

“ p ” Iteration Algorithm

C. D. Hall

The p iteration algorithm solves the orbit determination problem given two position vectors $\vec{\mathbf{r}}_1$, $\vec{\mathbf{r}}_2$, and the time of flight Δt . The reference for this material is Chapter 5 of Bate, Mueller and White[1].

Given $\vec{\mathbf{r}}_1$, $\vec{\mathbf{r}}_2$, and Δt , determine p , a , and ΔE .

The basis of the algorithm is to “guess” a value of p , compute a , f , \dot{f} , g , ΔE , and Δt . Then compare the computed Δt with the desired Δt . If it’s correct, then we’re done. Otherwise, we must update the guess for p and repeat.

The following constants are computed before beginning the iteration, and are used throughout the algorithm:

$$\begin{aligned}
 r_1 &= \|\vec{\mathbf{r}}_1\| \\
 r_2 &= \|\vec{\mathbf{r}}_2\| \\
 \Delta\nu &= \vec{\mathbf{r}}_1 \cdot \vec{\mathbf{r}}_2 / (r_1 r_2) \\
 \Delta\nu &= 2\pi - \Delta\nu \text{ if the “long way” trajectory is sought} \\
 k &= r_1 r_2 (1 - \cos \Delta\nu) \\
 \ell &= r_1 + r_2 \\
 m &= r_1 r_2 (1 + \cos \Delta\nu)
 \end{aligned}$$

The calculations of a , *etc.*, using the “guess” of p are as follows:

$$\begin{aligned}
 a(p) &= \frac{mkp}{(2m - \ell^2)p^2 + 2klp - k^2} \\
 f(p) &= 1 - \frac{r_2}{p}(1 - \cos \Delta\nu) \\
 \dot{f}(p) &= \sqrt{\frac{\mu}{p}} \tan \frac{\Delta\nu}{2} \left[\frac{1 - \cos \Delta\nu}{p} - \frac{1}{r_1} - \frac{1}{r_2} \right] \\
 g(p) &= r_1 r_2 \sin \Delta\nu / \sqrt{\mu p} \\
 \sin \Delta E &= -r_1 r_2 \dot{f} / \sqrt{\mu a} \\
 \cos \Delta E &= 1 - (1 - f)r_1/a \\
 \Delta E(a) &= \text{atan2}(\sin \Delta E, \cos \Delta E) \text{ atan2 does the quadrant check} \\
 \Delta t &= g + \sqrt{a^3/\mu}(\Delta E - \sin \Delta E)
 \end{aligned}$$

These equations can be thought of as providing a calculation of the form

$$F(p) = \Delta t - \Delta t_{\text{desired}}$$

Thus the algorithm is reduced to finding a root of $F(p)$. Note that if the value of p leads to a negative value of a , then the orbit is hyperbolic, and ΔF must be used instead of ΔE . This note only addresses the elliptical case.

Two additional quantities are useful in determining an initial guess for the iteration algorithm. If Newton's method is used, only one initial guess is required. However, if the secant method is used, two initial guesses are required. The version of the algorithm presented here uses the secant method, so we need two initial guesses for p .

A quadratic function of p appears in the denominator of the expression $a(p)$:

$$\frac{mkp}{(2m - \ell^2)p^2 + 2klp - k^2}$$

This quadratic has two roots:

$$p_i = \frac{k}{\ell + \sqrt{2m}} \text{ and } p_{ii} = \frac{k}{\ell - \sqrt{2m}}$$

Note that $p_i < p_{ii}$. These values of p correspond to the special cases of parabolic orbits connecting the two position vectors. Since $p \rightarrow \Delta t$ as described above, we can compute the Δt associated with these two parabolas (using the appropriate parabolic eccentric anomaly) and one can show that the minimum time for an elliptical transfer going the "short way" is $\Delta t(p_{ii})$. Similarly the minimum time for an elliptical transfer going the "long way" is $\Delta t(p_i)$. In each of these cases, if the desired Δt is greater than the corresponding minimum Δt , then the solution will be an elliptical orbit. Otherwise the solution will be a hyperbolic orbit.

For the initial guesses for the secant method, I have found that guesses that divide the interval $[p_i, p_{ii}]$ into thirds work well in most cases. That is, initialize p and $F(p)$ as follows:

$$\begin{aligned} p_0 &= \frac{1}{3}(2p_i + p_{ii}) \Rightarrow F_0 = \Delta t(p_0) - \Delta t_{\text{desired}} \\ p_1 &= \frac{1}{3}(p_i + 2p_{ii}) \Rightarrow F_1 = \Delta t(p_1) - \Delta t_{\text{desired}} \end{aligned}$$

Now select a tolerance TOL and a maximum number of iterations MAXIT. I generally use TOL = 10^{-11} and MAXIT = 20. Initialize the iteration number as ITNUM=0. The iterative algorithm is then

`while(|F1| > TOL & ITNUM < MAXIT)`

$$\begin{aligned} \Delta p &= -\frac{(p_0 - p_1)F_1}{F_0 - F_1} \\ p_n &= p_1 + \frac{\Delta p}{1 + \Delta p^2} \\ F_n &= \Delta t(p_n) - \Delta t_{\text{desired}} \\ p_0 &= p_1 \\ p_1 &= p_n \\ F_0 &= F_1 \\ F_1 &= F_n \\ \text{ITNUM} &= \text{ITNUM} + 1 \end{aligned}$$

Note the step limiter in the calculation of p_n . This keeps the algorithm from diverging if the value of Δp is large.

When the algorithm converges, you will have the values of p , a , f , g , \dot{f} , \dot{g} , ΔE , and Δt . You can use these to calculate \vec{v}_1 and \vec{v}_2 , and then compute the required Δv for a given rendezvous or intercept problem.

Targeting: An Illustrative Example

We want to launch from Blacksburg ($L = 37.23274^\circ$) at a time when $\theta = 0$, and to reach a target (either rendezvous or intercept) with a satellite with the following orbital elements: $a = 7000$ km (1.0975 DU), $e = 0$, $i = 63^\circ$, $\Omega = 0$, $\nu = 90^\circ$.

For the Gauss (or Lambert) problem the two vectors are

$$\begin{aligned}\vec{r}_1 &= R_\oplus (\cos L \hat{\mathbf{I}} + 0 \hat{\mathbf{J}} + \sin L \hat{\mathbf{K}}) \\ \vec{r}_2 &= a(0 \hat{\mathbf{I}} + \cos i \hat{\mathbf{J}} + \sin i \hat{\mathbf{K}})\end{aligned}$$

The angle between these two vectors is the change in true anomaly, $\Delta\nu = 1.0014$. Using this information, we can compute the time-of-flight as a function of the parameter, p . First we compute the semimajor axis a as a function of p . See Figure which shows a vs. p in the elliptical transfer orbit range, where p lies between p_i and p_{ii} . For this problem, $p_i = 0.1285$ and $p_{ii} = 1.9495$. Then we can compute the change in eccentric anomaly, ΔE , and then time-of-flight. We do this for the range of values of p between p_i and p_{ii} .

We can pick a time of flight to compute a specific transfer orbit.

For example, let's take the time of flight to be 5 TUs. The secant iteration method gives the following sequence:

Parameter, p	Semimajor axis, a	$\Delta t - \Delta t_{\text{desired}}$
1.3425	1.5245	7.4270
0.8842	0.9191	-3.7899
1.0354	1.0452	-3.9177
0.8294	0.8841	-3.7317
0.6009	0.7893	-3.3590
0.2080	1.2579	2.4809
0.3704	0.8115	-2.3170
0.2924	0.9010	-1.2836
0.1965	1.3798	3.8393
0.2680	0.9562	-0.7039
0.2569	0.9892	-0.3611
0.2453	1.0316	0.0777
0.2473	1.0234	-0.0069
0.2472	1.0240	-0.0001
0.2472	1.0241	0.0000
0.2472	1.0241	0.0000

From p and a , we can compute the eccentricity of the transfer orbit: $e = 0.8710$.

The next step is to solve for f , \dot{f} , g , \dot{g} , and get the velocity required at launch: $\vec{v}_1 = (\vec{r}_2 - f\vec{r}_1)/g$. Here $f = -1.0465$, $\dot{f} = -0.0511$, and $g = 1.8953$, from which $\dot{g} = -0.8647$. This leads to $\vec{v}_1 = 0.4481\hat{\mathbf{I}} + 0.2680\hat{\mathbf{J}} + 0.8665\hat{\mathbf{K}}$ and $\vec{v}_2 = -0.4282\hat{\mathbf{I}} - 0.2317\hat{\mathbf{J}} - 0.7802\hat{\mathbf{K}}$.

Now we know the initial position and velocity vectors at launch that are required to do the transfer in 5 TUs. We also know the velocity that the spacecraft will have at the target point. If we're doing a rendezvous, we can compute the velocity of the target and compute the required $\Delta\vec{v}$ at the rendezvous. If we want to intercept, then we can compute the relative velocity to determine the impact speed of the two objects.

How would we go about using this approach to solve a rendezvous or intercept problem?

For the given problem, we specified the point of target "interception" and the local sidereal time of the launch site at launch. If we want to do mission planning, we would need to know where the target is at a given time, from which we can compute where it will be after a given Δt . Thus we can look at a variety of Δt 's, each of which will give a different \vec{r}_2 , and compute the required $\Delta\vec{v}$.

References

- [1] Bate, R. R., Mueller, D. D., and White, J. E., *Fundamentals of Astrodynamics*, Dover, New York, 1971.