LOWFSSim
Integrated Model of LOWFS

Brandon D. Dube
Optical Engineer,
Optical Simulation and Analysis Group
November 11, 2021

brandon.dube@jpl.nasa.gov
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:

<table>
<thead>
<tr>
<th>Channels</th>
<th>Bandwidth (Hz)</th>
<th>Sample Rate (Hz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z2/Z3</td>
<td>20</td>
<td>1000</td>
</tr>
<tr>
<td>Z4-Z11</td>
<td>0.0016</td>
<td>0.1</td>
</tr>
</tbody>
</table>

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
# 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

    """
Morphological Response at LOCAM to Zernike inputs
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

11/11/21 These videos advances 80x slower than real-time
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

- User
- Preferred Execution Environment
  - Jupyter
  - SLURM
  - Python REPL
  - (Your own GUI?)
- LOWFSSim

✅ 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.

```python
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

from matplotlib import pyplot as plt
from prysm.conf import config
from prysm.ftools 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')
```

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

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.

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.
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”
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.

<table>
<thead>
<tr>
<th>Version / environment</th>
<th>Laptop (CPU)</th>
<th>Large server (CPU)</th>
<th>Large server (GPU)</th>
<th>Exascale server (GPU)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HLC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.19</td>
<td>446</td>
<td>588</td>
<td>8.35</td>
<td>2.41</td>
</tr>
<tr>
<td>0.20</td>
<td>58.7</td>
<td>46.2</td>
<td>1.89</td>
<td>0.43</td>
</tr>
<tr>
<td>Speedup</td>
<td>7.6x</td>
<td>12.7x</td>
<td>4.4x</td>
<td>5.6x</td>
</tr>
<tr>
<td>SPEC WFOV</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.19</td>
<td>1690</td>
<td>260</td>
<td>26.4</td>
<td>4.32</td>
</tr>
<tr>
<td>0.20</td>
<td>193</td>
<td>46.7</td>
<td>2.85</td>
<td>0.51</td>
</tr>
<tr>
<td>Speedup</td>
<td>8.75x</td>
<td>5.6x</td>
<td>9.26x</td>
<td>8.5x</td>
</tr>
</tbody>
</table>

Net 1037x

Net 3314x

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?