we would like to see a particle moving along the curve defined by

$$x(t) = 5\cos(t)$$
$$y(t) = 2\sin(t)$$
$$z(t) = t$$

for $t \in [0, 2\pi]$

Some sample code for work flow this might look like

```
%Animate a point moving along a 3D parametric curve
t = linspace(0,2*pi,100);
x = 5*cos(t);
y = 2*sin(t);
z = t;

%Animate the scenario
figure
for k=1:length(t)
    %Extract data about the current point
    t_k = t(k);
    x_k = x(k);
    y_k = y(k);
    z_k = z(k);

    %Where is the current point?
    plot3(x_k, y_k, z_k, 'go', 'LineWidth', 3, 'MarkerSize', 15)

    %Plot the entire curve
    hold on
    plot3(x, y, z, 'b-', 'LineWidth', 2);

    %Add plotting options
    grid on
    xlabel('x')
    ylabel('y')
    zlabel('z')
    title(['t = ',num2str(t_k)])
    view([30 35])

    %Save the frame
    movieVector(k) = getframe;
    clf
end

%Create a VideoWriter object and set properties
myWriter = VideoWriter('curve.avi');
myWriter.FrameRate = 20;

%Open the VideoWriter object, write the movie, and close the file
open(myWriter);
writeVideo(myWriter, movieVector);
close(myWriter);
```

Animation in Mathematica

Mathematica provides several functions to help with this process.

Step 1: Run Simulation

If your data is generated from an external source, you can import them. In many scenarios, you would like to plot a function as one or two parameters vary, so you simply need to define the function at this point.

Step 2: Draw/Render Scenario

We would like to step through the data and plot the scenario at each time. Therefore, it makes sense to

use a large 'for' loop to step through all the data. We can then use simple plotting commands (Plot, Plot3D, ParametericPlot, Graphics3D, etc.) to draw the scenario as appropriate. You can use the 'Show' command to draw multiple plots on the same figure.

Step 3: Take a Snapshot

Once the still frame is generated, we would like to take a snapshot of the scene. Mathematica allows you to simply assign the current figure to a vector of frames (see below)

Step 4: Advance Time

Your 'for' loop will automatically increment time for you. If desired, you may want to jump ahead in time (or skip several samples) if your time steps are small.

Step 5: Save Movie

Once you have generated a large vector of still images, you can have Mathematica put them together into an avi movie using the 'Export' function. Note that this will write the file to the location specified by 'Directory[]'.

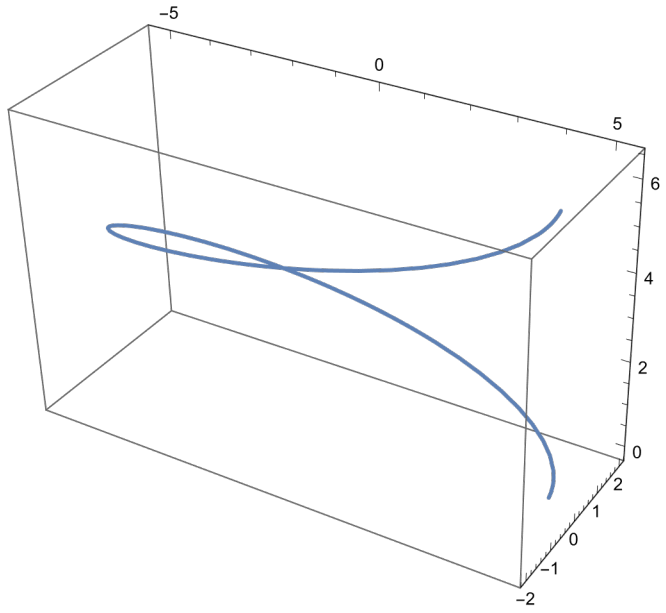
Step 1: Run Simulation (Define Parametrization)

We can first define the parametrization

In[ ]:=
```
x[t_] = 5 Cos[t];
y[t_] = 2 Sin[t];
z[t_] = t;

ParametricPlot3D[{x[t], y[t], z[t]}, {t, 0, 2 π}]
```

Out[ ]=



We can also setup containers defining the time steps that we would like to render the frame at. We also setup a container to hold all of the frames of animation.

In[•]:=
```
(*Create range of times*)
tRange = Range[0, 2 π, 2 π / 100];

(*Initialize a container to hold all the frames of
 animation. We initialize this container with all zeros*)
movieVector = ConstantArray[0, Length[tRange]];
```

Step 2: Draw/Render Scenario

Step 3: Take a Snapshot

Step 4: Advance Time

We can use a 'for' loop to create frames of animation.

In[•]:=
```
For[k = 1, k ≤ Length[tRange], k++,

 (*Where is the point currently?*)
 tk = tRange[[k]];
 xk = x[tk];
 yk = y[tk];
 zk = z[tk];

 (*Render the scenario.  Note that we
  store the entire frame into the movieVector container*)
 movieVector[[k]] = Show[
   (*Where is the current point?*)
   Graphics3D[
    {
     AbsolutePointSize[15], Green, Point[{xk, yk, zk}]
    }
   ],

   (*Plot the entire curve*)
   ParametricPlot3D[{x[t], y[t], z[t]}, {t, 0, 2 π}],

   (*Add plotting options*)
   AxesLabel → {"x", "y", "z"},
   PlotLabel → StringJoin["t = ", ToString[N[tk]]]
  ]
]
```

Step 5: Save Movie

Now export the movie

In[ ]:=
```
(*Export this to an avi file*)
fileName = "curveMathematica.avi";
Export[fileName, movieVector];

(*Note that 'Export' saves to the directory specified by'Directory[]'*)
Print["saved '", fileName, "' to '", Directory[], "'"]
```

••• General: Using a limited version of FFmpeg. Install FFmpeg to get more complete codec support.