```
using System;

namespace AtomicGroundStateEnergiesGaussian
{
    class HeliumAtom
    {
        private Integration integ;
        private double N;
        private int Z;
        public double integ1, integ2, integ3;
        public double integ4, integ5;

        public HeliumAtom()
        {
            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 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 exp1 = Math.Exp(-alpha[0] * r1);
            double exp2 = Math.Exp(-alpha[1] * r2);
            double exp3 = Math.Exp(-alpha[2] * r12);

            return exp1 * exp2 * exp3;
        }

        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[6];
            double[] upper = new double[6];
            int[] steps = new int[6];

            lower[0] = 0.001;
            lower[1] = 0.001;
            lower[2] = 0.001;
            lower[3] = 0.001;
            lower[4] = 0.001;
            lower[5] = 0.001;

            upper[0] = 10.0;
            upper[1] = 10.0;
```

```
53              upper[2] = 10.0;
54              upper[3] = 10.0;
55              upper[4] = 10.0;
56              upper[5] = 10.0;
57
58              for (int i = 0; i < 6; i++)
59                  steps[i] = nSteps;
60
61              double norm = Math.Sqrt(integ.Integrate(
62                  lower, upper, alpha, Psi2, 6, steps));
63
64              return norm;
65          }
66
67          public double Integrand1(double[] x, double[] alpha)
68          {
69              double r1 = Math.Sqrt(x[0] * x[0] + x[1] * x[1] + x[2] * x[2]);
70              double r2 = Math.Sqrt(x[3] * x[3] + x[4] * x[4] + x[5] * x[5]);
71              double r12 = Math.Sqrt(Math.Pow(x[0] - x[3], 2.0) +
72                  Math.Pow(x[1] - x[4], 2.0) + Math.Pow(x[2] - x[5], 2.0));
73              double term = -2.0 * alpha[0] / r1 + alpha[0] * alpha[0];
74              double mul1 = Math.Exp(-2.0 * alpha[0] * r1);
75              double mul2 = Math.Exp(-2.0 * alpha[1] * r2);
76              double mul3 = Math.Exp(-2.0 * alpha[2] * r12);
77
78              return N * N * term * mul1 * mul2 * mul3;
79          }
80
81          public double Integrand2(double[] x, double[] alpha)
82          {
83              double r1 = Math.Sqrt(x[0] * x[0] + x[1] * x[1] + x[2] * x[2]);
84              double r2 = Math.Sqrt(x[3] * x[3] + x[4] * x[4] + x[5] * x[5]);
85              double r12 = Math.Sqrt(Math.Pow(x[0] - x[3], 2.0) +
86                  Math.Pow(x[1] - x[4], 2.0) + Math.Pow(x[2] - x[5], 2.0));
87              double term = -2.0 * alpha[1] / r1 + alpha[1] * alpha[1];
88              double mul1 = Math.Exp(-2.0 * alpha[0] * r1);
89              double mul2 = Math.Exp(-2.0 * alpha[1] * r2);
90              double mul3 = Math.Exp(-2.0 * alpha[2] * r12);
91
92              return N * N * term * mul1 * mul2 * mul3;
93          }
94
95          public double Integrand3(double[] x, double[] alpha)
96          {
97              double r1 = Math.Sqrt(x[0] * x[0] + x[1] * x[1] + x[2] * x[2]);
98              double r2 = Math.Sqrt(x[3] * x[3] + x[4] * x[4] + x[5] * x[5]);
99              double r12 = Math.Sqrt(Math.Pow(x[0] - x[3], 2.0) +
100                 Math.Pow(x[1] - x[4], 2.0) + Math.Pow(x[2] - x[5], 2.0));
101             double term = 1.0 / r1;
102             double mul1 = Math.Exp(-2.0 * alpha[0] * r1);
103             double mul2 = Math.Exp(-2.0 * alpha[1] * r2);
104             double mul3 = Math.Exp(-2.0 * alpha[2] * r12);
```

```
105
106             return N * N * Z * term * mul1 * mul2 * mul3;
107         }
108
109         public double Integrand4(double[] x, double[] alpha)
110         {
111             double r1 = Math.Sqrt(x[0] * x[0] + x[1] * x[1] + x[2] * x[2]);
112             double r2 = Math.Sqrt(x[3] * x[3] + x[4] * x[4] + x[5] * x[5]);
113             double r12 = Math.Sqrt(Math.Pow(x[0] - x[3], 2.0) +
114                 Math.Pow(x[1] - x[4], 2.0) + Math.Pow(x[2] - x[5], 2.0));
115             double term = 1.0 / r2;
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] * r12);
119
120             return N * N * Z * term * mul1 * mul2 * mul3;
121         }
122
123         public double Integrand5(double[] x, double[] alpha)
124         {
125             double r1 = Math.Sqrt(x[0] * x[0] + x[1] * x[1] + x[2] * x[2]);
126             double r2 = Math.Sqrt(x[3] * x[3] + x[4] * x[4] + x[5] * x[5]);
127             double r12 = Math.Sqrt(Math.Pow(x[0] - x[3], 2.0) +
128                 Math.Pow(x[1] - x[4], 2.0) + Math.Pow(x[2] - x[5], 2.0));
129
130             if (r12 == 0)
131                 r12 = 0.01;
132
133             double term = 1.0 / r12;
134             double mul1 = Math.Exp(-2.0 * alpha[0] * r1);
135             double mul2 = Math.Exp(-2.0 * alpha[1] * r2);
136             double mul3 = Math.Exp(-2.0 * alpha[2] * r12);
137
138             return N * N * term * mul1 * mul2 * mul3;
139         }
140
141         public double Energy(double[] alpha, double beta, int nSteps, int Z)
142         {
143             double[] lower = new double[6];
144             double[] upper = new double[6];
145             int[] steps = new int[6];
146
147             lower[0] = lower[1] = lower[2] = lower[3] = lower[4] = lower[5] =
                  0.001;
148             upper[0] = upper[1] = upper[2] = upper[3] = upper[4] = upper[5] =
                  10.0;
149             steps[0] = steps[1] = steps[2] = steps[3] = steps[4] = steps[5] =
                  nSteps;
150
151             N = Math.Sqrt(integ.Integrate(
152                 lower, upper, alpha, Psi2, 6, steps));
153
```

```
154             this.Z = Z;
155 
156             integ1 = +integ.Integrate(lower, upper, alpha, Integrand1, 6, steps);
157             integ2 = +integ.Integrate(lower, upper, alpha, Integrand2, 6, steps);
158             integ3 = -integ.Integrate(lower, upper, alpha, Integrand3, 6, steps);
159             integ4 = -integ.Integrate(lower, upper, alpha, Integrand4, 6, steps);
160             integ5 = +integ.Integrate(lower, upper, alpha, Integrand5, 6, steps);
161 
162             return (integ1 + integ2 + integ3 + integ4 + beta * integ5) / (N * N);
163         }
164     }
165 }
```