In [16]:
```
import scipy.constants as const
import numpy as np
```

In [47]:
```
def alpha(E_gamma=None, nu=None):
    """ Should specify E_gamma in eV
    """
    mecsquared = const.value('electron mass energy equivalent in MeV')
* 10**6
    h = const.value('Planck constant in eV s')
    if E_gamma is None and nu is None:
        raise exception("Must specify photon properties")
    elif nu is None:
        a = E_gamma / mecsquared
    elif E_gamma is None:
        a = h * nu / mecsquared
    else:
        raise exception('Please only specify one photon property')

    return a
```

In [48]:
```
def T_e_max(E_gamma=None, nu=None):
    """ Specify E_gamma in eV
    """
    h = const.value('Planck constant in eV s')
    if E_gamma is None and nu is None:
        raise exception("Must specify photon properties")
    elif nu is None:
        a = alpha(E_gamma=E_gamma)
        t = E_gamma * 2.0 * a / (1.0 + 2.0 * a)
    elif E_gamma is None:
        a = alpha(nu=nu)
        t = h * nu * 2.0 * a / (1.0 + 2.0 * a)
    else:
        raise exception('Please only specify one photon property')

    return t
```

In [49]:
```
def theta_e(T_gamma, theta_gamma):
    a = alpha(E_gamma=T_gamma)
    inner = (1.0 + a) * np.tan(theta_gamma / 2.0)
    return np.arctan(1.0 / inner)
```

In [50]:
```
def T_e(T_gamma, theta):
    a = alpha(E_gamma=T_gamma)
    numerator = T_gamma * a * (1.0 - np.cos(theta))
    denominator = 1.0 + a * (1.0 - np.cos(theta))

    return numerator / denominator
```

Problem 5: Anderson 6.4

In [51]:
```
theta_prime = 60.0 * np.pi / 180.0
```

In [52]:
```
T_g = 140000
```

In [53]:
```
alpha(E_gamma=T_g)
```

Out[53]: 0.27397316778927894

Calculate the angle of scatter of the Compton electron

In [56]:
```
theta_e(T_g, theta_prime) * 180.0 / np.pi
```

Out[56]: 53.664449190568014

In [57]:
```
T_e(T_g, theta_prime)
```

Out[57]: 16867.500476176854

Problem 6

References:

- Mn-54 (http://www.nucleide.org/DDEP_WG/Nuclides/Mn-54_tables.pdf) gamma energy 834.855 keV
- Cs-137 (http://www.nucleide.org/DDEP_WG/Nuclides/Cs-137_tables.pdf) gamma energy 661.659 keV
- Na-22 (http://www.nucleide.org/DDEP_WG/Nuclides/Na-22_tables.pdf) gamma energy 1274.577 keV

Mn-54:

In [58]:
```
T_e_max(E_gamma=834855)
```

Out[58]: 639225.9473887982

In [59]:
```
T_e_max(E_gamma=661659)
```

Out[59]: 477335.86413384555

In [60]:
```
T_e_max(E_gamma=1274577)
```

Out[60]: 1061742.0485465585

Problem 7

In [61]:
```
def sigma_kn(E_gamma):
    r = const.value('classical electron radius') * 100 # convert from
m to cm

    a = alpha(E_gamma=E_gamma)

    answer = np.pi * r * r

    term1 = 2.0 * (1.0 + a) / a ** 2
    term1 *= (2.0 * (1.0 + a) / (1.0 + 2.0 * a) - np.log(1.0 + 2.0 * a
) / a)
    term1 += np.log(1.0 + 2.0 * a) / a
    term1 -= 2.0 * (1.0 + 3.0 * a) / (1.0 + 2.0 * a) ** 2

    return answer * term1
```

Part (a)

In [62]:
```
E = 661659.0 # eV
```

In [63]:
```
sigma_kn(E)
```

Out[63]: 2.5619923244178853e-25

Convert from cm^2 to b

In [64]:
```
sigma_kn(E) * 10**24
```

Out[64]: 0.25619923244178855

Part (b)

From the notes, $\sigma_{atomic} = \sigma_{Compton} = Z * \sigma_{KN}$

In [67]:
```
sigma_kn(E) * 10**24 * 82.0
```

Out[67]: 21.008337060226662

Part (c)

$(\mu/\rho)_{is} = n_m \sigma_{is}, n_m = \frac{N_A Z}{M_m}$

In [68]:
```
rho = 11.34 # g/cm^3
n_m = const.Avogadro * 82.0 / 207.2
```

In [70]:
```
n_m * sigma_kn(E) * 82.0 * rho
```

Out[70]: 56.77795443140117

Problem 8

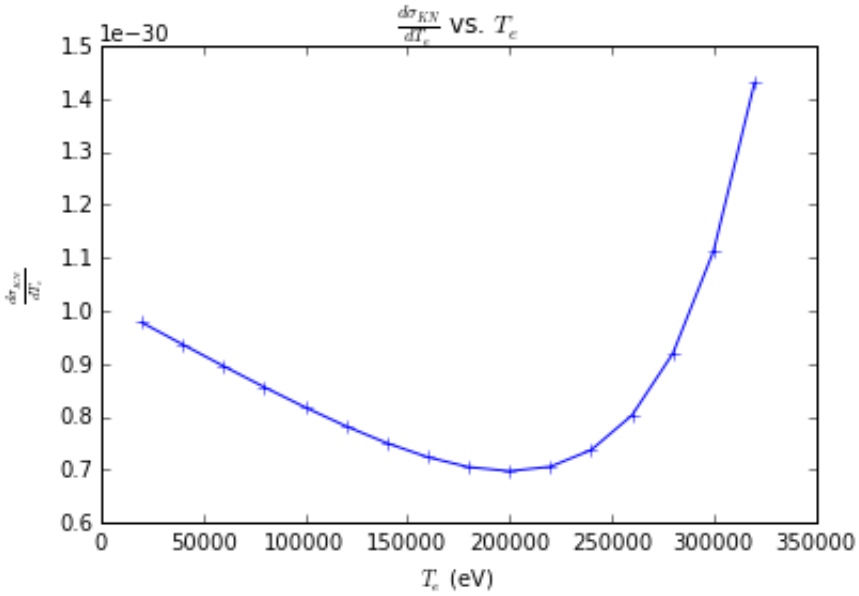
In [72]:
```
def sigma_prime(T_e, T_gamma=500000):
    r = const.value('classical electron radius') * 100 # convert to cm
    a = alpha(E_gamma=T_gamma)

    constant_part = np.pi * r * r / (a * T_gamma)

    other_part = T_e ** 2 / (T_gamma - T_e) ** 2
    other_part *= (1 / a ** 2 + (T_gamma - T_e) / T_gamma - 2 * (T_gam
ma - T_e) / (a * T_e))
    other_part += 2.0

    return other_part * constant_part
```

In [73]:
```
%matplotlib inline
```

In [74]:
```
import matplotlib.pyplot as plt
```

In [75]:
```
T_in = np.arange(20000, T_e_max(E_gamma=500000), 20000)
```

In [81]:
```
plt.plot(T_in, sigma_prime(T_in), 'b-+')
plt.ylabel(r'$\frac{d\sigma_{KN}}{dT_e}$')
plt.xlabel(r'$T_e$ (eV)')
plt.title(r'$\frac{d\sigma_{KN}}{dT_e}$ vs. $T_e$')
plt.show()
```



In [ ]: