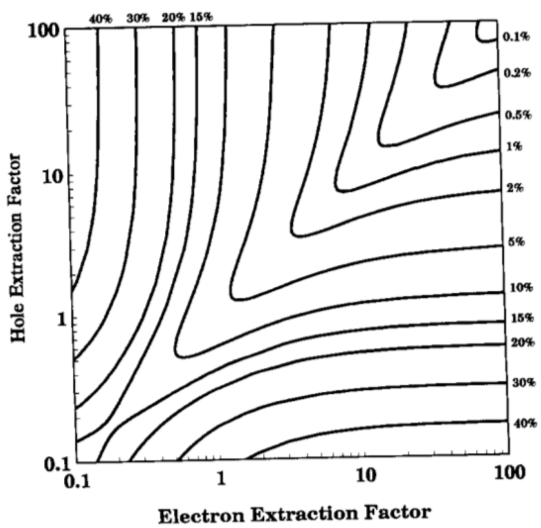
ME 701 – Development of Computer Applications In Mechanical Engineering
Homework 6 – Due 10/25/2017

Instructions: One TAR file `lastname_firstname.tar` that contains two Python files: `lastname_firstname_hw6_p1.py` and `lastname_firstname_hw6_p2.py`, one for each of the problems below.

Problem 1

Consider the following contour plot (from D.S. McGregor et al. NIM A **343** (1994)):

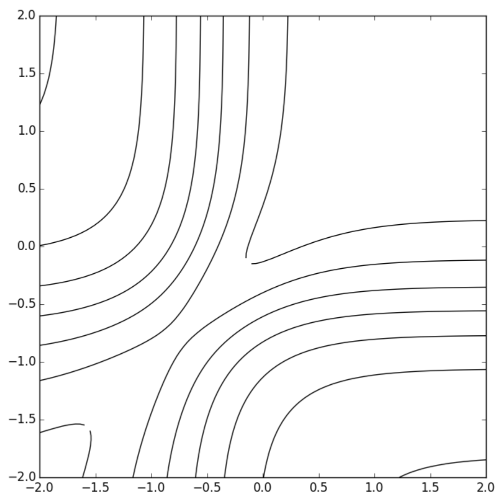


In class, I proposed the following solution (also available in the examples repository):

```python
import numpy as np
import matplotlib.pyplot as plt

plt.ioff()
exp = np.exp

def Q(rho_e, rho_h) :
    return rho_e + rho_e**2*(exp(-1.0/rho_e)-1.0) + \
           rho_h + rho_h**2*(exp(-1.0/rho_h)-1.0)

def sig_Q(rho_e, rho_h) :
    a = rho_e**2 + 2.*rho_e**3*(exp(-1.0/rho_e)-1) + \
        0.5*rho_e**3*(1-exp(-2.0/rho_e))
    b = rho_h**2 + 2.*rho_h**3*(exp(-1.0/rho_h)-1) + \
        0.5*rho_h**3*(1-exp(-2.0/rho_h))
    c = 2.*rho_e*rho_h + 2.*rho_e**2*rho_h*(exp(-1.0/rho_e)-1) + \
```

```
        2.*rho_h**2*rho_e*(exp(-1.0/rho_h)-1)
    d = 2.*(rho_e*rho_h)**2/(rho_e-rho_h)*(exp(-1.0/rho_e)-exp(-1.0/rho_h))
    return np.sqrt( a+b+c+d-Q(rho_e,rho_h)**2)

def R(rho_e, rho_h) :
    return 100*sig_Q(rho_e, rho_h)/Q(rho_e, rho_h)

n = 100
H = np.logspace(-2, 2, n)
E = np.logspace(-2, 2, n)

H, E = np.meshgrid(H, E, sparse=False, indexing='ij')
res = R(E, H)

plt.figure(1, figsize=(8,8))
plt.contour(np.log10(E),np.log10(H), res, colors='k')
plt.savefig('new_contour.png')
```

Execution of the code leads to the following:



This is on the right track, but several features are missing. Your job is to add the following:

1. **appropriate axis labels** (e.g., ‘Electron Extraction Factor’)
2. correct contour levels (i.e., 0.1, 0.2, 0.5, 1%, and so on). *Hint*: look up the documentation for `plt.contour`.
3. **correct x and y tick values** (e.g., -1 should be 0.1 and 2 should be 100) *Hint*: look up, e.g., `plt.xticks`.
4. **annotations for each contour line**. *Hint*: look up `plt.text`, paying specific attention to `fontsize`, `horizontalalignment`, and `verticalalignment`. You might also which to consider using `scipy.optimize.newton` to help you automatically find where text should

be located, e.g., you know that the upper-left 40% box should be located where $F(x) = 100 - R(\rho_e, 100) = 0$. However, you may simply place each text annotation manually. (+1/2 point if you use newton or equivalent)

5. **logarithmic minor tick marks**. Note the original has minor tick marks spaced logarithmically, whereas my solution has no minor tick marks. *Hint*: look up `plt.gca().yaxis`.

Problem 2

We are interested in examining how a time dependent problem changes with a parameter. We shall investigate the time dependent heat transfer equation in 1-D, i.e.,

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2}\,, \tag{1}$$

with the boundary conditions that the $T(x = 0) = 1$ for all time t and that $T(x = 1) = 0$ for all time t. We know that after an infinite amount of time, the solution is linear in x, but how do the solutions vary for a fixed time. Here's a short program for doing that:

```python
from __future__ import division
import numpy as np
import matplotlib.pyplot as plt

alpha = 1

def getTemp(alpha, L=1, tMax=0.1):
    dt = 0.00005
    dx = 0.01
    Nx = int(L / dx)
    dx = L / Nx
    Nt = int(tMax / dt)
    dt = tMax / Nt

    dx = L / Nx
    dt = tMax / Nt

    assert dt * alpha / dx ** 2 <= 0.5, 'Parameters are not numerically stable'

    temp = np.zeros(Nx)
    temp[0] = 1

    for i in range(Nt):
        temp[1:-1] += dt * alpha / dx ** 2 * (temp[0:-2] - 2 * temp[1:-1] + temp[2:])
    return temp, np.linspace(0, L, Nx)

T, x = getTemp(alpha)
plt.plot(x, T)
```

```
plt.show()
```

Your task is to produce an animation showing how the solution changes with increasing alpha. Explore the parameter range $\alpha \in [0, 1]$.