Table of Contents

Test actuator coordinates

Function to show examples of different actuator coordinates passed to the OOMAO functions

```
clear all;
```

```
 @(telescope)> Terminated!
 @(source)> Terminated!
 @(deformable mirror)> Terminated!
 @(gaussian influence fun)> Terminated!
 @(calibration vault)> Terminated!
```

AO Parameters

Here we define the main parameters for the WFS, the resolution, number of lenslets etc, as well as the turbulence parameters.

```
% We use 38x38 pixels for each Pyramid pupil.
nLenslet = 36;
nPx = 4*nLenslet;
nActuator = 12;

% Size of the telescope aperture.
D = 1.52;

% We define a telescope object with the given aperture and resolution.
tel =
 telescope(D,'fieldOfViewInArcMin',2.5,'resolution',nPx,'samplingTime',1/100);

% We use an R-band (lambda = 640nm) source to most closely match our
% laboratory laser (lambda = 635nm)
ngs = source('wavelength',photometry.R);
```

```
~~~~~~~~~~~~~~~~~~~
 BEWARE OF OOMAO!
~~~~~~~~~~~~~~~~~~~
 @(logBook)> Opening the log book!
 @(telescope)> Created!
___ TELESCOPE ___
 1.52m diameter full aperture with  1.81m^2 of light collecting area;
 the field-of-view is 2.50arcmin; the pupil is sampled with 144X144
 pixels
```

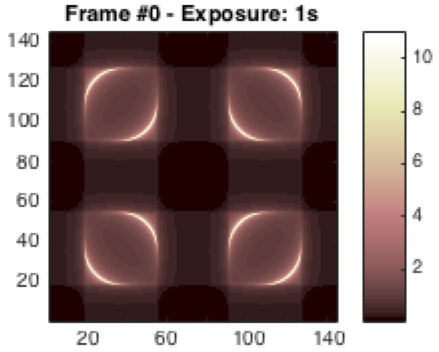
```
----------------------------------------------------
 @(source)> Created!
___ SOURCE ___
 Obj   zen[arcsec] azim[deg]  height[m]  lambda[micron] magnitude
  1      0.00        0.00         Inf      0.640           0.00
----------------------------------------------------
```

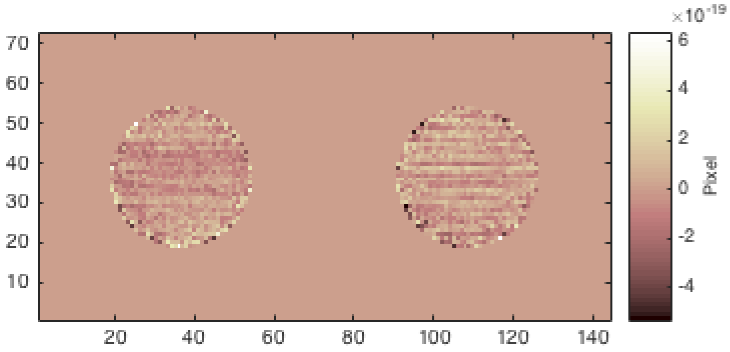
Definition and calibration of Pyramid WFS

```
wfs = pyramid(nLenslet,nPx,'modulation',0);

% The source if propagateed to the WFS through the telescope.
ngs = ngs.*tel*wfs;

%wfs.camera.readOutNoise = 0.0006;

% We initiate the sensor to get the reference slopes (our zero point).
wfs.INIT;
+wfs;

% The reference camera image and slope signals are displayed.
    subplot(2,1,1)
    imagesc(wfs.camera)
    subplot(2,1,2)
    slopesDisplay(wfs)
```

```
 @(telescope)> Created!
___ TELESCOPE ___
 -1.00m diameter full aperture with  0.79m^2 of light collecting area;
----------------------------------------------------
 @(source)> Created!
___ SOURCE ___
 Obj   zen[arcsec] azim[deg]  height[m]  lambda[micron] magnitude
  1      0.00        0.00         Inf      0.550           0.00
----------------------------------------------------
 @(detector)> Created!
 @(pyramid)> Created!
 @(source)> Computing the objective wavefront transmitance ...
 @(source)> Created!
___ SOURCE ___
 Obj   zen[arcsec] azim[deg]  height[m]  lambda[micron] magnitude
  1      0.00        0.00         Inf      0.640           0.00
----------------------------------------------------
 @(zernike polynomials)> Created!
___ ZERNIKE POLYNOMIALS ___
 . mode: 3
----------------------------------------------------
 @(telescope)> Created!
___ TELESCOPE ___
 1.00m diameter full aperture with  0.79m^2 of light collecting area;
 the pupil is sampled with 144X144 pixels
----------------------------------------------------
 @(source)> Computing the objective wavefront transmitance ...
```

```
@(source)> Terminated!
@(zernike polynomials)> Terminated!
@(telescope)> Terminated!
```





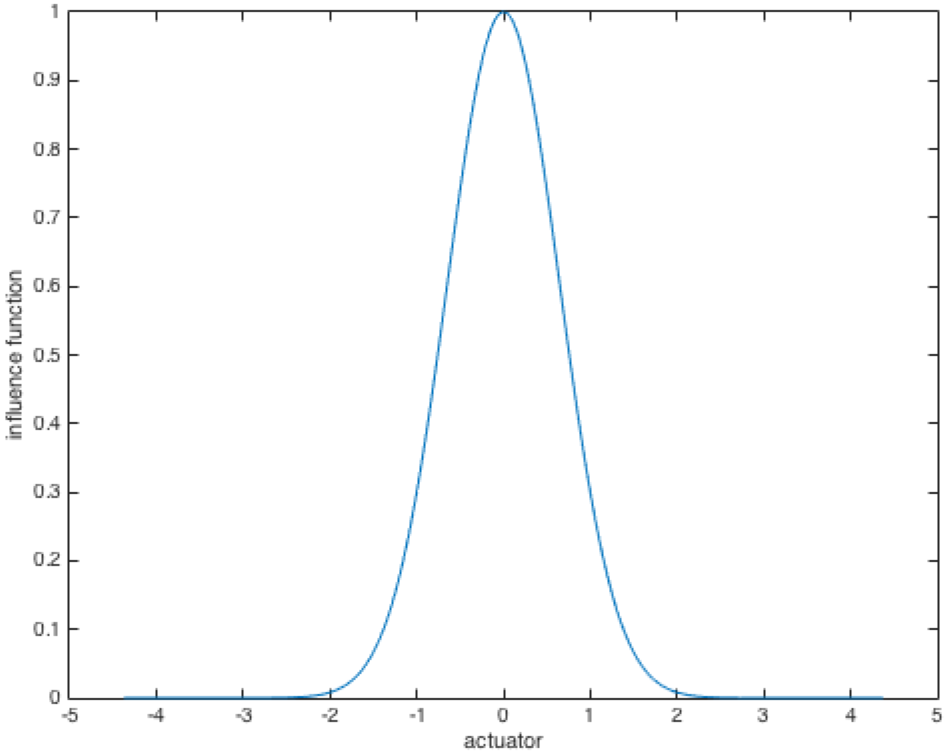
1) Deformable mirror: standard influence function

Using influence function with actuators in default coordinates Create influence function for individual actuators.

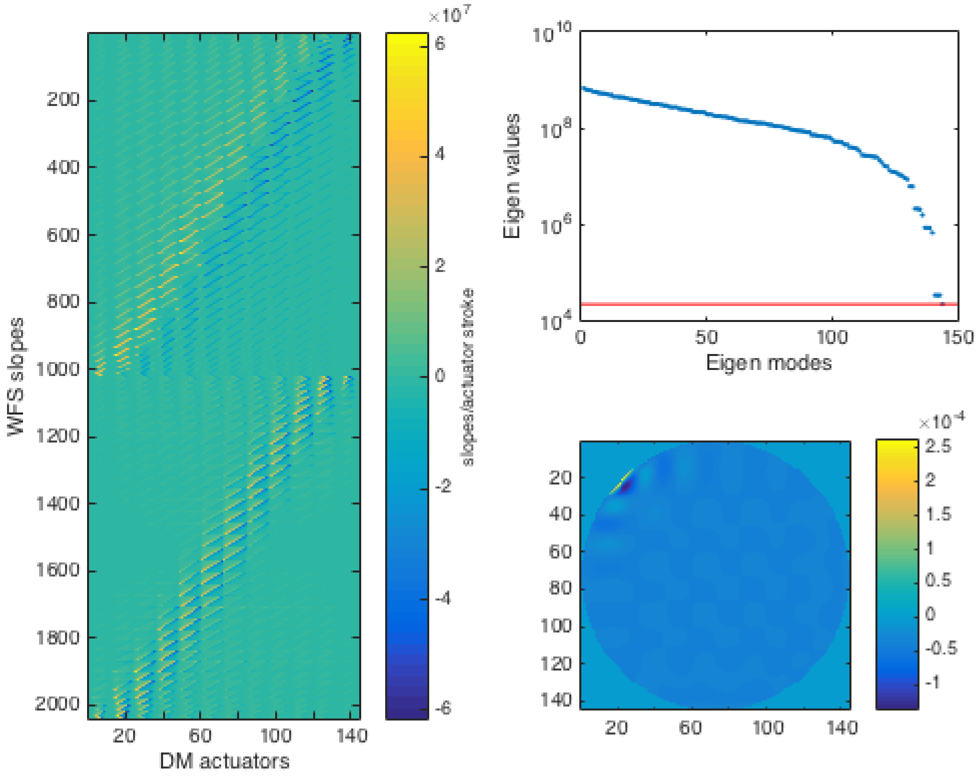
```
bif = gaussianInfluenceFunction(30/100);

% Plot influence function
figure
show(bif)

% DM with default actuator positions
vAct = logical(ones(nActuator,nActuator));
dm = deformableMirror(nActuator,'modes',bif,'resolution',nPx,...
    'validActuator',vAct);

% Interaction matrix
stroke = ngs.wavelength/10;
ngs=ngs.*tel;
dmCalib = calibration(dm,wfs,ngs,stroke);
```

```
% Plot first actuator IF
figure
    imagesc(reshape(dm.modes.modes(:,1),nPx,nPx));
    colorbar();
    axis tight square;
```

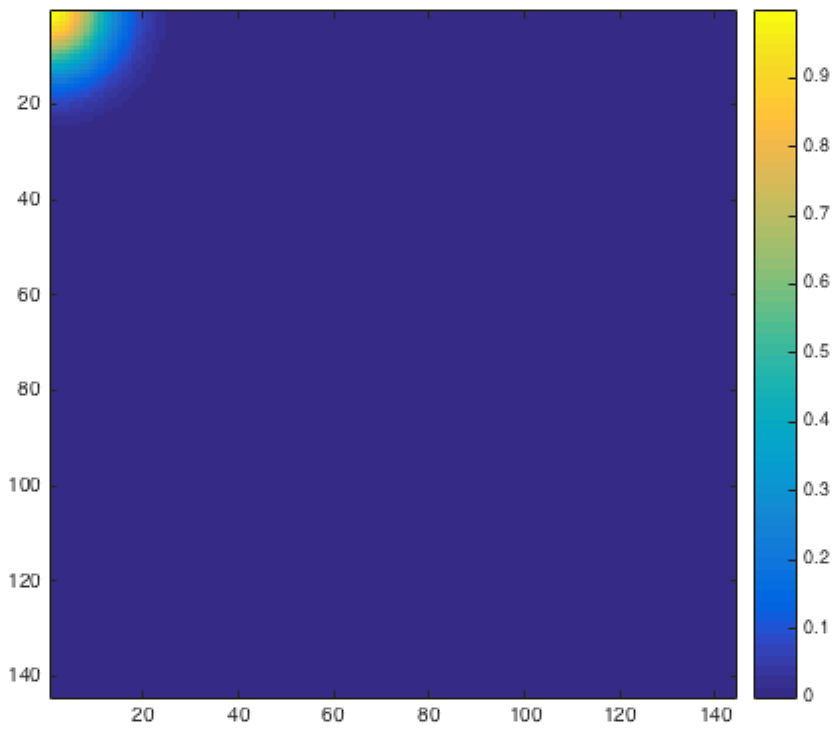
```
 @(gaussian influence fun)> Created!
 @(influenceFunction)> Computing the 2D DM zonal modes... ( 144,  144
 @(deformable mirror)> Created!
___ DEFORMABLE MIRROR ___
 12X12 actuators deformable mirror:
  . 144 controlled actuators
----------------------------------------------------
__ Poke Matrix Stats ___
 . computing time: 10.29s
 . size: 2040x144
 . non zeros values: 293760 i.e. 100.00%
 . min. and max. values: [ 3.98,-3.94]
 . mean and median of absolute values: [ 0.14, 0.03]
________________________
 @(calibration vault)> Created!
 @(calibration vault)> Computing the SVD of the calibration matrix!
 @(calibration vault)> Condition number 31654.5
```



influence function
actuator
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-5
-4
-3
-2
-1
0
1
2
3
4
5



WFS slopes
DM actuators
200
400
600
800
1000
1200
1400
1600
1800
2000
20
60
100
140
×10^7
6
4
2
0
-2
-4
-6
slopes/actuator stroke
Eigen values
Eigen modes
10^10
10^8
10^6
10^4
0
50
100
150
×10^-4
2.5
2
1.5
1
0.5
0
-0.5
-1
20
40
60
80
100
120
140



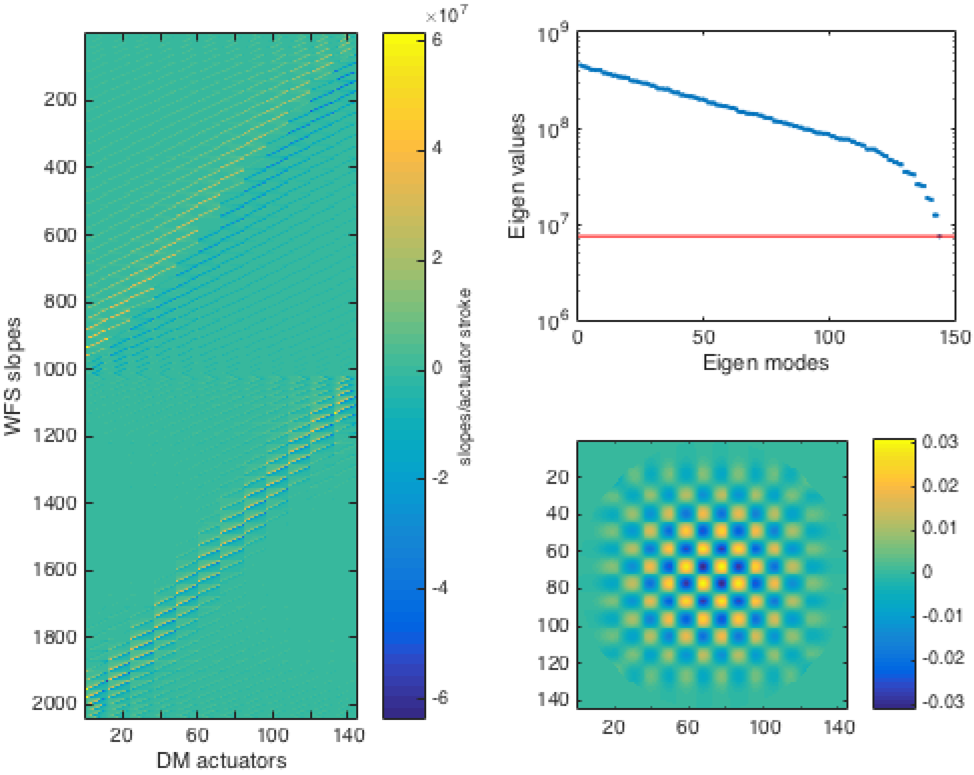
2) Deformable mirror: original IF class, defined actuator geometry

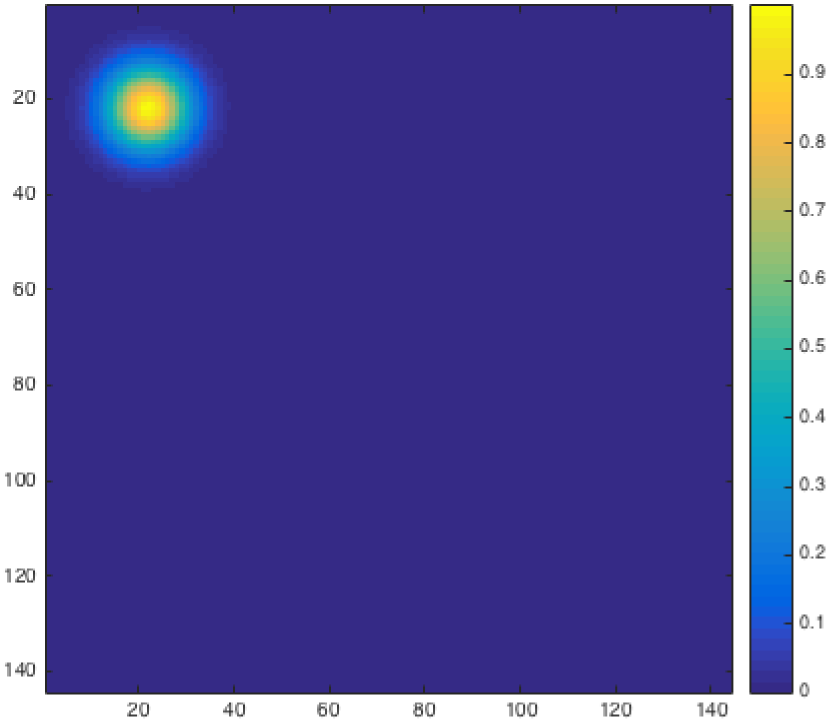
Using influence function with actuators at user defined positions. In this case the orginal class is used

```
bif = gaussianInfluenceFunction(30/100);
%bif = influenceFunction('monotonic',30/100);

% Defined coordinates (should be equivalent to default coordinates)
zoom = 1.0;
x = linspace(-(nActuator-1)/2,(nActuator-1)/2,nActuator)*zoom;
y = linspace(-(nActuator-1)/2,(nActuator-1)/2,nActuator)*zoom;
[X,Y] = meshgrid(x,y);
bif.actuatorCoord = X - 1i*Y;

% DM
vAct = logical(ones(nActuator^2,1));
dm = deformableMirror(nActuator^2,'modes',bif,'resolution',nPx,...
    'validActuator',vAct);

% Interaction matrix
stroke = ngs.wavelength/10;
ngs=ngs.*tel;
dmCalib = calibration(dm,wfs,ngs,stroke);
```

```
% Plot first actuator IF
figure
    imagesc(reshape(dm.modes.modes(:,1),nPx,nPx));
    colorbar();
    axis tight square;
```

```
 @(gaussian influence fun)> Created!
 @(gaussian influence fun)> Expected non-zeros: 944784
 @(gaussian influence fun)> Computing the 144 2D DM zonal modes...
 @(influenceFunction)> Computing the 2D DM zonal modes... ( 144, 144
 @(gaussian influence fun)> Actual non-zeros: 824464
 @(deformable mirror)> Created!
___ DEFORMABLE MIRROR ___
 144X144 actuators deformable mirror:
  . 144 controlled actuators
----------------------------------------------------
 @(deformable mirror)> Terminated!
 @(gaussian influence fun)> Terminated!
__ Poke Matrix Stats ___
 . computing time:  9.96s
 . size: 2040x144
 . non zeros values: 293760 i.e. 100.00%
 . min. and max. values: [ 3.92,-4.06]
 . mean and median of absolute values: [ 0.12, 0.03]
________________________
 @(calibration vault)> Created!
 @(calibration vault)> Computing the SVD of the calibration matrix!
 @(calibration vault)> Condition number 60.6957
 @(calibration vault)> Terminated!
```


WFS slopes
200
400
600
800
1000
1200
1400
1600
1800
2000
20
60
100
140
DM actuators
×10^7
6
4
2
0
-2
-4
-6
slopes/actuator stroke
Eigen values
10^9
10^8
10^7
10^6
0
50
100
150
Eigen modes
20
40
60
80
100
120
140
20
60
100
140
0.03
0.02
0.01
0
-0.01
-0.02
-0.03


20
40
60
80
100
120
140
20
40
60
80
100
120
140
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0

Published with MATLAB® R2015a