```
using System;

namespace AtomicGroundStateEnergiesGaussian
{
    class LithiumAtom
    {
        private Integration integ;
        private double N;
        private int Z;
        public double integ1, integ2, integ3;
        public double integ4, integ5, integ6;
        public double integ7, integ8, integ9;

        public LithiumAtom()
        {
            integ = new Integration();
        }

        public double Psi(double[] x, double[] alpha)
        {
            double r1 = Math.Sqrt(x[0] * x[0] + x[1] * x[1] + x[2] * x[2]);
            double r2 = Math.Sqrt(x[3] * x[3] + x[4] * x[4] + x[5] * x[5]);
            double r3 = Math.Sqrt(x[6] * x[6] + x[7] * x[7] + x[8] * x[8]);
            double r12 = Math.Sqrt(Math.Pow(x[0] - x[3], 2.0) +
                Math.Pow(x[1] - x[4], 2.0) + Math.Pow(x[2] - x[5], 2.0));
            double r13 = Math.Sqrt(Math.Pow(x[0] - x[6], 2.0) +
                Math.Pow(x[1] - x[7], 2.0) + Math.Pow(x[2] - x[8], 2.0));
            double r23 = Math.Sqrt(Math.Pow(x[3] - x[6], 2.0) +
                Math.Pow(x[4] - x[7], 2.0) + Math.Pow(x[5] - x[8], 2.0));
            double exp1 = Math.Exp(-alpha[0] * r1);
            double exp2 = Math.Exp(-alpha[1] * r2);
            double exp3 = Math.Exp(-alpha[2] * r3);
            double exp4 = Math.Exp(-alpha[3] * r12);
            double exp5 = Math.Exp(-alpha[4] * r13);
            double exp6 = Math.Exp(-alpha[5] * r23);

            return exp1 * exp2 * exp3 * exp4 * exp5 * exp6;
        }

        public double Psi2(double[] x, double[] alpha)
        {
            double psi = Psi(x, alpha);

            return psi * psi;
        }

        public double Normalize(double[] alpha, int nSteps)
        {
            double[] lower = new double[9];
            double[] upper = new double[9];
            int[] steps = new int[9];

```

```
53              lower[0] = 0.001;
54              lower[1] = 0.001;
55              lower[2] = 0.001;
56              lower[3] = 0.001;
57              lower[4] = 0.001;
58              lower[5] = 0.001;
59              lower[6] = 0.001;
60              lower[7] = 0.001;
61              lower[8] = 0.001;
62
63              upper[0] = 10.0;
64              upper[1] = 10.0;
65              upper[2] = 10.0;
66              upper[3] = 10.0;
67              upper[4] = 10.0;
68              upper[5] = 10.0;
69              upper[6] = 10.0;
70              upper[7] = 10.0;
71              upper[8] = 10.0;
72
73              for (int i = 0; i < 9; i++)
74                  steps[i] = nSteps;
75
76              double norm = Math.Sqrt(integ.Integrate(
77                  lower, upper, alpha, Psi2, 9, steps));
78
79              return norm;
80          }
81
82          public double Integrand1(double[] x, double[] alpha)
83          {
84              double r1 = Math.Sqrt(x[0] * x[0] + x[1] * x[1] + x[2] * x[2]);
85              double r2 = Math.Sqrt(x[3] * x[3] + x[4] * x[4] + x[5] * x[5]);
86              double r3 = Math.Sqrt(x[6] * x[6] + x[7] * x[7] + x[8] * x[8]);
87              double r12 = Math.Sqrt(Math.Pow(x[0] - x[3], 2.0) +
88                  Math.Pow(x[1] - x[4], 2.0) + Math.Pow(x[2] - x[5], 2.0));
89              double r13 = Math.Sqrt(Math.Pow(x[0] - x[6], 2.0) +
90                  Math.Pow(x[1] - x[7], 2.0) + Math.Pow(x[2] - x[8], 2.0));
91              double r23 = Math.Sqrt(Math.Pow(x[3] - x[6], 2.0) +
92                  Math.Pow(x[4] - x[7], 2.0) + Math.Pow(x[5] - x[8], 2.0));
93              double term = -2.0 * alpha[0] / r1 + alpha[0] * alpha[0];
94              double mul1 = Math.Exp(-2.0 * alpha[0] * r1);
95              double mul2 = Math.Exp(-2.0 * alpha[1] * r2);
96              double mul3 = Math.Exp(-2.0 * alpha[2] * r3);
97              double mul4 = Math.Exp(-2.0 * alpha[3] * r12);
98              double mul5 = Math.Exp(-2.0 * alpha[4] * r13);
99              double mul6 = Math.Exp(-2.0 * alpha[5] * r23);
100
101             return N * N * term * mul1 * mul2 * mul3 * mul4 * mul5 * mul6;
102         }
103
104         public double Integrand2(double[] x, double[] alpha)
```

```
105         {
106             double r1 = Math.Sqrt(x[0] * x[0] + x[1] * x[1] + x[2] * x[2]);
107             double r2 = Math.Sqrt(x[3] * x[3] + x[4] * x[4] + x[5] * x[5]);
108             double r3 = Math.Sqrt(x[6] * x[6] + x[7] * x[7] + x[8] * x[8]);
109             double r12 = Math.Sqrt(Math.Pow(x[0] - x[3], 2.0) +
110                 Math.Pow(x[1] - x[4], 2.0) + Math.Pow(x[2] - x[5], 2.0));
111             double r13 = Math.Sqrt(Math.Pow(x[0] - x[6], 2.0) +
112                 Math.Pow(x[1] - x[7], 2.0) + Math.Pow(x[2] - x[8], 2.0));
113             double r23 = Math.Sqrt(Math.Pow(x[3] - x[6], 2.0) +
114                 Math.Pow(x[4] - x[7], 2.0) + Math.Pow(x[5] - x[8], 2.0));
115             double term = -2.0 * alpha[1] / r2 + alpha[1] * alpha[1];
116             double mul1 = Math.Exp(-2.0 * alpha[0] * r1);
117             double mul2 = Math.Exp(-2.0 * alpha[1] * r2);
118             double mul3 = Math.Exp(-2.0 * alpha[2] * r3);
119             double mul4 = Math.Exp(-2.0 * alpha[3] * r12);
120             double mul5 = Math.Exp(-2.0 * alpha[4] * r13);
121             double mul6 = Math.Exp(-2.0 * alpha[5] * r23);
122 
123             return N * N * term * mul1 * mul2 * mul3 * mul4 * mul5 * mul6;
124         }
125 
126         public double Integrand3(double[] x, double[] alpha)
127         {
128             double r1 = Math.Sqrt(x[0] * x[0] + x[1] * x[1] + x[2] * x[2]);
129             double r2 = Math.Sqrt(x[3] * x[3] + x[4] * x[4] + x[5] * x[5]);
130             double r3 = Math.Sqrt(x[6] * x[6] + x[7] * x[7] + x[8] * x[8]);
131             double r12 = Math.Sqrt(Math.Pow(x[0] - x[3], 2.0) +
132                 Math.Pow(x[1] - x[4], 2.0) + Math.Pow(x[2] - x[5], 2.0));
133             double r13 = Math.Sqrt(Math.Pow(x[0] - x[6], 2.0) +
134                 Math.Pow(x[1] - x[7], 2.0) + Math.Pow(x[2] - x[8], 2.0));
135             double r23 = Math.Sqrt(Math.Pow(x[3] - x[6], 2.0) +
136                 Math.Pow(x[4] - x[7], 2.0) + Math.Pow(x[5] - x[8], 2.0));
137             double term = -2.0 * alpha[2] / r3 + alpha[2] * alpha[2];
138             double mul1 = Math.Exp(-2.0 * alpha[0] * r1);
139             double mul2 = Math.Exp(-2.0 * alpha[1] * r2);
140             double mul3 = Math.Exp(-2.0 * alpha[2] * r3);
141             double mul4 = Math.Exp(-2.0 * alpha[3] * r12);
142             double mul5 = Math.Exp(-2.0 * alpha[4] * r13);
143             double mul6 = Math.Exp(-2.0 * alpha[5] * r23);
144 
145             return N * N * term * mul1 * mul2 * mul3 * mul4 * mul5 * mul6;
146         }
147 
148         public double Integrand4(double[] x, double[] alpha)
149         {
150             double r1 = Math.Sqrt(x[0] * x[0] + x[1] * x[1] + x[2] * x[2]);
151             double r2 = Math.Sqrt(x[3] * x[3] + x[4] * x[4] + x[5] * x[5]);
152             double r3 = Math.Sqrt(x[6] * x[6] + x[7] * x[7] + x[8] * x[8]);
153             double r12 = Math.Sqrt(Math.Pow(x[0] - x[3], 2.0) +
154                 Math.Pow(x[1] - x[4], 2.0) + Math.Pow(x[2] - x[5], 2.0));
155             double r13 = Math.Sqrt(Math.Pow(x[0] - x[6], 2.0) +
156                 Math.Pow(x[1] - x[7], 2.0) + Math.Pow(x[2] - x[8], 2.0));
```

```
157             double r23 = Math.Sqrt(Math.Pow(x[3] - x[6], 2.0) +
158                 Math.Pow(x[4] - x[7], 2.0) + Math.Pow(x[5] - x[8], 2.0));
159             double term = 1.0 / r1;
160             double mul1 = Math.Exp(-2.0 * alpha[0] * r1);
161             double mul2 = Math.Exp(-2.0 * alpha[1] * r2);
162             double mul3 = Math.Exp(-2.0 * alpha[2] * r3);
163             double mul4 = Math.Exp(-2.0 * alpha[3] * r12);
164             double mul5 = Math.Exp(-2.0 * alpha[4] * r13);
165             double mul6 = Math.Exp(-2.0 * alpha[5] * r23);
166
167             return N * N * Z * term * mul1 * mul2 * mul3 * mul4 * mul5 * mul6;
168         }
169
170         public double Integrand5(double[] x, double[] alpha)
171         {
172             double r1 = Math.Sqrt(x[0] * x[0] + x[1] * x[1] + x[2] * x[2]);
173             double r2 = Math.Sqrt(x[3] * x[3] + x[4] * x[4] + x[5] * x[5]);
174             double r3 = Math.Sqrt(x[6] * x[6] + x[7] * x[7] + x[8] * x[8]);
175             double r12 = Math.Sqrt(Math.Pow(x[0] - x[3], 2.0) +
176                 Math.Pow(x[1] - x[4], 2.0) + Math.Pow(x[2] - x[5], 2.0));
177             double r13 = Math.Sqrt(Math.Pow(x[0] - x[6], 2.0) +
178                 Math.Pow(x[1] - x[7], 2.0) + Math.Pow(x[2] - x[8], 2.0));
179             double r23 = Math.Sqrt(Math.Pow(x[3] - x[6], 2.0) +
180                 Math.Pow(x[4] - x[7], 2.0) + Math.Pow(x[5] - x[8], 2.0));
181             double term = 1.0 / r2;
182             double mul1 = Math.Exp(-2.0 * alpha[0] * r1);
183             double mul2 = Math.Exp(-2.0 * alpha[1] * r2);
184             double mul3 = Math.Exp(-2.0 * alpha[2] * r3);
185             double mul4 = Math.Exp(-2.0 * alpha[3] * r12);
186             double mul5 = Math.Exp(-2.0 * alpha[4] * r13);
187             double mul6 = Math.Exp(-2.0 * alpha[5] * r23);
188
189             return N * N * Z * term * mul1 * mul2 * mul3 * mul4 * mul5 * mul6;
190         }
191
192         public double Integrand6(double[] x, double[] alpha)
193         {
194             double r1 = Math.Sqrt(x[0] * x[0] + x[1] * x[1] + x[2] * x[2]);
195             double r2 = Math.Sqrt(x[3] * x[3] + x[4] * x[4] + x[5] * x[5]);
196             double r3 = Math.Sqrt(x[6] * x[6] + x[7] * x[7] + x[8] * x[8]);
197             double r12 = Math.Sqrt(Math.Pow(x[0] - x[3], 2.0) +
198                 Math.Pow(x[1] - x[4], 2.0) + Math.Pow(x[2] - x[5], 2.0));
199             double r13 = Math.Sqrt(Math.Pow(x[0] - x[6], 2.0) +
200                 Math.Pow(x[1] - x[7], 2.0) + Math.Pow(x[2] - x[8], 2.0));
201             double r23 = Math.Sqrt(Math.Pow(x[3] - x[6], 2.0) +
202                 Math.Pow(x[4] - x[7], 2.0) + Math.Pow(x[5] - x[8], 2.0));
203             double term = 1.0 / r3;
204             double mul1 = Math.Exp(-2.0 * alpha[0] * r1);
205             double mul2 = Math.Exp(-2.0 * alpha[1] * r2);
206             double mul3 = Math.Exp(-2.0 * alpha[2] * r3);
207             double mul4 = Math.Exp(-2.0 * alpha[3] * r12);
208             double mul5 = Math.Exp(-2.0 * alpha[4] * r13);
```

```
209             double mul6 = Math.Exp(-2.0 * alpha[5] * r23);
210
211             return N * N * Z * term * mul1 * mul2 * mul3 * mul4 * mul5 * mul6;
212         }
213
214         public double Integrand7(double[] x, double[] alpha)
215         {
216             double r1 = Math.Sqrt(x[0] * x[0] + x[1] * x[1] + x[2] * x[2]);
217             double r2 = Math.Sqrt(x[3] * x[3] + x[4] * x[4] + x[5] * x[5]);
218             double r12 = Math.Sqrt(Math.Pow(x[0] - x[3], 2.0) +
219                 Math.Pow(x[1] - x[4], 2.0) + Math.Pow(x[2] - x[5], 2.0));
220
221             if (r12 == 0)
222                 r12 = 0.01;
223
224             double term = 1.0 / r12;
225             double mul1 = Math.Exp(-2.0 * alpha[0] * r1);
226             double mul2 = Math.Exp(-2.0 * alpha[1] * r2);
227             double mul3 = Math.Exp(-2.0 * alpha[3] * r12);
228
229             return N * N * term * mul1 * mul2 * mul3;
230         }
231
232         public double Integrand8(double[] x, double[] alpha)
233         {
234             double r1 = Math.Sqrt(x[0] * x[0] + x[1] * x[1] + x[2] * x[2]);
235             double r3 = Math.Sqrt(x[6] * x[6] + x[7] * x[7] + x[8] * x[8]);
236             double r13 = Math.Sqrt(Math.Pow(x[0] - x[6], 2.0) +
237                 Math.Pow(x[1] - x[7], 2.0) + Math.Pow(x[2] - x[8], 2.0));
238
239             if (r13 == 0)
240                 r13 = 0.01;
241
242             double term = 1.0 / r13;
243             double mul1 = Math.Exp(-2.0 * alpha[0] * r1);
244             double mul2 = Math.Exp(-2.0 * alpha[1] * r3);
245             double mul3 = Math.Exp(-2.0 * alpha[4] * r13);
246
247             return N * N * term * mul1 * mul2 * mul3;
248         }
249
250         public double Integrand9(double[] x, double[] alpha)
251         {
252             double r2 = Math.Sqrt(x[3] * x[3] + x[4] * x[4] + x[5] * x[5]);
253             double r3 = Math.Sqrt(x[6] * x[6] + x[7] * x[7] + x[8] * x[8]);
254             double r23 = Math.Sqrt(Math.Pow(x[3] - x[6], 2.0) +
255                 Math.Pow(x[4] - x[7], 2.0) + Math.Pow(x[5] - x[8], 2.0));
256
257             if (r23 == 0)
258                 r23 = 0.01;
259
260             double term = 1.0 / r23;
```

```
261             double mul1 = Math.Exp(-2.0 * alpha[0] * r2);
262             double mul2 = Math.Exp(-2.0 * alpha[2] * r3);
263             double mul3 = Math.Exp(-2.0 * alpha[5] * r23);
264
265             return N * N * term * mul1 * mul2 * mul3;
266         }
267
268         public double Energy(double[] alpha, double beta, int nSteps, int Z)
269         {
270             double[] lower = new double[9];
271             double[] upper = new double[9];
272             int[] steps = new int[9];
273
274             lower[0] = lower[1] = lower[2] = lower[3] = lower[4] = lower[5] =
                0.001;
275             lower[6] = lower[7] = lower[8] = 0.001;
276             upper[0] = upper[1] = upper[2] = upper[3] = upper[4] = upper[5] =
                10.0;
277             upper[6] = upper[7] = upper[8] = 10.0;
278             steps[0] = steps[1] = steps[2] = steps[3] = steps[4] = steps[5] =
                nSteps;
279             steps[6] = steps[7] = steps[8] = nSteps;
280
281             N = Normalize(alpha, nSteps);
282
283             this.Z = Z;
284
285             integ1 = +integ.Integrate(lower, upper, alpha, Integrand1, 9, steps);
286             integ2 = +integ.Integrate(lower, upper, alpha, Integrand2, 9, steps);
287             integ3 = +integ.Integrate(lower, upper, alpha, Integrand3, 9, steps);
288             integ4 = -integ.Integrate(lower, upper, alpha, Integrand4, 9, steps);
289             integ5 = -integ.Integrate(lower, upper, alpha, Integrand5, 9, steps);
290             integ6 = -integ.Integrate(lower, upper, alpha, Integrand6, 9, steps);
291             integ7 = +integ.Integrate(lower, upper, alpha, Integrand7, 9, steps);
292             integ8 = +integ.Integrate(lower, upper, alpha, Integrand8, 9, steps);
293             integ9 = +integ.Integrate(lower, upper, alpha, Integrand9, 9, steps);
294
295             double integrals = integ1 + integ2 + integ3 + integ4 + integ5 +
                integ6 +
296                 beta * (integ7 + integ8 + integ9) / (N * N);
297
298             return integrals;
299         }
300     }
301 }
```