AOloopControl

Olivier Guyon

Aug 9, 2016

Contents

1 Overview

1.1 Scope

AO loop control package

1.2 Usage

Scripts to run the software are located within the source code directory:

`./src/AOloopControl/scripts/`

The scripts can be linked to your working directory by executing the following command:

```
ln -s $PWD/syncscripts /myworkdirectory/syncscripts
```

Then, execute in your work directory:

```
./syncscripts
```

This will install all required scripts in workdirectory and install any packages required.

The main script is

```
./aolconf
```

1.3 Supporting scripts, aolconfscripts directory

Scripts in the `aolconfscripts` directory are part of the high-level ASCII control GUI

Script	Description
aolconf__DMfuncs	DM functions
aolconf__DMturb	DM turbulence functions
aolconf__funcs	Misc functions
aolconf__logfuncs	data and command logging
aolconf__menuconfigureloop	configure loop menu
aolconf__menucontrolloop	control loop menu
aolconf__menucontrolmatrix	control matrix menu
aolconf__menu__mkFModes	Make modes
aolconf__menurecord	
aolconf__menutestmode	Test mode menu
aolconf__menutop	Top level menu
aolconf__menuview	Data view menu
aolconf__readconf	Configuration read functions
aolconf__template	Template (not used)

1.4 Supporting scripts, auxscripts directory

Scripts in the `auxscripts` directory are called by aolconf to perform various tasks.

Script	Description
acquRespM	Acquire response matrix

1.5 Hardware simulation scripts

SCripts in the `aohardsim` directory are called to simulate hardware for testing / simulations

Script	Description
aosimDMstart	Start simulation DM shared mem
aosimDMrun	Simulates physical deformable mirror (DM)
aosimmkWF	creates properly sized wavefronts from pre-computed wavefronts
aosimWPyrFS	Simulates WFS

2 Hardware Simulation

2.1 Overview

The AOsim simulation architecture relies on individual processes that simulate subsystems. Each process is launched by a bash script. ASCII configuration files are read by each process. Data I/O can be done with low latency using shared memory and semaphores: a process operation (for example, the wavefront sensor process computing WFS signals) is typically triggered by a semaphore contained in the shared memory wavefront stream. A low-speed file system based alternative to shared memory and semaphores is also provided.

2.2 Processes and scripts: main WF control loop

2.2.1 Process `aosimmkWF`

`aosimmkWF` reads precomputed wavefronts and formats them for the simulations (pixel scale, temporal sampling).

Parameters for `aosimmkWF` are stored in configuration file:

File `aosimmkWF.conf.default` :

```
!INCLUDE "../scripts/aohardsim/aosimmkWF.conf.default"
```

2.2.2 Process `aosimDMrun`

File `aosimDMrun.conf.default` :

```
!INCLUDE "../scripts/aohardsim/aosimDMrun.conf.default"
```

2.2.3 Process `aosimPyrWFS`

File `aosimPyrWFS.conf.default` :

```
!INCLUDE "../scripts/aohardsim/aosimPyrWFS.conf.default"
```

2.3 AO loop control

The `aolconf` script is used to configure and launch the AO control loop. It can be configured with input/output from real hardware or a simulation of real hardware.

2.3.1 Shared memory streams

Script	Description
wf0opd	Wavefront OPD prior to wavefront correction

Script	Description
wf1opd	Wavefront OPD after correction (=wf0opd-2xdm05dispmap)
dm05disp	DM actuators positions
dm05dispmap	DM OPD map
WFSinst	Instantaneous WFS intensity
pWFSint	WFS intensity frame, time averaged to WFS frame rate and sampled to WFS camera pixels

2.3.2 Hardware simulation architecture

Close-loop simulation requires the following scripts to be launched to simulate the hardware, in the following order :

- `aosimDMstart`: This script creates DM channels (uses dm index 5 for simulation). Shared memory arrays `dm05disp00` to `dm05disp11` are created, along with the total displacement `dm05disp`. Also creates the `wf1opd` shared memory stream which is needed by `aosimDMrun` and will be updated by runWF. `wf1opd` is the master clock for the whole simulation, as it triggers DM shape computation and WFS image computation.
- `aosimDMrun`: Simulates physical deformable mirror (DM)
- `aosimmkWF`: Creates atmospheric wavefronts
- `aosimWFS`: Simulates WFS

Some key script variables need to coordinated between scripts. The following WF array size should match :

- `WFsize` in script `aosimDMstart`
- `ARRAYSIZE` in `aosimmkWF.conf`
- `ARRAYSIZE` in `aosimDMrun.conf`

The main hardware loop is between `aosimmkWF` and `aosimWFS`: computation of a wavefront by `aosimmkWF` is *triggered* by completion of a WFS instantaneous image computation by `aosimWFS`. The configuration files are configured for this link.

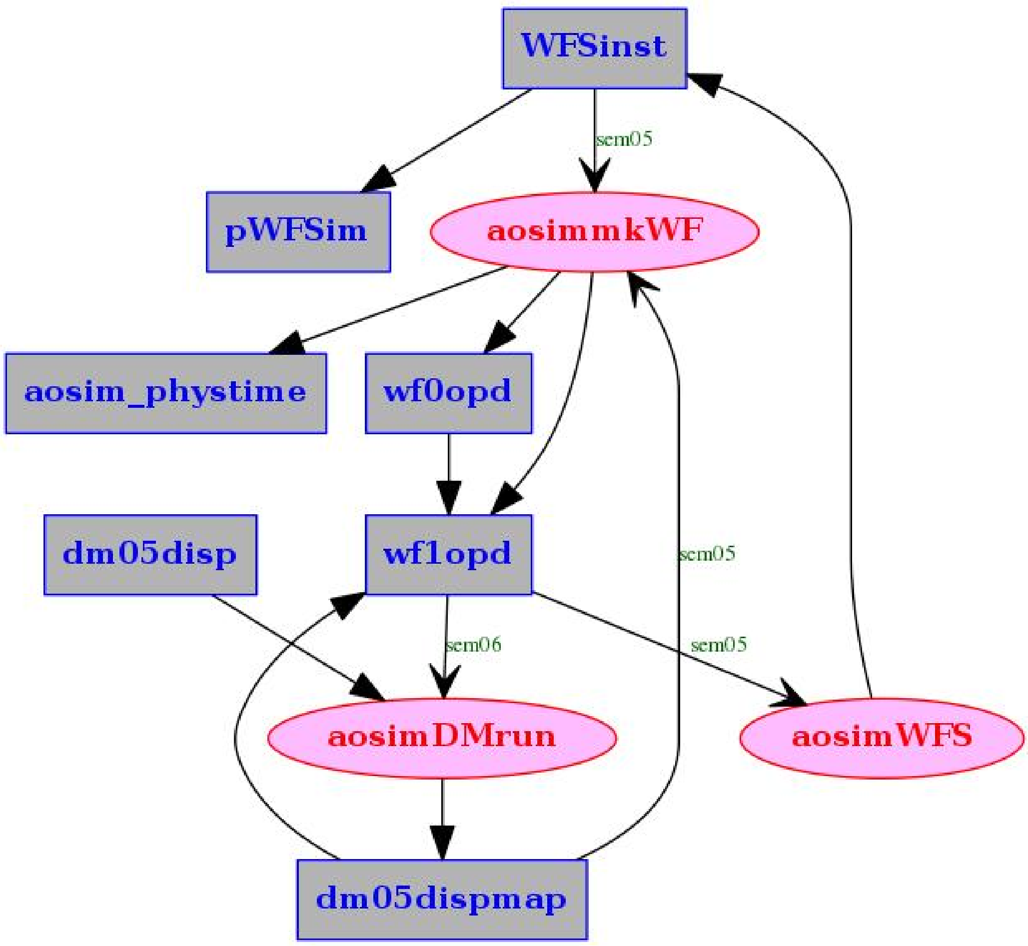

Figure 1: data flow

2.3.3 DM temporal response

The DM temporal response is assumed to be such that the distance between the current position p and desired displacement c values is multiplided by coefficient $a < 1$ at each time step dt. The corresponding step response is :

$c - p((k+1)dt) = (c - p(kdt))a$

$c - p(kdt) = (c - p0)a^k$

$p(kdt) = 1 - a^k$

The corresponding time constant is

$a^{\frac{t0}{dt}} = 0.5$

$\frac{t0}{dt} ln(a) = ln(0.5)$

$ln(a) = ln(0.5)dt/t0$

$a = 0.5^{\frac{dt}{t0}}$

2.4 Processes and scripts: system ouput

The output (corrected) wavefront is processed to compute ouput focal plane images, and optionally LOWFS image.

2.4.1 Process `aosimcoroLOWFS`

Computes coronagraphic image output and LOWFS image

File `aosimcoroLOWFS.conf.default`:

```
!INCLUDE "../scripts/aohardsim/aosimcoroLOWFS.conf.default"
```

2.4.2 Ouput simulation architecture

3 AOloopControl setup

3.1 Files

SM = shared memory

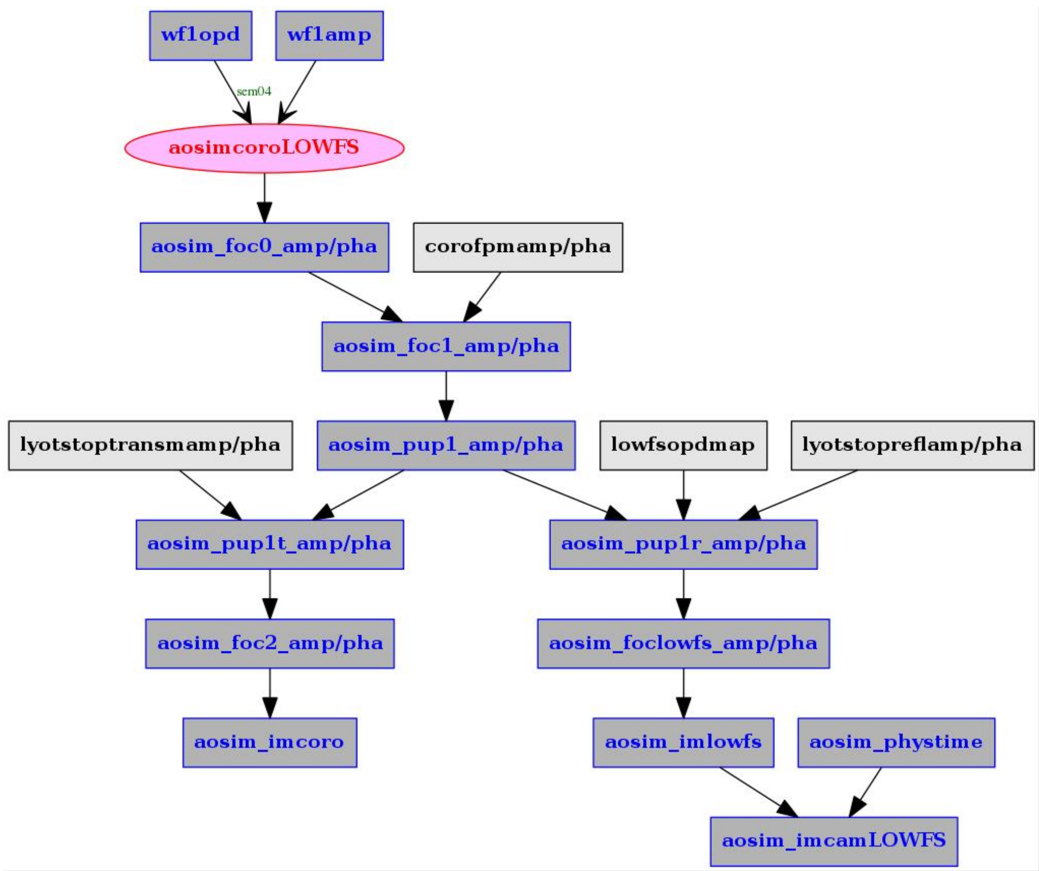

Figure 2: coroLOWFS data flow

File	stream	Description
./conf/HRM_DMmask.fits		DM mask to construct Hadamard RM pokes, created by **auxscripts/mkHpoke** if not present

3.2 GUI description

The script `aolconf` starts the main GUI, from which all setup and control can be done. The GUI consists of several main screens, as shown below.

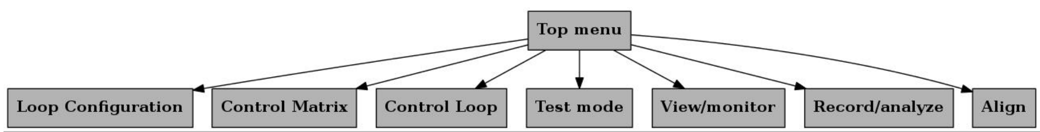

Figure 3: aolconf GUI screens

3.3 Setting up the hardware interfaces

- start aolconf with loop number and loop name (you can ommit these arguments when launching the script again):

```
aolconf -L 3 -N testsim
```

- **Set DM number** (`S` command in `Top Menu` screen). If the DM stream exists, you should see its x and y size in the two lines below. If not, you will need to enter the desired DM size and create the DM stream with the `initDM` command in the `Top Menu`.

- **autoconfigure streams** (`nolink` in `Top Menu` screen). This command automactically sets up the following symbolic links:

  - dm##disp03 is linked to aol#_dmC (loop dm control channel)
  - dm##disp00 is linked to aol#_dmO (flat offset channel)

  - dm##disp04 is linked to aol#_dmZP0 (zero point offset 0 actuation channel)
  - dm##disp05 is linked to aol#_dmZP1 (zero point offset 1 actuation channel)
  - dm##disp06 is linked to aol#_dmZP2 (zero point offset 2 actuation channel)
  - dm##disp07 is linked to aol#_dmZP3 (zero point offset 3 actuation channel)
  - dm##disp08 is linked to aol#_dmZP4 (zero point offset 4 actuation channel)
  - dm##disp is linked to aol#_dmdisp (total dm displacement channel)
  - dm##disp02 is linked to aol#_dmRM (response matrix actuation channel)

- **load Memory** (`M` in `Top Menu` screen). The dm performs the symbolic links to the DM channels.
- **link to WFS camera** (`wfs` to `Loop Configuration` screen). Select the WFS shared memory stream.

3.4 Acquiring a zonal response matrix

- **set response matrix parameters** in `Loop Configure` screen: amplitude, time delay, frame averaging, excluded frames
- **set normalization and Hadmard modes** in `Loop Configure` screen. Normalization should probably be set to 1.
- **start zonal response matrix acquisition** (`zrespon` in `Loop Configure` screen). The process runs in tmux session aol#zrepM.
- **stop zonal response matrix acquistion** (`zrespoff` in `Loop Configure` screen).

The following files are then created:

File	Archived location	Description
zrespmat.fits	zrespM/zrespM_${datestr}.fits	zonal response matrix
wfsref0.fits	wfsref0/wfsref0_${datestr}.fits	WFS reference (time-averaged image)
wfsmap.fits	wfsmap/wfsmap_${datestr}.fits	Map of WFS elements sensitivity
dmmap.fits	dmmap/dmmap_${datestr}.fits	Map of DM elements sensitivity
wfsmask.fits	wfsmask/wfsmask_${datestr}.fits	WFS pixels mask, derived from wfsmap
dmmaskRM.fits	dmmaskRM/dmmaskRM_${datestr}.fits	DM actuators mask, derived from dmmap by selecting actuators with strong response
dmslaved.fits	dmslaved/dmslaved_${datestr}.fits	slaved DM actuators: actuators near active actuators in dmmaskRM
dmmask.fits	dmmask/dmmask_${datestr}.fits	DM mask: all actuators controlled (union of dmmaskRM and dmslaved)

Note that at this point, the files are NOT loaded in shared memory, but the archieved file names are stored in "conf/conf_.txt" for future loading.

- **Load zrespm files into shared memory** (`SMloadzrm` in `Loop Configure` screen)

3.5 Acquiring a modal response matrix (optional)

In addition to the zonal response matrix, a modal response matrix can be acquired to improve sensitivity to low-oder modes.

To do so:

- activate `RMMon` to **toggle the modal RM on**.
- **select RM amplitude and maximum cycles per aperture (CPA)**

- **start the acquisiton** (`LOresp_on`)
- **stop the acquisiton** (`LOresp_off`)

The following files are then created:

File	Archived location	Description
LOrespmat.fits	LOrespM/LOrespM_${datestr}.fits	Modal response matrix
respM_LOmodes.fits	LODMmodes/LODMmodes_${datestr}.fits	Low-order modes
LOwfsref0.fits	LOwfsref0/LOwfsref0_${datestr}.fits	WFS reference measured during LO RM acquisition
LOwfsmap.fits	LOwfsmap/LOwfsmap_${datestr}.fits	Map of WFS elements sensitivity
LOdmmap.fits	LOdmmap/LOdmmap_${datestr}.fits	Map of DM elements sensitivity
LOwfsmask.fits	LOwfsmask/LOwfsmask_${datestr}.fits	WFS pixel mask, derived from wfsmap
LOdmmask.fits	LOdmmask/LOdmmask_${datestr}.fits	DM actuator mask, derived from dmmap by selecting actuators with strong response

Note that at this point, the files are NOT loaded in shared memory, but the archieved file names are stored in "conf/conf_.txt" for future loading.

- **Load LOrespm files into shared memory** (`SMloadmrm` in `Loop Configure` screen)

3.6 Building control matrix

- **set SVDlimit** (`SVDla` in `Control Matrix` screen). Set value is 0.1 as a starting point for a stable loop.
- **perform full CM computation** (`mkModes0` in `Control Matrix` screen). Enter first the number of CPA blocks you wish to use. Computation takes a few minutes, and takes place in tmux session `aol#mkmodes`.

The following files are created:

File	Archived location	Description
aolN_DMmodes	DMmodes/DMmodes_${datestr}.fits	DM modes
aolN_respM	respM/respM_${datestr}.fits	WFS response to DM modes

Block-specific files:

File	Archived location	Description
aolN_DMmodesbb	DMmodes/DMmodesbb_${datestr}.fits	DM modes for block bb
aolN_respMbb	respM/respMbb_${datestr}.fits	WFS response to DM modes for block bb
aolN_contrMbb.fits	contrM/contrMbb_${datestr}.fits	Control matrix for block bb
aolN_contrMcbb.fits	contrMc/contrMcbb_${datestr}.fits	Collapsed control matrix for block bb
aolN_contrMcactbb.fits	contrMcact/contrMcactbb_${datestr}.fits	Collapsed control matrix for block bb, only active actuators

Note that at this point, the files are NOT loaded in shared memory, but the archieved file names are stored in "conf/conf_.txt" for future loading.

- **Load CM files into shared memory** (`SMloadCM` in `Control Matrix` screen)

3.7 Running the loop: Choosing hardware mode (CPU/GPU)

There are multiple ways to perform the computations on CPU and/or GPUs. The main 3 parameters are:

- **GPU** : 0 if matrix multiplication(s) done on CPU, >0 for GPU use
- **CMmode** : 1 if using a combined matrix between WFS pixels and DM actuators, skipping intermediate computation of modes
- **GPUall** : if using GPUall, then the WFS reference subtraction is wrapped inside the GPU matrix multiplication

GPU	CMmode	GPUall	Description
>0	ON	ON	dark-subtracted WFS frame imWFS0 is multiplited by collapsed control matrix (only active pixels). normalization and WFS reference subtraction are wrapped in this GPU operation as subtraction of pre-computed vector output. This is the fastest mode.
>0	ON	OFF	WFS reference is subtracted from imWFS0 in CPU, yielding imWFS2. imWFS2 is multiplied by control matrix (only active pixels) in GPU.
>0	OFF	OFF	WFS reference is subtracted from imWFS0 in CPU, yiedling imWFS2. imWFS2 is multiplied (GPU) by control matrix to yield mode values. Mode coefficients then multiplied (GPU) by modes.

4 Virtual DM (dm-to-dm link)

In this mode, the AO loop controls a virtual DM. The virtual actuators are correspond to modes controlling the zero point offset of another loop. In this section, I assume that **loopA** is the main loop (directly controls a physical DM) and that **loopB** is the virtual loop.

5 Predictive control (experimental)

5.1 Scripts

File	Description
aolARPF	find auto-regressive predictive filter
aolARPFblock	AO find optimal AR linear predictive filter