astrodynamicss: Let's go to moon

Author: 申汀

Data：2016.1.6

Abstract:

In this paper, I did some astrodynamicss calculations to let a craft to be able to go to moon. The main work is how to find a route. For simplicity, I have to make some approximations and I suppose all this happened in the future. Finally, I make it and account for the time I spend. But this plan is not perfect, so many improvements are needed. After all, it's my first step trying to go into the space.

Keywords:

astrodynamics, route prediction, earth to moon, space traveling

Main Body:

Since 19th century, the population on earth is gradually getting crowded, which means we will use up some kinds of resources. So our prospect is no doubtfully space. But until now, we have still just been moon, and I have to say the procedure is really slow. Searching and studying space is the inclination of the future, and it's my habit coincidentally. So I choose this topic: astrodynamicss.

I was inspired when I saw the movie *Bring Me Home* some days ago. In the movie there was an Indian who did the astrodynamicss for NASA and successfully saved an astronaut. He calculated and

changed the route of the craft. So I think I could do the same thing.

But I need to start from the very beginning: how to go to moon. At first I thought it might be not very difficult, but eventually I found I was totally wrong. Even though I make many approximations, it's still not easy. And I'm very satisfied after I finished it.

1.How to choose a route

The first thing I need to consider is how to choose a route. This problem has already been solved by scientists so I want to find some hint. I've seen some pictures on television of Chang'E1, which showed the route of it. It was a curve. But I wonder why it was in this way? Why wasn't it just go to the moon? I suddenly realized it was because the inertia and for saving energy. If we want to travel in space, energy is unique, so saving energy is so crucial that we must consider about it.

Maybe the best choice is to rotating around earth first and then accelerate at same point for several times to go through the critical point. By this way could we do the least work. So I choose it. But meanwhile, I need to consider the rotation of moon. I need to know how many days it will take to go to the critical point so that I won't miss moon.

After arriving the critical point, then I need to change my velocity again cause moon is rotating but the craft is now relatively

“static” over earth. I made it has an additional tangential velocity according to its distance to earth and the angular velocity of moon in honor of the line which links earth and moon.

Getting Approached to moon, I make it orbit around moon. So now we could go to any place on it.

Fortunately, I finally verified this hypothesis.

2.Backgrounds: settings and approximations

Before I started, I made some settings of the situation. I assumed this journey happened in the future when we don’t need rockets but just crafts and the crafts wouldn’t need fuel but instead the nuclear power, which made its mass won’t change in the space. This is vital because if I need to calculate the change of mass, this problem would be more too difficult. But it’s also reasonable, since human have already used the nuclear power to get electricity and we could image it will be utilized in other ways before too long. So I set the engine power P.

Even though I had to make many approximations. I assumed the time of acceleration was infinitesimal over the time the journey takes. I assumed the orbit of moon is a circle (but we all now it’s obviously not) and I choose the average radius of the orbit.

3.Search for some data[1]

For calculation, I need some data:

The masses and radiuses of earth and moon.

The periodic time of moon.

The radius of moon's orbit.

The thickness of air on earth. (for orbiting)

4.Numerical calculation

The main algorithm I used is Runge-Kutta method. I use this method to predict the route of the craft. I know its initial speed, location and also those of moon's. This procedure is not very difficult but complex and need some patience.

Besides, I calculated the rough distance of critical point, where the gravitation of earth and moon is equilibrium. To go to moon, we should at least arrive at this point.

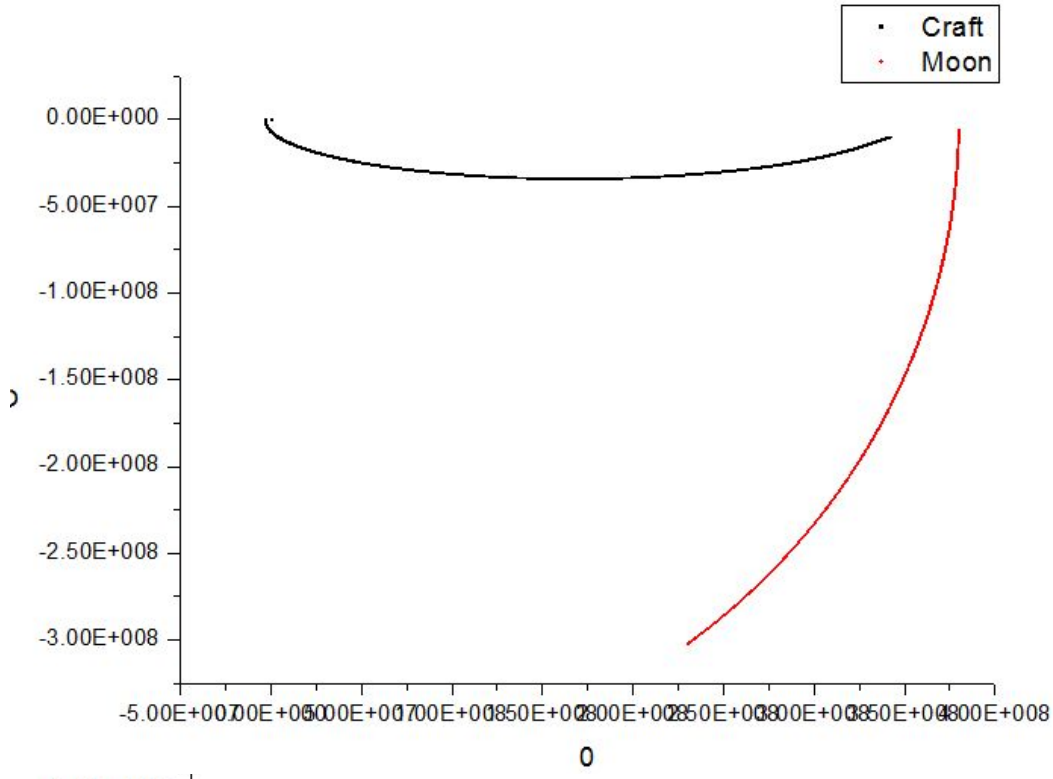
And at the end of the codes, I calculated the actual time this traveling totally costs.

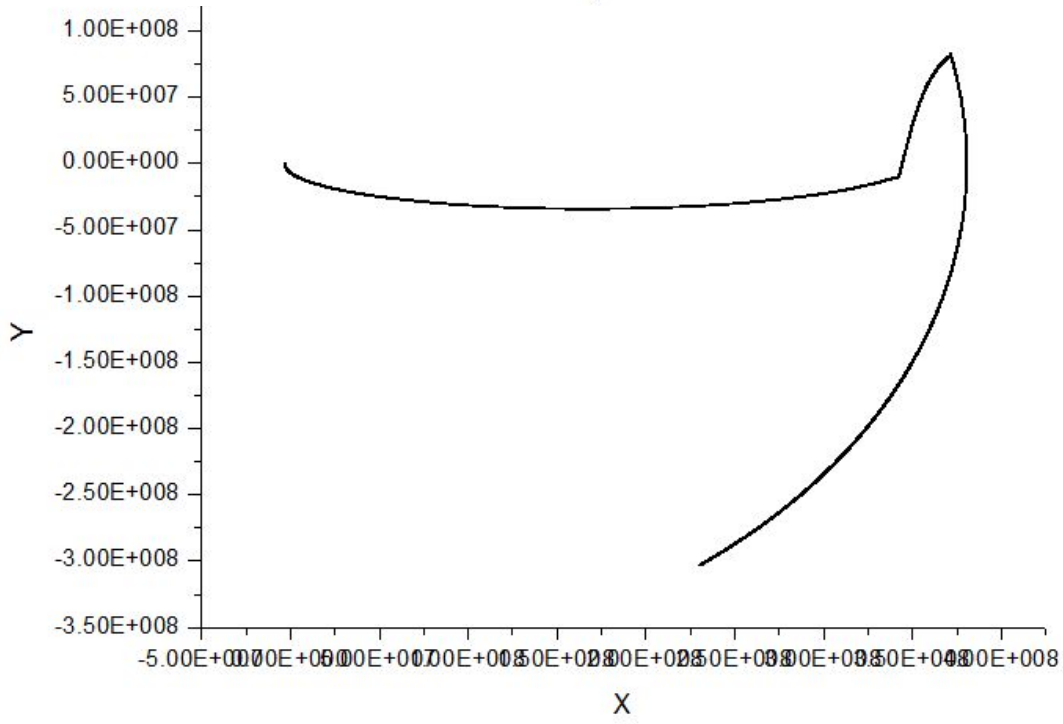
[1] From internet.

5.Final graphs

These are the graphs. The curve on them is the route of the craft, whereas the arc is that of moon's.

Congratulations! We approach moon.





6.Problems

1、In the end, we could see the period of changing velocity costs about 2 days. Actually it's not able to ignore it. Without considering about it, we may draw the wrong conclusion. If this happened in the reality, many people may die.

2、We make too many approximations, for simplicity, of course. But approximations will never occur in the real world. To improve it, we need to know the accurate route of moon and more other information.

3、The order of magnitude of power I set is meaningless, because if the engine supplied this power, the acceleration it caused would kill any human, even the strongest one on the planet. But even though, we still spend more than 2 days to change the speed. So we could image, in the future we need to find some solutions to deal with the problem of costing too much time to accelerate. It may be the theorem of relativity?

4、I have to admit I may not choose the most efficient route. In the graphs we could see the route is not very smooth. It's apparently not the best. It's still needed to be altered.

7.Codes

```
#include<iostream.h>
#include<math.h>
#include<fstream.h>
```

```
#define Pi 3.1415926

const double Me=5.965*1e24,Re=3.189*1e6;        //the mass and radius of earth
const double Mm=7.349*1e22,Rm=1.738*1e6;        //the mass and radius of moon
const double mm=1.5*1e4;                        //the mass of the craft
const double Dem=3.8*1e8;                       //the distance between earth and moon
const double WW=1.0*Pi/(27*24+7)/30.0;          //the angular velocity of moon over earth per minute
const double Orbit_e=Re+2.0*1e5;                //the orbital radius on earth
const double Orbit_m=Rm+5.0*1e3;                //the orbital radius on moon
const double G=6.67*1e-11;                      //the Gravitational constant
const double P=2.0*1e13*1e4;                        //set the power the engine supplies
const double t=4.0;                             //Assume how many days it will take to go to the critical point

double Critical(void)               //Calculate the critical distence between the craft and earth
{
    const double d1=(Me-sqrt(Me*Mm))/(Me-Mm);
    return Dem*d1;
}

double VelocityNeeded(double rr)                    //Calculate the minimum velocity needed to be enough to
                                                        go through the critical point
{
    const double W1=G*Me*mm*(1.0/Orbit_e-1.0/rr);    //the potential of earth that craft needs to overcome
    const double W2=G*Mm*mm*(1.0/(Dem-rr)-1.0/(Dem+Orbit_e));        //the potential given by moon
                                                            //According to the Kepler's law we have r1v1=r2v2
                                                        //and relate it with 1/2*mm*v1^2+W1-W2=1/2*mm*v0^2
    double v0=sqrt((W1-W2)*2.0*rr*rr/((rr*rr-Orbit_e*Orbit_e)*mm));    //we get this formula

    v0+=6.75;               //since it's critical, in case it fails to approach the point, I need to add a little extra speed
    return v0;
}

double Rungekutta_PredictOrbit(double v, double O, double t, double *xc, double *yc, double *vx, double *vy)
                                                    //simulate the origin position of moon and craft and their velocity
                                                    //to predict their orbits
{
    const double dt=0.1;
    const double dt2=dt*dt/2;
    const double dt3=dt*dt2/3;
    const double dt4=dt3*dt/4;

    double X_c=-1.0*Orbit_e,Y_c=0.0;                                    //set the initial position of craft
    double V_x=0.0,V_y=-1.0*v;                                          //set the initial velocity of craft
    const double Rc=Critical();
```

```
ofstream out("Predict_Orbit.txt");
out<<0.0<<'\t'<<0.0<<endl;

for(int i=0;i<t*(3600*24.0/dt);i++)
{
	if(i%(int)(60.0/dt)==0){
		O+=WW;
		const double X_m=Dem*cos(O),Y_m=Dem*sin(O);
		out<<i*dt/3600.0<<'\t'<<X_m<<'\t'<<Y_m<<endl;
	}
	const double X_m=Dem*cos(O),Y_m=Dem*sin(O);			//set the position of moon
	const double R1=sqrt(X_c*X_c+Y_c*Y_c);			//Calculate the distance between earth and craft

	const double R2=sqrt((X_m-X_c)*(X_m-X_c)+(Y_m-Y_c)*(Y_m-Y_c));
							//the distance between moon and craft
	const double A1=G*Me/R1/R1,A2=G*Mm/R2/R2;			//Calculate the acceleration
	const double A_x=-1.0*A1*X_c/R1+A2*(X_m-X_c)/R2;
	const double A_y=-1.0*A1*Y_c/R1+A2*(Y_m-Y_c)/R2;

	const double xd1=V_x;					const double yd1=V_y;
	const double Vxd1=A_x;					const double Vyd1=A_y;
	const double xd2=Vxd1;					const double yd2=Vyd1;
	const double Vxd2=-1.0*V_x*G*(Mm/R2/R2/R2+Me/R1/R1/R1);
	const double Vyd2=-1.0*V_y*G*(Mm/R2/R2/R2+Me/R1/R1/R1);
	const double xd3=Vxd2;					const double yd3=Vyd2;
	const double Vxd3=-1.0*A_x*G*(Mm/R2/R2/R2+Me/R1/R1/R1);
	const double Vyd3=-1.0*A_y*G*(Mm/R2/R2/R2+Me/R1/R1/R1);
	const double xd4=Vxd3;					const double yd4=Vyd3;
	const double Vxd4=-1.0*Vxd2*G*(Mm/R2/R2/R2+Me/R1/R1/R1);
	 const double Vyd4=-1.0*Vyd2*G*(Mm/R2/R2/R2+Me/R1/R1/R1);

	X_c+=xd1*dt+xd2*dt2+xd3*dt3+xd4*dt4;		Y_c+=yd1*dt+yd2*dt2+yd3*dt3+yd4*dt4;
	V_x+=Vxd1*dt+Vxd2*dt2+Vxd3*dt3+Vxd4*dt4;	V_y+=Vyd1*dt+Vyd2*dt2+Vyd3*dt3+Vyd4*dt4;

	out<<i*dt/3600.0<<'\t'<<X_c<<'\t'<<Y_c<<endl;
	//<<'\t'<<V_x<<'\t'<<V_y<<'\t'<<A_x<<'\t'<<A_y<<endl;

	if(Rc-R1<1000){					//check if it's approaching the critical point
		if(fabs(A_x)<0.001){			//check if the acceleration in x-axis is near zero
			*xc=X_c,*yc=Y_c;
			*vx=V_x,*vy=V_y;
			cout<<"============================"<<endl
				<<"approach the critical point"<<endl
```

```
                        <<"==========================="<<endl;
                    break;
                }
            }
        }

        out.close();
        return O;
}

double Rungekutta_PredictOrbit2(double V_x, double V_y, double X_c, double Y_c, double O, double *V2)
{
        const double dt=0.1;
        const double dt2=dt*dt/2;
        const double dt3=dt*dt2/3;
        const double dt4=dt3*dt/4;

        double tt=0.0;                                          //accout the total time we spend in this part

        ofstream out("Predict_Orbit.txt",ios::ate);

        for(int i=0;i<t*1500*24.0/dt;i++)
        {
            if(i%(int)(60.0/dt)==0){
                O+=WW;
                const double X_m=Dem*cos(O),Y_m=Dem*sin(O);
                out<<i*dt/3600.0<<'\t'<<X_m<<'\t'<<Y_m<<endl;
            }
            const double X_m=Dem*cos(O),Y_m=Dem*sin(O);                  //set the position of moon
            const double R1=sqrt(X_c*X_c+Y_c*Y_c);              //Calculate the distance between earth and craft
            const double R2=sqrt((X_m-X_c)*(X_m-X_c)+(Y_m-Y_c)*(Y_m-Y_c));
                                                                //the distance between moon and craft
            const double A1=G*Me/R1/R1,A2=G*Mm/R2/R2;                    //Calculate the acceleration
            const double A_x=-1.0*A1*X_c/R1+A2*(X_m-X_c)/R2;
            const double A_y=-1.0*A1*Y_c/R1+A2*(Y_m-Y_c)/R2;

            const double xd1=V_x;                                       const double yd1=V_y;
            const double Vxd1=A_x;                                      const double Vyd1=A_y;
            const double xd2=Vxd1;                                      const double yd2=Vyd1;
            const double Vxd2=-1.0*V_x*G*(Mm/R2/R2/R2+Me/R1/R1/R1);
           const double Vyd2=-1.0*V_y*G*(Mm/R2/R2/R2+Me/R1/R1/R1);
            const double xd3=Vxd2;                                      const double yd3=Vyd2;
            const double Vxd3=-1.0*A_x*G*(Mm/R2/R2/R2+Me/R1/R1/R1);
           const double Vyd3=-1.0*A_y*G*(Mm/R2/R2/R2+Me/R1/R1/R1);
```

```
            const double xd4=Vxd3;                                          const double yd4=Vyd3;
            const double Vxd4=-1.0*Vxd2*G*(Mm/R2/R2/R2+Me/R1/R1/R1);
           const double Vyd4=-1.0*Vyd2*G*(Mm/R2/R2/R2+Me/R1/R1/R1);

            X_c+=xd1*dt+xd2*dt2+xd3*dt3+xd4*dt4;              Y_c+=yd1*dt+yd2*dt2+yd3*dt3+yd4*dt4;
            V_x+=Vxd1*dt+Vxd2*dt2+Vxd3*dt3+Vxd4*dt4;      V_y+=Vyd1*dt+Vyd2*dt2+Vyd3*dt3+Vyd4*dt4;

            out<<i*dt/3600.0<<'\t'<<X_c<<'\t'<<Y_c<<endl;
            //<<'\t'<<V_x<<'\t'<<V_y<<'\t'<<A_x<<'\t'<<A_y<<endl;

            if(fabs(R2-Orbit_m)<100){                                   //check if we have approached moon
                        *V2=V_x*V_x+V_y*V_y;
                         cout<<"=========================================="<<endl
                               <<"Congratulations! We have arrived the moon! "<<endl
                               <<"=========================================="<<endl;
                         tt=1.0*i*dt/3600.0/24.0;
                         break;
            }
      }

      out.close();
      return tt;
}

void main(void)
{
      const double V1_e=sqrt(G*Me/Orbit_e);                             //the first universal velocity of earth
      const double V1_m=WW/60.0*Dem-sqrt(G*Mm/Orbit_m); //the first universal velocity of moon over earth

      const double rr=Critical();                                        //the criticle point I choose

      double v1=VelocityNeeded(rr);                                 //necessary velocity to go through critical point
      cout<<"the necessary velocity is      "<<v1<<endl;


      double oo=-24.0*60*t*WW;                                        //the corresponding angle of moon
      cout<<"the initial angle of moon is "<<oo/Pi<<"*Pi"<<endl;

      ofstream outfile("OriginPosition.txt");
      const double Xm=Dem*cos(oo),Ym=Dem*sin(oo);
      outfile<<0<<'\t'<<0<<'\n'<<Xm<<'\t'<<Ym<<'\n';                  //output the position of earth and moon
      outfile.close();

      double X_c=0.0,Y_c=0.0,V_x=0.0,V_y=0.0;
```

```
        oo=Rungekutta_PredictOrbit(v1,oo,t,&X_c,&Y_c,&V_x,&V_y);
        const double X_m=Dem*cos(oo),Y_m=Dem*sin(oo);

        cout<<"the positions of craft and moon are "<<endl
                <<"("<<X_c<<","<<Y_c<<")      ("<<X_m<<","<<Y_m<<")"<<endl;

        const double r1=sqrt(X_c*X_c+Y_c*Y_c),
                                r2=sqrt((X_m-X_c)*(X_m-X_c)+(Y_m-Y_c)*(Y_m-Y_c));

        double v2=sqrt(V_x*V_x+V_y*V_y);
        V_y+=WW/60.0*r1;                                        //some amendment caused by the rotation of moon
       double V2_x=V_x,V2_y=V_y;                                //Now we need to change the velocity of the craft
                                                                //but omit the time this procedure takes

        const double V2=sqrt(V2_x*V2_x+V2_y*V2_y);
        cout<<"the necessary velocity 2 is "<<V2<<'\t'<<"("<<V2_x<<","<<V2_y<<")"<<endl;

        double V3=0.0;
        double t2=Rungekutta_PredictOrbit2(V2_x,V2_y,X_c,Y_c,oo,&V3);//predict the orbit and meanwhile reserve
                                                                        //the time it takes and the final V3
        if(t2==0)cout<<"sorry we failed"<<endl;
        else{
                const double vv=VelocityNeeded(rr);
                const double W1=1.0/2*mm*(vv*vv-V1_e*V1_e);
                const double W2=1.0/2*mm*(V2*V2-v2*v2);
                const double W3=1.0/2*mm*(V3*V3-V1_m*V1_m);
                const double W=W1+W2+W3;
                const double t3=W/P;

                double T=t+t2+t3;
                cout<<"we spent "<<T<<" days and now we are at moon!"<<endl
                        <<t<<'\t'<<t2<<'\t'<<t3<<endl;
        }
}
```


Acknowledgment

Thanks for professor Dong's teaching. In this semester, I really learned something useful and I will keep studying to solve problems by computers!

Thank you!