GPU HACKATHON

#gpucesgahack

29 mayo - 1 junio
Santiago de Compostela

TEAM 2

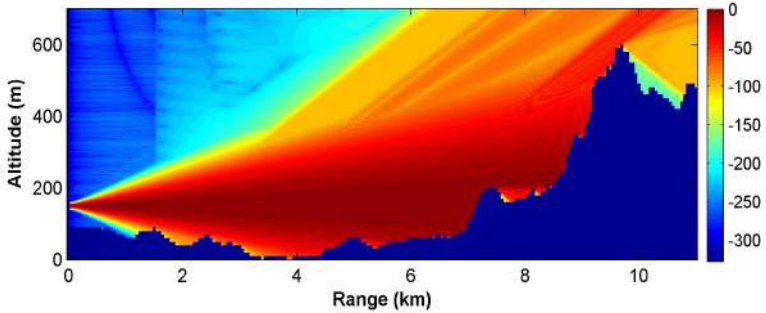
Rubén Nocelo López

Radio Systems
UNIVERSITY OF VIGO

GPU HACKATHON

Motivation

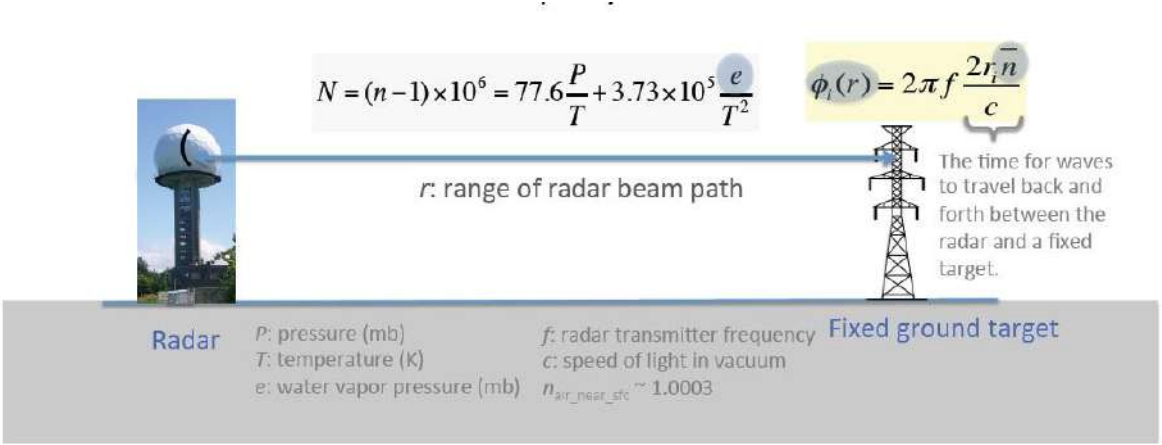
- Knowledge with enough spatial and temporal resolution of the near surface refractivity (N).
  - Forecast convective initiation and boundary layer processes.
  - Model of the radio wave propagation (Parabolic Equation).



GPU HACKATHON

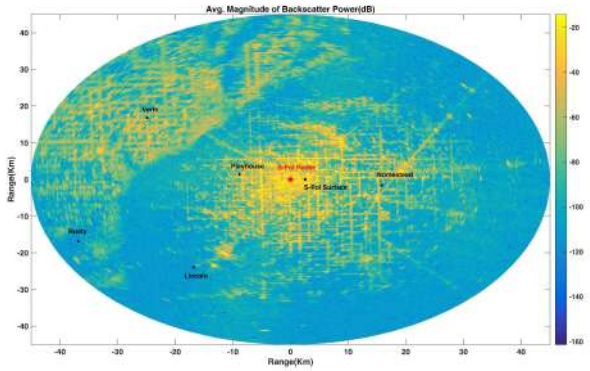
Background

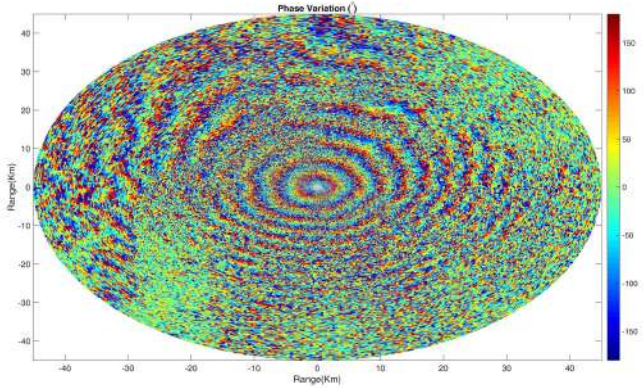
- Atmospheric refractivity can be obtained from radar phase measurements (Fabry).



Measurements

- Full azimuthal scans on the order of 0.01 degrees and 150 m in range.
- Several elevation angle scans





Algorithm

$$\Delta\phi(\Delta R,\Delta t)=\Delta\phi(R_1,\Delta t)-\Delta\phi(R_0,\Delta t)=[\phi(R_1,t_1)-\phi(R_1,t_0)]-[\phi(R_0,t_1)-\phi(R_0,t_0)]=\frac{4\pi f}{c}\times$$

$$\left\{\begin{array}{l}
\underbrace{\underbrace{\Delta\bar{n}(h_R,\Delta R,\Delta t)(R_{t_0}^{T_1}-R_{t_0}^{T_0})}_{f_I(\Delta R,\Delta t)}+\underbrace{\bar{n}(h_R,t_1)\,[\Delta R(T_1)-\Delta R(T_0)]}_{f_{II}(\Delta R,\Delta t,\bar{n}(t_1))}}_{\Delta\phi_H(\Delta R,\Delta t,\bar{n}(t_1))\ \equiv\ \textit{Horizontal Term}} \\
\underbrace{+\underbrace{\Delta\left[\frac{\partial n(\Delta t)}{\partial h}\right]\left[\frac{(h_{T1}-h_R)}{2}R_{t_0}^{T_1}-\frac{(h_{T0}-h_R)}{2}R_{t_0}^{T_0}\right]}_{g_I(\Delta R,\Delta t)}+\underbrace{\frac{\partial n(t_1)}{\partial h}\left(\frac{(h_{T1}-h_R)}{2}\Delta R(T_1)-\frac{(h_{T0}-h_R)}{2}\Delta R(T_0)\right)}_{g_{II}(\Delta R,\Delta t,\frac{\partial n(t_1)}{\partial h})}}_{\Delta\phi_V(\Delta R,\Delta t,\frac{\partial n(t_1)}{\partial h})\ \equiv\ \textit{Vertical Gradient Term}} \\
\underbrace{\underbrace{-\Delta\left[\frac{\partial n(\Delta t)}{\partial h}\frac{1}{12a_e(\Delta t)}\right](R'_{T_1}-R'_{T_0})}_{h_I(\Delta R,\Delta t)}\underbrace{-(R''_{T_1}-R''_{T_0})\frac{\partial n(t_1)}{\partial h}\frac{1}{12a_e(t_1)}}_{h_{II}(\Delta R,\Delta t,\frac{\partial n(t_1)}{\partial h})}}_{\Delta\phi_C(\Delta R,\Delta t,\frac{\partial n(t_1)}{\partial h})\ \equiv\ \textit{Earth's Curvature Term}}
\end{array}\right.$$

GPU HACKATHON

Real time measurements

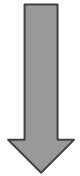
- Time between two consecutive scans: < 5 min.
- Samples per scan
  - Range: 250km/150m=1660 gates
  - Dual polarization (H/V)
  - I/Q data
  - Angles: 36000 azimuths (each 0.01 degree)
- Size: 236 MB

App

- **Receive Radar Data (FILE)**
- **Process headers and binary data**

MAIN 1

- **Obtain I/Q data, elevation and azimuth angles**



- Derive power and phase.
- Estimate Refractivity.

MAIN 2

GPU HACKATHON

Receive Radar Data (File) & Process headers and binary data

```
int main(){

———    Read_header_main

        while(!=(eof(fich_radar))){

            Read_header_data(Azimuth, Elevation,length(IQ));

            Read_data_binary

            Write(fich_azi,azimuth)

            Write(fich_elev,elevation)

            Write(fich_IQ,IQ)

        }

    }
```

1 fich_in => 3 fich_out
Time = 214 sg.

GPU HACKATHON

Decode I/Q data, elevation and azimuth angles

```
int main()
___
    {
        function(process_azi)   => 1%
        function(process_ele)   => 1%
        function(process_IQ)    => 98%
    }
```

Decode I/Q data, elevation and azimuth angles

```
Function Process_IQ

{---

    while(!=(eof(fich)))

        {

            (Unsigned int) data

            Read (fich,data) => 1 to 1

            Process(data)

            Write(fich,data_process)

        }

}
```

3 fich_in => 3 fich_out

Time = 45’

Time = 360 sg.

GPU HACKATHON

Decode I/Q data, elevation and azimuth angles

```
    Function Process_IQ
{
        (Unsigned int) Matrix_in[azimuths][range]
        Read (fich,Matrix_in) => Matrix_in[azimuths][range]
        Process_Matrix_in
        Write(fich,Matrix_out) => Matrix_out[azimuths][range]
}
```

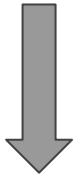
3 fich_in => 3 fich_out
Time = 200 sg.

App

- **Receive Radar Data**
- **Process headers and binary data**

MAIN 1 (214 sg)

- **Obtain I/Q data, elevation and azimuth angles**

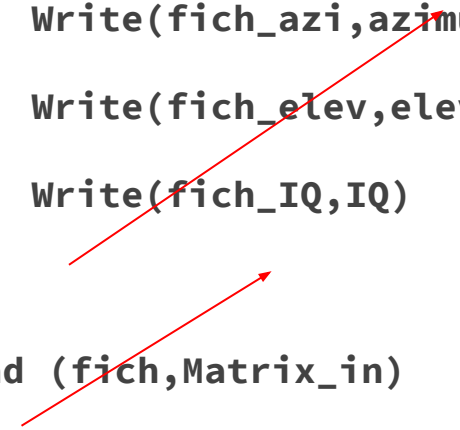


- Derive power and phase.
- Estimate Refractivity.

MAIN 2 (200 sg)

Process headers and binary data & Decode Data

```
int main(){

        Read header_main

        while(!=(eof(fich_radar))){

            Read header_data(Azimuth, Elevation,length(IQ));

            Read_data_binary

            Write(fich_azi,azimuth)

            Write(fich_elev,elevation)

            Write(fich_IQ,IQ)

        }

        Read (fich,Matrix_in)

        Process_Matrix_IQ

        Write(fich_Matrix_out) => Matrix_Voltage
```



1 fich_in => 3 fich_out
Time= 143 sg

Process_Matrix_In (Fich I/Q)

```
Function Process_Matrix_In{

for(ind1=0;ind1<l_fila;ind1++) => Same Azimuth and Elevations

     {

          for(ind2=0;ind2<l_columna_v;ind2++) => Range(10000 elements)

          {

               (float) Temp= matrix[ind1][ind2]

               If (conditions){

               Temp=decode1}

               Else{

               Temp=decode2}

               Voltage[ind1][ind2]=Temp;

}
```

ANALYSIS (DRAFT)

1. Algorithmic features
2. Parallel design patterns
3. Implementation features

Algorithmic features

- Spatial discretization? NO
  - Finite differences?
  - Finite elements?
  - Finite volumes?
  - Type of elements: triangle, square?
- Type of solver? NO
  - Direct vs Iterative (convergence criteria)?
  - Adaptive discretization of the domain?
  - Sparse/Irregular computations (e.g. sparse matrices -CRS-)?
  - Based on solving systems of linear equations?
- Compute-bounded? NO
- Memory-bounded? NO

Parallel design patterns

- Fully parallel loops? YES
- Parallel scalar reductions?
- Parallel sparse reductions?

Implementation features

- Shape of the data structure?
  - Pointer-based vs Array-based?
  - Recursive data structs (e.g. linked-list, tree)? NO
  - Array of structs (AoS) vs Struct of Arrays (AoS)? NO

Process_Matrix_In (Fich I/Q) =>Parallel Pattern

```
Function Process_Matrix_In{

#pragma omp parallel shared(matrix,voltage) default(none)

{

#pragma omp for

for(ind1=0;ind1<l_fila;ind1++) => Same Azimuth and Elevations

        {

                for(ind2=0;ind2<l_columna_v;ind2++) => Range(10000 elements)

                {

                        (float) Temp= matrix[ind1][ind2]

                        If (conditions){ Temp=decode1}

                        Else{Temp=decode2}

                        Voltage[ind1][ind2]=Temp;

                }

        }

}
```

Performance on Finisterrae (OpenMP on CPU)

Threads (#)	Time (s)
1	0.534
2	0.255
4	0.127
8	0.068

TODO List (2 weeks until Friday 23/June/2017)

- xxx

GPU HACKATHON