Simulation

Assignment 3

Name: Viviana Márquez
Code: 614132005
Date: Saturday, February 25th, 2017

---

1. Stability

(10 Points) Graph the potential for the following systems and identify all equilibrium points:

• $\dot{x} = 0$

Solution:
We integrate with respect to x in order to find an equation such that:

$$f(x) = -\frac{dV(x)}{dx}, \text{ where } V(x) \text{ is the potential.}$$

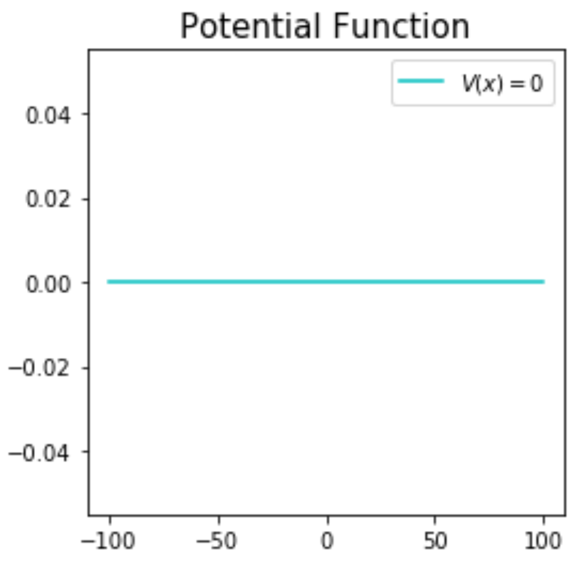
Then,

$$\int 0 \, dx = C$$

Therefore,

$$V(x) = 0$$

In [1]:
```
%matplotlib inline
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
mpl.rc("figure", figsize=(4, 4))

x = np.linspace(-100,100,100)
y = np.zeros(100)

plt.title('Potential Function', fontsize=15)
plt.plot(x,y,'c',label=r'$V(x)=0$')
plt.legend()
plt.show()
```



Let us identify all equilibrium points by:

$$V(x) = 0$$

$$\Rightarrow 0 = 0$$

- $\dot{x} = x^2$

Solution:
We integrate with respect to x in order to find an equation such that:

$$f(x) = -\frac{dV(x)}{dx}, \text{ where } V(x) \text{ is the potential.}$$

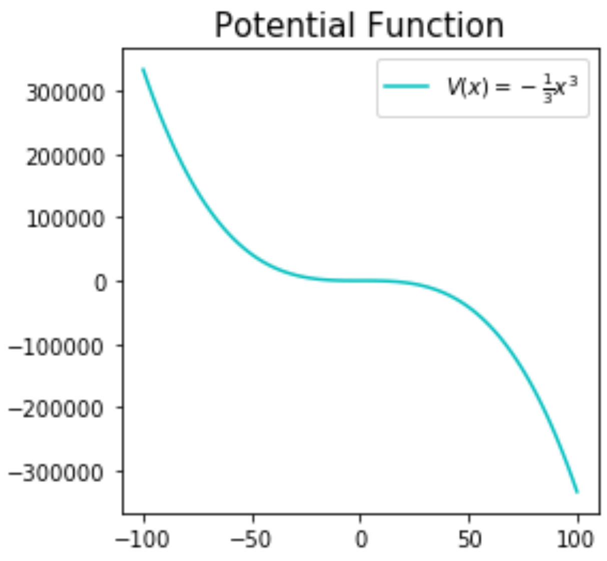
Then,

$$\int x^2\,dx = \frac{1}{3}x^3 + C$$

Therefore,

$$V(x) = -\frac{1}{3}x^3$$

In [2]:
```
x = np.linspace(-100,100,100)
y = (-1/3)*x**3

plt.title('Potential Function', fontsize=15)
plt.plot(x,y,'c',label=r'$V(x)=-\frac{1}{3}x^{3}$')
plt.legend()
plt.show()
```



Let us identify all equilibrium points by:

$$V(x) = 0$$

$$\Rightarrow x = 0$$

- $\dot{x} = x - x^3$

Solution:
We integrate with respect to x in order to find an equation such that:

$$f(x) = -\frac{dV(x)}{dx}, \text{ where } V(x) \text{ is the potential.}$$

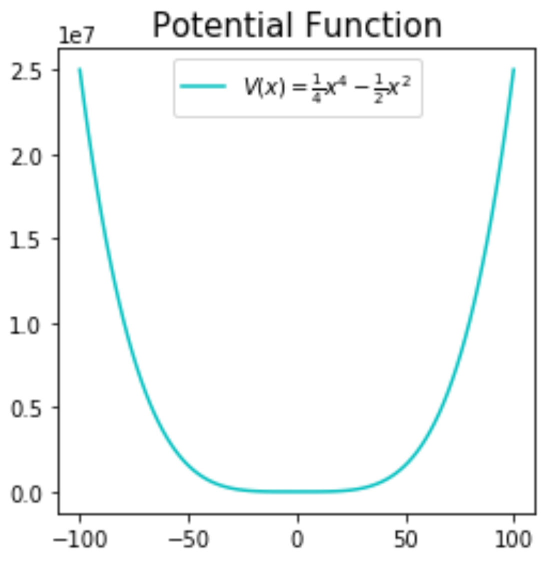
Then,

$$\int x - x^3 \, dx = \frac{1}{2}x^2 - \frac{1}{4}x^4 + C$$

Therefore,

$$V(x) = \frac{1}{4}x^4 - \frac{1}{2}x^2$$

In [3]:
```
x = np.linspace(-100,100,100)
y = ((1/4)*x**4)-((1/2)*x**2)

plt.title('Potential Function', fontsize=15)
plt.plot(x,y,'c',label=r'$V(x)=\frac{1}{4}x^{4} -\frac{1}{2}x^{2}$')
plt.legend()
plt.show()
```



Let us identify all equilibrium points by:

$$V(x) = 0$$

$$\Rightarrow x = 0, \sqrt{2}, -\sqrt{2}$$

2. Python

The following problems are for you to become familiarized with Python's syntax.

#1. (5 Points) Randomly generate two lists and write a program that returns a list that contains only the elements that are common between the lists (without duplicates). Make sure your program works on two lists of different sizes.

In [4]:
```
import random
rand1=[random.randrange(1,51) for _ in range (7)]
rand2=[random.randrange(1,51) for _ in range (13)]
print ("First random list:")
print (rand1)
print ("Second random list:")
print (rand2)
print ("Common elements:")
print(list(set(rand1).intersection(rand2)))
```

```
First random list:
[15, 48, 37, 17, 38, 22, 50]
Second random list:
[1, 13, 36, 6, 17, 32, 4, 48, 6, 38, 29, 11, 16]
Common elements:
[48, 17, 38]
```

#2. (10 Points) Ask the user for a number and determine whether the number is prime or not.

In [5]:
```
number = input('Type a number: ')

def prime(x):
    x = int(x)
    if (x<2):
        return ("%a is not a prime number."%x)
    for i in range(2,x):
        if not x%i:
           return ("%a is not a prime number."%x)
    return ("%a is a prime number."%x)

prime(number)
```

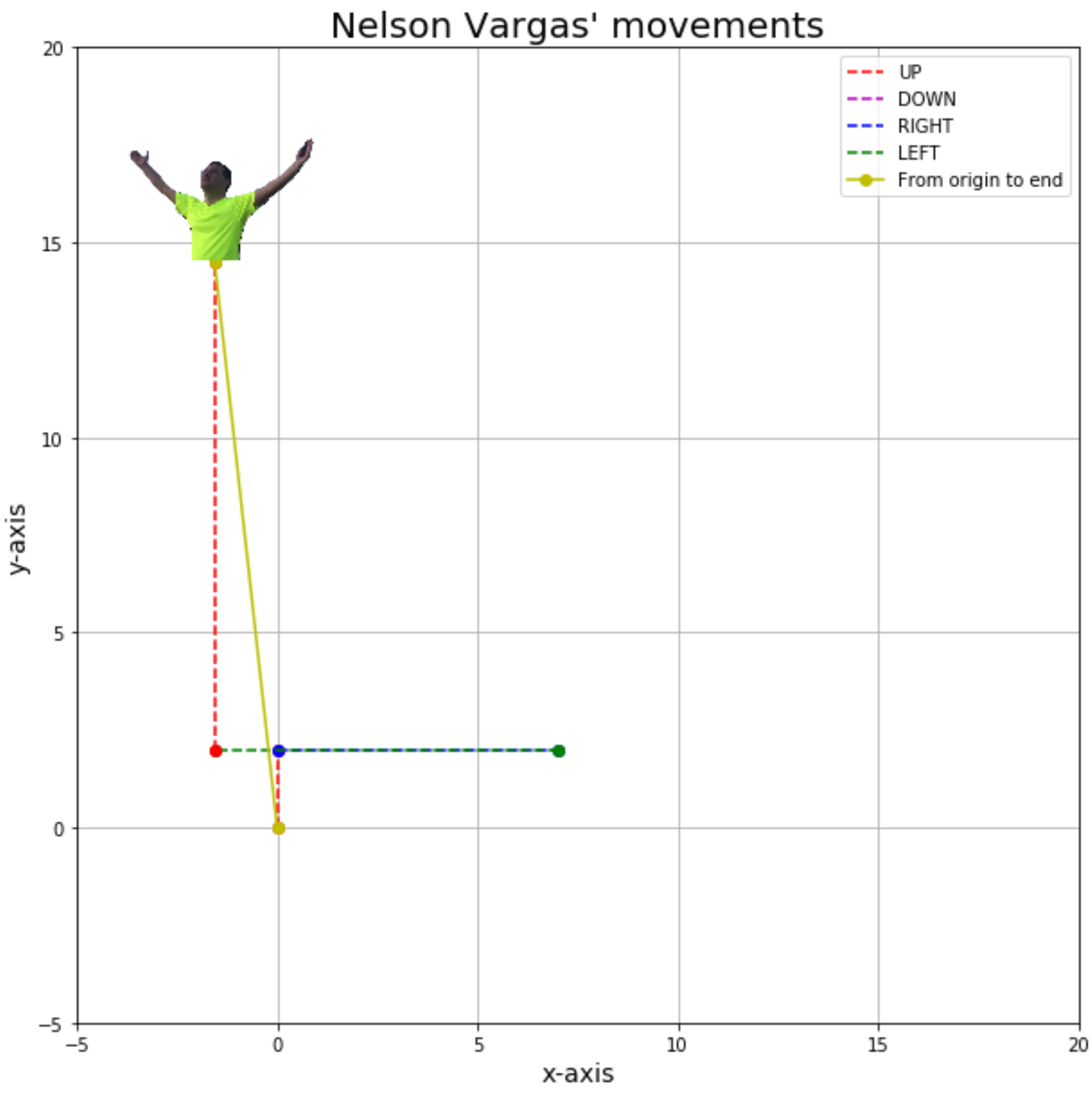
```
Type a number: 1373
```

Out[5]:
```
'1373 is a prime number.'
```

#3. (10 Points) Nelson Vargas moves in a plane starting from the origin. He can only move toward UP, DOWN, LEFT or RIGHT with certain given steps written by the user on a .txt file. Write a program to compute the euclidean distance from the starting point to the end point. If the distance is a float, then return the nearest integer.

In [6]:
```
with open('Data.txt', 'r') as f: #Change here name of the file.
    lines = f.readlines()

xi = 0
yi = 0
xf = 0
yf = 0

for i in range(len(lines)):
    line = lines[i].split(';')
    if line[0]=="UP":
        yf=yf+float(line[1])
    if line[0]=="DOWN":
        yf=yf-float(line[1])
    if line[0]=="RIGHT":
        xf=xf+float(line[1])
    if line[0]=="LEFT":
        xf=xf-float(line[1])

euclidean_distance = np.sqrt((xi-xf)**2+(yi-yf)**2)
mpl.rc("figure", figsize=(10, 10))
```

In [7]:
```
import matplotlib.image as image
from matplotlib._png import read_png
from matplotlib.offsetbox import AnnotationBbox, OffsetImage

xf1 = 0
xf2 = 0
yf1 = 0
yf2 = 0

fig, ax = plt.subplots()

#xg = np.linspace(-100,100,100)
for i in range(len(lines)):
    line = lines[i].split(';')
    if line[0]=="UP":
        yf2=yf2+float(line[1])
        plt.plot([xf1,xf2],[yf1,yf2],'ro--')
        yf1=yf2
    if line[0]=="DOWN":
        yf2=yf2-float(line[1])
        plt.plot([xf1,xf2],[yf1,yf2],'mo--')
        yf1=yf2
    if line[0]=="RIGHT":
        xf2=xf2+float(line[1])
        plt.plot([xf1,xf2],[yf1,yf2],'bo--')
        xf1=xf2
    if line[0]=="LEFT":
        xf2=xf2-float(line[1])
        plt.plot([xf1,xf2],[yf1,yf2],'go--')
        xf1=xf2
plt.title("Nelson Vargas' movements", fontsize=20)
plt.xlabel('x-axis', fontsize=14)
plt.ylabel('y-axis', fontsize=14)

up = plt.plot([0,0],[0,0], 'r--',label="UP")
down = plt.plot([0,0],[0,0], 'm--',label="DOWN")
right = plt.plot([0,0],[0,0], 'b--',label="RIGHT")
left = plt.plot([0,0],[0,0], 'g--',label="LEFT")
plt.plot([0,xf2],[0,yf2],'yo-', label='From origin to end')

arr_hand = read_png('Nelson.png')
imagebox = OffsetImage(arr_hand, zoom=0.2)
xy = [xf2+0.2, yf2+1.6]

ab = AnnotationBbox(imagebox, xy,
    xycoords='data',
    boxcoords="data",
    frameon=False)
ax.add_artist(ab)

axes = plt.gca()
axes.set_xlim([-5,20])
axes.set_ylim([-5,20])
mpl.rc("figure", figsize=(10, 10))

plt.legend()
plt.grid()
plt.show()

print ("Nelson Vargas moved an euclidean distance of %a." %int(round(e
uclidean_distance)))
```



```
Nelson Vargas moved an euclidean distance of 15.
```

3. Symbolic Computation

The whole idea of this exercise is to use the computer, not solving the problems by hand.

• (5 Points) Solve the following differential equation:

$$x\frac{df(x)}{dx} + f(x) - f(x)^2 = 0$$

```
In [8]: from sympy import *
        x, y = symbols('x y')
        init_printing(use_unicode=True)

        f = Function('f')
        dsolve(Eq(x*Derivative(f(x)) + (f(x)) - (f(x)) **2, 0))
```

Out[8]: $$f(x) = -\frac{C_1}{-C_1 + x}$$

```
In [9]: C1 = symbols('C1')
        print(checkodesol(Eq(x*Derivative(f(x)) + (f(x)) - (f(x)) **2, 0), Eq(
        f(x), -C1/(-C1+x)))) #My answer
        print(checkodesol(Eq(x*Derivative(f(x)) + (f(x)) - (f(x)) **2, 0), Eq(
        f(x), 1/(C1*x+1)))) #WolframAlpha

        (True, 0)
        (True, 0)
```

- (5 Points) Solve the following system of equations:

$$\begin{aligned} x + y &= 2 \\ 2x + y &= 3 \end{aligned}$$

```
In [10]: solve([Eq(x + y, 2), Eq(2*x+y, 3)], [x, y])
```

Out[10]: $\{x : 1, \quad y : 1\}$

- (5 Points) Find the eigenvalues and eigenvectors of the following matrices:

$$A_1 = \begin{pmatrix} 0,8 & 0,3 \\ 0,2 & 0,7 \end{pmatrix}$$

```
In [11]: M1 = Matrix([[0.8,0.3], [0.2,0.7]])
         M1.eigenvals()
```

Out[11]: $\left\{\frac{1}{2} : 1, \quad 1 : 1\right\}$

$$A_2 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

```
In [12]: M2 = Matrix([[0,1], [1,0]])
         M2.eigenvals()
```

Out[12]: $\{-1 : 1, \quad 1 : 1\}$