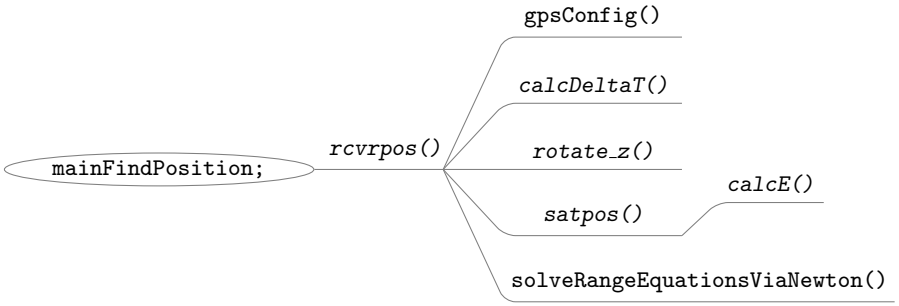
GPS: Computing the Receiver Position

Start by downloading the code framework for this assignment from Moodle[1]. The archive already contains the ephemerides and pseudoranges found as solution to the previous assignment; you will need them for this assignment.

To get an overall idea of how the program works, read through the provided code before you start writing your own code. At the same time, have a look at following graph that reflects the program structure[2]



Exercise 1. As you have seen in class, the ephemeris data transmitted in the satellites' bit stream allows you to compute the satellite position at any time (within limits). In the notes, we call t^* the receiver time at which we want to determine the position. We arbitrarily choose t^* to be the moment when we start receiving the first subframe (first among all "visible" satellites). With respect to this time, we need to know the times of flight up to a constant that we can choose, but shall be the same for all satellites. (This part was done in the previous assignment.) Finally we need to know the satellites positions at the time they emitted the signal received at receiver time t^*.

1. Complete `calcE()` that computes the eccentric anomaly $E^{(k)}$ at GPS time t. It takes the ephemeris data `ephdata` and the time t as arguments. (Step 2 from the class notes.)

2. Complete `calcDeltaT()` that returns the satellite offset $\delta t^{(k)}$ at any desired GPS time t. (Step 3 from the class notes.)

3. If we know $\delta t^{(k)}$ and $E^{(k)}$ for any GPS time t, we can determine the satellite position. Complete `satpos()`, which takes the ephemeris data `ephdata` and the GPS time t as arguments, and returns $p^{(k)}(t,t)$, the satellite position at time t in ECEF(t) coordinates. (Step 4 from the class notes.)
   Important Note: We have described ν from y_s and x_s via $\nu = \tan^{-1}\frac{y_s}{x_s}$. For this step, you have to use the `MATLAB` function `atan2(y_s,x_s)` (`mpmath.atan2(y_s,x_s)` for Python) and not `atan(y_s/x_s)`. The reason for this is that $\tan(\alpha)$ has the same value for α and $\alpha + \pi$. `atan(y_s/x_s)` will always return a value in $(-\frac{\pi}{2}, \frac{\pi}{2})$, which may or may not be the correct value. (It is the correct value iff `x_s` is positive.) `atan2(y_s,x_s)` will return the correct value in $(-\pi, \pi)$.

4. Implement `rotate_z(p, phi)` that determines the new coordinates of a point p after a rotation around the z axis by the angle `phi` (radians). You will need this function to relate the coordinates in ECEF systems frozen at different times.

5. Complete the missing parts in `rcvrpos()`.

At this point you may call `mainFindPosition`. You should obtain the following output:

[1] `gpsPosition_assignment.zip`

[2] Names ending by `()` denote functions, and names ending by `;` denote scripts. The functions in italic font are those that you will write as part of the assignment.

```
Receiver latitude : 46.518294
Receiver longitude : 6.562467
Receiver height : 484.440491
```

To find out where that is, you may use Google Maps:
`http://maps.google.com/maps?q=46.518294,6.562467`