HW4_Geodynamics

October 11, 2016

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from scipy.special import erfinv, erf, erfc, erfcinv
```

In [2]:
```
#Turcotte Problem 4.2
#Import data from the table.
data42=np.loadtxt('tab4-1.csv', delimiter=';', dtype='float')
```

In [3]:
```
def heatFlux(k, DT, thick):
    '''Returns the average heat flux q(W/m**2) from a single layer
    with k (W/m*K) thermal conductivity constant, thickess 'thick (m)'
    and with a temperature difference of DT (C, K)'''
    return k*DT/thick
#Extract data from initial matrix
depths=data42[:,0] #m
Ts=data42[:,1]+273 #K
#Calculate temperature differences
DeltaT=np.roll(Ts, -1)-Ts
DeltaT=DeltaT[:-1]
theks=data42[:,2][:-1] #W/m*K
#Calculate thicknesses
thickness=np.roll(depths, -1)-depths
thickness=thickness[:-1]
resultq=heatFlux(theks, DeltaT, thickness)*1e3 #mW m^-2
for i in range(len(resultq)):
    print 'The obtained value for the heat flux in the %d layer \n \
    is %.2f mW/m^2 \n '%(i+1, resultq[i])
print 'Then, the mean value for q across the section \
is %.2f mW/m^2'%np.mean(resultq)
```

```
The obtained value for the heat flux in the 1 layer
     is 74.04 mW/m^2

The obtained value for the heat flux in the 2 layer
     is 78.03 mW/m^2

The obtained value for the heat flux in the 3 layer
     is 111.60 mW/m^2

The obtained value for the heat flux in the 4 layer
     is 81.74 mW/m^2

The obtained value for the heat flux in the 5 layer
```

```
    is 72.95 mW/m^2

The obtained value for the heat flux in the 6 layer
    is 68.02 mW/m^2

Then, the mean value for q across the section is 81.06 mW/m^2
```

$$\frac{T-T_1}{T_0-T_1} = erfc\frac{y}{2\sqrt{\kappa t}}$$
$$T|_{y=0} = T_0$$
$$T|_{t=0} = T_1$$

Thus, $T_0 - T_1 = 10K$ or assume $T_0 = 10K,\ T_1 = 0K$
$y = 1m$; $T = 2K$

$$\eta = \frac{y}{2\sqrt{\kappa t}}$$
$$\sqrt{t} = \frac{y}{2\sqrt{\kappa}\eta}$$
$$t = \left(\frac{y}{2\sqrt{\kappa}\eta}\right)^2$$

In [4]:
```
# Turcotte Problem 4.32
#Given values
kappa=1. #mm**2/s
y=1e3 #mm
Tf=2. #K
T1=0
T0=T1+10
#Value of erfc with given conditions.
erfval=(Tf-T1)/(T0-T1)
#Value of the argument eta of erfc(eta).
eta=erfcinv(erfval)
#Time calculated from eta.
t=(3600.*24)**-1 * (y/(2*np.sqrt(kappa)*eta))**2 #days

print 'The time it takes to increase temperature in 2K at 1m depth \
is %.2f days'%t
```

```
The time it takes to increase temperature in 2K at 1m depth is 3.52 days
```

In [5]:
```
# Additional problem
```

The equation for the solidification of a sill is:

$$\frac{L\sqrt{\pi}}{c(T_m - T_0)} = \frac{\exp\left(-\lambda_2^2\right)}{\lambda_2(1 + erf\lambda_2)}$$

where

$$\lambda_2 = -\frac{y_m}{2\sqrt{\kappa t}} \Rightarrow \sqrt{t} = -\frac{y_m}{2\sqrt{\kappa}\lambda_2}$$

and y_m represents the depth from the surface of the sill to which it has solidified, so for knowing when it has solidified completely we shall say

$$y_m = b$$

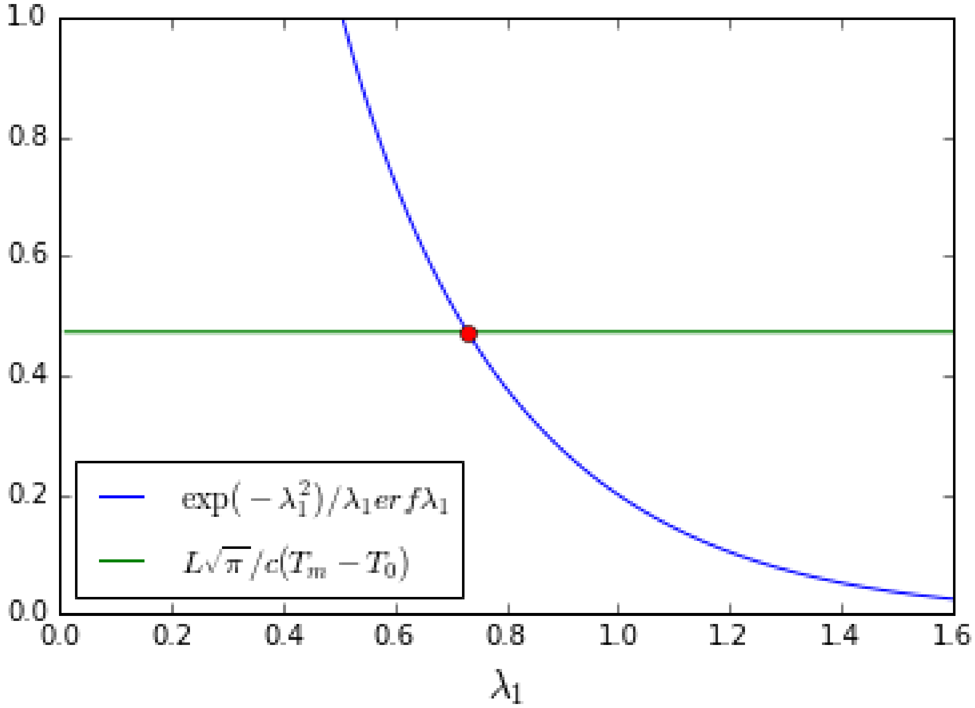
where b is the thickness of the sill.

In [6]:
```
#Array of possible lambda2s
lambda2=np.linspace(0.01, 1.6, 10000)

def theRightSide(lambda2):
    '''Returns the (adimensional) right side of the equation 4-149
    from Turcotte for a given value of lambda2'''
    return np.exp(-lambda2**2)/(lambda2*(1+erf(lambda2)))

#Array for possible values of the right side of the equation.
rightSide=theRightSide(lambda2)
#Given values.
L=320e3 #J/kg
c=1.2e3 #J/kgK
kappa=0.55*(1e-3)**2 #m**2/s
b=5. #m
Tm=1200. #K
T0=200. #K
#Left side of the equation 4-149 from Turcotte.
leftSide=L*np.sqrt(np.pi)/(c*(Tm-T0))
#Array with leftside constant values.
horiz=np.ones(len(lambda2))*leftSide
```

In [7]:
```
#Plot of both sides of the equation 4-149.
plt.plot(lambda2, rightSide, \
         label='$\exp(-\lambda_1^2)/\lambda_1 erf\lambda_1$')
plt.plot(lambda2, horiz, label='$L\sqrt{\pi}/c(T_m-T_0)$')
plt.xlim(0, 1.6)
plt.ylim(0, 1)
#Graphical solution for the equation.
idx = np.argwhere(np.isclose(rightSide, horiz, atol=1e-4)).reshape(-1)
plt.plot(lambda2[idx], rightSide[idx], 'ro')
plt.xlabel('$\lambda_1$', fontsize=15)
plt.legend(loc=0)
plt.show()
print 'The value for lambda that solves the equation is %.4f'%lambda2[idx]
```




```
The value for lambda that solves the equation is 0.7305
```

```python
In [8]: theLambda=lambda2[idx]
        #Time needed for the sill to solidify is calculated.
        t=b/(2*np.sqrt(kappa)*theLambda)
        t=(3600.*24)**-1 *t**2 #days
        print 'The time necessary for the sill \
        to be completely solidified is t=%.3f days'%t
```

```
The time necessary for the sill to be completely solidified is t=246.469 days
```

$$T - T_0 = \frac{erfc(\eta)}{erfc(-\lambda_2)}(T_m - T_0)$$

$$\eta = \frac{y}{2\sqrt{\kappa t}}$$

The sill - country rock boundary is defined to be $y = 0$, so $\eta = 0\ \forall t > 0$ at the interface. Therefore

$$T|_{y=0} = \frac{1}{erfc(-\lambda_2)}(T_m - T_0) + T_0$$

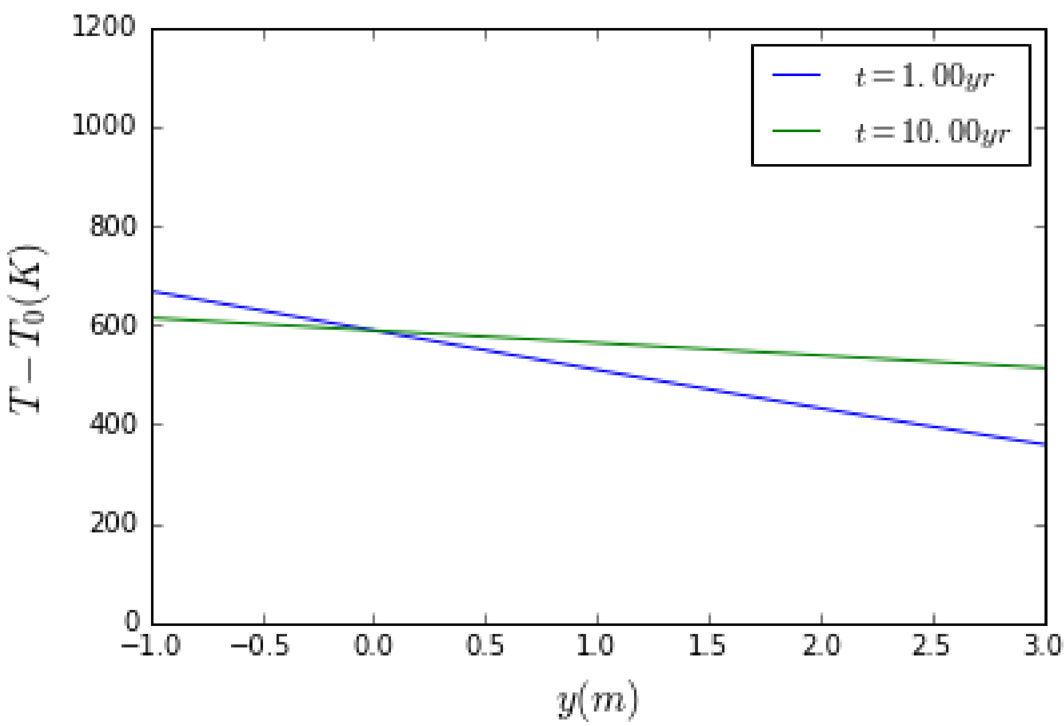
```python
In [9]: interfaceT=(erfc(-theLambda))**-1* (Tm-T0) + T0
        print 'The temperature at the sill - country rock interface \
        is %.3f K'%interfaceT
```

```
The temperature at the sill - country rock interface is 788.777 K
```

```
In [10]: def TmenT0(eta):
             return (Tm-T0)*erfc(eta)/(1.+erf(theLambda))
         def etafromy(y, t):
             return y/(2*np.sqrt(kappa*t))
```

```
In [14]: ys=np.linspace(-1, 3, 100)
         theEta=etafromy(ys, 1.)
         ayear=24*3600.*365.25#s/yr
         possibleTimes=ayear*np.array([1., 10])#s
         for i in possibleTimes:
             plt.plot(ys, TmenT0(etafromy(ys, i)), label='$t=%.2f yr$'%(i/ayear))
         plt.legend(loc=0)
         plt.ylim(0, 1200)
         plt.xlabel('$y(m)$', fontsize=15)
         plt.ylabel('$T-T_0 (K)$', fontsize=15)
```

```
Out[14]: <matplotlib.text.Text at 0x874a1d0>
```




```
In [11]:
```

```
In []:
```