Python DeCal

Week 9

Announcements

- Project Proposal Due Wednesday before lecture!
- Final Projects
  - Keep working on it - don’t stop lol
  - Keep coming to OHs
  - Deliverable due April 19th
  - Presentations on 4/19 and 4/21 (last two lectures)
- Attendance: https://forms.gle/W95TvquHAmtcxknz8

Recap and Discussion

Time to check-in with project groups:

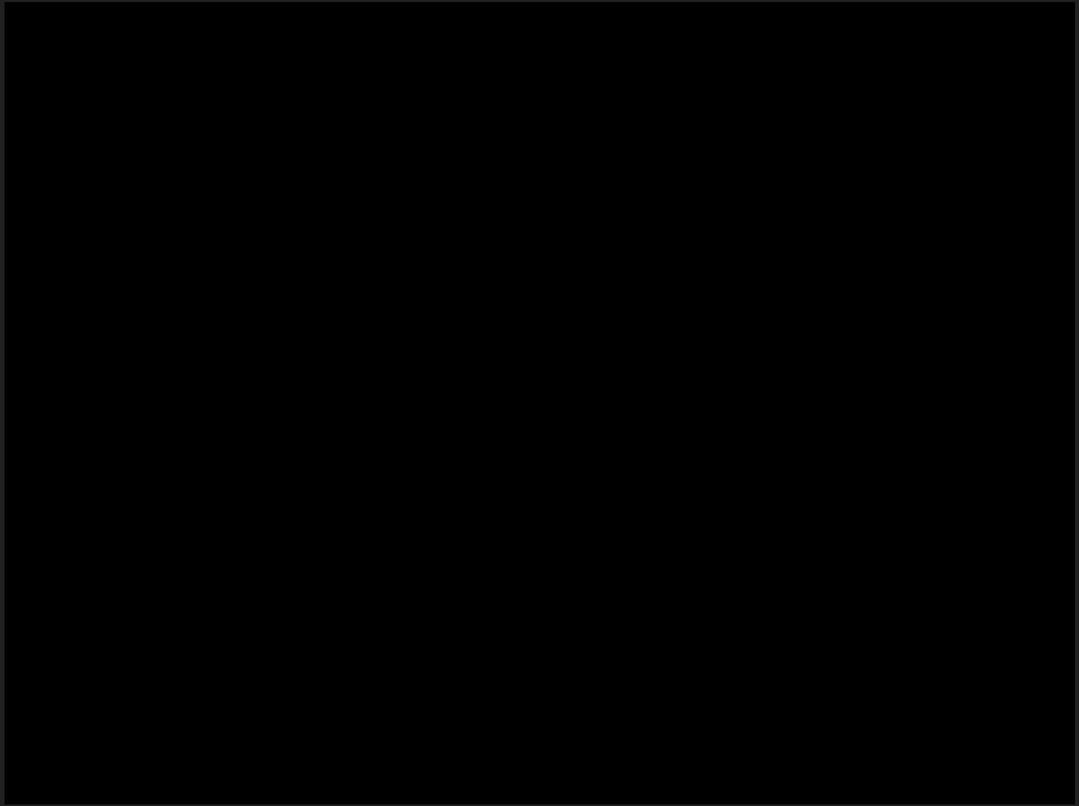
- How are final project proposals going?

- Are you planning to use **`class`** in your final project?

Intro to Animation Examples

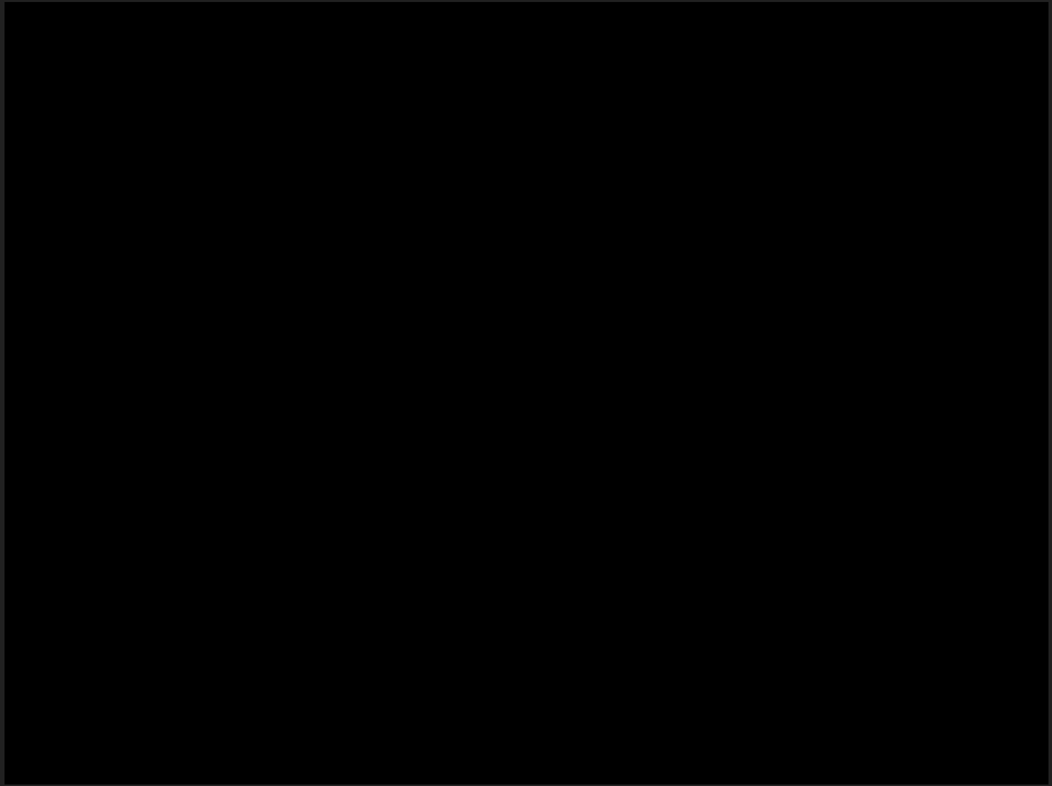
Saeed Mohanna’s

“Erosion o Crater Lake, Oregon”



Kyla Devilla’s “Infection Simulator”

Intro to Animation Examples

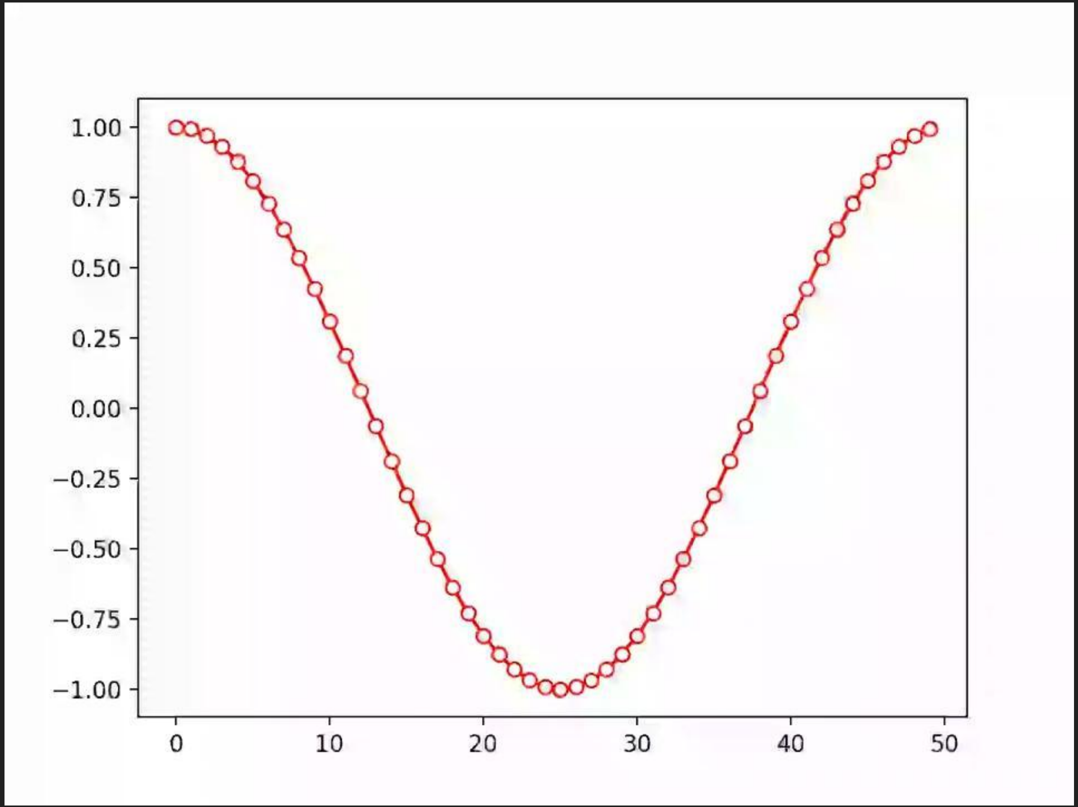


There are lots of different ways to animate!

But we're going to focus on
matplotlib.animation.FFMpegWriter
(documentation can be found [here])

matplotlib.animation.FuncAnimation is another common way to animate (documentation can be found [here])

A Moving Sine Wave (using FFMpeg)



A Moving Sine Wave (FFMpeg)

```
import matplotlib.pyplot as plt
import numpy as np
from matplotlib.animation import FFMpegWriter
```

Import necessary libraries:

from matplotlib.animation import FFMpegWriter

```
metadata = dict(title='Sinusoidal Wave Animation', artist='Matplotlib')
writer = FFMpegWriter(fps=15, metadata=metadata)
fig = plt.figure()

with writer.saving(fig, 'sinusoidal_wave.mp4', dpi=200):
    nf = 100
    for it in range(nf):
        n = 50
        y = np.zeros(n)
        f = 2.0*np.pi/n
        for i in range(n):
            y[i] = np.cos(f*(i+it)) + np.sin(f*it)*np.cos(3*f*(i+it))
        plt.clf()
        plt.plot(y, 'ro-',mfc='w')
        plt.show()
        plt.draw()
        plt.pause(0.1) # On Macs 0.01 seems fine, on PCs use 0.05 or 0.1
        writer.grab_frame()
```

This is the normal code for plotting sine wave

A Moving Sine Wave (FFMpeg)

```
metadata = dict(title='Sinusoidal Wave Animation', artist='Matplotlib')
writer = FFMpegWriter(fps=15, metadata=metadata)
fig = plt.figure()
```

- Metadata is optional and just fills in file information such as the title, artist, comment, etc. -- note that the title isn't the name of the file
- Initialize the writer -- this is what saves each frame to the mp4 file
  - fps -- frame rate for your mp4
- Initialize the plot figure where the data will be plotted and saved

A Moving Sine Wave (FFMpeg)

```
with writer.saving(fig, 'sinusoidal_wave.mp4', dpi=200):
    nf = 100
    for it in range(nf):
        n = 50
        y = np.zeros(n)
        f = 2.0*np.pi/n
        for i in range(n):
            y[i] = np.cos(f*(i+it)) + np.sin(f*it)*np.cos(3*f*(i+it))
        plt.clf()
        plt.plot(y, 'ro-',mfc='w')
        plt.show()
        plt.draw()
        plt.pause(0.1) # On Macs 0.01 seems fine, on PCs use 0.05 or 0.1
        writer.grab_frame()
```

- with writer.saving() -- saves each paused frame of the figure into the mp4 file name -- the file name must end in ‘.mp4’!!
- plt.clf() -- clears the frame, plt.draw() -- draws the plot on the figure
- plt.pause() -- pauses the current frame for x seconds
- writer.grab_frame() -- actually saves the frame with the writer

A Moving Sine Wave (using FuncAnimation)

```python
fig = plt.figure()
ax = plt.axes(xlim=(0, 3), ylim=(-3, 3))
line, = ax.plot([], [], lw=3)
```

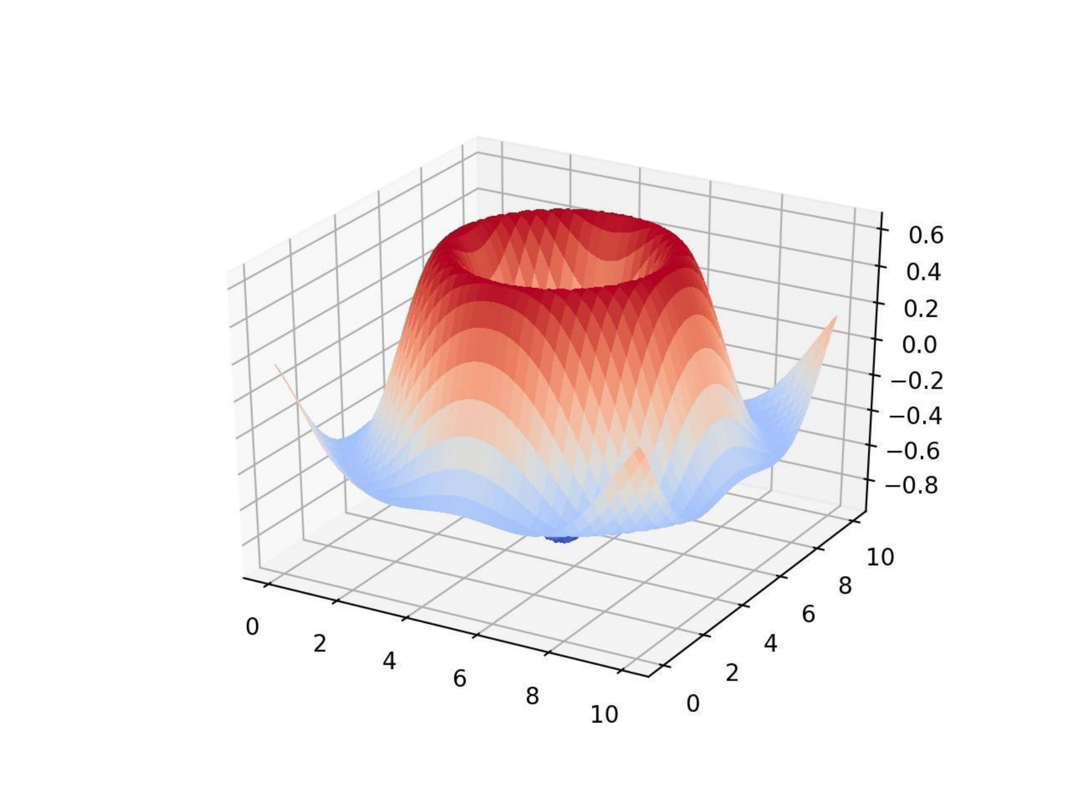
← Initializing figure for our animation

```python
# Plots the background of each frame
def init():
    line.set_data([], [])
    return line,

# Animation function
def animate(i):
    x = np.linspace(0, 3, 1000)
    y = 2 * np.sin(5 * np.pi * (x - 0.02 * i))  # try changing this function!
    line.set_data(x, y)
    return line,
```

← define functions for your background and how you want to animate

```python
anim = animation.FuncAnimation(fig, animate, init_func=init, frames=200, interval=20, blit=True)
anim.save('sine_wave_animation.mp4', fps=30, extra_args=['-vcodec', 'libx264'])
```

2D Moving Sine Wave (using FFMpeg)



2D Moving Sine Wave (using FFMpeg)

```
# On Macs, use:   %matplotlib osx
# On PCs, use:   %matplotlib qt
%matplotlib osx

import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.animation import FFMpegWriter
metadata = dict(title='Sinusoidal wave in 3D space', artist='Matplotlib')
writer = FFMpegWriter(fps=15, metadata=metadata,bitrate=200000)
fig = plt.figure(dpi=200)
```

- %matplotlib osx or %matplotlib qt allows a pop-up window to open to see the animation in real-time
- Import needed libraries including for 3D plots and colormap
- Initialize animation writer and plot figure

2D Moving Sine Wave (using FFMpeg)

```
n=151
L = 10.0
X = np.linspace(0, L, n)
Y = np.linspace(0, L, n)
X, Y = np.meshgrid(X, Y)
Z = np.zeros((n,n))
```

- Initialize the 3D meshgrid using np.linspace()
  - X, Y are the 2D axes and the Z axis is the vertical axis that plots the data values

2D Moving Sine Wave (using FFMpeg)

```
with writer.saving(fig, "wave2d.mp4", dpi=200):
    nf = 100
    for it in range(nf):
        if (it%10==0): print(it,end='')
        print('.',end='')

        f = 2.0*2.0*np.pi/n
        for i in range(n):
            for j in range(n):
                R = np.sqrt( (i-n/2)**2 + (j-n/2)**2 )
                Z[i,j] = np.sin(f*(R+it))*np.exp(-R/100.0)
        fig.clear()
        ax = fig.gca(projection='3d')
        ax.plot_surface(X, Y, Z, cmap=cm.coolwarm, antialiased=False)
        plt.draw()
        plt.pause(0.01)
        writer.grab_frame()
```

- Initialize ax with 3d projection and colormap
  - antialiased=False for faster plotting, though a bit jaggier
- Same format as the 1D wave earlier, except in this case the data is stored in Z, a 2D array instead of a 1D array (use a double for loop to update instead of one for loop)
- Also use ax.plot_surface() instead of plt.plot()

General Setup

```python
import matplotlib
from matplotlib.animation import FFMpegWriter
metadata = dict(title='My first animation in 3D', artist='Matplotlib')
writer = FFMpegWriter(fps=15, metadata=metadata,bitrate=200000)
fig = plt.figure(dpi=200)
```

← Import FFMpegWriter and set up metadata, writer, and figure

```python
with writer.saving(fig, "fig_name.mp4", dpi=200):
    for it in range(iterations):  # for each iteration it updates

        ## write code that updates your data

        plt.draw()
        plt.pause(0.01)  # runs the frame for 0.01 seconds
        writer.grab_frame()  # this saves the frame
```

← Create for loop that updates your data for the wanted number of iterations

** Be sure to include writer.grab_frame(), plt.pause() or nothing will be saved! **

Tips for Animating

- Keep in mind boundary conditions and make sure to initialize those and check them at every loop
  - Ex: if one of the edges should always be a certain value
- For turning differential equations into code, anything that is dx/dt can be turned into updating the array by delta x for every delta t step of the for loop
  - You can increment a for loop by a small delta t value like 0.00005
  - Ex: kinematic equations, heat diffusion, etc.
- Google is your friend!
- Check out the Animation Guide Jupyter Notebook on bCourses

DEMO

Forest Fire Example

```python
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
%matplotlib osx
```

```python
# Probability Parameters

D = 0.6 # Probability that A[i,j] starts as vegetation and not an empty cell
B = 0.25 # Probability that a burning cell becomes burnt
I = 0.2 # Probability that a burning cell turns its vegetation into burning
```

```python
def display(A):
    maxX = A.shape[0]
    maxY = A.shape[1]
    B = np.zeros((maxY, maxX))
    for ix in range(0,maxX):
        for iy in range(0,maxY):
            B[maxY-1-iy,ix] = A[ix,iy]

    fig.clear()
    plt.rcParams['figure.figsize'] = [10, 10/maxX*maxY]
    plt.imshow(B, cmap='Dark2');
    plt.axis('off');
    plt.show()
    plt.draw()
    plt.pause(0.01)
    writer.grab_frame()
```

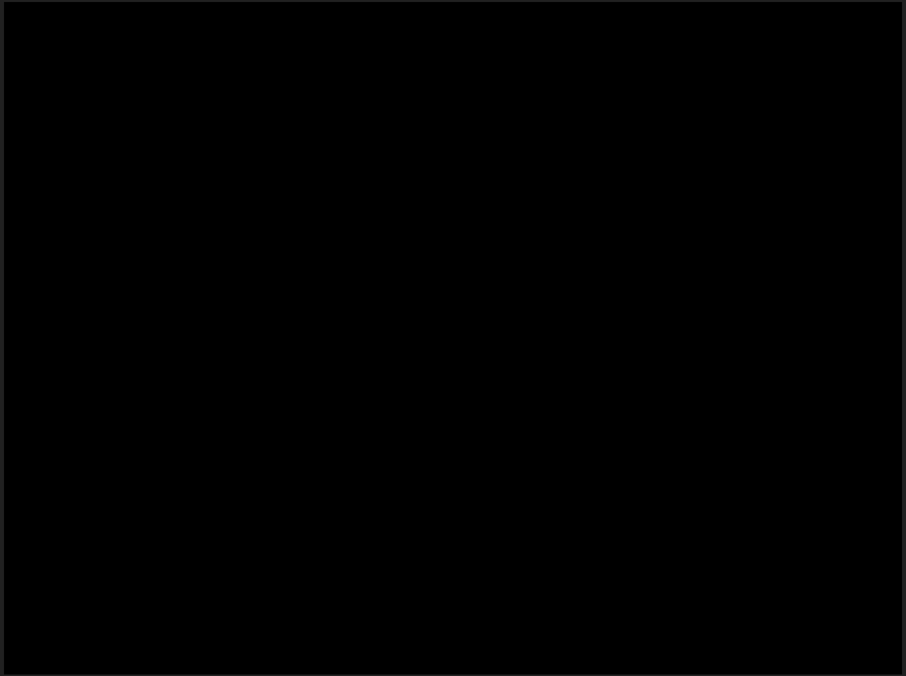
Forest Fire Example

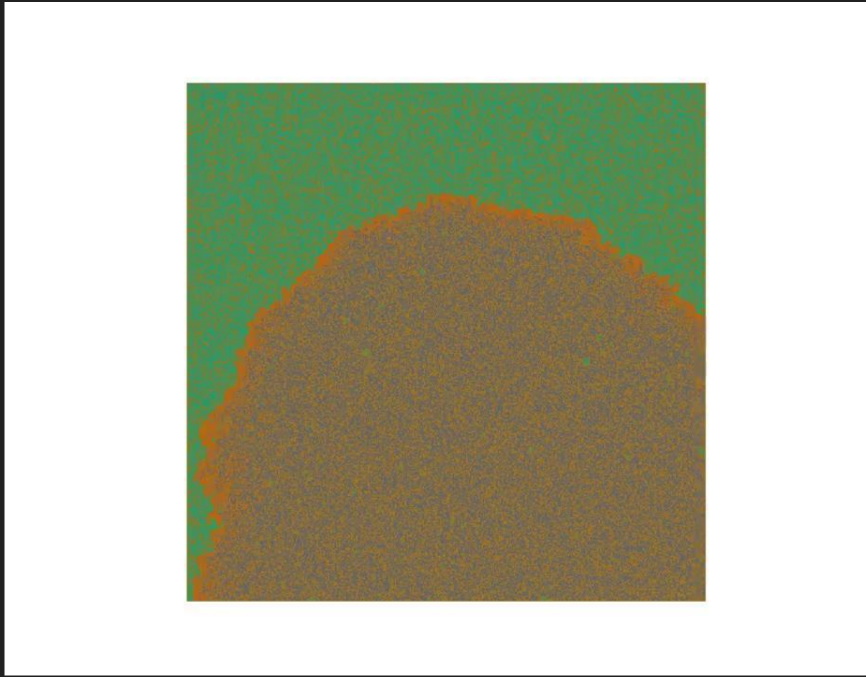
```
maxX, maxY = 200, 200
# Initialize matrix containing all 2D grid points A(x,y)
A = np.zeros((maxX, maxY))

# Fill points in as vegetation depending on D
# 0 = tree (green), 1 = burning (orange), 4 = empty (brown), 5 = burnt (gray)
# to match color map
for i in range(maxX):
    for j in range(maxY):
        p = np.random.random()
        if p < D:
            A[i,j] = 0
        elif p > .999:
            A[i,j] = 1
        else:
            A[i,j] = 4

# Make border all empty cells to simplify boundary conditions
A[:,0] = 4
A[:,maxY-1] = 4
A[maxX-1,:] = 4
A[0,:] = 4
A[0,0] = 5
```

Forest Fire Example

```python
from matplotlib.animation import FFMpegWriter
metadata = dict(title='Forest Fire Simulation', artist='Emily')
writer = FFMpegWriter(fps=15, metadata=metadata)
fig = plt.figure()
with writer.saving(fig, "Animation3.mp4", dpi=200):
    n_steps = 500
    for t in range(n_steps):
        A_new = A[:]
        burning = False
        for i in range(maxX):
            for j in range(maxY):
                if A[i,j] == 1: #if cell is burning
                    burning = True
                    #check if cell should become burnt
                    if np.random.random() < I:
                        A_new[i,j] = 5

                    #check if vegetation neighbors of burning cell should become burning
                    for g in range(i-1,i+2):
                        for h in range(j-1,j+2):
                            if A[g,h] == 0:
                                if np.random.random() < B:
                                    A_new[g,h] = 1
        A = A_new[:]
        if t % 10 == 0:
            plt.clf()
            plt.rcParams['figure.figsize'] = [10, 10/maxX*maxY]
            plt.imshow(A, cmap='Dark2');
            plt.axis('off');
            plt.show()
            plt.draw()
            plt.pause(0.01)
            writer.grab_frame()
        if burning == False:
            break
```

Forest Fire Example





Recap and Discussion

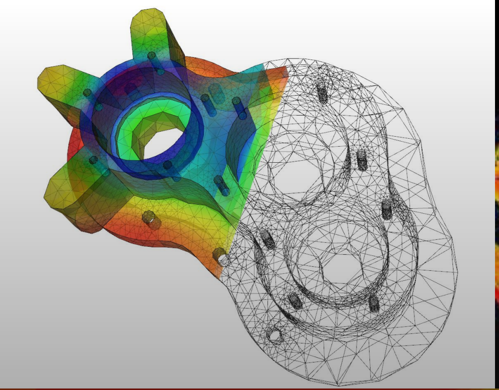
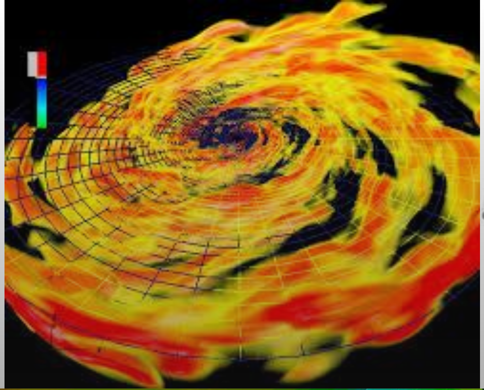
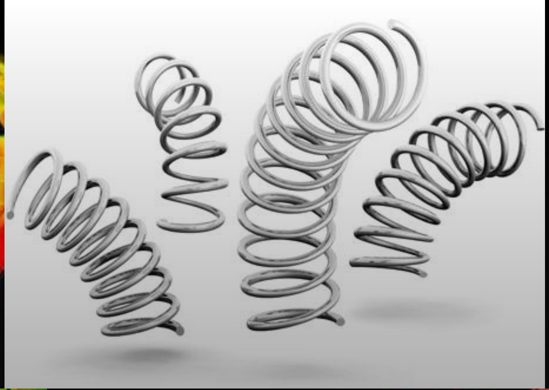
- What are some of the challenges that you have had when writing your project proposal?
  - LaTeX? Etc.


Wednesday

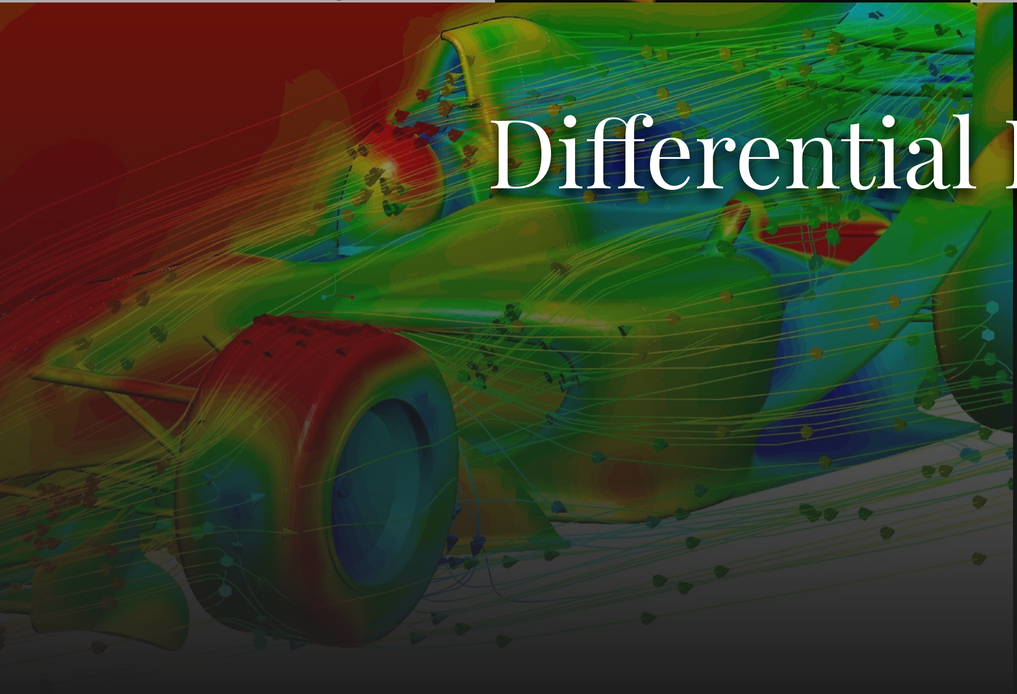
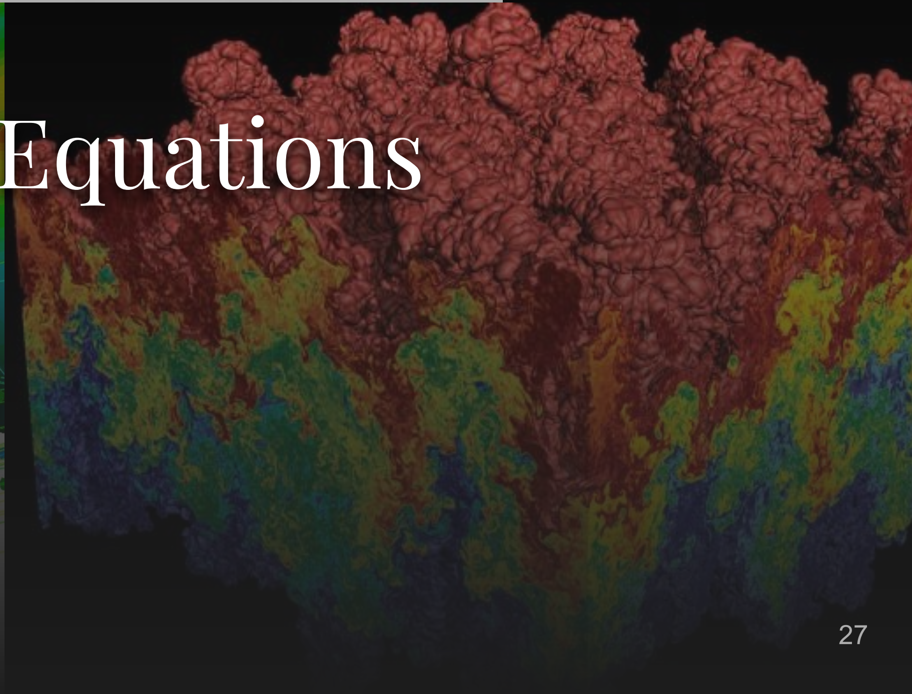
Attendance

https://forms.gle/sHSo7MLenXeR3ahd7

SHOW-AND-TELL


Differential Equations

Differential equations (DiffEQs)...

- They govern the world…
- You have already seen some and you will see more in the future

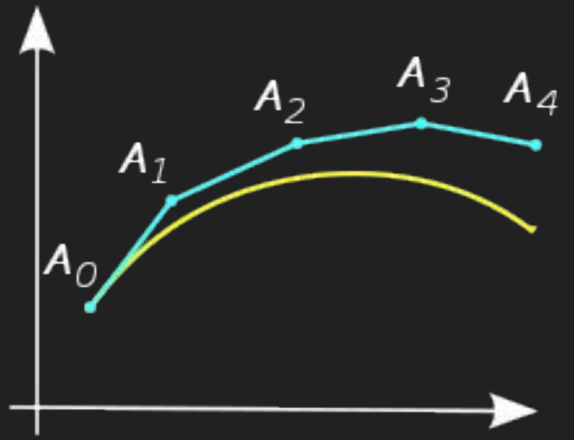
$$\frac{dx}{dt} = v(t)$$

$$\frac{d^2\mathbf{r}}{dt^2} = \frac{\mathrm{F}(\mathbf{r})}{m}$$

$$\frac{\partial^2 f(x,t)}{\partial t^2} = c^2 \frac{\partial^2 f(x,t)}{\partial x^2}$$

Solving ordinary differential equations (ODEs)...

- Euler's method (very similar to numerical integration)
    - Break down to segments
    - Specifying initial condition
        - You need to start from somewhere
        - Order of diff eq. = # of initial conditions
    - Need to specify your step size
- Assign! Not Append!



Solving ordinary differential equations (ODEs)...

$$\frac{dy}{dx} = xy \qquad\qquad (x_0, y_0) = (2, 3)$$

```
dx = 0.01
x0, y0 = 2, 3

def derivative(y,x):
    return x*y

x_arr = np.linspace(x0,5,301)
y_arr = np.zeros(301)
```

```
y_arr[0] = y0

for i in range(1, 301):
    dy = derivative(y_arr[i-1], x_arr[i-1]) * dx
    y = y_arr[i-1]+dy
    y_arr[i] = y
```

Scipy function as well...

https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.odeint.html

```
from scipy.integrate import odeint
```

```
odeint(func, y0, t)
```

```
def deriv(y,t):
    return …
```

Your function, make sure y is the first variable

Initial condition on y

Time point at which you would like to evaluate the function for

Equivalent to x_arr