mcpp_taller6_john_caro

September 28, 2016

1 Taller 6

Métodos Computacionales para Políticas Públicas - URosario

Entrega: viernes 30-sep-2016 11:59 PM

[John Alexander Caro Becerra] [Jhonalexbc@gmail.com]

1.1 Instrucciones:

- Guarde una copia de este *Jupyter Notebook* en su computador, idealmente en una carpeta destinada al material del curso.
- Modifique el nombre del archivo del *notebook*, agregando al final un guión inferior y su nombre y apellido, separados estos últimos por otro guión inferior. Por ejemplo, mi *notebook* se llamaría: mcpp_taller6_santiago_matallana
- Marque el *notebook* con su nombre y e-mail en el bloque verde arriba. Reemplace el texto "[Su nombre acá]" con su nombre y apellido. Similar para su e-mail.
- Desarrolle la totalidad del taller sobre este *notebook*, insertando las celdas que sea necesario debajo de cada pregunta. Haga buen uso de las celdas para código y de las celdas tipo *markdown* según el caso.
- Recuerde salvar periódicamente sus avances.
- Cuando termine el taller:

    1. Descárguelo en PDF. Si tiene algún problema con la conversión, descárguelo en HTML.
    2. Suba todos los archivos a su repositorio en GitHub, en una carpeta destinada exclusivamente para este taller, antes de la fecha y hora límites.

(Todos los ejercicios tienen el mismo valor.)

1.1.1 Resuelva las partes 1 y 3 de este documento.

```
In [228]: import numpy as np
          import scipy.linalg as la
          import matplotlib.pyplot as plt
          import math as math
```

2 1.1

```
In [202]: x = np.array(2)
          print(x)
```

```
2
```

3 1.2

```
In [203]: print(x*x) #comando para elevar al cuadrado
```

```
4
```

```
In [204]: print(x*x*x) #comando para elevar al cubo
```

```
8
```

4 1.3 y 1.4

```
In [205]: theta = np.array(90)
```

```
In [206]: print(np.sin(theta)) #comando para calcular el seno de theta
```

```
0.893996663601
```

```
In [207]: print(np.cos(theta)) #comando para calcular el coseno de theta
```

```
-0.448073616129
```

Los ángulos que se ingresan en np.cos() y en np.sin() se ingresan en radianes, es decir, estos comandos usan radianes

5 1.5 y 1.6

```
In [208]: meshPoints = np.linspace(-1,1,500)
```

```
In [209]: meshPoints
```

```
Out[209]: array([-1.        , -0.99599198, -0.99198397, -0.98797595, -0.98396794,
                 -0.97995992, -0.9759519 , -0.97194389, -0.96793587, -0.96392786,
                 -0.95991984, -0.95591182, -0.95190381, -0.94789579, -0.94388778,
                 -0.93987976, -0.93587174, -0.93186373, -0.92785571, -0.9238477 ,
                 -0.91983968, -0.91583166, -0.91182365, -0.90781563, -0.90380762,
```

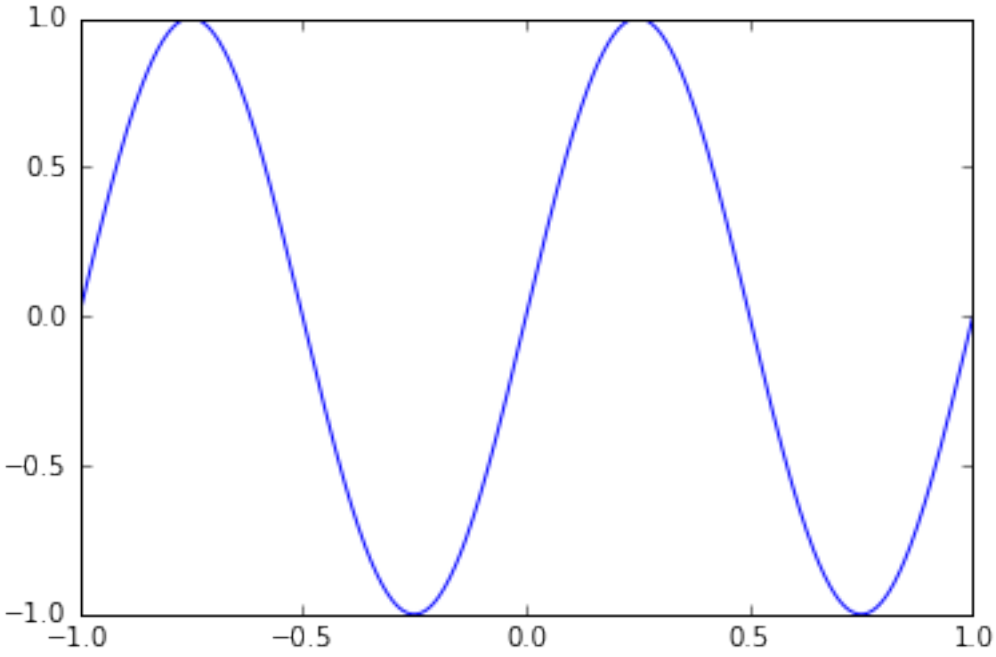
```
-0.8997996 , -0.89579158, -0.89178357, -0.88777555, -0.88376754,
-0.87975952, -0.8757515 , -0.87174349, -0.86773547, -0.86372745,
-0.85971944, -0.85571142, -0.85170341, -0.84769539, -0.84368737,
-0.83967936, -0.83567134, -0.83166333, -0.82765531, -0.82364729,
-0.81963928, -0.81563126, -0.81162325, -0.80761523, -0.80360721,
-0.7995992 , -0.79559118, -0.79158317, -0.78757515, -0.78356713,
-0.77955912, -0.7755511 , -0.77154309, -0.76753507, -0.76352705,
-0.75951904, -0.75551102, -0.75150301, -0.74749499, -0.74348697,
-0.73947896, -0.73547094, -0.73146293, -0.72745491, -0.72344689,
-0.71943888, -0.71543086, -0.71142285, -0.70741483, -0.70340681,
-0.6993988 , -0.69539078, -0.69138277, -0.68737475, -0.68336673,
-0.67935872, -0.6753507 , -0.67134269, -0.66733467, -0.66332665,
-0.65931864, -0.65531062, -0.65130261, -0.64729459, -0.64328657,
-0.63927856, -0.63527054, -0.63126253, -0.62725451, -0.62324649,
-0.61923848, -0.61523046, -0.61122244, -0.60721443, -0.60320641,
-0.5991984 , -0.59519038, -0.59118236, -0.58717435, -0.58316633,
-0.57915832, -0.5751503 , -0.57114228, -0.56713427, -0.56312625,
-0.55911824, -0.55511022, -0.5511022 , -0.54709419, -0.54308617,
-0.53907816, -0.53507014, -0.53106212, -0.52705411, -0.52304609,
-0.51903808, -0.51503006, -0.51102204, -0.50701403, -0.50300601,
-0.498998  , -0.49498998, -0.49098196, -0.48697395, -0.48296593,
-0.47895792, -0.4749499 , -0.47094188, -0.46693387, -0.46292585,
-0.45891784, -0.45490982, -0.4509018 , -0.44689379, -0.44288577,
-0.43887776, -0.43486974, -0.43086172, -0.42685371, -0.42284569,
-0.41883768, -0.41482966, -0.41082164, -0.40681363, -0.40280561,
-0.3987976 , -0.39478958, -0.39078156, -0.38677355, -0.38276553,
-0.37875752, -0.3747495 , -0.37074148, -0.36673347, -0.36272545,
-0.35871743, -0.35470942, -0.3507014 , -0.34669339, -0.34268537,
-0.33867735, -0.33466934, -0.33066132, -0.32665331, -0.32264529,
-0.31863727, -0.31462926, -0.31062124, -0.30661323, -0.30260521,
-0.29859719, -0.29458918, -0.29058116, -0.28657315, -0.28256513,
-0.27855711, -0.2745491 , -0.27054108, -0.26653307, -0.26252505,
-0.25851703, -0.25450902, -0.250501  , -0.24649299, -0.24248497,
-0.23847695, -0.23446894, -0.23046092, -0.22645291, -0.22244489,
-0.21843687, -0.21442886, -0.21042084, -0.20641283, -0.20240481,
-0.19839679, -0.19438878, -0.19038076, -0.18637275, -0.18236473,
-0.17835671, -0.1743487 , -0.17034068, -0.16633267, -0.16232465,
-0.15831663, -0.15430862, -0.1503006 , -0.14629259, -0.14228457,
-0.13827655, -0.13426854, -0.13026052, -0.12625251, -0.12224449,
-0.11823647, -0.11422846, -0.11022044, -0.10621242, -0.10220441,
-0.09819639, -0.09418838, -0.09018036, -0.08617234, -0.08216433,
-0.07815631, -0.0741483 , -0.07014028, -0.06613226, -0.06212425,
-0.05811623, -0.05410822, -0.0501002 , -0.04609218, -0.04208417,
-0.03807615, -0.03406814, -0.03006012, -0.0260521 , -0.02204409,
-0.01803607, -0.01402806, -0.01002004, -0.00601202, -0.00200401,
 0.00200401,  0.00601202,  0.01002004,  0.01402806,  0.01803607,
 0.02204409,  0.0260521 ,  0.03006012,  0.03406814,  0.03807615,
 0.04208417,  0.04609218,  0.0501002 ,  0.05410822,  0.05811623,
```

```
0.06212425,  0.06613226,  0.07014028,  0.0741483 ,  0.07815631,
0.08216433,  0.08617234,  0.09018036,  0.09418838,  0.09819639,
0.10220441,  0.10621242,  0.11022044,  0.11422846,  0.11823647,
0.12224449,  0.12625251,  0.13026052,  0.13426854,  0.13827655,
0.14228457,  0.14629259,  0.1503006 ,  0.15430862,  0.15831663,
0.16232465,  0.16633267,  0.17034068,  0.1743487 ,  0.17835671,
0.18236473,  0.18637275,  0.19038076,  0.19438878,  0.19839679,
0.20240481,  0.20641283,  0.21042084,  0.21442886,  0.21843687,
0.22244489,  0.22645291,  0.23046092,  0.23446894,  0.23847695,
0.24248497,  0.24649299,  0.250501  ,  0.25450902,  0.25851703,
0.26252505,  0.26653307,  0.27054108,  0.2745491 ,  0.27855711,
0.28256513,  0.28657315,  0.29058116,  0.29458918,  0.29859719,
0.30260521,  0.30661323,  0.31062124,  0.31462926,  0.31863727,
0.32264529,  0.32665331,  0.33066132,  0.33466934,  0.33867735,
0.34268537,  0.34669339,  0.3507014 ,  0.35470942,  0.35871743,
0.36272545,  0.36673347,  0.37074148,  0.3747495 ,  0.37875752,
0.38276553,  0.38677355,  0.39078156,  0.39478958,  0.3987976 ,
0.40280561,  0.40681363,  0.41082164,  0.41482966,  0.41883768,
0.42284569,  0.42685371,  0.43086172,  0.43486974,  0.43887776,
0.44288577,  0.44689379,  0.4509018 ,  0.45490982,  0.45891784,
0.46292585,  0.46693387,  0.47094188,  0.4749499 ,  0.47895792,
0.48296593,  0.48697395,  0.49098196,  0.49498998,  0.498998  ,
0.50300601,  0.50701403,  0.51102204,  0.51503006,  0.51903808,
0.52304609,  0.52705411,  0.53106212,  0.53507014,  0.53907816,
0.54308617,  0.54709419,  0.5511022 ,  0.55511022,  0.55911824,
0.56312625,  0.56713427,  0.57114228,  0.5751503 ,  0.57915832,
0.58316633,  0.58717435,  0.59118236,  0.59519038,  0.5991984 ,
0.60320641,  0.60721443,  0.61122244,  0.61523046,  0.61923848,
0.62324649,  0.62725451,  0.63126253,  0.63527054,  0.63927856,
0.64328657,  0.64729459,  0.65130261,  0.65531062,  0.65931864,
0.66332665,  0.66733467,  0.67134269,  0.6753507 ,  0.67935872,
0.68336673,  0.68737475,  0.69138277,  0.69539078,  0.6993988 ,
0.70340681,  0.70741483,  0.71142285,  0.71543086,  0.71943888,
0.72344689,  0.72745491,  0.73146293,  0.73547094,  0.73947896,
0.74348697,  0.74749499,  0.75150301,  0.75551102,  0.75951904,
0.76352705,  0.76753507,  0.77154309,  0.7755511 ,  0.77955912,
0.78356713,  0.78757515,  0.79158317,  0.79559118,  0.7995992 ,
0.80360721,  0.80761523,  0.81162325,  0.81563126,  0.81963928,
0.82364729,  0.82765531,  0.83166333,  0.83567134,  0.83967936,
0.84368737,  0.84769539,  0.85170341,  0.85571142,  0.85971944,
0.86372745,  0.86773547,  0.87174349,  0.8757515 ,  0.87975952,
0.88376754,  0.88777555,  0.89178357,  0.89579158,  0.8997996 ,
0.90380762,  0.90781563,  0.91182365,  0.91583166,  0.91983968,
0.9238477 ,  0.92785571,  0.93186373,  0.93587174,  0.93987976,
0.94388778,  0.94789579,  0.95190381,  0.95591182,  0.95991984,
0.96392786,  0.96793587,  0.97194389,  0.9759519 ,  0.97995992,
0.98396794,  0.98797595,  0.99198397,  0.99599198,  1.        ])
```

```
In [210]: meshPoints[52] #elemento 53a de meshPoimts

Out[210]: -0.79158316633266534
```

6 1.7

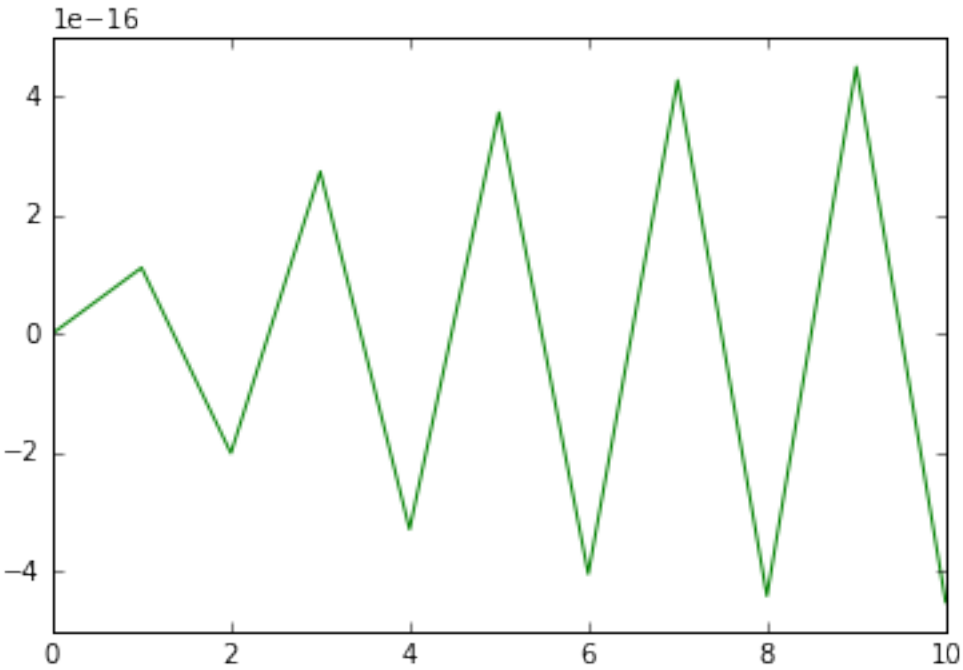
```
In [211]: %matplotlib inline

          plt.plot(meshPoints,np.sin(2*math.pi*meshPoints))
          plt.savefig("sinusoid on the interval [−1, 1].jpg")
```



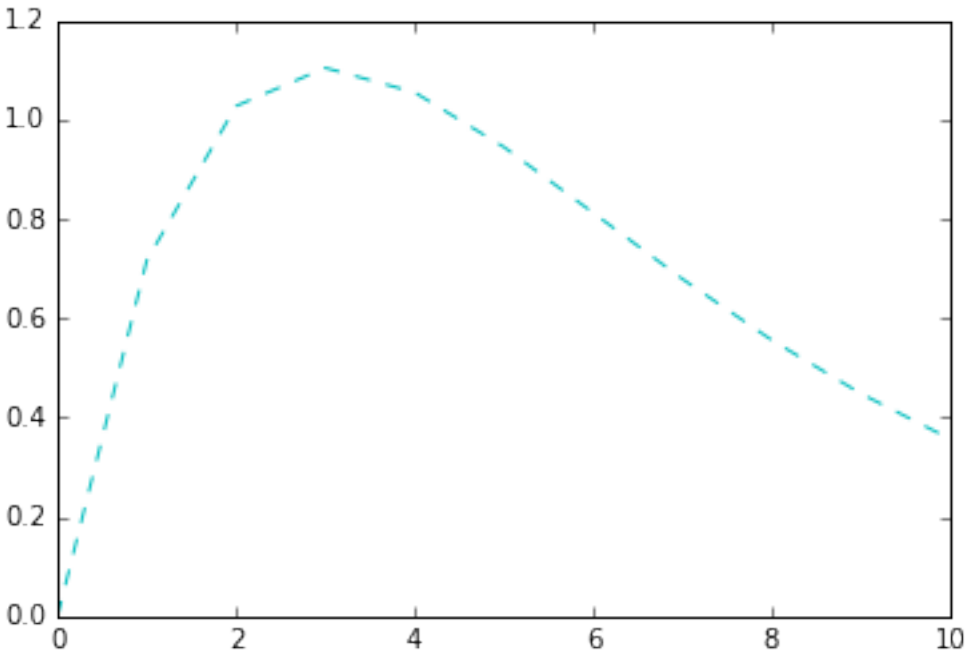
7 3.1

```
In [212]: def f(x):
              f = (math.e**(-(x/10)))*(np.sin(math.pi*x))
              return f

          def g(x):
              g = (x*(math.e**(-(x/3))))
              return g

In [213]: x = np.array([0,1,2,3,4,5,6,7,8,9,10])
          f(x)
```

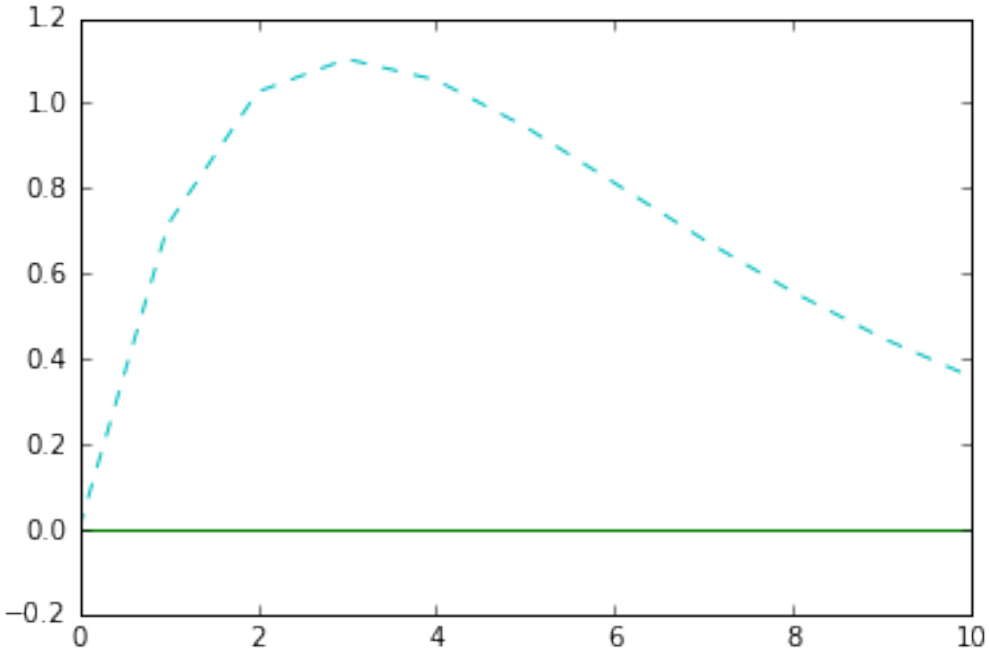
```
Out[213]: array([  0.00000000e+00,   1.10810625e-16,  -2.00531199e-16,
                   2.72172199e-16,  -3.28362120e-16,   3.71392916e-16,
                  -4.03260248e-16,   4.25699122e-16,  -4.40215422e-16,
                   4.48113809e-16,  -4.50522380e-16])

In [214]: g(x)

Out[214]: array([ 0.        ,  0.71653131,  1.02683424,  1.10363832,  1.05438855,
                  0.94437801,  0.8120117 ,  0.67880378,  0.55586761,  0.44808362,
                  0.35673993])

In [215]: plt.plot(x,f(x), "g");
          plt.savefig("función f(x).jpg")
```




```
In [216]: plt.plot(x, g(x), "c--");
          plt.savefig("función g(x).jpg")
```

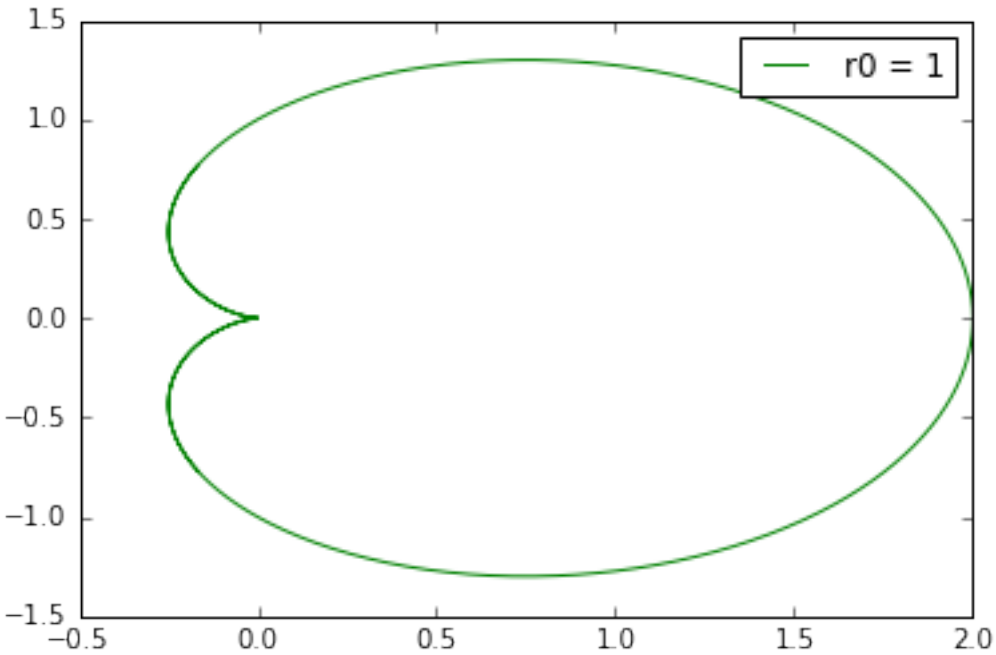



```
In [217]: plt.plot(x,f(x), "g", x, g(x), "c--");
          plt.savefig("funciones f(x) y g(x).jpg")
```



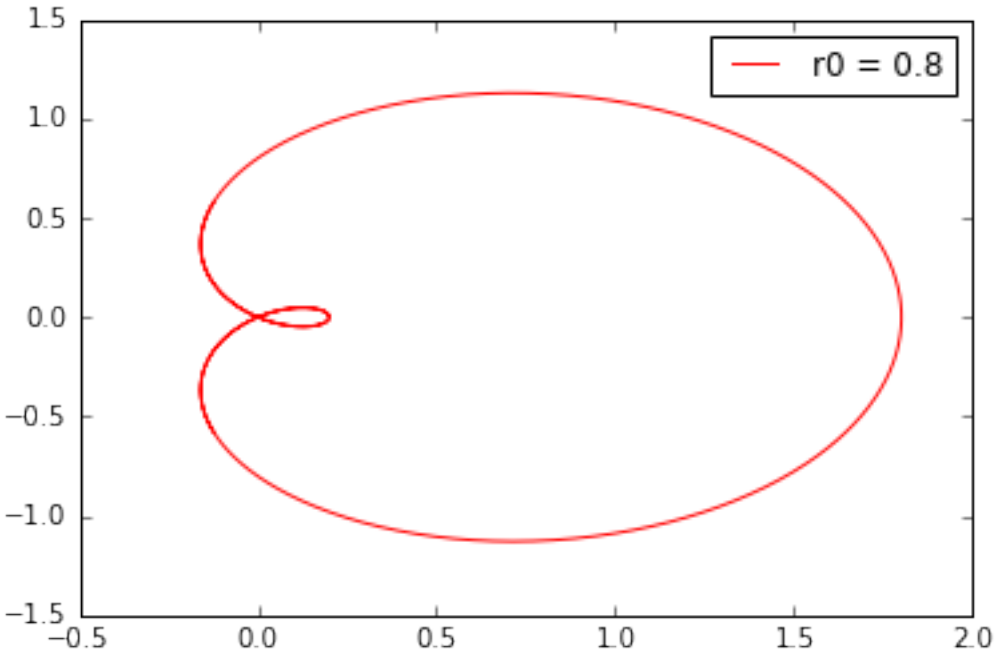
8 3.2

```
In [218]: def limacon(z,theta):
              r = z + np.cos(theta)
              x = r * np.cos(theta)
              y = r * np.sin(theta)
              return r,x,y

In [219]: z = np.array([1])
          theta = np.linspace(-4.5,4.5,2000)

In [220]: limacons = limacon(z,theta)

In [221]: xs1 = limacons[1]
          ys1 = limacons[2]

In [222]: plt.plot(xs1,ys1,"g",label="r0 = 1")
          plt.legend()
          plt.savefig("limaco con ro = 1.jpg")
```

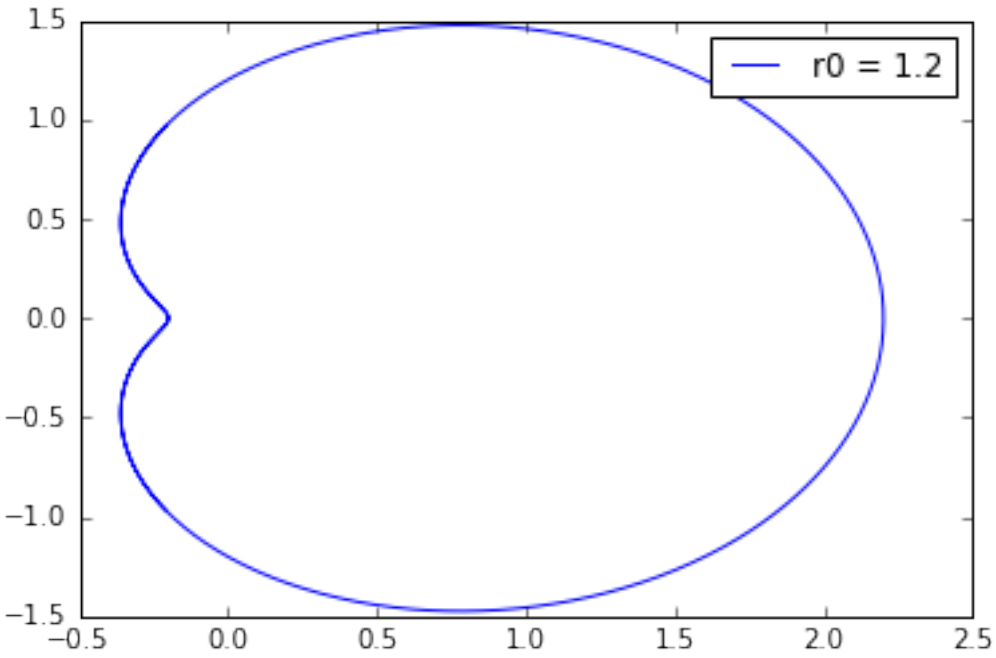



```
In [223]: z = np.array([0.8])
          theta = np.linspace(-4.5,4.5,2000)
          limacons = limacon(z,theta)
          xs2 = limacons[1]
          ys2 = limacons[2]
```

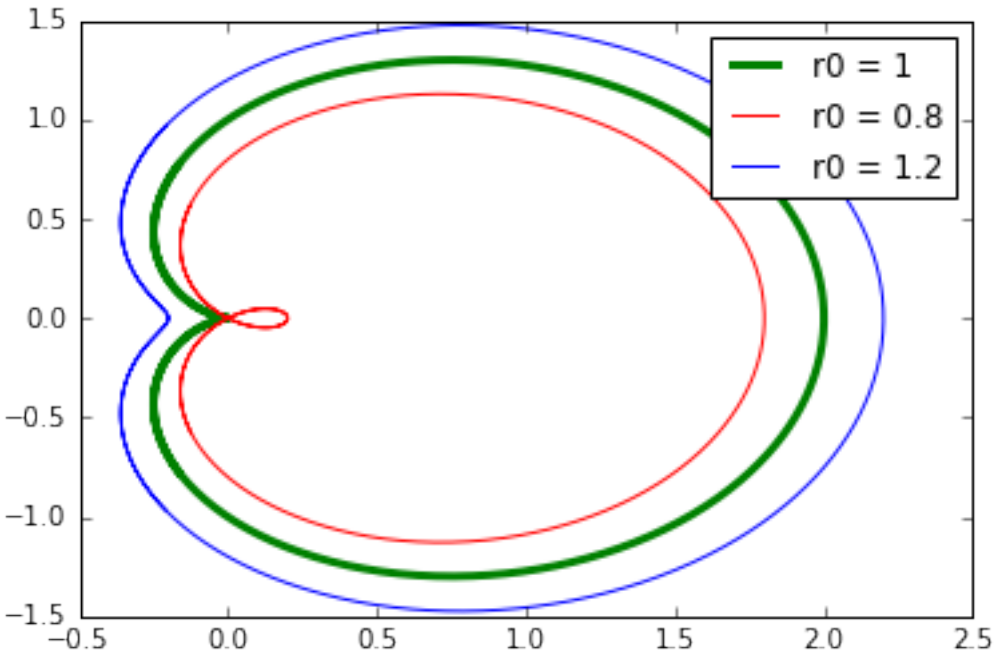
```
In [224]: plt.plot(xs2,ys2,"r",label="r0 = 0.8")
          plt.legend()
          plt.savefig("limaco con ro = 0,8.jpg")
```




```
In [225]: z = np.array([1.2])
          theta = np.linspace(-4.5,4.5,2000)
          limacons = limacon(z,theta)
          xs3 = limacons[1]
          ys3 = limacons[2]

In [226]: plt.plot(xs3,ys3,label="r0 = 1.2")
          plt.legend()
          plt.savefig("limaco con ro = 1,2.jpg")
```



In [229]:
```
plt.plot(xs1,ys1,"g",label="r0 = 1", linewidth=3)
plt.plot(xs2,ys2,"r",label="r0 = 0.8")
plt.plot(xs3,ys3,label="r0 = 1.2")
plt.legend()
plt.savefig("limacos.jpg")
```



In [ ]: