PHYS 240: Computational Physics Final Report. IKAROS Solar Sail

Andrew Dynneson

May 14, 2019

1 Summary.

The IKAROS solar sail is the first proof of concept that a solar sail can actually work. IKAROS was deployed by the Japan Aerospace Exploration Agency (JAXA). JAXA scientists measured force from solar radiation pressure of 1.12 mN at 1AU. Surface Area of Sail - 196 m^2 [IKAROS Website]. Mass is 315 kg. Over a 23-hour period, attitude control of the solar sail was demonstrated by JAXA to be changed by 1/2 degree.

The attempted fly-by of Venus was not successful [Howell 2014], which was estimated to take about 1/2 year[Mori 2010].

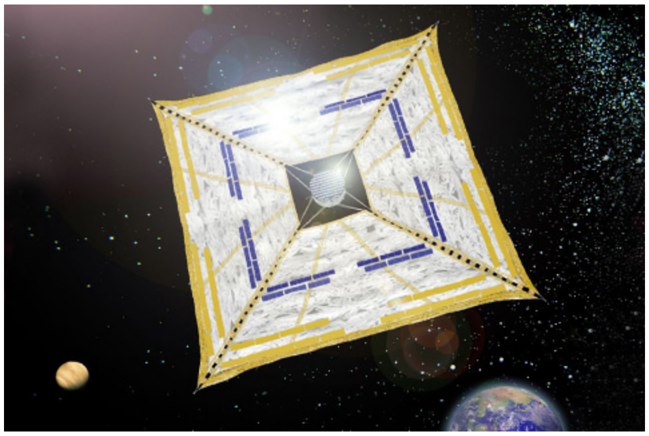


Figure 1: This figure caption describes the above graph in detail.

1.1 Project Goals.

To simulate inter-system navigation. In particular, simulate a fly-by of a celestial body.

Explain how the solar sail achieves attitude control by changing the color of the reflective 80 liquid-crystal panels. This is while the sail is spinning at 2rpm.

2 Equations Describing the Orbit.

This section uses the standard solution to the two-body problem from Celestial Mechanics. a denotes the semi-major axis of the elliptical orbit, while e is the ellipse's eccentricity. The semi-latus rectum is $p = a(1 - e^2)$. The true anomaly f, is the angle measured from periapisis (the point on the ellipse closest to the focus). In this case, the Sun is located at the focus which is closest to the periapsis point. Then, the distance from the Sun can be computed using the ellipse equation:

$$r = \frac{p}{1 + e\cos f}$$

$\mu = G \cdot M\odot$, where $G = 6.67 * 10^{-11}$ is Newton's gravitational constant, and $M\odot = 1.989 * 10^{30}$ kg is the mass of the Sun. The mean-motion can be computed from Kepler's 3rd Law: $n = \sqrt{\frac{\mu}{a^3}}$, since the mean-motion $n = \frac{2\pi}{\tau}$ has the period of the ellipse (τ) built into its definition.

The eccentric anomaly E is computed by numerically reversing Kepler's equation: $nt = E - e\sin E$. This was implemented using code found on github, which references [Murison 2006]. The specific angular momentum is $h = \sqrt{\frac{p}{\mu}}$

We start the solar sail at 1AU in an orbit similar to the orbit of the Earth. Somewhat arbitrarily, we begin at periapsis, $r_p = a(1-e)$. The eccentricity of this orbit is small, only about $e = 0.0167$. The starting major axis is $a♁ = 1.5214 * 10^{11}$ meters. The simulation stops and returns time of flight when the solar sail reaches the orbit either Venus ($a♀ = 1.0821 * 10^{11}$) or Mars ($a♂ = 2.279 * 10^{11}$).

3 Equations Describing Orbit Transfer.

3.1 Estimate for Solar Radiation Pressure [Rios-Reyes 2006].

$I_0 = 2.04 * 10^7$ W/m is the Sun's frequency integrated specific intensity. $c \approx 3.00 * 10^8$ m/s is the speed of light in a vacuum. $R\odot = 6.96 * 10^8$ m is Radius of Sun. r is the current distance of solar sail from Sun. $P(r)$ estimates the solar radiation pressure from the Sun, with current distance from the Sun as the only variable-input:

$$P(r) = \frac{2\pi I_0}{3c}\left(1 - \left[1 - \left(\frac{R\odot}{r}\right)^2\right]\right)^{1.5}$$

This estimate neglects angle of incidence of the solar rays. $P(r)$ predicts force of about 0.9 millinewtons at 1AU (for the IKAROS solar sail), which is pretty close to the actual force measured by JAXA scientists.

3.2 Perturbative Force Model.

Since the force for solar radiation is on the order of millinewtons, and the gravitational force is on the order of Newtons, a perturbation approximation method is appropriate for navigating within the solar system. First, we calculate the non-gravitational acceleration [specific force] by $F = P(r) * SA/m$, where SA is the surface area of the sail, and m is its mass. We then project the specific force into radial (R) and tangential (T) components. This takes into account the angle of incidence (θ) of the Sun's rays (reflected photons). Units of F are m/s^2:

$$d\overrightarrow{F} = R\hat{r} + T\hat{\theta} = F\cos(\theta)\hat{r} + F\sin(\theta)\hat{\theta}$$ [Murray and Dermott 1999]

3.3 Integrating Osculating Orbit Element Differentials.

The orbit elements eccentricity (e) and major axis (a) will change gradually due to the perturbing force of the solar radiation pressure. The amount by which they vary with time is given in differental-form, as they are implemented in the code. The change in orbit element is computed for a small increment of time (dt). For reference in derivative form (e.g. $\dot{e}$), check [Murray and Dermott 1999].

Ellipse eccentricity:

$$de = \sqrt{\frac{p}{\mu}}(R\sin f + T(\cos f + \cos E))dt$$

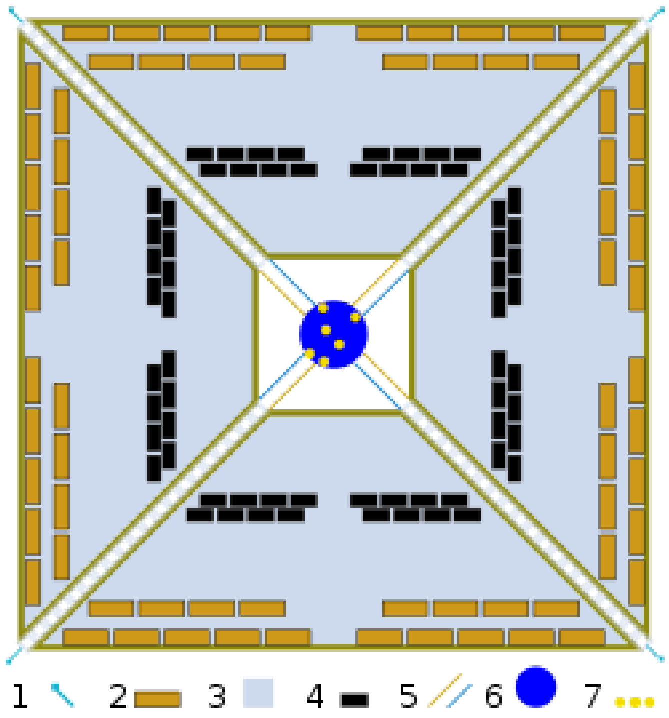


Figure 2: This figure caption describes the above graph in detail.

Ellipse major axis:

$$da = \frac{2}{n\sqrt{1-e^2}} \left(R \cdot e \sin f + T \cdot \frac{p}{r} \right) dt$$

3.4 Integration of differentials.

$$\Delta e = \int de \approx \sum \dot{e} \Delta t$$

$$\Delta a = \int da \approx \sum \dot{a} \Delta t$$

These sums were implemented by taking $e+ = de$ at each time-step. With the new osculating orbit elements, the current position r is computed, and compared with the orbit of $a♂$, for example. So, if the distance $a♂$ from the Sun is obtained, the program terminates and returns the time of flight.

4 Results and Conclusion.

From the solar radiation pressure, I wrote a program to simulate a fly-by of Mars in finite time. In theory, it should be possible to also simulate the fly-by of Venus, although it may require adjusting the angle of incidence during the transit. It might also be interesting to look into utilizing the Interplanetary Transport Network for efficient transit between celestial bodies. I would also have liked to provide more detail as to how to operate the attitude control.

Time-of-flight for transit to Mars:

- 40 years for a $1000m^2$ solar sail.
- 180 years for the $196m^2$ solar sail (IKAROS).

References

[1] https://en.wikipedia.org/wiki/IKAROS

[2] https://en.wikipedia.org/w/index.php?title=IKAROS&oldid=818169810

[3] http://global.jaxa.jp/projects/sas/ikaros/

[IKAROS Website] JAXA Website (2010). (This website has since been deactivated) https://web.archive.org/web/20100819085236/http://www.jspec.jaxa.jp/ikaros_channel/e/index.html

[Howell 2014] Elizabeth Howell (2014). Ikaros: First Successful Solar Sail https://www.space.com/25800-ikaros-solar-sail.html

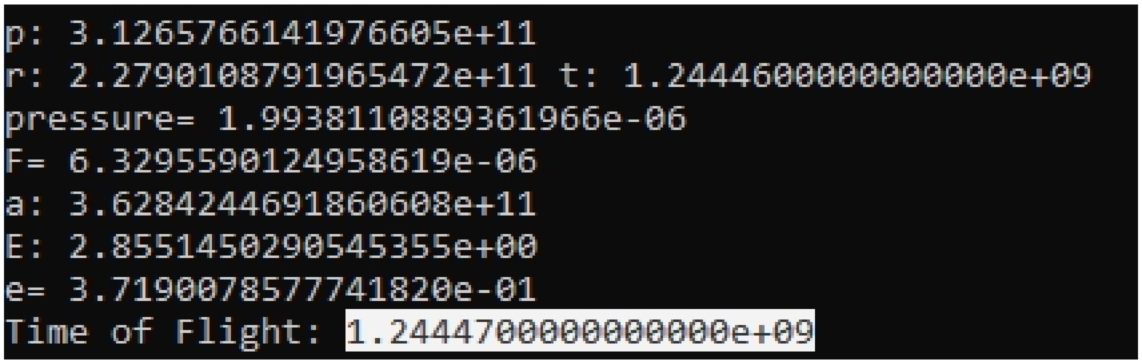
[Mori 2010] Mori et al (2010). First Solar Power Sail Demonstration by IKAROS. Trans. JSASS Aerospace Tech. Japan Vol. 8, No. ists27, pp. To 4-25-To 4-31

[Rios-Reyes 2006] Leonel Rios-Reyes (2006). Solar Sails: Modeling, Estimation, and Trajectory Control. *University of Michigan Thesis* (2.13).

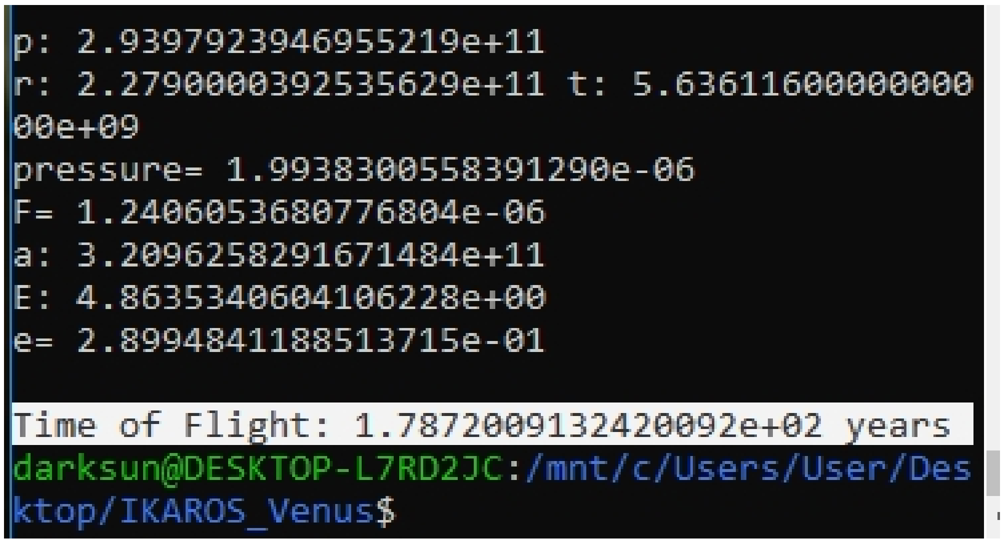
[Murison 2006] "A Practical Method for Solving the Kepler Equation", Marc A. Murison (2006).

[Murray and Dermott 1999] Murray, D. and Dermott, S. "Solar System Dynamics", Cambridge U. Press (1999).

A Sample Output.



```
p: 3.1265766141976605e+11
r: 2.2790108791965472e+11 t: 1.2444600000000000e+09
pressure= 1.9938110889361966e-06
F= 6.3295590124958619e-06
a: 3.6284244691860608e+11
E: 2.8551450290545355e+00
e= 3.7190078577741820e-01
Time of Flight: 1.2444700000000000e+09
```



```
p: 2.9397923946955219e+11
r: 2.2790000392535629e+11 t: 5.636116000000000
00e+09
pressure= 1.9938300558391290e-06
F= 1.2406053680776804e-06
a: 3.2096258291671484e+11
E: 4.8635340604106228e+00
e= 2.8994841188513715e-01

Time of Flight: 1.7872009132420092e+02 years
darksun@DESKTOP-L7RD2JC:/mnt/c/Users/User/Des
ktop/IKAROS_Venus$
```

B Software.

The code is implemented in C++. I have attached the program below.

```
//Solar Sail Navigation by Andrew Dynneson (2019).
//Succesful fly-by of Mars.
//Needs Inclination.
//Needs fly-by of Venus, which may require adjusting the angle
//while in transit.
#include <cmath>
#include <iostream>            // this has the cout, cin definitions
#include <iomanip>                // manipulators like setprecision
using namespace std;    // if omitted, then need std::cout, std::cin
double ecc_anomaly(double t, double period, double ecc, double time_peri);
double keplerstart3(double e, double M);
double eps3(double e, double M, double x);

int main() {
double dt=800.; //time interval in seconds
double t=0.;
double e=0.0167; //initial eccentricity
double a=1.496e11/(1.-e); //starts in heliocentric orbit at 1AU from Sun
double f=0.; //initial true anomaly
double r=a*(1.-e); //initial radial distance from Sun
double aVenus=1.0821e11; //fly-by of Venus
double aMars=2.279e11; //fly-by of Mars
double pi=3.14159265358979323846;
double mu=6.67e-011 * 1.989e30; // G*Mass_Sun
double m=315.; //mass of solar sail
double SA=196; //surface area of solar sail (not including center payload)
double I0=2.04e7; //frequency integrated specific intensity
double c=299792458; //speed of light in vacuum
double Rs=6.96e8; //radius of Sun
double theta= pi/4; //angle of incidence of sunlight

while((r>aVenus) && (r<aMars)) {

double p=a*(1-e*e); cout << "p: " << p << endl; //semilatus rectum
r=p/(1+e*cos(f)); //distance from Sun

cout << scientific << setprecision(16) << "r: " << r << " t: " << t << endl;
double pressure=2*pi*I0*(1-pow(1-pow(Rs/r,2),1.5))/(3*c); //estimated solar radiation pressure
cout << "pressure= " << pressure << endl;
double F=pressure*SA/m; cout << "F= " << F << endl; //estimated perturbing acceleration
double R=F*cos(theta); //radial acceleration component
double T=F*sin(theta); //angular acceleration component
double n=sqrt(mu / pow(a,3)); //mean motion from Kepler's 3rd Law
cout << "a: " << a << endl;
double E=ecc_anomaly(t, 2*pi/n, e, 0); cout << "E: " << E << endl;
double de=sqrt(p/mu)*(R*sin(f)+T*(cos(f)+cos(E)))*dt; //differential of eccentricity
e+=de; cout << "e= " << e << endl;
double da=2*dt*(R*e*sin(f)+T*p / r)/(n*sqrt(1-e*e)); //differential of major axis
a+=da;
double h=sqrt(p / mu); //specific angular momentum
double df=h*dt / r; //true anomaly differential
f+=df;
t+=dt;
cout << endl;
}

cout << scientific << "Time of Flight: " << t/(60*60*24*365) << " years" << endl;

return 0;
}

/**
Calculates the eccentric anomaly at time t by solving Kepler's equation.
See "A Practical Method for Solving the Kepler Equation", Marc A. Murison, 2006

@param t the time at which to calculate the eccentric anomaly.
@param period the orbital period of the planet
@param ecc the eccentricity of the orbit
@param t_peri time of periastron passage
@return eccentric anomaly.
*/
double ecc_anomaly(double t, double period, double ecc, double time_peri)
{
        double tol;
        if (ecc < 0.8) tol = 1e-14;
        else tol = 1e-13;

        double n = 2.*M_PI/period;  // mean motion
        double M = n*(t - time_peri);  // mean anomaly
        double Mnorm = fmod(M, 2.*M_PI);
        double E0 = keplerstart3(ecc, Mnorm);
        double dE = tol + 1;
        double E;
        int count = 0;
        while (dE > tol)
        {
                E = E0 - eps3(ecc, Mnorm, E0);
                dE = abs(E-E0);
                E0 = E;
                count++;
                // failed to converge, this only happens for nearly parabolic orbits
                if (count == 100) break;
        }
        return E;
}


//Provides a starting value to solve Kepler's equation.
//See "A Practical Method for Solving the Kepler Equation", Marc A. Murison, 2006

double keplerstart3(double e, double M)
{
        double t34 = e*e;
        double t35 = e*t34;
        double t33 = cos(M);
        return M + (-0.5*t35 + e + (t34 + 1.5*t33*t35)*t33)*sin(M);
}

double eps3(double e, double M, double x)
{
        double t1 = cos(x);
        double t2 = -1 + e*t1;
        double t3 = sin(x);
        double t4 = e*t3;
        double t5 = -x + t4 + M;
        double t6 = t5/(0.5*t5*t4/t2+t2);

        return t5/((0.5*t3 - 1/6*t1*t6)*e*t6+t2);
}
c
```