GPS Geodesy - LAB 1

MATLAB – A brief review

Assignments for the course (and perhaps for your term project as well) are easier done in Matlab. If you do not know Matlab already, it is very easy and quick to learn. You simply need a few basics to get started, you can learn the rest as you go. In the end, Matlab will save you a lot of time compared to more formal languages such as C or Fortran. Also, Matlab is a standard software tool used in research and industry so that knowing it can only be a plus. Finally, many other courses at Purdue use Matlab.

A number of Matlab tutorials can be found on the Web, for instance:

- A Practical Introduction to Matlab, by Mark Gockenbach (http://www.cs.unb.ca/courses/cs3113/Gocken/intro.html)
- An Introduction to Matlab, by David Griffith (http://www.maths.dundee.ac.uk/~ftp/na-reports/MatlabNotes.pdf)
- Or the list of Matlab resources listed at http://www.staff.ttu.ee/~alahe/aMatlab.html

Converting dates and time:

Dates and time come in a variety of formats. Everyone is familiar with the calendar date and time in year, month, day, hours, minutes, and seconds. But also in use are:

- Julian Date (JD): number of mean solar days elapsed since January 1st, 4713 B.C., 12:00
- Modified Julian Date (MJD) = JD – 2,400,00.5
    - Ex.: GPS standard epoch, JD = 2,444,244.5 (January 6th, 1980, 00:00 UTC)
    - Ex.: Standard epoch J2000.0, JD = 2,451,545.0 (January 1st, 2000, 12:00 UTC)
- Day Of Year (DOY): day since January 1st of the current year
- GPS calendar:
    - GPS week: Week since GPS standard epoch
    - GPS day of week: Sunday = 0 to Saturday = 6
    - GPS second: Second since GPS standard epoch

Assignment:

Write a non-interactive Matlab program for converting date between different formats, allowing for the following types of input (your code should be able to recognized without prompting the user):

- Year, Month, Day
- Year, Day of year
- Decimal year
- Julian Date

Output must be:

- Year
- Month
- Day of month
- Day of year
- GPS week
- Day of GPS week
- Second of GPS week
- Julian date
- Decimal year

Usage should be something like: […] = your_code[….]

Use following formulas (*e.g.*, Hoffman-Wellenhof book):

For a given civil date (year Y, month M, day D, time UT), conversion to JD is given by:

$$JD = \text{int}(365.25\ y) + \text{int}[30.6001(m+1)] + D + UT/24 + 1\ 720\ 981.5$$

where:

$y = Y - 1$ and $m = M + 12$ if $M < 2$ or $M = 2$

$y = Y$ and $m = M$ if $M > 2$

then:

$$MJD = JD - 2\ 400\ 000.5$$

Inverse conversion, from JD to civilian date:

$a = \text{int}(JD + 0.5)$

$b = a + 1537$

$c = \text{int}[(b-122.1)/365.25]$

$d = \text{int}(365.25\ c)$

$e = \text{int}[(b-d)/30.6001]$

$D = b - d - \text{int}(30.6001\ e) + \text{frac}(JD + 0.5)$

$M = e - 1 - 12\ \text{int}(e/14)$

$Y = c - 4715 - \text{int}[(7 + M) / 10]$

Day of the week (Monday = 0):

$N = \text{modulo}\{\text{int}(JD + 0.5), 7\}$

GPS week:

$GPS\_WEEK = \text{int}[(JD - 2\ 444\ 244.5) / 7]$

Verify that:

JD = 2 444 244.5 is January 6, 1980 (GPS standard epoch)

JD = 2 451 545.0 is January 1, 2000 (Current standard epoch J2000.0)

Deliverables:

- A .m file called `gpsdate.m`

- A screen output of your code at work

Extra credit if your Matlab code is "vectorized", *i.e.*, handles vectors as input data.

It's all about rotations…:

1. Derive the equation that transforms a circle of radius *a* into an ellipse of semi-major axis *a* and semi-minor axis *b*, such that one can write:
$$X_e = RX_c$$
where Xe and Xc are the position vectors for a point on the circle and a point on the ellipse and *R* the transformation matrix. The ellipse and the circle share the same origin and *a* is collinear to the (O,X) axis.

2. Use *a*=6378.136 km and flattening *f*=1/298 for the ellipse, and "extending" it in 3D as an oblate spheroid, find the radius *r* of a sphere of identical volume as the spheroid.

3. Write a Matlab program to plot:
   - In 2D: a 2D cross-section in the (O,X,Z) plane showing, superimposed, (1) the initial circle of radius *a*, (2) the ellipse, and (3) the sphere of radius *r*.
   - In 3D: a view of the spheroid with, superimposed (e.g. using a different color or symbols) an ellipse whose plane makes a 65^o angle with the (O,X,Y) plane and intersects the (O,X,Y) plane 30^o east of the positive (O,X) axis. Experiment with the "view" command to make the figure easy to read.

Deliverables:
- The derivation for questions 1 and 2 above.
- A .m file called `sphe2ell.m`
- Three plots, as defined above, item 3.

Important note on deliverables:

- When applicable, deliverables should include the derivation of the mathematical steps implemented in your Matlab code.
- Matlab programs (.m files) must have in their header: your name and the lab number.
- Matlab programs should include as many comment lines as possible (lines starting with %).
- Plots must have their axes labeled, including units, a title that includes your name and the lab number, and, if need be (*e.g.*, several), a legend.
- This applies to all labs for this class.