The Effect of Nuclear Waste on the Environment

New Mexico

Super Computing Challenge

Final Report

April 1, 2009

Team # 67

Manzano High School

Team Members:

Steven Benner – Senior

Scott Wilson – Senior

Platon Krasin- Junior

Teacher :

Stephen Schum

Executive summary

The United States is using nuclear energy as a source of power. As a result of the use of this energy, nuclear waste is constantly being created and is being transported to locations in which it can be properly contained. Our team's problem is to model how the most commonly used radioactive materials and their particles will spread if not properly contained.

The elements and isotopes that our model must contain are not stable and are constantly changing according to their own half-life. As a team we have dedicated a large part of our C++ code in order to find the resulting masses of well-known, unstable, isotopes after a certain amount of time. We have also used our code to find the amount of alpha and beta partials that are released during the specific amount of time. As the particle goes through its half-life it also creates a daughter product that has its own half-life. The isotope will continue to half-life until it eventually becomes a stable element. In our code we have included Uranium 235, 238, Plutonium 239 and their daughter products as our main elements of study because they are the isotopes that are most commonly used in nuclear reactors.

We hope to conclude our project by using the results from our C++ code in order to create a model that can show the spread of the radioactive particles, as well as the decay of a specific radioactive isotope.

Purpose and Problem

How nuclear power plants choose store radioactive waste has become a major issue that is facing the nation. In America we have been using nuclear power plants as a source of energy. The problem that has occurred as a result of our use of nuclear power plants is that a large amount of radioactive waste is being created and needs to be disposed. The nuclear materials that are being used are able to create health hazards for the surrounding people and agriculture. Because of these hazards, there are many concerns about the effect of nuclear waste on our environment. The main concern that we wish to address is the spread of the nuclear contaminants over time.

The scope of our problem involves the United States as a whole; however the potential problem of radiation spreading from nuclear waste is a major concern for New Mexico. The concern in New Mexico is raised because nuclear waste is transported through it, Uranium is a natural resource in New Mexico, and there is potential for a Uranium enrichment plant to be placed in New Mexico.

Information

All radioactive isotopes go through a specific half-life and emit alpha and beta particles that are dangerous if exposed to living organisms. After the element goes through a half-life, it is transformed into a different isotope (called a daughter product) until it becomes stable. Each isotope that our problem is dealing with has a predictable chain of daughter products that is called a decay series. At certain points in the decay series, the isotope may either beta decay or alpha decay into a different element. However, there is an estimated ratio for whether the isotopes will Beta or Alpha decay.

Computer Program in C++

In order to eventually achieve our goal of creating a model of the spread of radioactive material, we have created a code in C++ that calculates the resulting mass of an isotope and the isotopes in its decay series after going through a half-life or a specified amount of time. Our code includes the decay series (or daughter products) of the isotopes because the isotope will decay into different radioactive elements that have different half-life, until the isotope becomes stable. The code that we created in C++ also calculates the amount of radioactive particles (Alpha and Beta) that are released during an amount time that is specified by an outside user.

We solved our problem by using an adaptation of the half-life formula (resulting mass = initial amount / 2^(N), where 'N' = elapsed time / half-life) on the initial isotope and the isotope in the decay series. The program requires the user to input how much time will pass (in days) and which radioactive isotope(s) will be used. In order to allow the input of a variety of isotopes and elements, we created the program to allow the user to enter in the atomic number and the isotope of the substances that are commonly used for nuclear power (eg. Uranium 238, Uranium 235, Plutonium ect…). We accomplished this task by creating a library of the main elements, isotopes and half-life that can be called upon by the user. We have also created our code to allow for two different isotopes to be entered and calculated in order to save time if there is more than one isotope present.

The resulting mass of the isotope is found and is converted into the number of atoms that decayed. The number of alpha and/ or beta partials that were created and spread during the process is then found by using the number of atoms that decayed. The number of atoms that have decayed from the initial isotope and its daughter products determines how many beta and alpha particles are formed do to each atom's transformation/decay into the next isotope or element in the decay series.

Eventually, the initial mass, isotope, resulting mass, decay series, time passed and the amount of Alpha and Beta particles will be included into a model that will be based in NetLogo.

Main Half –Life Formula

The main half-life formula that is with the initial isotope, and, if required, the next isotope in the initial isotope's decay series:

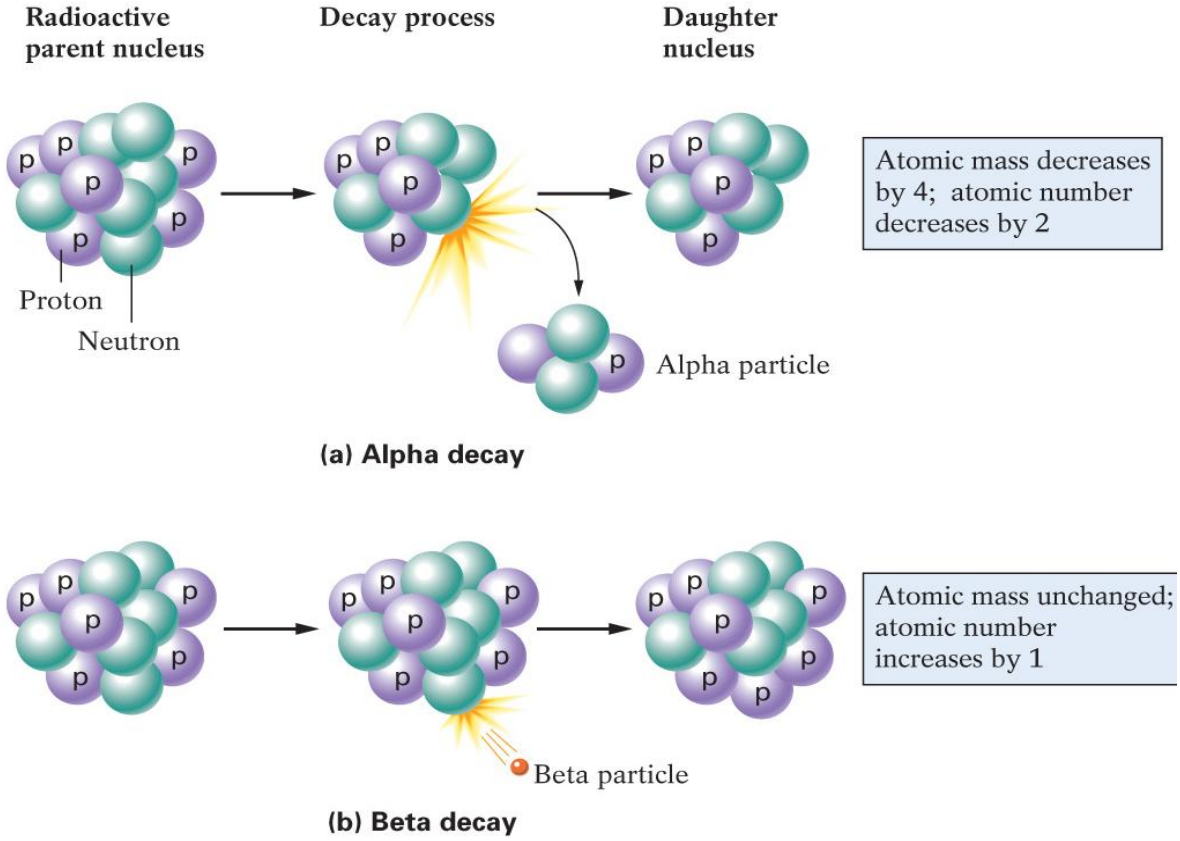
rmass[e][iso] = mass[e][iso]*pow(0.5,(t/(hl[e][iso][0]*pow(10,hl[e][iso][1]))))

Where:

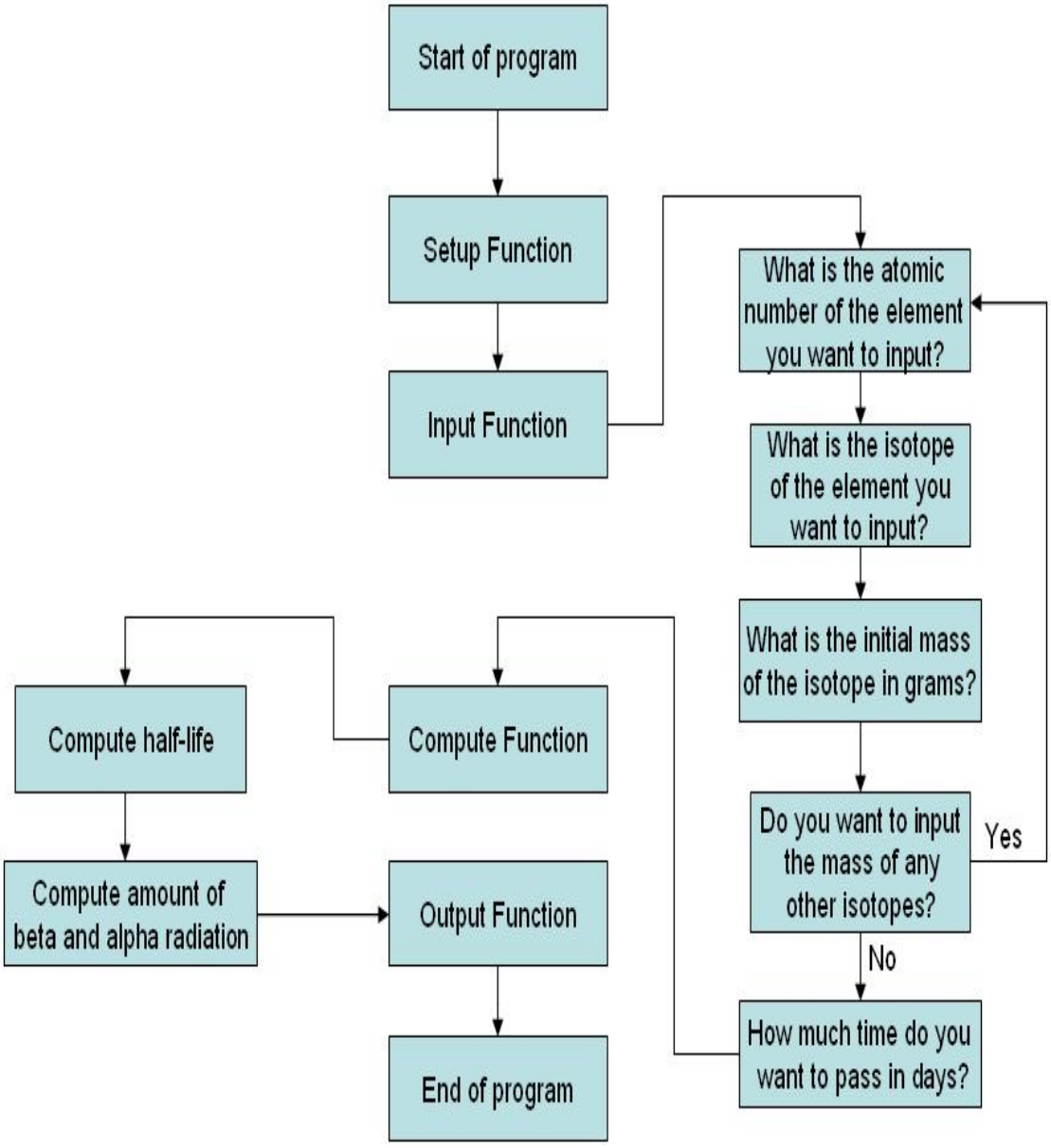
rmass = resulting mass (in grams) , mass = initial mass, t = time (in days),

e = element, iso = isotope, hl = half-life

Diagram of Alpha and Beta Decay



C++ Code Flowchart:



Example Tables:

Uranium 238 Decay Series Table

Element	Half life
Uranium 238- 92	4.5 billion
Thorium 234- 90	24.1 days
Protactinium 234- 91	1 minute
Uranium 234- 92	245, 000 years
Thorium 230- 90	76,000 years
Radium 226- 88	1,600 years
Radon 222- 86	3.8 days
Polonium 218- 84	3.0 minutes
Lead 214- 82	27 minutes
Bismuth 214- 83	20 minutes
Polonium 214- 84 or Thallium 210- 81	130 microseconds
Lead 210- 82	22 years
Bismuth 210- 83	5 days
Polonium 210- 84	138

(Blue text = initial isotope)

Uranium 235 Decay Series Table

Element	Half life
Uranium 235- 92	704 million
Thorium 231- 90	25.5 hours
Protactinium 231-91	32500 years
Actinium 227- 89	21.6 years
Thorium 227- 90 or Francium 223- 87	18.7 days or 22 minutes
Radium 223- 88	11.4 days
Radon 219- 86	3.92-second
Polonium 215- 84	138.376 days
Lead 211- 82	36.1 minutes
Bismuth 211- 83	2.16 min
Polonium 211-84 or Thallium 207- 81	0.52 second or 4.8 min
Lead 207-82	Stable

(Blue text = initial isotope)

Results

There are many possible results from our code, so we created an example by using the results of running our program with the isotope U-238 with an initial mass of 1000 grams and a time period of 100 days.

element	isotope	resulting mass
92	238	1000
90	234	2.37648e-09
91	234	0
92	234	3.97953e-08
90	230	3.08248e-14
88	226	7.6961e-20
86	222	0
84	218	0
82	214	0
83	214	0
84	214	0
81	210	0
82	210	9.05035e-24
83	210	0
84	210	4.74476e-26

82	206	0
94	239	0
92	235	0
90	231	0
91	231	0
89	227	0
90	227	0
87	223	0
88	223	0
86	219	0
84	215	0
82	211	0
83	211	0
84	211	0
81	207	0
82	207	0

Alpha decay

1.06705e+14 particles

Beta decay

2.04827e+14 particles

Conclusion

By analyzing the results from our C++ simulation of how an isotope decays over time, it can be concluded that uncontained nuclear waste, even in a short period of time, will emit a large amount Alpha and Beta particles. It can be seen that because of the large quantity of particles, nuclear waste that is improperly contained has the potential to cause a great amount of harm to the surrounding environment and people if the Alpha and Beta particles spread.

What we will accomplish in the future

We will actually create the visual model of the spread of the radioactive particles in NetLogo. We will incorporate the results that we have gained from our C++ code in order to accomplish a visual representation of the creation and spread of the alpha and beta partials. We will also include a representation of the resulting mass from the decaying initial isotope as well as the new isotopes that are created through the decay chain.

References

Elliott, T. G., Haase, Kathleen A. Harper, Herzog, Nelson, Friedrich Schiller, Paul W. Zitzewitz, and Zorn. Physics: Principles and Problems. Columbus, OH: Mcgraw-Hill/Glencoe, 2004.

"NRC: Radioactive Waste." NRC: Home Page. 19 Dec. 2008 <http://www.nrc.gov/waste.html>.

"Radioactive waste - Wikipedia, the free encyclopedia." Wikipedia, the free encyclopedia. 19 Dec. 2008 <http://en.wikipedia.org/wiki/Nuclear_waste>.

"Yucca Mountian Update." www.state.nv.us. 20 Feb. 2009 <www.state.nv.us/nucwaste/yucca/ymupdate/WIPP.jpg>.

"What is Nuclear Waste?." wiseGEEK: clear answers for common questions. 19 Dec. 2008 <http://www.wisegeek.com/what-is-nuclear-waste.htm>.

Wilbraham, Antony C.. Chemistry. New York City: Addison-Wesley Pub (Sd), 1987.

"NRC: Radioactive Waste." NRC: Home Page. 19 Dec. 2008 <http://www.nrc.gov/waste.html>.

Code in C++

```
#include <iostream>
#include <cmath>

using namespace std;

class material
{
	public:
	double mass[95][239];
	double rmass[95][239];
	double dmass[95][239];
	double hl[95][239][2];
	int dseq[31][2];
	double adecay;
	double bdecay;
	double t;
	void setup(void);
	void input(void);
	void compute(void);
	void output(void);
	void hllib(void);
	void dseqset(void);
};
int main()
{
	material m1;
	m1.t = 0;
	m1.setup();
```

```
        m1.input();
        m1.hllib();
        m1.dseqset();
        m1.compute();
        m1.output();
        system("PAUSE");
        return 0;
}

void material::setup()
{
    adecay = 0;
    bdecay = 0;
    for(int i=0;i<=94;i++)
    {
          for(int x=0;x<=238;x++)
          {
          mass[i][x]= 0;
          rmass[i][x]= 0;
          dmass[i][x]= 0;
          }
    }
}

void material::input()
{
        int element = 0;
        int iso = 0;
        int d;
        int x = 1;
        while(x == 1)
```

```
        {
          cout << "What is the element?" << endl;
          cin >> element;
          cout << "What isotope of the material?" << endl;
          cin >> iso;
          cout << "What is the mass of the material?" << endl;
          cin >> mass[element][iso];
          cout << "Do you want to input the mass of any"
              <<          " other elements?" << endl;
          cin >> d;
          if(d == 0)
          {
                  x = 0;
          }
        }
        cout << "\nHow much time do you want to pass?" << endl;
        cin >> t;
        if(t==0)
        {
                cout << "\nError: improper input." << endl;
        }
}

void material::compute()
{
        int e;
        int iso;
        for (int y = 0; y<=30; y++)
        {
                e = dseq[y][0];
                iso = dseq[y][1];
```

```
                    rmass[e][iso] =
mass[e][iso]*pow(0.5,(t/(hl[e][iso][0]*pow(10,hl[e][iso][1]))));
                    dmass[e][iso] = mass[e][iso] - rmass[e][iso];

            if (y == 1 || y == 2 || y == 8 || y == 11 || y == 12 || y == 13 ||
                y == 18 || y == 22 || y == 26 || y == 29)
            {
                bdecay += dmass[e][iso] / iso * (6.022*pow(10.0,23));
            }
            else
            {
                adecay += dmass[e][iso] / iso * (6.022*pow(10.0,23));
            }

                    if (y == 9)
                    {
                    mass[dseq[y+1][0]][dseq[y+1][1]] += dmass[e][iso]*0.9998;
                    mass[dseq[y+2][0]][dseq[y+2][1]] += dmass[e][iso]*0.0002;
            }
                    else if (y == 20)
                    {
                    mass[dseq[y+1][0]][dseq[y+1][1]] += dmass[e][iso]*0.986;
                    mass[dseq[y+2][0]][dseq[y+2][1]] += dmass[e][iso]*0.014;
            }
                    else if (y == 27)
                    {
                    mass[dseq[y+1][0]][dseq[y+1][1]] += dmass[e][iso]*0.0028;
                    mass[dseq[y+2][0]][dseq[y+2][1]] += dmass[e][iso]*0.9972;
            }
              else if (y == 21 || y == 28 || y == 10)
              {
```

```
                    mass[dseq[y+2][0]][dseq[y+2][1]] += dmass[e][iso];
                }
                else if (y == 30 || y == 15)
                {}
                    else
                    {
                mass[dseq[y+1][0]][dseq[y+1][1]] += dmass[e][iso];
                }
                }
        }

        void material::output()
        {
            cout << "element" << "\t" << "isotope" << "\t" << "resulting mass" <<
endl;
            for(int y = 0; y<=30; y++)
            {
                    cout << dseq[y][0] << "\t" << dseq[y][1] << "\t"
                << rmass[dseq[y][0]][dseq[y][1]] << endl;
            }
            cout << "alpha decay" << endl << adecay << endl << endl;
            cout << "beta decay" << endl << bdecay << endl;
        }

        void material::hllib(void)
        {
            hl[92][238][0] = 1.643625;
            hl[92][238][1] = 12;
            hl[90][234][0] = 2.41;
            hl[90][234][1] = 1;
            hl[91][234][0] = 6.944444;
```

```
hl[91][234][1] = -4;
hl[92][234][0] = 8.948625;
hl[92][234][1] = 7;
hl[90][230][0] = 2.7759;
hl[90][230][1] = 7;
hl[88][226][0] = 5.844;
hl[88][226][1] = 5;
hl[86][222][0] = 2.638889;
hl[86][222][1] = -3;
hl[84][218][0] = 2.083333;
hl[84][218][1] = -3;
hl[82][214][0] = 1.875;
hl[82][214][1] = -2;
hl[83][214][0] = 1.388888;
hl[83][214][1] = -2;
hl[84][214][0] = 1.504629;
hl[84][214][1] = -9;
hl[82][210][0] = 8.0355;
hl[82][210][1] = 3;
hl[83][210][0] = 3.472222;
hl[83][210][1] = -3;
hl[84][210][0] = 1.38;
hl[84][210][1] = 2;
hl[82][206][0] = 0;
hl[82][206][1] = 0;

hl[94][239][0] = 2.411;
hl[94][239][1] = 4;
hl[92][235][0] = 2.57136;
hl[92][235][1] = 11;
hl[90][231][0] = 1.0625;
```

```
    hl[90][231][1] = 0;
    hl[91][231][0] = 1.1870625;
    hl[91][231][1] = 7;
    hl[89][227][0] = 7.8894;
    hl[89][227][1] = 3;
    hl[90][227][0] = 18.7;
    hl[90][227][1] = 1;
    hl[87][223][0] = 1.527777;
    hl[87][223][1] = -2;
    hl[88][223][0] = 11.4;
    hl[88][223][1] = 1;
    hl[86][219][0] = 4.537037;
    hl[86][219][1] = -5;
    hl[84][215][0] = 1.38376;
    hl[84][215][1] = 2;
    hl[82][211][0] = 2.506944;
    hl[82][211][1] = -2;
    hl[83][211][0] = 15;
    hl[83][211][1] = -3;
    hl[84][211][0] = 6.018518;
    hl[84][211][1] = -6;
    hl[81][207][0] = 3.333333;
    hl[81][207][1] = -3;
    hl[82][207][0] = 0;
    hl[82][207][1] = 0;
}

void material::dseqset(void)
{
        dseq[0][0] = 92;
        dseq[0][1] = 238;
```

```
dseq[1][0] = 90;
dseq[1][1] = 234;
dseq[2][0] = 91;
dseq[2][1] = 234;
dseq[3][0] = 92;
dseq[3][1] = 234;
dseq[4][0] = 90;
dseq[4][1] = 230;
dseq[5][0] = 88;
dseq[5][1] = 226;
dseq[6][0] = 86;
dseq[6][1] = 222;
dseq[7][0] = 84;
dseq[7][1] = 218;
dseq[8][0] = 82;
dseq[8][1] = 214;
dseq[9][0] = 83;
dseq[9][1] = 214;
dseq[10][0] = 84;
dseq[10][1] = 214;
dseq[11][0] = 81;
dseq[11][1] = 210;
dseq[12][0] = 82;
dseq[12][1] = 210;
dseq[13][0] = 83;
dseq[13][1] = 210;
dseq[14][0] = 84;
dseq[14][1] = 210;
dseq[15][0] = 82;
dseq[15][1] = 206;
```

```
        dseq[16][0] = 94;
        dseq[16][1] = 239;
        dseq[17][0] = 92;
        dseq[17][1] = 235;
        dseq[18][0] = 90;
        dseq[18][1] = 231;
        dseq[19][0] = 91;
        dseq[19][1] = 231;
        dseq[20][0] = 89;
        dseq[20][1] = 227;
        dseq[21][0] = 90;
        dseq[21][1] = 227;
        dseq[22][0] = 87;
        dseq[22][1] = 223;
        dseq[23][0] = 88;
        dseq[23][1] = 223;
        dseq[24][0] = 86;
        dseq[24][1] = 219;
        dseq[25][0] = 84;
        dseq[25][1] = 215;
        dseq[26][0] = 82;
        dseq[26][1] = 211;
        dseq[27][0] = 83;
        dseq[27][1] = 211;
        dseq[28][0] = 84;
        dseq[28][1] = 211;
        dseq[29][0] = 81;
        dseq[29][1] = 207;
        dseq[30][0] = 82;
        dseq[30][1] = 207;
}
```