



"Cyclone" Multi-Doppler Tutorial

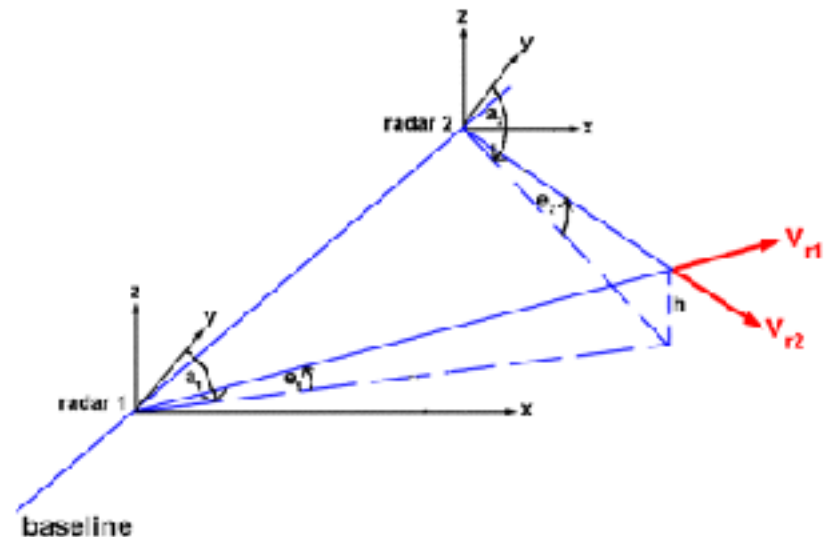
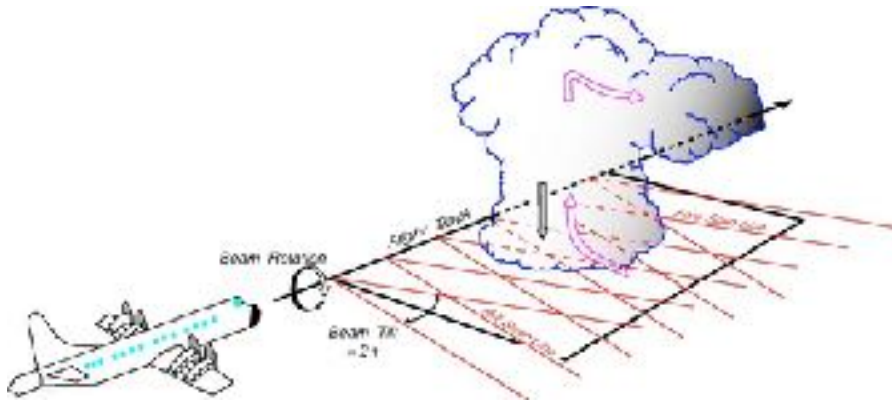
FRACTL, SAMURAI

Michael Bell

Multi-Doppler Synthesis

$$V_r = u \sin(a) \cos(e) + v \cos(a) \cos(e) + (w + w_t) \sin(e)$$

Desired minimum angle separation of $\sim 30^\circ$ away from 'baseline' for dual Doppler, but wind dependent

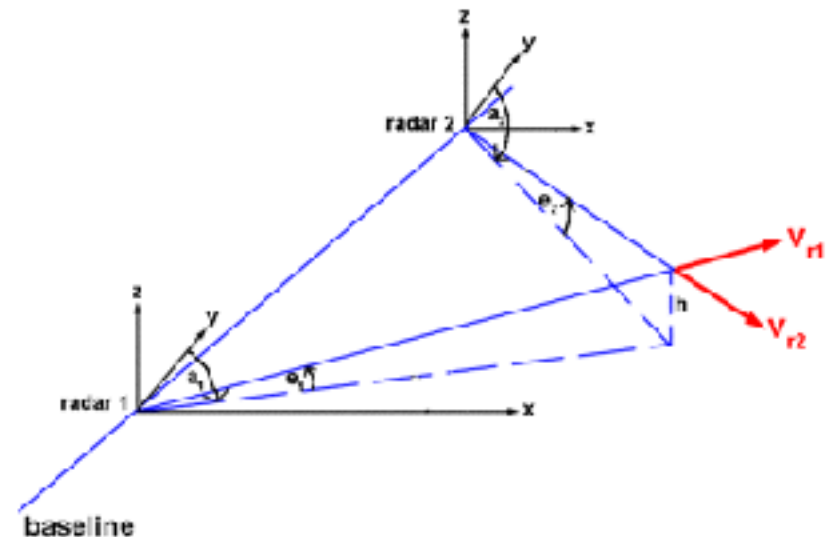
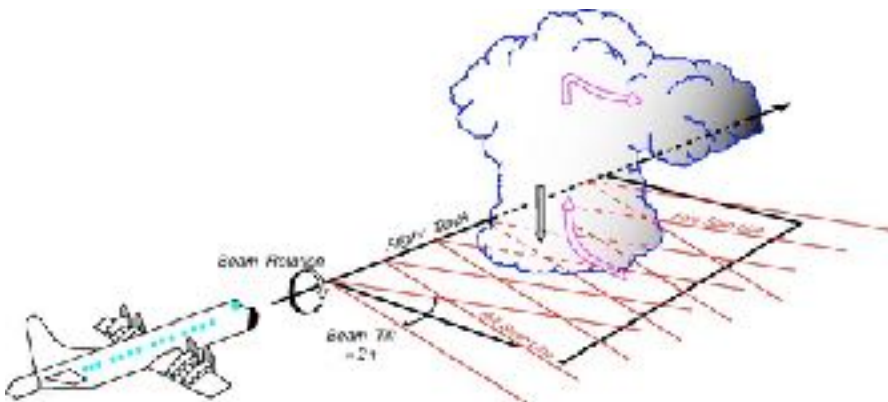


Multi-Doppler Synthesis

$$V_r = u \sin(a) \cos(e) + v \cos(a) \cos(e) + (w + w_t) \sin(e)$$

$$\begin{pmatrix} \sin(a_1) & \cos(a_1) \\ \sin(a_2) & \cos(a_2) \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} V_{r1} \\ V_{r2} \end{pmatrix}$$

$$-\frac{\partial \rho w}{\partial z} = \frac{\partial \rho u}{\partial x} + \frac{\partial \rho v}{\partial y}$$



Multi-Doppler Synthesis

$$V_r = u \sin(a) \cos(e) + v \cos(a) \cos(e) + (w + w_t) \sin(e)$$

$$\begin{pmatrix} \sin(a_1) & \cos(a_1) \\ \sin(a_2) & \cos(a_2) \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} V_{r1} \\ V_{r2} \end{pmatrix} \quad -\frac{\partial \rho w}{\partial z} = \frac{\partial \rho u}{\partial x} + \frac{\partial \rho v}{\partial y}$$

$$Ax = b$$

the solutions are

$$\begin{aligned} \hat{u} &= \frac{D_1 B_2 - D_2 B_1}{D} + \hat{W} \frac{B_1 C_2 - B_2 C_1}{D} = u' + \epsilon_u \hat{W} \\ \hat{v} &= \frac{D_2 A_1 - D_1 A_2}{D} + \hat{W} \frac{A_2 C_1 - A_1 C_2}{D} = v' + \epsilon_v \hat{W}, \end{aligned}$$

where the determinant of coefficients

$$D = A_1 B_2 - A_2 B_1.$$

Multi-Doppler Synthesis

$$V_r = u \sin(a) \cos(e) + v \cos(a) \cos(e) + (w + w_t) \sin(e)$$

$$\begin{pmatrix} \sin(a_1) & \cos(a_1) \\ \sin(a_2) & \cos(a_2) \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} V_{r1} \\ V_{r2} \end{pmatrix} \quad -\frac{\partial \rho w}{\partial z} = \frac{\partial \rho u}{\partial x} + \frac{\partial \rho v}{\partial y}$$

$$Ax = b$$

$$\|r\|^2 = \|b - Ax\|^2$$

Multi-Doppler Synthesis

$$V_r = u \sin(a) \cos(e) + v \cos(a) \cos(e) + (w + w_t) \sin(e)$$

Three (or more) radar solution ('Normal' equations)

$$\begin{pmatrix} \sum \sin^2 a \cos^2 e & \sum \sin a \cos a \cos^2 e & \sum \sin a \cos e \sin e \\ \sum \sin a \cos a \cos^2 e & \sum \cos^2 a \cos^2 e & \sum \cos a \cos e \sin e \\ \sum \sin a \cos e \sin e & \sum \cos a \cos e \sin e & \sum \sin^2 e \end{pmatrix} \begin{pmatrix} u \\ v \\ w + w_t \end{pmatrix} = \begin{pmatrix} \sum V_r \sin a \cos e \\ \sum V_r \cos a \cos e \\ \sum V_r \sin e \end{pmatrix}$$

$$\textbf{Local?} \quad \|r\|^2 = \|b - Ax\|^2 \quad \textbf{Global?}$$

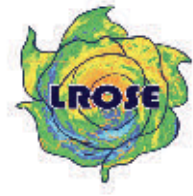
Mini-workshop tutorial will only cover practical application due to limited time, but will be covered in much more detail in pending documentation and in fall workshop

Different analysis techniques have advantages / disadvantages

| Traditional “Local” solver (FRACTL , CEDRIC) | Variational “Global” solver (SAMURAI , HRD, Multi-Dopp) |
|--|---|
| Fast computation | Computationally demanding |
| Point-by-point error diagnostics | Global error diagnostics |
| Doppler data only | Other data sources possible |

- **FRACTL (Fast Reorder and CEDRIC Technique in LROSE)**
- **SAMURAI (Spline Analysis at Mesoscale Utilizing Radar and Airborne Instrumentation)**

To Run FRACTL



To see all command line options:

```
fractl -h
```

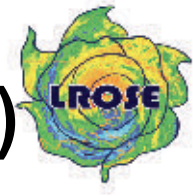
To create a default parameter file, use the -print_params command line option:

```
fractl -print_params > my_fractl.params
```

To run the application, invoke the -params command line option:

```
fractl -params my_fractl.params
```


To Run SAMURAI (new LROSE version)



To see all command line options:

```
samurai -h
```

To create a default parameter file, use the -print_params command line option:

```
samurai -print_params > my_samurai.params
```

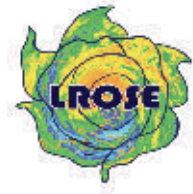
To run the application, invoke the -params command line option:

```
samurai -params my_samurai.params
```

Older XML format is backward compatible

Mandatory

Things to change in FRACTL params



Lat/Lon Origin

```
< projLat0 = 0;  
---  
> projLat0 = 29.4719;  
265c265  
< projLon0 = 0;  
---  
> projLon0 = -94.8787;
```

Input/Output

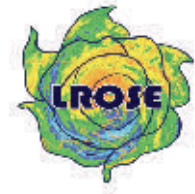
```
453c453  
< inDir = "not_set";  
---  
> inDir = "input";  
485c485  
< outTxt = "not_set";  
---  
> outTxt = "fractl_verif.txt";
```

Variable names

```
512c512  
< radialName = "not_set";  
---  
> radialName = "VEL";  
522c522  
< dbzName = "not_set";  
---  
> dbzName = "REF";  
532c532  
< ncpName = "not_set";  
---  
> ncpName = "SW";
```

FRACTL can use for NCP field to do basic, automatic QC, but if you have already QCed data then you can ignore it and set it to any field

Optional



Things to change in FRACTL params

Grid dimensions

```
190c190
< zGrid = "0.5";
---
> zGrid = "0.0,16.0,1.0";
200c200
< yGrid = "1.0";
---
> yGrid = "-100.0,100.0,1.0";
210c210
< xGrid = "1.0";
---
> xGrid = "-100.0,100.0,1.0";
```

If increment only is given, FRACTL will automatically determine min/max dimensions for the radar data supplied

Optional

Things to change in FRACTL params



Condition Number Cutoff

```
// Maximum value for a cell ConditionNumber.  
//  
//  
// Type: double  
//  
conditionNumberCutoff = 100;
```

Lower values have higher quality winds

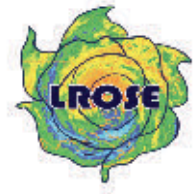
Leise filtering

```
549c549  
< uvFilter = FILTER_NONE;  
---  
> uvFilter = FILTER_LEISE;  
560c560  
< wFilter = FILTER_NONE;  
---  
> wFilter = FILTER_LEISE;  
571c571  
< uvSteps = 1;  
---  
> uvSteps = 2;
```

Leise filter is the same one from CEDRIC

Mandatory

Things to change in SAMURAI params



Use background

```
107c107
< load_background =
TRUE;
---
> load_background =
FALSE;
```

Input/Output

```
197c197
< data_directory = "not_set";
---
> data_directory = "input";
209c209
< output_directory = "not_set";
---
> output_directory = "output";
```

Variable names

```
464c464
< radar_dbz = "DBZ";
---
> radar_dbz = "REF";
476c476
< radar_vel = "VG";
---
> radar_vel = "VEL";
```

Also need to set SW
field, which in this case
is the same as default

Mandatory

Things to change in SAMURAI params



Grid dimensions

```
302c302
< i_min = 0;
---
> i_min = -100.0;
310c310
< i_max = 0;
---
> i_max = 100.0;
318c318
< i_incr = 0;
---
> i_incr = 1.0;
```

```
326c326
< j_min = 0;
---
> j_min = -100.0;
334c334
< j_max = 0;
---
> j_max = 100.0;
342c342
< j_incr = 0;
---
> j_incr = 1.0;
```

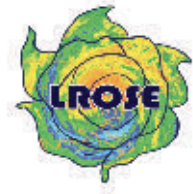
```
358c358
< k_max = 0;
---
> k_max = 16.0;
366c366
< k_incr = 0;
---
> k_incr = 1.0;
```

If you want lat/lon or pressure output, set to desired increment, Otherwise set to -1 (will be default in next release)

```
1316c1316
< output_latlon_increment = 1;
---
> output_latlon_increment = -1;
1324c1324
< output_pressure_increment = 1;
---
> output_pressure_increment = -1;
```

Mandatory

Things to change in SAMURAI



Reference Time In param file

```
392c392  
< ref_time = "not_set";  
---  
> ref_time = "15:00:00";
```

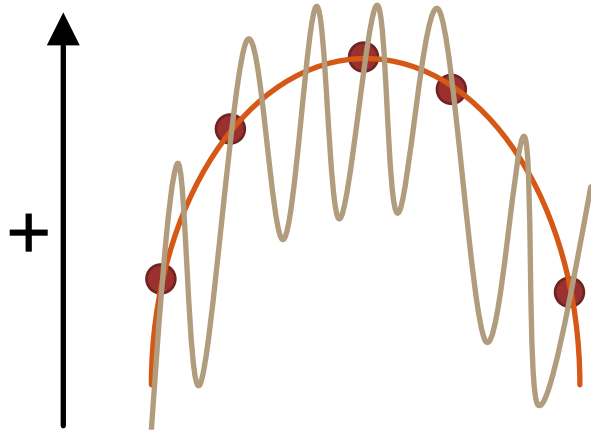
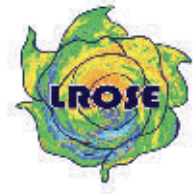
Reference Center file YYYYMMDD.cen

```
140001 29.4719 -94.8787 0 0  
140002 29.4719 -94.8787 0 0  
140003 29.4719 -94.8787 0 0  
140004 29.4719 -94.8787 0 0  
140005 29.4719 -94.8787 0 0  
140006 29.4719 -94.8787 0 0  
140007 29.4719 -94.8787 0 0  
...
```

Can generate reference center file with provided Perl script
([samurai lineartrack.pl](#))

Optional

Things to change in SAMURAI



- Multiple spline interpolations are exact solutions to a set of points depending on how many knots you allow the analysis to have
- SAMURAI uses a 3D variational technique to fit splines to the data, but the scales resolved by the analysis have to be set by the user
- Low-pass filters are used to control the resolved scales and avoid overfitting with too many knots
 - Both the orange and brown curves are valid solutions in the example to the left, but the orange one better represents the scales resolved by this data distribution

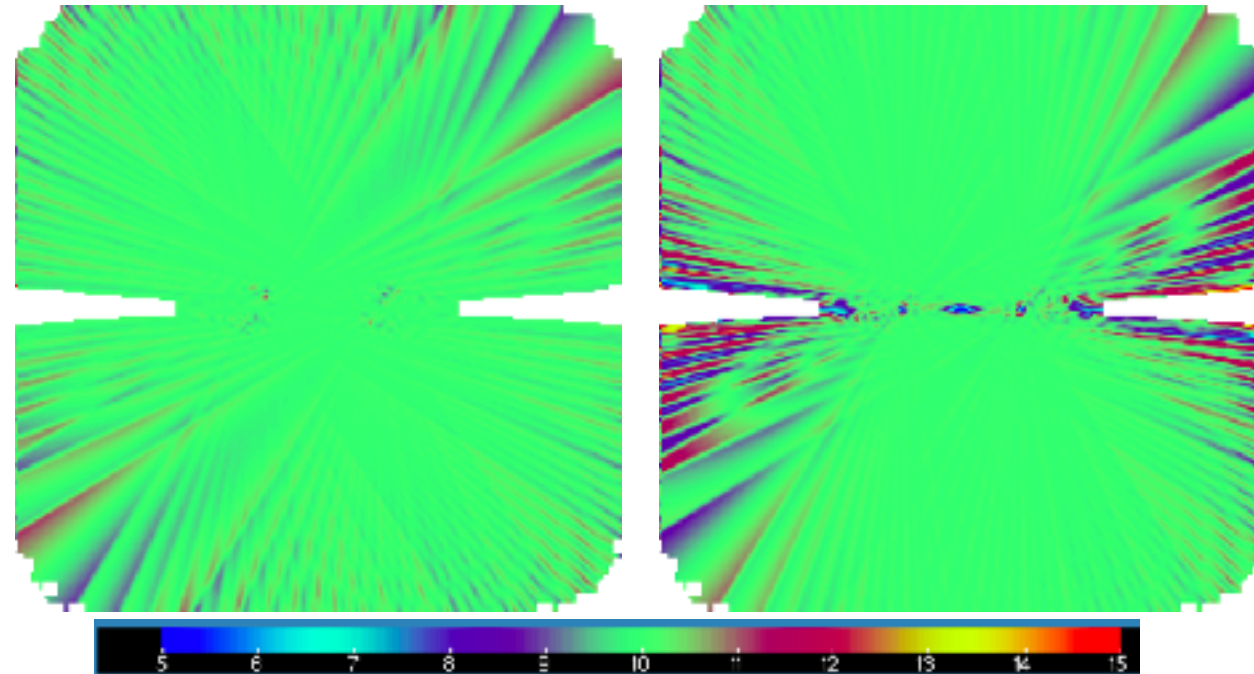
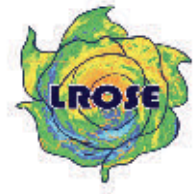
Gaussian and Spline filter lengths
are set in grid units (eg. 2x)

1570c1570
< 4,

> 2,

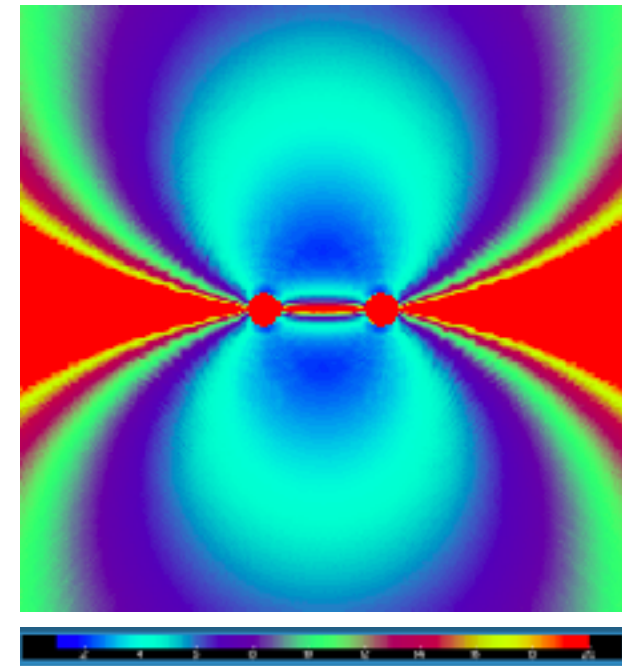
More detailed discussion of filtering beyond scope of this tutorial, but being written up

Expected FRACTL output (using ncview)



U

V

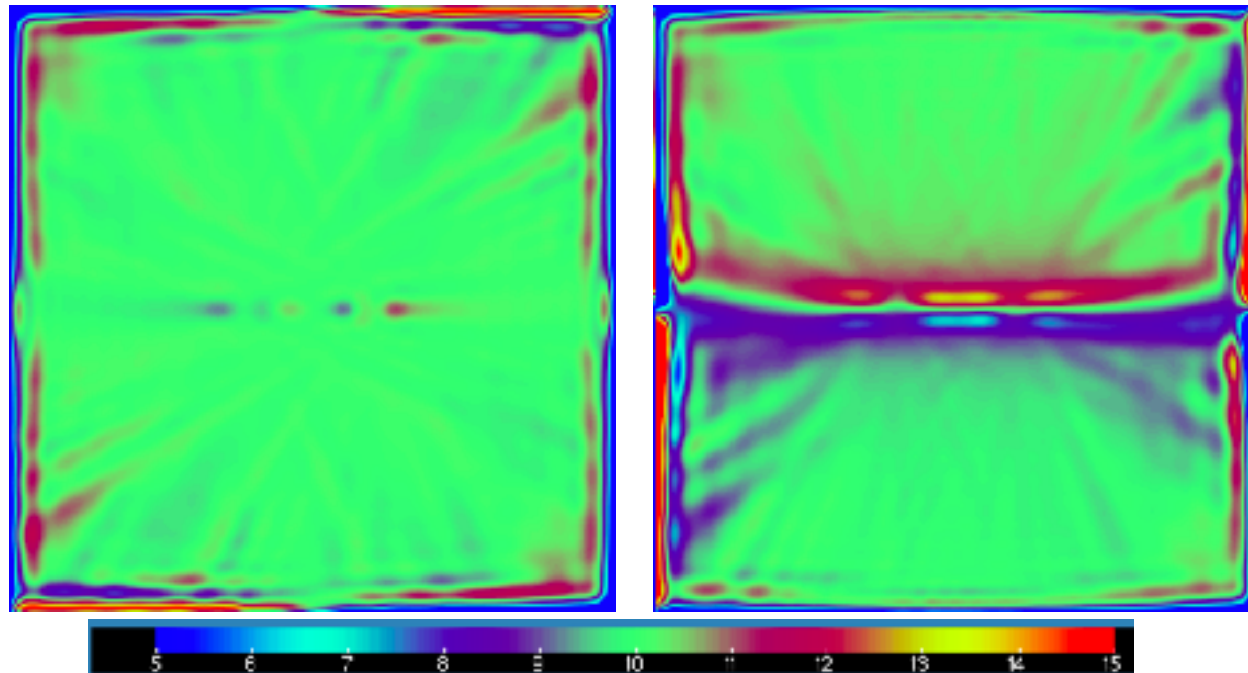
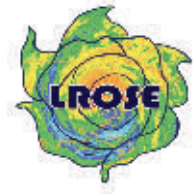


Condition
Number

Analytic solution is $U=10$, $V=10$ m/s

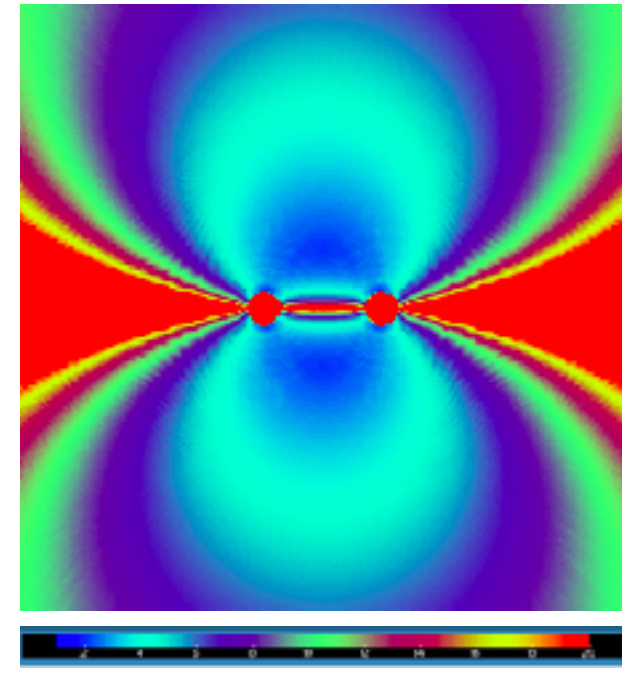
Retrieved solution is within 1-2 m/s where condition number is small

Expected SAMURAI output (using ncview)



U

V



Condition Number
(from FRACTL)

Analytic solution is $U=10$, $V=10$ m/s

Retrieved solution is within 1-2 m/s where condition number is small

Current Status and Next Steps

- Basic documentation available, more being written
- Subset of CEDRIC functionality available in FRACTL, more to be implemented
- FRACTL/SAMURAI integration and Radx2Grid integration are in experimental stage
- Continued optimization to make faster
 - CISL mods will lead to 3-4x speed-up for SAMURAI
 - FRACTL fast but still single-core, will add parallel features for even faster performance