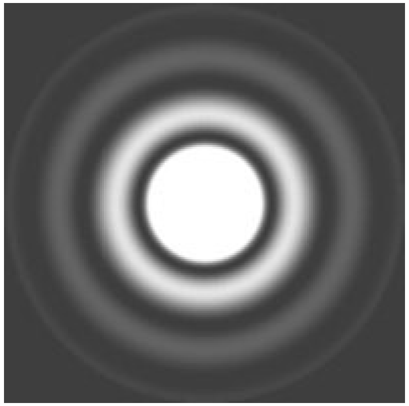
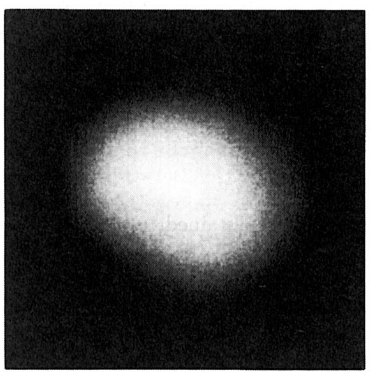
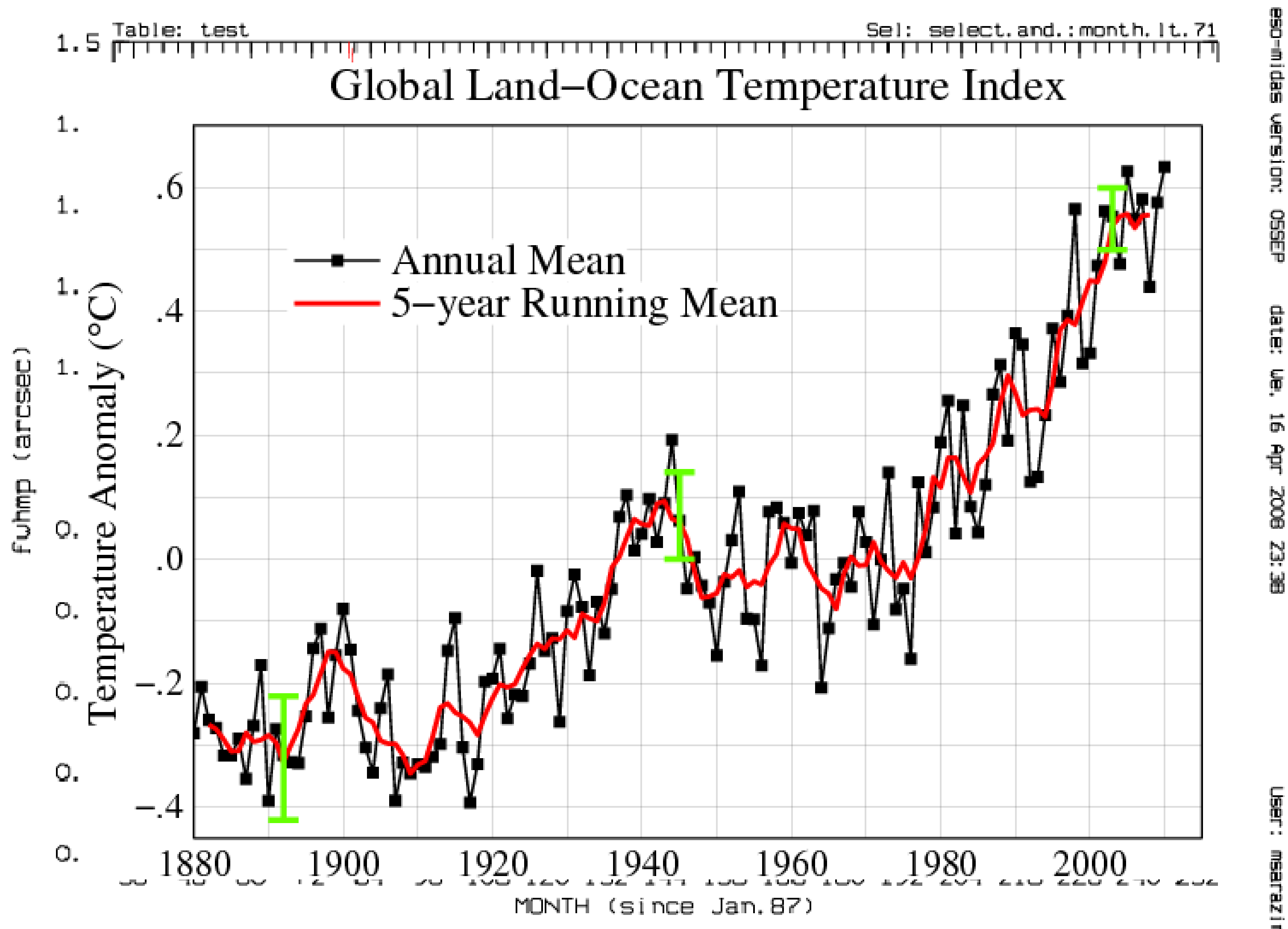
Adaptive optics simulations etc

Alastair Basden
Durham University

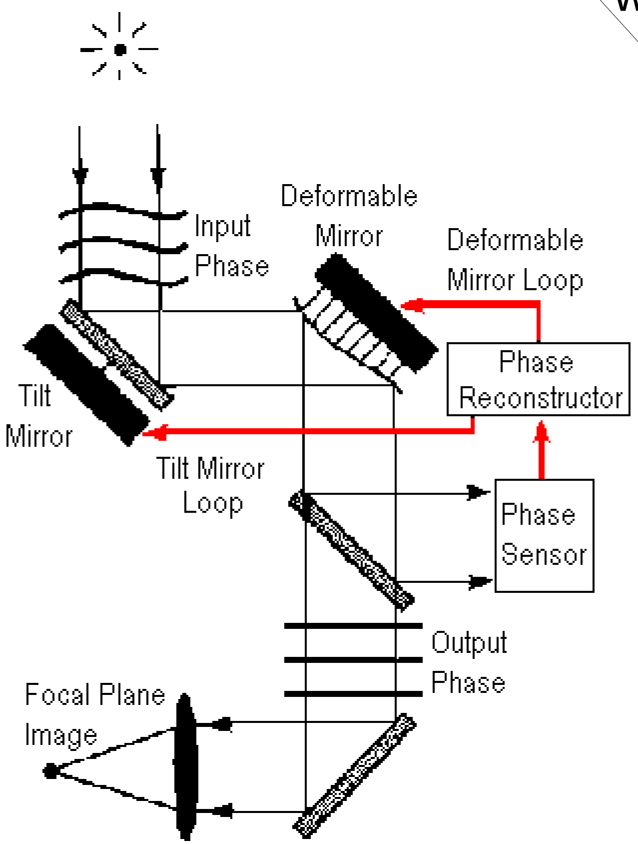
AO: Introduction

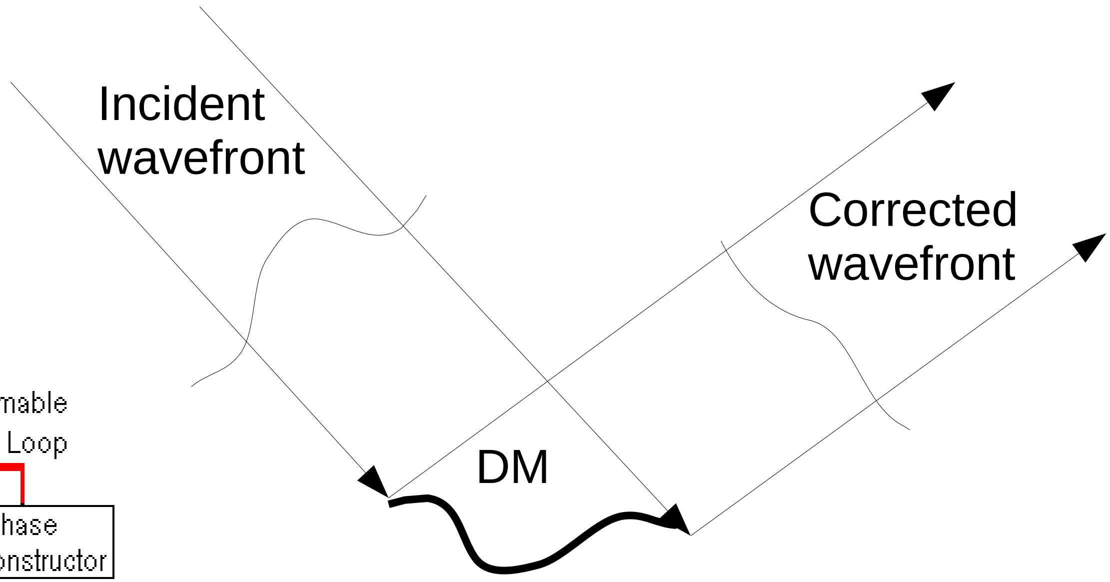


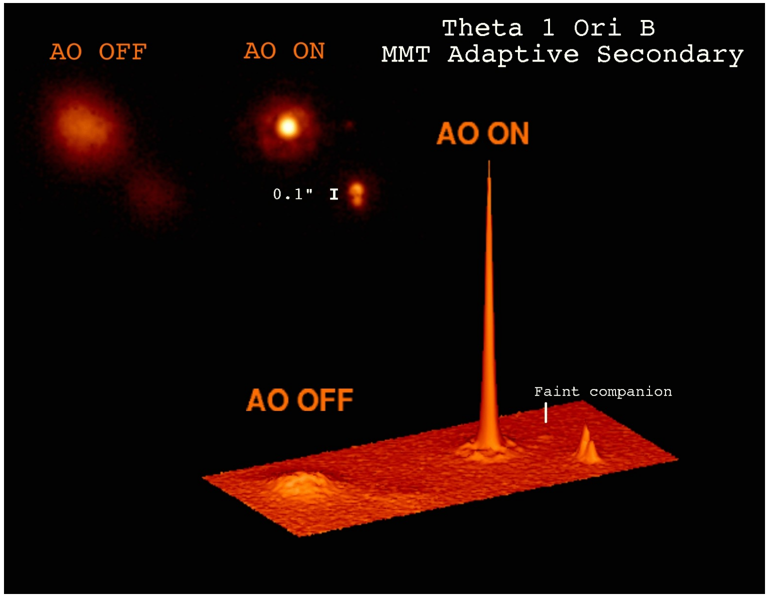




AO: Correction







AO: Performance

- How well will a given AO system perform?
- We need to simulate it to find out...

AO: Simulation

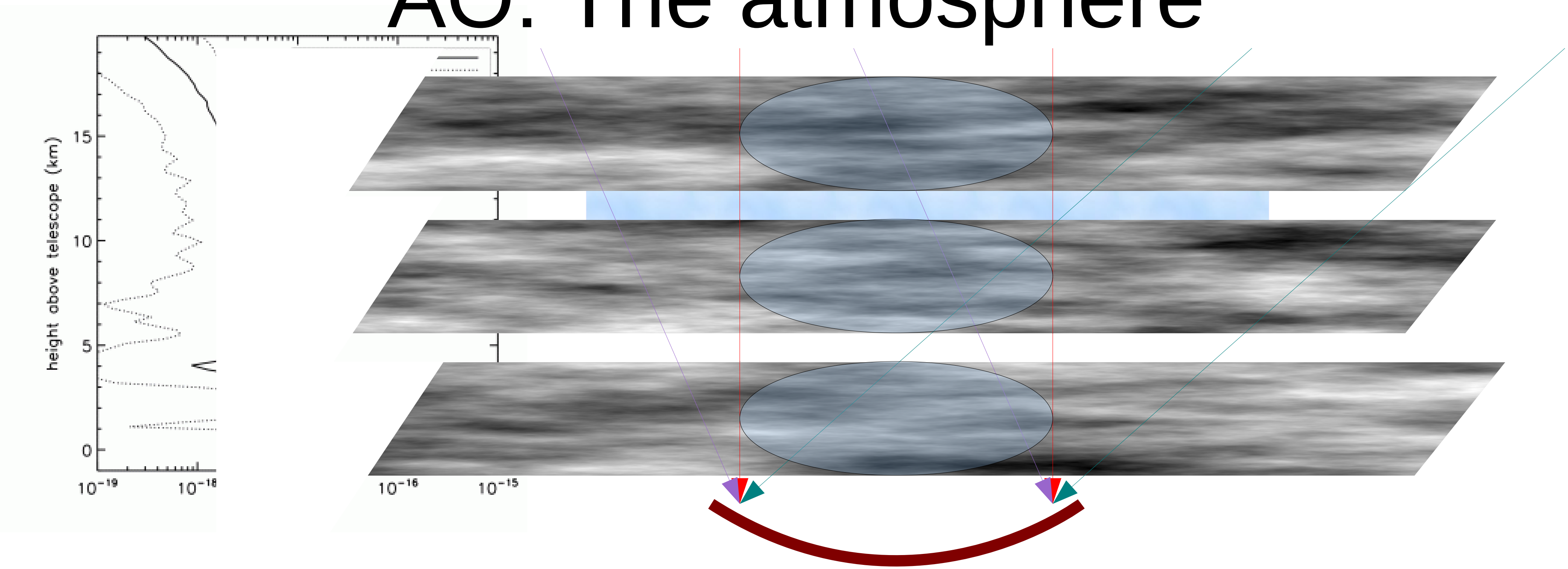
- Lots of components:
  - At
  - Te
  - W
  - D
  - W
  - M
  - Science cameras

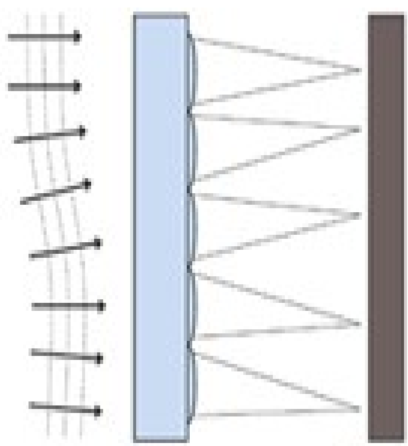
2 main simulation categories:

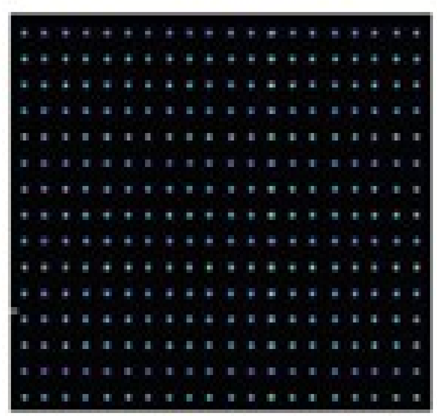
Analytical (usually Fourier based)

Monte-Carlo (physical/geometrical optics, time series etc)

AO: The atmosphere







AO: Monte-Carlo

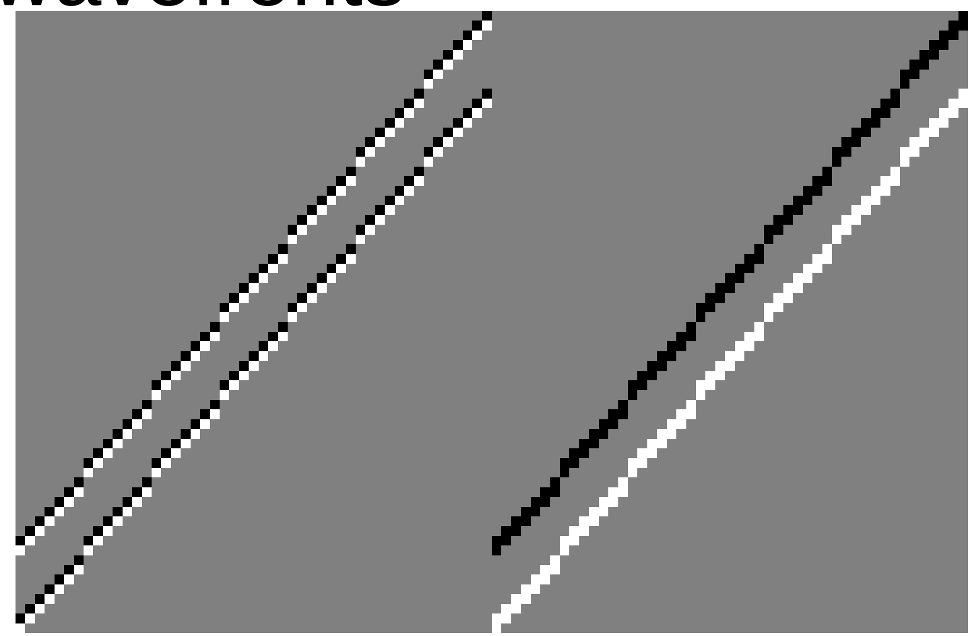
- Break into timesteps:
  - Translate phase screens (wind velocity, frozen turbulence etc)
    - Gives time varying turbulence
  - Introduce DM effect
    - Shaped on previous timestep
  - Measure wavefront with wavefront sensor
    - CCD noise, shot noise etc
  - Make science image, and integrate with previous images
  - Compute new DM shape (wavefront reconstruction)

AO: 3 stages

- Calibrations
- Calculations
- Atmosphere/telescope simulation
  - Full end-to-end simulation
  - Parallelised with MPI and threading

AO: Calibrations

- Need to know how the DM affects the wavefronts



- Matrix equation: Ax=b

$$A\begin{pmatrix}1\\0\\0\\0\\\cdots\\0\end{pmatrix}=\begin{pmatrix}s_1\\s_2\\s_3\\\cdots\\s_n\end{pmatrix}$$

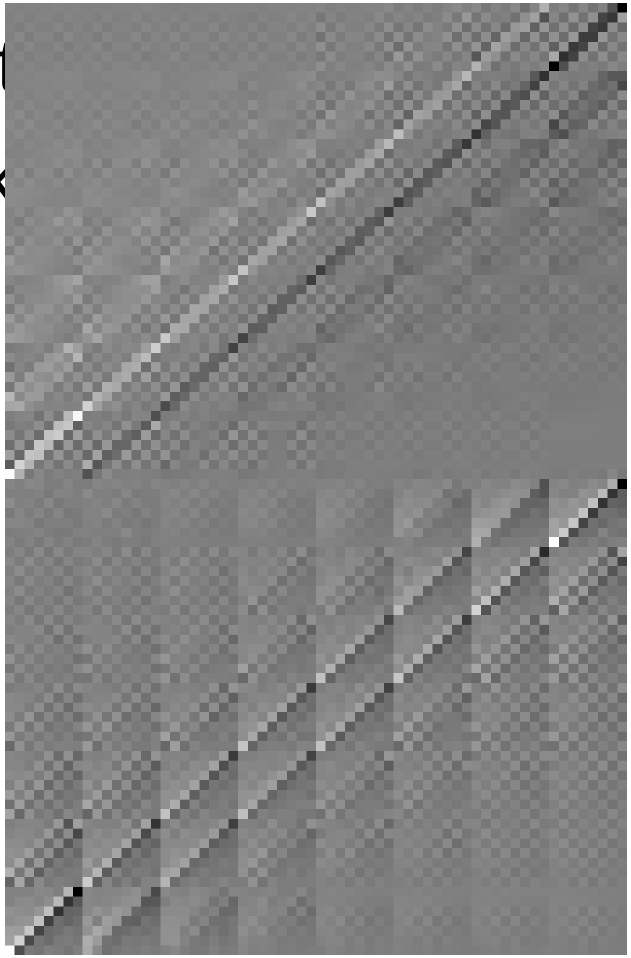
$$A:\rightarrow\begin{pmatrix}1 & \ldots & m\\ \ldots & \ldots & \ldots\\ n & \ldots & \ldots\end{pmatrix}$$

m: Number of actuators

n: Number of slopes

AO: Calculations

- Solve Ax=b
  - b is the slope measurement
  - A is the interaction matrix (k
  - x are the values to be put o



- $x = A^{-1}b$
- So, compute A^{-1}
  - Pseudo inverse: $(A^TA)^{-1}A^T$

AO: Simulation

- Wavefront sensor measures slopes, b
  - Reformat into a vector
- Compute the corresponding correction:
  - $x=A^{-1}b$
- Apply x to the mirror:
  - reshape x to 2D, and apply cubic spline interpolation

Monte-Carlo: challenges

- Larger simulations take longer to run
  - Higher order or larger telescope
- Scales as O(D^4) for reconstruction
- The inversion step (one off) scales as O(D^6)
  - Nasty... but only once per simulation
  - WHT: 4.2m → 0.05s
  - ELT: 42m → 14 hours!!! (39m → 9 hours)
    - Actually more like 4-5 hours

AO: Alternative solvers

- Inversion can take too long
- Use an iterative solver (a direct solver):
  - Ax=b
  - No inversion necessary (A^{-1} not needed)
  - Conjugate gradient algorithm favoured
  - But must be done every simulation time-step
    - Takes longer to run, replaces the matrix-vector multiplication
    - $O(D^4)$ but with much larger pre-factor

A lesson

- Ca
- Co
- Si

Use the most appropriate algorithms...

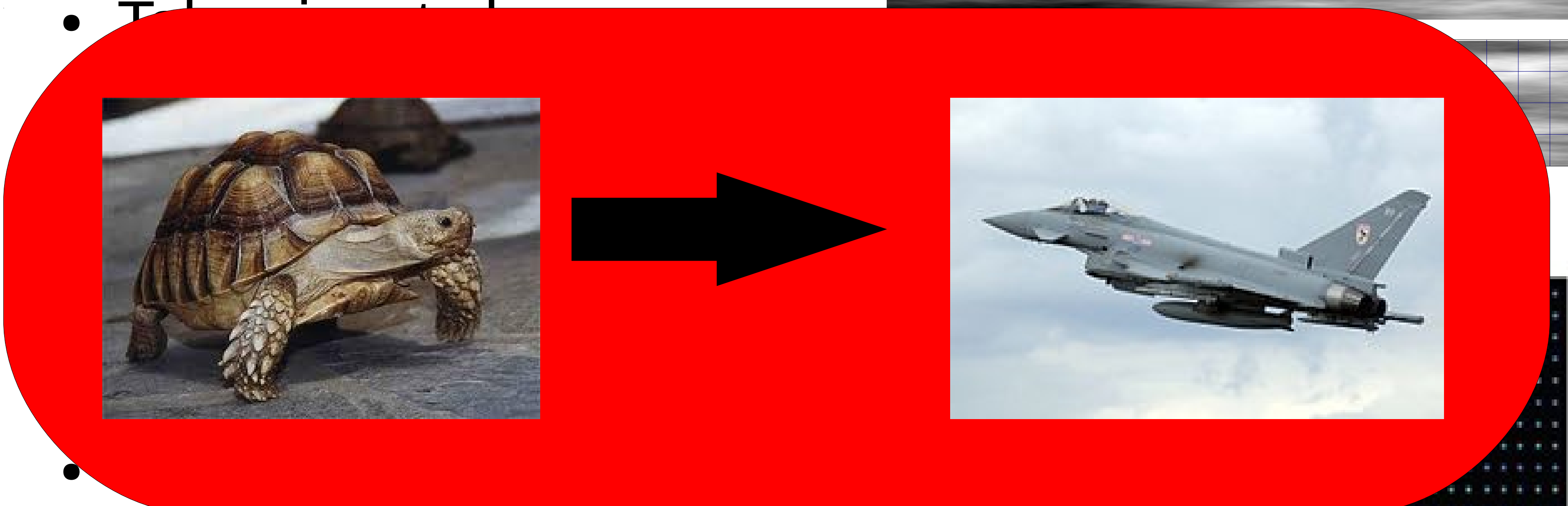
Change reconstruction algorithm:
Compute: 0 hours
Simulate: 8 hours
2 hour gain
Also quick view into expected performance:
30 minutes to first result (was 4.5 hours)
8.5 hours to end (was 10.5)

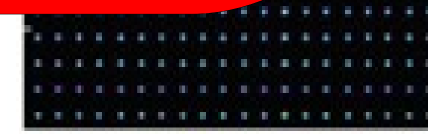
AO: Hardware acceleration

- Cray XD1 supercomputer
- 12 Opterons, 6 FPGAs
- Circa 2004

AO: Wavefront sensor acceleration

- Wavefront sensor module:





  - In an FPGA: 600x speedup!!!
    - 9 months FTE

Amdahls Law

- The speedup … rovement to a comput… on P of that compu… nt has a speedup o…

s=600

If P=0.9:
Speedup 9.9x

If P=0.5:
Speedup 2x

If P=0.1:
Speedup 1.1x

CPU improvements

- Wait a year: CPUs will improve
- Will this render hardware acceleration worthless?
  - Depends on:
    - simulation type
    - achievable speedup
    - effort required
    - reusability

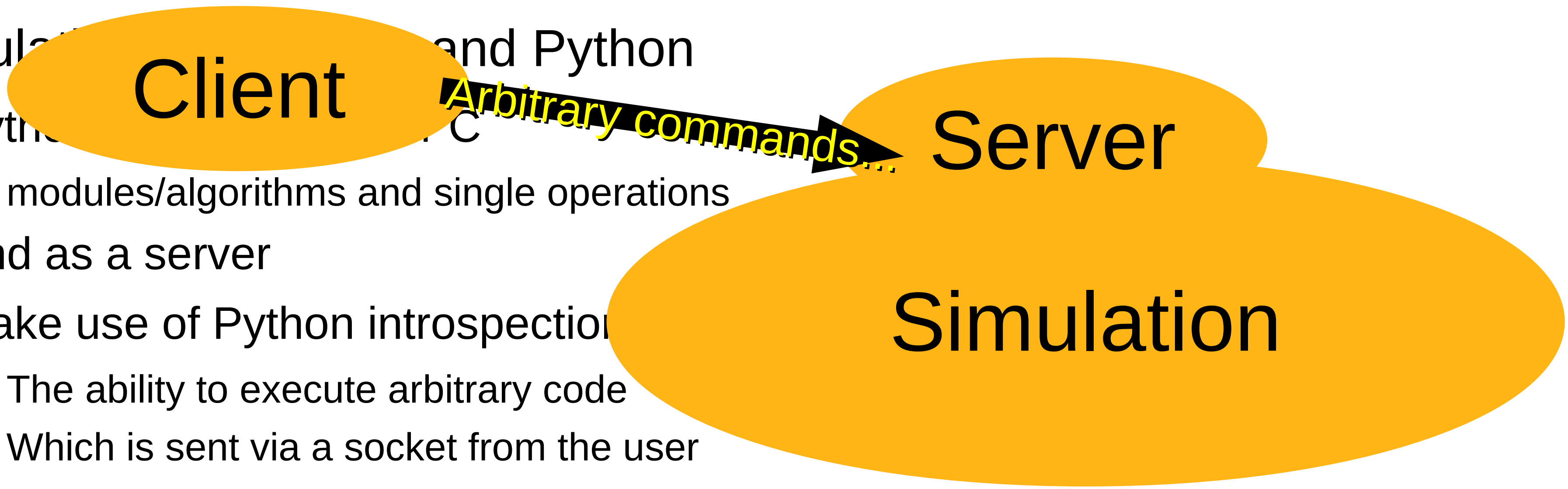
•For GPUs, effort usually small/medium
- Code reusable – should work with new GPUs
- Though performance gain can actually be a loss if done badly

Simulation usability

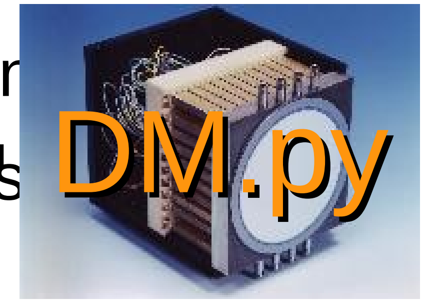
- Tweaking an AO simulation is important
  - While it is running...
  - Allows a quick investigation of parameters
    - to help decide on a parameter space to explore
  - And helps debug (why isn't performance what we hoped for!)
- Diagnostics also important – plots, printouts etc
- How can we do this?
  - Turn the simulation into a server
    - Clients connect, and can then send commands/request data
  - Use shared memory for parameters
    - Clients can modify the shared memory – but more dangerous

Our approach to usability

- Simul… and Python
  - Pyth… C
    - modules/algorithms and single operations
  - And as a server
  - Make use of Python introspection
    - The ability to execute arbitrary code
    - Which is sent via a socket from the user
  - C modules written such that important parameters are accessible and changeable from Python
    - Trade-off between flexibility and implementation time
- Generally a good approach for this type of simulation
  - Unanticipated changes can be made
  - Prototyping in Python before (eventual) speedup in C
  - Debugging made easy – can view all parameters/data in the simulation



Next step: Real-time simulation



- Big differe nce between simulation and on
- So: Use a s much of an on-sky system as

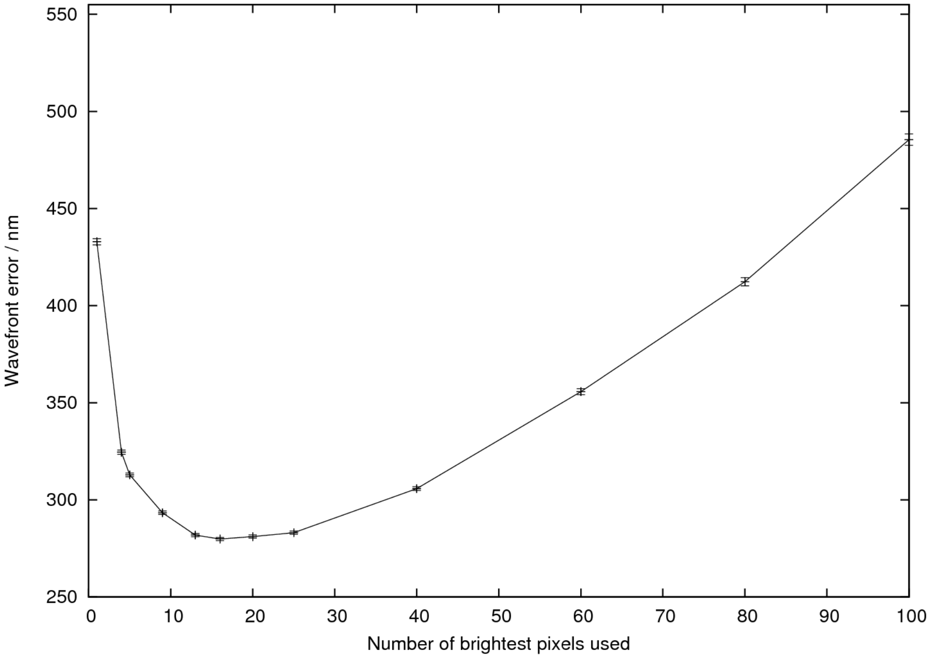
control system (DARC)

plementation of algorithms

figuration etc

- tmosphere
- ons – sinc
  - Real-time for mid-order A
- Allows development/testi

Finally: Some results



- In AO it is standard practise to:
  - Validate with other simulation tools (independent codes)
  - Validate with on-sky results – the ultimate test

Future plans

- Enabling of advanced AO simulations
  - Different operational modes
    - Ground layer AO, laser tomographic AO, extreme AO
  - Speckle suppression
  - Coronograph simulation
- Extensive use of Hamilton cluster
  - Use existing hardware – reduce power consumption
- More end-to-end details
  - Integrated Zemax models
- GPU acceleration
  - Faster, better Performance/Watt
- Database caching