# Spacecraft Trajectory Optimisation Using Particle Swarm Optimisation Methods

**Jake Watson**

24th January, 2018

**Abstract**

# Contents

Introduction ..... 3
1.1 Introduction ..... 3
1.2 Patched Conics ..... 4
1.3 Calculating Orbital Insertion Velocity ..... 5
1.4 Lambert Method ..... 8
1.5 Particle Swarm Optimisation ..... 8
Trajectory Optimiser ..... 11
1.6 Class Overview ..... 12
1.7 Outline of Operation ..... 13

# Chapter 1

# Introduction

Spacecraft trajectory optimisation is the problem of determining the trajectory of a spacecraft that satisfies path constraints, such as the duration, while minimizing a certain performance index. The objective of this work was to implement a basic trajectory optimiser for high-thrust spacecraft, travelling between Earth and Mars.

## 1.1 Introduction

Finding the optimal trajectory is vital for any space mission design. If a manned mission could carry only enough supplies to sustain life for a certain period of time, then it is necessary to determine the trajectory of the spacecraft with a duration shorter than this. Other variables, such as the total change in velocity, which will govern the amount of fuel required by the spacecraft, can also be optimised.

Chemical rockets were the first propulsion systems to be used in spacecraft design. These systems rely on large volumes of fuel to deliver high thrust, but are limited in their ability to deliver high impulse due to the limitation in fuel capacity. Because the fuel is bulky and heavy, only enough can be carried to ignite the engines for a short time. For the rest of their time in flight, the spacecraft creates no thrust.

Because the burn time of this type of spacecraft is so short compared to the total time of flight, the propulsion can be modelled as impulsive - the spacecraft is approximated to have instantaneous acceleration and deceleration at the start and end of the trajectory.

## 1.2 Patched Conics

This project used the well-known 'patched-conics' approximation in order to find suitable interplanetary transfers. In the 2-body problem, it is possible to use conic sections to analytically find the path between two planets. However with the introduction of more planets (n-body problem), this approach fails.

As a result, the 'patched-conics' method was developed. This method is based on separating the trajectory into three parts:

1) Hyperbolic path from periapse to the edge of the sphere of influence of the start body.

2) Heliocentric transfer path from the sphere of influence of the start body to the sphere of influence of the end planet.

3) Hyperbolic path from the edge of the sphere of influence of the target body to the periapse of the target body.

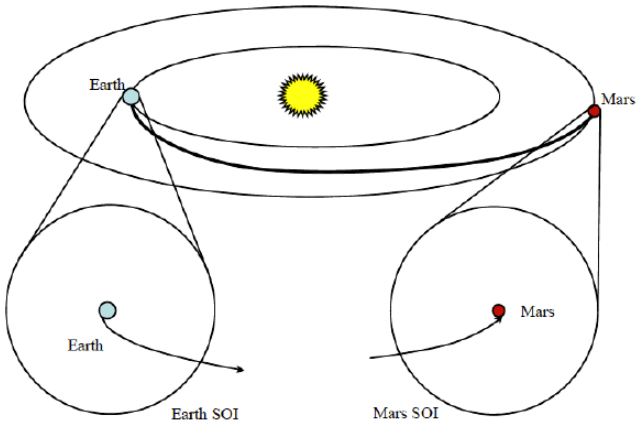


*Figure 1: Summary of the patched conics concept.*

If the velocities at the edge of the spheres of influence at each body match to the velocity of the transfer orbit, then the patched-conics method will provide a good approximation to the true trajectory.

Each of the three sections of the trajectory is modelled as a 2-body problem, with the spacecraft travelling under the influence of a central gravitational body. In the first and third sections, that body is the start and target planets respectively, while in the second interplanetary section the central body is the sun. This simplification can be made because the dominant gravitational source outside of the sphere of influence of a planet is the sun.

## 1.3 Calculating Orbital Insertion Velocity

In this project, the spacecraft is imagined to be in a stable parking orbit around Earth at the beginning of its trajectory. In order to leave this orbit and embark on its interplanetary transfer, it must increase it's velocity (modelled as instantaneous acceleration) and travel along a hyperbolic path until it reaches the edge of the sphere of influence of Earth. Once it reaches the sphere of influence of Mars, this process is reversed - it will decrease its velocity instantaneously and enter a stable parking orbit around Mars.

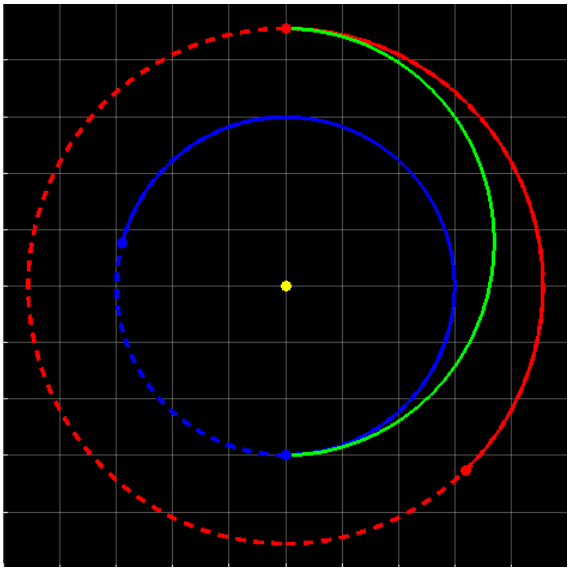

*Figure 2: Geometry of the transfer orbit.*

For a spacecraft leaving Earth orbit, we must first know the orbital velocity of Earth. This is given by:

$$v_E = \sqrt{\frac{\mu_S}{a_E}}$$

Where $v_e$ is the orbital velocity of Earth, $mu_s$ is the gravitational parameter of the sun, and $a_e$ is the orbital radius of Earth.

Next we will need the transfer semimajor axis, given by:

$$a_T = \sqrt{\frac{a_E + a_M}{2}}$$

Where $a_T$ is the semimajor axis of the transfer orbit, $a_E$ is the orbital radius of Earth and $a_M$ is the orbital radius of Mars.

Now we can use the Vis-Viva equation to calculate the velocity of the spacecraft on departure from Earth orbit.

$$v_{TP} = \sqrt{\mu_E (\frac{2}{a_E} - \frac{1}{a_T})}$$

Where $v_{TP}$ is the velocity of the spacecraft at the periapsis of the transfer orbit (leaving the sphere of influence of the Earth), $mu_E$ is the gravitational parameter of the Earth, $a_E$ is the orbital radius of the Earth and $a_T$ is the semimajor axis of the transfer orbit.

Now, given the velocity of the spacecraft at departure, and the orbital velocity of the Earth, we can find the hyperbolic excess velocity of the spacecraft at departure(the difference between the velocity of the spacecraft at the transfer periapsis and the orbital velocity of Earth).

$$v_{E->TP} = v_{TP} - v_E$$

Where $v_{E->TP}$ is the hyperbolic excess velocity of departure.

Next, we can use the hyperbolic excess velocity of departure to calculate the semimajor axis of the hyperbolic path of departure,

$$a_{E,S} = -\frac{\mu_E}{v_{E->TP}^2}$$

Finally, putting it all together, we can calculate the change in velocity Δv required to change from Earth parking orbit to the periapsis of the transfer orbit.

$$\Delta v = \sqrt{\mu_E \left(\frac{2}{r_{E,S}} - \frac{1}{a_{E,S}}\right)} - \sqrt{\frac{\mu_E}{r_{E,S}}}$$

## 1.4 Lambert Method

To determine the path of the interplanetary trajectory, the Lambert method of Universal Variables was used. The Lambert method was discovered by Johann Heinrich Lambert, and provides the initial and final velocity vectors of the trajectory, given:

1) The starting position of the trajectory
2) The end position of the trajectory
3) The duration of the transfer

A detailed description of this method shall not be covered in this project. The implementation of the Lambert method is based on Algorithm 52 found in Fundamentals of Astrophysics and Applications by Vallado.

## 1.5 Particle Swarm Optimisation

*CHAPTER 2: TRAJECTORY OPTIMISER DESIGN*

Particle swarm optimisation is a computational method of optimising a problem, by using a 'swarm' of candidate solutions, or 'particles', to search the solution space.

The method randomly generates a large number of particles (solutions), giving them a position in the search space, and a velocity. Each particle's velocity changes with each iteration, and is influenced by that particle's best known position (best solution that particle has found so far), and by the best known position of the entire swarm. These best known positions are updated whenever a particle discovers a better solution.

This guidance from other particles is the characteristic of a swarm, and distinguishes it from a crowd of particles moving independently from one another in the swarm. Additionally, the particle's velocity has an element of randomness, ensuring that the particles search the solution space fully and do not get stuck in local minima.

PSO has several advantages over other methods of optimisation: firstly, it makes no assumptions about the problem being optimised and can search large solution spaces. It also does not use the gradient of the problem, allowing it to be used on non-differentiable functions. This is in contrast to classic optimisation methods such as gradient descent.

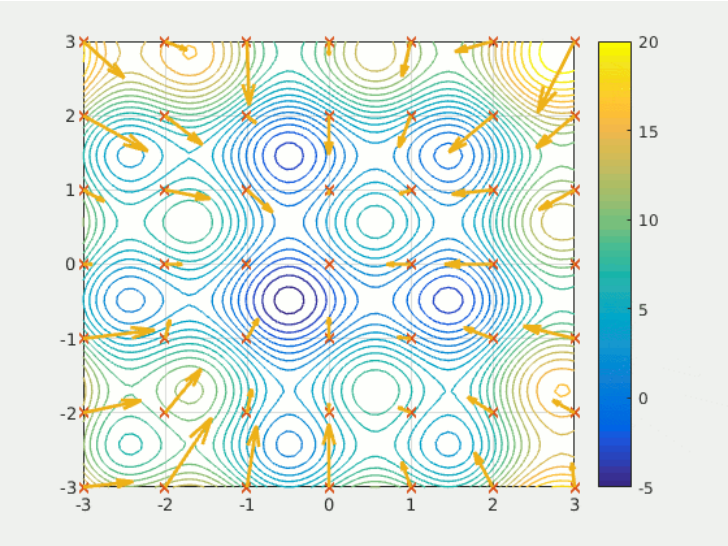


*Figure 3: A contour plot of the solution space, with the particles imposed.*

In this project, the aim was to use the Particle Swarm Optimisation method described above, to minimise the change in velocity Δv associated with an interplanetary transfer from Earth to Mars, and back again.

The change in velocity required for a space mission dictates the amount of fuel that must be carried for the mission. For a higher change in velocity, more fuel must be carried, while for a lower change in velocity less fuel is needed. This matters because spacecraft are extremely limited in terms of weight and volume, so it is important to minimise the amount of fuel required, and therefore to choose a trajectory that will minimise Δv.

In the next Chapter, the design of the optimiser and trajectory calculator is outlined.

# Chapter 2

# Trajectory Optimiser

*CHAPTER 2: TRAJECTORY OPTIMISER DESIGN*

This chapter details the design and implementation of the trajectory calculator, and of the particle swarm optimiser.

## 1.6 Class Overview

The trajectory optimiser was coded in Java, thanks to its use in SCISYS, and due to its large mathematical libraries. Additionally, some code in the Solar-Simulation class was already made in Java.

The trajectory optimiser was split into several classes, each performing different functions. A broad overview of each class and its use is now presented:

1) PSO: This class contains the particle swarm optimisation algorithm itself. This is covered in more depth in a later section. The class creates a large number of Trajectory objects, to act as the particles. Each Trajectory is created with:

a) randomised start date,

b) transfer duration from Earth to Mars (in days),

c) stay duration (days spent in Mars orbit),

d) transfer duration from Mars to Earth (in days).

The class requires the input of an end-year value, which creates an upper limit on the values of the start dates for the Trajectories. The lower bound on the start dates is hard-coded as 1/1/2020.

2) SolarSim: This class is responsible for creating the ephemeris for Earth and Mars from the hard-coded start date (1/1/2020) until the given end date. It uses the LeapFrog algorithm in order to solve the n-body problem of simulating the movement of every planet in the solar system, and stores the position and velocity of Earth and Mars for every day between the start date and end date in an Ephemeris object. The initial starting positions of the planets on the start date are taken from NASA data.

3) LambertSolver: This class implements Lamberts method of Universal Variables. Given an initial and final position, a trajectory start date, and a transfer duration, it will return the initial and final velocities of the transfer (periapsis and apoapsis velocities).

4) VelocityChange: This class determines the total change in velocity associated with any given Trajectory. This value is passed back to the PSO object.

5) Main: This is the main class of the simulation, which instantiates all of the other classes.

## 1.7 Outline of Operation

This section outlines the process by which the optimiser will determine the optimum trajectory for any given period.

1) The SolarSim class is instantiated and passed the value of end_date. It then simulates each planet in the Solar System, and stores the position and velocity for Earth and Mars once for each day between the start date and end date in an Ephemeris object.

2) The PSO class is instantiated and passed the value of end_date. The PSO instance generates a large number of Trajectory objects with randomised characteristics.

3) To determine the start_position and end_position values for each Trajectory, the start_date and end_date are compared to the values of Earth's and Mars' positions stored in the Ephemeris object.

4) To determine the start_velocity and end_velocity for each Trajectory, the LambertSolver class is passed the start_position, end_position, transfer_duration, and start_date of the Trajectory in question.

5) Finally, to determine the $\Delta v$ value for each Trajectory object, the VelocityChange class is passed the start_velocity and end_velocity values for the Trajectory in question.

6) Each Trajectory now has values for all of start_position, end_position, start_velocity, end_velocity, start_date, and transfer_duration.

7) These completed Trajectories are now passed back to the PSO instance, which moves forward one iteration in its algorithm.

8) Each Trajectory will now have its values for the following variables modified by the PSO algorithm:

a) start date,

b) transfer duration from Earth to Mars (in days),

c) stay duration (days spent in Mars orbit),

d) transfer duration from Mars to Earth (in days).

The steps from 3-8 will now be repeated until the PSO algorithm achieves convergence.

9) Finally, once the PSO algorithm has converged, the 5 best Trajectories are returned as the output of the optimiser. These Trajectories will each be separated in start_date by a set time, so as to give a range of useful Trajectories.