In [10]:
```
from pyne import data
import numpy as np
import scipy.constants as const
import tabulate
```

Define useful constants

I_{Al} from text

$\rho = 2.70 g/cm^3$ from [Wikipedia entry for Alumnium (https://en.wikipedia.org/wiki/Aluminium)](https://en.wikipedia.org/wiki/Aluminium)

In [11]:
```
m_e = const.value('electron mass energy equivalent in MeV')
I_Al = 163 * 10**-6 # eV to MeV
r_0 = const.value('classical electron radius') * 100 # m to cm
z = 1
Z = 13
N_A = const.value('Avogadro constant')
M_m = data.atomic_mass('Al')
rho = 2.70
pi = const.pi
m_p = const.value('proton mass energy equivalent in MeV')
```

Useful conversion functions

In [30]:
```
def beta(gamma):
    g = gamma*gamma
    b = (1 - g ** -1) ** 0.5
    # return ((g - 1.0) / g) ** 0.5
    return b
def gamma(T, m):
    return (T / m) + 1.0
def beta_2_T(T, m):
    denominator = (T + m) ** 2.0
    numerator = T * (T + 2.0 * m)
    return (numerator / denominator)
def beta_T(T, m):
    return beta_2_T(T, m) ** 0.5
def percent_error(truth, model):
    return abs(truth - model) / truth
```

In [31]:
```
assert(beta(gamma(m_p, m_p)) == beta_T(m_p, m_p))
```

Input energies

In [13]:
```
T = [10, 100, 500]
```

In [36]:
```
def S_classical(T):
    # We're dealing with an incident proton now
    m = m_p

    # Get our incident particle energy in terms of Beta
    b = beta_2_T(T, m)

    first_part = 4 * pi * r_0**2 * m_e
    incident_particle_part = z**2 / b
    medium_part = Z * N_A * rho / M_m
    log_term = 2 * m_e * gamma(T, m)**2 * b / I_Al
    last_part = np.log(log_term)

    return first_part * incident_particle_part * medium_part * last_pa
rt
def S_relativistic(T):
    # We're dealing with an incident proton now
    m = m_p

    # Get our incident particle energy in terms of Beta
    b = beta_2_T(T, m)

    first_part = 4 * pi * r_0**2 * m_e
    incident_particle_part = z**2 / b
    medium_part = Z * N_A * rho / M_m
    log_term = 2 * m_e * gamma(T, m)**2 * b / I_Al
    last_part = np.log(log_term) - b

    return first_part * incident_particle_part * medium_part * last_pa
rt
```

Classical Results

In [37]:
```
out = []
headers = ['T', 'Classical', 'Relativistic', 'Beta^2', 'Gamma^2', 'Per
cent Error']
for t in T:
    out.append(
        (t,
         S_classical(t),
         S_relativistic(t),
         beta_2_T(t, m_p),
         gamma(t, m_p) ** 2,
         percent_error(S_relativistic(t), S_classical(t)))
    )
print tabulate.tabulate(out, headers=headers)
```

```
  T    Classical    Relativistic     Beta^2    Gamma^2    Percent Er
ror
---  -----------  --------------  ---------  ---------  ------------
---
 10     93.3096          92.9102  0.0209798    1.02143       0.00429
954
100     15.7951          15.3956  0.183351     1.22452       0.02594
71
500      6.28905          5.88958  0.574426     2.34977       0.06782
67
```

Problem 4

In [16]:
```
t = 2.5
```

From fig 2.11, with $\frac{T/A}{Z} = \frac{2.5}{13} = 0.19$, the shell correction is approximately $\lambda = 0.1$

In [17]:
```
def S_relativistic_effective_charge(T):
    # We're dealing with an incident proton now
    m = m_p

    # Get our incident particle energy in terms of Beta
    b = beta_T(T, m)

    # From Anderson fig 2.11
    shell_correction = 0.1

    first_part = 4 * pi * r_0**2 * m_e
    incident_particle_part = z**2 / b**2
    medium_part = Z * N_A * rho / M_m
    log_term = 2 * m_e * gamma(T, m)**2 * b**2 / I_Al
    last_part = np.log(log_term) - b**2 - shell_correction

    return first_part * incident_particle_part * medium_part * last_pa
rt
```

No corrections:

In [18]:
```
S_classical(t)
```

Out[18]: 264.18888757811038

With corrections:

In [19]:
```
S_relativistic_effective_charge(t)
```

Out[19]: 256.26320216888456

In [20]:
```
M_m
```

Out[20]: 26.981538530999998

In [21]:
```
r_0
```

Out[21]: 2.8179403227e-13

In [22]:
```
m_p
```

Out[22]: 938.2720813

In [23]:
```
m_e
```

Out[23]:
```
0.5109989461
```

In [ ]:
```
beta()
```