# Matlab based toolkits used to interface with optical design software for NASA's James Webb Space Telescope

Joe Howard
JWST OTE Lead Optical Designer
NASA - Goddard Space Flight Center

*2007 SPIE Optics and Photonics, San Diego, August 28*

NORTHROP GRUMMAN
Space Technology
Ball ITT Industries Engineered for life ATK

# Acknowledgments

Special thanks to

Blair Unger (University of Rochester)

and

Mark Wilson (NASA)

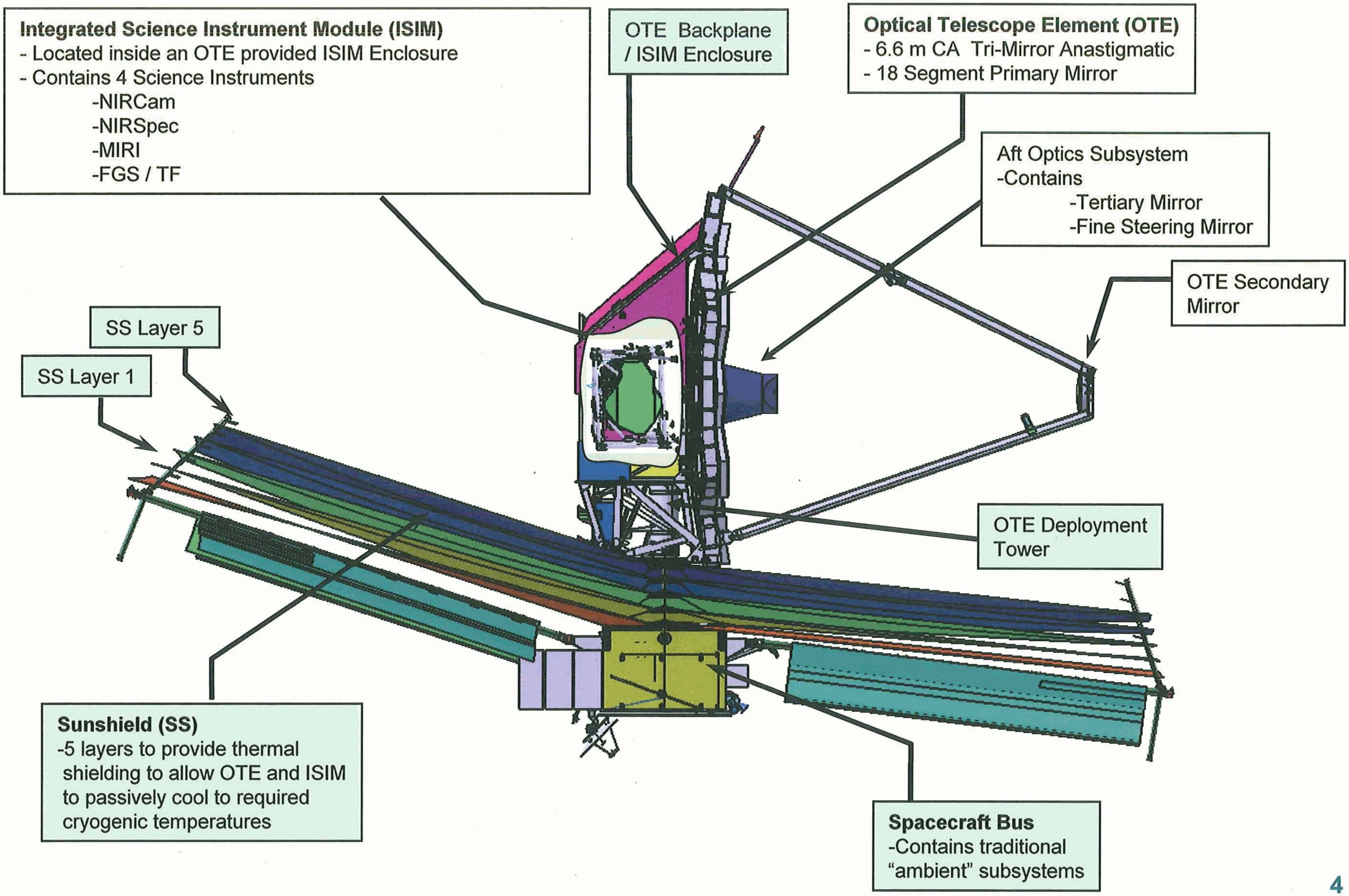
for their assistance in creating the

Matlab-CodeV Toolkit

# Outline

1. Introduction to JWST
2. Brief overview of Matlab toolkits
   - CodeV Toolkit
   - OSLO Toolkit
   - Zemax Toolkit
3. Examples of use with JWST
   - Wavefront sensitivities
   - Alignment simulations
4. Where to get them
5. Concluding remarks

# James Webb Space Telescope (JWST)



**Integrated Science Instrument Module (ISIM)**
- Located inside an OTE provided ISIM Enclosure
- Contains 4 Science Instruments
  - -NIRCam
  - -NIRSpec
  - -MIRI
  - -FGS / TF

OTE Backplane / ISIM Enclosure

**Optical Telescope Element (OTE)**
- 6.6 m CA Tri-Mirror Anastigmatic
- 18 Segment Primary Mirror

Aft Optics Subsystem
-Contains
  -Tertiary Mirror
  -Fine Steering Mirror

OTE Secondary Mirror

SS Layer 5

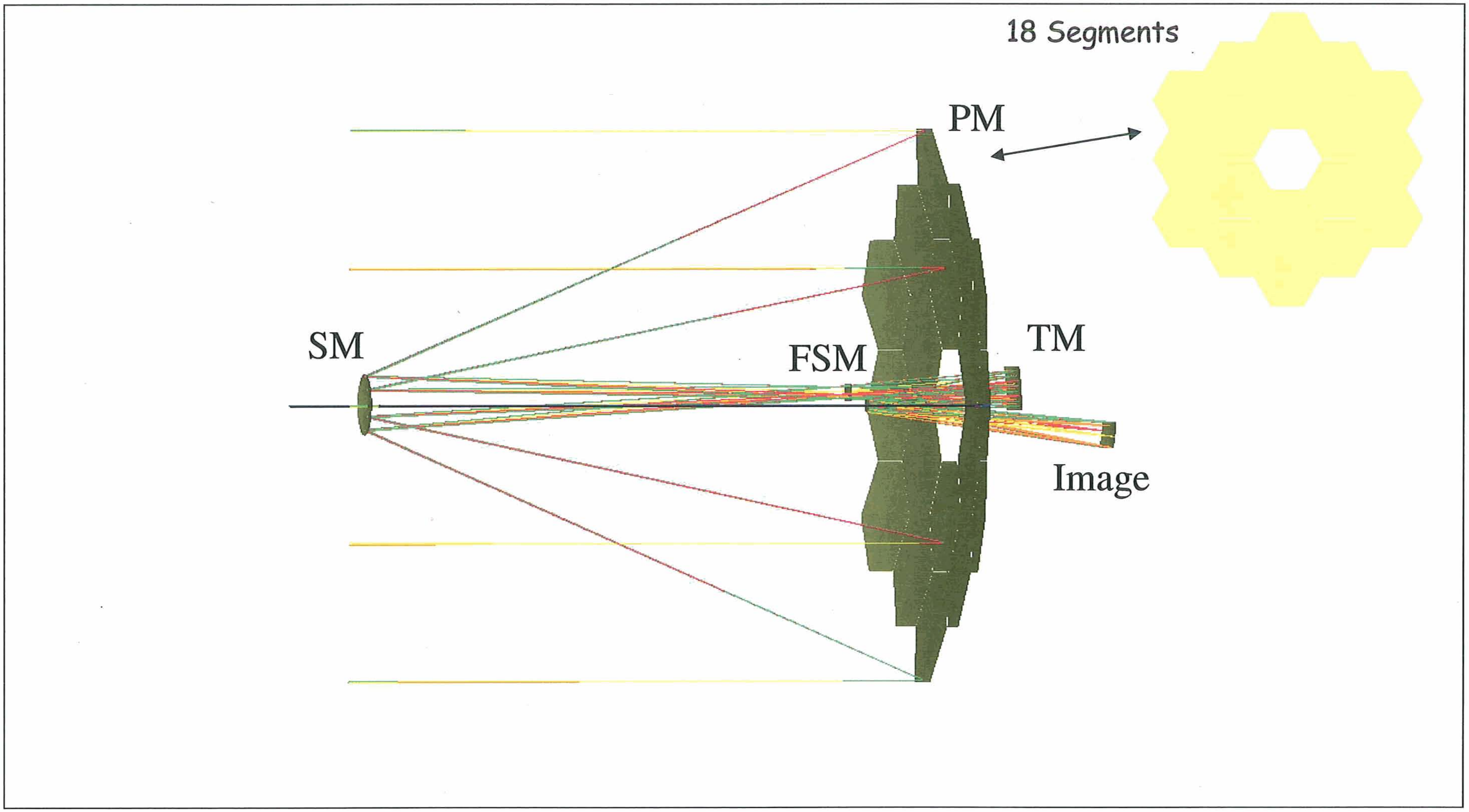
SS Layer 1

OTE Deployment Tower

**Sunshield (SS)**
-5 layers to provide thermal shielding to allow OTE and ISIM to passively cool to required cryogenic temperatures

**Spacecraft Bus**
-Contains traditional "ambient" subsystems

# Optical Design of JWST



Three-Mirror-Anastigmat (TMA) wide-field telescope design

# Outline

**Part 1: Toolkit Overview:**

- Purpose of the Toolkit
  - Intended audience
  - Value added over CodeV alone
- Toolkit Layout
  - System Functions / Utilities
  - Lens Info / Manipulation Functions
  - Analysis Tools / Ray Tracing Functions
- How Matlab gets data from CodeV
  - COM+
  - The CodeV Buffer
- Function Layout
  - Inputs: Use and syntax of individual functions
  - Outputs: Displaying data and using raw matrix data
- Using cvHELP

**Part 2: A Brief Tutorial**

- Starting CodeV from Matlab, loading a lens file, and getting system data
- OPD Maps -cvPMA
- PSFs - cvPSF
- Ray Tracing
  - Single rays - cvr, cvRSI
  - Grids of rays - cvRayGrid, cvRayTra
- Encircled Energy - cvENC
- Sensitivity Analysis
  - Decentering / Tilting Lens Elements
  - Linear Optical Model: cvLOM
- Displaying a lens from CodeV in Matlab

**Part 3: Live demonstration** (hopefully)

- OPD output using cvPMA
- PSF output using cvPSF

# Purpose of the Toolkit

- Intended audience
  - **Non-CodeV Users:** Those with optical design and analysis experience wanting to perform analyses using CodeV.
  - **CodeV Users:** Those wanting additional analysis functions and an easy method to run myriad sensitivity analyses.
- Value added over CodeV alone
  - Commands are standardized and easy to pick up for anyone with Matlab experience.
  - Data output is Matlab MAT files. Graphical functions are easily modified.
  - Sensitivities can be analyzed with simple "for" loops.

# Toolkit Layout

## System Functions / Utilities

| | |
|---|---|
| cvon | - establishes the COM link between Matlab and CodeV |
| cvoff | - kills the COM link between CodeV and Matlab |
| cvin | - inputs .seq file into CodeV COM |
| cvopen | - inputs .len file into CodeV COM |
| cvsave | - saves current lens file under pathfilename |

## Lens Info / Manipulation Functions

| | |
|---|---|
| cvgetf | - gets the desired fieldpoints in CodeV |
| cvsetf | - sets fieldpoints in CodeV |
| cvgetw | - gets desired wavelengths and weights in CodeV |
| cvsetw | - sets wavelengths and weights in CodeV |
| cvsetz | - sets active zoom positions in CodeV |
| cvgetap | - gets the aperture data from the current lens |
| cvims | - gets image surface number for lens in CodeV |
| cvshift | - perturbs the CodeV file by rigid body motion |
| cvrbshift | - "shifts" the decenters for single surface 'surfnum' |
| cvgc | - get the global coordinates for a lens |
| cvlensdata | - CVLENSDATA gets the lens data for the current lens |
| cvdraw | - draws the current lens in Matlab |

# Toolkit Layout Cont.

## Ray Tracing Functions

| | |
|---|---|
| **cvr** | - gets CodeV ray trace data for reference rays from database |
| **cvray** | - gets CodeV ray trace data from RAYRSI macro function |
| **cvRSI** | - gets CodeV ray trace data from RSI command |
| **cvraygrid** | - returns a grid of ray data for a particular surface and datatype |
| **cvraytra** | - uses CodeV RAYTRA engine to calculated a grid of rays. |

## Analysis Tools

| | |
|---|---|
| **cvENC** | - gets CodeV PSF based encircled energy for fieldpoint 1 |
| **cvPMA** | - gets CodeV exit pupil wavefront, mask, RefRad, f/#, and focal length data |
| **cvPSF** | - gets CodeV PSF for fieldpoint 1, normalized to perfect lens (Strehl) |
| **cvsens** | - gets CodeV sensitivity data at image based on |
| **cvWAV** | - gets wavefront analysis data from CodeV |
| **cvspot** | - graphs a spot diagram for a given field and ray density |
| **cvlom** | - calculates the linear optical model for the given parameters |

# Toolkit Layout Cont.

## Utilities

**cvbufgetarray** - gets continuous data from rows and cols in CodeV buffer

**cvbufgetrow** - gets continuous data from rows in CodeV buffer

**cvbufnum** - gets a single number from the applicable CodeV buffer

**cvbufstr** - gets a single string from the applicable CodeV buffer

**cvcmd** - sends command to CodeV command line over existing COM link

**cvEVA** - Sends command to CodeV command line over existing COM link

**cvgetINT** - Reads in an int file and plots it

## Project Specific Scripts

**cvnircam** - this script loads JWST NIRCAM Short wavelength channel into CodeV

**cvjwst** - This script loads JWST segmented OTE file into CodeV, and puts fieldpoints into the workspace.

# How Matlab gets data from CodeV

- **CodeV and COM**
  - **CodeV API** (Application Programming Interface) uses the Windows **COM** (Component Object Model) interface for passing commands and data between programs.
  - Any VB, C++, program can send commands to CodeV and get data back from CodeV (Refer to the *CODE V API Reference Guide for details.)*
  - Many functions are built into the COM for easy calls to ray trace data, OPD data, and PSFs. Other data can be sent to the CodeV buffer.

- **The CodeV Buffer**
  - The CodeV Worksheet Buffer allows for fast access to any data output from CodeV.
  - Once CodeV writes data to the buffer, COM can immediately read that data Matlab

# Function Layout

- **Inputs**: Use and syntax of individual functions

```
[output1, output2, …] = function(input1, input2, …);
```

  - "function" is any of the functions starting with CV
  - "input" is the particular input requested by the function
  - "output" is the variable name where the output is going

- **Outputs**: Displaying data and using raw matrix data (example)
  - To display a PSF to screen, omit the outputs

`cvpsf(128,64);` will create a PSF plot with a transform grid of 128 and 64 rays across the diameter.

  - To output the PSF data to a matrix, enter the outputs preceding the function.

`[psf,grid_spacing] = cvpsf(128,64);` will create 2 arrays: psf- containing the psf data (128 by 128), and grid_spacing- containing the grid spacing for the psf (single value).

# Toolkit output- data or graphical

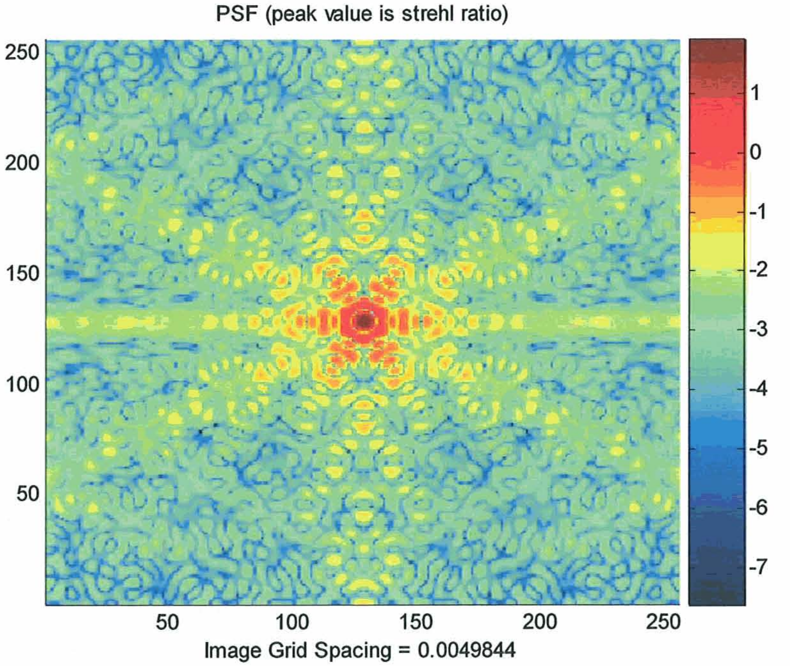
**Input:** `[psf, grid_spacing] = cvpsf(256,64);`

```
psf(124:134,124:134) =
3.4408    2.3350    0.9989    0.2010    0.0471    0.0909    0.0471    0.2010    0.9989    2.3350    3.4408
3.0428    0.8452    0.0551    1.8657    4.9077    6.3941    4.9077    1.8657    0.0551    0.8452    3.0428
1.6787    0.0032    3.1947   12.2560   22.6098   27.1931   22.6098   12.2560    3.1947    0.0032    1.6787
0.5103    1.2104   11.2099   30.6998   50.9006   59.5509   50.9006   30.6998   11.2099    1.2104    0.5103
0.0719    3.5264   19.9491   48.6187   77.2257   89.3051   77.2257   48.6187   19.9491    3.5264    0.0719
0.0199    4.5292   23.2390   55.0783   86.5443   99.7826   86.5443   55.0783   23.2390    4.5292    0.0199
0.0939    3.1336   18.5847   45.8443   73.1451   84.6886   73.1451   45.8443   18.5847    3.1336    0.0939
0.6374    0.8681    9.5687   27.0564   45.3521   53.2123   45.3521   27.0564    9.5687    0.8681    0.6374
1.8470    0.0102    2.3169    9.7872   18.5329   22.4334   18.5329    9.7872    2.3169    0.0102    1.8470
3.0027    0.9877    0.0012    1.1263    3.3522    4.4724    3.3522    1.1263    0.0012    0.9877    3.0027
3.0544    2.2104    1.0918    0.3145    0.0374    0.0052    0.0374    0.3145    1.0918    2.2104    3.0544

grid_spacing =
0.0049843814 mm
```

Numeric Output

Graphical Output



# Using cvHELP

- To look up the syntax for any function, simple type "***help function***" where "function" is the CodeV function you want.

- To see a list of all the Matlab-CodeV toolkit functions, simply type "***cvhelp***" and a list with descriptions of all the functions will appear. Then just click on one.

- Each function has help on command syntax and usage. Also most functions have defaults set so you can omit all or most inputs for quick analyses.

# Tutorial: Starting CodeV from Matlab, loading a lens file, and getting system data

1. Start a CodeV server session:
   - Type `cvon;`
   - A variable called CVhandle will appear. This shows that CodeV has been started.
2. Load a lens file:
   - For .seq files, use: `cvin(‘filename’);` (you need the '' part)
     or just `cvin;` (Matlab will graphically prompt you for a filename.)
   - For .len files, use `cvopen(‘filename’);`
     or just `cvopen;` (Matlab will graphically prompt you for a filename.)
3. Get fields and wavelengths:
   - To get field info. and put it in the workspace, type: `[fx,fy] = cvgetf;`
   - To get system wavelengths and weights, type: `[wl,wt] = cvgetw;`
4. Set fields and wavelengths:
   - To set fields or wavelengths, either enter a vector containing all of the parameters or enter the field number. If a field number is entered, all others will be removed and that field will now be field 1.
   - Set field angle to field 10: `cvsetf(10);` Set wavelength to wl 5: `cvsetw(5);`
   - Set field to x, y angles = 0: `cvsetf(0,0);` Set wl to 2000nm with weight 1: `cvsetw(2000,1);`

# Tutorial: OPD Maps - cvPMA

```
[opd,mask,pin,Xfno,Yfno,RefRad,Xfl,Yfl]=
    cvPMA(TGR,fit_option,f,w,z,NRD_GRI);
```

## INPUTS:

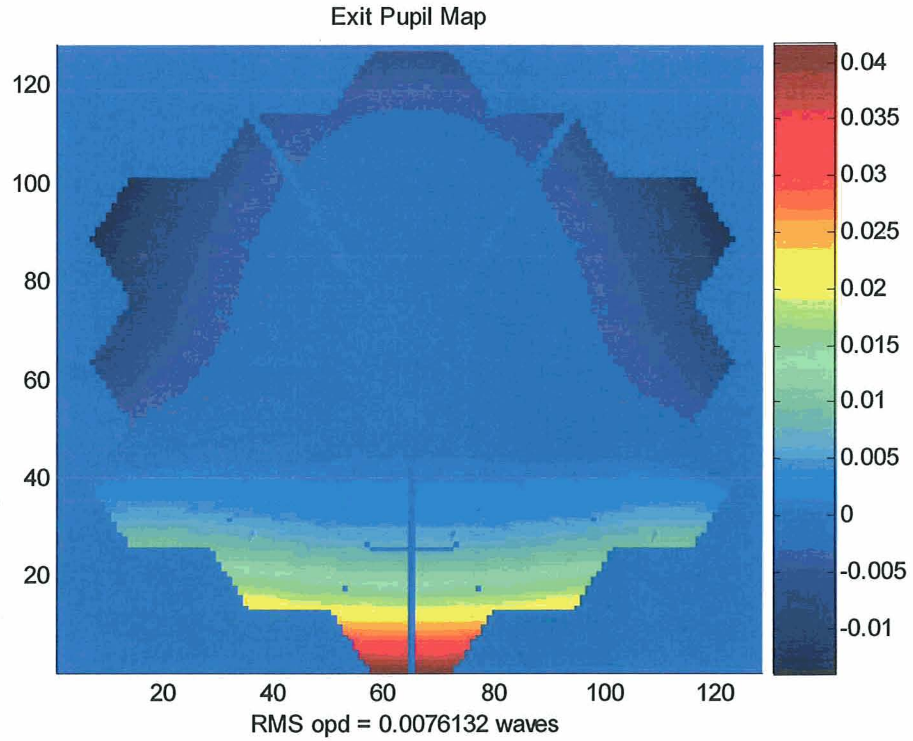
- **TGR** of square array, power of 2
- **fit_option**, default = 0 (no fit), 1 = best fit tilt removed, 2 = best fit focus
- **f,w,z** = field number, wavelength number, zoom position (all default=1)

## OUTPUT:

- opd, mask, pin (pupil intensity), X Y real f/#, reference radius, X Y real focal length

## NOTES:

- Only gets data from first output in CodeV lens file. Field 1 and wave 1 should be redefined for data of interest!
- Since defaults are defined for most input parameters, use only as many output as you need- no need to output everything.
- If all outputs are omitted, cvPMA will return a figure of the OPD



# Tutorial: Point Spread Functions - cvPSF

```
[psf,grid_spacing] =
   cvPSF(TGR,NRD_GRI,PGR,PRO);
```

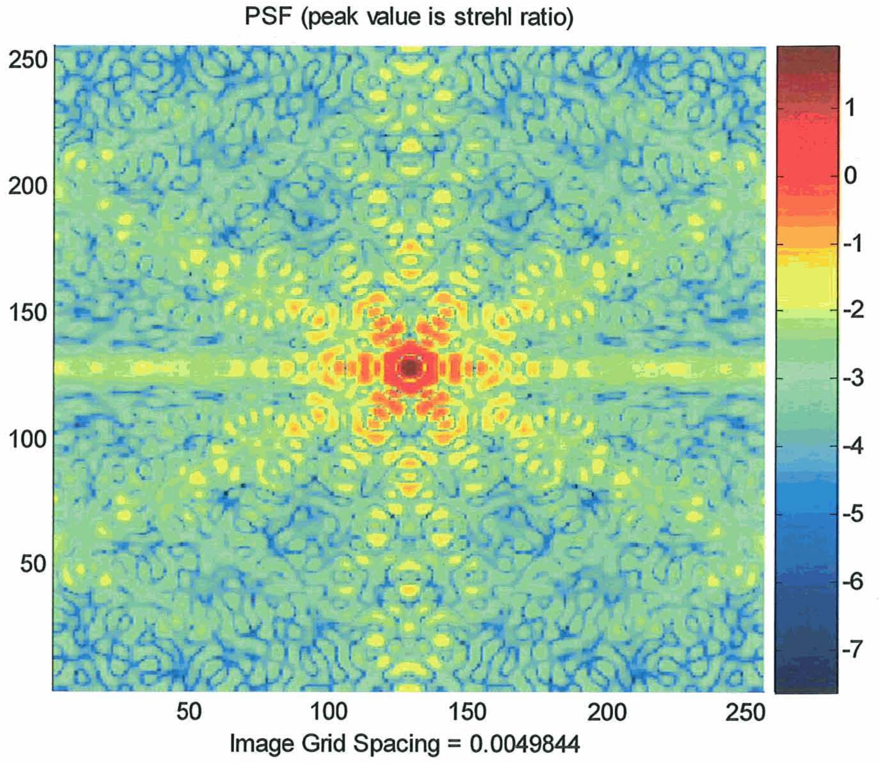
INPUTS:

- TGR = Transform Grid Size
- NRD_GRI = # of rays across pupil diameter
- PGR = Size of output array, default = TGR
- PRO = Use propagate equations for defocused image

OUTPUT:

- psf: A matrix of the PSF data (size = PGR x PGR)
- grid_spacing: Image plane pixel size in lens units

Command used to create plot: `cvPSF(256,64);`



# Tutorial: Ray Tracing

Single rays

- cvr - allows the user to quickly look at reference ray parameters (position and direction cosines)

Typing `cvr('y',1,1,cvims,1,1)` returns `-2.1356e-005`, the chief ray height for a field of 0,0 at the image plane.

- cvRSI - returns ray data for a single ray at every surface

- Grids of rays
  - cvRayGrid - produces a grid of ray parameters for a specific surface. Any database object can be retrieved (i.e. position, direction, AOI, etc...
  - cvRayTra - uses a fast ray trace engine to calculate ray properties at the image plane. (Used in cvspot)

# Tutorial: cvSpot

```
[raygridx,raygridy] =
  cvspot(fnum,wavnum,znum,gridsize,center);
```

INPUTS:

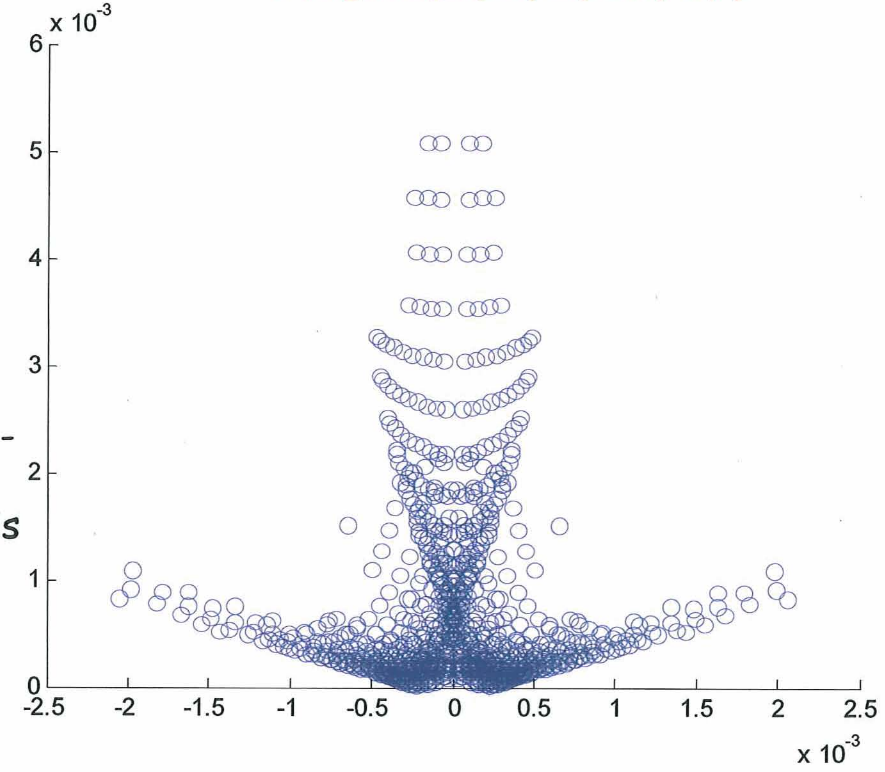
- fnum : can be a vector of field numbers (def = 1)
- wavnum, znum: (defaults =1)
- gridsize: number of rays across diameter to be traced.
- center: no centering, centered on zero, multiple fields

OUTPUT:

- radgridx,y: the grid of data at the image plane (x- and y- coordinates.
- *A* scatter plot showing the spot diagram (output is in lens units)

Command used to create plot:

```
cvspot(1,1,1,40,1);
```



# Tutorial: Encircled Energy - cvENC

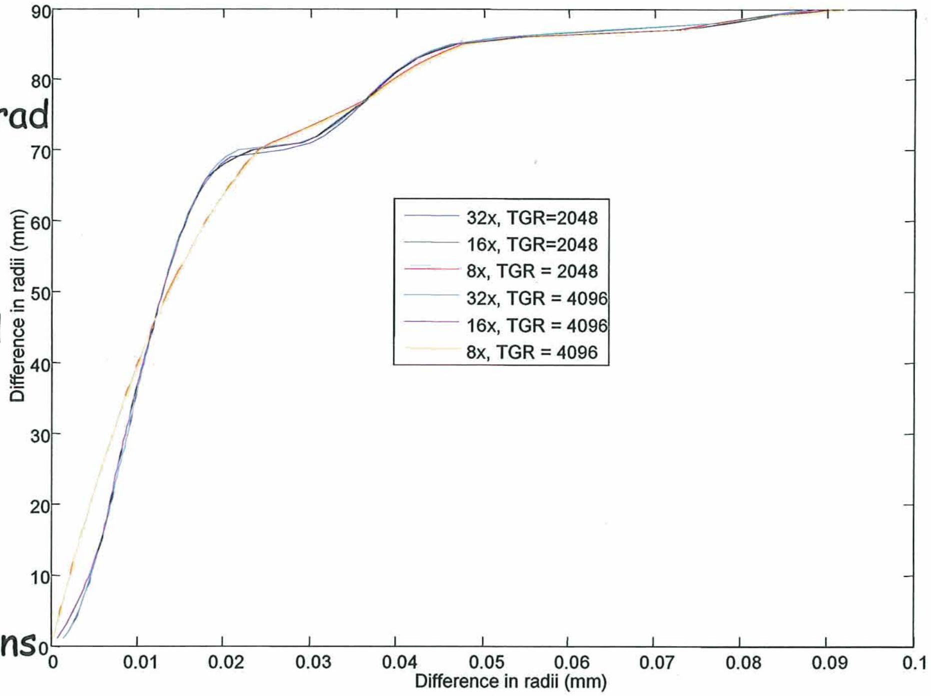
```
[percent,radii] =
  cvENC(PER,TGR,NRD_GRI,PGR,PRO,CEN);
```

INPUTS

- PER = percent values to get encircled energy rad
- TGR = Transform Grid Size
- NRD_GRI = # of rays across pupil diameter
- PGR = Size of output array, default = TGR
- PRO = Use propagate equations for defocused image
- CEN = Centered on Chief Ray or Centroid

OUTPUT

- Percent = percentages at which radii were calculated
- radii = circle size encompassing % energy in lens units

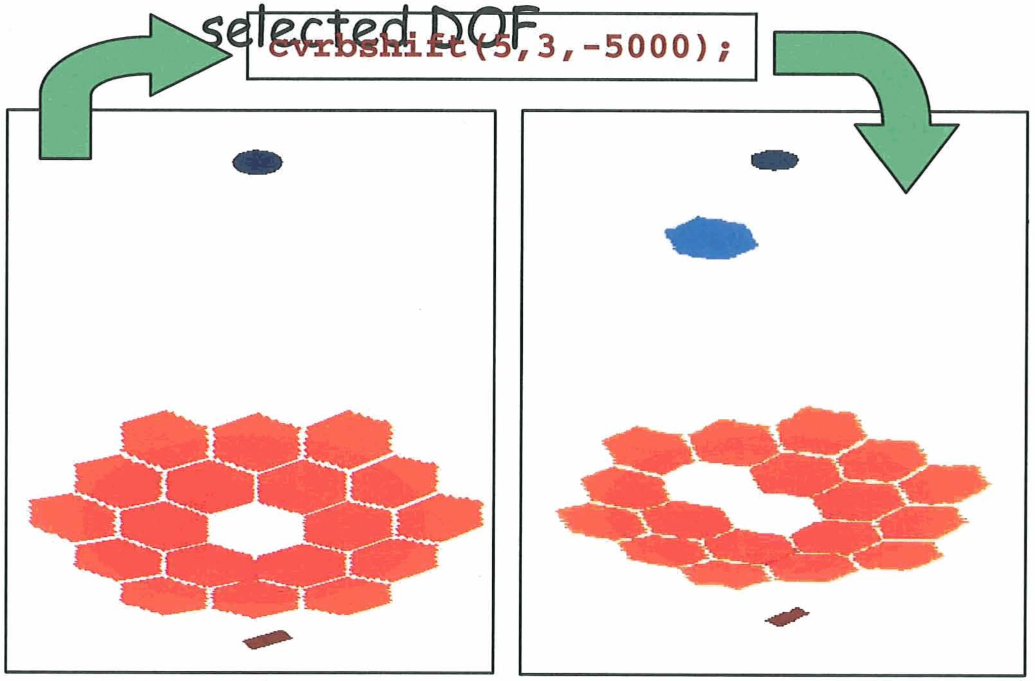


Data created with a simple “for” loop:

```
for ii=1:6
     [radii(:,ii),percent] =
     cvENC(percent,TGR(ii),NRD(ii),TGR(ii));
end
```

# Tutorial: Sensitivity Analysis

- Decentering / Tilting Lens Elements:
  - **cvshift** - shifts the selected surface in 6 DOF at once.
  - **cvrbshift** - shifts the selected surface in 1 selected DOF.



```
cvrbshift(5,3,-5000);
```

Shift surface 5 by -5000mm along the z-axis

- Sensitivities / Linear Optical Model:
  - **cvLOM** - This function outputs the L, W, and C matrices needed for integrated modeling activities. cvlom is a specific form of cvsens, with the additional ability to take data with respect to a stationary exit pupil.

  **cvsens** - computes the sensitivity of the selected surfaces to 6DOF. The output is a stack of matrices. Using a **zern_fit** routine, one can easily access a simple numeric sensitivity output.

# Getting lens data from CodeV

```
[lens] = cvlensdata(refsurf);
```

- cvlensdata returns all of the construction parameters for the current lens file as a Matlab structure array

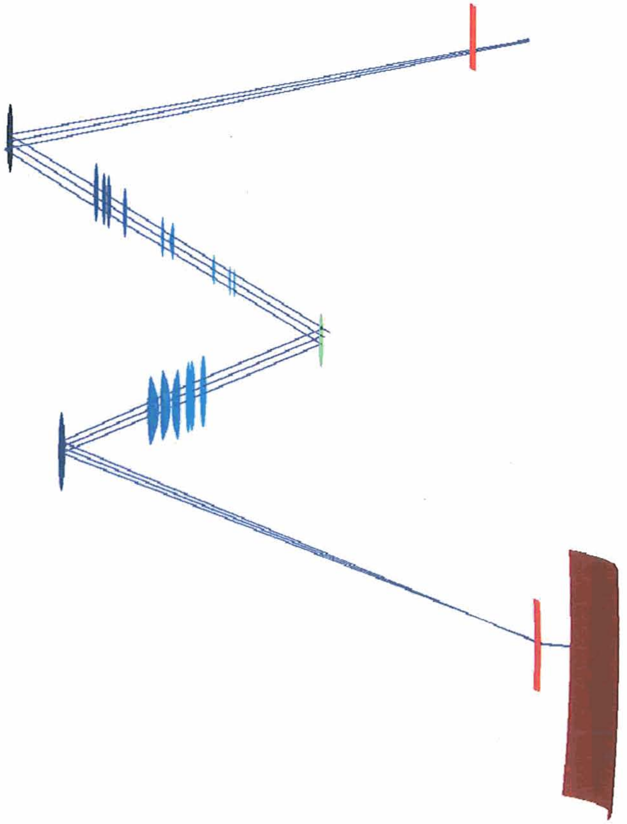
```
lens =
       counts: [1x1 struct]
     stopsurf: 2
          dim: 2
           fx: 0
           fy: 0.00175000000000
          wvl: [1x1 struct]
           wl: [1x1 struct]
         surf: [1x35 struct]
     g_coords: [35x6 double]
```

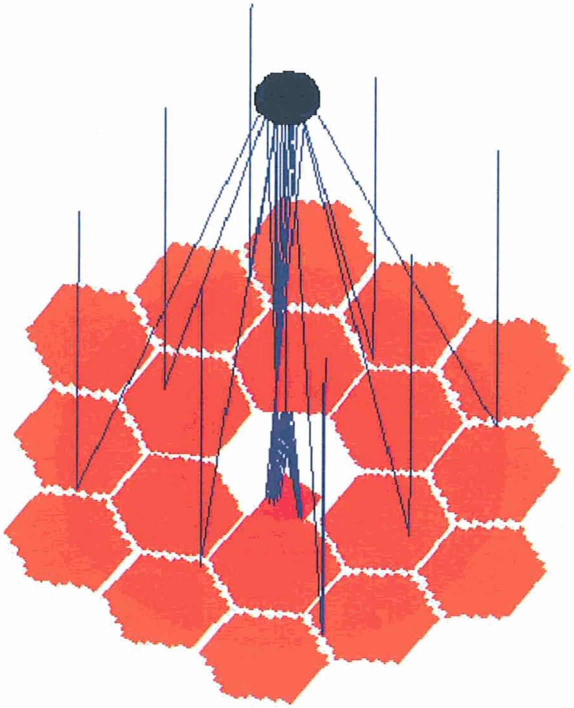
```
lens.counts =
           numz: 2
        surface: 35
           numf: 1
           numw: 1
```

```
lens.surf =
1x35 struct array with fields:
    label
    number
    shape
    rdx
    rdy
    k
    ap
```

# Tutorial: Displaying a lens from CodeV in Matlab

`cvdraw(surfaces);` Draws the selected surfaces in 3D and plot the reference rays for the system.





# GO NAVY!!! BEAT ARMY!!!

Any Questions?