AOloopControl

Olivier Guyon

Feb 21, 2017

Contents

1 Overview

1.1 Scope

AO loop control package

1.2 Installing the AdaptiveOpticsControl package

Source code is available on the AdaptiveOpticsControl git hub repository.

Download the latest tar ball (.tar.gz file), uncompress, untar and execute in the source directory (`./AdaptiveOpticsControl-<version>/`)

```
./configure
```

Include recommended high performance compile flags for faster execution speed:

```
./configure CFLAGS='-Ofast -march=native'
```

If you have installed CUDA and MAGMA libraries:

```
./configure CFLAGS='-Ofast -march=native' --enable-cuda --enable-magma
```

The executable is built with:

```
make
make install
```

The executable is `./AdaptiveOpticsControl-<version>/bin/AdaptiveOpticsControl`

1.3 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 from `src/AOloopControl/scripts/`:

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

Then, execute in your work directory:

```
./syncscripts -s <SRCdir>
```

where is the source directory for scripts. This will install all required scripts in workdirectory and install any packages required.

The main script is

```
./aolconf
```

1.4 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

Script	Description
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)

1.5 Supporting scripts, auxscripts directory

Scripts in the `auxscripts` directory are called by aolconf to perform various tasks. To list all commands, type in the `auxscripts` directory :

```
./listcommands
```

The available commands are listed in the table below.

`./mkHpoke`	`Compute real-time WFS residual image`
`./aolMeasureTiming`	`Measure loop timing`
`./aolCleanLOrespmat`	`Measure zonal resp matrix`
`./aolRMmeas_sensitivity`	`Measure photon sensitivity of zonal response matrix`
`./aolmon`	`Display AO loop stats`
`./acquRespM`	`Acquire response matrix`
`./aolctr`	`AO control process`
`./listrunproc`	`List running AOloop processes`
`./MeasDMmodesRec`	`Measure AO loop DM modes recovery`
`./aolARPFblock`	`AO find optimal AR linear predictive filter (single blo`
`./aolRM2CM`	`Align Pyramid camera`
`./aolCleanZrespmat`	`Cleans zonal resp matrix`
`./mkDMslaveActprox`	`Create DM slaved actuators map`
`./xptest`	`Compute cross-product of a data cube`
`./aolInspectDMmap`	`Inspect DM map`
`./aolARPFautoUpdate`	`Automatic update of AR linear predictive filter`
`./aolCleanZrespmat2`	`Cleans zonal resp matrix`
`./mkDMslaveAct`	`Create DM slaved actuators map`

./aolReadConfFile	AOloop load file to stream
./aolLinSim	AO Linear Simulator
./aolApplyARPF	Apply AR linear predictive filter
./aolARPF	AO find optimal AR linear predictive filter
./aolSetmcLimit	Compute real-time WFS residual image
./aolautotunegains	Automatic gain tuning
./aolmkMasks	Create AO wfs and DM masks
./aolmkmodesM	CREATE CM MODES FOR AO LOOP, MODAL DM
./aolMeasureLOrespmat	Acquire modal response matrix
./waitonfile	Wait for file to appear
./predFiltApplyRT	Apply predictive filter to stream
./aoloffloadloop	DM offload loop
./aolmkWFSres	Compute real-time WFS residual image
./aolWFSresoffloadloop	Compute real-time WFS residual image
./aollindm2wfsim	Convert DM stream to WFS image stream
./aolApplyARPFblock	Apply AR linear predictive filter (single block)
./aolmcoeffs2dmmap	GPU-based MODE COEFFS -> DM MAP
./aolmkmodes	Create modes for AO loop
./aolMeasureZrespmat2	Acquire zonal response matrix
./processTelemetryPSDs	Process telemetry: create open and closed loop PSDs
./aolzploopon	WFS zero point offset loop
./aollinsimDelay	Introduce DM delay
./shmimzero	Set shared memory image stream to zero
./aolmkmodes2	Create modes for AO loop
./alignPyrTT	Align Pyramid TT
./aolgetshmimsize	Get shared memory image size
./aolMeasureZrespmat	Acquire zonal response matrix
./xp2test	Compute cross-product of two data cubes
./waitforfilek	Wait for file to appear and then remove it
./aolmkLO_DMmodes	Create LO DM modes for AO loop
./aolscangain	AO scan gain for optimal value
./aol_dmCave	dmC temporal averaging
./alignPcam	Align Pyramid camera
./aolMeasureLOrespmat2	Acquire modal response matrix
./MeasureLatency	Measure AO system response latency
./aolARPFautoApply	Apply real-time AR linear predictive filter
./aolPFcoeffs2dmmap	GPU-based predictive filter coeffs -> DM MAP
./modesextractwfs	Extract mode values from WFS images

`./Fits2shm`	`Copy FITS files to shared memor`
`./aolblockstats`	`Extract mode values from WFS images, sort per block`
`./aolrun`	`Run AO control loop`
`./aolMergeRMmat`	`Merge HO and LO resp matrices`
`./selectLatestTelemetry`	`Compute real-time WFS residual image`
`./MeasLoopModeResp`	`Measure AO loop temporal response`

1.6 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

There are 3 ways for users to simulate hardware

- Provide an external simulation that adheres to AOloopControl input/output conventions
- Use the physical hardware simulation provided by the package
- Use the linear hardware simulation: this option is fastest, but only captures linear relationships between DM actuators and WFS signals

2.2 Physical hardware simulation

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.

2.2.1 Quick start

You can launch the simulator quickly with the following steps:

- go into directory `aohardsim`
- create symbolic link `atmwf` to atmospheric wavefront simulation directory. For example:

  `ln -s /data/AtmWF/wdir00/ atmwf`
- execute master script './runAOhsim'

2.2.2 Processes and scripts: main WF control loop

2.2.2.1 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"
```

2.2.2.2 Process aosimDMrun

File `aosimDMrun.conf.default` :

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

2.2.2.3 Process aosimPyrWFS

File `aosimPyrWFS.conf.default` :

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

2.2.3 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.

2.2.3.1 Shared memory streams

Script	Description
wf0opd	Wavefront OPD prior to wavefront correction
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

2.2.3.2 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

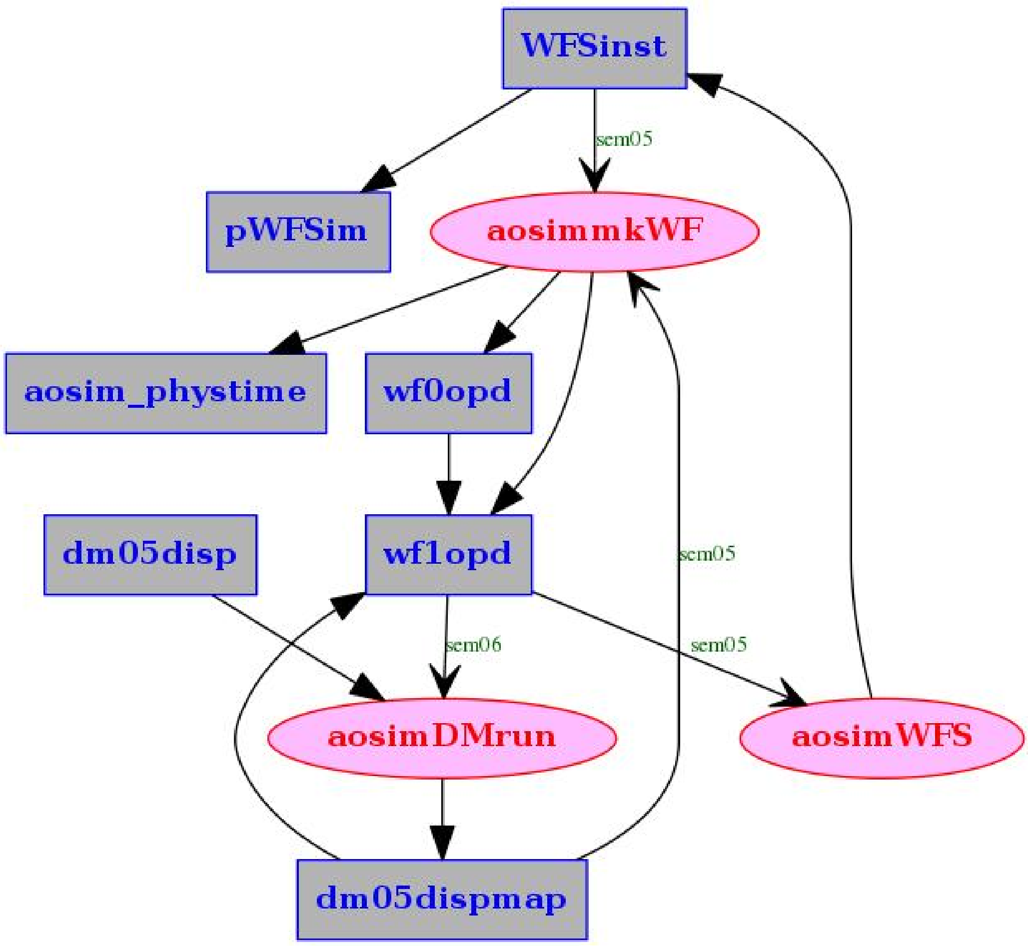

Figure 1: data flow

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.

2.2.3.3 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

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

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

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

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

2.2.4 Processes and scripts: system ouput

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

2.2.4.1 Process `aosimcoroLOWFS`

Computes coronagraphic image output and LOWFS image

File `aosimcoroLOWFS.conf.default`:

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

2.2.4.2 Ouput simulation architecture

2.3 Linear Hardware Simulation

2.3.1 Overview

The Linear Hardware Simulation (LHS) uses a linear response matrix to compute the WFS image from the DM state. It is significantly faster than the Physical Hardware Simulation (PHS).

3 AOloopControl setup

3.1 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.

3.2 Setting up the hardware interfaces

3.2.1 Manual setup

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

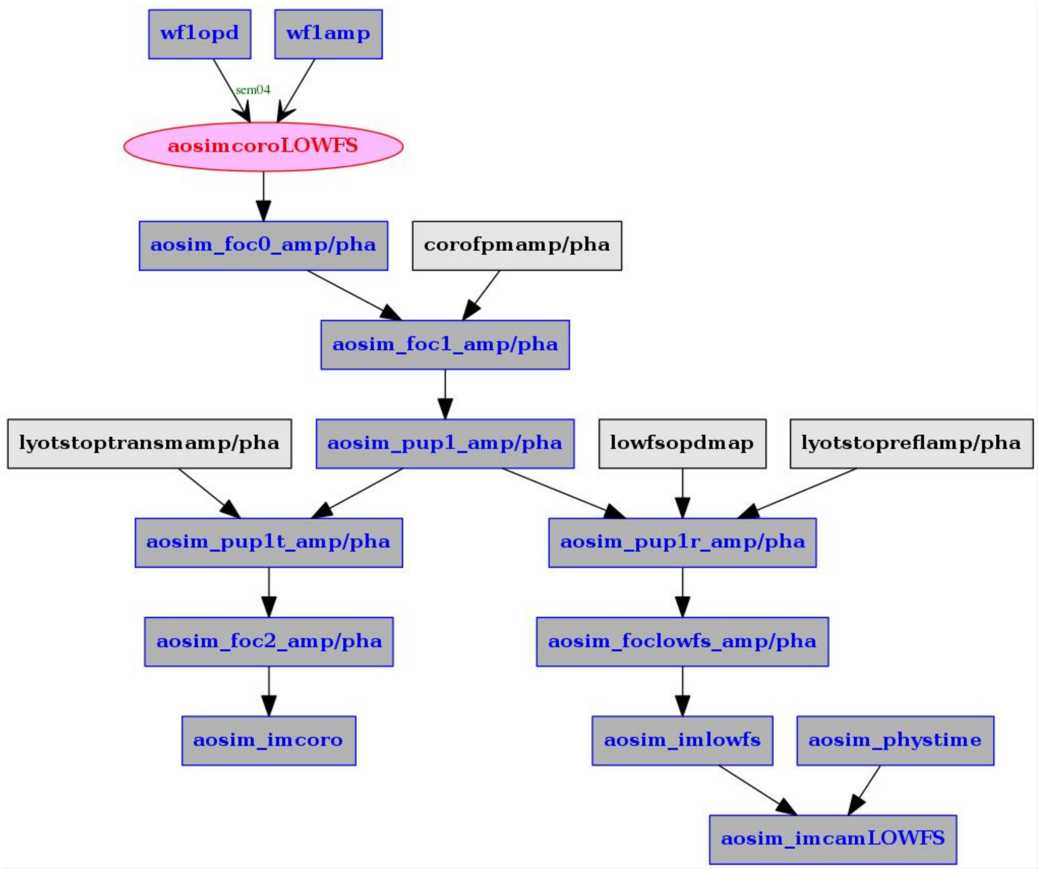

Figure 2: coroLOWFS data flow

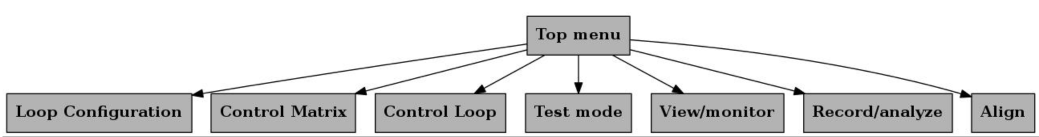

Figure 3: aolconf GUI screens

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

The loop name (`testsim` in the above example) will both allocate a name for the loop and execute an optional custom setup script. The software package comes with a few such pre-made custom scripts for specific systems / examples. When the `-N` option is specified, the custom setup script `./setup/setup_<name>` is ran. The script may make some of the steps described below optional.

- **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 to enter the desired DM size and create the DM stream with the `initDM` command in the `Top Menu`.
- **autoconfigure DM streams** There are two possible setup configurations:
  - **set physical DM** (`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)
  - **set virtual DM**, which is a link to another DM (`dmolink` in `Top Menu` screen)

- **OPTIONAL: set DM delay** ('setDMdelayON' and 'setDMdelayval' in `Top Menu` screen)
- **(Re-)Start DM comb if needed** ('stopDM' and 'initDM' commands in `Top Menu` screen)
- **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.

3.2.2 Setup script

An `aosetup` script may be used to perform all these operations. Inspect the content of directory `aosetup` to see such scripts. You may use or modify as needed. If you use a `aosetup` script, execute it from the working directory, and then start aolconf:

```
./aosetup/aosetup_<myLoop>
./aolconf
```

3.3 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_${datestr}.fits	WFS pixel mask, derived from wfsmap
dmmaskRM.fits	dmmaskRM/dmmaskRM_${datestr}.fits	DM actuator 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 the staging area "conf_zrm_staged/conf_streamname.txt" for future loading.

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

3.4 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 the staging area "conf_mrm_staged//conf_streamname.txt" for future loading.

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

3.5 Automatic system calibration (recommended)

The automatic system calibration performs all steps listed above under zonal and modal response matrix acquisition.

The old calibrations are archived as follows:

- "conf_zrm_staged" and "conf_mrm_staged" hold the new configuration (zonal and modal respectively)
- "conf_zrm_staged.000" and "conf_mrm_staged.000" hold the previous configuration (previously "conf_zrm_staged" and "conf_mrm_staged")
- "conf_zrm_staged.001" and "conf_mrm_staged.001" hold the configuration previously named "conf_zrm_staged.000" and "conf_mrm_staged.000"
- etc for a total of 20 configuration

3.6 Managing configurations

At any given time, the current configuration (including control matrices if they have been computed) can be saved using the `SAVE CURRENT SYSTEM CALIBRATION` command. Saving a configuration will save all files in the conf directory into a user-specified directory.

Previously saved configurations can be loaded with the `LOAD SAVED SYSTEM CALIBRATION` command. This will load saved files into the conf directory and load all files into shared memory.

3.7 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)

3.8 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. This is the number GPUs to use for matrix mult.
- **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	Matrix	Features	Description
>0	ON	ON	contrMcact	fastest no mcoeff	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	contrMcact		WFS reference is subtracted from imWFS0 in CPU, yielding imWFS2. imWFS2 is multiplied by control matrix (only active pixels) in GPU.
>0	OFF	OFF	contrM		MWFS 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.
0	ON	-	contrMcact		imWFS2 is multiplied by control matrix (only active pixels) in CPU
0	OFF	-	contrM		imWFS2 multiplied by modal control matrix

3.9 Auxilliary processes

A number of auxilliary processes can be running in addition to the main loop operation.

3.9.1 Extract WFS modes

Launches script `./auxscripts/modesextractwfs` :

```
!INCLUDE "../scripts/auxscripts/modesextractwfs"
```

Converts WFS residuals into modes.

3.9.2 Extract open loop modes

Launches script C function (CPU-based):

```
key       :    aolcompolm
module    :    AOloopControl.c
info      :    compute open loop mode values
syntax    :    <loop #>
example   :    aolcompolm 2
C call    :   long AOloopControl_ComputeOpenLoopModes(long loop)
```

This function is entirely modal, and assumes that the WFS modes (see section above) are computed. The key input to the function is `aolN_modeval`, the WFS residual mode values. The function uses this telemetry and knowledge of loop gain and mult factor to track open loop mode values.

Optionally, it also includes `aolN_modeval_pC`, the predictive control mode values that are added to the correction in predictive mode.

3.9.3 Running average of dmC

Launches script `./auxscripts/aol_dmCave 0.0005` :

```
!INCLUDE "../scripts/auxscripts/aol_dmCave"
```

3.9.4 Compute and average wfsres

Launches script `./auxscripts/aolmkWFSres 0.0005` :

```
!INCLUDE "../scripts/auxscripts/aolmkWFSres"
```

4 Offsetting

4.1 Overview

Input channels are provided to offset the AO loop convergence point. By default, **DM channels 04, 05, 06, 07, and 08 are dedicated to zero-point offsetting**. The DM channels are sym-linked to `aolN_dmZP0` - `aolN_dmZP7`.

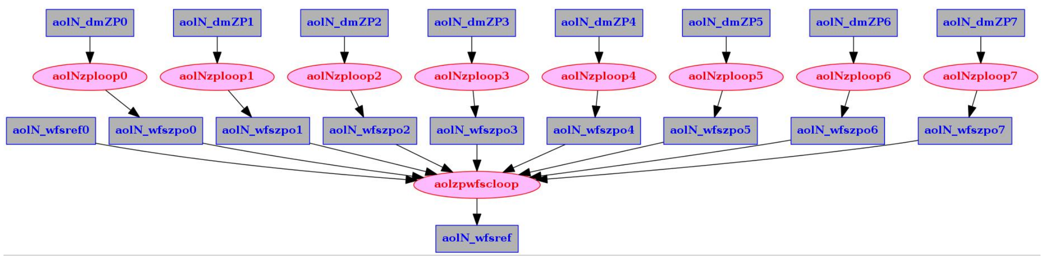

Figure 4: WFS zero point offsetting

4.2 DM offsets

4.2.1 Zonal CPU-based zero point offset

CPU-based zero point offsets will compute WFS offsets from the zero point offset DM channels (04-11) and apply them to the `aolN_wfsref` stream. To activate this features, the user needs to :

- **Toggle the zero point offset loop process ON** (`LPzpo`) prior to starting the loop.

Cfits command `aolzpwfscloop` (C function `AOloopControl_WFSzeropoint_sum_update_loop`) launches a loop that monitors shared memory streams `aolN_wfszpo0` to `aolN_wfszpo3`, and updates the WFS reference when one of these has changed. The loop is running insite tmux session `aolNwfszpo`, and is launched when the loop is closed (`Floopon`) if the loop zero point offset flag is toggled on (`LPzpo`)

- **Activate individual zero point offset channels** (`zplon0` to `zplon4`).

Every time one of the activated DM channel changes, the corresponding wfs `aolN_wfszpo#` zero point offset is CPU-computed.

4.2.2 GPU-based zero point offset

A faster GPU-based zero point offset from DM to WFS is provided for each of the 8 offset channels. GPU-based and CPU-based offsetting for a single channel are mutually exclusive.

4.3 WFS offsets

5 Controlling offsets from another loop

5.1 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.

- Create a separate working directory for **loopB**, allocate a separate loop number and loop name
- Choose DM index number (`S`)
- Select number of **loopA** modes controlled by **loopB**. The number is entered as DM x size (`dmxs` in `Top menu`)
- Enter 1 for DM y size (`dmys` in `Top menu`)
- **Link loop DM to external loop** (`dmolink` in `Top menu`). Select the loop number to link to (**loopA**), and select an offset channel. This will set up several key files:
  - **dm2dmM** : **loopA** modes controlled by **loopB**
  - **dm2dmO** : symbolic link to **loopA** DM channel controlled by **loopB**
  - **dmwrefRM** : **loopA** WFS response to modes controlled by **loopB**
  - **dmwrefO** : **loopA** WFS zero point offset
- **Choose DM or WFS offset mode**. For WFS offset mode (faster), toggle to 1 (`dmwref1`). Note that the DMcomb process will perform the offsetting
- **(Re)-create DM streams and run DMcomb process** (`initDM`)

Commands to the **loopB** DM should now propagate to modal commands to **loopA**..

5.2 Running the loop

The next steps are similar to the ones previously described, with the following important differences:

- The control matrix should be computed in zonal mode (no modal CPA block decomposition)

6 Predictive control (experimental)

6.1 Overview

Predictive control is implemented in two processes:

- The optimal auto-regressive (AR) filter predicting the current state from previous states is computed. The AR filter is computed from open-loop estimates, so the processes computing open-loop telemetry need to be running.
- the AR filter is applied to write a prediction buffer, which can be written asynchronously from the main loop steps.

The predictive filter is modal, and adopts the same modes as the main control loop.

6.2 Scripts

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

7 REFERENCE

7.1 Semaphores

dm00disp	
0	?
6	-> compute simulated linear WFS image

aol0_dmdisp	
0	?
7	-> compute delayed DM : aol0_dmdispD

aol0_imWFS0	
0	?
2	-> extract modes from WFS image script `./auxscripts/modesextractwfs`