The Compute and Control for Adaptive Optics (cacao) real-time control software package

**Arnaud Sevin, Julien Bernard, Damien Gradatour**
Observatoire de Paris. (France)

**Software engineering & development**
Integration with COMPASS simulation environment, RT hardware (GPU, FPGA) and RT processes management/monitoring

**Olivier Guyon, Julien Lozi, Nour Skaf**
Subaru Telescope / SCExAO (US, Japan)

**cacao development and On-sky testing for ExAO applications**

**Hatem Ltaief & Dalal Sukkari**
KAUST (Saudi Arabia)

**HPC expertise**
Develop and provide linear algebra libraries for Machine Learning

**Users / co-developers**
Keck Observatory
Sylvain Cetre

Kernel project/OCA
Frantz Martinache

MagAO-X
Jared Males

Subaru Telescope
Christophe Clergeon

**Facility-class AO (NGS & LGS)**

**Focal plane WFS/C**

**Extreme-AO**

**Facility-class AO (NGS & LGS)**

**Alison Wong & Barnaby Norris**
Machine learning / AI

**Kernel project/OCA**
Frantz Martinache

**MagAO-X**
Jared Males

**Subaru Telescope**
Christophe Clergeon

**Extreme-AO**

**Facility-class AO (NGS & LGS)**

**Users / co-developers**

**Facility-class AO (NGS & LGS)**

**Focal plane WFS/C**

**Extreme-AO**

**Facility-class AO (NGS & LGS)**

**Alison Wong & Barnaby Norris**
Machine learning / AI

**Kernel project/OCA**
Frantz Martinache

**MagAO-X**
Jared Males

**Subaru Telescope**
Christophe Clergeon

**Extreme-AO**

**Facility-class AO (NGS & LGS)**

**Users / co-developers**

**Facility-class AO (NGS & LGS)**

**Focal plane WFS/C**

**Extreme-AO**

**Facility-class AO (NGS & LGS)**

**Alison Wong & Barnaby Norris**
Machine learning / AI

**Kernel project/OCA**
Frantz Martinache

**MagAO-X**
Jared Males

**Subaru Telescope**
Christophe Clergeon

**Extreme-AO**

**Facility-class AO (NGS & LGS)**
cacao’s goals

Support today & tomorrow’s off-the-shelf powerful computing hardware
→ manycore systems (CPU, GPU) and FPGAs
→ high performance computing engine to requirements of large scale AO systems

.. and advanced AO systems and algorithms
→ flexible modular software architecture
→ scalable solution
→ built-in (and growing) machine learning support for predictive control, sensor fusion
→ facilitates asynchronous links between sensors and loops
→ full speed telemetry to disk for post-processing
Foreseen applications: Extreme-AO, MCAO, Tomographic AO ...

Community effort, fully open-source
→ no proprietary / closed source roadblocks
→ enables easy/quick implementation of new algorithms
→ adopts standard data stream format

Easy to adopt and use: short path from ideas to real-time implementation
→ abstract away / hide HPC complexity
→ manage challenging real-time timing constraints for the user
→ provide high-level GPU/CPU configuration
Current Status

Provides low-latency to run control loops
→ Use mixed CPU & GPU resources, configured to RTC computer system
On SCExAO, control matrix is 14,000 x 2000. Matrix-vector computed in ~100us using 15% of RTC resources @ 3kHz

Portable, open source, modular, COTS hardware
→ No closed-source driver
→ std Linux install (no need for real-time OS)
→ using NVIDIA GPUs, also working on FPGA use
→ All code on github: https://github.com/cacao-org/cacao

Easy for collaborators to improve/add processes
→ Hooks to data streams in Python or C
→ Template code, easy to adapt and implement new algorithms
→ Provide abstraction of link between loops
→ Toolkit includes viewers, data logger, low-latency TCP transfer of streams
cacao RTC: current status & on-sky performance for ExAO

Supports high frame rate conventional ExAO operation

SCExAO: 1200 modes corrected at 3.5 kHz (input: 120x120 Pyramid image, output: 2000 actuator DM)

On-sky visible image (750nm), log scale (VAMPIRES)

Supports advanced operation modes for enhanced science performance … Why?

Mixed CPU/GPU solution provides ample computing power and also supports advanced operation modes:

- Model-free predictive control using machine learning approach → deeper contrast, fainter stars

- On-sky response matrix acquisition while loop is running (<2 sec to acquire 2000-mode response matrix) → improved calibration → higher performance control

- Real-time links between control loops, sensor fusion → speckle control, low-order modes correction
Main Guiding Principle: data owns IPC
Processes sign up to semaphores

Build AO control loop as a finite set of CPU-managed processes. Processes can manage computations on GPGPU(s)

**Interprocess communication (IPC) is contained in data**: cacao streams are held in shared memory and contain semaphores
→ Complexities of IPC are handled by compiler and Kernel (semaphores, shared memory)

Semaphore-based IPC has us-level latency

Write process posts all data stream semaphores

Each of these processes waits on a single semaphore
A simple example

Frame Grabber
camera acquisition

Shared memory
data & metadata

WFS image

Data logger

Send to other
computer

Multiply by
Control Matrix

Reads

semaphore #0
semaphore #1
semaphore #2
semaphore #3

Reads

semaphore #0
semaphore #1
semaphore #2
semaphore #3

Reads

semaphore #0
semaphore #1
semaphore #2
semaphore #3

Incremental DM displacement

Shared memory
data & metadata

semaphore #0
0
semaphore #1
0
semaphore #2
0
semaphore #3
0

semaphore #9
0

Total DM displacement

Shared memory
data & metadata

semaphore #0
0
semaphore #1
0
semaphore #2
0
semaphore #3
0

semaphore #9
0

Frame Grabber
camera acquisition

DM electronics
driver

x gain and add

DM electronics
driver
Process posts semaphores

Frame Grabber
Camera acquisition

WFS image

Shared memory data & metadata

semaphore #0 1
semaphore #1 1
semaphore #2 1
semaphore #3 1
semaphore #9 1

Data logger

Send to other computer

Multiply by Control Matrix

incremental DM displacement

Shared memory data & metadata

semaphore #0 0
semaphore #1 0
semaphore #2 0
semaphore #3 0
semaphore #9 0

x gain and add

Total DM displacement

Shared memory data & metadata

semaphore #0 0
semaphore #1 0
semaphore #2 0
semaphore #3 0
semaphore #9 0

DM electronics driver
Semaphores posted

Frame Grabber

Shared memory data & metadata

WFS image

Data logger

Send to other computer

Multiply by Control Matrix

x gain and add

DM electronics driver

DM electronics

Driver
sem_wait launches next processes

WFS image

Shared memory data & metadata

semaphore #0 0
semaphore #1 0
semaphore #2 1
semaphore #3 0
semaphore #9 1

WFS image

Data logger

Reads

semaphore #0 0
semaphore #1 0
semaphore #2 1
semaphore #3 0

Send to other computer

semaphore #0 0
semaphore #1 0
semaphore #2 1
semaphore #3 0

Multiply by Control Matrix

semaphore #0 0
semaphore #1 0
semaphore #2 1
semaphore #3 0

semaphore #9 0

WFS image

Incremental DM displacement

Shared memory data & metadata

semaphore #0 0
semaphore #1 0
semaphore #2 0
semaphore #3 0
semaphore #9 0

DM electronics driver

DM electronics driver

Send to other computer

semaphore #0 0
semaphore #1 0
semaphore #2 1
semaphore #3 0

Total DM displacement

Shared memory data & metadata

semaphore #0 0
semaphore #1 0
semaphore #2 0
semaphore #3 0
processes run...

Frame Grabber
Camera acquisition writes

WFS image

Shared memory data & metadata

Data logger reads

Send to other computer reads

Multiply by Control Matrix reads

Frame Grabber writes

Data logger

Send to other computer

Multiply by Control Matrix

incremental DM displacement

Shared memory data & metadata

Total DM displacement

Shared memory data & metadata

x gain and add

DM electronics driver
... asynchronously

Frame Grabber camera acquisition

Shared memory data & metadata

WFS image

Data logger

Send to other computer

Multiply by Control Matrix

Data logger

Send to other computer

Data logger

Frame Grabber camera acquisition

Shared memory data & metadata

Shared memory data & metadata

DM electronics driver

Incremental DM displacement

Total DM displacement

x gain and add

DM electronics driver
Frame Grabber

camera acquisition

WFS image

Shared memory data & metadata

semaphore #0 0
semaphore #1 0
semaphore #2 1
semaphore #3 0
semaphore #9 1

Data logger

read

Send to other computer

read

Multiply by Control Matrix

read

incremental DM displacement

Shared memory data & metadata

semaphore #0 1
semaphore #1 1
semaphore #2 1
semaphore #3 1
semaphore #9 1

DM electronics

driver

x gain and add

Total DM displacement

Shared memory data & metadata

semaphore #0 0
semaphore #1 0
semaphore #2 0
semaphore #3 0
semaphore #9 0
Frame Grabber camera acquisition writes to Shared memory data & metadata.

WFS image reads semaphore #0, semaphore #1, semaphore #9, semaphore #2, semaphore #3.

Data logger reads semaphore #0, semaphore #1, semaphore #9.

Send to other computer reads semaphore #0, semaphore #9.

Multiply by Control Matrix reads semaphore #0, semaphore #9.

incremental DM displacement writes to Shared memory data & metadata.

x gain and add multiplies by 0.

Total DM displacement writes to Shared memory data & metadata.

DM electronics driver reads from Shared memory data & metadata.
Frame Grabber
camera acquisition
writes

WFS image

Shared memory
data & metadata

semaphore #0: 0
semaphore #1: 0
semaphore #2: 1
semaphore #3: 0
semaphore #9: 1

Data logger
reads

Send to other computer
reads

Multiply by Control Matrix
reads

incremental DM displacement

Shared memory
data & metadata

semaphore #0: 1
semaphore #1: 1
semaphore #2: 0
semaphore #3: 1
semaphore #9: 1

x gain and add

Total DM displacement

Shared memory
data & metadata

semaphore #0: 1
semaphore #1: 1
semaphore #2: 0
semaphore #3: 1
semaphore #9: 1

DM electronics
driver

DM electronics
driver

DM electronics
driver
processes execution can overlap

Frame Grabber
camera acquisition

WFS image

Shared memory
data & metadata

semaphore #0: 0
semaphore #1: 0
semaphore #2: 1
semaphore #3: 0
semaphore #9: 1

Data logger
reads

Send to other
computer
reads

Multiply by
Control Matrix
reads

Data logger
writes

Frame Grabber
camera acquisition

Shared memory
data & metadata

semaphore #0: 1
semaphore #1: 1
semaphore #2: 0
semaphore #3: 1
semaphore #9: 1

Send to other
computer

semaphore #0: 1
semaphore #1: 1
semaphore #2: 0
semaphore #3: 1
semaphore #9: 1

semaphore #0: 0
semaphore #1: 0
semaphore #2: 1
semaphore #3: 0
semaphore #9: 1

semaphore #0: 1
semaphore #1: 1
semaphore #2: 0
semaphore #3: 1
semaphore #9: 1

semaphore #0: 0
semaphore #1: 0
semaphore #2: 1
semaphore #3: 0
semaphore #9: 1

DM electronics
driver

x gain and add

incremental DM displacement

semaphore #0: 1
semaphore #1: 1
semaphore #2: 0
semaphore #3: 1
semaphore #9: 1

Total DM displacement

semaphore #0: 1
semaphore #1: 1
semaphore #2: 0
semaphore #3: 1
semaphore #9: 1

semaphore #0: 1
semaphore #1: 1
semaphore #2: 0
semaphore #3: 1
semaphore #9: 1

semaphore #0: 0
semaphore #1: 0
semaphore #2: 1
semaphore #3: 0
semaphore #9: 1

semaphore #0: 0
semaphore #1: 0
semaphore #2: 1
semaphore #3: 0
semaphore #9: 1

semaphore #0: 0
semaphore #1: 0
semaphore #2: 1
semaphore #3: 0
semaphore #9: 1

semaphore #0: 0
semaphore #1: 0
semaphore #2: 1
semaphore #3: 0
semaphore #9: 1

semaphore #0: 0
semaphore #1: 0
semaphore #2: 1
semaphore #3: 0
semaphore #9: 1
Shared memory data & metadata

Frame Grabber
Camera acquisition

WFS image

Semaphores:

- Semaphore #0: 0
- Semaphore #1: 0
- Semaphore #2: 1
- Semaphore #3: 0
- Semaphore #9: 1

Shared memory data & metadata

Data logger
Send to other computer
Multiply by Control Matrix

Reads:

- Data logger
- Send to other computer
- Multiply by Control Matrix

Writes:

- Frame Grabber

Send to other computer

Multiplier by Control Matrix

Reads:

- Data logger
- Send to other computer
- Multiply by Control Matrix

Incremental DM displacement

Shared memory data & metadata

Semaphores:

- Semaphore #0: 1
- Semaphore #1: 1
- Semaphore #2: 0
- Semaphore #3: 1
- Semaphore #9: 1

Total DM displacement

Shared memory data & metadata

Semaphores:

- Semaphore #0: 1
- Semaphore #1: 1
- Semaphore #2: 0
- Semaphore #3: 1
- Semaphore #9: 1

X gain and add

DM electronics driver

DM electronics
driver
Unused semaphores go >1

Frame Grabber
- camera acquisition
  - writes
  - WFS image
    - Shared memory data & metadata
      - semaphore #0: 1
      - semaphore #1: 1
      - semaphore #2: 0
      - semaphore #3: 1
      - semaphore #9: 1

Data logger
- reads
- Send to other computer
  - reads
- Multiply by Control Matrix
  - reads

incremental DM displacement

Shared memory data & metadata
- semaphore #0: 1
- semaphore #1: 1
- semaphore #2: 0
- semaphore #3: 1
- semaphore #9: 1

Total DM displacement

Shared memory data & metadata
- semaphore #0: 1
- semaphore #1: 1
- semaphore #2: 0
- semaphore #3: 1
- semaphore #9: 1

x gain and add

DM electronics driver

Multiply by Control Matrix
- reads
Frame Grabber camera acquisition writes to:

- WFS image
- Shared memory data & metadata

Reads from:

- Data logger
- Send to other computer
- Multiply by Control Matrix

Processes:

- Data logger sends to other computer
- Data logger multiplies by Control Matrix

Shared memory data & metadata:

- Semaphores: 
  - Semaphore #0
  - Semaphore #1
  - Semaphore #2
  - Semaphore #3
  - Semaphore #9

DM electronics driver:

- DM electronics
- Driver

DM displacement:

- Incremental DM displacement
- Total DM displacement

Operations:

- Incremental DM displacement
  - X gain and add
  - Shared memory data & metadata

- Total DM displacement
  - X gain and add
  - Shared memory data & metadata
  - DM electronics driver
A more powerful example (SCExAO control loop)

- **Telemetry**
  - `aol#_modeval_ol_logbuff0`
  - `aol#_modeval_ol_logbuff1`

- **Predictive Filter**
  - (shown here for block #0)
  - Predictive filter compute
    - `aol#PFb0apply` in `aol0RT1`
  - Predictive filter engine
    - `aol#PFb0apply` in `aol0RT1`
  - Predicted mode coefficients
    - `aol#_modevalPFb0`

- **Open loop mode coefficients**
  - Buffer for predictive block
    - `aol#_modevalol_PFb0`

- **Extract WFS modes**
  - [aol#mexwfs] in `aol#RT1`
    - `auxscripts/modesextractwfs`
    - GPU or CPU

- **Direct DM Write → actuators**
  - `if CMMODE=0`
    - `DM primary Write`
      - `DM "actuators"`

- **DM map (test)**
  - `script aolOLcoeffs2dmmap`

- **DM map (test)**
  - `script aolPFcoeffs2dmmap`

- **DM filtered Write**
  - [aol#dmfw] in `aol0RT`
    - `auxscripts/aolmcoeffs2dmmap`
    - GPU or CPU

- **Current modal DM correction, filtered**
  - `aol#_modeval_dm_now_filt`

- **Modal clipping**
  - `aol#_DMmode_LIMIT`

- **Modal DM correction, circular buffer**
  - `aol#_modeval_dm_C`

- **Predicted DM map**
  - `script aolPFcoeffs2dmmap`

- **Current modal DM correction**
  - `aol#_modeval_ol_logbuff0`
  - `aol#_modeval_ol_logbuff1`

- **Predicted mode coefficients**
  - `aol#_modevalPFres`
  - `aol#_modevalPFsync`

- **DM map (test)**
  - `aol#_dmPFout`

- **WFS modes → modes**
  - [aol#mexwfs] in `aol#RT1`
    - `auxscripts/modesextractwfs`
    - GPU or CPU

- **Extract Open Loop WFS modes**
  - [aol#meol] in `aol#RT`
    - `runs in AoloopControl, CPU`

- **Main process**
  - `[aol#run] in `aol#RT`
    - `script auxscripts/aolrun`
    - CPU (+ GPU)

- **Note:** DM map & coefficients show correction applied
  - → open loop = WFS residual – dm
  - → Wfresidual = Open loop WF + dm
  - → dm = Wfresidual – open loop

- **Timings**
  - `latency[frame] = hardlatency_frame + wfsmextrlatency_frame`

- **Semaphores**
  - `sem10+block`
  - `sem2`
  - `sem3`
  - `sem4`
  - `sem5`
  - `sem6`
  - `sem7`
  - `sem8`
  - `sem9`

- **DM mode coefficients**
  - `aol#_DMmode_GAIN`
  - `aol#_DMmode_MULTF`
  - `aol#_DMmode_LIMIT`

- **DM primary Write**
  - `DM primary Write`
  - `DM "actuators"`
  - `DM filtered Write`

- **Telemetry**
  - `dark subtract`
  - `if DMprimaryWrite_ON`

- **Current modal DM correction**
  - `aol#_DMmode_GAIN`
  - `aol#_DMmode_MULTF`
  - `aol#_DMmode_LIMIT`

- **Direct DM Write → actuators**
  - `if CMMODE=0`
    - `Direct DM Write → actuators`

- **Main process**
  - `[aol#run] in `aol#RT`
    - `script auxscripts/aolrun`
    - CPU (+ GPU)
Fastest way to write on DM is to multiply WFS image by control matrix → DM actuators

Direct DM Write → actuators

if CMMODE=0

if DMprimaryWrite_ON

DM primary Write

Telemetry

Predicted mode coefficients

Predictive Filter

(aolPFb0apply in aol0RT)

DM map (test)

dtccll

DM map (test

script aolOLcoeffs2dmmap

Predicted DM map

[aol#dmPFout]

| DM map (test)
| script aolOLcoeffs2dmmap

Predicted DM map

[aol#dmPFout]

Current modal DM correction buffer for predictive block

[aol#_modevalPF_C]

sem3

DM map (test)

script aolPFcoeffs2dmmap

predicted DM map

[aol#_dmPFout]

Mode clipping

aol#_DMmode_LIMIT

Compute

Telemetry

Main process

[aol#run] in aol#RT

script auxscripts/aolrun

CPU (+ GPU)

DM filtered Write

GPU-based DM filtered write

[aol#dmfw] in aol0RT

auxscripts/aolmcoeffs2dmmap

GPU or CPU

Current modal DM correction

[aol#_modeval_dm_now_filt]

Current modal DM correction, filtered

[aol#_modeval_dm_now]

Current modal DM correction

[aol#_modeval_dm]

Current modal DM correction

[aol#_modeval_dm_corr]

Current modal DM correction

[aol#_modeval_dm_core]

Current modal DM correction, filtered

[aol#_modeval_dm_core]

Modal clipping

[aol#_DMmode_GAIN]

Loop ARPF gain

[aol#_DMmode_GAIN]

[aol#_DMmode_MULTF]

Loop mult

[aol#_DMmode_GAIN]

Main process [aol#run] in aol#RT script auxscripts/aolrun CPU (+ GPU)

WFS pixels → modes

Complacency frame (measured by aolMeasureTiming)

WFS image

Direct DM Write → actuators

DM primary Write

DM "actuators"

Telemetry

Predicted mode coefficients

Predictive Filter

(aolPFb0apply in aol0RT)

DM map (test)

script aolOLcoeffs2dmmap

Predicted DM map

[aol#dmPFout]

Current modal DM correction buffer for predictive block

[aol#_modevalPF_C]

sem3

DM map (test)

script aolPFcoeffs2dmmap

predicted DM map

[aol#_dmPFout]

Mode clipping

aol#_DMmode_LIMIT

Compute

Telemetry

Main process

[aol#run] in aol#RT

script auxscripts/aolrun

CPU (+ GPU)

DM filtered Write

GPU-based DM filtered write

[aol#dmfw] in aol0RT

auxscripts/aolmcoeffs2dmmap

GPU or CPU

Current modal DM correction

[aol#_modeval_dm_now_filt]

Current modal DM correction, filtered

[aol#_modeval_dm_now]

Current modal DM correction

[aol#_modeval_dm_corr]

Current modal DM correction

[aol#_modeval_dm_core]

Current modal DM correction, filtered

[aol#_modeval_dm_core]

Modal clipping

[aol#_DMmode_GAIN]

Loop ARPF gain

[aol#_DMmode_GAIN]

[aol#_DMmode_MULTF]

Loop mult

[aol#_DMmode_GAIN]

Main process [aol#run] in aol#RT script auxscripts/aolrun CPU (+ GPU)

WFS pixels → modes

Complacency frame (measured by aolMeasureTiming)

WFS image

Direct DM Write → actuators

DM primary Write

DM "actuators"

Telemetry

Predicted mode coefficients

Predictive Filter

(aolPFb0apply in aol0RT)

DM map (test)

script aolOLcoeffs2dmmap

Predicted DM map

[aol#dmPFout]

Current modal DM correction buffer for predictive block

[aol#_modevalPF_C]

sem3

DM map (test)

script aolPFcoeffs2dmmap

predicted DM map

[aol#_dmPFout]

Mode clipping

aol#_DMmode_LIMIT

Compute

Telemetry

Main process

[aol#run] in aol#RT

script auxscripts/aolrun

CPU (+ GPU)

DM filtered Write

GPU-based DM filtered write

[aol#dmfw] in aol0RT

auxscripts/aolmcoeffs2dmmap

GPU or CPU

Current modal DM correction

[aol#_modeval_dm_now_filt]

Current modal DM correction, filtered

[aol#_modeval_dm_now]

Current modal DM correction

[aol#_modeval_dm_corr]

Current modal DM correction

[aol#_modeval_dm_core]

Current modal DM correction, filtered

[aol#_modeval_dm_core]

Modal clipping

[aol#_DMmode_GAIN]

Loop ARPF gain

[aol#_DMmode_GAIN]

[aol#_DMmode_MULTF]

Loop mult

[aol#_DMmode_GAIN]

Main process [aol#run] in aol#RT script auxscripts/aolrun CPU (+ GPU)

WFS pixels → modes

Complacency frame (measured by aolMeasureTiming)

WFS image

Direct DM Write → actuators

DM primary Write

DM "actuators"

Telemetry

Predicted mode coefficients

Predictive Filter

(aolPFb0apply in aol0RT)

DM map (test)

script aolOLcoeffs2dmmap

Predicted DM map

[aol#dmPFout]

Current modal DM correction buffer for predictive block

[aol#_modevalPF_C]

sem3

DM map (test)

script aolPFcoeffs2dmmap

predicted DM map

[aol#_dmPFout]

Mode clipping

aol#_DMmode_LIMIT

Compute

Telemetry

Main process

[aol#run] in aol#RT

script auxscripts/aolrun

CPU (+ GPU)

DM filtered Write

GPU-based DM filtered write

[aol#dmfw] in aol0RT

auxscripts/aolmcoeffs2dmmap

GPU or CPU

Current modal DM correction

[aol#_modeval_dm_now_filt]

Current modal DM correction, filtered

[aol#_modeval_dm_now]

Current modal DM correction

[aol#_modeval_dm_corr]

Current modal DM correction

[aol#_modeval_dm_core]

Current modal DM correction, filtered

[aol#_modeval_dm_core]

Modal clipping

[aol#_DMmode_GAIN]

Loop ARPF gain

[aol#_DMmode_GAIN]

[aol#_DMmode_MULTF]

Loop mult

[aol#_DMmode_GAIN]
Write modal telemetry buffers

Modal clipping

Predictive filter engine

Predictive filter compute

Modal telemetry buffers

Direct DM Write → actuators

if CMMODE=0

Current modal DM correction

Current modal DM correction

if DMprimaryWrite_ON

Extract WFS modes

GPU-based DM filtered write

DM filtered Write

Direct DM Write → actuators

Current modal DM correction, filtered

Note: DM map & coefficients show correction applied
→ open loop = WFS residual – dm
→ Wfresidual = Open loop WF + dm
→ dm = Wfresidual – open loop

Computation timing

DM primary Write
Apply Predictive Filter

WFS image

Predicted DM map

Extract Open Loop WFS modes
[aol#meol] in aolRT
runs in AoloopControl, CPU

Open loop mode coefficients

Predicted DM map

Extract WFS modes
[aol#mexwfs] in aolRT
GPU or CPU

Compute Telemetry

Main process
[aol#run] in aol#RT
script auxscripts/aolrun
CPU (+ GPU)

if CMMODE=0

Direct DM Write → actuators

if DMprimaryWrite_ON

Direct DM Write

DM “actuators”

DM primary Write

WFS pixels

WFS pixels → modes

DM map (test)

script aolOLcoeffs2dmmap

Predicted DM map

DM map (test)

script aolPFcoeffs2dmmap

Predicted DM map

DM filtered Write

Note: DM map & coefficients show correction applied

open loop = WFS residual – dm
Wfresidual = Open loop WF + dm
dm = Wfresidual – open loop

Computation [aol#poll] in aol#RT

auxscripts/aolmcoeffs2dmmap
GPU or CPU
Linking loops / handling multiple WFSs and DMs
SCExAO Light path

Facility AO

- Modulated Visible PyWFS 0.4-1.0 μm
- Weakly/un-modulated NearIR PyWFS 0.8-2.0 μm
- 50x50 DM 3.5 kHz
- Coronagraph
- Photonic nuller
- IRD 1-2 μm HR spectrograph
- Sci path viewing cam
- 111 DM segmented
- FIRST Polarimetry Interferometry
- VAMPIRES (2 cameras) Polarimetry Dual band Aperture masking
- RHEA visible IFU
- coronagraphic LOWFS
- CHARIS nearIR IFU
- MKIDS focal plane WFS
- SAPHIRA Imager

Active WF correction
Dedicated WFS
Visitor port
Mixed science/WFS
LOWFS LOOP
loopnb = 1
DMindex = 01

MASTER LOOP
PyWFS LOOP
loopnb = 0
DMindex = 00

FPWFS LOOP
loopnb = 2

LOWFS modes (modal actuators)
- Offset (flat)
- Response matrix
- AO control
- Zero pt offset
- Mode DM maps
- Mode PyWFS resp

Main DM channels (physical actuators)
- Offset (flat)
- TT LQG
- PyWFS RM
- PyWFS control
- Astrom grid
- Speckle probes
- Speckle control
- Zernike offsets
- Astrom offset
- LOWFS offset
- FPWFS probe offset
- PyWFS offset
- LOWFS resp matrix
- PyWFS reference
- PyWFS mod maps
- FPWFS probe images
- FP solution
- Mode FPWFS resp

LOWFS + HOWFS + FPWFS wavefront control architecture
Linking multiple control loops (zero point offsetting)

A control loop can offset the convergence point of another loop @> kHz (GPU or CPU)
Example: speckle control, LOWFS need to offset pyramid control loop
THIS IS DONE TRANSPARENTLY FOR USER → don't pay attention to the diagram below!

DM channels, loop 0

Loop 0 DM modes

Loop 0 WFS modes

DM channels, loop 0

LOWFS offsetting notes

From PyWFS loop:
- Do not turn on zonal offsetting ZP1
- Turn on zero pt offset process

script "aolWFSresoffloadloop"
slow offload of WFS average
Good timing **knowledge and stability** is essential for:

→ Pseudo-open loop reconstruction

→ Fast response matrix acquisition

→ Predictive control
End of real-time computation processes
End-to-end timing jitter measured by monitoring completion time of last real-time stream: modal pseudo-open loop coefficients.

Jitter includes following components:
- Hardware synchronization (PyWFS tip-tilt mirror)
- Camera readout
- Data transfer over TCP link
- All real-time computations, CPU and GPU
- Time measurement errors
End-to-end Timing Jitter Histogram, measured @ 2kHz

- 80% within +/- 3.92 us
- 9.01% Delay > 5 us
- 0.645% Delay > 10 us
- 0.15% Delay > 20 us
- 0.005% Delay > 40 us

10% of loop iteration @ 2 kHz
Hardware Latency measured on SCExAO

Time between DM command issued and corresponding WFS signal observed (Camera readout + TCP transfer + processing + DM electronics)

Measurement noise +/- 25 us lines

Sum squared difference between two WFS frames
Definition:
Time offset between **DM command issued**, and **mid-point between 2 consecutive WFS frames with largest difference**

SCExAO measured hardware latencies:

- **1 kHz**: 1253 / 1260 / 1269 → 1261 us
- **1.5 kHz**: 1083 / 1065 / 1081 → 1076 us
- **2 kHz**: 987 / 982 / 985 → 985 us
- **2.5 kHz**: 922 / 921 / 926 → 923 us
- **3 kHz**: 881 / 876 / 884 → 880 us

Difference 2kHz - 3kHz = 105 us
Expected difference = (1/2000 - 1/3000)/2 = 83 us
→ 22us discrepancy

Difference 1kHz - 3kHz = 361 us
Expected difference = (1/1000 - 1/3000)/2 = 333 us
→ 28us discrepancy

HardwareLatency = DM soft + DM elec + DM phys + CAM readout/transfer + CAM processing + ½ exposure time

HardwareLatency = N x cam_exposure + dt
Fast RM acquisition (4000 Hadamard pokes in 2s @ 2 kHz) + Removing temporal DM response from response matrix by using two poke sequences

Temporal bleeding from previous poke pattern (should be removed from RM)

Camera readout RF coupling between pixels ~1% electronic ghosts at 2kHz frame rate Needs to be kept in RM

RM assembled from single poke sequence:
+- + - +

RM assembled from average of two poke sequences:
+- + - +
+- + + +

RMs reconstructed from Hadamard pokes, 2kHz modulation (DM moves during EMCCD frame transfer)
Multi-channel DM virtualization & timing knowledge/stability → on-sky response matrix acquisition, while ExAO loop running

Left: WFS reference
Right: Response to single actuator poke (one of 2000)

RM measurement @ 2kHz takes 4000 pokes = 2 sec
Multiple RMs averaged to increase SNR
Self-learning AO control

Conventional AO:
Resp Matrix is **measured**
CM computed as pseudo-inverse of RM

Self-learning AO control:
Optimally use recent (predictive control) and auxiliary (sensor fusion) measurements→ control matrix is very big, and usually impossible to measure

CM is **derived** from WFS(s) telemetry using machine learning approaches
Open loop reconstruction
Comparison between gain values

G=0.000 → over-estimates OL values
All G>0.0 reconstructions match at %-level

G=0.000 test relies entirely on WF residuals for OL estimation
G>0.000 tests rely mostly on DM values for OL estimation

Test shown here uses full speed RM acquisition which underestimates RM by ~15% due to DM time-of-motion → reconstructed WFs from WFS are over-estimated by ~15%
On-sky predictive control matrix
(modal representation, 100 modes shown)

Conventional AO would have control matrix
100 x 100 elements
Identity matrix

Optimal control adds elements outside of diagonal

Predictive control adds these blocks to control matrix

Last WFS measurement
WFS measurement Step -1
WFS measurement Step -2
WFS measurement Step -3

2kHz, target #1
1kHz, faint source
2kHz, target #2
Prediction control matrix

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>B</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>C</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
</tr>
<tr>
<td>D</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
</tr>
<tr>
<td>E</td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
</tr>
</tbody>
</table>
On-sky results (2 kHz, 50 sec update)

Average of 54 consecutives 0.5s images (26 sec exposure), 3 mn apart
Same star, same exposure time, same intensity scale
Standard deviation improved by 2.5x

Standard deviation of 54 consecutives 0.5s images (26 sec exposure), 3 mn apart
Same star, same exposure time, same intensity scale
Focal Plane WFS/C
OCA/KERNEL – developed software

- Address NCPA
- Asymmetric mask (pupil)
- On-sky closed-loop control
- Focal plane based WFS
  Low-order (Zernike and LWE) modes.
- mode compatible with coronagraphy in development
Speckle Control

Speckle nulling, in the lab and on-sky (no XAO).

Experience limited by detector readout noise and speed.

KERNEL project: C-RED-ONE camera.

From:
- 114 e- RON
- 170 Hz frame rate

To:
- 0.8 e- RON
- 3500 Hz frame rate

Expect some updates
Conventional Lyot Coronagraph, Broadband light: 0.9-1.7 um (62% wide band)

- Average raw contrast [15-20 I/D] = 2.3e-6
- Average raw contrast stability [15-20 I/D] = 5.5e-8
- Average raw contrast stability [11-34 I/D] = 1.1e-8 (averaged value within white box)

![Graphs showing raw contrast and contrast stability](image-url)

- Raw contrast and contrast stability (open loop)
- Contrast stability vs. timescale (Tip-tilt removed)
Near-IR spatial LDFC validation @ SCExAO
Frame rate = 170 Hz, Lyot coronagraph in near-IR (1.55um, 50nm wide band)

C=1.25e-5 speckle at 13 \lambda/D separation

(a) No aberration LDFC ON
(b) LDFC OFF speckle injected
(c) LDFC ON

Dynamic test, 591 modes controlled
Aberration char. timescale = 160 ms
LDFC loop frequency = 170 Hz
1.6 frames latency

Measured LDFC BF signal response to single actuator poke
Coherent Speckle Differential Imaging
Thank you!

https://github.com/cacao-org/cacao