**Illinois Institute of Technology**
Spring 2005

**Physics 240**
***Computational Physics***
Prof. N. Solomey

## Final Project Report

# Lagrange-point Seeker

by Nil Valls

### Introduction

Josef Lagrange discovered five points of equilibrium in the Earth-Sun system, namely Lagrange points. These are points where all the force vectors due to the nature of the system cancel out. Usually, in a planetary system, the significant forces are gravitational, centrifuge and Coriolis.

## Objective

To develop a program to seek for Lagrange points in a two-dimensional system of N static masses, provided a value and location for each mass and a starting point.

## Description of the method

For this project, I developed a method inspired in the technique golfers use to put the ball in the hole. A test mass (ball) is located at an initial searching point (beginning of the course). Then, the 360º of the circumference around the point are scanned, and the direction that leads to the lowest potential is chosen. The mass is then moved a distance *dr*, proportional to the distance between the masses, in the chosen direction. The potential around the test mass is scanned again and the new direction is chosen. The mass is then moved a new distance *dr*', which is reduced as the potential decreases. Just like a golfer hits the ball farther as he is far away from the hole, and reduces the range of each hit as he gets closer to the hole (green). Once the potential is zero or within the threshold, the position of that

point is output as a Lagrange point (hole). In order to seek for another Lagrange point, a new starting point must be picked.

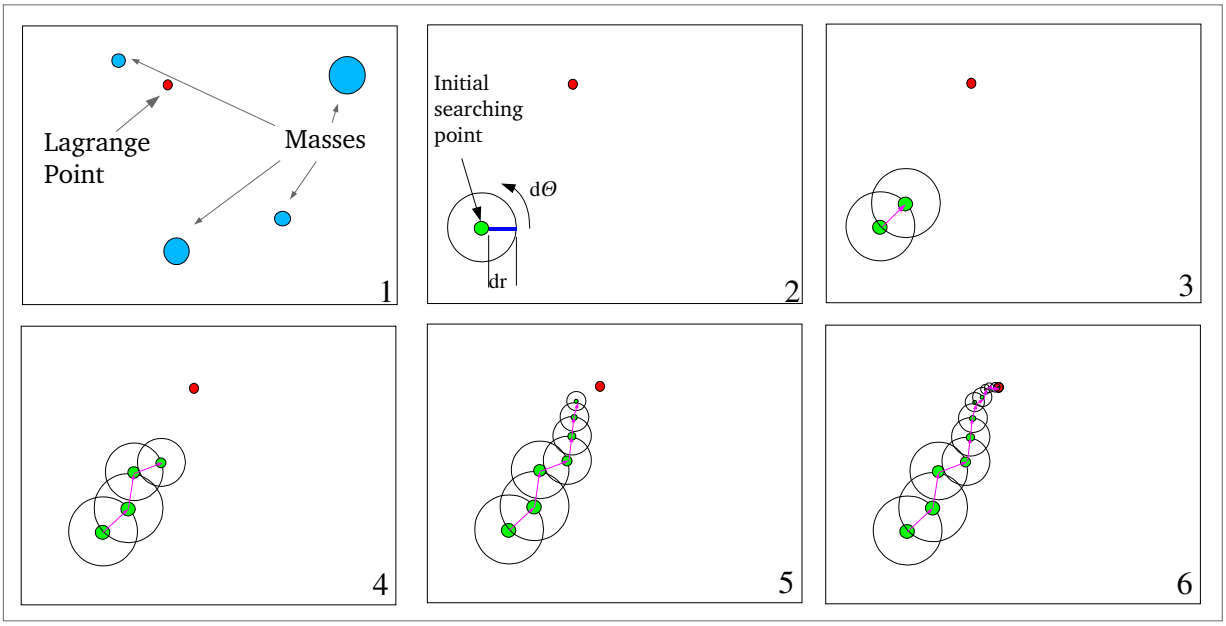


***Figure 1:*** *sequence of the basic steps followed by the Lagrange point seeker. After scanning the 360º, the direction of the gradient to the lowest potential is chosen. The test mass (green) is moved dr in that direction. Note that the dr is smaller as the test mass becomes closer to the LP. Also note that, although dΘ is fixed, the decrease in dr contributes to the angular accuracy of the polar scan.*

## Description of the program

- **The User Prompt**

The Lagrange-point seeker (LPS) is programed in Fortran77. It considers a 100 by 100 plane. The user is asked for the number of masses, value and Cartesian coordinates of the location of each mass.

```
* Asking for the value of each mass and its position

21   do i=1,N
31    write (6,*) 'Enter mass value (0-100) for mass number ', i
      read (5,*) m(i)

      if (m(i).gt.0.and.m(i).le.100) then
        goto 32
      else
       write (6,*) 'Out of limits. Please reenter value for mass ', i
       write (6,*) 'Please enter a value between 0 and 100'
       goto 31
      endif
32    write (6,*) 'Enter x coordinate (+-100) for mass number ', i
      read (5,*) x(i)
     if (x(i).ge.-100.and.x(i).le.100) then
      goto 33
      else
       write (6,*) 'Out of limits.'
       write (6,*) 'Please reenter x coordinate for mass number ', i
       write (6,*) 'Please enter a value between -100 and 100'
       goto 32
       endif
```

Note, from the segment of code above, that the LPS refuses data which are out of the limits. Also, as you may appreciate, each measure is stored as an array. Hence, there exists an array of magnitude N for the x-position, one for the y-position, one for the mass, and so forth. The user inputs are easily read by using loops, which change the allocation of each datum in each particular array.

Finally, the user is asked for a guess in the search. This guess, having an extremely low probability of hit, is used as the initial searching point. Because gravity is unidirectional (inwards), the static Lagrange point(s) will always lay within the area bounded by the masses. Consequently, in order to avoid the acceptance of far points which have very low potential, the initial guess is restricted to the inner 80x80 square. In addition, in the case in which the user sets an initial searching point with a mass already on it, the potential goes to infinity. To solve this, if the initial guess coincides with the location of one of the masses, the initial guess or searching point is shifted a small quantity (0.333units).

```
* Asking for initial searching point (just a guess)

5    write (6,*) 'Please specify initial search point'
     write (6,*) 'x coordinate (+-80):'
     read (5,*) x0
     write (6,*) 'y coordinate(+-80):'
     read (5,*) y0
  do i=1,N
    if (x0.eq.x(i).and.y0.eq.y(i)) then
      x0=x0+0.333
      y0=y0+0.333
     else
    endif
    enddo
```

- **Finding the net gravitational force**

For this program, it is important to be able to find the net gravitational force at any point. Moreover, it is essential to obtain a vector for each force so that they can be added vectorially. The *GRAVFORCE* subroutine does this in a simple way.

```
    do i=1,N
      r=sqrt((x(i)-xi)**2+(y(i)-yi)**2)
      Fx(i)=m(i)*(x(i)-xi)/(r**2)
      Fy(i)=m(i)*(y(i)-yi)/(r**2)
      Fxtot=Fxtot+Fx(i)
      Fytot=Fytot+Fy(i)
    enddo

   Fg=sqrt(Fxtot**2+Fytot**2)
```

Using Newton's Principle of Universal Gravitation, this subroutine finds the gravitational force vector from each gravitational source, i.e., mass. By looping through all of the masses, the vectorial quantities are added. As a final result, only the magnitude of the vector sum is necessary, hence the square root of the sum of the squares of each component is calculated.

As the test mass approaches to a Lagrange point, the net gravitational force needs to be calculated more and more precisely. For this reason, this subroutine returns the net gravitational force with double precision. As we will see, other subroutines will shape the precision of the figures as they need to be.

You may note that the gravitational constant is not present in the code. In fact, the whole program is written in such a way that it can be easily modified to the characteristics of the systems that needs to be studied. For our purposes, the gravitational constant is not relevant. On the other hand, the inverted r squared relationship is extremely determining.

- **The Polar Scanner**

As described formerly, the LSP finds the direction to the lowest potential from any point. This is accomplished by the *RADSCAN* subroutine.

```
***  RADIAL SCANNER
* (returns x0,y0 as new point coordinates, and Fgn1  as previous netFg)
     subroutine radscan(x0,y0,dx,dy,dr,Fgk)
   -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -

     do j = 0, fintheta
     arg=(j*dtheta)*pi/180
     xi=x0+dr*Cos(arg)
     yi=y0+dr*Sin(arg)
     Call gravforce (xi, yi, Fg)

      if (Fg.lt.Fg0) then
               Fgk=Fg
               dx=xi
               dy=yi
              else
             endif
             Fg0=Fgk

             enddo
             return
             end
```

This subroutine is given a distance *dr*, proportional to the magnitude of the net gravitational force at the present point. *GRAVFORCE* is called for each single point laying on the circumference around the point in question, and given the coordinates of the new point in question, which depend on the angle and distance from the present point.. The angular increments are set by default at 1º. *RADSCAN* compares each new value of the net gravitational force to the lowest recorded in the present scan. In the end, of the 359 pairs of coordinates, only the one that corresponds to the lowest net gravitational force is returned.

- **Guided walk**

Having now the right tools to map the space and head to a Lagrange point, let us analyze the main routine of the LSP.

RADSCAN is given the initial searching point, which the user input. It returns the coordinates of the new point (with lower potential), *dr* away from the present point. RADSCAN is now called with the coordinates of the new point. New coordinates are returned. This process is repeated a certain amount of times, easily definable. By default, it is set to 100 iterations. Nevertheless, it has been observed that only between 10 and 20 are needed, since the LSP adapts the magnitude of *dr* to the proximity of the test mass with respect to the Lagrange point.

```
do i=0,100
call radscan(x0,y0,dx,dy,dr,Fgk)

      x0=dx
      y0=dy
  Fratio=((Fginit/Fgk))
  dr=dr/Fratio
  Fginit=Fgk
enddo
xl=nint(dx)
yl=nint(dy)
```

Note from this piece of code how the new *dr* is defined by quotient of the old *dr* and the *Fratio*, where the *Fratio* is the ratio of the net gravitational forces between the previous point and the present one.

Finally, once the loop is complete, the last coordinates returned by RADSCAN are considered the coordinates of a Lagrange point. Even though the coordinates are returned as real values, as a final answer, they are rounded off to the nearest integer. This is sufficient precision considering the dimensions of the workspace.

- **More LPs?**

Since only a single LP can be found per analysis, the user is queried whether a new analysis should be started by choosing a different starting point. Depending on the system that is being analyzed, multiple LPs may be found. With the current state of development of the LSP, the analysis has to be restarted every time we attempt to find a new LP.

## Results

Some examples with the corresponding results:

| | $x$ | $y$ | $m$ |
|---|---|---|---|
| M1 | 50 | 30 | 10 |
| M2 | -70 | 30 | 10 |
| M3 | -70 | -10 | 10 |
| M4 | 50 | -10 | 10 |
| Initial Point | 60 | -70 | |
| **Lagrange Point** | **-10** | **10** | |

| | $x$ | $y$ | $m$ |
|---|---|---|---|
| M1 | 20 | 25 | 35 |
| M2 | -40 | 81 | 5 |
| M3 | 56 | -67 | 33 |
| M4 | -16 | -32 | 7 |
| M5 | 8 | 19 | 14 |
| Initial Point | 8 | -40 | |
| **Lagrange Point** | **38** | **-34** | |

## Conclusions

The Lagrange-Point Seeker is fully functional. The “guided walk” approach is very accurate, as we can see from cases in which, because of symmetry, we can easily know the location of a Lagrange point. Moreover, the precision of the program is considerable, and could be easily enhanced if needed.

## Final comments

This program could be useful to find some of the Lagrange points of systems with multiple masses, for example the solar system, for which analytic calculations become chaotic. Also, dynamic features could be implemented. The program can be found in */home/vallnil/final/* under the name of *lagrange.exe*, within the LS112 cluster.