Chapter_4_1D_Poly_System

May 27, 2019

1 Solving Polynomial Systems Exactly

```
In [1]: import sympy as sp
        import numpy as np
        import time
        import matplotlib.pyplot as plt
```

To begin, we perform do the example given at the start of the Chapter.

```
In [2]: # define the polynomial variables
        x1, x2 = sp.symbols('x1, x2')

        #some useful functions

        #
        Nor = lambda p : sp.poly(sp.LT(p,x2, x1,order = 'grlex',domain = 'QQ')/sp.LM(p,x2, x1,
        Normal = lambda S : S*(1/Nor(S))

        #gets the leading monomial
        LM = lambda p : sp.LM(p,x1,x2,order = 'grlex',domain = 'QQ')
        LCM  = lambda p1, p2 : sp.poly(sp.lcm(LM(p1),LM(p2)), x2, x1, domain = 'QQ')

        #computes the Syzygy terms
        Syz = lambda p1, p2 : sp.poly(LCM(p1,p2)*(1/LM(p1)),x2, x1,domain = 'QQ')*(p1)


        #function for multivariate polynomial division -- Sympy implimentation has an error.
        def divide(p1,p2, order = 'grlex' ):

            mon_1 = p1.monoms(order = order)
            mon_2 = p2.monoms(order = order)

            if np.all((np.array(mon_1[0]) - np.array(mon_2[0]))>=0) == True :

                times = sp.LT(p1, order = order)/sp.LT(p2, order = order)

                new = p1 - p2 * times    #order important
```

```python
            return divide(new, p2 , order = order)

        return p1


sp.init_printing()
```

In [3]:
```python
r1 = sp.poly(x1**2 + 4*x2**2 - 4, x2, x1, domain = 'QQ')
r1 = Normal(r1)
r1
```

Out[3]:

$$\operatorname{Poly}\left(\frac{x_1^2}{4} + x_2^2 - 1, x_2, x_1, domain = \mathbb{Q}\right)$$

A sanity check shows that the notation above is that $x_2 > x_1$

In [4]:
```python
r2 = sp.poly(9*x1**2 + x2**2 - 2*x2 - 8, x2, x1, domain = 'QQ')
r2 = Normal(r2)
r2
```

Out[4]:

$$\operatorname{Poly}\left(9x_1^2 + x_2^2 - 2x_2 - 8, x_2, x_1, domain = \mathbb{Q}\right)$$

In [5]:
```python
R = [r1,r2]
```

In [6]:
```python
Syz(R[0],R[1])
```

Out[6]:

$$\operatorname{Poly}\left(\frac{x_1^2}{4} + x_2^2 - 1, x_2, x_1, domain = \mathbb{Q}\right)$$

In [7]:
```python
Syz(R[1],R[0])
```

Out[7]:

$$\operatorname{Poly}\left(9x_1^2 + x_2^2 - 2x_2 - 8, x_2, x_1, domain = \mathbb{Q}\right)$$

In [8]:
```python
S = sp.expand(Syz(R[0],R[1]) - Syz(R[1],R[0]))
S = Normal(S)
S
```

Out[8]:

$$\operatorname{Poly}\left(x_1^2 - \frac{8x_2}{35} - \frac{4}{5}, x_2, x_1, domain = \mathbb{Q}\right)$$

This polynomial cannot be reduced by the polynomials in our current generating set.

```
In [9]: divide(divide(S,R[0], order = 'grlex' ),R[1])
```

Out[9]:

$$\mathrm{Poly}\left(x_1^2 - \frac{8x_2}{35} - \frac{4}{5}, x_2, x_1, domain = \mathbb{Q}\right)$$

The functions and procedure given above are enough to compute a Gr"obner basis. As we do non wish to dwell on this point, we simply use Sympy's function to gain the (reduced) Grobner basis.

```
In [10]: example_grlex = sp.groebner(R,x2,x1,order = 'grlex',domain = 'QQ')[:]
         example_grlex
```

Out[10]:

$$\left[\mathrm{Poly}\left(x_2^2 + \frac{2x_2}{35} - \frac{4}{5}, x_2, x_1, domain = \mathbb{Q}\right), \quad \mathrm{Poly}\left(x_1^2 - \frac{8x_2}{35} - \frac{4}{5}, x_2, x_1, domain = \mathbb{Q}\right)\right]$$

Or in lexicographic ordering

```
In [11]: example_lex = sp.groebner(R,x2,x1,order = 'lex',domain = 'QQ')[:]
         example_lex
```

Out[11]:

$$\left[\mathrm{Poly}\left(-\frac{35x_1^2}{8} + x_2 + \frac{7}{2}, x_2, x_1, domain = \mathbb{Q}\right), \quad \mathrm{Poly}\left(x_1^4 - \frac{1944x_1^2}{1225} + \frac{144}{245}, x_2, x_1, domain = \mathbb{Q}\right)\right]$$

```
In [12]: print('For Graded Lex ordering:')
         for i in range(len(example_grlex)):
             print(sp.LM(example_grlex[i],x2,x1,order = 'grlex'))

         print('For Lex ordering:')
         for i in range(len(example_grlex)):
             print(sp.LM(example_lex[i],x2,x1,order = 'lex'))
```

```
For Graded Lex ordering:
x2**2
x1**2
For Lex ordering:
x2
x1**4
```

Have a function that finds one dimensional affine moments in exact arithmetic for an arbitrary one-dimensional affine IFS.

```
In [13]: def FindMoments(a = [sp.Rational(1,3),sp.Rational(1,3)],b = [0,sp.Rational(2,3)],p =

             #form a local dictionary of the variables input

             d = {'a': a, 'b': b , 'p' : p, 'n' :n}

             assert(len(d['a'])==len(d['b'])==len(d['p']))

             #get the number of IFS maps

             N = len(d['a'])

             for i in range(1,N+1):

                 d.update({'i':i})

                 exec('a'+str(i) + "= a[i-1]",d)
                 exec('b'+str(i) + "= b[i-1]",d)
                 exec('p'+str(i) + "= p[i-1]",d)

                 d.update({'zero':sp.zeros(d['n'])})

                 exec('A'+str(i) + "= zero",d)

             for k in range(1,N+1):

                 d.update({'k':k})

                 for i in range(d['n']):

                     d.update({'i':i})

                     for j in range(i+1):

                         d.update({'j':j,'Bi':int( sp.binomial(i,j) )})

                         exec('A'+str(k)+'[i,j] = a'+str(k)+'**j*b'+str(k)+'**(i-j)*Bi' ,d)

             E = sp.zeros(d['n'])

             d.update({'E':E})

             for i in range(N):

                 exec('E += p'+str(i+1)+'*A'+str(i+1) ,d)

             E = d['E'] - sp.eye(d['n'])
```

```
        #find nullspace and normalise

        M = E.nullspace()

        M = M[0]/(M[0][0])

        return M
```

```
In [14]: def poly_system(M, N=2, p =[sp.Rational(1,2),sp.Rational(1,2)], rational = True):

        poly = -M

        n = len(M)

        for i in range(1,N+1):

            exec('a'+str(i) + "= sp.Symbol('a_'+str("+str(i) +") , rational=True)",globals
            exec('b'+str(i) + "= sp.Symbol('b_'+str("+str(i) +") , rational=True)",globals
            exec('A'+str(i) + "= sp.zeros(n,n)")

            if len(p)==N:

                exec('p'+str(i) + "= p["+str(i-1)+"]")

            else:

                exec('p'+str(i) + "= sp.Symbol('p_'+str("+str(i) +") , rational=True)",glo


        for k in range(1,N+1):

            for i in range(n):
                for j in range(i+1):

                    exec('A'+str(k)+'[i,j] = a'+str(k)+'**j*b'+str(k)+'**(i-j)*int(sp.bino

            poly += eval("p"+str(k)+"*A"+str(k)+"*M")

        if rational:
            poly = sp.nsimplify(poly)

        return poly
```

Print the system for a Cantor set

```
In [15]: Cantor = poly_system(FindMoments(),p = [])
         Cantor
```

```
Out[15]:
```

$$\left[\begin{array}{c} p_1 + p_2 - 1 \\ \frac{a_1 p_1}{2} + \frac{a_2 p_2}{2} + b_1 p_1 + b_2 p_2 - \frac{1}{2} \\ \frac{3p_1}{8} a_1^2 + a_1 b_1 p_1 + \frac{3p_2}{8} a_2^2 + a_2 b_2 p_2 + b_1^2 p_1 + b_2^2 p_2 - \frac{3}{8} \\ \frac{5p_1}{16} a_1^3 + \frac{9b_1}{8} a_1^2 p_1 + \frac{3a_1}{2} b_1^2 p_1 + \frac{5p_2}{16} a_2^3 + \frac{9b_2}{8} a_2^2 p_2 + \frac{3a_2}{2} b_2^2 p_2 + b_1^3 p_1 + b_2^3 p_2 - \frac{5}{16} \\ \frac{87p_1}{320} a_1^4 + \frac{5b_1}{4} a_1^3 p_1 + \frac{9p_1}{4} a_1^2 b_1^2 + 2a_1 b_1^3 p_1 + \frac{87p_2}{320} a_2^4 + \frac{5b_2}{4} a_2^3 p_2 + \frac{9p_2}{4} a_2^2 b_2^2 + 2a_2 b_2^3 p_2 + b_1^4 p_1 + b_2^4 p_2 - \frac{87}{320} \\ \frac{31p_1}{128} a_1^5 + \frac{87b_1}{64} a_1^4 p_1 + \frac{25p_1}{8} a_1^3 b_1^2 + \frac{15p_1}{4} a_1^2 b_1^3 + \frac{5a_1}{2} b_1^4 p_1 + \frac{31p_2}{128} a_2^5 + \frac{87b_2}{64} a_2^4 p_2 + \frac{25p_2}{8} a_2^3 b_2^2 + \frac{15p_2}{4} a_2^2 b_2^3 + \frac{5a_2}{2} b_2^4 p_2 + b_1^5 p_1 + b_2^5 \end{array}\right.$$

```
In [16]: t = time.time()
         Cantor_grlex = sp.groebner(Cantor,b1,b2,a1,a2,p1,p2,order = 'grlex',domain = 'QQ')
         t =  time.time() - t
         print('This computation took ' + str(t) + ' seconds')
         Cantor_grlex[:]
```

This computation took 1.254598617553711 seconds

Out[16]:

$$\left[a_1^4 a_2^2 - a_1^4 - \frac{64a_1^2}{27} a_2^2 + \frac{16a_2}{27} a_1^2 b_2 - \frac{8a_2}{27} a_1^2 + \frac{16a_1^2}{27} b_2^2 - \frac{16b_2}{27} a_1^2 + \frac{8a_1^2}{3} + \frac{8a_1}{27} a_2^3 + \frac{16a_1}{27} a_2^2 b_2 - \frac{8a_1}{27} a_2^2 - \frac{8a_1}{3} a_2 - \frac{16a_1}{3} b_2\right.$$

```
In [17]: grlex_lm = []
         for i in range(len(Cantor_grlex)):
             grlex_lm.append(sp.LM(Cantor_grlex[i],b1,b2,a1,a2,p1,p2,order = 'grlex'))
         grlex_lm
```

Out[17]:

$$\left[a_1^4 a_2^2, \quad a_1^2 a_2^4, \quad a_1^2 a_2^2 b_1, \quad a_2^4 b_1, \quad a_1^2 b_2^3, \quad a_2^2 b_2^2 p_2, \quad a_1^4 b_2, \quad a_1^2 a_2^2 b_2, \quad a_2^2 b_2 p_2^2, \quad a_2^4 p_2, \quad a_2^2 b_1^2, \quad a_1^2 b_1 b_2, \quad b_2^3 p_2, \quad b_1^2 b_2,\right.$$

Above is the corner set for our reduced Gr''obner basis. Note that our closed set enclosed by this border is unbounded. For example, the monommial terms $\{p_2^i\}_{i=1}^{\infty}$ cannot be reduced with the monomials above.

```
In [18]: t = time.time()
         Cantor_lex = sp.groebner(Cantor,b1,b2,a1,a2,p1,p2,order = 'lex',domain = 'QQ')
         t =  np.round(time.time() - t,4)
         print('This computation took ' + str(t) + ' seconds')
         Cantor_lex[:]
```

This computation took 30.427 seconds

Out[18]:

$$\left[\frac{a_1^2 p_2}{8} - \frac{a_1^2}{8} + \frac{a_1 a_2}{4} + \frac{a_1 b_2}{2} - \frac{a_1}{4} - \frac{a_2^2 p_2}{8} + \frac{a_2 b_1}{2} - \frac{a_2}{4} + b_1 b_2 - \frac{b_1}{2} - \frac{b_2}{2} + \frac{3}{8}, \quad \frac{a_1 a_2^2}{2} - \frac{a_1}{2} - a_2^3 p_2^2 + \frac{3p_2}{2} a_2^3 + a_2^2 b_1 - 2\right.$$

```
In [19]: lex_lm = []
         for i in range(len(Cantor_lex)):
             lex_lm.append(sp.LM(Cantor_lex[i],b1,b2,a1,a2,p1,p2,order = 'lex'))
         lex_lm
```

Out[19]:

$$\left[b_1 b_2, \quad a_2^2 b_1, \quad b_1 p_2, \quad a_1^2 b_2^2, \quad b_2^2 p_2, \quad a_1^4 b_2, \quad a_1^2 b_2 p_2, \quad a_2^4 b_2 p_2, \quad a_2^2 b_2 p_2^3, \quad a_1^4 p_2, \quad a_1^2 a_2^2, \quad a_1^2 p_2^3, \quad a_2^8 p_2, \quad a_2^6 p_2^3, \quad p\right.$$

Again, the monommial terms $\{p_2^i\}_{i=1}^{\infty}$ cannot be reduced with the monomials above.

Notice above that p_2 is the smallest term in the lexicographic ordering, and creates the source of the unbounded polynomials. Thus, in the choice of the ordering, we may choose which variable becomes unbounded. Making the assumption of the Open Set Condition, we get the balance property $p_i = a_i^D$ where D is the Hausdorff dimension of the set. We will exploit this now.

```
In [20]: t = time.time()
         Cantor_lex_a_d = sp.groebner(Cantor,b1,b2,a1,p1,a2,p2,order = 'lex',domain = 'QQ')
         t =  np.round(time.time() - t,4)
         print('This computation took ' + str(t) + ' seconds')
         Cantor_lex_a_d[-1]
```

```
This computation took 38.5978 seconds
```

Out[20]:

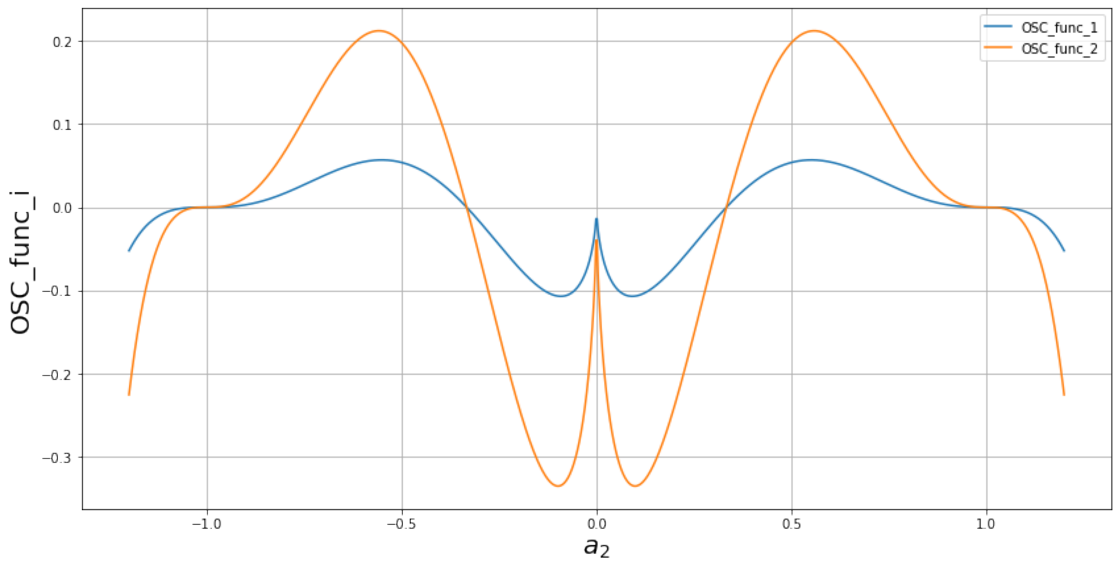
$$a_2^6 p_2^3 - \frac{5a_2^6}{2} p_2^2 + a_2^6 p_2 - 3a_2^4 p_2^3 + \frac{15a_2^4}{2} p_2^2 - 3a_2^4 p_2 + 3a_2^2 p_2^3 - \frac{15a_2^2}{2} p_2^2 + 3a_2^2 p_2 - p_2^3 + \frac{5p_2^2}{2} - p_2$$

Through our ordering chosen, we have isolated the variables a_2 and p_2 as one of the generators for our Gr"obner basis. Exploiting the OSC, we may reduce the polynomial above to one that is a univariate polynomial.

```
In [21]: D = np.log(2)/np.log(3)

         OSC_func_1 = sp.lambdify(a2, Cantor_lex_a_d[-1].subs(p2,sp.Abs(a2)**(D)))
         OSC_func_2 = sp.lambdify(a2, Cantor_lex_a_d[-2].subs(p2,sp.Abs(a2)**(D)))

         plt.figure(figsize=(14,7))
         z = np.linspace(-1.2,1.2,1000)
         plt.xlabel('$a_2$',size = 20)
         plt.ylabel('OSC_func_i',size = 20)
         plt.plot(z , OSC_func_1(z), label = 'OSC_func_1')
         plt.plot(z , OSC_func_2(z), label = 'OSC_func_2')
         plt.legend()
         plt.grid()
```



These are 'easy' functions to find the roots of. One could use Newton's method to achieve these roots to machine accuracy (with being careful around the point $(0,0)$ where the functions given are non-differentiable). Given the simplicity, we just check them below.

```
In [23]: [OSC_func_1(0),OSC_func_1(1),OSC_func_1(-1),OSC_func_1(1/3),OSC_func_1(-1/3)]
```

Out[23]:

$$[0.0, \quad 0.0, \quad 0.0, \quad 1.9567680809018384e-15, \quad 1.9567680809018384e-15]$$

As one can see, there are obvious trivial solutions at the values $a_2 = \pm 1$. The third last equation in our Grobner basis reveals that the root at zero is also trivial as $a_2 = 0 \Rightarrow a_1 = 1$.

With this observation, we aim to solve our polynomial system exactly without the assumption of the OSC.

```
In [24]: def poly_system(M, N=2, p =[sp.Rational(1,2),sp.Rational(1,2)], rational = True):

             poly = -M

             n = len(M)

             for i in range(1,N+1):

                 exec('a'+str(i) + "= sp.Symbol('a_'+str("+str(i) +") , rational=True)",globals
                 exec('b'+str(i) + "= sp.Symbol('b_'+str("+str(i) +") , rational=True)",globals
                 exec('A'+str(i) + "= sp.zeros(n,n)")

                 if len(p)==N:

                     exec('p'+str(i) + "= p["+str(i-1)+"]")
```

```
            else:

                exec('p'+str(i) + "= sp.Symbol('p_'+str("+str(i) +") , rational=True)",gl

        for k in range(1,N+1):

            for i in range(n):
                for j in range(i+1):

                    exec('A'+str(k)+'[i,j] = a'+str(k)+'**j*b'+str(k)+'**(i-j)*int(sp.bin

            poly += eval("p"+str(k)+"*A"+str(k)+"*M")

        if rational:
            poly = sp.nsimplify(poly)

        return poly
```

In [25]:
```
def poly_constraint(N = 2, a = [a1,a2]):

        assert len(a) == N


        P = []

        for i in range(1,N+1):

            for j in range(1,3):

                    exec('t'+str(i)+str(j) + "= sp.Symbol('t_'+str("+str(i)+str(j) +") , 

                    if j==1:

                        P.append(eval('(a[i-1] - 1)'+'*t'+str(i)+str(j) + '-1' ) )

                    if j==2:

                        P.append(eval('(a[i-1] + 1)'+'*t'+str(i)+str(j) + '-1' ) )

        return sp.Matrix(P)
```

In [26]:
```
con = poly_constraint()
con
```

Out[26]:

$$\begin{bmatrix} t_{11}(a_1 - 1) - 1 \\ t_{12}(a_1 + 1) - 1 \\ t_{21}(a_2 - 1) - 1 \\ t_{22}(a_2 + 1) - 1 \end{bmatrix}$$

```
In [27]: Cantor
```

Out[27]:

$$\left[\begin{matrix} p_1 + p_2 - 1 \\ \frac{a_1 p_1}{2} + \frac{a_2 p_2}{2} + b_1 p_1 + b_2 p_2 - \frac{1}{2} \\ \frac{3p_1}{8} a_1^2 + a_1 b_1 p_1 + \frac{3p_2}{8} a_2^2 + a_2 b_2 p_2 + b_1^2 p_1 + b_2^2 p_2 - \frac{3}{8} \\ \frac{5p_1}{16} a_1^3 + \frac{9b_1}{8} a_1^2 p_1 + \frac{3a_1}{2} b_1^2 p_1 + \frac{5p_2}{16} a_2^3 + \frac{9b_2}{8} a_2^2 p_2 + \frac{3a_2}{2} b_2^2 p_2 + b_1^3 p_1 + b_2^3 p_2 - \frac{5}{16} \\ \frac{87p_1}{320} a_1^4 + \frac{5b_1}{4} a_1^3 p_1 + \frac{9p_1}{4} a_1^2 b_1^2 + 2a_1 b_1^3 p_1 + \frac{87p_2}{320} a_2^4 + \frac{5b_2}{4} a_2^3 p_2 + \frac{9p_2}{4} a_2^2 b_2^2 + 2a_2 b_2^3 p_2 + b_1^4 p_1 + b_2^4 p_2 - \frac{87}{320} \\ \frac{31p_1}{128} a_1^5 + \frac{87b_1}{64} a_1^4 p_1 + \frac{25p_1}{8} a_1^3 b_1^2 + \frac{15p_1}{4} a_1^2 b_1^3 + \frac{5a_1}{2} b_1^4 p_1 + \frac{31p_2}{128} a_2^5 + \frac{87b_2}{64} a_2^4 p_2 + \frac{25p_2}{8} a_2^3 b_2^2 + \frac{15p_2}{4} a_2^2 b_2^3 + \frac{5a_2}{2} b_2^4 p_2 + b_1^5 p_1 + b_2^5 \end{matrix}\right.$$

```
In [28]: Cantor_con = sp.Matrix.vstack(Cantor,con)
         Cantor_con
```

Out[28]:

$$\left[\begin{matrix} p_1 + p_2 - 1 \\ \frac{a_1 p_1}{2} + \frac{a_2 p_2}{2} + b_1 p_1 + b_2 p_2 - \frac{1}{2} \\ \frac{3p_1}{8} a_1^2 + a_1 b_1 p_1 + \frac{3p_2}{8} a_2^2 + a_2 b_2 p_2 + b_1^2 p_1 + b_2^2 p_2 - \frac{3}{8} \\ \frac{5p_1}{16} a_1^3 + \frac{9b_1}{8} a_1^2 p_1 + \frac{3a_1}{2} b_1^2 p_1 + \frac{5p_2}{16} a_2^3 + \frac{9b_2}{8} a_2^2 p_2 + \frac{3a_2}{2} b_2^2 p_2 + b_1^3 p_1 + b_2^3 p_2 - \frac{5}{16} \\ \frac{87p_1}{320} a_1^4 + \frac{5b_1}{4} a_1^3 p_1 + \frac{9p_1}{4} a_1^2 b_1^2 + 2a_1 b_1^3 p_1 + \frac{87p_2}{320} a_2^4 + \frac{5b_2}{4} a_2^3 p_2 + \frac{9p_2}{4} a_2^2 b_2^2 + 2a_2 b_2^3 p_2 + b_1^4 p_1 + b_2^4 p_2 - \frac{87}{320} \\ \frac{31p_1}{128} a_1^5 + \frac{87b_1}{64} a_1^4 p_1 + \frac{25p_1}{8} a_1^3 b_1^2 + \frac{15p_1}{4} a_1^2 b_1^3 + \frac{5a_1}{2} b_1^4 p_1 + \frac{31p_2}{128} a_2^5 + \frac{87b_2}{64} a_2^4 p_2 + \frac{25p_2}{8} a_2^3 b_2^2 + \frac{15p_2}{4} a_2^2 b_2^3 + \frac{5a_2}{2} b_2^4 p_2 + b_1^5 p_1 + b_2^5 \\ t_{11}(a_1 - 1) - 1 \\ t_{12}(a_1 + 1) - 1 \\ t_{21}(a_2 - 1) - 1 \\ t_{22}(a_2 + 1) - 1 \end{matrix}\right.$$

```
In [29]: t = time.time()
         Cantor_con_grlex = sp.groebner(Cantor_con,b1,b2,t11,t12,a1,t21,t22,a2,p1,p2,order = '
         t =  np.round(time.time() - t,4)
         print('This computation took ' + str(t) + ' seconds')
         Cantor_con_grlex[:]
```

```
This computation took 14.6668 seconds
```

Out[29]:

$$\left[a_2 b_2 - \frac{a_2}{2} + b_2^2 - b_2 + \frac{1}{6}, \quad a_1^2 - \frac{1}{9}, \quad a_2^2 - \frac{1}{9}, \quad \frac{a_1}{2} + \frac{a_2}{2} + b_1 + b_2 - 1, \quad \frac{9a_1}{8} + t_{11} + \frac{9}{8}, \quad \frac{9a_1}{8} + t_{12} - \frac{9}{8}, \quad \frac{9a_2}{8} + t\right.$$

```
In [30]: con_grlex_lm = []
         for i in range(len(Cantor_con_grlex)):
             con_grlex_lm.append(sp.LM(Cantor_con_grlex[i],b1,b2,t11,t12,a1,t21,t22,a2,p1,p2,o
         con_grlex_lm
```

Out[30]:

$$\left[b_2^2, \quad a_1^2, \quad a_2^2, \quad b_1, \quad t_{11}, \quad t_{12}, \quad t_{21}, \quad t_{22}, \quad p_1, \quad p_2\right]$$

Now the border basis keeps finitely many terms. This creates a system we may readily solve.

```
In [31]: sp.Matrix(sp.solve(Cantor_con_grlex)[:])
```

Out[31]:

$$\begin{bmatrix}
\left\{a_1 : -\frac{1}{3}, \quad a_2 : -\frac{1}{3}, \quad b_1 : \frac{1}{3}, \quad b_2 : 1, \quad p_1 : \frac{1}{2}, \quad p_2 : \frac{1}{2}, \quad t_{11} : -\frac{3}{4}, \quad t_{12} : \frac{3}{2}, \quad t_{21} : -\frac{3}{4}, \quad t_{22} : \frac{3}{2}\right\} \\
\left\{a_1 : -\frac{1}{3}, \quad a_2 : -\frac{1}{3}, \quad b_1 : 1, \quad b_2 : \frac{1}{3}, \quad p_1 : \frac{1}{2}, \quad p_2 : \frac{1}{2}, \quad t_{11} : -\frac{3}{4}, \quad t_{12} : \frac{3}{2}, \quad t_{21} : -\frac{3}{4}, \quad t_{22} : \frac{3}{2}\right\} \\
\left\{a_1 : -\frac{1}{3}, \quad a_2 : \frac{1}{3}, \quad b_1 : \frac{1}{3}, \quad b_2 : \frac{2}{3}, \quad p_1 : \frac{1}{2}, \quad p_2 : \frac{1}{2}, \quad t_{11} : -\frac{3}{4}, \quad t_{12} : \frac{3}{2}, \quad t_{21} : -\frac{3}{2}, \quad t_{22} : \frac{3}{4}\right\} \\
\left\{a_1 : -\frac{1}{3}, \quad a_2 : \frac{1}{3}, \quad b_1 : 1, \quad b_2 : 0, \quad p_1 : \frac{1}{2}, \quad p_2 : \frac{1}{2}, \quad t_{11} : -\frac{3}{4}, \quad t_{12} : \frac{3}{2}, \quad t_{21} : -\frac{3}{2}, \quad t_{22} : \frac{3}{4}\right\} \\
\left\{a_1 : \frac{1}{3}, \quad a_2 : -\frac{1}{3}, \quad b_1 : 0, \quad b_2 : 1, \quad p_1 : \frac{1}{2}, \quad p_2 : \frac{1}{2}, \quad t_{11} : -\frac{3}{2}, \quad t_{12} : \frac{3}{4}, \quad t_{21} : -\frac{3}{4}, \quad t_{22} : \frac{3}{2}\right\} \\
\left\{a_1 : \frac{1}{3}, \quad a_2 : -\frac{1}{3}, \quad b_1 : \frac{2}{3}, \quad b_2 : \frac{1}{3}, \quad p_1 : \frac{1}{2}, \quad p_2 : \frac{1}{2}, \quad t_{11} : -\frac{3}{2}, \quad t_{12} : \frac{3}{4}, \quad t_{21} : -\frac{3}{4}, \quad t_{22} : \frac{3}{2}\right\} \\
\left\{a_1 : \frac{1}{3}, \quad a_2 : \frac{1}{3}, \quad b_1 : 0, \quad b_2 : \frac{2}{3}, \quad p_1 : \frac{1}{2}, \quad p_2 : \frac{1}{2}, \quad t_{11} : -\frac{3}{2}, \quad t_{12} : \frac{3}{4}, \quad t_{21} : -\frac{3}{2}, \quad t_{22} : \frac{3}{4}\right\} \\
\left\{a_1 : \frac{1}{3}, \quad a_2 : \frac{1}{3}, \quad b_1 : \frac{2}{3}, \quad b_2 : 0, \quad p_1 : \frac{1}{2}, \quad p_2 : \frac{1}{2}, \quad t_{11} : -\frac{3}{2}, \quad t_{12} : \frac{3}{4}, \quad t_{21} : -\frac{3}{2}, \quad t_{22} : \frac{3}{4}\right\}
\end{bmatrix}$$

2 Solving Polynomial Systems Numerically

```
In [32]: import scipy
         from scipy.optimize import least_squares
```

Define the matrix that encodes the affine transformation and the corresponding operator.

```
In [33]: def A(a,b,n=5):

             A = np.zeros((n,n))

             for i in range(n):

                 for j in range(i+1):

                         A[i,j] = (a**j)*(b**(i-j))*scipy.special.binom(i,j)

             return A
```

```
In [34]: def Phi(x,n=5):

             assert len(x)%3 == 0

             N = int(len(x)/3)
```

```
    Phi = np.zeros((n,n))

    for i in range(N):

        p = x[i]

        a = x[N+i]

        b = x[2*N+i]

        Phi += p*A(a,b,n=n)

    return Phi
```

Make a (numeric) function to find the moment of a one dimensional affine IFS.

```
In [35]: def find_moments(x,n=5,power = 10):

    assert len(x)%3 == 0

    N = int(len(x)/3)

    Phi = np.zeros((n,n))

    for i in range(N):

        p = x[i]

        a = x[N+i]

        b = x[2*N+i]

        Phi += p*A(a,b,n=n)

    return np.linalg.matrix_power(Phi,power)@np.ones(n)
```

Make a function that take the moment data as input and evaluates the (constrained) polynomial system.

```
In [36]: def eval_P(x,M, constrain = True):

    n = len(M)

    assert len(x)%5 == 0

    N = int(len(x)/5)

    t1 = x[0:N]
```

```
        t11 = x[N:2*N]
        a = x[3*N:4*N]
        F1 = (1-a)*t1**2 - 1
        F2 = (a+1)*t11**2 - 1
        F3 = Phi(x[2*N:5*N],n=n)@M - M
        if not constrain:
            return F3
        P = np.hstack((F1,F2,F3))
        return P
```

Compute the n^{th} line of the Pascal matrix for use later.

```
In [37]: def pascal(n):
             line = [1]
             for k in range(n):
                 line.append(line[k] * (n-k) / (k+1))
             return line
```

Compute vector of powers of a and b in an efficient manner.

```
In [38]: def pow_vec_a(a,n=5):
             av = np.zeros(n)
             av[0] = 1
             for i in range(1,n):
                 av[i] = av[i-1]*a
             return av
```

```
In [39]: def pow_vec_b(b,n=5):
             bv = np.zeros(n)
```

```
        bv[-1] = 1

        for i in range(1,n):

            bv[n-i-1] = bv[n-i]*b

        return bv
```

Use the functions above to compute the partial derivatives.

```
In [40]: def p_jac(a,b,M):

             n = len(M)

             av = pow_vec_a(a,n)

             bv = pow_vec_b(b,n)

             pas = pascal(n-1)

             return np.dot(av*M,bv*pas)
```

```
In [41]: def a_jac(a,b,p,M):

             n = len(M)

             if n == 1:
                 return 0

             M_new = M[1:]

             av = pow_vec_a(a,n-1)

             bv = pow_vec_b(b,n-1)

             pas = pascal(n-1)[1:]

             return p*np.dot(range(1,n)*av*pas,bv*M_new)
```

```
In [42]: def b_jac(a,b,p,M):

             n = len(M)

             if n == 1:
                 return 0

             M_new = M[:-1]

             av = pow_vec_a(a,n-1)
```

```
    bv = pow_vec_b(b,n-1)

    pas = pascal(n-1)[:-1]

    n_j = n-np.array(range(1,n))

    return p*np.dot(n_j*av*pas,bv*M_new)
```

Put the functions above to compute the Jacobian of the polynomial system.

```
In [43]: def Jac_P(x,M):

             assert len(x)%5 == 0

             N = int(len(x)/5)

             nvar = len(x)

             neq = len(M) + 2*N

             t1 = x[0:N]

             t11 = x[N:2*N]

             p = x[2*N:3*N]

             a = x[3*N:4*N]

             b = x[4*N:5*N]

             jac = np.zeros((neq,nvar))

             for i in range(N):

                 #first dummy varibles
                 jac[i,i] = 2*t1[i]*(1-a[i])

                 jac[i,3*N+i] = -t1[i]**2

                 #second dummy varibles
                 jac[N+i,N+i] = 2*t11[i]*(a[i]+1)

                 jac[N+i,3*N+i] = t11[i]**2


             for j in range(len(M)):
```

```
            for i in range(N):

                #probabilites

                jac[2*N+j,2*N+i] = p_jac(a[i],b[i],M[:j+1])

                #a values

                jac[2*N+j,3*N+i] = a_jac(a[i],b[i],p[i],M[:j+1])

                #b values

                jac[2*N+j,4*N+i] =  b_jac(a[i],b[i],p[i],M[:j+1])


        return jac
```

Numerical experiment begins here

```
In [98]: x = np.array([1,1,1,1,1/2,1/2,1/3,1/3,0,2/3])

In [45]: M = find_moments([1/2,1/2,1/3,1/3,0,2/3],n=10, power = 50)
         M

Out[45]: array([1.        , 0.5       , 0.375     , 0.3125    , 0.271875  ,
                0.2421875 , 0.21924365, 0.20094651, 0.18603433, 0.17366291])

In [46]: eval_P(x,M)

Out[46]: array([-1.00000000e+00, -1.00000000e+00, -1.00000000e+00, -1.00000000e+00,
                 0.00000000e+00,  0.00000000e+00,  0.00000000e+00, -5.55111512e-17,
                 0.00000000e+00,  0.00000000e+00,  0.00000000e+00, -2.77555756e-17,
                 0.00000000e+00,  2.77555756e-17])

In [74]: x0 = x + 0.1*np.random.randn(len(x))
         print(x0)

[ 0.02831433  0.18467622 -0.02133333 -0.24568871  0.54188834  0.34097766
  0.32691135  0.46605978 -0.04557475  0.72298733]


In [48]: P_min = lambda x : eval_P(x,M)
         Jac_min = lambda x : Jac_P(x,M)
         ans_sci = least_squares(P_min,x0,jac=Jac_min,verbose = 1)
         ans_sci.x

`gtol` termination condition is satisfied.
Function evaluations 33, initial cost 2.3821e+00, final cost 8.4610e-24, first-order optimality
```

```
Out[48]: array([-1.22474487e+00, -1.22474487e+00, -8.66025404e-01,  8.66025404e-01,
                 5.00000000e-01,  5.00000000e-01,  3.33333333e-01,  3.33333333e-01,
                -6.67085947e-12,  6.66666667e-01])
```

```
In [49]: def minimize(x0,M=M,func = eval_P,jac = Jac_P,num_jac=False,it = 1000,alpha = 1,ver =

             f = lambda x : func(x,M=M)


             if num_jac:

                 JAC_func = nd.Jacobian(f)

             else:

                 JAC_func = lambda x : jac(x,M)

             for i in range(it):

                 F = f(x0)

                 F_norm = np.linalg.norm(F,2)

                 jac_x0 = JAC_func(x0)

                 jac_F = jac_x0.T@F

                 jac_norm = np.linalg.norm(jac_F,2)

                 x0 = x0 - alpha*( ( F_norm/(jac_norm) )**2 )*jac_F #((F_norm)/(jac_norm**2))

                 if i%1000==0 and ver:
                     print('current function value:')
                     print(F_norm)
                     print('current x value:')

                     print('p:')
                     print(x0[2*N:3*N])
                     print('a:')
                     print(x0[3*N:4*N])
                     print('b:')
                     print(x0[4*N:5*N])

             return x0
```

```
In [50]: def chaos_moments(IFS , it = 1000, burn = 50):

             assert len(IFS)%3==0
```

```
    N = int(len(IFS)/3)
    p = IFS[:N]
    a = IFS[N:2*N]
    b = IFS[2*N:3*N]
    x = np.random.rand()
    X = np.zeros(it-burn)
    for i in range(it):
        P = np.random.rand()
        psum = p[0]
        for j in range(N):
            if psum > P:
                x = a[j]*x + b[j]
                if i>burn:
                    X[i - burn] = x
                break
            else:
                psum += p[j+1]
    return X
def elt(IFS,it = 1000,burn = 50,size = 6):
    X = chaos_moments(IFS , it = it, burn = burn)
    ans = np.zeros(size)
    for k in range(size):
        ans[k] = np.sum(np.power(X,k),axis = 0)/len(X)
    return ans
```

```
In [51]: #N*5 variables
         #n degree polynomial
         N = 2
         n = 10
         it = 1000 + 50 #plus 50 for the burn in

         Cant = [1/2,1/2,1/3,1/3,0,2/3]

         assert 3*N < n+1

         M0 = elt(Cant,it = it,size = n+1, burn = 50)

         print(M0)
```

```
[1.         0.5007407  0.3738223  0.31061918 0.26968128 0.23987053
 0.21692144 0.19869808 0.18391216 0.17170051 0.16145335]
```

```
In [52]: #x00 = np.random.rand(5*N)

         #this starting point was randomly generated. It shows that the standard numerical sol
         #but with pre-conditioning a solution can be found.

         x00 = np.array([0.41023015, 0.03003084, 0.5925644,  0.70754645, 0.0543441,  0.9686045

         print('x0:')
         print(x00)

         print('p:')
         print(np.round(x00[2*N:3*N],2))
         print('a:')
         print(np.round(x00[3*N:4*N],2))
         print('b:')
         print(np.round(x00[4*N:5*N],2))
```

```
x0:
[0.41023015 0.03003084 0.5925644  0.70754645 0.0543441  0.96860452
 0.28806056 0.92054965 0.83828167 0.24376342]
p:
[0.05 0.97]
a:
[0.29 0.92]
b:
[0.84 0.24]
```

```
In [53]: P_min0 = lambda x : eval_P(x,M0)
         Jac_min0 = lambda x : Jac_P(x,M0)
```

```
ans_sci0 = least_squares(P_min0,x00,jac=Jac_min0,verbose = 2)
ans_sci0.x
```

Iteration	Total nfev	Cost	Cost reduction	Step norm	Optimality
0	1	2.1591e+00			1.93e+01
1	3	7.1671e-01	1.44e+00	4.78e-01	2.74e+00
2	4	5.0885e-01	2.08e-01	9.56e-01	1.05e+00
3	6	2.4798e-01	2.61e-01	4.78e-01	5.56e-01
4	7	5.8825e-02	1.89e-01	9.56e-01	6.89e-01
5	9	3.3831e-02	2.50e-02	4.78e-01	1.28e+00
6	10	1.2482e-04	3.37e-02	4.78e-01	1.60e-02
7	13	9.6110e-05	2.87e-05	5.97e-02	1.14e-02
8	14	9.4087e-05	2.02e-06	1.19e-01	3.99e-02
9	15	7.7279e-05	1.68e-05	2.99e-02	1.63e-03
10	16	7.1972e-05	5.31e-06	5.97e-02	9.70e-03
11	17	7.1334e-05	6.38e-07	1.19e-01	4.00e-02
12	18	5.8289e-05	1.30e-05	2.99e-02	1.59e-03
13	19	5.4317e-05	3.97e-06	5.97e-02	9.30e-03
14	20	5.3836e-05	4.81e-07	1.19e-01	3.74e-02
15	21	4.3922e-05	9.91e-06	2.99e-02	1.44e-03
16	22	4.0881e-05	3.04e-06	5.97e-02	8.14e-03
17	24	3.8784e-05	2.10e-06	2.99e-02	1.76e-03
18	25	3.6091e-05	2.69e-06	5.97e-02	7.40e-03
19	27	3.4114e-05	1.98e-06	2.99e-02	1.60e-03
20	28	3.1794e-05	2.32e-06	5.97e-02	6.62e-03
21	30	2.9859e-05	1.93e-06	2.99e-02	1.42e-03
22	31	2.7947e-05	1.91e-06	5.97e-02	5.82e-03
23	32	2.5605e-05	2.34e-06	5.97e-02	5.21e-03
24	33	2.3448e-05	2.16e-06	5.97e-02	4.76e-03
25	34	2.1458e-05	1.99e-06	5.97e-02	4.35e-03
26	35	1.9624e-05	1.83e-06	5.97e-02	3.99e-03
27	36	1.7936e-05	1.69e-06	5.97e-02	3.68e-03
28	37	1.6381e-05	1.56e-06	5.97e-02	3.41e-03
29	38	1.4948e-05	1.43e-06	5.97e-02	3.74e-03
30	39	1.3626e-05	1.32e-06	5.97e-02	4.09e-03
31	40	1.2405e-05	1.22e-06	5.97e-02	4.42e-03
32	41	1.1276e-05	1.13e-06	5.97e-02	4.72e-03
33	42	1.0234e-05	1.04e-06	5.97e-02	5.00e-03
34	43	9.2715e-06	9.62e-07	5.97e-02	5.24e-03
35	44	8.3856e-06	8.86e-07	5.97e-02	5.45e-03
36	45	7.5724e-06	8.13e-07	5.97e-02	5.62e-03
37	46	6.8287e-06	7.44e-07	5.97e-02	5.77e-03
38	47	6.1511e-06	6.78e-07	5.97e-02	5.88e-03
39	48	5.5363e-06	6.15e-07	5.97e-02	5.97e-03
40	49	4.9803e-06	5.56e-07	5.97e-02	6.03e-03
41	50	4.4793e-06	5.01e-07	5.97e-02	6.07e-03
42	51	4.0289e-06	4.50e-07	5.97e-02	6.10e-03
43	52	3.6250e-06	4.04e-07	5.97e-02	6.11e-03

44	53	3.2634e-06	3.62e-07	5.97e-02	6.11e-03
45	54	2.9401e-06	3.23e-07	5.97e-02	6.10e-03
46	55	2.6512e-06	2.89e-07	5.97e-02	6.08e-03
47	56	1.7258e-06	9.25e-07	1.49e-02	3.11e-04
48	57	1.6847e-06	4.11e-08	2.99e-02	1.53e-03
49	58	1.5946e-06	9.01e-08	2.99e-02	1.50e-03
50	59	1.5115e-06	8.31e-08	2.99e-02	1.50e-03
51	60	1.4332e-06	7.83e-08	2.99e-02	1.49e-03
52	61	1.3595e-06	7.37e-08	2.99e-02	1.49e-03
53	62	1.2900e-06	6.95e-08	2.99e-02	1.48e-03
54	63	1.2244e-06	6.55e-08	2.99e-02	1.48e-03
55	64	1.1626e-06	6.18e-08	2.99e-02	1.47e-03
56	65	1.1044e-06	5.83e-08	2.99e-02	1.47e-03
57	66	1.0493e-06	5.50e-08	2.99e-02	1.46e-03
58	67	9.9741e-07	5.19e-08	2.99e-02	1.46e-03
59	68	9.4837e-07	4.90e-08	2.99e-02	1.45e-03
60	69	9.0204e-07	4.63e-08	2.99e-02	1.45e-03
61	70	8.5826e-07	4.38e-08	2.99e-02	1.44e-03
62	71	8.1688e-07	4.14e-08	2.99e-02	1.44e-03
63	72	7.7774e-07	3.91e-08	2.99e-02	1.43e-03
64	73	7.4072e-07	3.70e-08	2.99e-02	1.43e-03
65	74	7.0568e-07	3.50e-08	2.99e-02	1.42e-03
66	75	6.7251e-07	3.32e-08	2.99e-02	1.42e-03
67	76	6.4111e-07	3.14e-08	2.99e-02	1.41e-03
68	77	6.1136e-07	2.98e-08	2.99e-02	1.41e-03
69	78	5.8316e-07	2.82e-08	2.99e-02	1.40e-03
70	79	5.5644e-07	2.67e-08	2.99e-02	1.40e-03
71	80	5.3110e-07	2.53e-08	2.99e-02	1.39e-03
72	81	5.0706e-07	2.40e-08	2.99e-02	1.39e-03
73	82	4.8426e-07	2.28e-08	2.99e-02	1.38e-03
74	83	4.6261e-07	2.16e-08	2.99e-02	1.38e-03
75	84	4.4206e-07	2.06e-08	2.99e-02	1.37e-03
76	85	4.2254e-07	1.95e-08	2.99e-02	1.37e-03
77	86	4.0399e-07	1.85e-08	2.99e-02	1.36e-03
78	87	3.8637e-07	1.76e-08	2.99e-02	1.36e-03
79	88	3.6962e-07	1.68e-08	2.99e-02	1.36e-03
80	89	3.5368e-07	1.59e-08	2.99e-02	1.35e-03
81	90	3.3853e-07	1.52e-08	2.99e-02	1.35e-03
82	91	3.2411e-07	1.44e-08	2.99e-02	1.34e-03
83	92	3.1038e-07	1.37e-08	2.99e-02	1.34e-03
84	93	2.9732e-07	1.31e-08	2.99e-02	1.34e-03
85	94	2.8487e-07	1.24e-08	2.99e-02	1.33e-03
86	95	2.7302e-07	1.19e-08	2.99e-02	1.33e-03
87	96	2.6172e-07	1.13e-08	2.99e-02	1.33e-03
88	97	2.5096e-07	1.08e-08	2.99e-02	1.32e-03
89	98	2.4069e-07	1.03e-08	2.99e-02	1.32e-03
90	99	2.3090e-07	9.79e-09	2.99e-02	1.32e-03
91	100	2.2157e-07	9.34e-09	2.99e-02	1.31e-03

92	101	2.1266e-07	8.91e-09	2.99e-02	1.31e-03
93	102	2.0415e-07	8.50e-09	2.99e-02	1.31e-03
94	103	1.9603e-07	8.12e-09	2.99e-02	1.30e-03
95	104	1.8828e-07	7.75e-09	2.99e-02	1.30e-03
96	105	1.8088e-07	7.40e-09	2.99e-02	1.30e-03
97	106	1.7380e-07	7.07e-09	2.99e-02	1.29e-03
98	107	1.6704e-07	6.76e-09	2.99e-02	1.29e-03
99	108	1.6058e-07	6.46e-09	2.99e-02	1.29e-03
100	109	1.5440e-07	6.18e-09	2.99e-02	1.29e-03
101	110	1.4849e-07	5.91e-09	2.99e-02	1.28e-03
102	111	1.4284e-07	5.65e-09	2.99e-02	1.28e-03
103	112	1.3743e-07	5.41e-09	2.99e-02	1.28e-03
104	113	1.3225e-07	5.18e-09	2.99e-02	1.28e-03
105	114	1.2730e-07	4.95e-09	2.99e-02	1.27e-03
106	115	1.2256e-07	4.74e-09	2.99e-02	1.27e-03
107	116	1.1801e-07	4.54e-09	2.99e-02	1.27e-03
108	117	1.1366e-07	4.35e-09	2.99e-02	1.27e-03
109	118	1.0949e-07	4.17e-09	2.99e-02	1.28e-03
110	119	1.0549e-07	4.00e-09	2.99e-02	1.28e-03
111	120	1.0166e-07	3.83e-09	2.99e-02	1.28e-03
112	121	9.7992e-08	3.67e-09	2.99e-02	1.29e-03
113	122	9.4470e-08	3.52e-09	2.99e-02	1.29e-03
114	123	9.1093e-08	3.38e-09	2.99e-02	1.29e-03
115	124	8.7852e-08	3.24e-09	2.99e-02	1.30e-03
116	125	8.4743e-08	3.11e-09	2.99e-02	1.30e-03
117	126	8.1759e-08	2.98e-09	2.99e-02	1.30e-03
118	127	7.8894e-08	2.86e-09	2.99e-02	1.31e-03
119	128	7.6144e-08	2.75e-09	2.99e-02	1.31e-03
120	129	7.3503e-08	2.64e-09	2.99e-02	1.31e-03
121	130	7.0966e-08	2.54e-09	2.99e-02	1.32e-03
122	131	6.8529e-08	2.44e-09	2.99e-02	1.32e-03
123	132	6.6187e-08	2.34e-09	2.99e-02	1.32e-03
124	133	6.3936e-08	2.25e-09	2.99e-02	1.32e-03
125	134	6.1772e-08	2.16e-09	2.99e-02	1.33e-03
126	135	5.9692e-08	2.08e-09	2.99e-02	1.33e-03
127	136	5.7692e-08	2.00e-09	2.99e-02	1.33e-03
128	137	5.5768e-08	1.92e-09	2.99e-02	1.33e-03
129	138	5.3917e-08	1.85e-09	2.99e-02	1.34e-03
130	139	5.2137e-08	1.78e-09	2.99e-02	1.34e-03
131	140	5.0423e-08	1.71e-09	2.99e-02	1.34e-03
132	141	4.8774e-08	1.65e-09	2.99e-02	1.34e-03
133	142	4.7186e-08	1.59e-09	2.99e-02	1.34e-03
134	143	4.5657e-08	1.53e-09	2.99e-02	1.35e-03
135	144	4.4185e-08	1.47e-09	2.99e-02	1.35e-03
136	145	4.2767e-08	1.42e-09	2.99e-02	1.35e-03
137	146	4.1401e-08	1.37e-09	2.99e-02	1.35e-03
138	147	4.0085e-08	1.32e-09	2.99e-02	1.35e-03
139	148	3.8817e-08	1.27e-09	2.99e-02	1.36e-03

140	149	3.7595e-08	1.22e-09	2.99e-02	1.36e-03
141	150	3.6416e-08	1.18e-09	2.99e-02	1.36e-03
142	151	3.5280e-08	1.14e-09	2.99e-02	1.36e-03
143	152	3.4185e-08	1.10e-09	2.99e-02	1.36e-03
144	153	3.3128e-08	1.06e-09	2.99e-02	1.37e-03
145	154	3.2109e-08	1.02e-09	2.99e-02	1.37e-03
146	155	3.1126e-08	9.83e-10	2.99e-02	1.37e-03
147	156	3.0177e-08	9.49e-10	2.99e-02	1.37e-03
148	157	2.7109e-08	3.07e-09	7.47e-03	8.35e-05
149	158	2.6844e-08	2.65e-10	1.49e-02	3.44e-04
150	159	2.6417e-08	4.27e-10	1.49e-02	3.43e-04
151	160	2.5999e-08	4.18e-10	1.49e-02	3.43e-04
152	161	2.5588e-08	4.11e-10	1.49e-02	3.43e-04
153	162	2.5185e-08	4.03e-10	1.49e-02	3.44e-04
154	163	2.4789e-08	3.96e-10	1.49e-02	3.44e-04
155	164	2.4399e-08	3.89e-10	1.49e-02	3.44e-04
156	165	2.4017e-08	3.82e-10	1.49e-02	3.44e-04
157	166	2.3642e-08	3.76e-10	1.49e-02	3.44e-04
158	167	2.3273e-08	3.69e-10	1.49e-02	3.44e-04
159	168	2.2910e-08	3.62e-10	1.49e-02	3.45e-04
160	169	2.2554e-08	3.56e-10	1.49e-02	3.45e-04
161	170	2.2204e-08	3.50e-10	1.49e-02	3.45e-04
162	171	2.1860e-08	3.44e-10	1.49e-02	3.45e-04
163	172	2.1522e-08	3.38e-10	1.49e-02	3.45e-04
164	173	2.1190e-08	3.32e-10	1.49e-02	3.46e-04
165	174	2.0864e-08	3.26e-10	1.49e-02	3.46e-04
166	175	2.0544e-08	3.21e-10	1.49e-02	3.46e-04
167	176	2.0229e-08	3.15e-10	1.49e-02	3.46e-04
168	177	1.9919e-08	3.10e-10	1.49e-02	3.46e-04
169	178	1.9615e-08	3.04e-10	1.49e-02	3.46e-04
170	179	1.9316e-08	2.99e-10	1.49e-02	3.47e-04
171	180	1.9022e-08	2.94e-10	1.49e-02	3.47e-04
172	181	1.8733e-08	2.89e-10	1.49e-02	3.47e-04
173	182	1.8449e-08	2.84e-10	1.49e-02	3.47e-04
174	183	1.8170e-08	2.79e-10	1.49e-02	3.47e-04
175	184	1.7896e-08	2.74e-10	1.49e-02	3.47e-04
176	185	1.7626e-08	2.70e-10	1.49e-02	3.48e-04
177	186	1.7361e-08	2.65e-10	1.49e-02	3.48e-04
178	187	1.7101e-08	2.61e-10	1.49e-02	3.48e-04
179	188	1.6844e-08	2.56e-10	1.49e-02	3.48e-04
180	189	1.6592e-08	2.52e-10	1.49e-02	3.48e-04
181	190	1.6345e-08	2.48e-10	1.49e-02	3.48e-04
182	191	1.6101e-08	2.43e-10	1.49e-02	3.48e-04
183	192	1.5862e-08	2.39e-10	1.49e-02	3.49e-04
184	193	1.5626e-08	2.35e-10	1.49e-02	3.49e-04
185	194	1.5395e-08	2.31e-10	1.49e-02	3.49e-04
186	195	1.5167e-08	2.28e-10	1.49e-02	3.49e-04
187	196	1.4944e-08	2.24e-10	1.49e-02	3.49e-04

188	197	1.4723e-08	2.20e-10	1.49e-02	3.49e-04
189	198	1.4507e-08	2.16e-10	1.49e-02	3.49e-04
190	199	1.4294e-08	2.13e-10	1.49e-02	3.50e-04
191	200	1.4085e-08	2.09e-10	1.49e-02	3.50e-04
192	201	1.3879e-08	2.06e-10	1.49e-02	3.50e-04
193	202	1.3676e-08	2.03e-10	1.49e-02	3.50e-04
194	203	1.3477e-08	1.99e-10	1.49e-02	3.50e-04
195	204	1.3281e-08	1.96e-10	1.49e-02	3.50e-04
196	205	1.3088e-08	1.93e-10	1.49e-02	3.50e-04
197	206	1.2899e-08	1.90e-10	1.49e-02	3.50e-04
198	207	1.2712e-08	1.87e-10	1.49e-02	3.51e-04
199	208	1.2529e-08	1.83e-10	1.49e-02	3.51e-04
200	209	1.2348e-08	1.80e-10	1.49e-02	3.51e-04
201	210	1.2171e-08	1.78e-10	1.49e-02	3.51e-04
202	211	1.1996e-08	1.75e-10	1.49e-02	3.51e-04
203	212	1.1824e-08	1.72e-10	1.49e-02	3.51e-04
204	213	1.1655e-08	1.69e-10	1.49e-02	3.51e-04
205	214	1.1489e-08	1.66e-10	1.49e-02	3.51e-04
206	215	1.1325e-08	1.64e-10	1.49e-02	3.52e-04
207	216	1.1164e-08	1.61e-10	1.49e-02	3.52e-04
208	217	1.1005e-08	1.59e-10	1.49e-02	3.52e-04
209	218	1.0849e-08	1.56e-10	1.49e-02	3.52e-04
210	219	1.0696e-08	1.53e-10	1.49e-02	3.52e-04
211	220	1.0545e-08	1.51e-10	1.49e-02	3.52e-04
212	221	1.0396e-08	1.49e-10	1.49e-02	3.52e-04
213	222	1.0250e-08	1.46e-10	1.49e-02	3.52e-04
214	223	1.0106e-08	1.44e-10	1.49e-02	3.52e-04
215	224	9.9643e-09	1.42e-10	1.49e-02	3.53e-04
216	225	9.8249e-09	1.39e-10	1.49e-02	3.53e-04
217	226	9.6876e-09	1.37e-10	1.49e-02	3.53e-04
218	227	9.5525e-09	1.35e-10	1.49e-02	3.53e-04
219	228	9.4195e-09	1.33e-10	1.49e-02	3.53e-04
220	229	9.2886e-09	1.31e-10	1.49e-02	3.53e-04
221	230	9.1598e-09	1.29e-10	1.49e-02	3.53e-04
222	231	9.0330e-09	1.27e-10	1.49e-02	3.53e-04
223	232	8.9081e-09	1.25e-10	1.49e-02	3.53e-04
224	233	8.7852e-09	1.23e-10	1.49e-02	3.53e-04
225	234	8.6642e-09	1.21e-10	1.49e-02	3.54e-04
226	235	8.5450e-09	1.19e-10	1.49e-02	3.54e-04
227	236	8.4278e-09	1.17e-10	1.49e-02	3.54e-04
228	237	8.3123e-09	1.15e-10	1.49e-02	3.54e-04
229	238	8.1986e-09	1.14e-10	1.49e-02	3.54e-04
230	239	8.0867e-09	1.12e-10	1.49e-02	3.54e-04
231	240	7.9764e-09	1.10e-10	1.49e-02	3.54e-04
232	241	7.8679e-09	1.09e-10	1.49e-02	3.54e-04
233	242	7.7610e-09	1.07e-10	1.49e-02	3.54e-04
234	243	7.6558e-09	1.05e-10	1.49e-02	3.54e-04
235	244	7.5522e-09	1.04e-10	1.49e-02	3.54e-04

236	245	7.4501e-09	1.02e-10	1.49e-02	3.55e-04
237	246	7.3496e-09	1.00e-10	1.49e-02	3.55e-04
238	247	7.2507e-09	9.90e-11	1.49e-02	3.55e-04
239	248	7.1532e-09	9.75e-11	1.49e-02	3.55e-04
240	249	7.0572e-09	9.60e-11	1.49e-02	3.55e-04
241	250	6.9627e-09	9.45e-11	1.49e-02	3.55e-04
242	251	6.8695e-09	9.31e-11	1.49e-02	3.55e-04
243	252	6.7778e-09	9.17e-11	1.49e-02	3.55e-04
244	253	6.6875e-09	9.03e-11	1.49e-02	3.55e-04
245	254	6.5985e-09	8.90e-11	1.49e-02	3.55e-04
246	255	6.5109e-09	8.76e-11	1.49e-02	3.55e-04
247	256	6.4245e-09	8.63e-11	1.49e-02	3.56e-04
248	257	6.3395e-09	8.50e-11	1.49e-02	3.56e-04
249	258	6.2557e-09	8.38e-11	1.49e-02	3.56e-04
250	259	6.1732e-09	8.25e-11	1.49e-02	3.56e-04
251	260	6.0919e-09	8.13e-11	1.49e-02	3.56e-04
252	261	6.0118e-09	8.01e-11	1.49e-02	3.56e-04
253	262	5.9328e-09	7.89e-11	1.49e-02	3.56e-04
254	263	5.8551e-09	7.77e-11	1.49e-02	3.56e-04
255	264	5.7785e-09	7.66e-11	1.49e-02	3.56e-04
256	265	5.7030e-09	7.55e-11	1.49e-02	3.56e-04
257	266	5.6287e-09	7.44e-11	1.49e-02	3.56e-04
258	267	5.5554e-09	7.33e-11	1.49e-02	3.56e-04
259	268	5.4832e-09	7.22e-11	1.49e-02	3.56e-04
260	269	5.4121e-09	7.11e-11	1.49e-02	3.57e-04
261	270	5.3420e-09	7.01e-11	1.49e-02	3.57e-04
262	271	5.2729e-09	6.91e-11	1.49e-02	3.57e-04
263	272	5.2048e-09	6.81e-11	1.49e-02	3.57e-04
264	273	5.1377e-09	6.71e-11	1.49e-02	3.57e-04
265	274	5.0716e-09	6.61e-11	1.49e-02	3.57e-04
266	275	5.0065e-09	6.51e-11	1.49e-02	3.57e-04
267	276	4.9423e-09	6.42e-11	1.49e-02	3.57e-04
268	277	4.8790e-09	6.33e-11	1.49e-02	3.57e-04
269	278	4.8166e-09	6.24e-11	1.49e-02	3.57e-04
270	279	4.7552e-09	6.15e-11	1.49e-02	3.57e-04
271	280	4.6946e-09	6.06e-11	1.49e-02	3.57e-04
272	281	4.6349e-09	5.97e-11	1.49e-02	3.57e-04
273	282	4.5760e-09	5.89e-11	1.49e-02	3.57e-04
274	283	4.5180e-09	5.80e-11	1.49e-02	3.57e-04
275	284	4.4608e-09	5.72e-11	1.49e-02	3.58e-04
276	285	4.4044e-09	5.64e-11	1.49e-02	3.58e-04
277	286	4.3489e-09	5.56e-11	1.49e-02	3.58e-04
278	287	4.2941e-09	5.48e-11	1.49e-02	3.58e-04
279	288	4.2401e-09	5.40e-11	1.49e-02	3.58e-04
280	289	4.1869e-09	5.32e-11	1.49e-02	3.58e-04
281	290	4.1344e-09	5.25e-11	1.49e-02	3.58e-04
282	291	4.0826e-09	5.17e-11	1.49e-02	3.58e-04
283	292	4.0316e-09	5.10e-11	1.49e-02	3.58e-04

284	293	3.9813e-09	5.03e-11	1.49e-02	3.58e-04
285	294	3.9317e-09	4.96e-11	1.49e-02	3.58e-04
286	295	3.8828e-09	4.89e-11	1.49e-02	3.58e-04
287	296	3.8346e-09	4.82e-11	1.49e-02	3.58e-04
288	297	3.7871e-09	4.75e-11	1.49e-02	3.58e-04
289	298	3.7402e-09	4.69e-11	1.49e-02	3.58e-04
290	299	3.6940e-09	4.62e-11	1.49e-02	3.58e-04
291	300	3.6485e-09	4.56e-11	1.49e-02	3.58e-04
292	301	3.6035e-09	4.49e-11	1.49e-02	3.59e-04
293	302	3.5592e-09	4.43e-11	1.49e-02	3.59e-04
294	303	3.5155e-09	4.37e-11	1.49e-02	3.59e-04
295	304	3.4724e-09	4.31e-11	1.49e-02	3.59e-04
296	305	3.4299e-09	4.25e-11	1.49e-02	3.59e-04
297	306	3.3880e-09	4.19e-11	1.49e-02	3.59e-04
298	307	3.3467e-09	4.13e-11	1.49e-02	3.59e-04
299	308	3.3059e-09	4.08e-11	1.49e-02	3.59e-04
300	309	3.2657e-09	4.02e-11	1.49e-02	3.59e-04
301	310	3.2261e-09	3.97e-11	1.49e-02	3.59e-04
302	311	3.1869e-09	3.91e-11	1.49e-02	3.59e-04
303	312	3.1484e-09	3.86e-11	1.49e-02	3.59e-04
304	313	3.1103e-09	3.80e-11	1.49e-02	3.59e-04
305	314	3.0728e-09	3.75e-11	1.49e-02	3.59e-04
306	315	3.0358e-09	3.70e-11	1.49e-02	3.59e-04
307	316	2.9993e-09	3.65e-11	1.49e-02	3.59e-04
308	317	2.9632e-09	3.60e-11	1.49e-02	3.59e-04
309	318	2.9277e-09	3.55e-11	1.49e-02	3.59e-04
310	319	2.8926e-09	3.51e-11	1.49e-02	3.59e-04
311	320	2.8581e-09	3.46e-11	1.49e-02	3.59e-04
312	321	2.8239e-09	3.41e-11	1.49e-02	3.60e-04
313	322	2.7903e-09	3.37e-11	1.49e-02	3.60e-04
314	323	2.7571e-09	3.32e-11	1.49e-02	3.60e-04
315	324	2.7243e-09	3.28e-11	1.49e-02	3.60e-04
316	325	2.6920e-09	3.23e-11	1.49e-02	3.60e-04
317	326	2.6601e-09	3.19e-11	1.49e-02	3.60e-04
318	327	2.6287e-09	3.15e-11	1.49e-02	3.60e-04
319	328	2.5976e-09	3.10e-11	1.49e-02	3.60e-04
320	329	2.5670e-09	3.06e-11	1.49e-02	3.60e-04
321	330	2.5368e-09	3.02e-11	1.49e-02	3.60e-04
322	331	2.5070e-09	2.98e-11	1.49e-02	3.60e-04
323	332	2.4775e-09	2.94e-11	1.49e-02	3.60e-04
324	333	2.4485e-09	2.90e-11	1.49e-02	3.60e-04
325	334	2.4199e-09	2.86e-11	1.49e-02	3.60e-04
326	335	2.3916e-09	2.83e-11	1.49e-02	3.60e-04
327	336	2.3637e-09	2.79e-11	1.49e-02	3.60e-04
328	337	2.3362e-09	2.75e-11	1.49e-02	3.60e-04
329	338	2.3090e-09	2.72e-11	1.49e-02	3.60e-04
330	339	2.2822e-09	2.68e-11	1.49e-02	3.60e-04
331	340	2.2557e-09	2.65e-11	1.49e-02	3.60e-04

332	341	2.2296e-09	2.61e-11	1.49e-02	3.60e-04
333	342	2.2038e-09	2.58e-11	1.49e-02	3.60e-04
334	343	2.1784e-09	2.54e-11	1.49e-02	3.60e-04
335	344	2.1533e-09	2.51e-11	1.49e-02	3.60e-04
336	345	2.1285e-09	2.48e-11	1.49e-02	3.61e-04
337	346	2.1041e-09	2.45e-11	1.49e-02	3.61e-04
338	347	2.0799e-09	2.41e-11	1.49e-02	3.61e-04
339	348	2.0561e-09	2.38e-11	1.49e-02	3.61e-04
340	349	2.0326e-09	2.35e-11	1.49e-02	3.61e-04
341	350	2.0094e-09	2.32e-11	1.49e-02	3.61e-04
342	351	1.9864e-09	2.29e-11	1.49e-02	3.61e-04
343	352	1.9638e-09	2.26e-11	1.49e-02	3.61e-04
344	353	1.9415e-09	2.23e-11	1.49e-02	3.61e-04
345	354	1.9194e-09	2.20e-11	1.49e-02	3.61e-04
346	355	1.8977e-09	2.18e-11	1.49e-02	3.61e-04
347	356	1.8762e-09	2.15e-11	1.49e-02	3.61e-04
348	357	1.8550e-09	2.12e-11	1.49e-02	3.61e-04
349	358	1.8340e-09	2.09e-11	1.49e-02	3.61e-04
350	359	1.8134e-09	2.07e-11	1.49e-02	3.61e-04
351	360	1.7930e-09	2.04e-11	1.49e-02	3.61e-04
352	361	1.7728e-09	2.01e-11	1.49e-02	3.61e-04
353	362	1.7529e-09	1.99e-11	1.49e-02	3.61e-04
354	363	1.7333e-09	1.96e-11	1.49e-02	3.61e-04
355	364	1.7139e-09	1.94e-11	1.49e-02	3.61e-04
356	365	1.6948e-09	1.91e-11	1.49e-02	3.61e-04
357	366	1.6758e-09	1.89e-11	1.49e-02	3.61e-04
358	367	1.6572e-09	1.87e-11	1.49e-02	3.61e-04
359	368	1.6387e-09	1.84e-11	1.49e-02	3.61e-04
360	369	1.6205e-09	1.82e-11	1.49e-02	3.61e-04
361	370	1.6026e-09	1.80e-11	1.49e-02	3.61e-04
362	371	1.5848e-09	1.77e-11	1.49e-02	3.61e-04
363	372	1.5673e-09	1.75e-11	1.49e-02	3.61e-04
364	373	1.5500e-09	1.73e-11	1.49e-02	3.61e-04
365	374	1.5329e-09	1.71e-11	1.49e-02	3.61e-04
366	375	1.5160e-09	1.69e-11	1.49e-02	3.62e-04
367	376	1.4994e-09	1.67e-11	1.49e-02	3.62e-04
368	377	1.4829e-09	1.65e-11	1.49e-02	3.62e-04
369	378	1.4666e-09	1.63e-11	1.49e-02	3.62e-04
370	379	1.4506e-09	1.61e-11	1.49e-02	3.62e-04
371	380	1.4347e-09	1.59e-11	1.49e-02	3.62e-04
372	381	1.4191e-09	1.57e-11	1.49e-02	3.62e-04
373	382	1.4036e-09	1.55e-11	1.49e-02	3.62e-04
374	383	1.3883e-09	1.53e-11	1.49e-02	3.62e-04
375	384	1.3732e-09	1.51e-11	1.49e-02	3.62e-04
376	385	1.3583e-09	1.49e-11	1.49e-02	3.62e-04
377	386	1.3436e-09	1.47e-11	1.49e-02	3.62e-04
378	387	1.3291e-09	1.45e-11	1.49e-02	3.62e-04
379	388	1.3147e-09	1.44e-11	1.49e-02	3.62e-04

380	389	1.3005e-09	1.42e-11	1.49e-02	3.62e-04
381	390	1.2865e-09	1.40e-11	1.49e-02	3.62e-04
382	391	1.2727e-09	1.38e-11	1.49e-02	3.62e-04
383	392	1.2590e-09	1.37e-11	1.49e-02	3.62e-04
384	393	1.2455e-09	1.35e-11	1.49e-02	3.62e-04
385	394	1.2321e-09	1.33e-11	1.49e-02	3.62e-04
386	395	1.2190e-09	1.32e-11	1.49e-02	3.62e-04
387	396	1.2059e-09	1.30e-11	1.49e-02	3.62e-04
388	397	1.1931e-09	1.29e-11	1.49e-02	3.62e-04
389	398	1.1804e-09	1.27e-11	1.49e-02	3.62e-04
390	399	1.1678e-09	1.26e-11	1.49e-02	3.62e-04
391	400	1.1554e-09	1.24e-11	1.49e-02	3.62e-04
392	401	1.1431e-09	1.23e-11	1.49e-02	3.62e-04
393	402	1.1310e-09	1.21e-11	1.49e-02	3.62e-04
394	403	1.1191e-09	1.20e-11	1.49e-02	3.62e-04
395	404	1.1072e-09	1.18e-11	1.49e-02	3.62e-04
396	405	1.0956e-09	1.17e-11	1.49e-02	3.62e-04
397	406	1.0840e-09	1.15e-11	1.49e-02	3.62e-04
398	407	1.0726e-09	1.14e-11	1.49e-02	3.62e-04
399	408	1.0613e-09	1.13e-11	1.49e-02	3.62e-04
400	409	1.0502e-09	1.11e-11	1.49e-02	3.62e-04
401	410	1.0392e-09	1.10e-11	1.49e-02	3.62e-04
402	411	1.0283e-09	1.09e-11	1.49e-02	3.62e-04
403	412	1.0176e-09	1.07e-11	1.49e-02	3.62e-04
404	413	1.0070e-09	1.06e-11	1.49e-02	3.62e-04
405	414	9.9648e-10	1.05e-11	1.49e-02	3.62e-04
406	415	9.8611e-10	1.04e-11	1.49e-02	3.62e-04
407	416	9.7587e-10	1.02e-11	1.49e-02	3.63e-04
408	417	9.6574e-10	1.01e-11	1.49e-02	3.63e-04
409	418	9.5573e-10	1.00e-11	1.49e-02	3.63e-04
410	419	9.4585e-10	9.89e-12	1.49e-02	3.63e-04
411	420	9.3607e-10	9.77e-12	1.49e-02	3.63e-04
412	421	9.2641e-10	9.66e-12	1.49e-02	3.63e-04
413	422	9.1687e-10	9.55e-12	1.49e-02	3.63e-04
414	423	8.8601e-10	3.09e-11	3.73e-03	2.24e-05
415	424	8.8305e-10	2.96e-12	7.47e-03	9.07e-05
416	425	8.7843e-10	4.62e-12	7.47e-03	9.06e-05
417	426	8.7384e-10	4.59e-12	7.47e-03	9.06e-05
418	427	8.6927e-10	4.56e-12	7.47e-03	9.06e-05
419	428	8.6474e-10	4.54e-12	7.47e-03	9.06e-05
420	429	8.6023e-10	4.51e-12	7.47e-03	9.06e-05
421	430	8.5574e-10	4.48e-12	7.47e-03	9.06e-05
422	431	8.5129e-10	4.46e-12	7.47e-03	9.06e-05
423	432	8.4685e-10	4.43e-12	7.47e-03	9.06e-05
424	433	8.4245e-10	4.41e-12	7.47e-03	9.07e-05
425	434	8.3807e-10	4.38e-12	7.47e-03	9.07e-05
426	435	8.3372e-10	4.35e-12	7.47e-03	9.07e-05
427	436	8.2939e-10	4.33e-12	7.47e-03	9.07e-05

428	437	8.2508e-10	4.30e-12	7.47e-03	9.07e-05
429	438	8.2080e-10	4.28e-12	7.47e-03	9.07e-05
430	439	8.1655e-10	4.25e-12	7.47e-03	9.07e-05
431	440	8.1232e-10	4.23e-12	7.47e-03	9.07e-05
432	441	8.0812e-10	4.20e-12	7.47e-03	9.07e-05
433	442	8.0394e-10	4.18e-12	7.47e-03	9.07e-05
434	443	7.9978e-10	4.16e-12	7.47e-03	9.07e-05
435	444	7.9565e-10	4.13e-12	7.47e-03	9.07e-05
436	445	7.9155e-10	4.11e-12	7.47e-03	9.07e-05
437	446	7.8746e-10	4.08e-12	7.47e-03	9.07e-05
438	447	7.8340e-10	4.06e-12	7.47e-03	9.07e-05
439	448	7.7937e-10	4.04e-12	7.47e-03	9.07e-05
440	449	7.7535e-10	4.01e-12	7.47e-03	9.07e-05
441	450	7.7137e-10	3.99e-12	7.47e-03	9.07e-05
442	451	7.6740e-10	3.97e-12	7.47e-03	9.07e-05
443	452	7.6346e-10	3.94e-12	7.47e-03	9.07e-05
444	453	7.5953e-10	3.92e-12	7.47e-03	9.07e-05
445	454	7.5564e-10	3.90e-12	7.47e-03	9.07e-05
446	455	7.5176e-10	3.88e-12	7.47e-03	9.07e-05
447	456	7.4791e-10	3.85e-12	7.47e-03	9.07e-05
448	457	7.4408e-10	3.83e-12	7.47e-03	9.07e-05
449	458	7.4027e-10	3.81e-12	7.47e-03	9.07e-05
450	459	7.3648e-10	3.79e-12	7.47e-03	9.07e-05
451	460	7.3272e-10	3.77e-12	7.47e-03	9.07e-05
452	461	7.2897e-10	3.74e-12	7.47e-03	9.07e-05
453	462	7.2525e-10	3.72e-12	7.47e-03	9.07e-05
454	463	7.2155e-10	3.70e-12	7.47e-03	9.07e-05
455	464	7.1787e-10	3.68e-12	7.47e-03	9.07e-05
456	465	7.1421e-10	3.66e-12	7.47e-03	9.07e-05
457	466	7.1057e-10	3.64e-12	7.47e-03	9.07e-05
458	467	7.0696e-10	3.62e-12	7.47e-03	9.07e-05
459	468	7.0336e-10	3.60e-12	7.47e-03	9.07e-05
460	469	6.9979e-10	3.58e-12	7.47e-03	9.07e-05
461	470	6.9623e-10	3.55e-12	7.47e-03	9.07e-05
462	471	6.9270e-10	3.53e-12	7.47e-03	9.07e-05
463	472	6.8918e-10	3.51e-12	7.47e-03	9.07e-05
464	473	6.8569e-10	3.49e-12	7.47e-03	9.07e-05
465	474	6.8221e-10	3.47e-12	7.47e-03	9.07e-05
466	475	6.7876e-10	3.45e-12	7.47e-03	9.07e-05
467	476	6.7532e-10	3.44e-12	7.47e-03	9.07e-05
468	477	6.7191e-10	3.42e-12	7.47e-03	9.07e-05
469	478	6.6851e-10	3.40e-12	7.47e-03	9.07e-05
470	479	6.6513e-10	3.38e-12	7.47e-03	9.07e-05
471	480	6.6178e-10	3.36e-12	7.47e-03	9.07e-05
472	481	6.5844e-10	3.34e-12	7.47e-03	9.07e-05
473	482	6.5512e-10	3.32e-12	7.47e-03	9.08e-05
474	483	6.5182e-10	3.30e-12	7.47e-03	9.08e-05
475	484	6.4853e-10	3.28e-12	7.47e-03	9.08e-05

476	485	6.4527e-10	3.26e-12	7.47e-03	9.08e-05
477	486	6.4203e-10	3.25e-12	7.47e-03	9.08e-05
478	487	6.3880e-10	3.23e-12	7.47e-03	9.08e-05
479	488	6.3559e-10	3.21e-12	7.47e-03	9.08e-05
480	489	6.3240e-10	3.19e-12	7.47e-03	9.08e-05
481	490	6.2923e-10	3.17e-12	7.47e-03	9.08e-05
482	491	6.2607e-10	3.15e-12	7.47e-03	9.08e-05
483	492	6.2294e-10	3.14e-12	7.47e-03	9.08e-05
484	493	6.1982e-10	3.12e-12	7.47e-03	9.08e-05
485	494	6.1672e-10	3.10e-12	7.47e-03	9.08e-05
486	495	6.1363e-10	3.08e-12	7.47e-03	9.08e-05
487	496	6.1056e-10	3.07e-12	7.47e-03	9.08e-05
488	497	6.0752e-10	3.05e-12	7.47e-03	9.08e-05
489	498	6.0448e-10	3.03e-12	7.47e-03	9.08e-05
490	499	6.0147e-10	3.02e-12	7.47e-03	9.08e-05
491	500	5.9847e-10	3.00e-12	7.47e-03	9.08e-05
492	501	5.9549e-10	2.98e-12	7.47e-03	9.08e-05
493	502	5.9252e-10	2.97e-12	7.47e-03	9.08e-05
494	503	5.8957e-10	2.95e-12	7.47e-03	9.08e-05
495	504	5.8664e-10	2.93e-12	7.47e-03	9.08e-05
496	505	5.8373e-10	2.92e-12	7.47e-03	9.08e-05
497	506	5.8083e-10	2.90e-12	7.47e-03	9.08e-05
498	507	5.7794e-10	2.88e-12	7.47e-03	9.08e-05
499	508	5.7508e-10	2.87e-12	7.47e-03	9.08e-05
500	509	5.7223e-10	2.85e-12	7.47e-03	9.08e-05
501	510	5.6939e-10	2.84e-12	7.47e-03	9.08e-05
502	511	5.6657e-10	2.82e-12	7.47e-03	9.08e-05
503	512	5.6377e-10	2.80e-12	7.47e-03	9.08e-05
504	513	5.6098e-10	2.79e-12	7.47e-03	9.08e-05
505	514	5.5821e-10	2.77e-12	7.47e-03	9.08e-05
506	515	5.5545e-10	2.76e-12	7.47e-03	9.08e-05
507	516	5.5271e-10	2.74e-12	7.47e-03	9.08e-05
508	517	5.4998e-10	2.73e-12	7.47e-03	9.08e-05
509	518	5.4727e-10	2.71e-12	7.47e-03	9.08e-05
510	519	5.4457e-10	2.70e-12	7.47e-03	9.08e-05
511	520	5.4189e-10	2.68e-12	7.47e-03	9.08e-05
512	521	5.3923e-10	2.67e-12	7.47e-03	9.08e-05
513	522	5.3657e-10	2.65e-12	7.47e-03	9.08e-05
514	523	5.3394e-10	2.64e-12	7.47e-03	9.08e-05
515	524	5.3131e-10	2.62e-12	7.47e-03	9.08e-05
516	525	5.2871e-10	2.61e-12	7.47e-03	9.08e-05
517	526	5.2611e-10	2.59e-12	7.47e-03	9.08e-05
518	527	5.2353e-10	2.58e-12	7.47e-03	9.08e-05
519	528	5.2097e-10	2.57e-12	7.47e-03	9.08e-05
520	529	5.1842e-10	2.55e-12	7.47e-03	9.08e-05
521	530	5.1588e-10	2.54e-12	7.47e-03	9.08e-05
522	531	5.1336e-10	2.52e-12	7.47e-03	9.08e-05
523	532	5.1085e-10	2.51e-12	7.47e-03	9.08e-05

524	533	5.0835e-10	2.50e-12	7.47e-03	9.08e-05
525	534	5.0587e-10	2.48e-12	7.47e-03	9.08e-05
526	535	5.0340e-10	2.47e-12	7.47e-03	9.08e-05
527	536	5.0095e-10	2.45e-12	7.47e-03	9.08e-05
528	537	4.9851e-10	2.44e-12	7.47e-03	9.08e-05
529	538	4.9608e-10	2.43e-12	7.47e-03	9.08e-05
530	539	4.9366e-10	2.41e-12	7.47e-03	9.08e-05
531	540	4.9126e-10	2.40e-12	7.47e-03	9.08e-05
532	541	4.8887e-10	2.39e-12	7.47e-03	9.08e-05
533	542	4.8650e-10	2.38e-12	7.47e-03	9.08e-05
534	543	4.8414e-10	2.36e-12	7.47e-03	9.08e-05
535	544	4.8179e-10	2.35e-12	7.47e-03	9.08e-05
536	545	4.7945e-10	2.34e-12	7.47e-03	9.08e-05
537	546	4.7713e-10	2.32e-12	7.47e-03	9.09e-05
538	547	4.7482e-10	2.31e-12	7.47e-03	9.09e-05
539	548	4.7252e-10	2.30e-12	7.47e-03	9.09e-05
540	549	4.7023e-10	2.29e-12	7.47e-03	9.09e-05
541	550	4.6796e-10	2.27e-12	7.47e-03	9.09e-05
542	551	4.6570e-10	2.26e-12	7.47e-03	9.09e-05
543	552	4.6345e-10	2.25e-12	7.47e-03	9.09e-05
544	553	4.6121e-10	2.24e-12	7.47e-03	9.09e-05
545	554	4.5899e-10	2.22e-12	7.47e-03	9.09e-05
546	555	4.5677e-10	2.21e-12	7.47e-03	9.09e-05
547	556	4.5457e-10	2.20e-12	7.47e-03	9.09e-05
548	557	4.5239e-10	2.19e-12	7.47e-03	9.09e-05
549	558	4.5021e-10	2.18e-12	7.47e-03	9.09e-05
550	559	4.4804e-10	2.16e-12	7.47e-03	9.09e-05
551	560	4.4589e-10	2.15e-12	7.47e-03	9.09e-05
552	561	4.4375e-10	2.14e-12	7.47e-03	9.09e-05
553	562	4.4162e-10	2.13e-12	7.47e-03	9.09e-05
554	563	4.3950e-10	2.12e-12	7.47e-03	9.09e-05
555	564	4.3739e-10	2.11e-12	7.47e-03	9.09e-05
556	565	4.3530e-10	2.10e-12	7.47e-03	9.09e-05
557	566	4.3321e-10	2.08e-12	7.47e-03	9.09e-05
558	567	4.3114e-10	2.07e-12	7.47e-03	9.09e-05
559	568	4.2908e-10	2.06e-12	7.47e-03	9.09e-05
560	569	4.2703e-10	2.05e-12	7.47e-03	9.09e-05
561	570	4.2499e-10	2.04e-12	7.47e-03	9.09e-05
562	571	4.2296e-10	2.03e-12	7.47e-03	9.09e-05
563	572	4.2094e-10	2.02e-12	7.47e-03	9.09e-05
564	573	4.1893e-10	2.01e-12	7.47e-03	9.09e-05
565	574	4.1694e-10	2.00e-12	7.47e-03	9.09e-05
566	575	4.1495e-10	1.99e-12	7.47e-03	9.09e-05
567	576	4.1297e-10	1.98e-12	7.47e-03	9.09e-05
568	577	4.1101e-10	1.96e-12	7.47e-03	9.09e-05
569	578	4.0906e-10	1.95e-12	7.47e-03	9.09e-05
570	579	4.0711e-10	1.94e-12	7.47e-03	9.09e-05
571	580	4.0518e-10	1.93e-12	7.47e-03	9.09e-05

572	581	4.0326e-10	1.92e-12	7.47e-03	9.09e-05
573	582	4.0134e-10	1.91e-12	7.47e-03	9.09e-05
574	583	3.9944e-10	1.90e-12	7.47e-03	9.09e-05
575	584	3.9755e-10	1.89e-12	7.47e-03	9.09e-05
576	585	3.9567e-10	1.88e-12	7.47e-03	9.09e-05
577	586	3.9379e-10	1.87e-12	7.47e-03	9.09e-05
578	587	3.9193e-10	1.86e-12	7.47e-03	9.09e-05
579	588	3.9008e-10	1.85e-12	7.47e-03	9.09e-05
580	589	3.8824e-10	1.84e-12	7.47e-03	9.09e-05
581	590	3.8640e-10	1.83e-12	7.47e-03	9.09e-05
582	591	3.8458e-10	1.82e-12	7.47e-03	9.09e-05
583	592	3.8277e-10	1.81e-12	7.47e-03	9.09e-05
584	593	3.8096e-10	1.80e-12	7.47e-03	9.09e-05
585	594	3.7917e-10	1.79e-12	7.47e-03	9.09e-05
586	595	3.7738e-10	1.78e-12	7.47e-03	9.09e-05
587	596	3.7561e-10	1.78e-12	7.47e-03	9.09e-05
588	597	3.7384e-10	1.77e-12	7.47e-03	9.09e-05
589	598	3.7209e-10	1.76e-12	7.47e-03	9.09e-05
590	599	3.7034e-10	1.75e-12	7.47e-03	9.09e-05
591	600	3.6860e-10	1.74e-12	7.47e-03	9.09e-05
592	601	3.6687e-10	1.73e-12	7.47e-03	9.09e-05
593	602	3.6515e-10	1.72e-12	7.47e-03	9.09e-05
594	603	3.6344e-10	1.71e-12	7.47e-03	9.09e-05
595	604	3.6174e-10	1.70e-12	7.47e-03	9.09e-05
596	605	3.6005e-10	1.69e-12	7.47e-03	9.09e-05
597	606	3.5836e-10	1.68e-12	7.47e-03	9.09e-05
598	607	3.5669e-10	1.67e-12	7.47e-03	9.09e-05
599	608	3.5502e-10	1.67e-12	7.47e-03	9.09e-05
600	609	3.5337e-10	1.66e-12	7.47e-03	9.09e-05
601	610	3.5172e-10	1.65e-12	7.47e-03	9.09e-05
602	611	3.5008e-10	1.64e-12	7.47e-03	9.09e-05
603	612	3.4845e-10	1.63e-12	7.47e-03	9.09e-05
604	613	3.4682e-10	1.62e-12	7.47e-03	9.09e-05
605	614	3.4521e-10	1.61e-12	7.47e-03	9.09e-05
606	615	3.4360e-10	1.61e-12	7.47e-03	9.09e-05
607	616	3.4200e-10	1.60e-12	7.47e-03	9.09e-05
608	617	3.4042e-10	1.59e-12	7.47e-03	9.09e-05
609	618	3.3883e-10	1.58e-12	7.47e-03	9.09e-05
610	619	3.3726e-10	1.57e-12	7.47e-03	9.09e-05
611	620	3.3570e-10	1.56e-12	7.47e-03	9.09e-05
612	621	3.3414e-10	1.56e-12	7.47e-03	9.09e-05
613	622	3.3259e-10	1.55e-12	7.47e-03	9.09e-05
614	623	3.3105e-10	1.54e-12	7.47e-03	9.09e-05
615	624	3.2952e-10	1.53e-12	7.47e-03	9.09e-05
616	625	3.2800e-10	1.52e-12	7.47e-03	9.09e-05
617	626	3.2648e-10	1.52e-12	7.47e-03	9.09e-05
618	627	3.2497e-10	1.51e-12	7.47e-03	9.09e-05
619	628	3.2347e-10	1.50e-12	7.47e-03	9.09e-05

620	629	3.2198e-10	1.49e-12	7.47e-03	9.09e-05
621	630	3.2049e-10	1.48e-12	7.47e-03	9.09e-05
622	631	3.1902e-10	1.48e-12	7.47e-03	9.09e-05
623	632	3.1755e-10	1.47e-12	7.47e-03	9.09e-05
624	633	3.1609e-10	1.46e-12	7.47e-03	9.09e-05
625	634	3.1463e-10	1.45e-12	7.47e-03	9.09e-05
626	635	3.1318e-10	1.45e-12	7.47e-03	9.09e-05
627	636	3.1174e-10	1.44e-12	7.47e-03	9.09e-05
628	637	3.1031e-10	1.43e-12	7.47e-03	9.09e-05
629	638	3.0889e-10	1.42e-12	7.47e-03	9.09e-05
630	639	3.0747e-10	1.42e-12	7.47e-03	9.09e-05
631	640	3.0606e-10	1.41e-12	7.47e-03	9.09e-05
632	641	3.0466e-10	1.40e-12	7.47e-03	9.09e-05
633	642	3.0326e-10	1.40e-12	7.47e-03	9.09e-05
634	643	3.0187e-10	1.39e-12	7.47e-03	9.09e-05
635	644	3.0049e-10	1.38e-12	7.47e-03	9.10e-05
636	645	2.9912e-10	1.37e-12	7.47e-03	9.10e-05
637	646	2.9775e-10	1.37e-12	7.47e-03	9.10e-05
638	647	2.9639e-10	1.36e-12	7.47e-03	9.10e-05
639	648	2.9504e-10	1.35e-12	7.47e-03	9.10e-05
640	649	2.9369e-10	1.35e-12	7.47e-03	9.10e-05
641	650	2.9236e-10	1.34e-12	7.47e-03	9.10e-05
642	651	2.9102e-10	1.33e-12	7.47e-03	9.10e-05
643	652	2.8970e-10	1.33e-12	7.47e-03	9.10e-05
644	653	2.8838e-10	1.32e-12	7.47e-03	9.10e-05
645	654	2.8707e-10	1.31e-12	7.47e-03	9.10e-05
646	655	2.8576e-10	1.31e-12	7.47e-03	9.10e-05
647	656	2.8446e-10	1.30e-12	7.47e-03	9.10e-05
648	657	2.8317e-10	1.29e-12	7.47e-03	9.10e-05
649	658	2.8189e-10	1.29e-12	7.47e-03	9.10e-05
650	659	2.8061e-10	1.28e-12	7.47e-03	9.10e-05
651	660	2.7934e-10	1.27e-12	7.47e-03	9.10e-05
652	661	2.7807e-10	1.27e-12	7.47e-03	9.10e-05
653	662	2.7681e-10	1.26e-12	7.47e-03	9.10e-05
654	663	2.7556e-10	1.25e-12	7.47e-03	9.10e-05
655	664	2.7431e-10	1.25e-12	7.47e-03	9.10e-05
656	665	2.7307e-10	1.24e-12	7.47e-03	9.10e-05
657	666	2.7184e-10	1.23e-12	7.47e-03	9.10e-05
658	667	2.7061e-10	1.23e-12	7.47e-03	9.10e-05
659	668	2.6939e-10	1.22e-12	7.47e-03	9.10e-05
660	669	2.6817e-10	1.22e-12	7.47e-03	9.10e-05
661	670	2.6697e-10	1.21e-12	7.47e-03	9.10e-05
662	671	2.6576e-10	1.20e-12	7.47e-03	9.10e-05
663	672	2.6457e-10	1.20e-12	7.47e-03	9.10e-05
664	673	2.6338e-10	1.19e-12	7.47e-03	9.10e-05
665	674	2.6219e-10	1.18e-12	7.47e-03	9.10e-05
666	675	2.6101e-10	1.18e-12	7.47e-03	9.10e-05
667	676	2.5984e-10	1.17e-12	7.47e-03	9.10e-05

668	677	2.5867e-10	1.17e-12	7.47e-03	9.10e-05
669	678	2.5751e-10	1.16e-12	7.47e-03	9.10e-05
670	679	2.5636e-10	1.15e-12	7.47e-03	9.10e-05
671	680	2.5521e-10	1.15e-12	7.47e-03	9.10e-05
672	681	2.5406e-10	1.14e-12	7.47e-03	9.10e-05
673	682	2.5293e-10	1.14e-12	7.47e-03	9.10e-05
674	683	2.5180e-10	1.13e-12	7.47e-03	9.10e-05
675	684	2.5067e-10	1.13e-12	7.47e-03	9.10e-05
676	685	2.4955e-10	1.12e-12	7.47e-03	9.10e-05
677	686	2.4843e-10	1.11e-12	7.47e-03	9.10e-05
678	687	2.4732e-10	1.11e-12	7.47e-03	9.10e-05
679	688	2.4622e-10	1.10e-12	7.47e-03	9.10e-05
680	689	2.4512e-10	1.10e-12	7.47e-03	9.10e-05
681	690	2.4403e-10	1.09e-12	7.47e-03	9.10e-05
682	691	2.4294e-10	1.09e-12	7.47e-03	9.10e-05
683	692	2.4186e-10	1.08e-12	7.47e-03	9.10e-05
684	693	2.4079e-10	1.08e-12	7.47e-03	9.10e-05
685	694	2.3971e-10	1.07e-12	7.47e-03	9.10e-05
686	695	2.3865e-10	1.07e-12	7.47e-03	9.10e-05
687	696	2.3759e-10	1.06e-12	7.47e-03	9.10e-05
688	697	2.3653e-10	1.05e-12	7.47e-03	9.10e-05
689	698	2.3548e-10	1.05e-12	7.47e-03	9.10e-05
690	699	2.3444e-10	1.04e-12	7.47e-03	9.10e-05
691	700	2.3340e-10	1.04e-12	7.47e-03	9.10e-05
692	701	2.3237e-10	1.03e-12	7.47e-03	9.10e-05
693	702	2.3134e-10	1.03e-12	7.47e-03	9.10e-05
694	703	2.3031e-10	1.02e-12	7.47e-03	9.10e-05
695	704	2.2929e-10	1.02e-12	7.47e-03	9.10e-05
696	705	2.2828e-10	1.01e-12	7.47e-03	9.10e-05
697	706	2.2727e-10	1.01e-12	7.47e-03	9.10e-05
698	707	2.2627e-10	1.00e-12	7.47e-03	9.10e-05
699	708	2.2527e-10	9.99e-13	7.47e-03	9.10e-05
700	709	2.2428e-10	9.94e-13	7.47e-03	9.10e-05
701	710	2.2329e-10	9.89e-13	7.47e-03	9.10e-05
702	711	2.2230e-10	9.84e-13	7.47e-03	9.10e-05
703	712	2.2133e-10	9.79e-13	7.47e-03	9.10e-05
704	713	2.2035e-10	9.74e-13	7.47e-03	9.10e-05
705	714	2.1938e-10	9.69e-13	7.47e-03	9.10e-05
706	715	2.1842e-10	9.64e-13	7.47e-03	9.10e-05
707	716	2.1746e-10	9.60e-13	7.47e-03	9.10e-05
708	717	2.1650e-10	9.55e-13	7.47e-03	9.10e-05
709	718	2.1555e-10	9.50e-13	7.47e-03	9.10e-05
710	719	2.1461e-10	9.46e-13	7.47e-03	9.10e-05
711	720	2.1367e-10	9.41e-13	7.47e-03	9.10e-05
712	721	2.1273e-10	9.36e-13	7.47e-03	9.10e-05
713	722	2.1180e-10	9.32e-13	7.47e-03	9.10e-05
714	723	2.1087e-10	9.27e-13	7.47e-03	9.10e-05
715	724	2.0995e-10	9.22e-13	7.47e-03	9.10e-05

716	725	2.0903e-10	9.18e-13	7.47e-03	9.10e-05
717	726	2.0812e-10	9.13e-13	7.47e-03	9.10e-05
718	727	2.0721e-10	9.09e-13	7.47e-03	9.10e-05
719	728	2.0630e-10	9.04e-13	7.47e-03	9.10e-05
720	729	2.0540e-10	9.00e-13	7.47e-03	9.10e-05
721	730	2.0451e-10	8.96e-13	7.47e-03	9.10e-05
722	731	2.0362e-10	8.91e-13	7.47e-03	9.10e-05
723	732	2.0273e-10	8.87e-13	7.47e-03	9.10e-05
724	733	2.0185e-10	8.82e-13	7.47e-03	9.10e-05
725	734	2.0097e-10	8.78e-13	7.47e-03	9.10e-05
726	735	2.0010e-10	8.74e-13	7.47e-03	9.10e-05
727	736	1.9923e-10	8.70e-13	7.47e-03	9.10e-05
728	737	1.9836e-10	8.65e-13	7.47e-03	9.10e-05
729	738	1.9750e-10	8.61e-13	7.47e-03	9.10e-05
730	739	1.9664e-10	8.57e-13	7.47e-03	9.10e-05
731	740	1.9579e-10	8.53e-13	7.47e-03	9.10e-05
732	741	1.9494e-10	8.49e-13	7.47e-03	9.10e-05
733	742	1.9410e-10	8.44e-13	7.47e-03	9.10e-05
734	743	1.9326e-10	8.40e-13	7.47e-03	9.10e-05
735	744	1.9242e-10	8.36e-13	7.47e-03	9.10e-05
736	745	1.9159e-10	8.32e-13	7.47e-03	9.10e-05
737	746	1.9076e-10	8.28e-13	7.47e-03	9.10e-05
738	747	1.8994e-10	8.24e-13	7.47e-03	9.10e-05
739	748	1.8912e-10	8.20e-13	7.47e-03	9.10e-05
740	749	1.8830e-10	8.16e-13	7.47e-03	9.10e-05
741	750	1.8749e-10	8.12e-13	7.47e-03	9.10e-05
742	751	1.8668e-10	8.08e-13	7.47e-03	9.10e-05
743	752	1.8588e-10	8.04e-13	7.47e-03	9.10e-05
744	753	1.8508e-10	8.00e-13	7.47e-03	9.10e-05
745	754	1.8428e-10	7.96e-13	7.47e-03	9.10e-05
746	755	1.8349e-10	7.93e-13	7.47e-03	9.10e-05
747	756	1.8270e-10	7.89e-13	7.47e-03	9.10e-05
748	757	1.8191e-10	7.85e-13	7.47e-03	9.10e-05
749	758	1.8113e-10	7.81e-13	7.47e-03	9.10e-05
750	759	1.8035e-10	7.77e-13	7.47e-03	9.10e-05
751	760	1.7958e-10	7.74e-13	7.47e-03	9.10e-05
752	761	1.7881e-10	7.70e-13	7.47e-03	9.10e-05
753	762	1.7804e-10	7.66e-13	7.47e-03	9.10e-05
754	763	1.7728e-10	7.63e-13	7.47e-03	9.10e-05
755	764	1.7652e-10	7.59e-13	7.47e-03	9.10e-05
756	765	1.7577e-10	7.55e-13	7.47e-03	9.10e-05
757	766	1.7502e-10	7.52e-13	7.47e-03	9.10e-05
758	767	1.7427e-10	7.48e-13	7.47e-03	9.10e-05
759	768	1.7352e-10	7.44e-13	7.47e-03	9.10e-05
760	769	1.7278e-10	7.41e-13	7.47e-03	9.10e-05
761	770	1.7205e-10	7.37e-13	7.47e-03	9.10e-05
762	771	1.7131e-10	7.34e-13	7.47e-03	9.10e-05
763	772	1.7058e-10	7.30e-13	7.47e-03	9.10e-05

764	773	1.6986e-10	7.27e-13	7.47e-03	9.10e-05
765	774	1.6913e-10	7.23e-13	7.47e-03	9.10e-05
766	775	1.6841e-10	7.20e-13	7.47e-03	9.10e-05
767	776	1.6770e-10	7.16e-13	7.47e-03	9.10e-05
768	777	1.6698e-10	7.13e-13	7.47e-03	9.10e-05
769	778	1.6627e-10	7.09e-13	7.47e-03	9.10e-05
770	779	1.6557e-10	7.06e-13	7.47e-03	9.10e-05
771	780	1.6487e-10	7.03e-13	7.47e-03	9.10e-05
772	781	1.6417e-10	6.99e-13	7.47e-03	9.10e-05
773	782	1.6347e-10	6.96e-13	7.47e-03	9.10e-05
774	783	1.6278e-10	6.93e-13	7.47e-03	9.10e-05
775	784	1.6209e-10	6.89e-13	7.47e-03	9.10e-05
776	785	1.6140e-10	6.86e-13	7.47e-03	9.10e-05
777	786	1.6072e-10	6.83e-13	7.47e-03	9.10e-05
778	787	1.6004e-10	6.80e-13	7.47e-03	9.10e-05
779	788	1.5936e-10	6.76e-13	7.47e-03	9.10e-05
780	789	1.5869e-10	6.73e-13	7.47e-03	9.10e-05
781	790	1.5802e-10	6.70e-13	7.47e-03	9.10e-05
782	791	1.5735e-10	6.67e-13	7.47e-03	9.10e-05
783	792	1.5669e-10	6.64e-13	7.47e-03	9.10e-05
784	793	1.5603e-10	6.60e-13	7.47e-03	9.10e-05
785	794	1.5537e-10	6.57e-13	7.47e-03	9.10e-05
786	795	1.5472e-10	6.54e-13	7.47e-03	9.10e-05
787	796	1.5407e-10	6.51e-13	7.47e-03	9.10e-05
788	797	1.5342e-10	6.48e-13	7.47e-03	9.10e-05
789	798	1.5277e-10	6.45e-13	7.47e-03	9.10e-05
790	799	1.5213e-10	6.42e-13	7.47e-03	9.10e-05
791	800	1.5149e-10	6.39e-13	7.47e-03	9.10e-05
792	801	1.5086e-10	6.36e-13	7.47e-03	9.10e-05
793	802	1.5022e-10	6.33e-13	7.47e-03	9.10e-05
794	803	1.4960e-10	6.30e-13	7.47e-03	9.10e-05
795	804	1.4897e-10	6.27e-13	7.47e-03	9.10e-05
796	805	1.4834e-10	6.24e-13	7.47e-03	9.10e-05
797	806	1.4772e-10	6.21e-13	7.47e-03	9.10e-05
798	807	1.4711e-10	6.18e-13	7.47e-03	9.10e-05
799	808	1.4649e-10	6.15e-13	7.47e-03	9.10e-05
800	809	1.4588e-10	6.12e-13	7.47e-03	9.10e-05
801	810	1.4527e-10	6.09e-13	7.47e-03	9.10e-05
802	811	1.4466e-10	6.07e-13	7.47e-03	9.10e-05
803	812	1.4406e-10	6.04e-13	7.47e-03	9.10e-05
804	813	1.4346e-10	6.01e-13	7.47e-03	9.10e-05
805	814	1.4286e-10	5.98e-13	7.47e-03	9.10e-05
806	815	1.4226e-10	5.95e-13	7.47e-03	9.10e-05
807	816	1.4167e-10	5.92e-13	7.47e-03	9.10e-05
808	817	1.4108e-10	5.90e-13	7.47e-03	9.10e-05
809	818	1.4050e-10	5.87e-13	7.47e-03	9.10e-05
810	819	1.3991e-10	5.84e-13	7.47e-03	9.10e-05
811	820	1.3933e-10	5.81e-13	7.47e-03	9.10e-05

812	821	1.3875e-10	5.79e-13	7.47e-03	9.10e-05
813	822	1.3818e-10	5.76e-13	7.47e-03	9.10e-05
814	823	1.3760e-10	5.73e-13	7.47e-03	9.10e-05
815	824	1.3703e-10	5.71e-13	7.47e-03	9.10e-05
816	825	1.3646e-10	5.68e-13	7.47e-03	9.10e-05
817	826	1.3590e-10	5.65e-13	7.47e-03	9.10e-05
818	827	1.3534e-10	5.63e-13	7.47e-03	9.10e-05
819	828	1.3478e-10	5.60e-13	7.47e-03	9.10e-05
820	829	1.3422e-10	5.57e-13	7.47e-03	9.10e-05
821	830	1.3366e-10	5.55e-13	7.47e-03	9.10e-05
822	831	1.3311e-10	5.52e-13	7.47e-03	9.10e-05
823	832	1.3256e-10	5.50e-13	7.47e-03	9.10e-05
824	833	1.3201e-10	5.47e-13	7.47e-03	9.10e-05
825	834	1.3147e-10	5.45e-13	7.47e-03	9.10e-05
826	835	1.3093e-10	5.42e-13	7.47e-03	9.10e-05
827	836	1.3039e-10	5.40e-13	7.47e-03	9.10e-05
828	837	1.2985e-10	5.37e-13	7.47e-03	9.10e-05
829	838	1.2932e-10	5.35e-13	7.47e-03	9.10e-05
830	839	1.2878e-10	5.32e-13	7.47e-03	9.10e-05
831	840	1.2825e-10	5.30e-13	7.47e-03	9.10e-05
832	841	1.2773e-10	5.27e-13	7.47e-03	9.10e-05
833	842	1.2720e-10	5.25e-13	7.47e-03	9.10e-05
834	843	1.2668e-10	5.22e-13	7.47e-03	9.10e-05
835	844	1.2616e-10	5.20e-13	7.47e-03	9.10e-05
836	845	1.2564e-10	5.17e-13	7.47e-03	9.10e-05
837	846	1.2513e-10	5.15e-13	7.47e-03	9.10e-05
838	847	1.2462e-10	5.13e-13	7.47e-03	9.10e-05
839	848	1.2410e-10	5.10e-13	7.47e-03	9.10e-05
840	849	1.2360e-10	5.08e-13	7.47e-03	9.10e-05
841	850	1.2309e-10	5.06e-13	7.47e-03	9.10e-05
842	851	1.2259e-10	5.03e-13	7.47e-03	9.10e-05
843	852	1.2209e-10	5.01e-13	7.47e-03	9.10e-05
844	853	1.2159e-10	4.99e-13	7.47e-03	9.10e-05
845	854	1.2109e-10	4.96e-13	7.47e-03	9.10e-05
846	855	1.2060e-10	4.94e-13	7.47e-03	9.10e-05
847	856	1.2011e-10	4.92e-13	7.47e-03	9.10e-05
848	857	1.1962e-10	4.90e-13	7.47e-03	9.10e-05
849	858	1.1913e-10	4.87e-13	7.47e-03	9.10e-05
850	859	1.1864e-10	4.85e-13	7.47e-03	9.10e-05
851	860	1.1816e-10	4.83e-13	7.47e-03	9.10e-05
852	861	1.1768e-10	4.81e-13	7.47e-03	9.10e-05
853	862	1.1720e-10	4.79e-13	7.47e-03	9.10e-05
854	863	1.1672e-10	4.76e-13	7.47e-03	9.10e-05
855	864	1.1625e-10	4.74e-13	7.47e-03	9.10e-05
856	865	1.1578e-10	4.72e-13	7.47e-03	9.10e-05
857	866	1.1531e-10	4.70e-13	7.47e-03	9.10e-05
858	867	1.1484e-10	4.68e-13	7.47e-03	9.10e-05
859	868	1.1438e-10	4.66e-13	7.47e-03	9.10e-05

860	869	1.1391e-10	4.63e-13	7.47e-03	9.10e-05
861	870	1.1345e-10	4.61e-13	7.47e-03	9.10e-05
862	871	1.1299e-10	4.59e-13	7.47e-03	9.10e-05
863	872	1.1253e-10	4.57e-13	7.47e-03	9.10e-05
864	873	1.1208e-10	4.55e-13	7.47e-03	9.10e-05
865	874	1.1163e-10	4.53e-13	7.47e-03	9.10e-05
866	875	1.1118e-10	4.51e-13	7.47e-03	9.10e-05
867	876	1.1073e-10	4.49e-13	7.47e-03	9.10e-05
868	877	1.1028e-10	4.47e-13	7.47e-03	9.10e-05
869	878	1.0984e-10	4.45e-13	7.47e-03	9.10e-05
870	879	1.0939e-10	4.43e-13	7.47e-03	9.10e-05
871	880	1.0895e-10	4.41e-13	7.47e-03	9.10e-05
872	881	1.0851e-10	4.39e-13	7.47e-03	9.10e-05
873	882	1.0808e-10	4.37e-13	7.47e-03	9.10e-05
874	883	1.0764e-10	4.35e-13	7.47e-03	9.10e-05
875	884	1.0721e-10	4.33e-13	7.47e-03	9.10e-05
876	885	1.0678e-10	4.31e-13	7.47e-03	9.10e-05
877	886	1.0635e-10	4.29e-13	7.47e-03	9.10e-05
878	887	1.0592e-10	4.27e-13	7.47e-03	9.10e-05
879	888	1.0550e-10	4.25e-13	7.47e-03	9.10e-05
880	889	1.0507e-10	4.23e-13	7.47e-03	9.10e-05
881	890	1.0465e-10	4.21e-13	7.47e-03	9.10e-05
882	891	1.0423e-10	4.19e-13	7.47e-03	9.10e-05
883	892	1.0382e-10	4.17e-13	7.47e-03	9.10e-05
884	893	1.0340e-10	4.16e-13	7.47e-03	9.10e-05
885	894	1.0299e-10	4.14e-13	7.47e-03	9.10e-05
886	895	1.0257e-10	4.12e-13	7.47e-03	9.10e-05
887	896	1.0217e-10	4.10e-13	7.47e-03	9.10e-05
888	897	1.0176e-10	4.08e-13	7.47e-03	9.10e-05
889	898	1.0135e-10	4.06e-13	7.47e-03	9.10e-05
890	899	1.0095e-10	4.04e-13	7.47e-03	9.10e-05
891	900	1.0054e-10	4.03e-13	7.47e-03	9.10e-05
892	901	1.0014e-10	4.01e-13	7.47e-03	9.10e-05
893	902	9.9744e-11	3.99e-13	7.47e-03	9.10e-05
894	903	9.9346e-11	3.97e-13	7.47e-03	9.10e-05
895	904	9.8951e-11	3.95e-13	7.47e-03	9.10e-05
896	905	9.8557e-11	3.94e-13	7.47e-03	9.10e-05
897	906	9.8165e-11	3.92e-13	7.47e-03	9.10e-05
898	907	9.7775e-11	3.90e-13	7.47e-03	9.10e-05
899	908	9.7387e-11	3.88e-13	7.47e-03	9.10e-05
900	909	9.7000e-11	3.87e-13	7.47e-03	9.10e-05
901	910	9.6615e-11	3.85e-13	7.47e-03	9.10e-05
902	911	9.6232e-11	3.83e-13	7.47e-03	9.10e-05
903	912	9.5850e-11	3.82e-13	7.47e-03	9.10e-05
904	913	9.5471e-11	3.80e-13	7.47e-03	9.10e-05
905	914	9.5092e-11	3.78e-13	7.47e-03	9.10e-05
906	915	9.4716e-11	3.76e-13	7.47e-03	9.10e-05
907	916	9.4341e-11	3.75e-13	7.47e-03	9.10e-05

908	917	9.3968e-11	3.73e-13	7.47e-03	9.10e-05
909	918	9.3597e-11	3.71e-13	7.47e-03	9.10e-05
910	919	9.3227e-11	3.70e-13	7.47e-03	9.10e-05
911	920	9.2859e-11	3.68e-13	7.47e-03	9.10e-05
912	921	9.2492e-11	3.67e-13	7.47e-03	9.10e-05
913	922	9.2127e-11	3.65e-13	7.47e-03	9.10e-05
914	923	9.1764e-11	3.63e-13	7.47e-03	9.10e-05
915	924	9.1402e-11	3.62e-13	7.47e-03	9.10e-05
916	925	9.1042e-11	3.60e-13	7.47e-03	9.10e-05
917	926	9.0684e-11	3.58e-13	7.47e-03	9.10e-05
918	927	9.0327e-11	3.57e-13	7.47e-03	9.10e-05
919	928	8.9972e-11	3.55e-13	7.47e-03	9.10e-05
920	929	8.9618e-11	3.54e-13	7.47e-03	9.10e-05
921	930	8.9266e-11	3.52e-13	7.47e-03	9.10e-05
922	931	8.8915e-11	3.51e-13	7.47e-03	9.10e-05
923	932	8.8566e-11	3.49e-13	7.47e-03	9.10e-05
924	933	8.8219e-11	3.47e-13	7.47e-03	9.10e-05
925	934	8.7873e-11	3.46e-13	7.47e-03	9.10e-05
926	935	8.7528e-11	3.44e-13	7.47e-03	9.10e-05
927	936	8.7185e-11	3.43e-13	7.47e-03	9.10e-05
928	937	8.6844e-11	3.41e-13	7.47e-03	9.10e-05
929	938	8.6504e-11	3.40e-13	7.47e-03	9.10e-05
930	939	8.6166e-11	3.38e-13	7.47e-03	9.10e-05
931	940	8.5829e-11	3.37e-13	7.47e-03	9.10e-05
932	941	8.5493e-11	3.35e-13	7.47e-03	9.10e-05
933	942	8.5160e-11	3.34e-13	7.47e-03	9.10e-05
934	943	8.4827e-11	3.32e-13	7.47e-03	9.10e-05
935	944	8.4496e-11	3.31e-13	7.47e-03	9.10e-05
936	945	8.4166e-11	3.30e-13	7.47e-03	9.10e-05
937	946	8.3838e-11	3.28e-13	7.47e-03	9.10e-05
938	947	8.3512e-11	3.27e-13	7.47e-03	9.10e-05
939	948	8.3186e-11	3.25e-13	7.47e-03	9.10e-05
940	949	8.2863e-11	3.24e-13	7.47e-03	9.10e-05
941	950	8.2540e-11	3.22e-13	7.47e-03	9.10e-05
942	951	8.2219e-11	3.21e-13	7.47e-03	9.10e-05
943	952	8.1900e-11	3.20e-13	7.47e-03	9.10e-05
944	953	8.1581e-11	3.18e-13	7.47e-03	9.10e-05
945	954	8.1265e-11	3.17e-13	7.47e-03	9.10e-05
946	955	8.0949e-11	3.15e-13	7.47e-03	9.10e-05
947	956	8.0635e-11	3.14e-13	7.47e-03	9.10e-05
948	957	8.0323e-11	3.13e-13	7.47e-03	9.10e-05
949	958	8.0011e-11	3.11e-13	7.47e-03	9.10e-05
950	959	7.9701e-11	3.10e-13	7.47e-03	9.10e-05
951	960	7.9393e-11	3.09e-13	7.47e-03	9.10e-05
952	961	7.9085e-11	3.07e-13	7.47e-03	9.10e-05
953	962	7.8780e-11	3.06e-13	7.47e-03	9.10e-05
954	963	7.8475e-11	3.05e-13	7.47e-03	9.10e-05
955	964	7.8172e-11	3.03e-13	7.47e-03	9.10e-05

```
     956             965       7.7870e-11      3.02e-13        7.47e-03        9.10e-05
     957             966       7.7569e-11      3.01e-13        7.47e-03        9.10e-05
     958             967       7.7270e-11      2.99e-13        7.47e-03        9.10e-05
     959             968       7.6972e-11      2.98e-13        7.47e-03        9.10e-05
     960             969       7.6675e-11      2.97e-13        7.47e-03        9.10e-05
     961             970       7.6380e-11      2.95e-13        7.47e-03        9.10e-05
     962             971       7.6086e-11      2.94e-13        7.47e-03        9.10e-05
     963             972       7.5793e-11      2.93e-13        7.47e-03        9.10e-05
     964             973       7.5501e-11      2.92e-13        7.47e-03        9.10e-05
     965             974       7.5211e-11      2.90e-13        7.47e-03        9.10e-05
     966             975       7.4922e-11      2.89e-13        7.47e-03        9.10e-05
     967             976       7.4634e-11      2.88e-13        7.47e-03        9.10e-05
     968             977       7.4347e-11      2.87e-13        7.47e-03        9.10e-05
     969             978       7.4062e-11      2.85e-13        7.47e-03        9.10e-05
     970             979       7.3778e-11      2.84e-13        7.47e-03        9.10e-05
     971             980       7.3495e-11      2.83e-13        7.47e-03        9.10e-05
     972             981       7.3213e-11      2.82e-13        7.47e-03        9.10e-05
     973             982       7.2933e-11      2.80e-13        7.47e-03        9.10e-05
     974             983       7.2654e-11      2.79e-13        7.47e-03        9.10e-05
     975             984       7.2376e-11      2.78e-13        7.47e-03        9.10e-05
     976             985       7.2099e-11      2.77e-13        7.47e-03        9.10e-05
     977             986       7.1823e-11      2.76e-13        7.47e-03        9.10e-05
     978             987       7.1549e-11      2.74e-13        7.47e-03        9.10e-05
     979             988       7.1275e-11      2.73e-13        7.47e-03        9.10e-05
     980             989       7.1003e-11      2.72e-13        7.47e-03        9.10e-05
     981             990       7.0732e-11      2.71e-13        7.47e-03        9.10e-05
     982             991       7.0462e-11      2.70e-13        7.47e-03        9.10e-05
     983             992       7.0194e-11      2.69e-13        7.47e-03        9.10e-05
     984             993       6.9926e-11      2.67e-13        7.47e-03        9.10e-05
     985             994       6.9660e-11      2.66e-13        7.47e-03        9.10e-05
     986             995       6.9395e-11      2.65e-13        7.47e-03        9.10e-05
     987             996       6.9131e-11      2.64e-13        7.47e-03        9.10e-05
     988             997       6.8868e-11      2.63e-13        7.47e-03        9.10e-05
     989             998       6.8606e-11      2.62e-13        7.47e-03        9.10e-05
     990             999       6.8346e-11      2.61e-13        7.47e-03        9.10e-05
     991            1000       6.8086e-11      2.60e-13        7.47e-03        9.10e-05
The maximum number of function evaluations is exceeded.
Function evaluations 1000, initial cost 2.1591e+00, final cost 6.8086e-11, first-order optimali
```

```
Out[53]: array([ 9.74799504e+00,  1.23722532e+01,  7.08974503e-01,  7.08264472e-01,
                -1.63976031e+00,  2.63976066e+00,  9.89476285e-01,  9.93467158e-01,
                 8.25578884e-03,  5.12702681e-03])
```

```
In [54]: print('p:')
         print(np.round(ans_sci0.x[2*N:3*N],2))
         print('a:')
         print(np.round(ans_sci0.x[3*N:4*N],2))
```

```
print('b:')
print(np.round(ans_sci0.x[4*N:5*N],2))
```

```
p:
[-1.64  2.64]
a:
[0.99 0.99]
b:
[0.01 0.01]
```

Run Altman iteration, notice does not converge to a solution fast enough.

```
In [55]: ans0 = minimize(x00,M,func = eval_P,jac = Jac_P,it = 50000,alpha = 1)
         print(ans0)
```

```
current function value:
1.948288337969674
current x value:
p:
[0.01324846 0.94275908]
a:
[0.27954766 0.794987  ]
b:
[0.82537406 0.10089225]
current function value:
0.010213893184832243
current x value:
p:
[0.4128026  0.58677862]
a:
[-0.17561489  0.41631621]
b:
[0.97704017 0.01971222]
current function value:
0.002170198303562757
current x value:
p:
[0.43819975 0.56197506]
a:
[-0.24569332  0.45241101]
b:
[ 0.99354151 -0.01429564]
current function value:
0.002731233983096262
current x value:
p:
[0.44249437 0.55756047]
a:
```

```
[-0.255687    0.45640679]
b:
[ 0.99494447 -0.01919274]
current function value:
0.001987452255747147
current x value:
p:
[0.44386302 0.55620796]
a:
[-0.25856529  0.45734543]
b:
[ 0.9954293  -0.02054957]
current function value:
0.0013269165036542412
current x value:
p:
[0.44489579 0.55524195]
a:
[-0.26053077  0.45788638]
b:
[ 0.9958717  -0.02145084]
current function value:
0.0020014103753316324
current x value:
p:
[0.44600887 0.55393159]
a:
[-0.26328086  0.45832413]
b:
[ 0.99584449 -0.02256725]
current function value:
0.0010287890905309753
current x value:
p:
[0.44752553 0.55283218]
a:
[-0.26510901  0.45839139]
b:
[ 0.99699913 -0.02329286]
current function value:
0.002913344340721696
current x value:
p:
[0.44769099 0.55227421]
a:
[-0.26635386  0.45815406]
b:
[ 0.99630795 -0.02360343]
```

```
current function value:
0.004277570154172879
current x value:
p:
[0.44824483 0.55173294]
a:
[-0.26726463  0.45779218]
b:
[ 0.99644262 -0.02379677]
current function value:
0.005190586219703062
current x value:
p:
[0.46071056 0.53927496]
a:
[-0.28622229  0.44272909]
b:
[ 0.99826424 -0.0251125 ]
current function value:
0.0012277474393107258
current x value:
p:
[0.46012933 0.53994969]
a:
[-0.28458368  0.44117217]
b:
[ 0.99822195 -0.02390922]
current function value:
0.000999655593797123
current x value:
p:
[0.45987946 0.54024219]
a:
[-0.28382861  0.44031634]
b:
[ 0.99819604 -0.02330095]
current function value:
0.0011320636617897705
current x value:
p:
[0.45962214 0.54045894]
a:
[-0.28325935  0.43944619]
b:
[ 0.99800774 -0.02274652]
current function value:
0.001113571753743183
current x value:
```

```
p:
[0.45950212 0.54057842]
a:
[-0.2828748   0.43873079]
b:
[ 0.99793516 -0.0223269 ]
current function value:
0.0010290963660938696
current x value:
p:
[0.45946598 0.54062837]
a:
[-0.2826467   0.43818698]
b:
[ 0.99791899 -0.02203406]
current function value:
0.0008629367740329887
current x value:
p:
[0.45953932 0.54064656]
a:
[-0.28243299  0.43767934]
b:
[ 0.99807742 -0.02177146]
current function value:
0.000875048864508976
current x value:
p:
[0.45954609 0.54061575]
a:
[-0.28237269  0.43712212]
b:
[ 0.99799749 -0.02153116]
current function value:
0.0009227411606792261
current x value:
p:
[0.45957124 0.54055146]
a:
[-0.2823955   0.43661759]
b:
[ 0.99789618 -0.02134016]
current function value:
0.0009275008157588542
current x value:
p:
[0.45965349 0.54046417]
a:
```

```
[-0.28244391  0.43614056]
b:
[ 0.99787423 -0.02117409]
current function value:
0.0010187520276010332
current x value:
p:
[0.45973072 0.54035694]
a:
[-0.28255151  0.43569103]
b:
[ 0.9978053  -0.02103731]
current function value:
0.0008764164499726639
current x value:
p:
[0.45988207 0.54026089]
a:
[-0.28260333  0.43532983]
b:
[ 0.99792112 -0.02092553]
current function value:
0.0008316828321758604
current x value:
p:
[0.46005765 0.54013423]
a:
[-0.28270114  0.43491106]
b:
[ 0.99802653 -0.020808  ]
current function value:
0.0009959611047287923
current x value:
p:
[0.46009721 0.53999178]
a:
[-0.2829317   0.43450656]
b:
[ 0.99781288 -0.02072428]
current function value:
0.000790054816899287
current x value:
p:
[0.4604576  0.53987452]
a:
[-0.28289804  0.434149  ]
b:
[ 0.99833466 -0.02060853]
```

```
current function value:
0.0009302062522975747
current x value:
p:
[0.46038371 0.53972123]
a:
[-0.28323807  0.43377484]
b:
[ 0.99785964 -0.02055986]
current function value:
0.0010135530828621778
current x value:
p:
[0.4605104  0.53957155]
a:
[-0.28343544  0.43339847]
b:
[ 0.99781956 -0.02048577]
current function value:
0.0007853700876623726
current x value:
p:
[0.46084675 0.53944891]
a:
[-0.28343867  0.43307265]
b:
[ 0.998282   -0.02039193]
current function value:
0.0010237480807688587
current x value:
p:
[0.46081003 0.53926803]
a:
[-0.2838135   0.43266108]
b:
[ 0.99783085 -0.02034324]
current function value:
0.0008327638661589647
current x value:
p:
[0.46101523 0.53913921]
a:
[-0.28392233  0.43234853]
b:
[ 0.99800223 -0.02027453]
current function value:
0.000821997950705816
current x value:
```

```
p:
[0.46117931 0.53898329]
a:
[-0.2841131   0.43198369]
b:
[ 0.99803066 -0.02020684]
current function value:
0.0007709911825067339
current x value:
p:
[0.46146084 0.53885018]
a:
[-0.28417941  0.43166126]
b:
[ 0.99835687 -0.0201288 ]
current function value:
0.0008615016968262785
current x value:
p:
[0.46143103 0.53869152]
a:
[-0.28451047  0.43131569]
b:
[ 0.99796658 -0.02009303]
current function value:
0.0008188922877998227
current x value:
p:
[0.4615964  0.53855778]
a:
[-0.28465893  0.43100738]
b:
[ 0.99804388 -0.02003454]
current function value:
0.0008352616417658811
current x value:
p:
[0.46173241 0.53840285]
a:
[-0.28486946  0.43065703]
b:
[ 0.99801507 -0.01997593]
current function value:
0.0008371567264641944
current x value:
p:
[0.46188448 0.53824634]
a:
```

```
[-0.28507235  0.43030285]
b:
[ 0.99801732 -0.01991503]
current function value:
0.0010407074930214525
current x value:
p:
[0.46198222 0.53808644]
a:
[-0.28531945  0.42994743]
b:
[ 0.9978968  -0.01986202]
current function value:
0.000935306411987267
current x value:
p:
[0.46213361 0.53795311]
a:
[-0.28547761  0.42964492]
b:
[ 0.99794535 -0.01980742]
current function value:
0.0007810926505804853
current x value:
p:
[0.46236131 0.53782413]
a:
[-0.28557492  0.42934478]
b:
[ 0.99816576 -0.01974248]
current function value:
0.0007954381056078637
current x value:
p:
[0.46248632 0.53767084]
a:
[-0.28579065  0.42900213]
b:
[ 0.99811697 -0.01968739]
current function value:
0.0007350838982358949
current x value:
p:
[0.46294186 0.53752447]
a:
[-0.28576832  0.42866563]
b:
[ 0.99878674 -0.01958377]
```

```
current function value:
0.0007730115896063003
current x value:
p:
[0.46282023 0.53736093]
a:
[-0.28617115  0.42830194]
b:
[ 0.99819173 -0.0195639 ]
current function value:
0.0008691850027673894
current x value:
p:
[0.46289272 0.53720691]
a:
[-0.28642446  0.42796294]
b:
[ 0.99802911 -0.01951668]
current function value:
0.0008507214716036483
current x value:
p:
[0.46304935 0.53705596]
a:
[-0.28661402  0.42762237]
b:
[ 0.99805259 -0.01945739]
current function value:
0.0007526285933661094
current x value:
p:
[0.46328097 0.53692635]
a:
[-0.28671002  0.42732059]
b:
[ 0.99828059 -0.01939181]
current function value:
0.0008838023271913695
current x value:
p:
[0.46333891 0.53675137]
a:
[-0.28701467  0.42693725]
b:
[ 0.9980433  -0.01934159]
current function value:
0.0009302383540898558
current x value:
```

```
p:
[0.46347792 0.53659984]
a:
[-0.2872174   0.42659666]
b:
[ 0.99802791 -0.01928456]
current function value:
0.0007714892218836925
current x value:
p:
[0.46367711 0.536475  ]
a:
[-0.28732627  0.42630775]
b:
[ 0.99819636 -0.01922478]
current function value:
0.0007997331774862376
current x value:
p:
[0.46379917 0.53632323]
a:
[-0.28754108  0.4259678 ]
b:
[ 0.9981441  -0.01917008]
current function value:
0.0008462909753673872
current x value:
p:
[0.46392494 0.53617192]
a:
[-0.28775242  0.42562828]
b:
[ 0.9981007  -0.01911484]
[ 0.88112982  1.31907819  1.18512926  0.83762634  0.4639519   0.53600828
 -0.28805585  0.4252714   0.99782049 -0.01907142]
```

In [58]:
```
ans_final = least_squares(P_min0,ans0,jac=Jac_min0,verbose = 2,ftol = 1e-15)

print('p:')
print(ans_final.x[2*N:3*N])
print('a:')
print(ans_final.x[3*N:4*N])
print('b:')
print(ans_final.x[4*N:5*N])
```

```
   Iteration     Total nfev        Cost      Cost reduction    Step norm     Optimality
       0              1         5.3264e-06                                    9.23e-03
```

```
       1              3         8.9934e-07      4.43e-06       4.26e-02       2.01e-03
       2              5         2.3652e-07      6.63e-07       2.13e-02       4.47e-04
       3              7         1.6364e-07      7.29e-08       1.06e-02       1.20e-04
       4              8         1.5853e-07      5.10e-09       2.13e-02       4.80e-04
       5              9         9.6957e-08      6.16e-08       5.32e-03       3.01e-05
       6             10         7.9128e-08      1.78e-08       1.06e-02       1.20e-04
       7             12         6.6783e-08      1.23e-08       5.32e-03       2.96e-05
       8             13         5.2902e-08      1.39e-08       1.06e-02       1.21e-04
       9             15         4.2550e-08      1.04e-08       5.32e-03       2.99e-05
      10             16         3.2540e-08      1.00e-08       1.06e-02       1.21e-04
      11             18         2.4126e-08      8.41e-09       5.32e-03       3.02e-05
      12             19         1.7889e-08      6.24e-09       1.06e-02       1.22e-04
      13             20         1.1227e-08      6.66e-09       1.06e-02       1.22e-04
      14             21         6.9470e-09      4.28e-09       1.06e-02       1.23e-04
      15             22         5.0006e-09      1.95e-09       1.06e-02       1.23e-04
      16             23         1.8449e-09      3.16e-09       3.77e-03       1.55e-05
      17             24         1.7961e-09      4.88e-11       3.80e-05       1.23e-09
`gtol` termination condition is satisfied.
Function evaluations 24, initial cost 5.3264e-06, final cost 1.7961e-09, first-order optimality
p:
[0.50814067 0.49185935]
a:
[-0.35093961  0.31698432]
b:
[1.00171691 0.00600237]
```

In [107]:
```python
def solve_Cantor(IFS = Cant,x0 = np.round(x,2),size = 10, pre_cond = 2000, max_10 = 

    max_it = 10**max_10

    X = chaos_moments(IFS , it = max_it + 50, burn = 50)

    ans = []

    M = []

    for j in range(1,max_10+1):

        i = 10**j

        print('Data points observed:')
        print(i)

        X_trun = X[:i]


        M_trun = np.zeros(size)
```

```
            for k in range(size):

                M_trun[k] = np.sum(np.power(X_trun,k),axis = 0)/len(X_trun)

            P_mint = lambda x : eval_P(x,M_trun)
            Jac_mint = lambda x : Jac_P(x,M_trun)

            x0_pre = minimize(x0,M_trun,func = eval_P,jac = Jac_P,it = pre_cond,alpha = 1

            print('Preconditioned:')
            print(x0_pre)

            ans_final = least_squares(P_mint,x0_pre,jac=Jac_mint,verbose = 0,ftol = 1e-15

            ans.append(ans_final.x)

            print('Solution:')

            print(ans_final.x[4:])

            M.append(M_trun)

        return ans,M
```

```
In [108]: cant_ans = [1/2,1/2,1/3,1/3,1,0]
          Cant_it, M_it = solve_Cantor()
```

```
Data points observed:
10
Preconditioned:
[1.0946476  1.14583795 0.92630219 0.89854429 0.88069767 0.11929286
 0.16545113 0.23819797 0.01602549 0.61003836]
Solution:
[9.00703430e-01 9.92965736e-02 3.47205515e-01 3.51222288e-05
 6.56309710e-03 6.68694230e-01]
Data points observed:
100
Preconditioned:
[1.17363645 1.26206386 0.88596003 0.853678   0.52080311 0.47918554
 0.27400627 0.37215348 0.01583372 0.63372473]
Solution:
[0.53211083 0.46788918 0.31580296 0.35964743 0.00475922 0.64618282]
Data points observed:
1000
Preconditioned:
[1.17379758 1.24163325 0.88589051 0.8602312  0.47496931 0.52499957
 0.27420605 0.35128651 0.02379852 0.645297  ]
```

```
Solution:
[0.48094725 0.51905276 0.32283649 0.34721955 0.00407328 0.65000095]
Data points observed:
10000
Preconditioned:
[1.15929976 1.2193038  0.89230864 0.86794746 0.50190615 0.49787675
 0.25593075 0.32691679 0.03416432 0.67119509]
Solution:
[ 0.50493738  0.49506262  0.33429318  0.33182207 -0.00115013  0.66910369]
Data points observed:
100000
Preconditioned:
[1.16945789 1.24269358 0.88777283 0.85988729 0.48303449 0.51701448
 0.2688134  0.35257479 0.01902159 0.64806267]
Solution:
[ 4.99812052e-01  5.00187948e-01  3.33651290e-01  3.32887473e-01
 -4.62099763e-04  6.67110821e-01]
Data points observed:
1000000
Preconditioned:
[1.17184739 1.24338254 0.88673258 0.85965988 0.48332528 0.51672843
 0.27179216 0.35330058 0.01777582 0.6474924 ]
Solution:
[ 5.00182138e-01  4.99817861e-01  3.33469987e-01  3.33115700e-01
 -1.55551105e-04  6.66924306e-01]
Data points observed:
10000000
Preconditioned:
[1.16166678 1.22173384 0.89123483 0.86708391 0.49645403 0.5034225
 0.25896558 0.32979558 0.03334382 0.66804453]
Solution:
[ 5.00012630e-01  4.99987369e-01  3.33493786e-01  3.33092758e-01
 -8.48986215e-05  6.66916409e-01]
Data points observed:
100000000
Preconditioned:
[1.16237346 1.22146169 0.8909171  0.86721916 0.49670436 0.50357945
 0.25989329 0.33042535 0.03316675 0.66924395]
Solution:
[ 4.99925408e-01  5.00074591e-01  3.33362282e-01  3.33294316e-01
 -1.73763141e-05  6.66701667e-01]
```

The backwards error bound given in the tex document in shown to be computationally correct, in that the error of our polynomial roots is the same as the error in out moment values.

```
In [113]: plt.figure(figsize=(14,7))
```

```
plt.plot(-np.log10(np.sum(np.abs(M_it-M),axis = 1)), label = 'Input Error')
plt.plot(-np.log10(np.sum(np.abs(np.array(Cant_it)[:,4:] - x[4:]), axis = 1)), label
plt.ylabel('Digits of Accuracy')
plt.xlabel('Power of 10 iterations')
plt.legend()
```

Out[113]: <matplotlib.legend.Legend at 0x21b338ef0f0>