pr135_Transitions

March 16, 2020

1 ^{135}Pr - Transition probabilities

1.1 Numerical calculus

1.1.1 Author: Robert Poenaru

Algorithm for calculating the transition probabilities for the isotope.

Experimental data taken from **TO BE DETERMINED**

[1]:
```
import numpy as np
from matplotlib import pyplot as plt
from scipy import special as sp
from matplotlib import rc
```

1.2 Define the prerequisite methods used in the elliptic variables

- the angular momentum components j_1, j_2
- the inertia function A, where $A = f(I, j_2, A_1, A_2)$
- the 2nd component of the odd particle's a.m. j_2 is a function $j_2 = g(j, \theta)$

  $A = f(I, g(j, \theta), A_1, A_2)$

[2]:
```
# define the inertia factors A1, A2, A3
def inertiaFactor(mois):
    a1=1.0/(2.0*mois[0])
    a2=1.0/(2.0*mois[1])
    a3=1.0/(2.0*mois[2])
    factors=(a1,a2,a3)
    return factors
# define the components of the angular momenta $j$ for the odd-particle
def jComponent(oddSpin,theta):
    thetaDegrees=theta*np.pi/180.0
    j1=oddSpin*np.cos(thetaDegrees)
    j2=oddSpin*np.sin(thetaDegrees)
    jcomp=(j1,j2)
    return jcomp
#define the inertial function
def aFct(spin,oddSpin,theta,mois):
    j2=jComponent(oddSpin,theta)[1]
```

```
    a1=inertiaFactor(mois)[0]
    a2=inertiaFactor(mois)[1]
    term=a2*(1.0-j2/spin)-a1
    return term
```

1.3 Defining the elliptic variables

- $u = f(A, A_1, A_3)$
- $v_0 = f(A, A_1, j_1)$

[3]:
```
#define the variabile u
def u(spin,oddSpin,theta,mois):
    a3=inertiaFactor(mois)[2]
    a1=inertiaFactor(mois)[0]
    a=aFct(spin,oddSpin,theta,mois)
    term=(a3-a1)/a
    return term

#define the variabile v0
def vZero(spin,oddSpin,theta,mois):
    a1=inertiaFactor(mois)[0]
    j1-Jcomponent(oddSpin,theta)[0]
    a=aFct(spin,oddSpin,theta,mois)
    term=a1*j1/a
    return (-1.0)*term

#define the variable k
def k(spin,oddSpin,theta, mois):
    return np.sqrt(u(spin,oddSpin,theta,mois))
```

1.4 norm and oscillator length methods

[ ]:
```

```