# GPS L1 C/A SDR

University of Colorado Boulder

北京理工大學
BEIJING INSTITUTE OF TECHNOLOGY

# Introduction (1/1)

- **This version SDR of the GPS L1 C/A is based on the one in the CD attached with the book**

  Kai, B., Akos, D. M., Bertelsen, N., Rinder, P., & Jensen, S. H. (2007). A Software-Defined GPS and Galileo Receiver. Birkhäuser Boston.

- **In order to make all the SDRs have a similar structure, the original version is slightly modified.**
- **Some changes:**

  **- initSettings.m**

  **- acquisition.m**

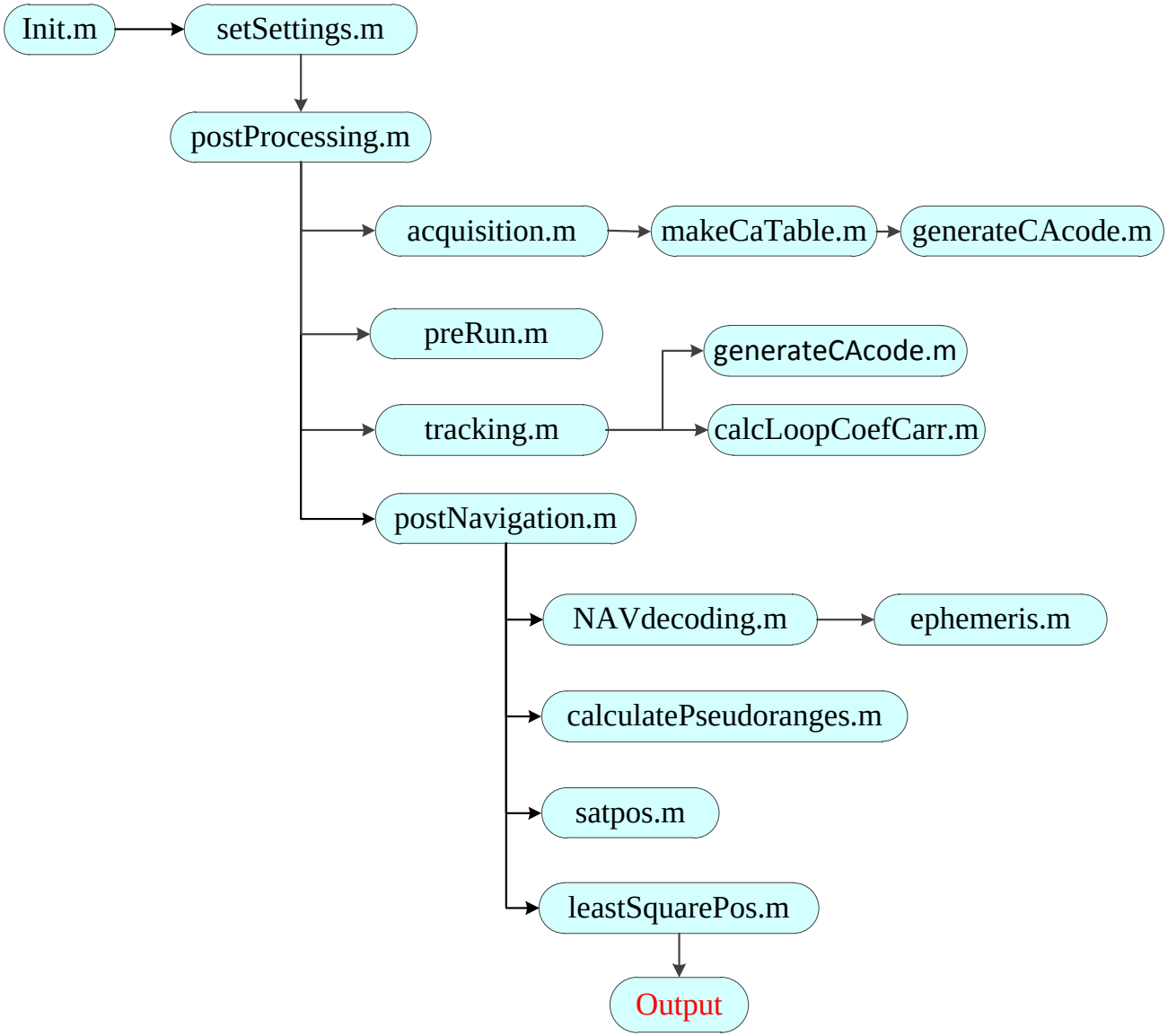
  **- tracking.m**

  **- postNavigation.m**

  **- calculatePseudoranges.m**

  **- NAVdecoding.m**

# SDR processing procedure



# initSettings.m (1/1)

## ➢ Acquisition settings

- Sampling rate threshold for downsampling
  - If input signal sampling freq. is too high for FFT computation, it can be resampled to a lower freq. to save processing time.

```
settings.resamplingThreshold = 8e6;
```

- Enable/dissable use of downsampling for acquisition

```
settings.resamplingflag  = 0;          % 0 – Off,  1 – On
```

## ➢ Tracking loops settings

- PLL and DLL integration time equals the C/A code length

```
settings.intTime = 0.001;
```

# acquisition.m (1/5)

- **Selectable downsampling**
  - (1) Filter out signal power outside the main lobe of CM code to avoid spectral folding

```
fs = settings.samplingFreq;
IF = settings.IF;
BW = 1.2e6*2;
w1 = (IF)-BW/2;
w2 = (IF)+BW/2;
wp = [w1*2/fs-0.002 w2*2/fs+0.002];
b  = fir1(700,wp);
longSignal = filtfilt(b,1,longSignal);
```

# acquisition.m (2/5)

## ➢ Selectable downsampling

- (2) Calculate upper and lower boundary of the acceptable sampling Freq. range:

```
fu = settings.IF + BW/2;
n = floor(fu/BW);
if (n<1)
   n = 1;
end
lowerFreq = 2*fu/n;
fl = settings.IF - BW/2;
if(n>1)
   upperFreq = 2*fl/(n-1);
else
   upperFreq = lowerFreq;
end
```

# acquisition.m (3/5)

- **Selectable downsampling**

(3) Find downsampling frequency

- Take the center of the acceptable sampling Freq. range as downsampling frequency.

```
settings.samplingFreq = ceil((lowerFreq + upperFreq)/2);
```

(4) Downsample input IF signals

```
signalLen = floor((length(longSignal)-1) /oldFreq * settings.samplingFreq);
index = ceil((0:signalLen-1)/settings.samplingFreq * oldFreq);
index(1) = 1;
longSignal = longSignal(index);
```

# acquisition.m (4/5)

- **Selectable downsampling**

  (5) Equivalent IF after downsampling

  - Resampling is equivalent to down-converting the original IF by integer times of resampling freq.
  - **settings.IF = rem(settings.IF,settings.samplingFreq);**

# acquisition.m (5/5)

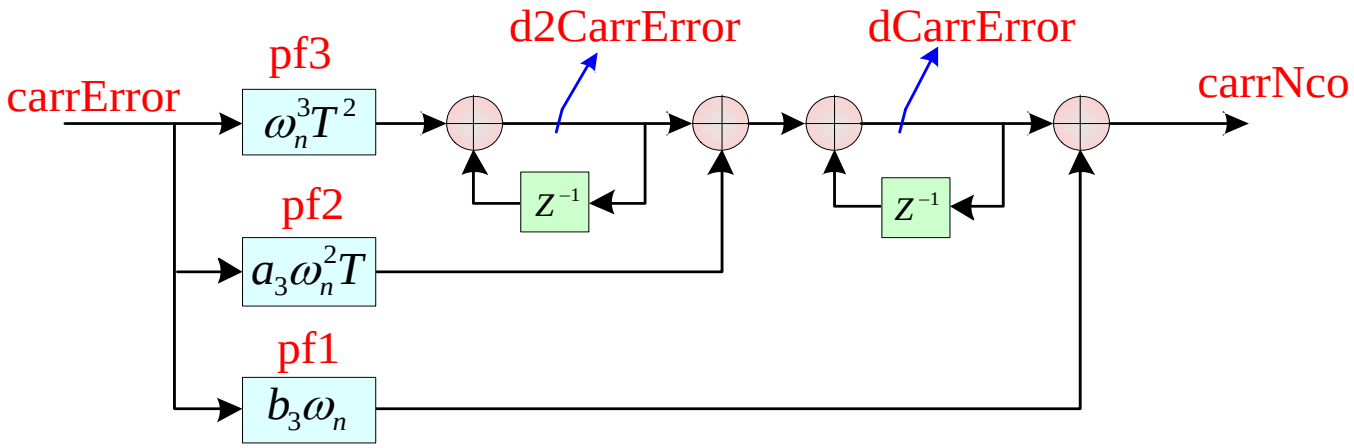
- **Acquisition implementation**

  **- Change fine acquisition results to correspond with original sampling frequency, if down-sampling is carried out.**

```
acqResults.codePhase(PRN) = floor((codePhase - 1)/ settings.samplingFreq * oldFreq)+1;
if (settings.IF >= settings.samplingFreq/2)
    IF_temp = settings.samplingFreq - settings.IF;
   doppler = IF_temp - acqResults.carrFreq(PRN);
else
  doppler = acqResults.carrFreq(PRN) - settings.IF;
end
```

# tracking.m (1/2)

- **A 3-order PLL was used to improve dynamic performance of carrier tracking**



# tracking.m (2/2)

- **PLL filter coefficients calculation**
  - `[pf3,pf2,pf1] = calcLoopCoefCarr(settings);`
- **NCO update**
  - `d2CarrError = d2CarrError + carrError * pf3;`
  - `dCarrError = d2CarrError + carrError * pf2 + dCarrError;`
  - `carrNco = dCarrError + carrError * pf1;`
- **Save remCodePhase for pseudorange calculation**
  - `trackResults(channelNr).remCodePhase(loopCnt) = remCodePhase;`

# NAVdecoding.m (1/1)

- **Bit and frame synchronization**

  **- Same as the original 'findPreambles.m'**

- **Ephemeris decoding**

  **- Prepare data for ephemeris decoding**

  - `navBitsSamples = I_P_InputBits(subFrameStart - 20 : subFrameStart + (1500 * 20) -1)';`
  - `navBitsSamples = reshape(navBitsSamples, 20, (size(navBitsSamples, 1) / 20));`
  - `navBits = sum(navBitsSamples);`
  - `navBits = (navBits > 0);`
  - `navBitsBin = dec2bin(navBits);`

  **- Decoding ephemeris**

  - `[eph, TOW] = ephemeris(navBitsBin(2:1501)', navBitsBin(1));`

# postNavigation.m (1/1)

- **Main changes**

  **- Start and end of measurement points**

  - **Measurement point begins at the latest start of first messages of all channels (firstSubFrame), and consider Doppler smoothing**

```
for channelNr = activeChnList
   sampleStart(channelNr) = ...
      trackResults(channelNr).absoluteSample(firstSubFrame(channelNr));
   sampleEnd(channelNr) = trackResults(channelNr).absoluteSample(end);
End
…
```

**- Correct and save local time**

```
localTime = localTime - xyzdt(4)/settings.c;
navSolutions.localTime(currMeasNr) = localTime;
```

**- Update local time**

- **Except first time, local time is updated by measurement step**

```
localTime = localTime + measSampleStep/settings.samplingFreq ;
```

# calculatePseudoranges.m (1/2)

- **Signal transmitting time of measurement points**

  **- Use method of hardware receiver**

  **- More straightforward and more accurate**

$$t_t = TOW + \left(\frac{codePhase}{code\ length} + integar\ number\ of\ C/A\ code\right) * 0.001$$

```
codePhase     = trackResults(channelNr).remCodePhase(index) +  ...
                codePhaseStep * (currMeasSample - ...
                trackResults(channelNr).absoluteSample(index) );
transmitTime(channelNr) =  (codePhase/settings.codeLength + index - ...
                subFrameStart(channelNr)) * settings.codeLength/...
                settings.codeFreqBasis + TOW(channelNr);
```

- **Initialize signal receiving time (local time) $t_r$**

  **- Use transmitting time and settings.startOffset**

```
maxTime   = max(transmitTime(channelList));
localTime = maxTime + settings.startOffset/1000;
```

# calculatePseudoranges.m (2/2)

- **Calculate pseudorange (PR)**

$$PR = (t_r - t_t) * c$$

**where, $c$ - light speed**

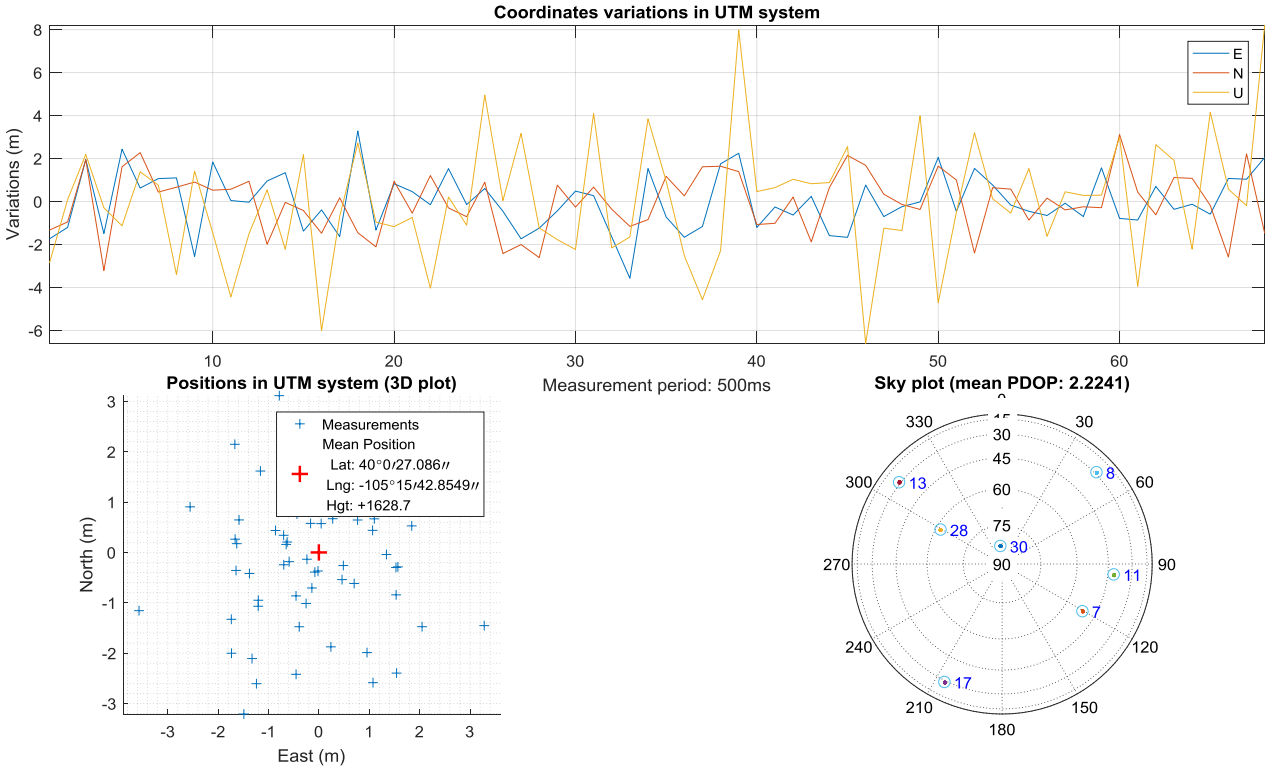
- pseudoranges = (localTime - transmitTime) * settings.c;

# leastSquarePos.m (1/2)

- **Receiver position calculation**

  **- Not changed**

- **Fix results**




*** The IF signal is collected on the roof of the Discovery Learning Center, CU-Boulder, USA.**

# leastSquarePos.m (2/2)

- **Fix results**

# References


- Xie Gang. Principles of GNSS: GPS, Glonass and Galileo. Publishing house of electronics industry, Beijing, Sep. , 2013.
- Kai, B., Akos, D. M., Bertelsen, N., Rinder, P., & Jensen, S. H.. A Software-Defined GPS and Galileo Receiver. Birkhäuser Boston, 2007.

# Thank you!


**Dennis Akos dma@colorado.edu**

**Nagaraj C S nagarajcs@colorado.edu**

**Yafeng Li lyf8118@126.com**



University of Colorado Boulder

北京理工大學 BEIJING INSTITUTE OF TECHNOLOGY