Siesta in parallel and How-To obtain Siesta

Eduardo Anglada

Special thanks:

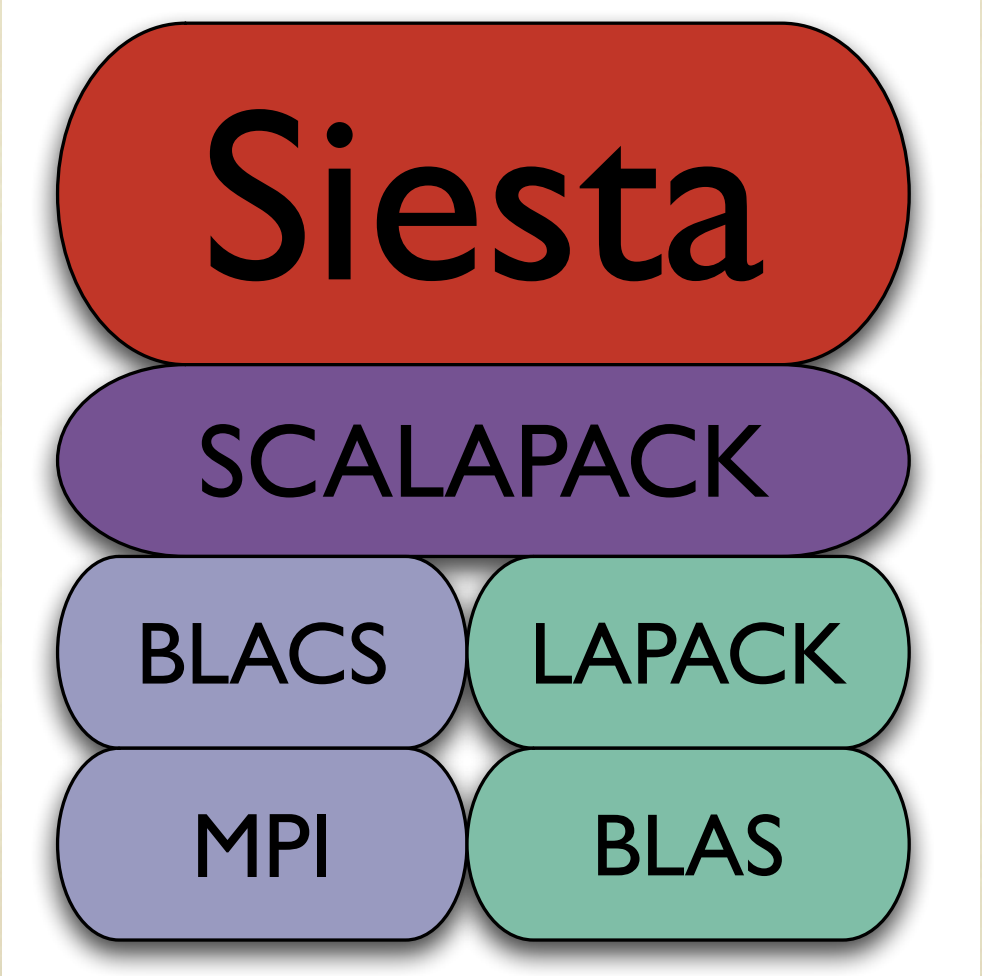
Toby White

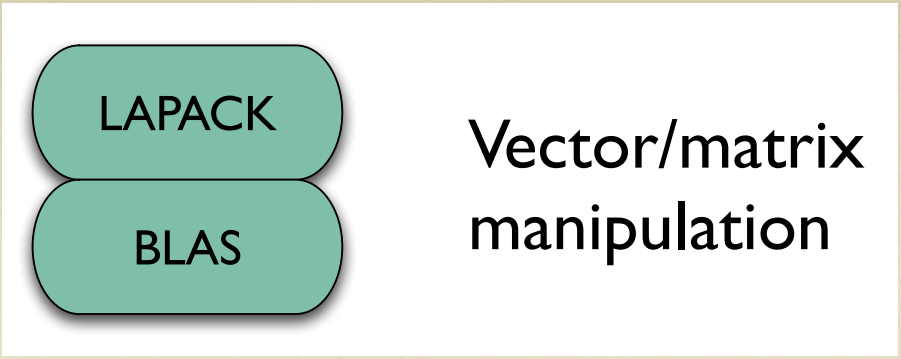
Dept. Earth Sciences, University of Cambridge

OUTLINE

- How to **build** Siesta in parallel
- How to **run** Siesta in parallel
- How Siesta **works** in parallel
- How to use parallelism **efficiently**
- How to obtain Siesta

How-to build siesta in parallel





ATLAS BLAS:
http://atlas.sf.net
Free, open source (needs separate LAPACK)

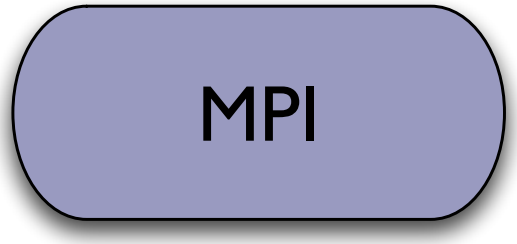
GOTO BLAS:
http://www.tacc.utexas.edu/resources/software/#blas
Free, registration required, source available (needs separate LAPACK)

Intel MKL (Math Kernel Library):
Intel compiler only
Not cheap (but often installed on supercomputers)

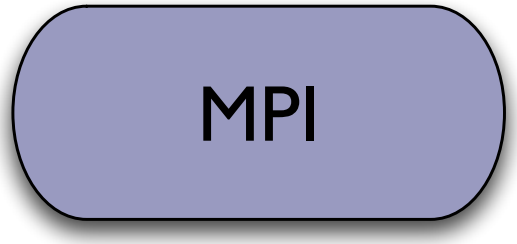
ACML (AMD Core Math Library)
http://developer.amd.com/acml.jsp
Free, registration required. Versions for most compilers.

Sun Performance Library
http://developers.sun.com/sunstudio/perflib_index.html
Free, registration required. Only for Sun compilers (Linux/Solaris)

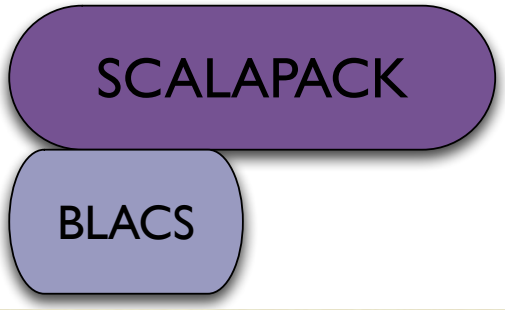
IBM ESSL (Engineering & Science Subroutine Library)
http://www-03.ibm.com/systems/p/software/essl.html
Free, registration required. Only for IBM compilers (Linux/AIX)



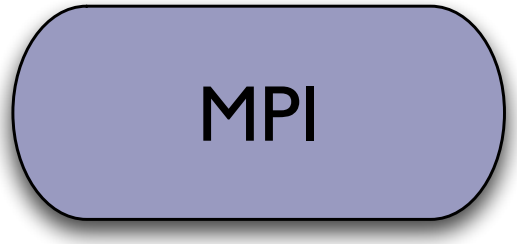
Parallel communication



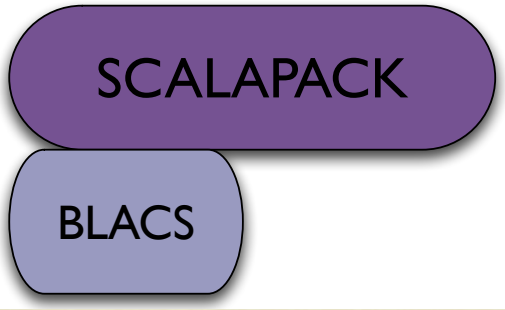
Parallel communication



Parallel linear algebra

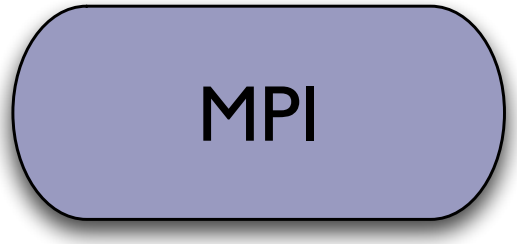


Parallel communication

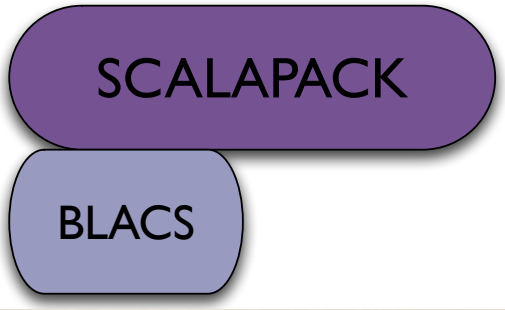


Parallel linear algebra

1. Built "by hand":
   - Difficult! Run all the tests before compiling siesta!
2. INTEL MPI+CMKL
   - Fast! Easy to use (even in non intel CPUs)
3. IBM MPI+PESSL
4. PGI:
   - PGI CDK Cluster Development Kit. For PGI compilers
5. Absoft
6. PathSCALE: for AMD CPUs



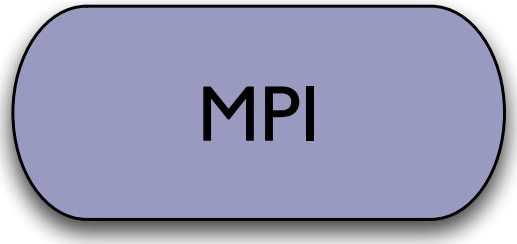
Parallel communication



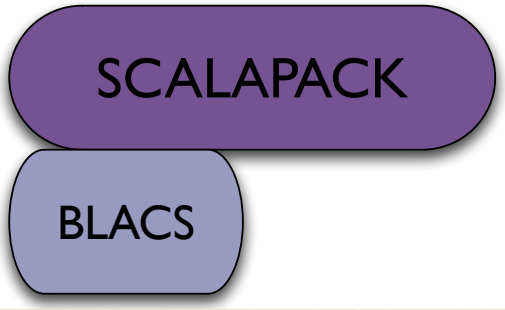
Parallel linear algebra

1. Built "by hand":
   - Difficult! Run all the tests before compiling siesta!
2. INTEL MPI+CMKL
   - Fast! Easy to use (even in non intel CPUs)
3. IBM MPI+PESSL
4. PGI:
   - PGI CDK Cluster Development Kit. For PGI compilers
5. Absoft
6. PathSCALE: for AMD CPUs

See Src/Sys for examples

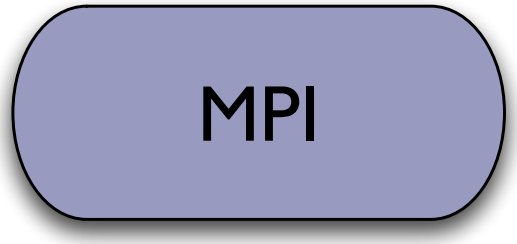


Parallel communication

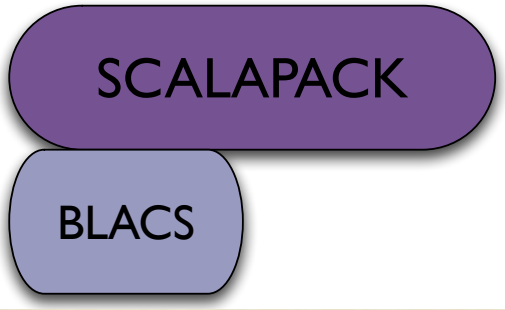


Parallel linear algebra

1. Built "by hand":
   - Difficult! Run all the tests before compiling siesta!
2. INTEL MPI+CMKL
   - Fast! Easy to use (even in non intel CPUs)
3. IBM MPI+PESSL
4. PGI:
   - PGI CDK Cluster Development Kit. For PGI compilers
5. Absoft
6. PathSCALE: for AMD CPUs



Parallel communication



Parallel linear algebra

1. Built "by hand":
   - Difficult! Run all the tests before compiling siesta!
2. INTEL MPI+CMKL
   - Fast! Easy to use (even in non intel CPUs)
3. IBM MPI+PESSL
4. PGI:
   - PGI CDK Cluster Development Kit. For PGI compilers
5. Absoft
6. PathSCALE: for AMD CPUs

If you are completely lost, SIESTA can guess:

```
./configure --enable-mpi
```

but not usually successfully

COMPILING TIPS!

- Do NOT mix & match compilers and libraries
- Some supercomputers have multiple versions of everything installed - several compilers, several sets of numerical libraries.
- Keep everything consistent ! !

How to run SIESTA in parallel

- I can't tell you! But - no change needed in `input.fdf`

On command line, or in jobscript:

```
mpirun -np 4 ./siesta < input.fdf > output.out
```

Sometimes `mprun`, sometimes omitted

Sometimes different flag ...

Sometimes need explicit full path

Sometimes standard input fails on MPI

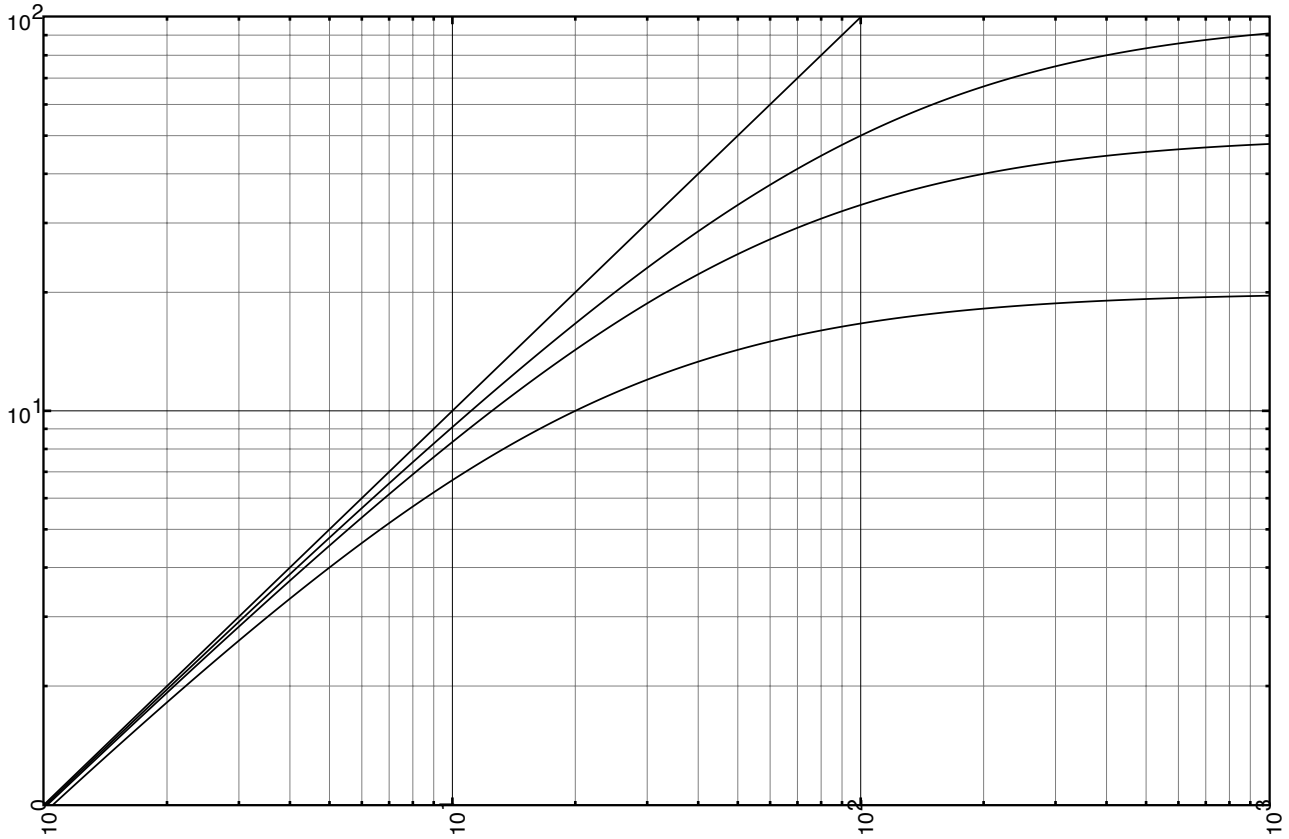
Sometimes standard output fails on MPI

Read your supercomputer documentation!

Principles of parallel Programming

Speedup

$$\frac{T_1}{T_n}$$



nodes

Parallel efficiency

Amdahl's Law:

$$T = T_s + T_p$$

$$T_s \neq 0$$

$$T_p \geq k/n$$

In the best case, for high enough n,
serial time always dominates.

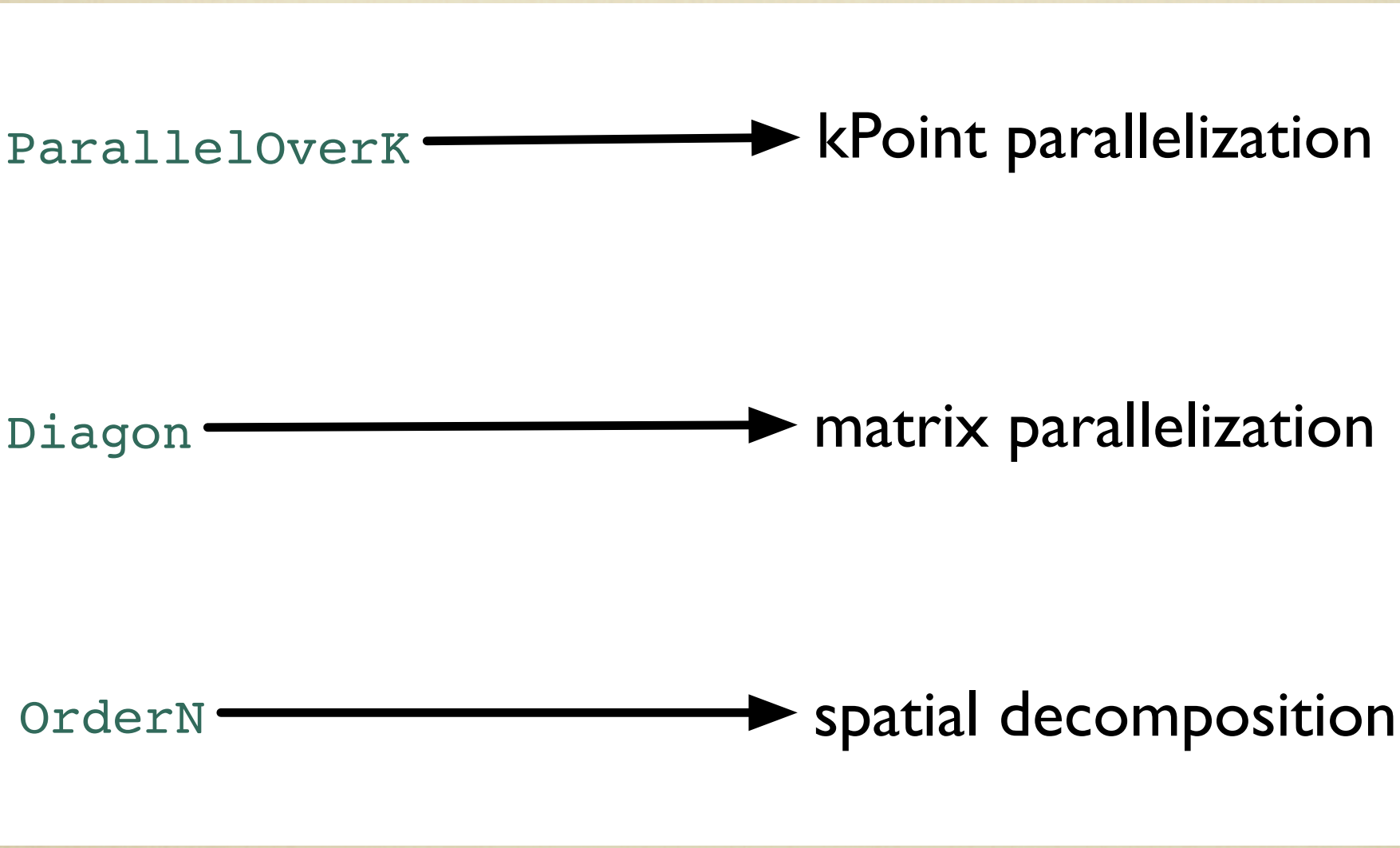
Total CPU time:

Amdahl's Law

Communications Latency

Load balancing

How Siesta works in parallel



KPOINT PARALLELIZATION

Almost perfect parallelism

small ***Ts***

small *latency*

(Number of kPoints) > (Number of Nodes)

FDF LABEL: ParallelOverK

Parallel Diagonalization

`diagon` matrix parallelization

SCALAPACK

- Siesta solves a General eigenvalue problem: Difficult in parallel!

$$\mathbf{Ax} = \lambda\mathbf{Bx},$$

- Key to scaling: orbital distribution

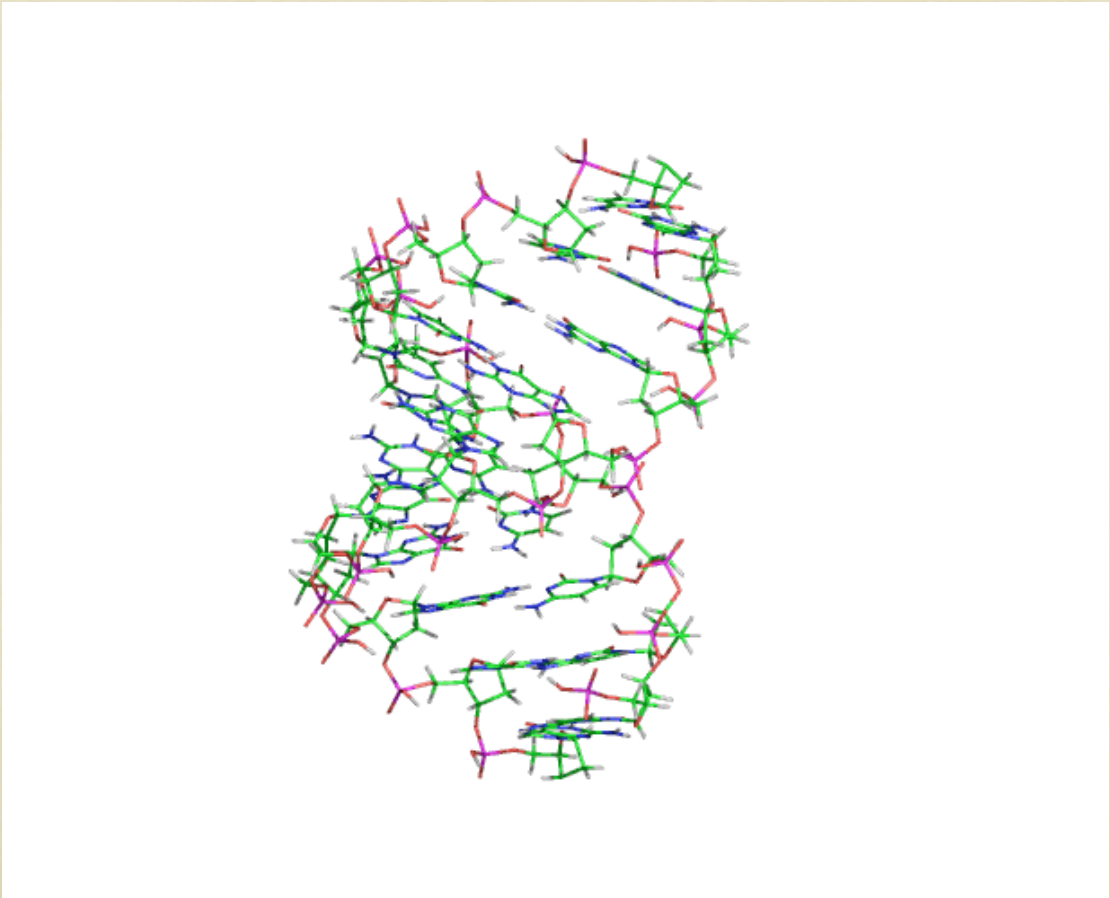
Parallelization: Orbitals distribution

Data blocks

8 CPUs

Parallelization: Orbitals distribution

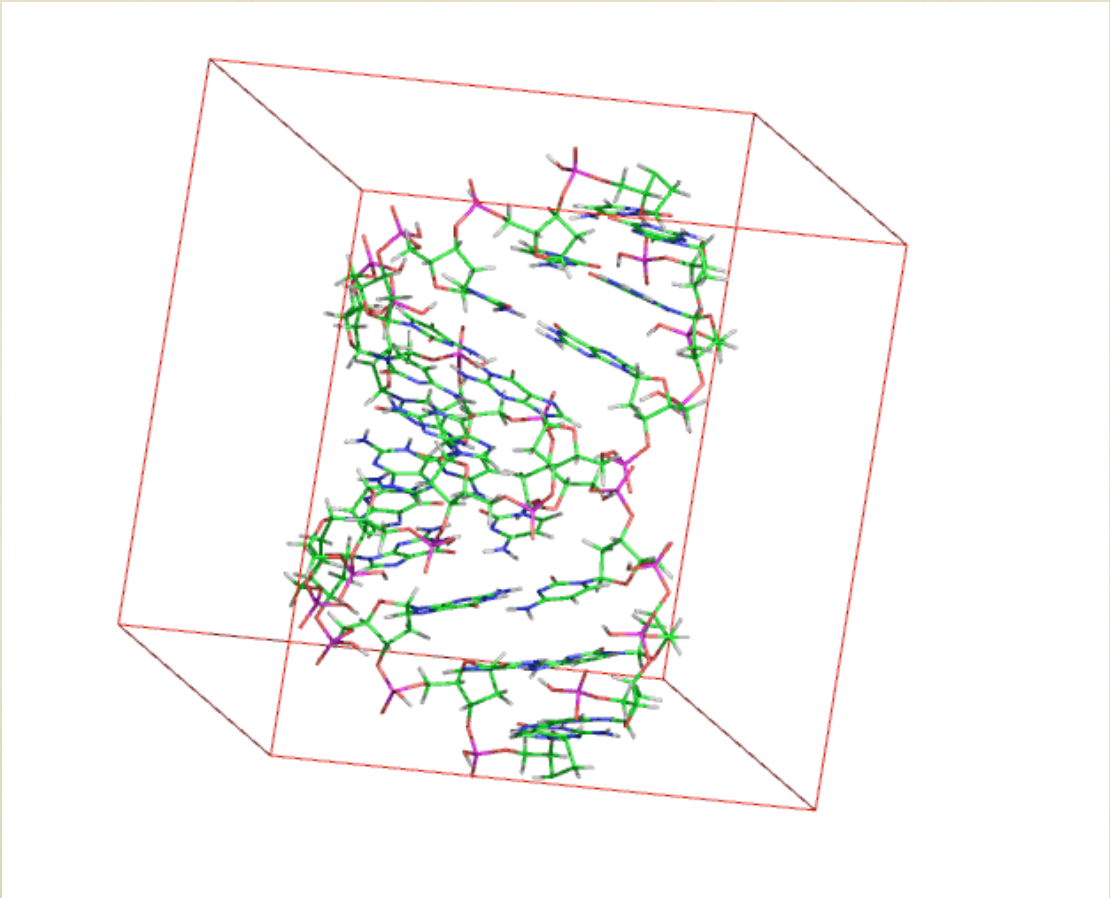
Data blocks



8 CPUs

Parallelization: Orbitals distribution

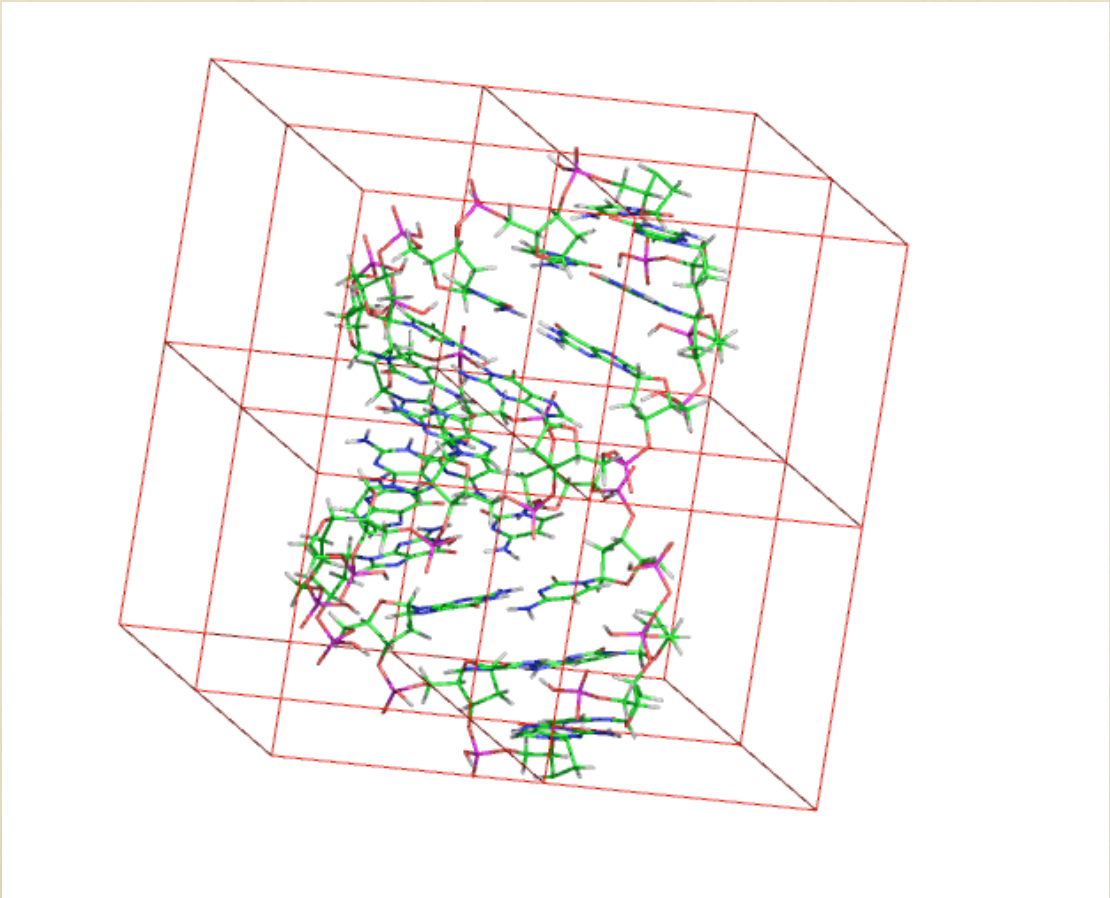
Data blocks



8 CPUs

Parallelization: Orbitals distribution

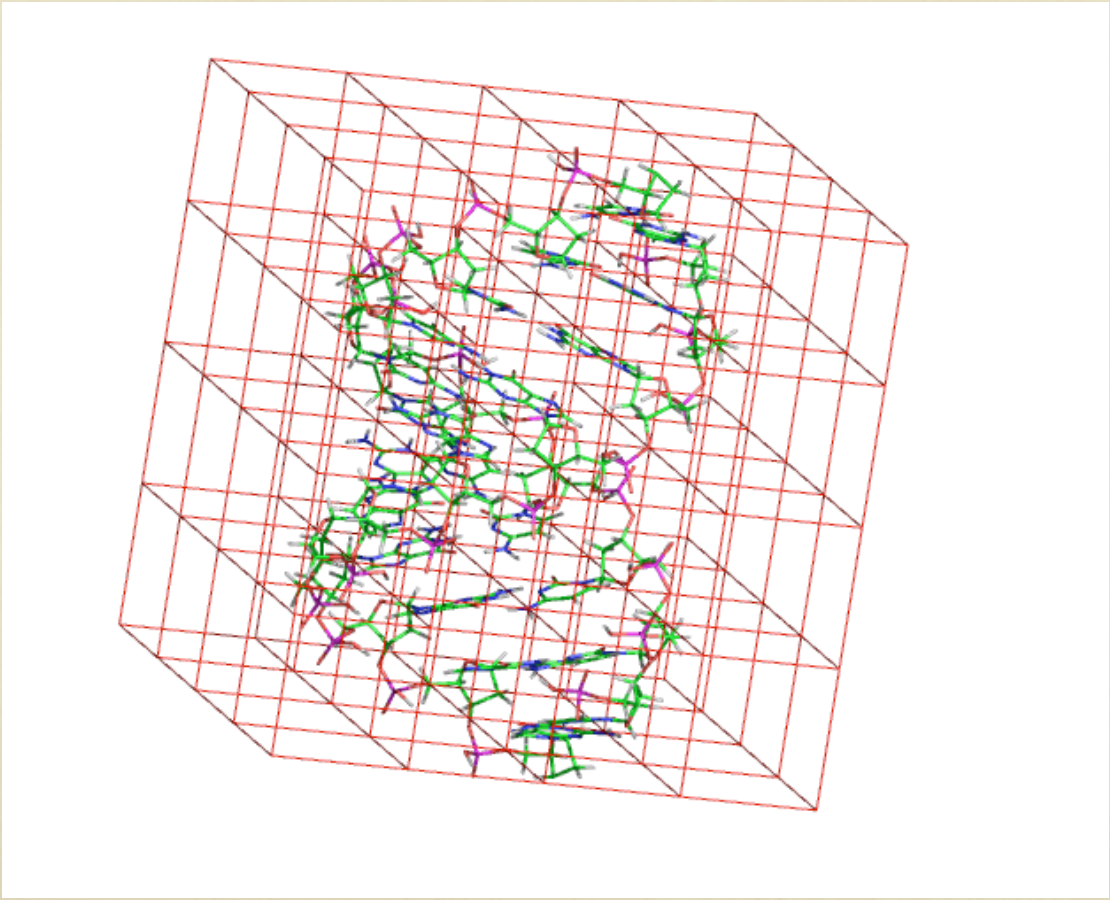
Data blocks



8 CPUs

Parallelization: Orbitals distribution

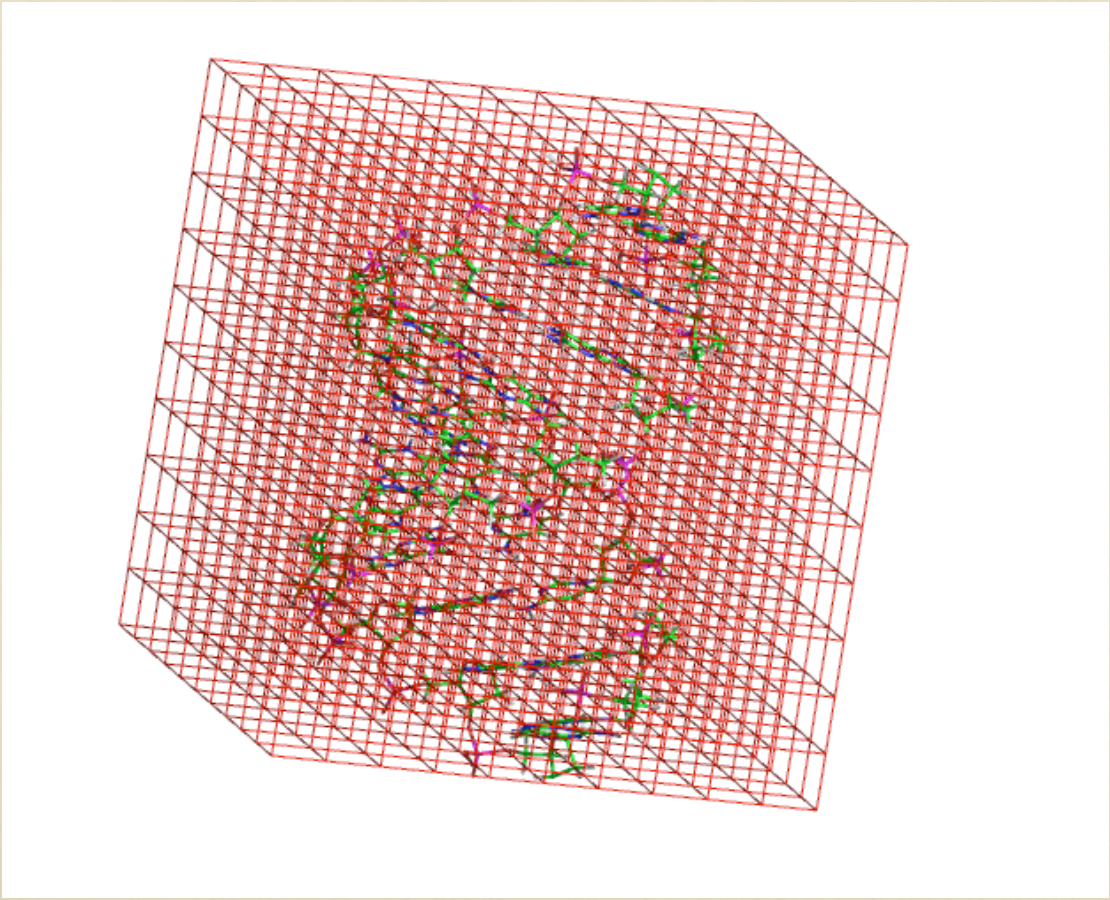
Data blocks



8 CPUs

Parallelization: Orbitals distribution

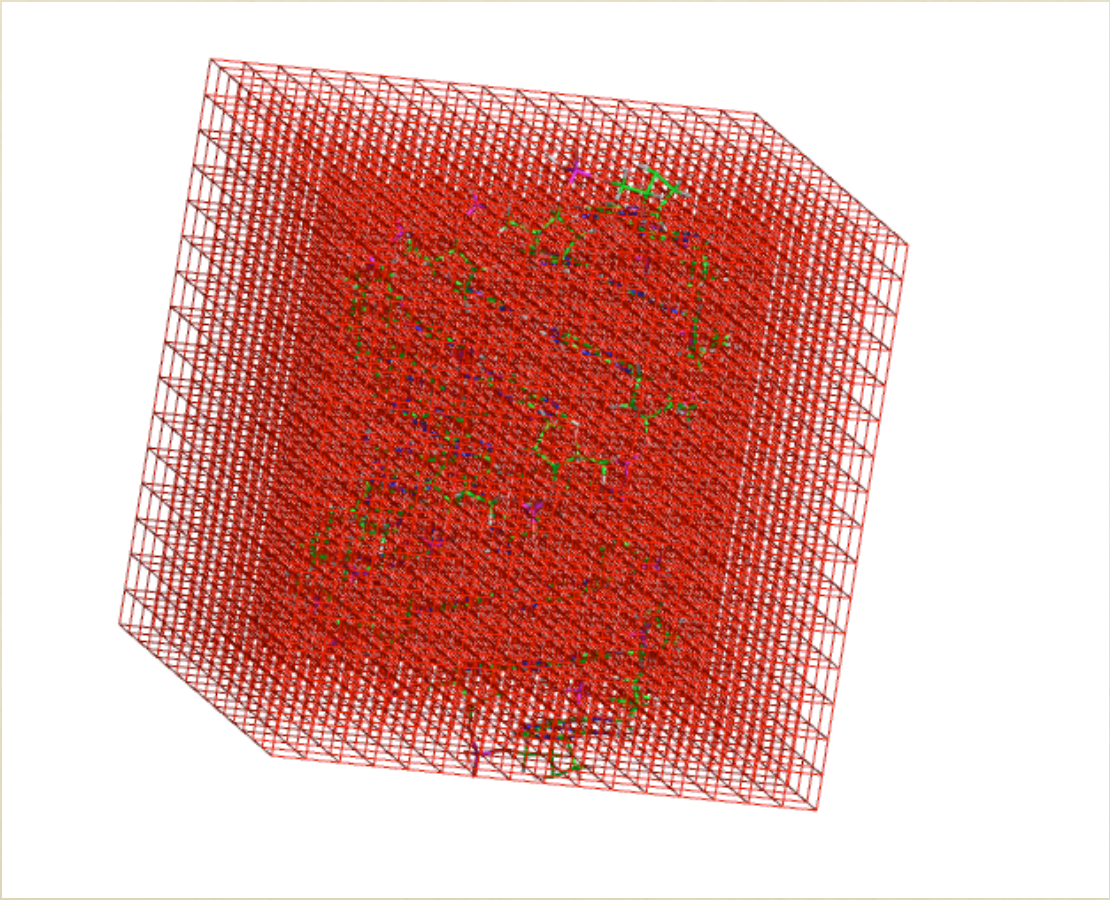
Data blocks



8 CPUs

Parallelization: Orbitals distribution

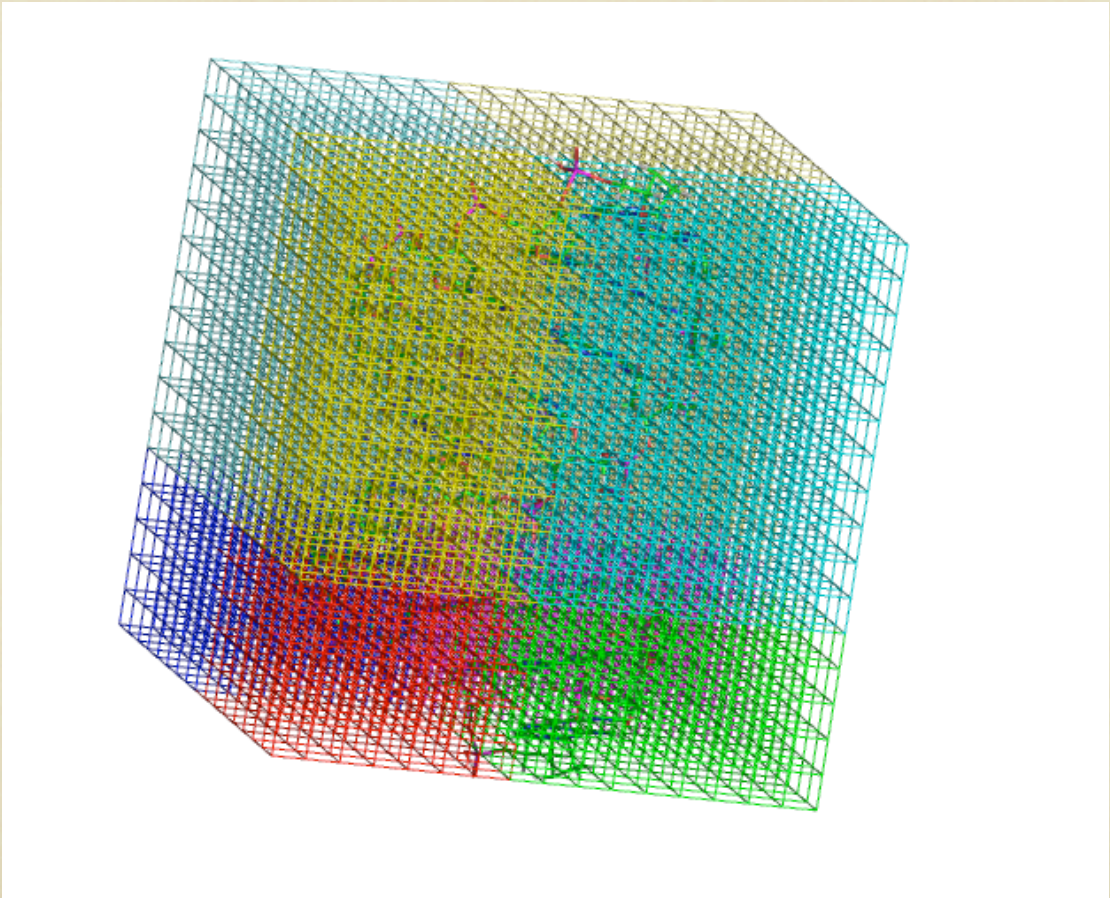
Data blocks



8 CPUs

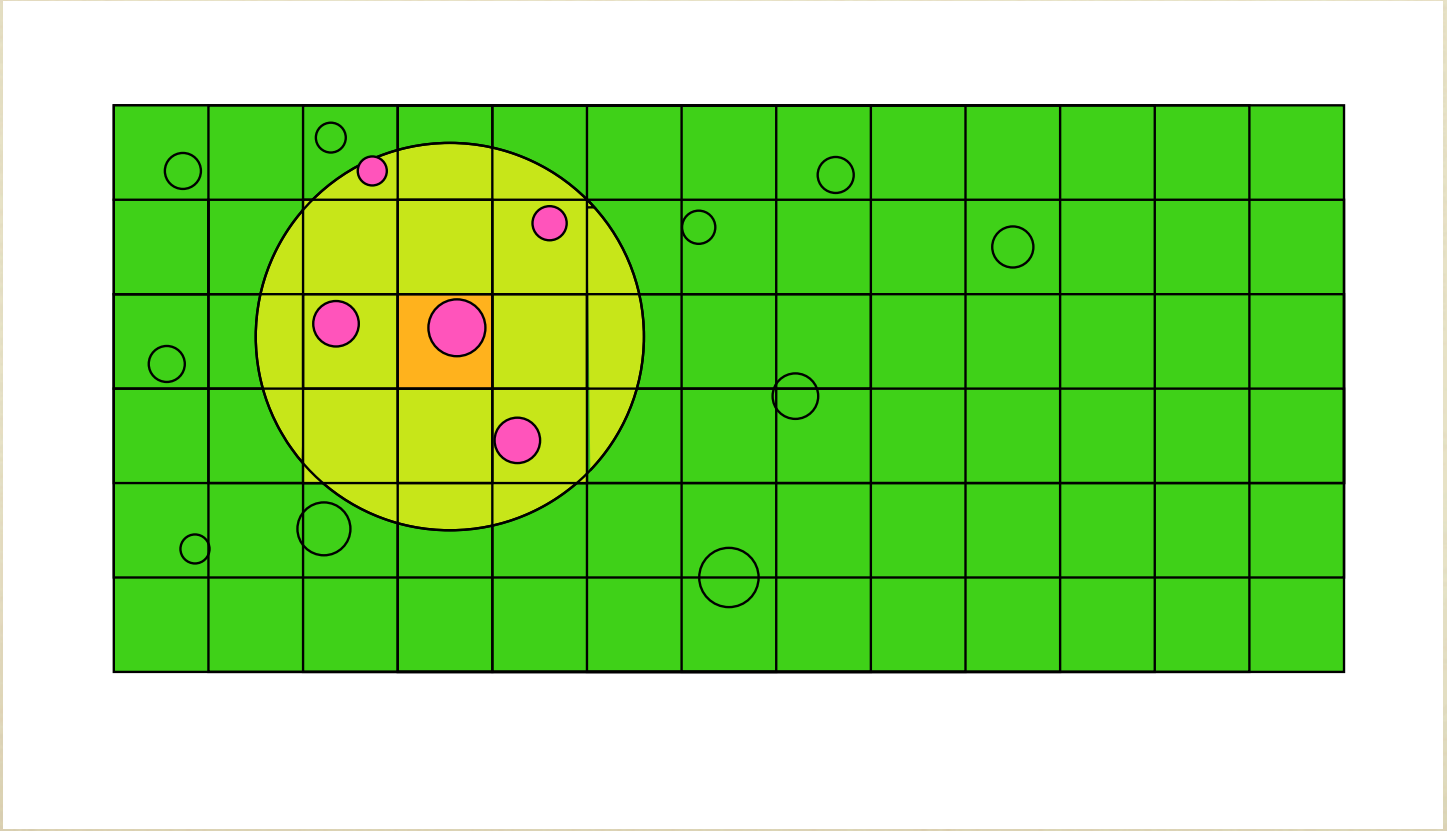
Parallelization: Orbitals distribution

Data blocks



8 CPUs

Diagonalization: Orbitals distribution vs Communication



Orbital Distribution

FDF: `Blocksize`
(control load balancing/communications)

FDF: `Diag.Memory`
(only in case of failure)

Fine-tuning/debugging only:

- `Diag.Use2D`
- `Diag.NoExpert`
- `Diag.DivideAndConquer`
- `Diag.AllInOne`

Order(N) parallelization

`RcSpatial`

(control load balancing/communications)

`ON.LowerMemory`

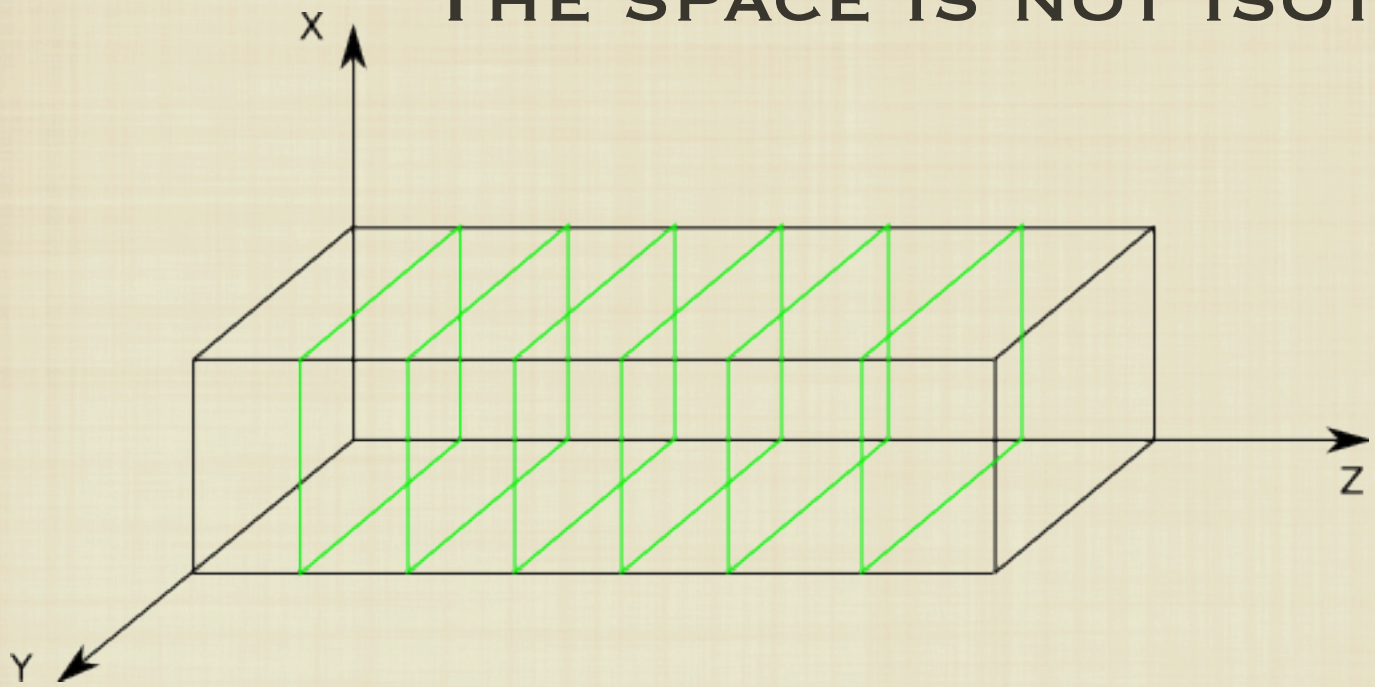
(does what it says!)

DISTRIBUTION OF 3D MESH POINTS

The space is not isotropic!!!

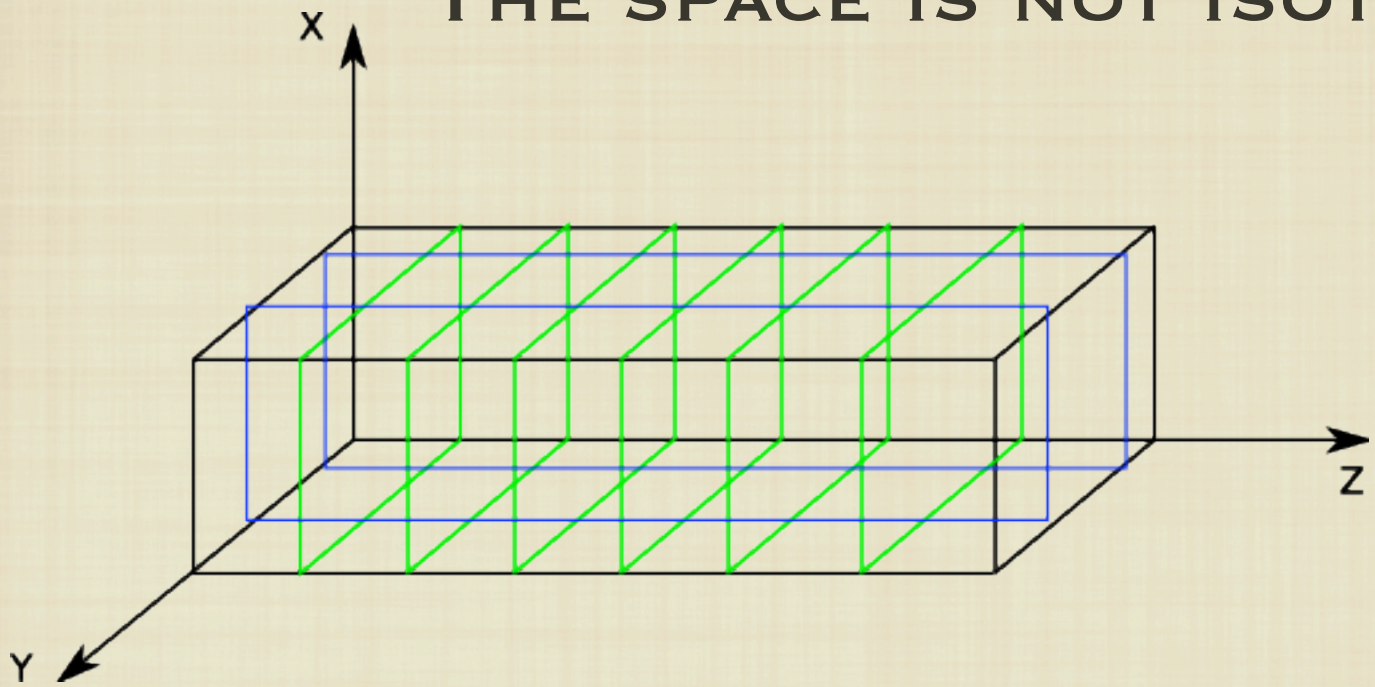
DISTRIBUTION OF 3D MESH POINTS

The space is not isotropic!!!



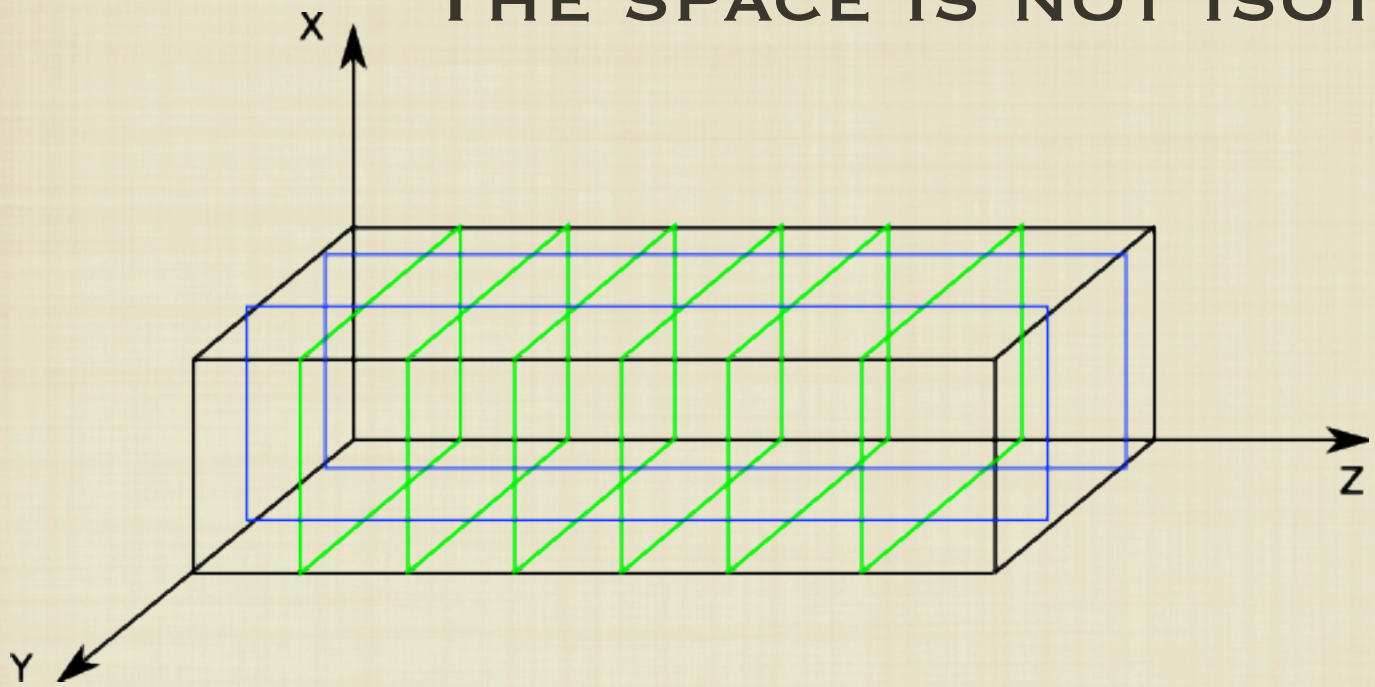
DISTRIBUTION OF 3D MESH POINTS

THE SPACE IS NOT ISOTROPIC!!!



DISTRIBUTION OF 3D MESH POINTS

THE SPACE IS NOT ISOTROPIC!!!

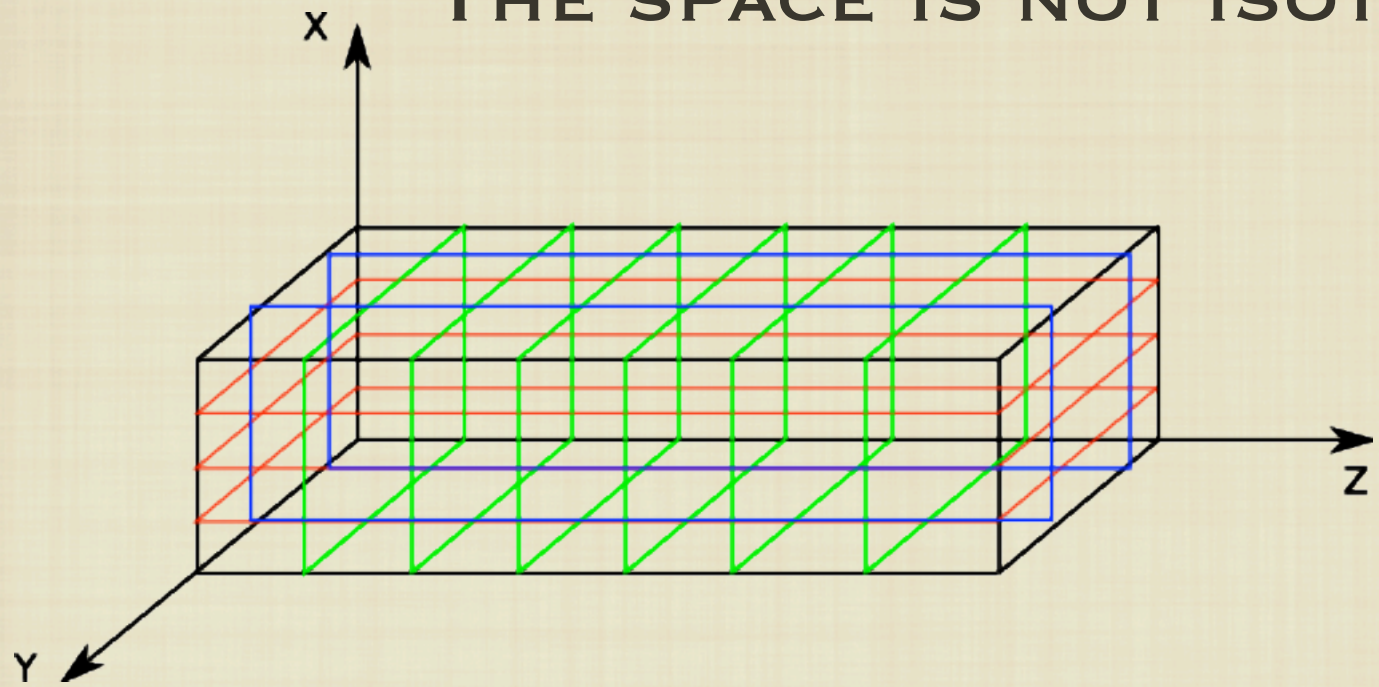


FDF LABEL: ProcessorY

- specifies the dimension of the processor grid in the Ydirection and must be a factor of the total number of processors

Distribution of 3D mesh points

The space is not isotropic!!!



FDF Label: ProcessorY

- specifies the dimension of the processor grid in the Ydirection and must be a factor of the total number of processors

Memory usage for all parallel options

FDF: `DirectPhi`
(cache orbital values)

FDF: `SaveMemory`
(does what it says!)

Trade off memory usage for speed

Timing: parallelization strategy

Look at timings:
scaling by nodes
- for whole program
- for each routine

Play with options and observe results

```
* Maximum dynamic memory allocated : Node    2 =   341 MB

* Maximum memory occured during globalise

timer: CPU execution times:
timer:  Routine       Calls   Time/call    Tot.time        %
timer:  siesta            1   31607.699   31607.699   100.00
timer:  Setup             1       0.680       0.680     0.00
timer:  bands             1       0.000       0.000     0.00
timer:  writewave         1       0.000       0.000     0.00
timer:  KSV_init          1       0.000       0.000     0.00
timer:  IterMD           10    3160.701   31607.009   100.00
timer:  hsparse          11      19.964     219.608     0.69
timer:  overfsm          20       1.608      32.151     0.10
timer:  IterSCF         110     285.160   31367.577    99.24
timer:  kinefsm          20       1.634      32.687     0.10
timer:  nlefsm           20       6.829     136.578     0.43
timer:  DHSCF           110      79.841    8782.531    27.79
timer:  DHSCF1            1       0.830       0.830     0.00
timer:  DHSCF2           10      55.891     558.908     1.77
timer:  REORD           680       0.012       8.352     0.03
timer:  POISON          120       4.515     541.752     1.71
timer:  DHSCF3          110      68.240    7506.443    23.75
timer:  rhoofd          110      20.537    2259.027     7.15
timer:  cellXC          110      30.186    3320.431    10.51
timer:  vmat            110      12.794    1407.332     4.45
timer:  setglobal        19       0.014       0.269     0.00
timer:  setglobalB       10       0.904       9.039     0.03
timer:  setglobalF       10       2.798      27.983     0.09
timer:  gradient        734      15.281   11216.541    35.49
timer:  globaliseF     1468       2.329    3419.667    10.82
timer:  globaliseB     1468       1.925    2825.608     8.94
timer:  globaliseC      743       0.611     454.166     1.44
timer:  ener3           634      15.340    9725.756    30.77
timer:  denmat          100      12.498    1249.832     3.95
timer:  DHSCF4           10      71.518     715.178     2.26
timer:  dfscf            10      63.564     635.640     2.01
```

Getting SIESTA

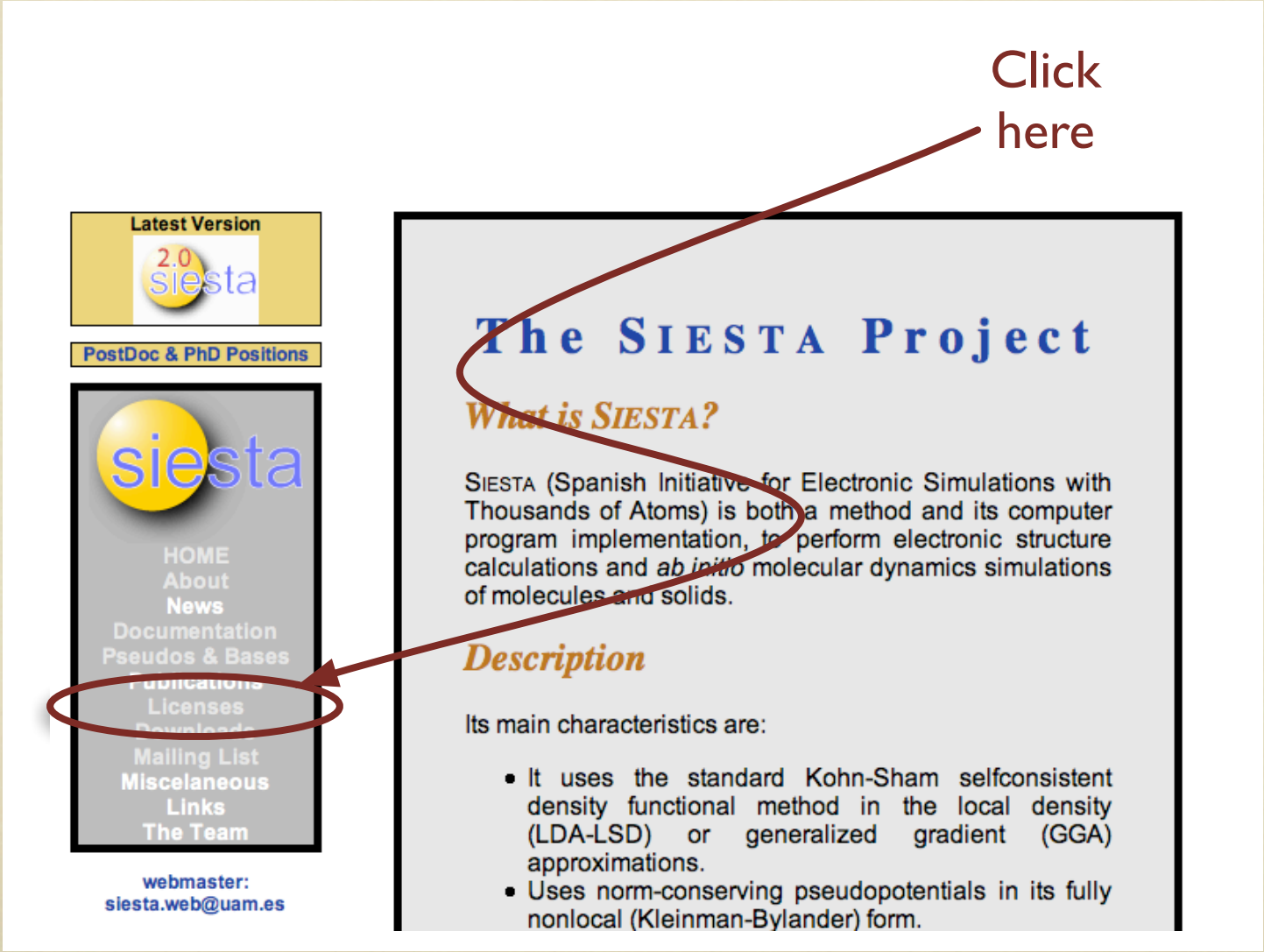
LICENSE

Siesta/Siesta_LICENSE

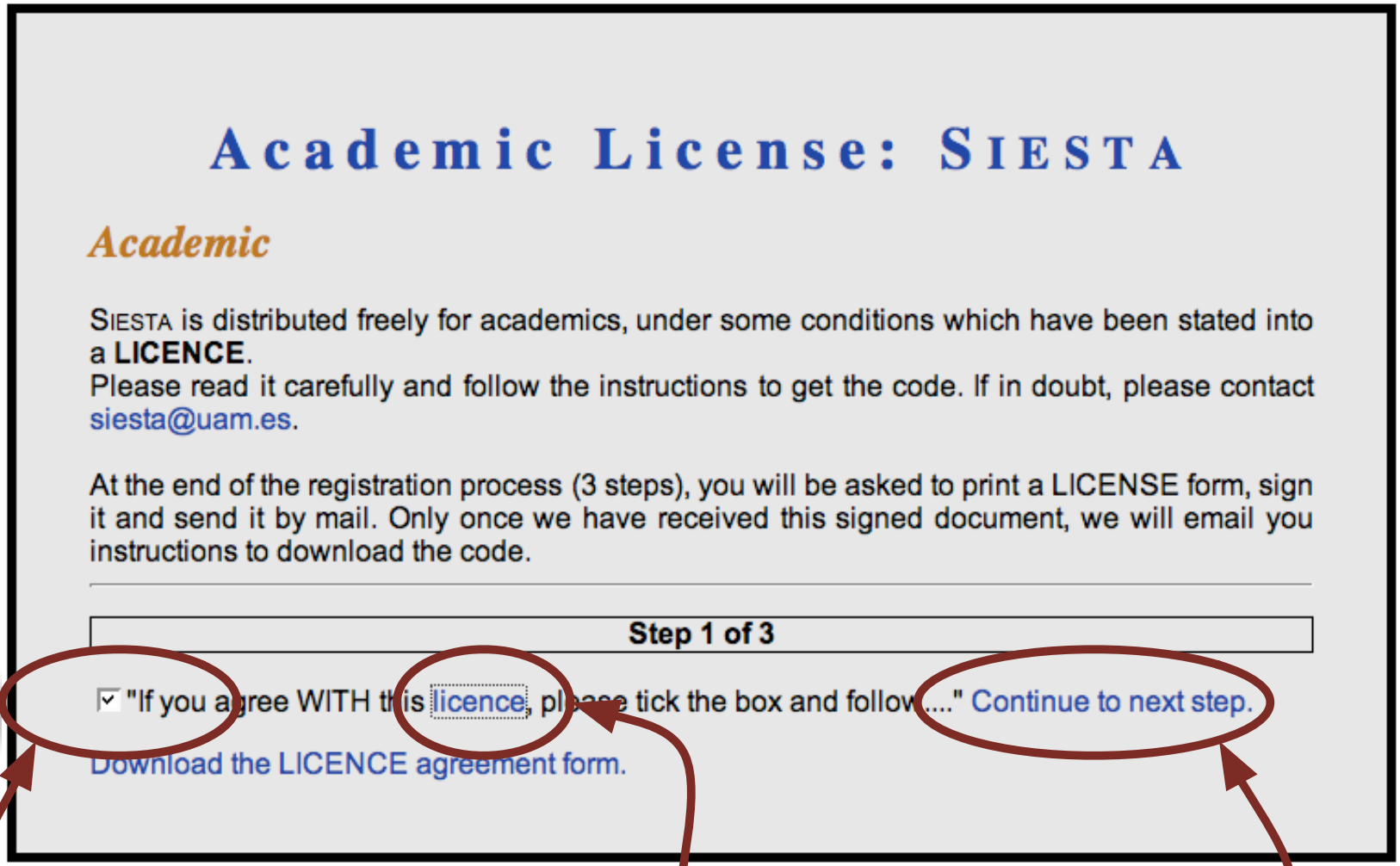
- "academic institutions (including universities and non-military public research laboratories) and to academic purposes, i.e., leading to open publication of results, with neither withhold of information nor intentional delay in publication."
- 1. *Self-consistent order-N density-functional calculations for very large systems*, P. Ordejón, E. Artacho and J. M. Soler, Phys. Rev. B (Rapid Comm.) **53**, R10441 (1996).
  2. *The Siesta method for ab initio order-N materials simulation*, José M. Soler, Emilio Artacho, Julian D. Gale, Alberto García, Javier Junquera, Pablo Ordejón and Daniel Sánchez-Portal, J. Phys.: Condens. Matter **14**, 2745 (2002).
- "The above-mentioned right to use the code is extensive to the members of the research group of the Licensee as long as the use is in collaboration with the Licensee leading to co-authored publication(s)."
- "Licensees are permitted to modify Siesta for their own private use; in any paper containing results wholly or partially derived from the use of a modified version of Siesta, the authors are required to state that a privately modified version of Siesta is used"

Getting a license

http://www.uam.es/siesta



First step


Academic License: SIESTA
Academic
SIESTA is distributed freely for academics, under some conditions which have been stated into a LICENCE.
Please read it carefully and follow the instructions to get the code. If in doubt, please contact siesta@uam.es.
At the end of the registration process (3 steps), you will be asked to print a LICENSE form, sign it and send it by mail. Only once we have received this signed document, we will email you instructions to download the code.
Step 1 of 3
"If you agree WITH this licence, please tick the box and follow ...." Continue to next step.
Download the LICENCE agreement form.
Click here...
...and here...
...and here!

Read the license ...

SIESTA ACADEMIC LICENCE for INDIVIDUALS

Motivation and Preamble

The SIESTA program has been devised for its general use in research within the academic community. Some conditions have been defined for the use, distribution, and modification of SIESTA, which the authors consider fair and within the common fair practices in the academic community.

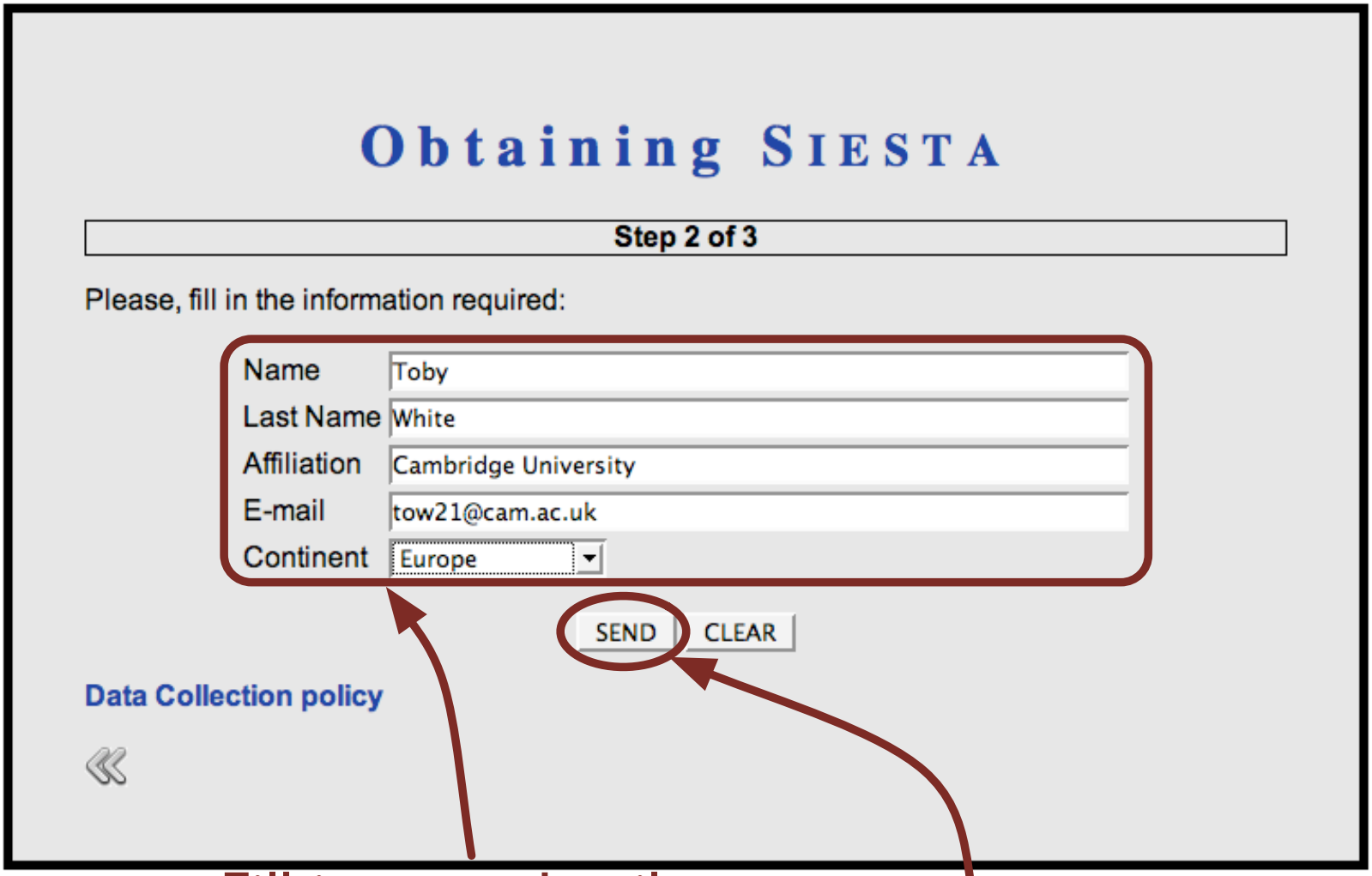
1. Definitions

"The Authors" are:

- Emilio Artacho, University of Cambridge
- Julian Gale, Curtin University of Technology, Perth
- Alberto García, Universidad del País Vasco, Bilbao
- Javier Junquera, Rutgers University
- Pablo Ordejón, ICMAB-CSIC, Barcelona
- Daniel Sánchez-Portal, Universidad del País Vasco, San Sebastián
- José M. Soler, Universidad Autónoma de Madrid

Although we, the Authors, acknowledge that a limited number of auxiliary subroutines were written by (or are based on previous subroutines written by) other authors consider that the implementation of all the basic algorithms of the SIESTA program is ours.

Second Step


Obtaining SIESTA
Step 2 of 3
Please, fill in the information required:
Name
Toby
Last Name
White
Affiliation
Cambridge University
E-mail
tow21@cam.ac.uk
Continent
Europe
SEND
CLEAR
Data Collection policy
Fill in your details...
...and press Send

Third and Last step

Obtaining SIESTA

Step 3 of 3

Download the SIESTA LICENCE (PDF file) and send **one signed** hard copy to:

Patricia Álvarez c/o Pablo Ordejón
Instituto de Ciencia de Materiales de Barcelona, CSIC
Campus de la UAB
08193 Bellaterra
Barcelona
Spain

Download the LICENCE

Once we receive your signed LICENSE, we will send you an email with instructions to download the package.

Thank you, Toby White

Sign the license and
PUT IT IN THE POST!

Optional: Mailing list

❖ *Sign up to mailing list*

Email to: LISTSERV@LISTSERV.UAM.ES

no subject, contents:

```
SUBSCRIBE SIESTA-L your_name
```

Receive confirmation ...

Email again:

no subject, contents:

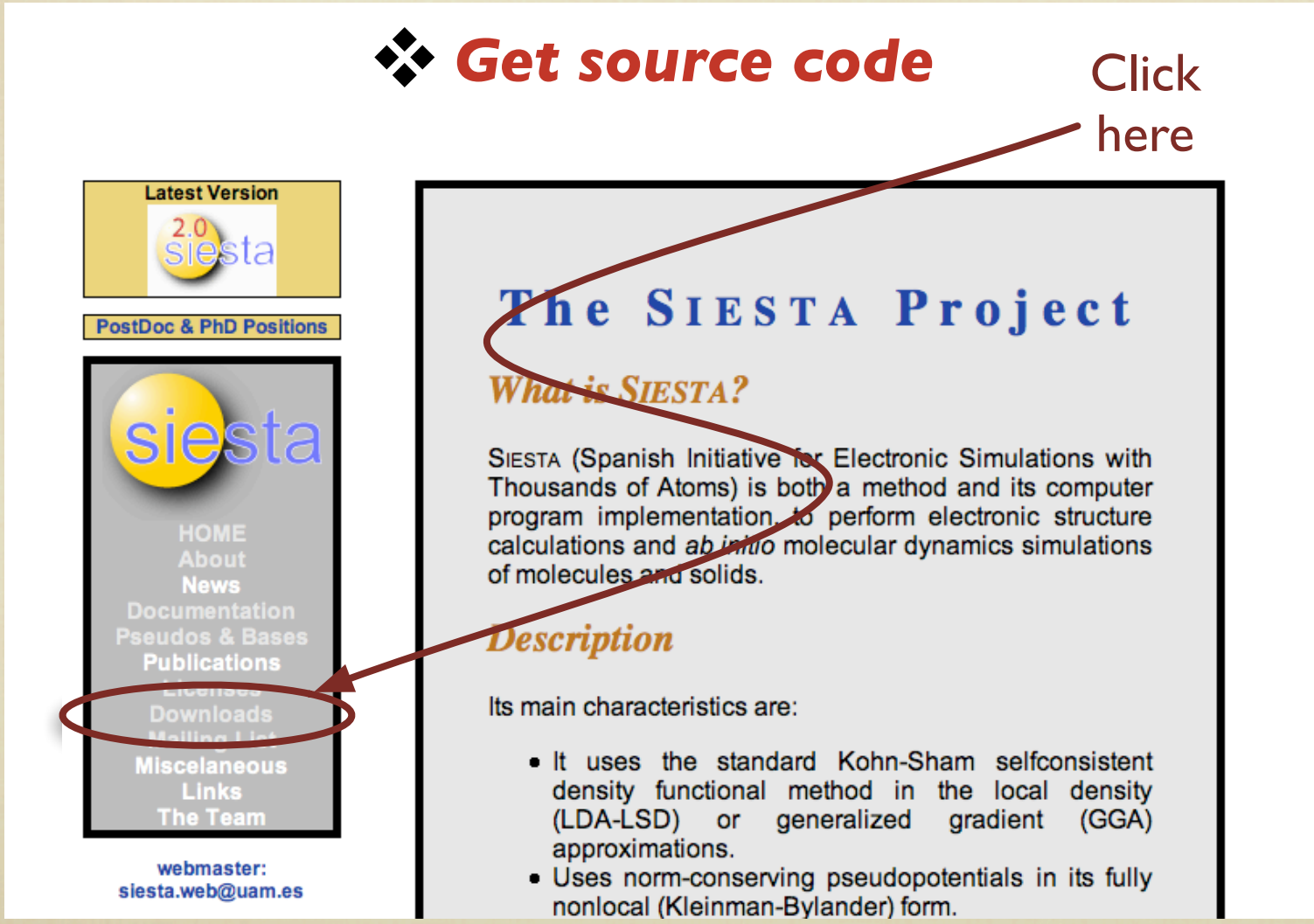
```
PW ADD your_password
```

Read archives at:

http://cygni.fmc.uam.es/mailing-list

Get The Source Code

http://www.uam.es/siesta



Pseudos and BASIS

Generate them (see lecture and practical session)

Ask on mailing list

Check the database!!

http://www.uam.es/siesta

Other resources

Andrei Postnikov's utilities (and see his talk and exercises)
http://www.home.uni-osnabrueck.de/apostnik/download.html

Lev Kantorovich
http://www.cmmp.ucl.ac.uk/~lev/codes/lev00/index.html

CMLComp (and see talk and exercises on SIESTA XML)
http://cmlcomp.org

GDIS
http://gdis.sf.net

Enjoy your siesta experience!