Roman Coronagraph Instrument
Deep Dive Technical Series

LOWFSSim
Integrated Model of LOWFS

Brandon D. Dube
Optical Engineer,
Optical Simulation and Analysis Group

November 11, 2021

brandon.dube@jpl.nasa.gov

Jet Propulsion Laboratory
California Institute of Technology

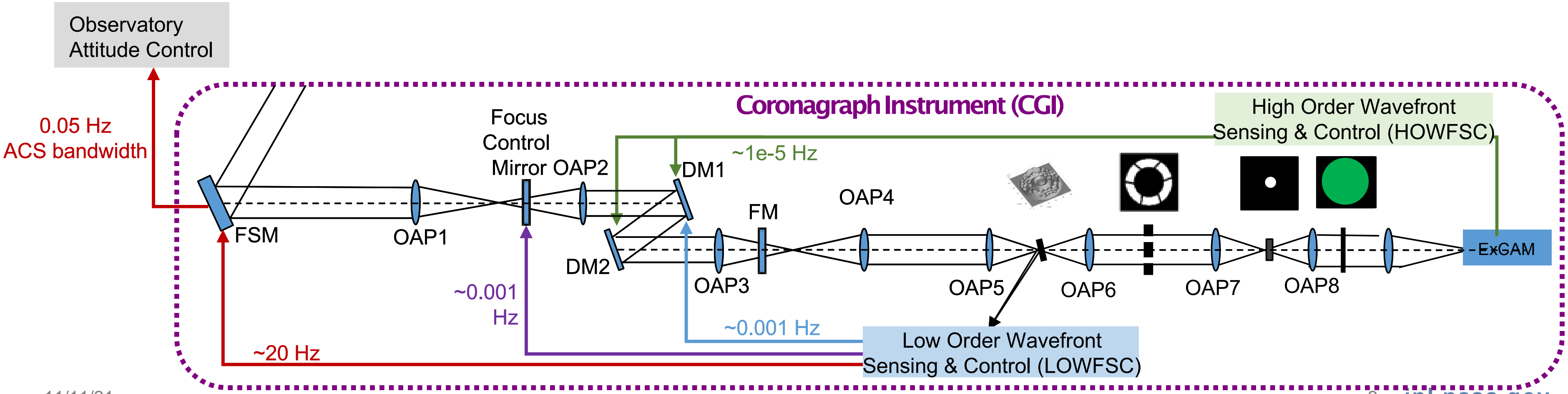
CL 21-5732

What is LOWFS?

LOWFS is the Low Order Wavefront Sensing and Control system aboard Roman-CGI.

LOWFS incorporates a **modal** Zernike Wavefront Sensor (ZWFS). It senses Z2-Z11 in Noll indexed Zernike polynomials.

It has ten control loops, spanning tip/tilt up to primary spherical aberration.



CGI Acronyms

LOWFS = Low-Order Wavefront Sensing & Control

HLC = Hybrid Lyot Coronagraph

SPC = Shaped Pupil Coronagraph

NFOV = Narrow Field-of-View

WFOV = Wide Field-of-View

LOS = Line-of-Sight

EMCCD = Electron Multiplying Charged Coupled Device

Why LOWFSSim?

What if I want to know what's happening at the full **1kHz** that hardware LOWFS runs at?

What if I want to do a study of multiple parameters and their influence on performance?

What if I want to iterate my analysis quickly; changing inputs based on what I'm seeing in real-time?

To do these things, the model has to be **fast.**

For the answers to be meaningful, the model has to be **accurate.**

What is LOWFSSim?

LOWFSSim is a model of the LOWFS system that includes:

- Diffraction
- Thin Film effects and Polarization at the occulter
- Radiometry
- Detector Effects
- Wavefront Sensing (exactly the flight algorithm, but not the flight implementation)
- Wavefront Control*
- All CGI configurations (HLC-NFOV, SPC-SPEC, SPC-WFOV)

The detector model is based on *Hirsch et al* 2013 https://doi.org/10.1371/journal.pone.0053671

Public repository: https://github.com/nasa-jpl/lowfssim
LOWFSSim is based on https://github.com/brandondube/prysm

Wavefront Control*

The public release of LOWFSSim includes the necessary building blocks to perform wavefront control but does not include the “prescription” for the flight controllers. Users of public LOWFSSim must design their own based on the top-level parameters:

Channels	Bandwidth (Hz)	Sample Rate (Hz)
Z2/Z3	20	1000
Z4-Z11	0.0016	0.1

Among the examples is a notebook that models closed-loop Z2, Z3, but the controller initialization is redacted and variables used are not present in the public source code.

The 0.1Hz system is derived from averages of the 1000Hz Zernike coefficient estimates; the estimator is not a multi-rate system, only the controllers are.

Wavefront Control*

A DM model is not included. The model supports injection by any phase/amplitude screen at the DM1 and DM2 planes. A user can plug in their own DM model this way, or utilize the idealized phase errors built into the model.

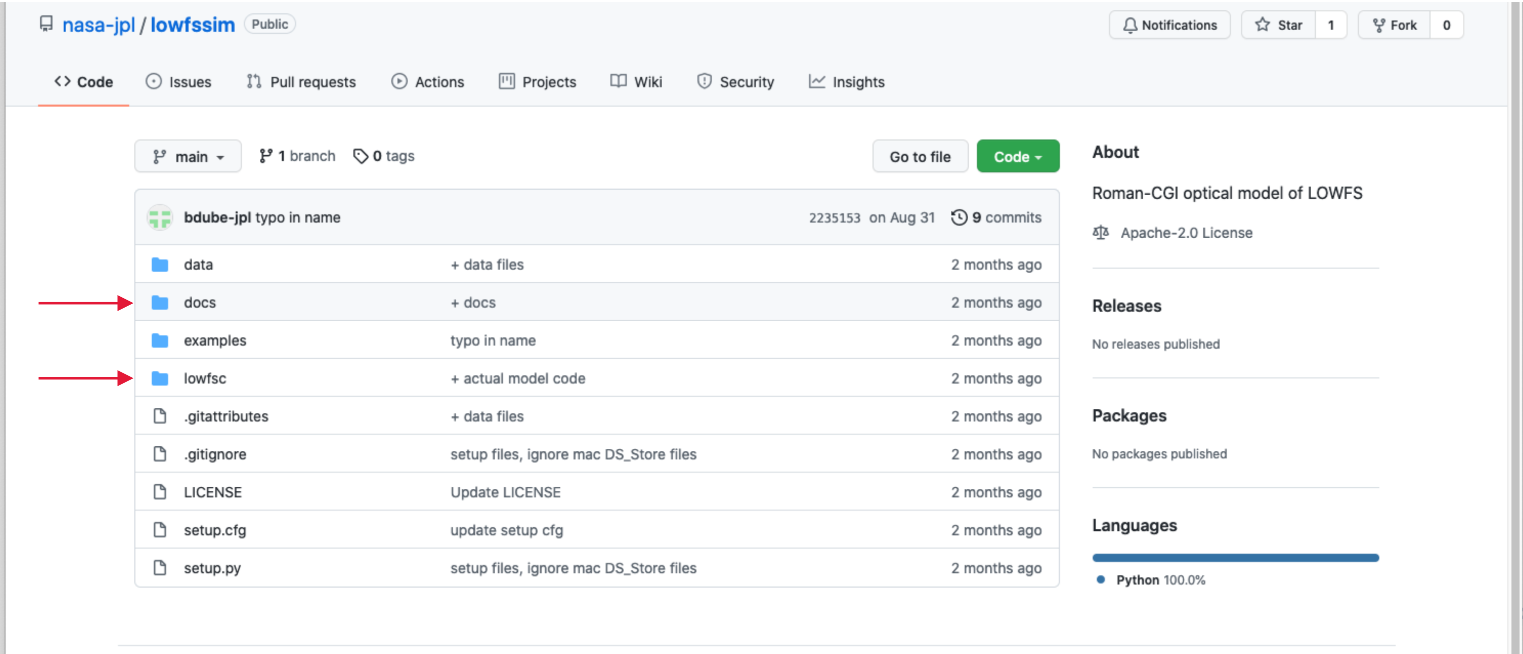
The omission of the actual flight controllers in the public release is not a request for the community to contribute new control designs.

Installing LOWFSSim

Download or git clone

then `pip install -e .` in the main lowfssim directory

To use a GPU, install the relevant version of cupy for your CUDA version

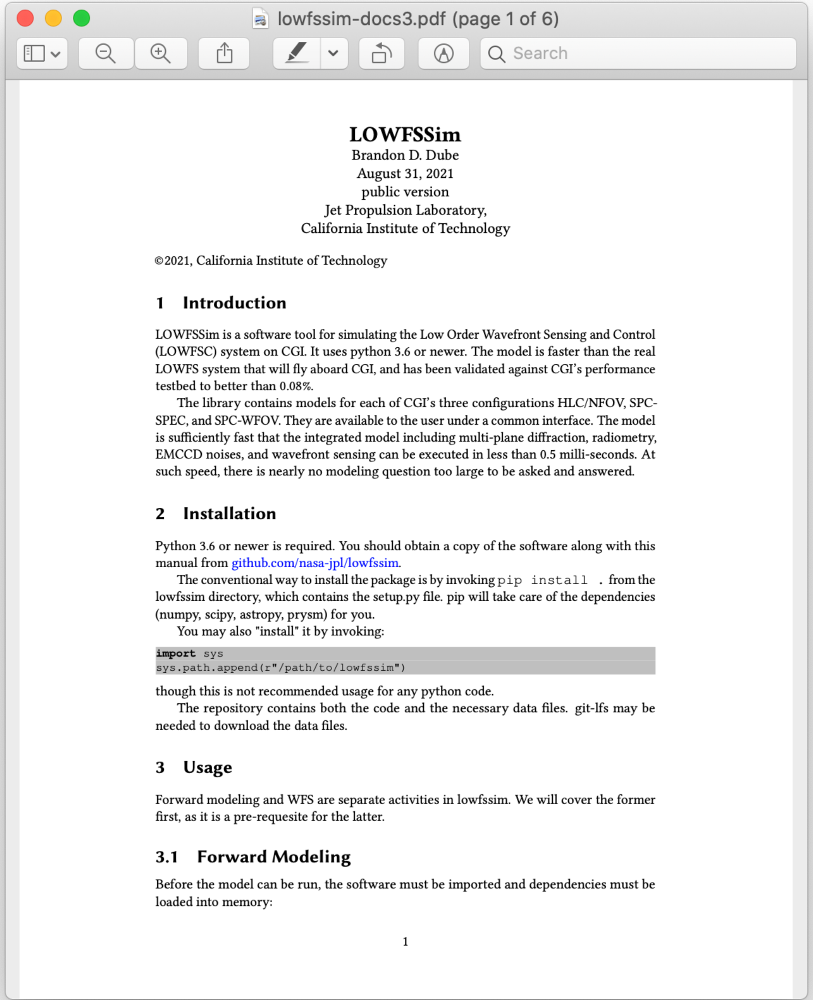


More Docs

```python
# props.py
def polychromatic(wvls, weights, data, zernikes=None,
                  locam_misfocus=0, locam_shear=(0, 0), pool=None):
"""Polychromatic forward model of LOWFS.

Parameters
----------
wvls : iterable of float
     wavelengths of light, microns
weights : iterable of float
     spectral weights to apply to the data
data : data.DesignData
     object with properties of:
     roman_pupil - amplitude map of roman pupil
     dm1_wfe - phase map in nm from DM1
     dm2_wfe - phase map in nm from DM2
     fpm - callable with a single argument of wavelength
           (in um) that produces a complex reflection map of the FPM
     pupil_mask - amplitude map of a pupil mask (i.e., for SPC)
zernikes : ndarray
      vector of Zernike coefficients, same length as dd.seed_zernikes
locam_misfocus : float, optional
      misfocus or misconjugation of the LOWFS camera, millimeters
locam_shear : tuple of (int, int)
      shear in x and y of the image on locam, in oversampled space.
      The model has 8x oversampling at locam, so locam_shear=(1,0)
      shears by 1/8px on the output grid
pool : mapper
      a type which has a map method, multiprocessing thread and
      process pools work, as do concurrent futures equivalents

"""
```


lowfssim-docs3.pdf (page 1 of 6)

LOWFSSim

Brandon D. Dube
August 31, 2021
public version
Jet Propulsion Laboratory,
California Institute of Technology

©2021, California Institute of Technology

1 Introduction

LOWFSSim is a software tool for simulating the Low Order Wavefront Sensing and Control (LOWFSC) system on CGI. It uses python 3.6 or newer. The model is faster than the real LOWFS system that will fly aboard CGI, and has been validated against CGI's performance testbed to better than 0.08%.

The library contains models for each of CGI's three configurations HLC/NFOV, SPC-SPEC, and SPC-WFOV. They are available to the user under a common interface. The model is sufficiently fast that the integrated model including multi-plane diffraction, radiometry, EMCCD noises, and wavefront sensing can be executed in less than 0.5 milli-seconds. At such speed, there is nearly no modeling question too large to be asked and answered.

2 Installation

Python 3.6 or newer is required. You should obtain a copy of the software along with this manual from github.com/nasa-jpl/lowfssim.

The conventional way to install the package is by invoking `pip install .` from the lowfssim directory, which contains the setup.py file. pip will take care of the dependencies (numpy, scipy, astropy, prysm) for you.

You may also "install" it by invoking:

```python
import sys
sys.path.append(r"/path/to/lowfssim")
```

though this is not recommended usage for any python code.

The repository contains both the code and the necessary data files. git-lfs may be needed to download the data files.

3 Usage

Forward modeling and WFS are separate activities in lowfssim. We will cover the former first, as it is a pre-requesite for the latter.

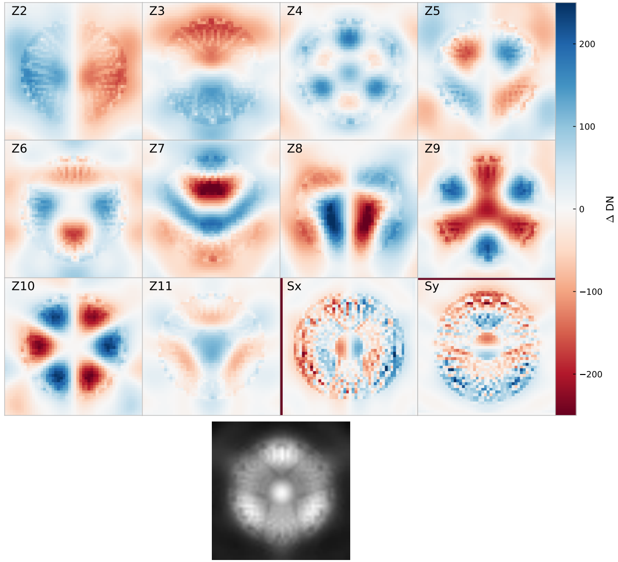
3.1 Forward Modeling

Before the model can be run, the software must be imported and dependencies must be loaded into memory:

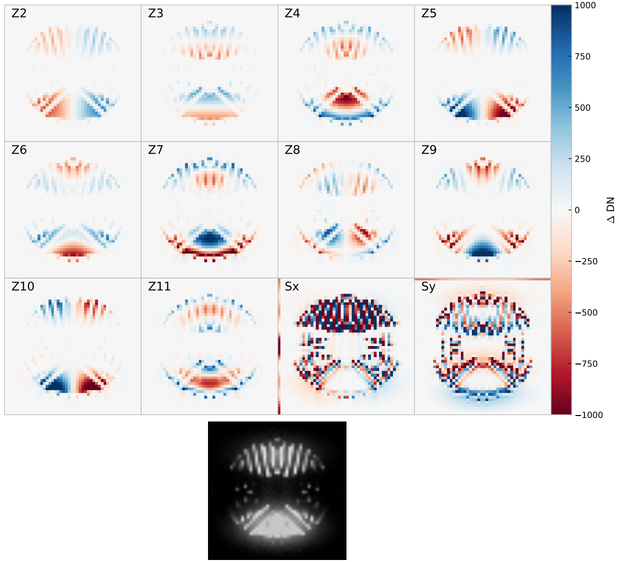
1

Morphological Response at LOCAM to Zernike inputs

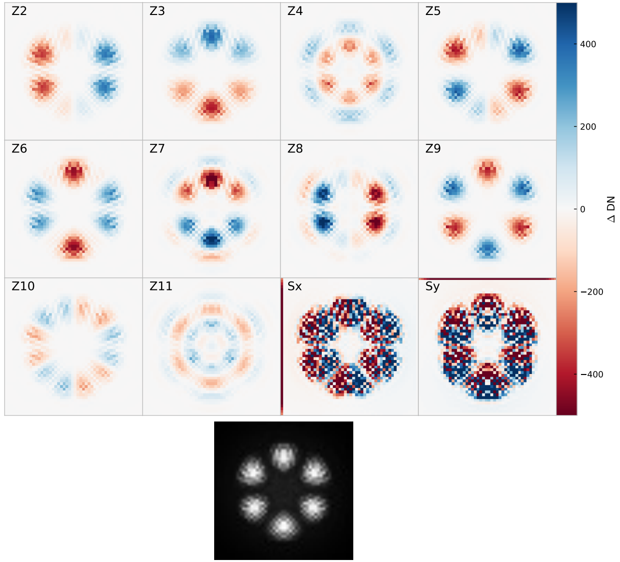
HLC



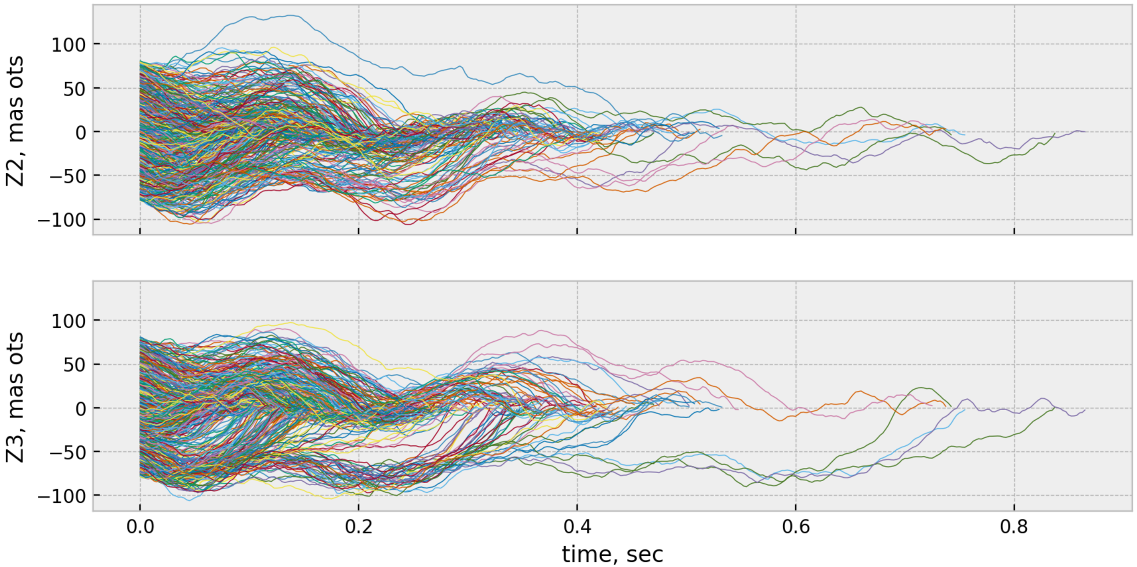
SPC-SPEC



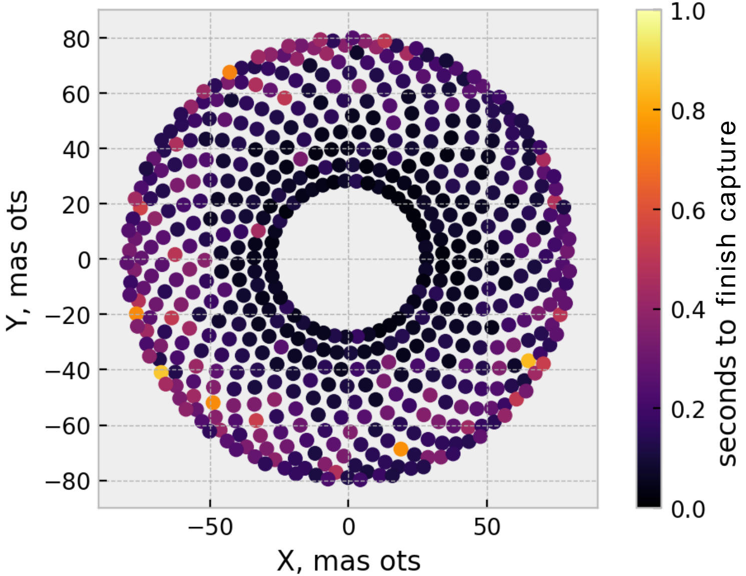
SPC-WFOV



Closed loop LOS capture and control under dynamic perturbation



559 trials super-imposed; millions of model evaluations lowfssim can complete about 3 of these trials per second in real time

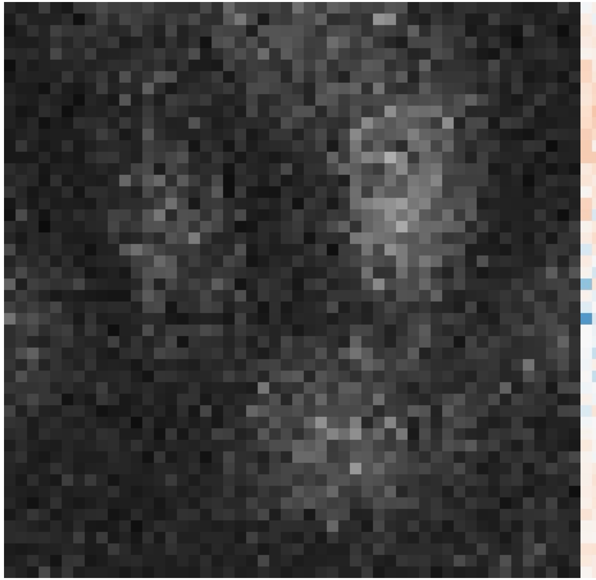


The starting points, color coded by how long it took for LOS capture to conclude

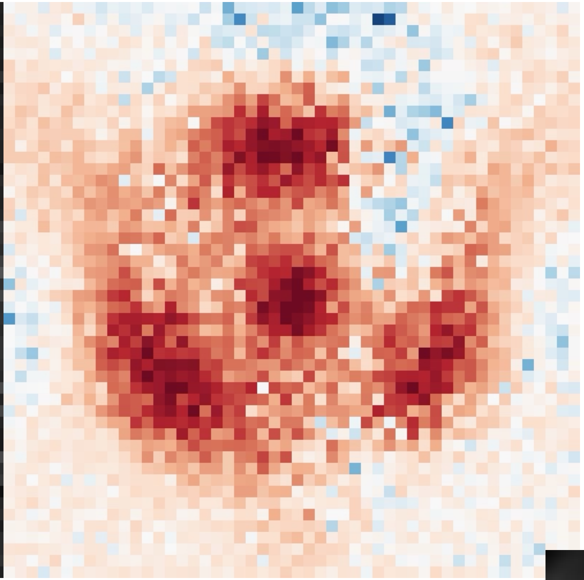
 mas ots = milliarcseconds on-the-sky

Timeseries of LOCAM frames during Target Star capture

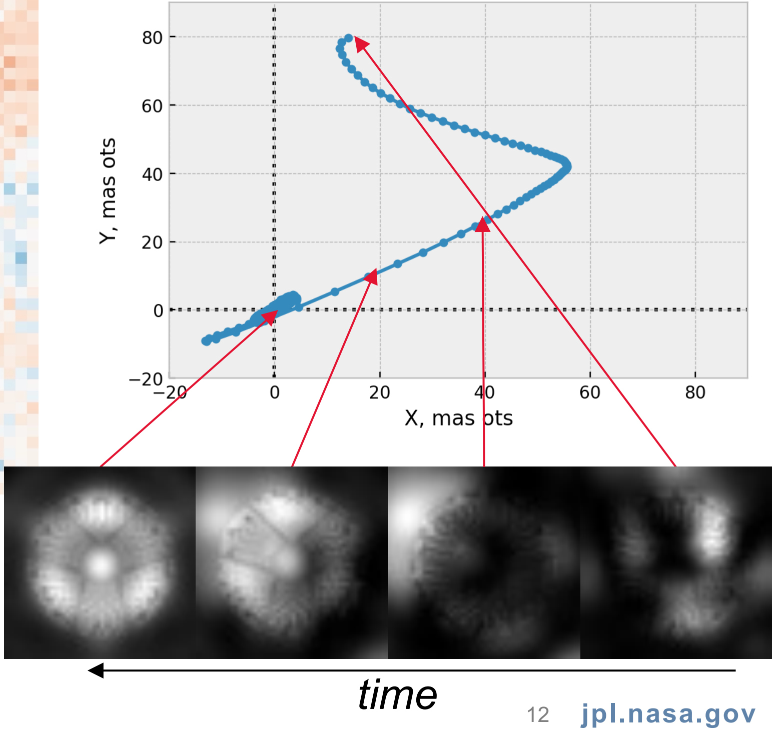
Raw Images



Differential Images



Followed LOS trajectory

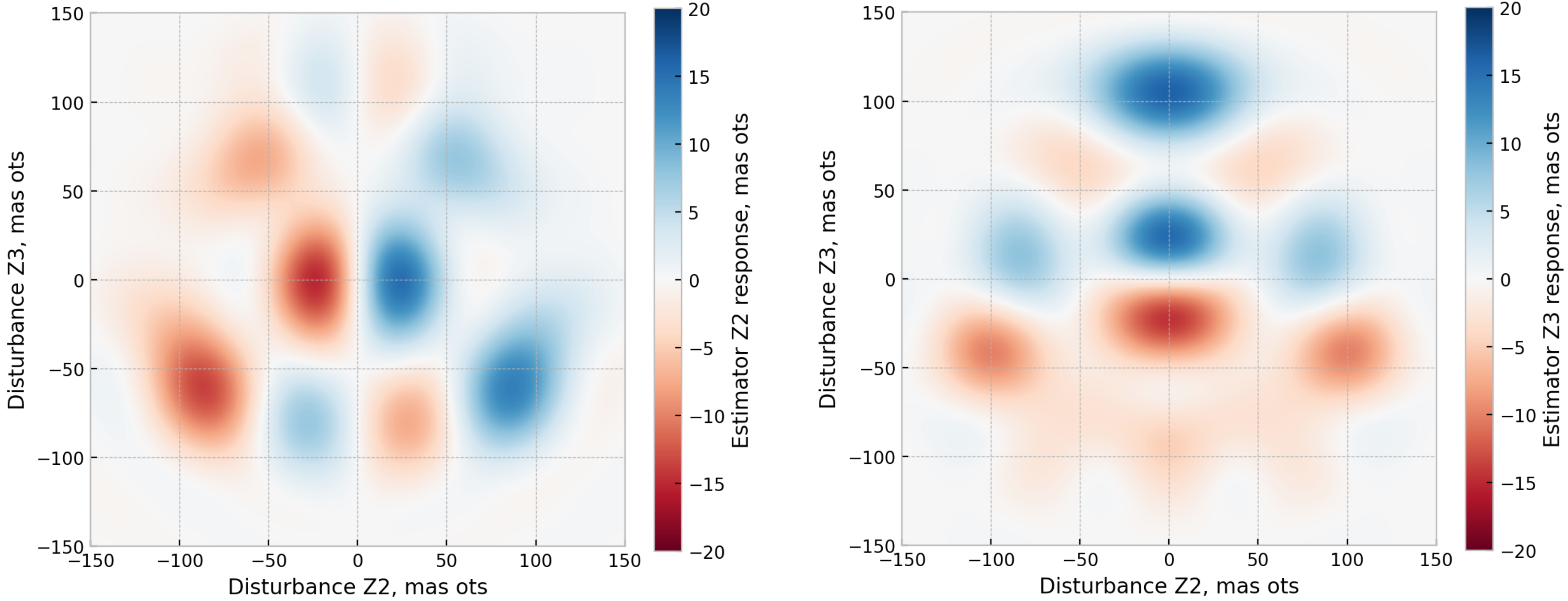


The ideal differential image is pure white

the Target Star is 47 Uma

These videos advances 80x slower than real-time

2D Response of LOWFS Estimator to Input Z2, Z3



These graphics show the response of the LOWFS estimator in Z2 and Z3 to a grid of real Z2, Z3 inputs. It shows the unusual response of the estimator over a large dynamic range (~3/4λ RMS).

Intermission

The proceeding slides introduce LOWFS and LOWFSSim and some interesting uses of the model.

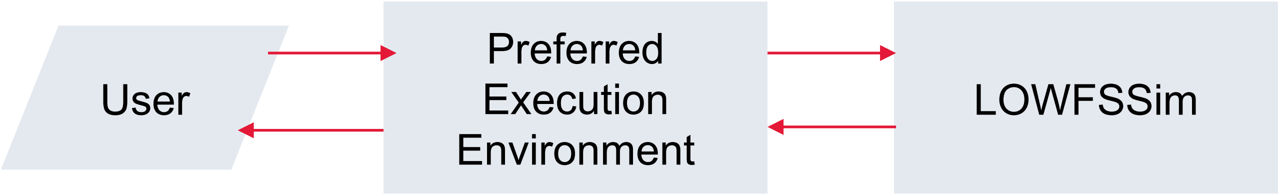
The following slides are something of a tutorial on using the model.

Any **Questions** on the content up-to-now?

Examples

1. A 2D map of the Z2, Z3 estimates over an extended dynamic range
2. Reference/Target star offsets
3. Line of Sight capture and closed-loop control of Z2/Z3

Sidebar: the User's Environment



Jupyter
SLURM
Python REPL
(Your own GUI?)

✅ Laptop
✅ Workstation
✅ Server
✅ Supercomputer
✅ Interactive execution
❌ Define-then-run config files (but you could wrap LOWFSSim with config files)
❌ High level of ceremony involved in running the model

Driving LOWFSSim: setup

This wall of imports reflects that part of lowfssim's speed comes from bringing everything to the top level; there is no system inside the model that forces the pieces to go together in a certain way.

```
from pathlib import Path

import numpy as np

from lowfsc import props, control
from lowfsc.data import DesignData
from lowfsc.spectral import StellarDatabase, LOWFS_BANDPASS, ThroughputDatabase
from lowfsc.automate import flt_chop_seq
from lowfsc.emccd import EMCCD
```

These lines of code configure everything for the GPU and single precision floats; for CPU, skip them

```
from matplotlib import pyplot as plt

from prysm.conf import config
from prysm.fttools import mdft

import cupy as cp
from cupyx.scipy import fft as cpfft

from prysm.mathops import np as pnp, fft as pfft

mdft.clear()
pfft._srcmodule = cpfft
pnp._srcmodule = cp
config.precision = 32

root = Path('~/proj/wfirst/lowfs/data')
```

Driving LOWFSSim: calibration

...still not quite done loading things; this creates the camera and box that holds the DM phase screens, Roman pupil, the focal plane mask, etc. “wt” is the Zernike weights, and will be reused over and over again

```
dd = DesignData.hlc_design(root)

cam = EMCCD.cgi_camera()
dd.seed_zernikes(range(2,12))
wt = pnp.zeros(10)
```

This chops the estimator with 5 nm each of Z2..Z11. The chop “step” size is important to LOWFS performance and should not be changed. Non-diagonal elements of this matrix can be used to move multiple Zernike modes at once during calibration and model how these produce sensing errors.

Gains are the scalars used to calibrate each Zernike mode’s linear responsivity.

refz is where in Zernike space the estimator is calibrated to and becomes the zero point for differential estimates and control.

```
refz = wt.copy()
refz[:] = 0
chops = cp.eye(10, dtype=refz.dtype)*5
gains = cp.diag(chops)
```

refst and targst are the reference and target stars.
(Vmag, spectral type, EM gain)
wref, wtarg are the spectral weights for reference and target stars. R is the LOWFS estimator.

```
refst = (2.25, 'g0v', 20)
targst = (5, 'g0v', 150)

d = flt_chop_seq(wvl, dd, refz, chops, gains,
ref=refst, targ=targst, cam=cam)

wref, wtarg, R = d['wref'], d['wtarg'], d['R']
```

Driving LOWFSSim: images, estimates

A useful test that you have not made a mistake is to make sure the estimator recognizes its chops as themselves (5 nm, small off-diagonal terms).

The estimator returns 17 things in a vector, python indices 1:11 are Z2..Z11. The PDF docs enumerate the meaning of each index.

Immediately below for, im has units of e-/sec. After cam.expose it is an average of 10,000 images with units of DN.

If you look at the source code, `props.polychromatic` does very little other than the actual mechanics of diffraction. It does not draw any masks, compute any grids, etc. This is a significant component of lowfssim's speed.

The key computation performance principle used by LOWFSSim is "load inputs once, run many times"

```
np.set_printoptions(suppress=True, precision=2, linewidth=120)
cam.em_gain = targst[2]
for i, row in enumerate(chops):
    im = props.polychromatic(wvl, wtarg, dd, zernikes=row)
    im = cam.expose(im, 10_000).mean(axis=0, dtype=im.dtype)
    e = R.estimate(im)
    resprow = e[1:11]
    print(resprow)
[ 4.97  0.01  0.08 -0.02 -0.    0.05  0.01 -0.04  0.01  0.06]
[-0.01  4.94 -0.04 -0.02  0.01 -0.02  0.02  0.    0.02  0.03]
[ 0.01  0.06  4.79 -0.03 -0.01  0.04 -0.03  0.06  0.01  0.14]
[ 0.03  0.07 -0.05  4.97  0.03 -0.   -0.    0.04 -0.02 -0.  ]
[ 0.08 -0.07  0.05  0.03  4.95  0.    0.01  0.02 -0.04  0.05]
[ 0.03 -0.02 -0.01 -0.01  0.12  5.04 -0.01  0.05 -0.02  0.11]
[ 0.01  0.01 -0.01 -0.01 -0.12 -0.03  5.    0.09 -0.03  0.1 ]
[-0.01  0.    0.04 -0.01 -0.01  0.05 -0.01  5.01  0.    0.07]
[-0.04 -0.06  0.06  0.02 -0.04  0.05 -0.02  0.    5.   -0.06]
[-0.    0.01  0.08 -0.01  0.02  0.05 -0.05  0.05 -0.01  4.96]
```

Computational Performance

On an Nvidia A100 GPU, LOWFSSim is able to model the system in closed loop at ~2.2kHz in real-time. **All times in ms.**
0.19 and 0.20 refer to prysm versions. Public lowfssim = v0.20. 0.19 is a legacy version at JPL.

machines at JPL: Laptop (CPU), Large server (CPU), Large server (GPU)

Texas Advanced Computing Center (Supercomputer): Exascale server (GPU)

HLC

Version / environment	Laptop (CPU)	Large server (CPU)	Large server (GPU)	Exascale server (GPU)
0.19	446	588	8.35	2.41
0.20	58.7	46.2	1.89	0.43
Speedup	7.6x	12.7x	4.4x	5.6x

Net 1037x

1.2x faster → 24x faster → 4.3x faster

31x faster

137x faster

SPEC WFOV

Version / environment	Laptop (CPU)	Large server (CPU)	Large server (GPU)	Exascale server (GPU)
0.19	1690	260	26.4	4.32
0.20	193	46.7	2.85	0.51
Speedup	8.75x	5.6x	9.26x	8.5x

Net 3314x

4.1x faster → 16.4x faster → 5.6x faster

67.7x faster

378x faster

lowfssim can be accelerated >1000x by using GPUs

Conclusions

LOWFSSim is a high fidelity model of LOWFS and represents the state of the art in diffraction modeling and is now available to the public to use and modify to model the next great observatories.

LOWFS itself represents the state of the art in low order wavefront sensing and control.

LOWFS will soon be in space, stabilizing CGI and enabling exoplanet science.

Questions?

Jet Propulsion Laboratory
California Institute of Technology

jpl.nasa.gov