Applications of Space Science PHYC40730

Satellite Navigation Assignment

(Submission Deadline: See Module's Brightspace)

Background: You are somewhere in Ireland with a Sat Nav receiver. The receiver obtains signals from four satellites. At the same instant (epoch), the four satellites transmit their positions together with a time marker. Each student is given a unique set of satellite parameters. In other words each student is in a different location.

Your Task: You are required to find out where you are. Specifically you should determine the latitude, longitude and elevation of your position. You should also be able to determine the value of the clock offset in your receiver.

Satellite Data: You are provided with a MATLAB function *Receiver.p*. The *p* file extension means that the file cannot be inspected, so you cannot reverse engineer it to find out where you are – just like real life! Here is some information regarding the use of this file:

The FUNCTION:

```
Receiver.p is a function for generating synthetic satellite data.

Usage:  SatData = Receiver(sn)

Input parameter sn is an integer made up of the last 4 digits of your
student number. This ensures with reasonably high probability that you
get data that is unique to you.

Output SatData is a 4x3 array containing information relating to
four satellites:
SatData(k,1) = Latitude of satellite k in degrees
SatData(k,2) = Longitude of satellite k in degrees
SatData(k,3) = Measured time of flight from satellite k to your receiver
               in seconds (includes unknown clock error).

CONSTANTS:
The mean radius of the earth is 6371km.
The speed of light is 299792458m/s

ASSUMPTIONS
All satellites are assumed to have identical altitudes of 20200km
above mean earth surface.
The satellite clocks are assumed to be perfectly synchronised with
each other.
```

Example

Suppose the last four digits of your student number are 1234, then at the MATLAB prompt you type:

```
>> format long
```

```
>> SatNav = Receiver(1234)
```

Which yields the result

```
SatNav =

  43.317571061717160  29.650548793445520   0.069180817014274
  32.246508366306692  22.029844311747031   0.069997328131571
  40.367259049368002 -61.976116987436995   0.071777976764458
  85.251944027582041 -19.884065541290695   0.070200113342556
```

So your first satellite is at a latitude of 43.317571061717160 degrees and a longitude of 29.650548793445520 degrees and the measured time of flight is 0.069180817014274 seconds. Note the precision displayed here is greater than would apply in practice but we won't worry about this in this exercise.

You should write code that will produce an output something like this:

```
Calculated latitude and longitude (can be copied and pasted into Google Maps):
53.8071564,-8.7027332  degrees
Calculated Elevation = 19.99 metres
Calculated receiver clock offset = -1191.52 microseconds
```

Entering the latitude and longitude information into Google Maps yields the following:

Computing Environment: The preferred environment for this exercise is MATLAB. If you really don't know MATLAB then you can use another package. However you must not use a high level equation solver. You are required to code the steps involved in implementing Newton's algorithm. So, if you are using MATLAB you should not use any toolboxes. If for

any reason you cannot run *Receiver.p* please email me at nam.tran@ucd.ie and I will send you your problem parameters. Use the subject heading "SatNav Query" and don't forget to give me the last four digits of your student number.

Solving the Equations: Referring to Fig 1, the parameters of the problem are

Sat Cartesian coordinates: $(x_k, y_k, z_k),\ k = 1,2,3,4$

Times of flight: $t_k,\ k = 1,2,3,4$

Receiver clock offset: Δt

Pseudoranges: $d_k,\ k = 1,2,3,4$

Common pseudorange error: $b = c\Delta t$

So the position of the receiver (x, y, z) can be found by solving the set of equations:

$$\sqrt{(x - x_k)^2 + (y - y_k)^2 + (z - z_k)^2} + b = d_k,\ k = 1,2,3,4$$

You may find it convenient to rewrite these equations in the following form

$$f_k(x, y, z, b) = (x - x_k)^2 + (y - y_k)^2 + (z - z_k)^2 - (d_k - b)^2 = 0, k = 1,2,3,4$$

We have four equations and four unknowns (x, y, z, b) so we should be able to find an exact solution using a multivariable version of Newton's method.

Let $U = (x, y, z, b)^T = (u_1, u_2, u_3, u_4)^T$ be the vector of unknowns

and $F = (f_1, f_2, f_3, f_4)^T$ be a vector of functions.

If U_0 is an initial guess for U, Newton's iteration is:

$$U_{n+1} = U_n - \mathcal{J}^{-1}(U_n)F(U_n)$$

Where $\mathcal{J}$ is a 4x4 Jacobian matrix with elements $\mathcal{J}_{i,j} = {\partial f_i}/{\partial u_j}$.

The iteration continues until convergence or timeout.

HINT: The computation involves repeatedly evaluating the quantity $\mathcal{J}^{-1}F$ which is the product of a matrix inverse and a vector. This type of calculation occurs frequently in optimisation problems. There is a temptation to simply invert the matrix and then multiply the result by the vector F. However, matrix inversion can be a slow process and fraught with numerical problems. But we actually don't need the matrix inverse, only the vector result $\mathcal{J}^{-1}F$. Fortunately there are ways of finding the result we want without actually inverting the matrix. MATLAB provides a very simple way of doing this in the form of the 'backslash' \ operator. If A is a square matrix, A\B is roughly the

same as inv(A)*B, except it is computed in a different way. Type *help mldivide* (or *doc mldivide*) in Matlab's command window to find out more.

Coordinate systems: You will need to convert the data provided by the receiver function from the geocentric spherical coordinate system into the geocentric Cartesian coordinate system. When your calculation is finished you should convert your position back into the spherical system i.e. latitude, longitude (both in degrees) and elevation (in metres). Fig. 2 and Fig. 3 show the Cartesian and spherical systems.

Your Report: This should contain a brief description of what you achieved. You should give the output text that your code produces as in the example above. You can, if you wish, include a map showing the location you have determined. The report should contain a listing of your code – include this as a separate file so that I can run it. Please also include your code as an appendix in your main report in case I have difficulty opening the actual code file. You should include a short discussion on your methods and results. In your discussion you might consider commenting on matters such as the rate of convergence of the algorithm and the effect of the choice of the initial condition U_0.

Submission Format: Please submit your report in PDF format and your code listing in its native format (e.g. as a MATLAB m file) to module's Brightspace. Use your name and student number as your filenames e.g. Tran12345678.pdf or Tran.m.

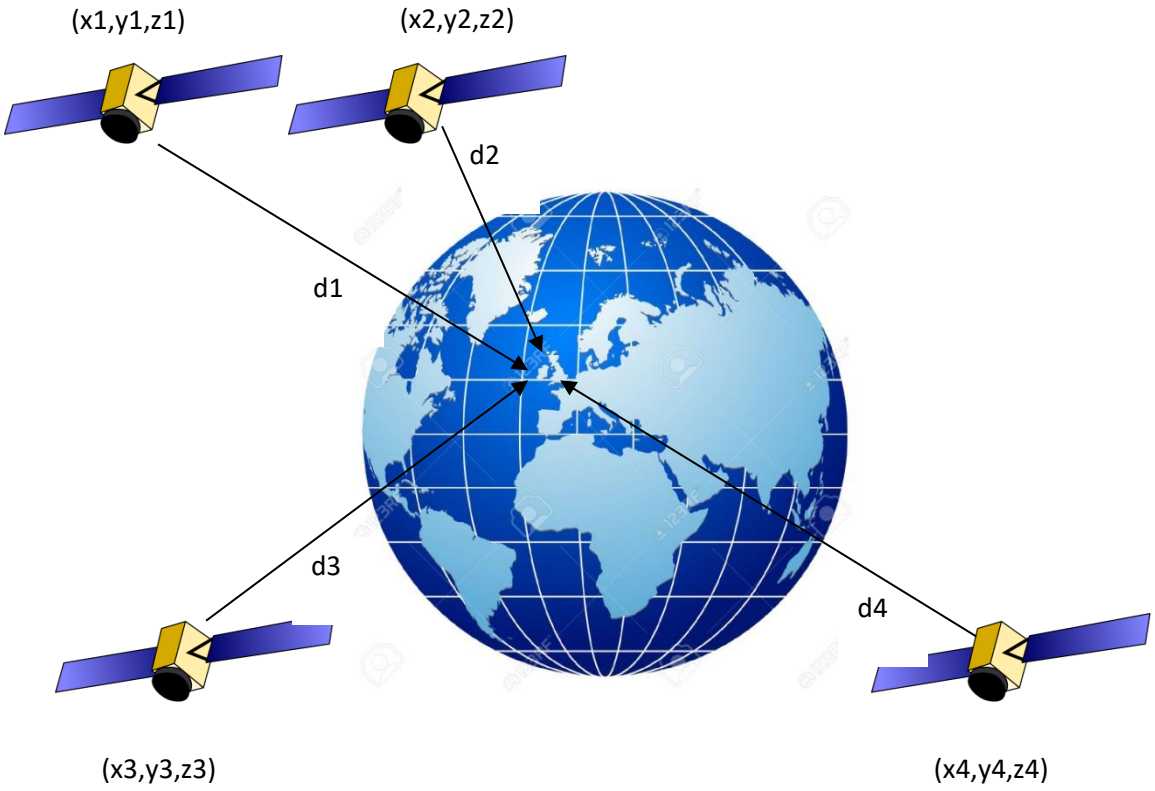

Fig. 1 The satellite configuration for the assignment.

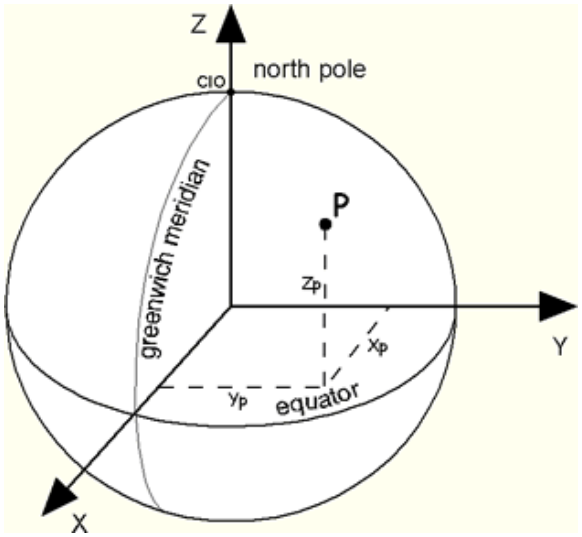

Fig.2 The Geocentric Cartesian Coordinate System.

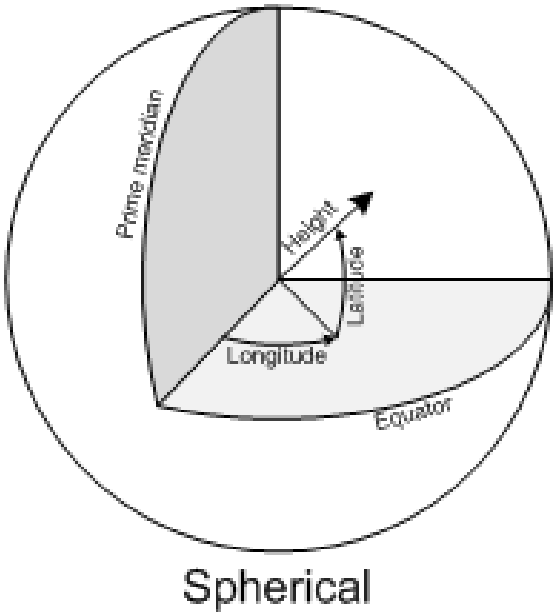

Fig. 3 The Geocentric Spherical Coordinate System.