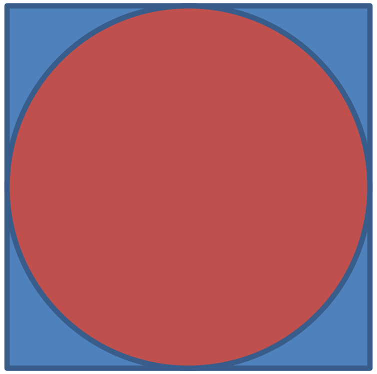
Prelab Exercise

Think about the following while MATLAB is booting up:

Archimedes is watching the God of Luck, Hermes, play darts. As it happens Hermes is pretty drunk so his aim is off. Since he is Hermes he still manages to make all the darts land within a square that perfectly encloses the circular dart board:



Seeing this, Archimedes immediately devises a plan to trick Hermes into divulging the secret value of the number π.

What's his plan?
Hint: Think about the area of the two shapes

Exercise

- Code up **The Plan** in MATLAB.

Hint: Look up rand() in the help file.

ODE

- Solve the following ODE analytically:

$$\frac{ds}{dt} = t$$

Think: separation of variables.

Euler's Method

$$\frac{dy(t)}{dt} = f(t, y(t))$$

$$y(t + \Delta t) = y(t) + f(t, y(t))\Delta \mathrm{t} + O(\Delta t^2)$$

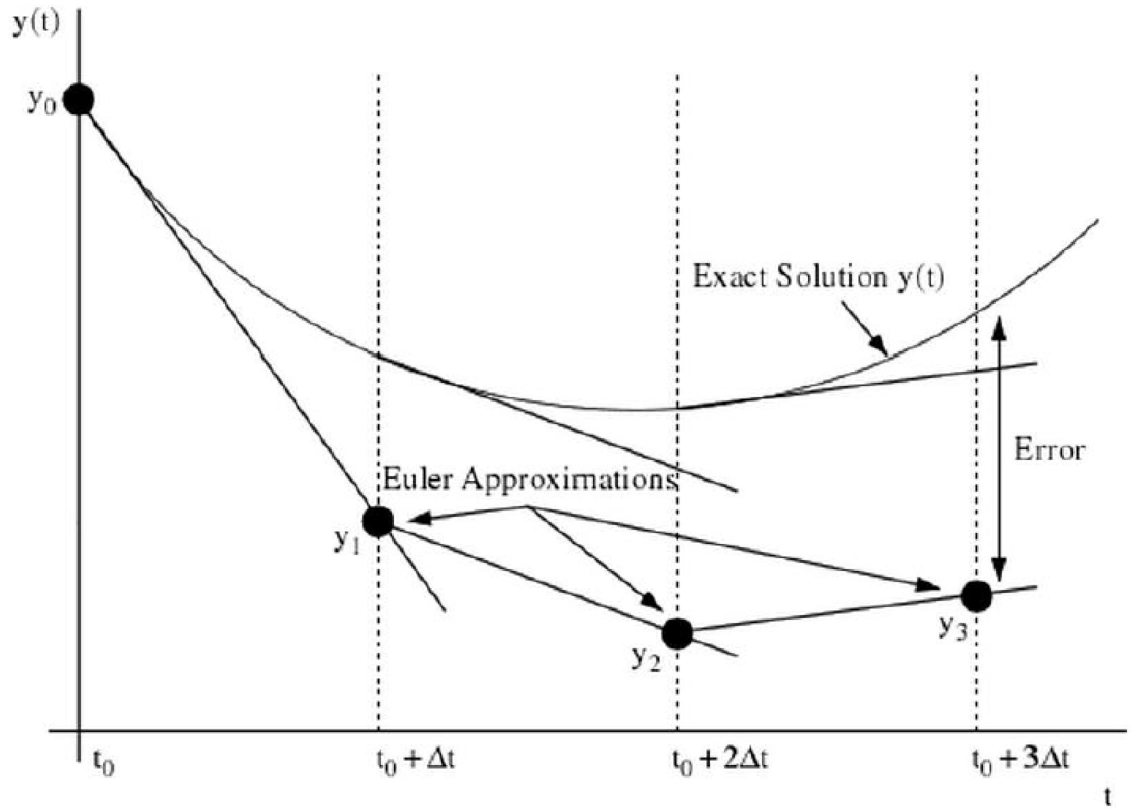


Figure 1. Graphical description of the Euler method (Kutz, 2013)

- Solve for s(t) numerically with Euler's method with s(0) = 1 and time stamps at t = 0:delta_t:50;

Vary the step size between 0.1 to 1 and observe how the result changes.

- Plot the values side by side on the same figure.
- Compute the differences and plot them.

How do we do this with 4-th order RK?
… related Q: what is a 4-th order RK?

4-th order RK

Liken it to *averaging* various first order derivs

$\frac{dy}{dt} = f(y,t)$, with stepsize *h*

Define:

$$k_1(y_1,t_1) = \frac{dy}{dt} = \mathrm{f(y_1,t_1)}$$
$$k_2(y_1,t_1) = f(y_1 + 0.5h\, k_1, t + 0.5h)$$
$$k_3(y_1,t_1) = f(y_1 + 0.5h\, k_2, t + 0.5h)$$
$$k_4(y_1,t_1) = f(y_1 + hk_3, t + h)$$

$$y_2(t_1 + h) = y_1(t_1) + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

$$\text{EX: } \frac{dy}{dt} = yt * 5, \quad y = \exp\left(\frac{t^2}{2}\right) * 5$$

Liken it to *averaging*

$\frac{dy}{dt} = f(y,t)$, with stepsize h

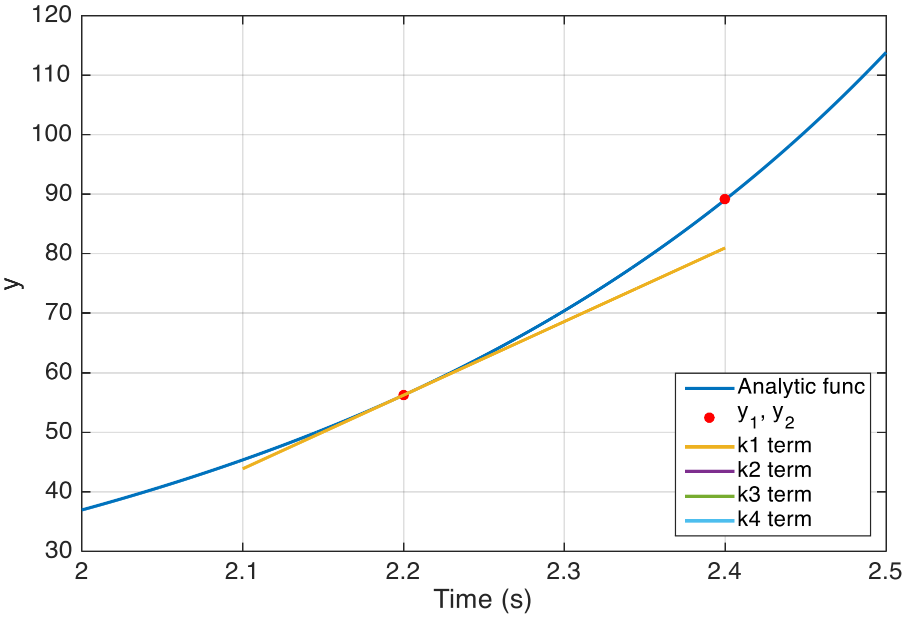
Define:

$$k_1(y_1, t_1) = \frac{dy}{dt} = \mathrm{f}(\mathrm{y}_1, \mathrm{t}_1)$$

$$k_2(y_1, t_1) = f(y_1 + 0.5h\, k_1, t + 0.5h)$$

$$k_3(y_1, t_1) = f(y_1 + 0.5h\, k_2, t + 0.5h)$$

$$k_4(y_1, t_1) = f(y_1 + hk_3, t + h)$$



$$y_2(t_1 + h) = y_1(t_1) + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

$$\text{EX: } \frac{dy}{dt} = yt*5, \qquad y = \exp\left(\frac{t^2}{2}\right) * 5$$

Liken it to *averaging*

$\frac{dy}{dt} = f(y,t)$, with stepsize h

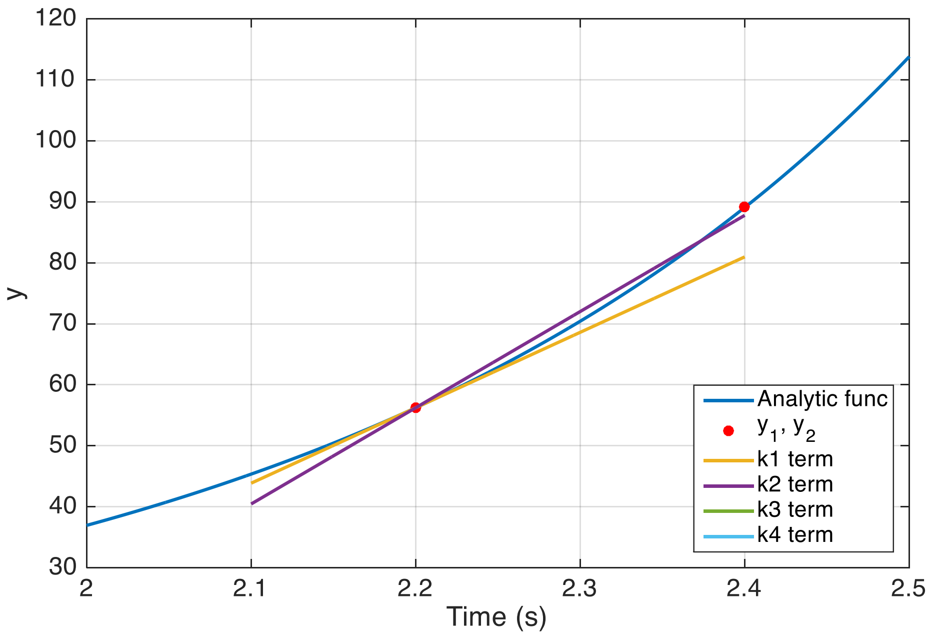
Define:

$$k_1(y_1, t_1) = \frac{dy}{dt} = \mathrm{f}(\mathrm{y}_1, \mathrm{t}_1)$$

$$k_2(y_1, t_1) = f(y_1 + 0.5h\, k_1, t + 0.5h)$$

$$k_3(y_1, t_1) = f(y_1 + 0.5h\, k_2, t + 0.5h)$$

$$k_4(y_1, t_1) = f(y_1 + hk_3, t + h)$$



$$y_2(t_1 + h) = y_1(t_1) + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

$$\text{EX: } \frac{dy}{dt} = yt*5, \quad y = \exp\left(\frac{t^2}{2}\right) * 5$$

Liken it to *averaging*

$\frac{dy}{dt} = f(y,t)$, with stepsize *h*

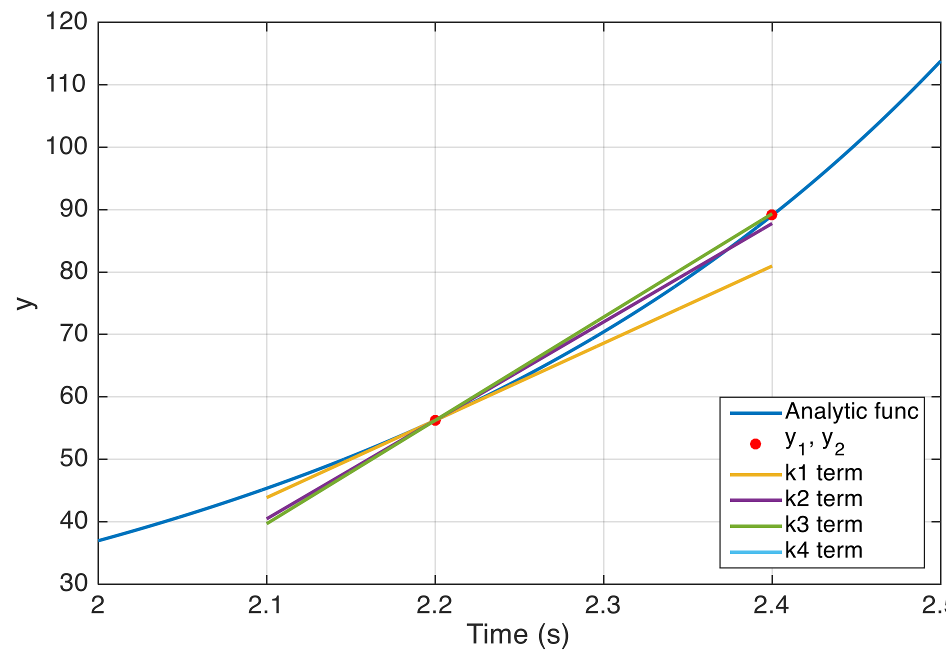
Define:

$$k_1(y_1, t_1) = \frac{dy}{dt} = \mathrm{f}(\mathrm{y}_1, \mathrm{t}_1)$$

$$k_2(y_1, t_1) = f(y_1 + 0.5h\, k_1, t + 0.5h)$$

$$k_3(y_1, t_1) = f(y_1 + 0.5h\, k_2, t + 0.5h)$$

$$k_4(y_1, t_1) = f(y_1 + hk_3, t + h)$$



$$y_2(t_1 + h) = y_1(t_1) + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

$$\text{EX: } \frac{dy}{dt} = yt * 5, \quad y = \exp\left(\frac{t^2}{2}\right) * 5$$

Liken it to *averaging*

$\frac{dy}{dt} = f(y,t)$, with stepsize *h*

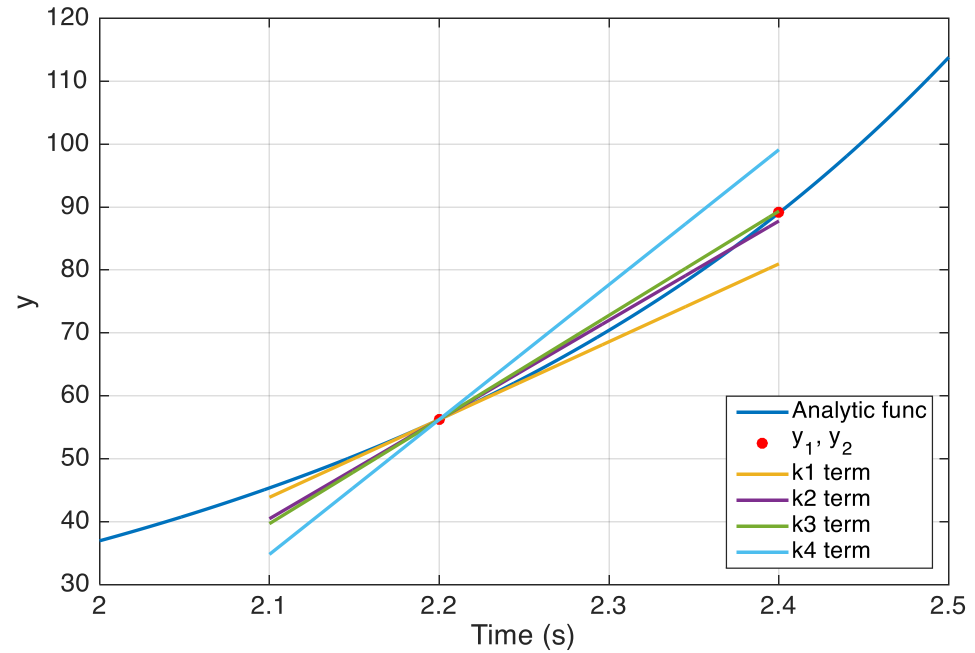
Define:

$$k_1(y_1,t_1) = \frac{dy}{dt} = \mathrm{f}(\mathrm{y}_1,\mathrm{t}_1)$$

$$k_2(y_1,t_1) = f(y_1 + 0.5h\, k_1, t + 0.5h)$$

$$k_3(y_1,t_1) = f(y_1 + 0.5h\, k_2, t + 0.5h)$$

$$k_4(y_1,t_1) = f(y_1 + hk_3, t + h)$$



$$y_2(t_1 + h) = y_1(t_1) + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

Exercise

Code up your own version of RK4 and use it to solve

$$\frac{dy}{dt} = yt * 5, \qquad y_0 = 5$$

For t = [0:0.001:3]; Recall RK4 formula:

$$k_1(y_1, t_1) = \frac{dy}{dt} = \mathrm{f(y_1, t_1)}$$

$$k_2(y_1, t_1) = f(y_1 + 0.5h\, k_1, t + 0.5h)$$

$$k_3(y_1, t_1) = f(y_1 + 0.5h\, k_2, t + 0.5h)$$

$$k_4(y_1, t_1) = f(y_1 + hk_3, t + h)$$

$$y_2(t_1 + h) = y_1(t_1) + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

Second Order ODE

- Consider the equation of motion for a spring system:

$$\frac{d^2S}{dt^2} = -k * S$$

Can we write this as first order ODE(s)?

Exercise

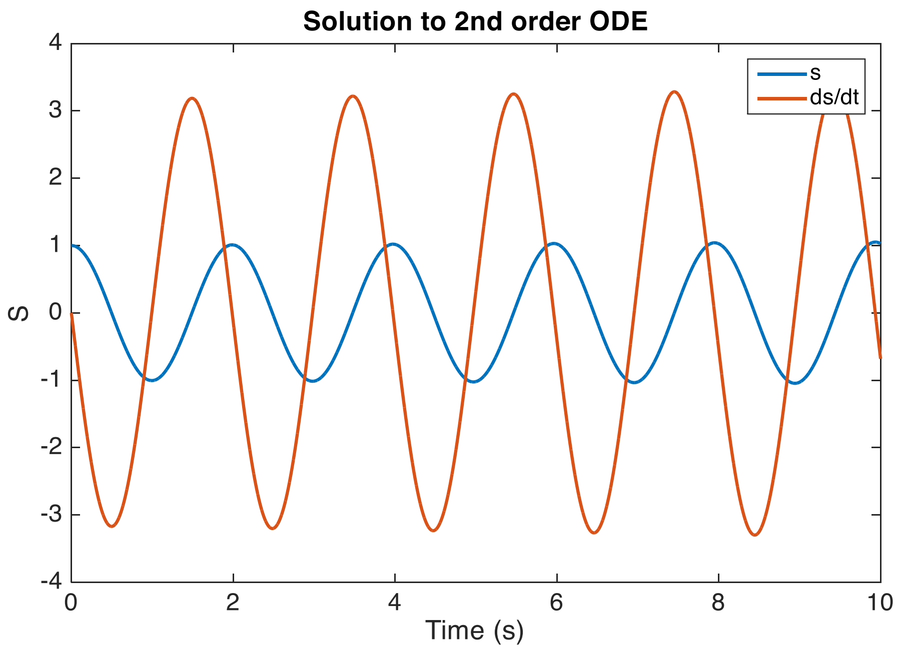
Solve (numerically)

$$\frac{d^2S}{dt^2} = -k * S$$

using Euler's method. Use the following parameter & initial values:

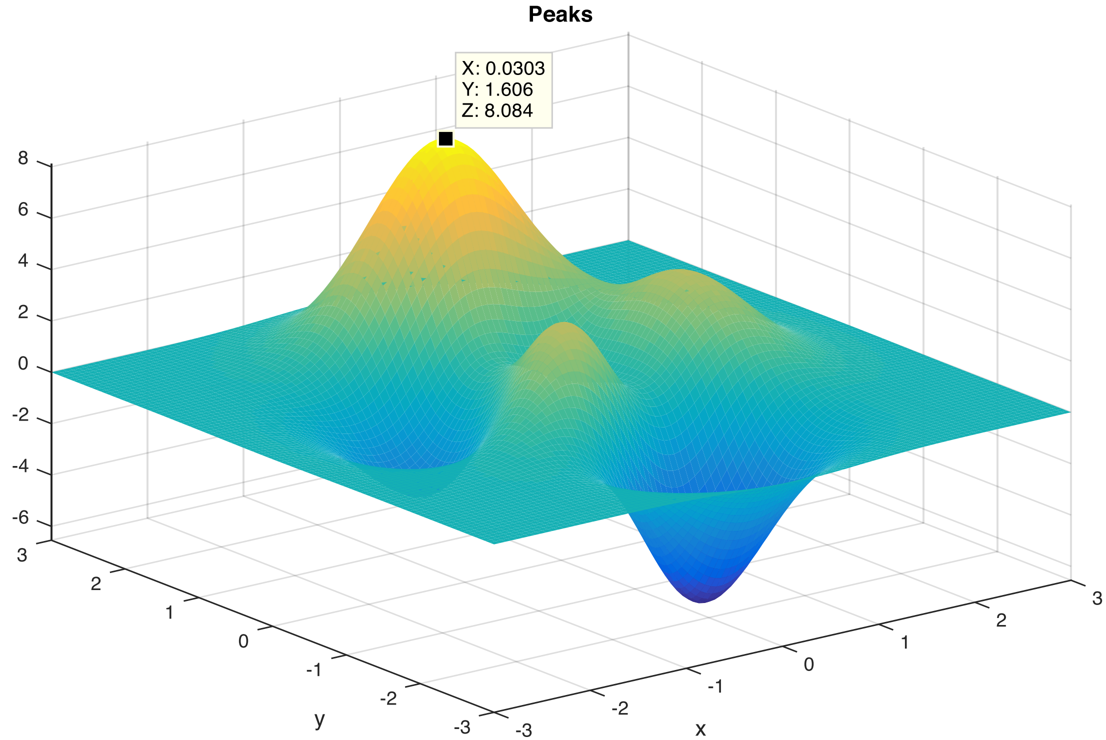
k = 10, S(0) = 1, S'(0) = 0

Plot S(t) as a function of t for t = 0 to 10. Experiment with step size.



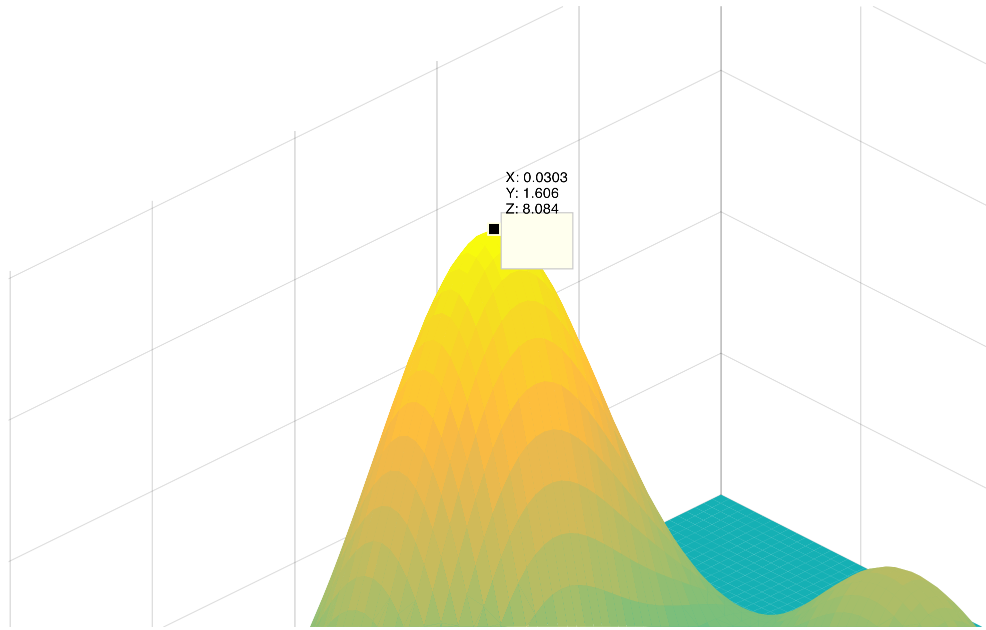
Fun problem: numerical optimization

- Need to get our rover to the bottom of the hill
- Rover can see around itself in all directions, but not very far



Fun problem: numerical optimization *setup_rover.m*

- Need to get our rover to the bottom of the hill
- Rover can see around itself in all directions, but not very far
- Let's say rover can identify "change in height" around it

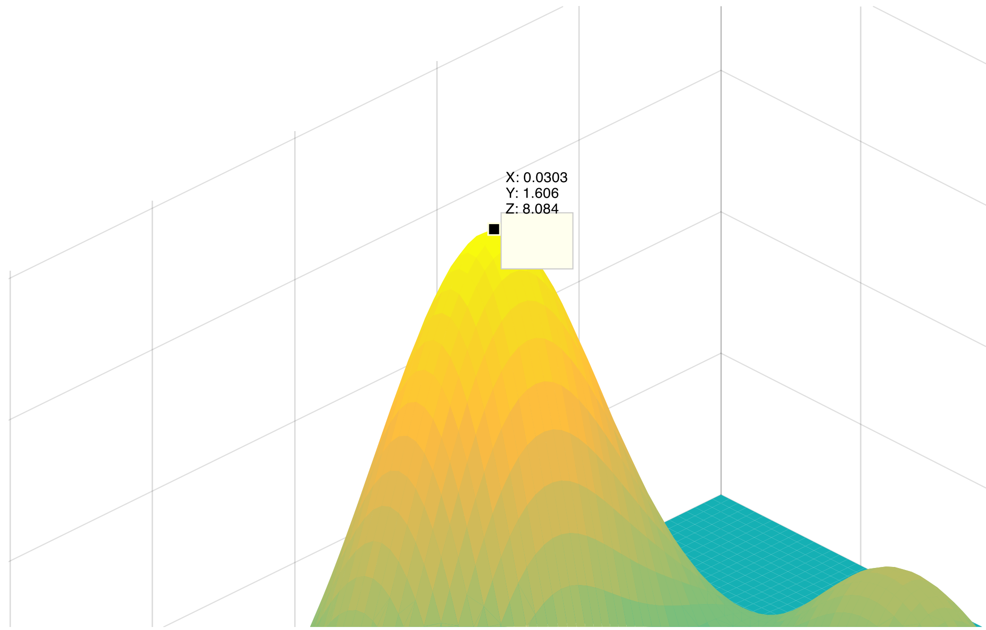


Change in height surrounding rover

	X-1	X+0	X+1
Y+1	-0.02	-0.01	-0.11
Y+0	-0.01	Rover	-0.10
Y-1	-0.06	-0.05	-0.16

Fun problem: numerical optimization *setup_rover.m*

- Need to get our rover to the bottom of the hill
- Rover can see around itself in all directions, but not very far
- Let's say rover can identify "change in height" around it



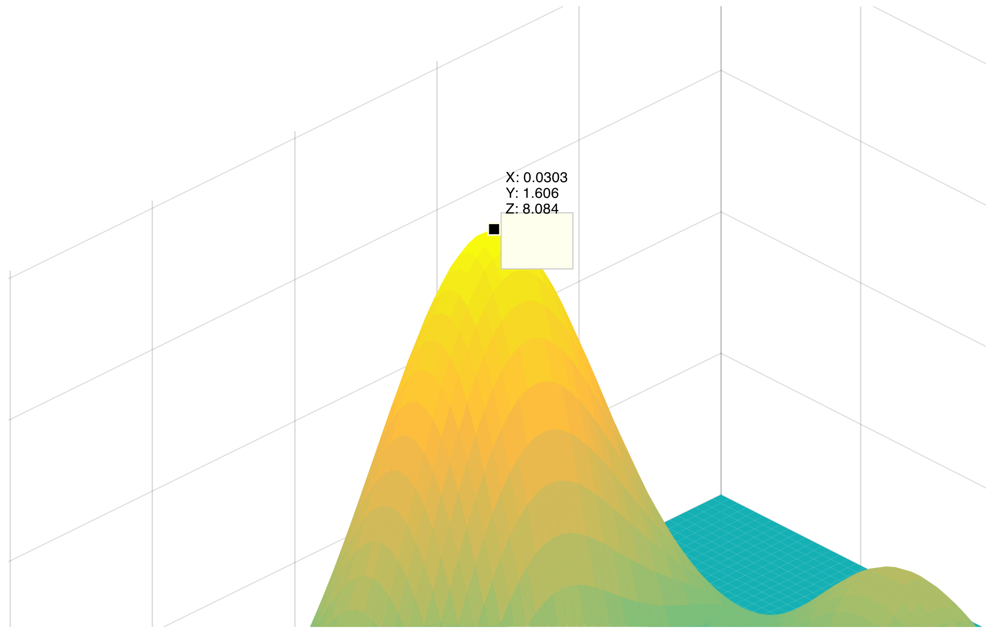
Change in height surrounding rover

	X-1	X+0	X+1
Y+1	-0.02	-0.01	-0.11
Y+0	-0.01	Rover	-0.10
Y-1	-0.06	-0.05	-0.16

- Where should rover move?
- Given x_0 = 75, y_0 = 50, how do we build this matrix?
- How do we make the rover move?

Fun problem: numerical optimization *setup_rover.m*

- Need to get our rover to the bottom of the hill
- Rover can see around itself in all directions, but not very far
- Let's say rover can identify "change in height" around it



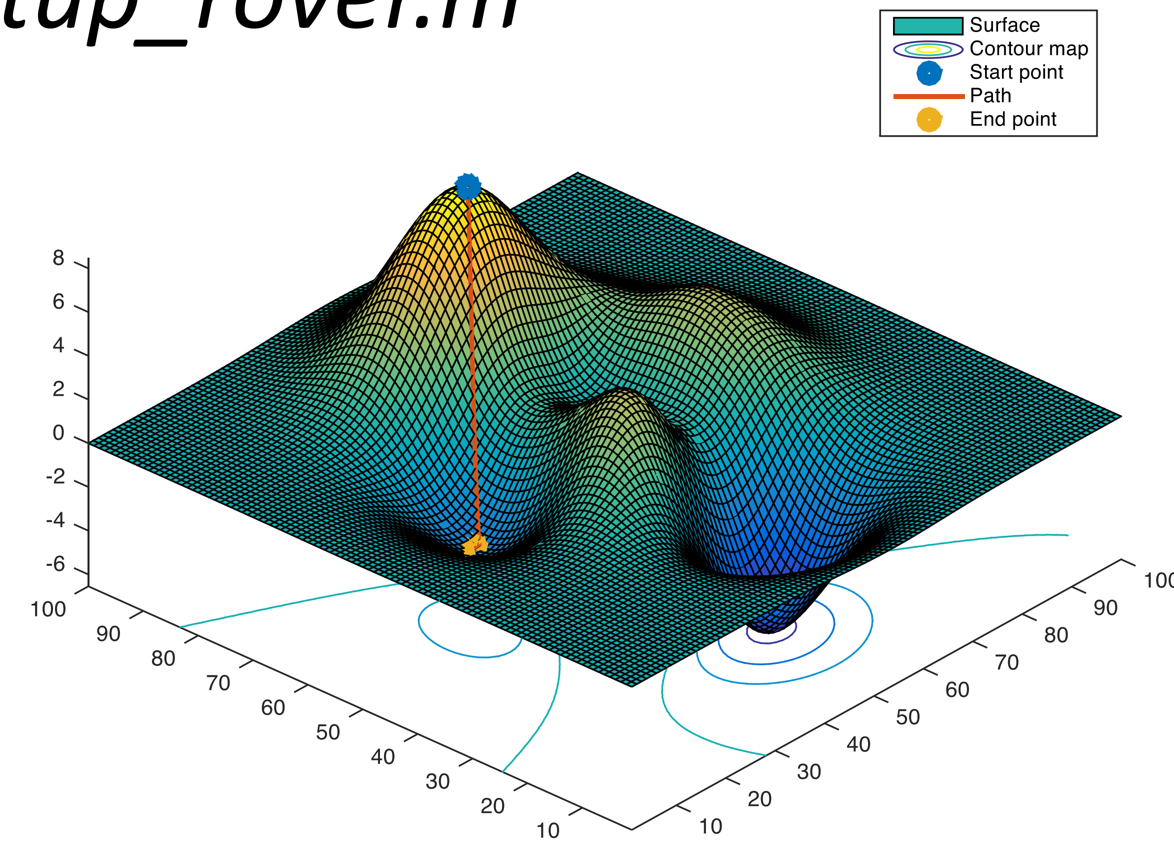
Change in height surrounding rover

	X-1	X+0	X+1
Y+1	-0.02	-0.01	-0.11
Y+0	-0.01	Rover	-0.10
Y-1	-0.06	-0.05	-0.16

- Where should rover move?
- Given x_0 = 75, y_0 = 50, how do we build this matrix?
- How do we make the rover move?

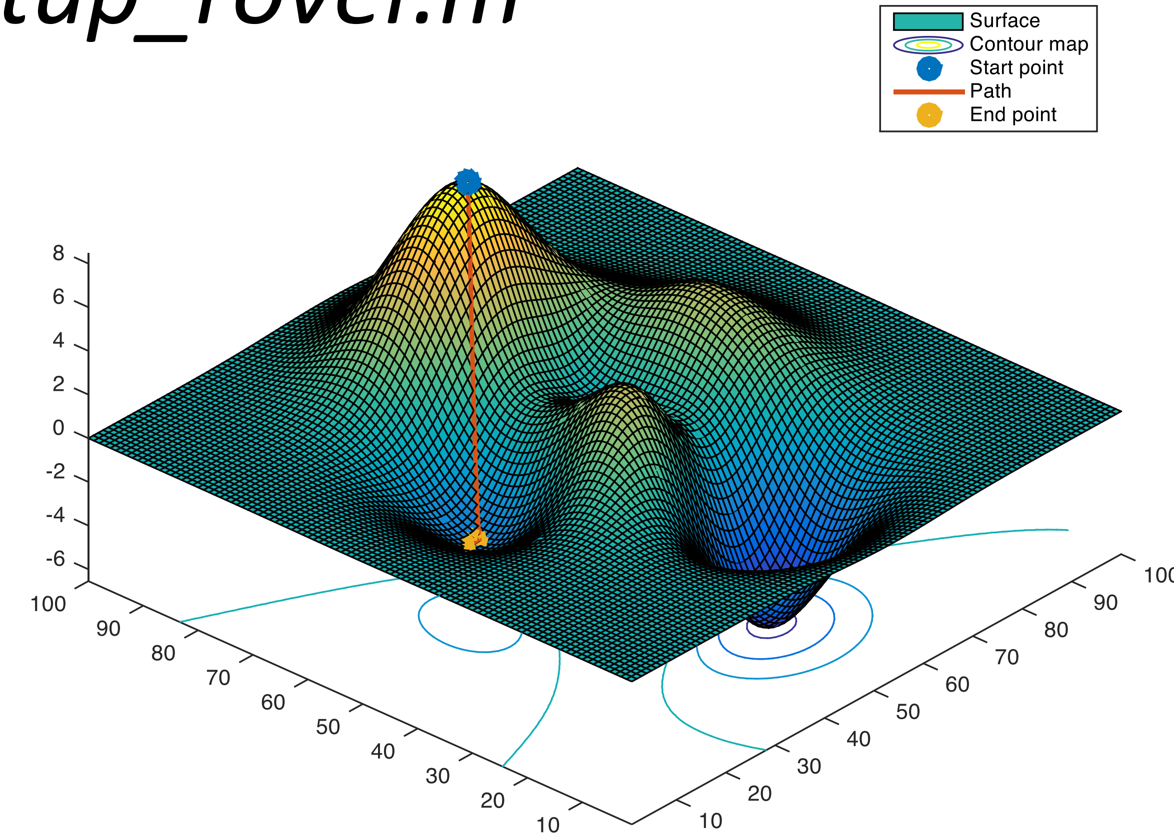
Fun problem: numerical optimization *setup_rover.m*

- Need to get our rover to the bottom of the hill
- We've found the bottom of *a* hill, but not *the* hill!
- What could we do to solve this problem?
- Would more rovers help?
- Should they start at the same position?
- What if our rover was a bit more chaotic?



- Note that this method uses first derivative methods (which we now know how to do!)
- This is called ***STEEPEST DESCENT OPTIMIZATION*** (also ***GRADIENT DESCENT***)

Fun problem: numerical optimization *setup_rover.m*

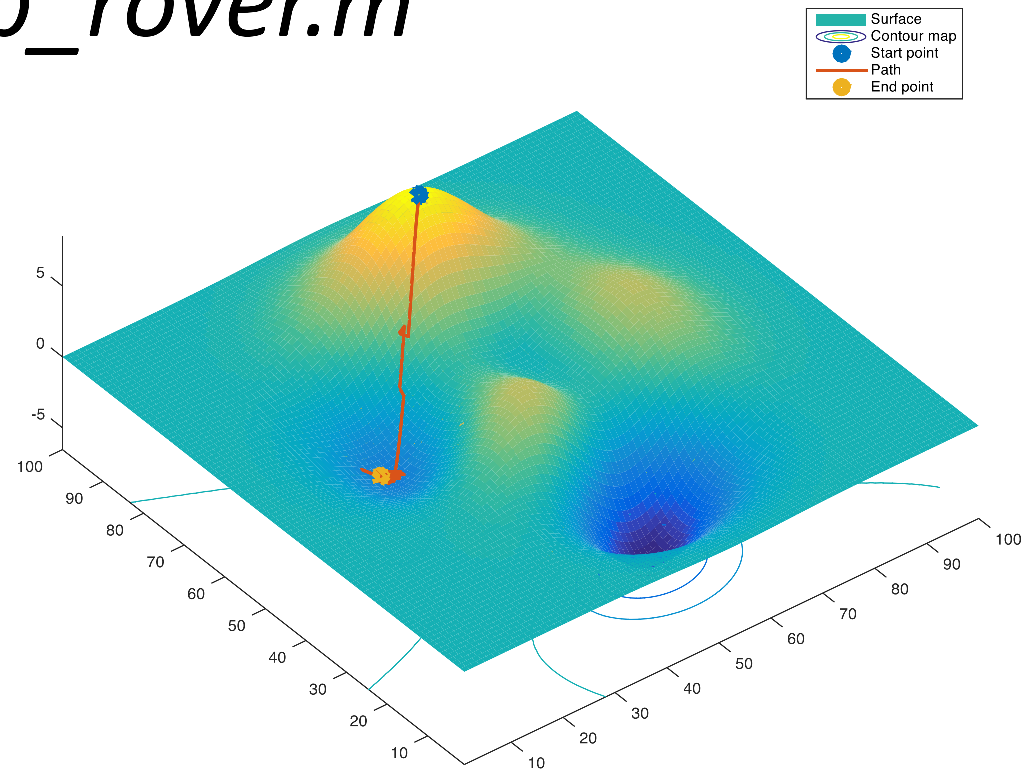


- Need to get our rover to the bottom of the hill
- We've found the bottom of *a* hill, but not *the* hill!
- What could we do to solve this problem?
- Would more rovers help?
- Should they start at the same position?
- What if our rover was a bit more chaotic?

- Note that this method uses first derivative methods (which we now know how to do!)
- This is called *STEEPEST DESCENT OPTIMIZATION* (also *GRADIENT DESCENT*)

Fun problem: numerical optimization

setup_rover.m

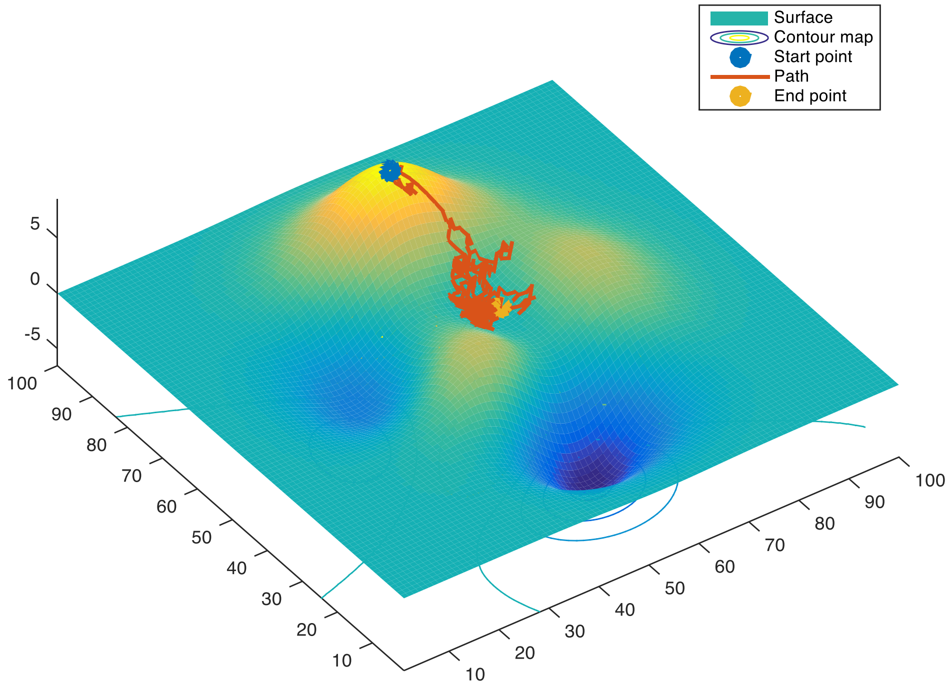


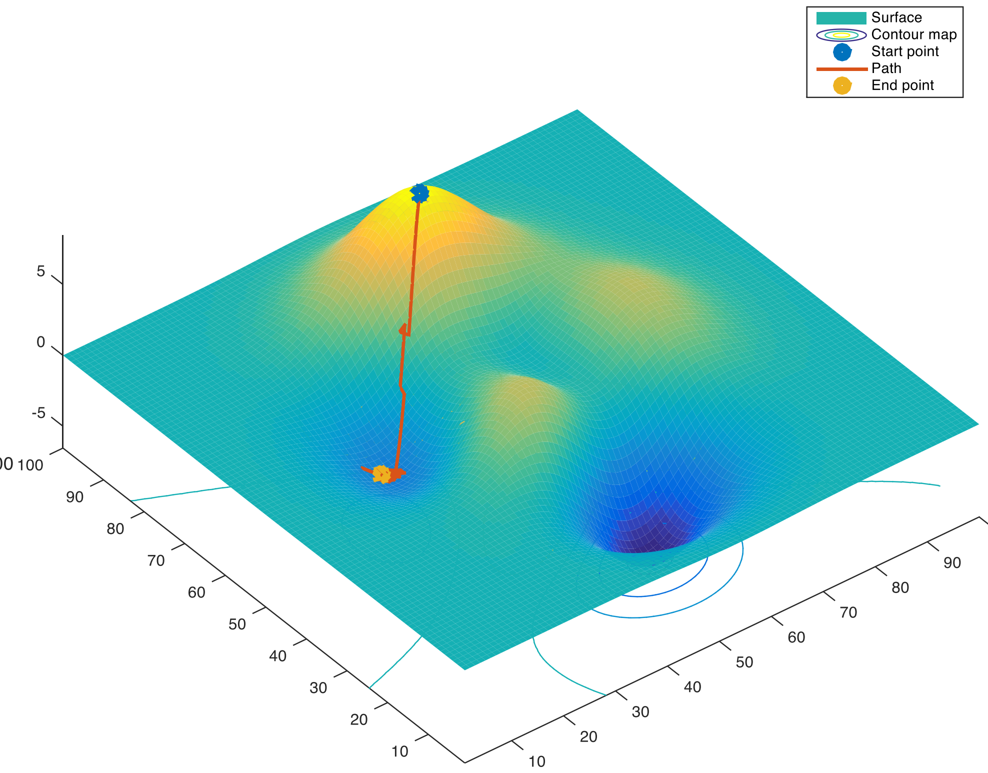
- What if our rover was a bit more chaotic?
  - Every so often, the rover steps in a completely random direction.

	X-1	X+0	X+1
Y+1	-0.02	-0.01	-0.11
Y+0	-0.01	Rover	-0.10
Y-1	-0.06	-0.05	-0.16

```
if round(rand-0.3)
        x_inc = (b2-2);
        y_inc = -(b(b2)-2);
    else
        %Random walk
        x_inc = round(3*rand()-1.5);
        y_inc = round(3*rand()-1.5);
    end
```

Fun problem: numerical optimization
setup_rover.m



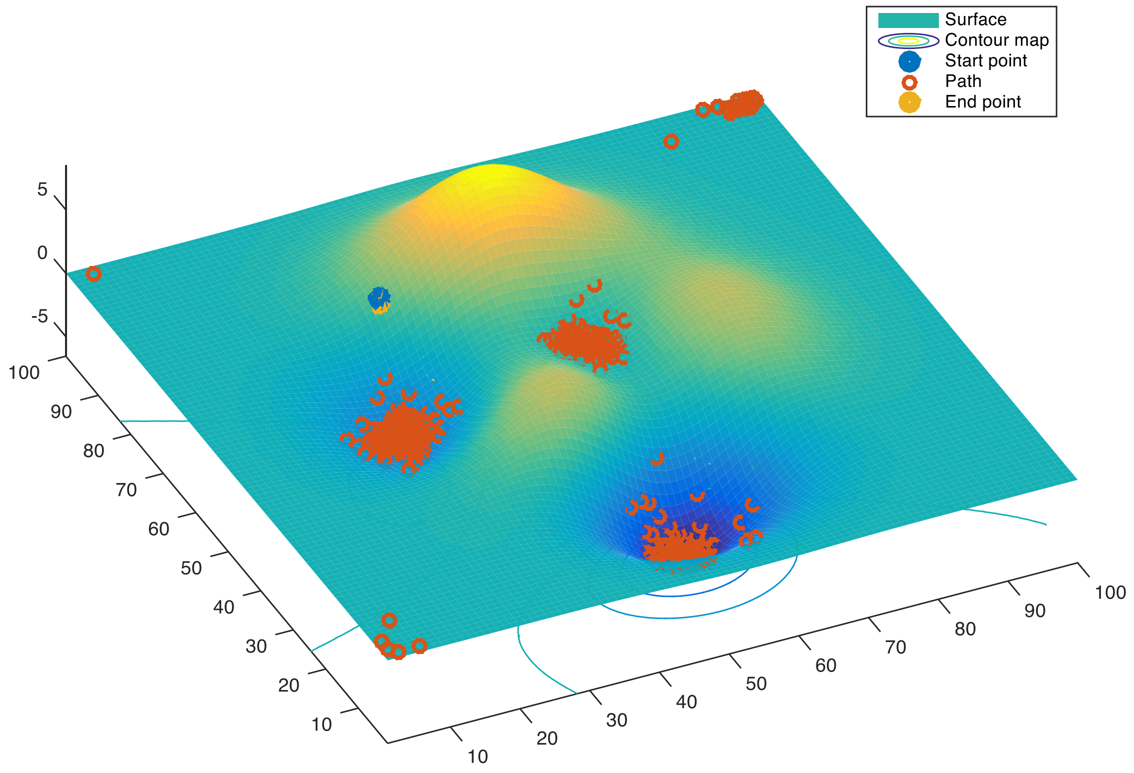


- 85% Chaos
- Now we're getting somewhere!

- 20% Chaos

Fun problem: numerical optimization *setup_rover.m*

- What if we have multiple chaotic rovers?
  - 500 rovers
  - 85% Random walk probability
  - Random starting point



Fun problem: numerical optimization *setup_rover.m*

- What if we have multiple chaotic rovers?
  - 500 rovers
  - 40% Random walk probability
  - Random starting point
  - Look at histogram, we have found the min!