Untitled

January 12, 2017

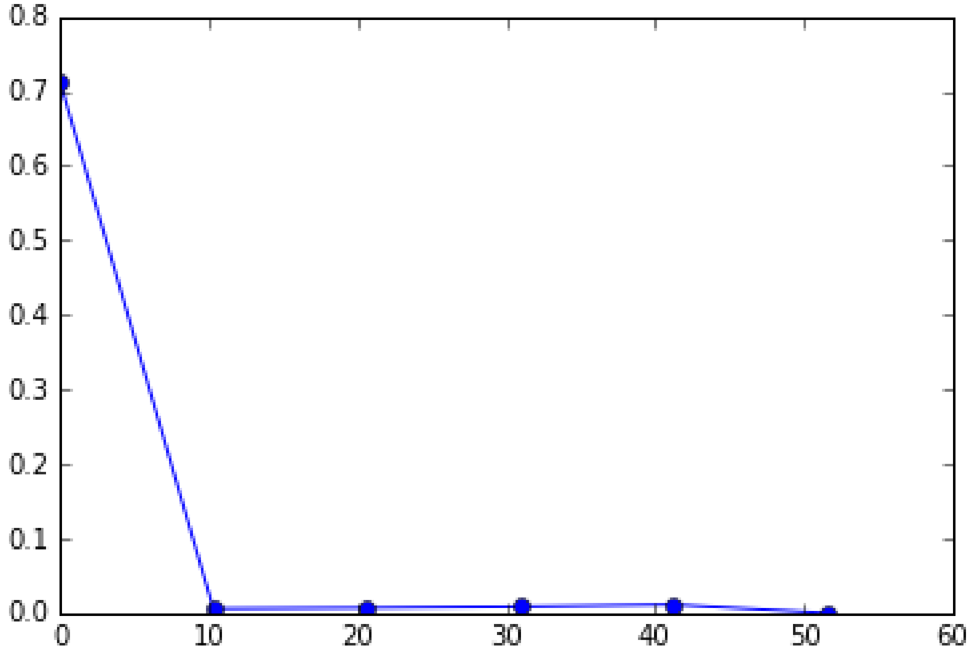
```
In [39]: import numpy as np
         import scipy as sp
         import matplotlib.pyplot as plt
         %matplotlib inline
         %autosave 30
```

```
Autosaving every 30 seconds
```

```
In [40]: def section(s1,s2,N,k,L):
             a=(s2-s1)/L
             return (a*(k-1)*L/N+a*k*L/N)/2.+s1
```

```
In [41]: def matrixA(E,N,L):
             A=np.zeros((N+1,N+1))
             h=L/N
             A[0,0]=E*section(s1,s2,N,1,L)/h
             A[1,0]=-E*section(s1,s2,N,1,L)/h
             A[0,1]=-E*section(s1,s2,N,1,L)/h
             A[1,1]=E*section(s1,s2,N,1,L)/h
             i=1
             while i<N:
                 A[i,i]=A[i,i]+E*section(s1,s2,N,i+1,L)/h
                 A[i+1,i]=-E*section(s1,s2,N,i+1,L)/h
                 A[i,i+1]=-E*section(s1,s2,N,i+1,L)/h
                 A[i+1,i+1]=E*section(s1,s2,N,i+1,L)/h
                 i=i+1
             return A
```

```
In [42]: def vecteurB(rho,omega,N,L):
             B=np.zeros((N+1,1))
             h=L/N
             B[0,0]=rho*section(s1,s2,N,1,L)*omega**2*h*h/6.
             B[1,0]=rho*section(s1,s2,N,1,L)*omega**2*h*h/3.
             i=1
             while i<N:
                 B[i,0]=B[i,0]+rho*section(s1,s2,N,i+1,L)*omega**2*h*(h*i/3.+h*(i+1
                 B[i+1,0]=rho*section(s1,s2,N,i+1,L)*omega**2*h*(h*i/6.+h*(i+1)/3.)
                 i=i+1
             return B

In [43]: def u(E,L,rho,omega,N,s1,s2,r): #Solution analytique pour S variable.
             a=(s2-s1)/L
             b=s1
             y=((omega**2*rho)/(36*E*a**3))*(a*r*(-4*a**2*r**2-3*a*b*r+6*b**2)-6*(b
             return y

In [44]: A=matrixA(E,N,L)
         B=vecteurB(rho,omega,N,L)
         A[0,0]=1
         A[0,1]=0
         B[0,0]=0
         uh=np.linalg.solve(A,B)
         ue=u(E,L,rho,omega,N,s1,s2,r)

In [45]: E=21300*10**6
         s1=16.2
         s2=6.7
         N=5
         L=51.5
         rho=1600
         omega=2*np.pi

In [46]: r=np.linspace(0,L,num=N+1)
         ue=u(E,L,rho,omega,N,s1,s2,r)
         error=np.zeros(N+1)
         for i in range (0,N):
             ue=u(E,L,rho,omega,N,s1,s2,r)
             error[i]=(np.absolute(ue[i]-uh[i])/ue[i])
         plt.plot(r,error,marker='o')

Out[46]: [<matplotlib.lines.Line2D at 0x8d70ef0>]
```



```
In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:
```