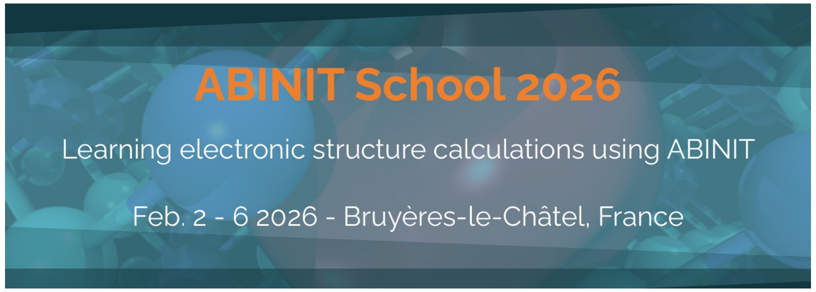

ABINIT School 2026

Learning electronic structure calculations using ABINIT

Feb. 2 - 6 2026 - Bruyères-le-Châtel, France

Using ABINIT on High-Performance Computers

Hands-on session

Titanium near melting

A first use of ABINIT on HPC

- **Objectives**
  - Learn how to execute ABINIT on a modern supercomputer
  - Use all parallelism levels: MPI, openMP, GPU
  - Use several diagonalization algorithms
- **Material: titanium**
  - First use a small simulation cell (32 atoms, 256 bands)
  - Then increase the cell size (128/256 atoms, 1024/2048 bands)
  - 2 k points, very small plane-wave cut-off

Preliminary information

Basic concepts for parallelism

On a supercomputer, you can use:

- **Message Passing parallelism (MPI)**
  Computing units (cores) access to their own memory and shared messages

- **Multithreading (openMP)**
  Computing units (cores) share the same memory on a « node »
  Inti ROME nodes are made of 128 cores,
  Inti GH200 nodes are made of 288 cores

- **Graphics Processing Units (GPU)**
  Parts of the computation can be offloaded to GPU
  Inti GH200 nodes are connected to 4 GPUs (Nvidia H200 90Gb)

Preliminary information

Run ABINIT on *Inti* supercomputer

- All the necessary files can be found in **`ABISCHOOL/handson_parallelism/`**. Note that `REF/` folder contains the solution of the exercices ; you are supposed to try by yourself before reading it.
- ABINIT documentation for is available at: https://docs.abinit.org
- You should execute ABINIT in each subirectory (`1-xxx`, `2-xxx`, etc.)
- `x.abo` (output) and `abinit.log` (standard output) will be located in the current directory.
- `TMP/` directory contains temporary files, as well as *slurm* output and error files.
- `ATOMICDATA/` folder contains the pseudopotential.

Preliminary information

Run ABINIT on *Inti* supercomputer

- Edit a batch submission script (example given on the next slide and in the shared folder)

```
gedit job.sub
```

- Then submit the script with:

```
ccc_msub job.sub
```

- You should adapt the following entries:

`-r <job_name>`	: the name of the job. Customize your job name to recognize it
`-T <#_seconds>`	: the time limit
`-n <#_MPI>`	: the number of MPI processes
`-c <#_threads_per_MPI>`	: the number of openMP threads per MPI process
`-q <partition>`	: the name of the partition: *rome* for CPU, *gh200-bxi* for CPU/GPU
`-E '--reservation=XXXXXXX'`	: change this every day
`xxxx.abi`	: the ABINIT input file

- Useful commands:

`ccc_mpp [-u <user>]`	: to list the running/waiting jobs
`ccc_mdel [job-id]`	: to cancel a job
`ccc_mpinfo`	: to see the status of queues

Preliminary information

The *Inti/slurm* submission script

```
#!/bin/bash
#For the ABINIT school only - Change it every day
#MSUB -E '--reservation=Formation-Abinit-gpu-20260204'

#MSUB -r abinit_school # Name of the job

#Adapt this according to your needs
#MSUB -n 4              # Number of MPI processes
#MSUB -c 32             # Number of threads
#MSUB -T 1800           # Max. time in seconds
#MSUB -q gh200-bxi      # Partition: rome (cpu) or gh200-bxi (cpu/gpu)

#Dont touch this
#MSUB -o TMP/%I.o       # Standard output. %I is the job id
#MSUB -e TMP/%I.e       # Error output.    %I is the job id
#MSUB -m work,scratch   # List of used file systems

module use /ccc/[. . .]/ABISCHOOL/ENV/MODULES

module purge
module load abinit/${SLURM_JOB_PARTITION}

[. . .]
```

ABINIT keywords - 1

A few keywords related to parallelism

- **np_spkpt** : number of **MPI processes** attributed to the **k-point** (and **spin**) parallelism
- **npband** : number of **MPI processes** attributed to the parallelism over **bands**
- **OMP_NUM_THREADS** : number of « **openMP** » **tasks** attributed to the parallelism over **bands**. It is not defined in the ABINIT input file but in the environment (i.e. the submission script)
- **gpu_option** : keyword to activate the use of **GPU**:
  « GPU_DISABLED » ➞ no GPU, « GPU_OPENMP » ➞ use of GPU
- **wfoptalg** : **algorithm** used to solve the eigenproblem :

  `wfoptalg=111` = Locally Optimal Block Preconditioned Conjugate Gradient (LOBPCG), default

  `wfoptalg=114` = Chebyshev Filtering

ABINIT keywords - 2

How to configure parallelism over bands

- **`nblock_lobpcg`** : relevant only to the **LOBPCG algorithm**, not the *Chebyshev Filtering* algorithm. For LOBPCG case, the number of blocks has to be configured. Each block is computed in parallel using **`npband`** MPI processes and **`OMP_NUM_THREADS`** tasks. Blocks are processed one after the other. Keep **`nblock_lobpcg`** small (default is 1), increase it for very large systems.

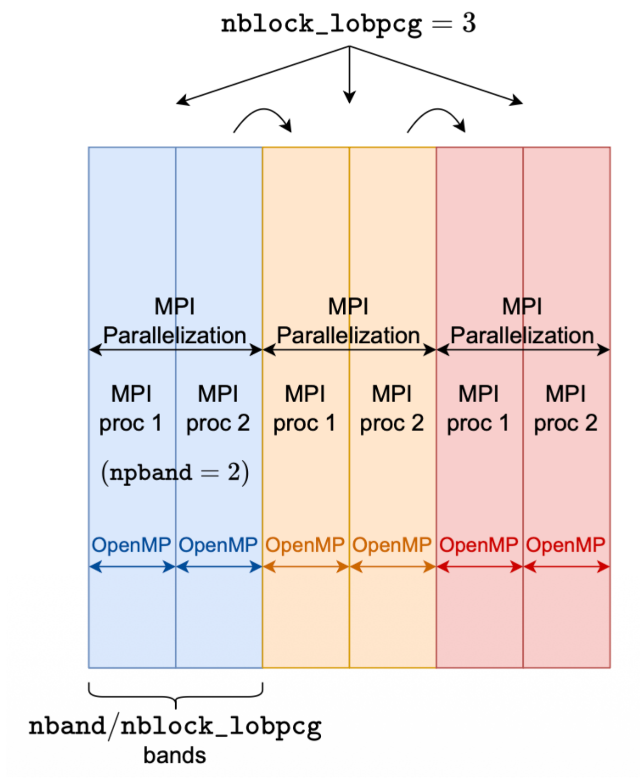
Magic formula:

$$\underbrace{n_{band}}_{\text{Total number of bands (nband in input file)}} = \underbrace{nblock_{LOBPCG}}_{\text{Number of blocks (nblock\_lobpcg in input file)}} \times \underbrace{np_{band}}_{\text{Number of MPI processes (npband in input file)}} \times \underbrace{M}_{\text{Number of bands per MPI (OMP\_NUM\_THREADS in submission script has to divide it)}}$$

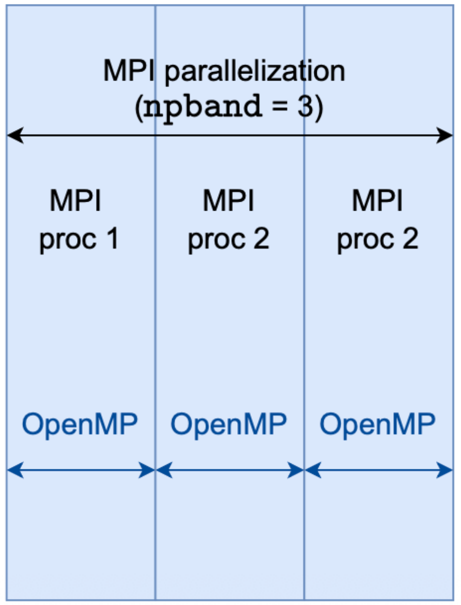
For Chebyshev Filtering, just consider that $nblock_{LOBPCG} = 1$

ABINIT parallelism – Sketch

LOBPCG algorithm

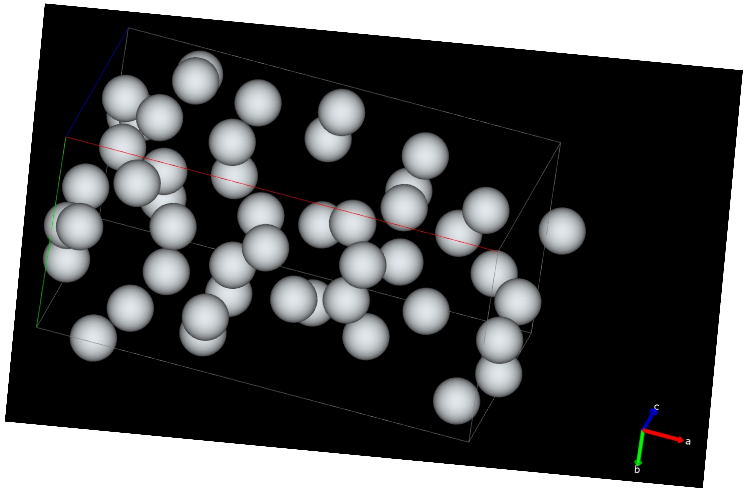


Chebyshev filtering algorithm



1- Sequential run

First run – Small simulation cell – 32 atoms – 256 bands

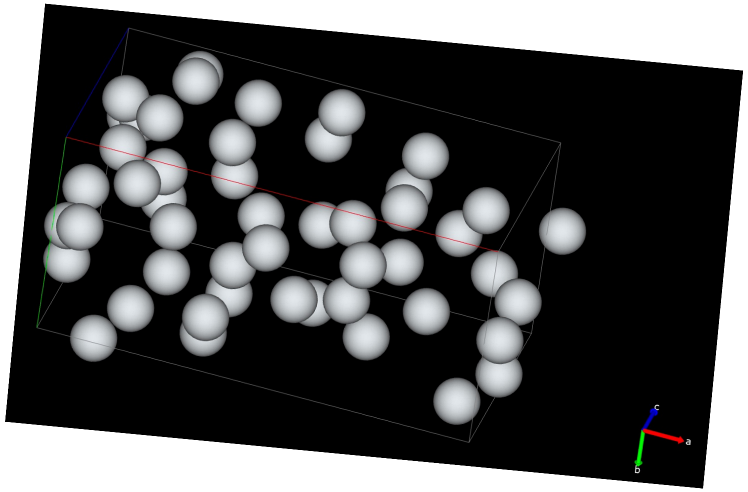


- Enter the `1-sequential-32atoms` directory
- Edit `ti32.abi` and `job.sub` files
  Set all keywords related to parallelism to run in sequential mode
  What would be the most efficient number of blocks?
- Run ABINIT on 1 CPU core
- Note the time needed by this computation
  « `Proc.0 individual time` » at the end of the `abo` file

Tip: we are using **LOBPCG** algorithm. In sequential, it is more efficient when it uses blocks of smaller size.
Try with 1 block, then with 16 blocks.
(this number has to divide `nband`)

2- Parallelism over k-points (and spin)

Small simulation cell – 32 atoms – 256 bands

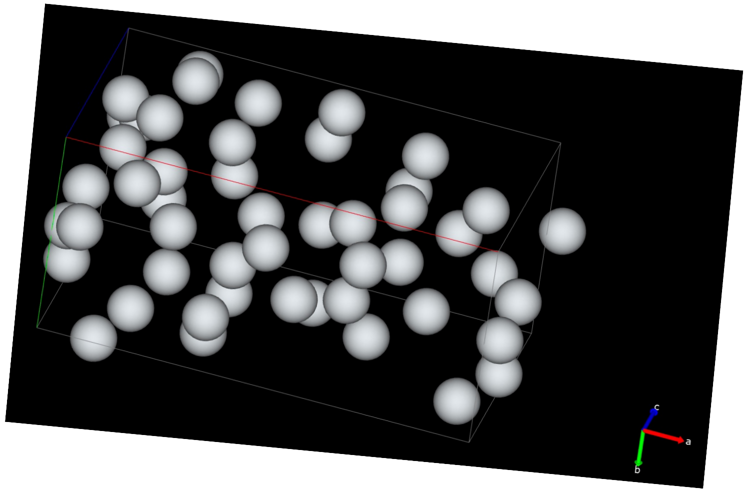


- Enter the `2-mpi-kpt-32atoms` directory
- Edit `ti32.abi` and `job.sub` files
  *Set all keywords related to parallelism to run with the **k-points** distributed on **2 cores***
- Run ABINIT on 2 CPU cores
- Note the time needed by this computation « `Proc.0 individual time` » at the end of the `abo` file and compare it with the sequential case

Tip: we are using **LOBPCG** algorithm. In sequential, it is more efficient when it uses blocks of smaller size.

3- Parallelism over bands

Small simulation cell – 32 atoms – 256 bands

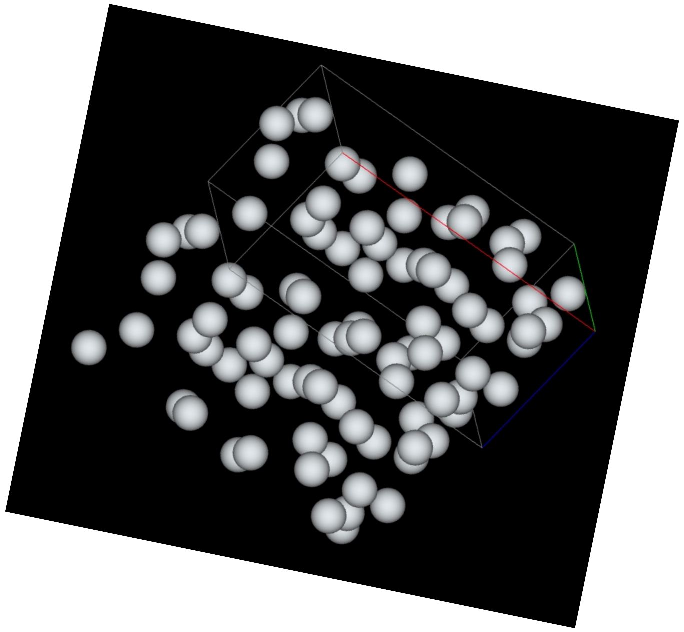


- Enter the `3-mpi-kptband-32atoms` directory
- Edit `ti32.abi` and `job.sub` files
  *Set all keywords related to parallelism to run with the **k-points** distributed on **2 cores** and **bands** distributed over **16 cores***
- Run ABINIT on 32 CPU cores
- Note the time needed by this computation and compare it with the previous cases

Tip: using **LOBPCG** algorithm with parallelism over bands, having larger blocks would be optimal. Try with 2 blocks.

4- Parallelism over bands – follow-up

Medium-sized simulation cell – 64 atoms – 512 bands

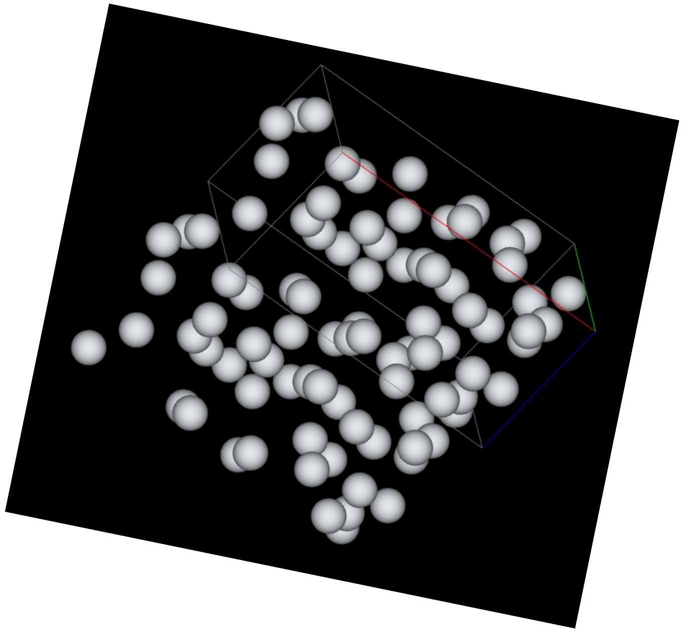


- Enter the `4-mpi-64atoms` directory
- Edit `ti64.abi` and `job.sub` files
  *Set all keywords related to parallelism to run with the **k-points** distributed on **2 cores** and **bands** distributed over **32 cores***
- Run ABINIT on 64 CPU cores
- Note the time needed by this computation

Tip: for this case, try to use LOBPCG algorithm with **2 blocs**

5- Hybrid MPI-openMP parallelism

Medium-sized simulation cell – 64 atoms – 512 bands



- Enter the `5-mpi-openmp-64atoms` directory
- Edit `ti64.abi` and `job.sub` files
  Run exactly the same calculation as before
  *with the **k-points** distributed on **2 cores***
  *and **bands** distributed over **4 cores***
  *using **8 openMP threads***
- Run ABINIT on 64 CPU cores
- Note the time needed by this computation
  and compare it to the previous run (using only MPI)
  At constant ressources, is it worth using hybrid parallelism ?

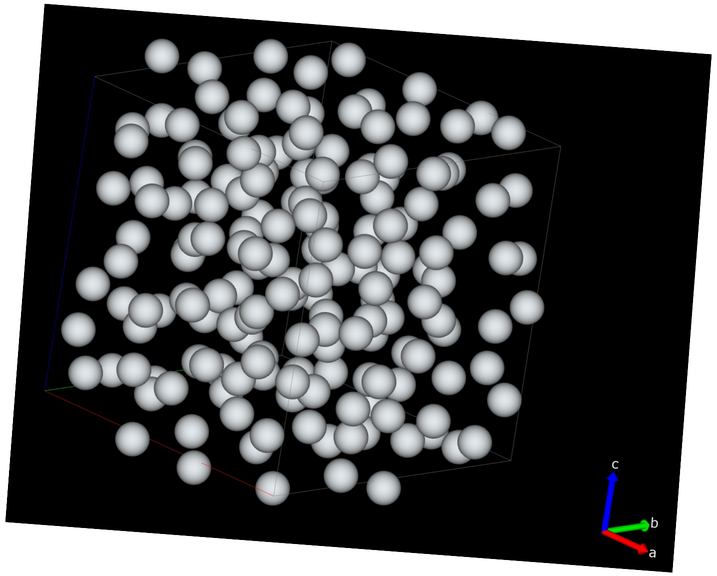
Tip: for this case, **2 blocks**

Note: the `cprj_in_memory` keyword activates an optimized implementation

6- Chebyshev-filtering algorithm

Larger simulation cell – 128 atoms – 1024 bands

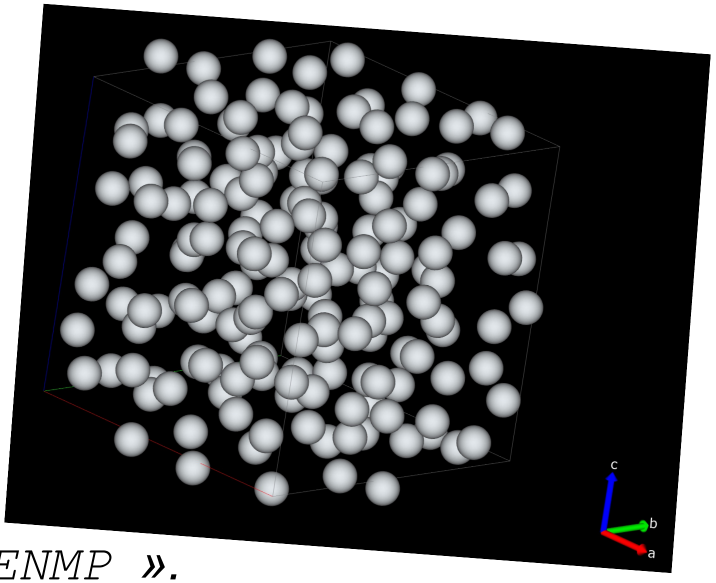
- Enter `6-mpi-openmp-chebfi-128atoms`
- Edit `ti128.abi` and `job.sub` files
  - Change to the **Chebyshev filtering** algoritm, i.e. set `wfoptalg=111`. Note that `nblock_lobpcg` is no more needed (no more blocks).
  - *Set all keywords related to parallelism to run with the **k-points** distributed on **2 cores** and **bands** distributed over **32 cores** using **8 openMP threads***
- Run ABINIT on 512 CPU cores
- Note the time needed by this computation



7- And now using one GPU!

Simulation cell – 128 atoms – 1024 bands

- Enter `7-gpu-128atoms`
- Edit `job.sub`
  - You must use the `gh200-bxi` partition (CPU/GPU) if not already done
  - Each GPU is associated to one MPI process, so run ABINIT with one 1 MPI process.
- Edit `ti128.abi`
  - *Activate the GPU by setting gpu_option to « `GPU_OPENMP` ».*
  - *Run exactly the same calculation as before with the **k-points** distributed on **1 core** and **bands** distributed over **1 core***
- Run ABINIT on one node



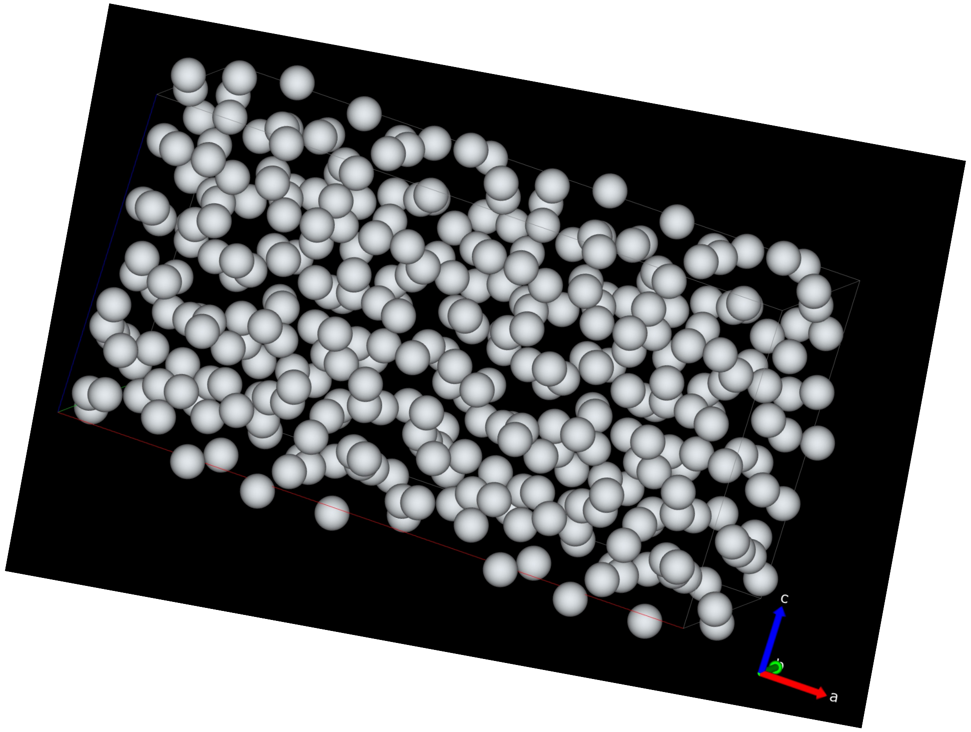
Tip: check in the **`abinit.log`** file that the GPU is detected

Note: with the GPUs, you must impose « #MSUB –c 72 »

8- Larger cell in multi-GPU mode

Large simulation cell – 256 atoms – 2048 bands

- Enter `8-gpu-256atoms`
- Edit `ti256.abi` and `job.sub` files
  *Set all keywords related to parallelism to run with the **k-points** distributed on **2 cores** and **bands** distributed over **2 cores** using **8 openMP threads***
- Run ABINIT on one node
- Note the time needed by this computation
  Not so much, right?



9- Large cell – Follow-up

Large simulation cell – 256 atoms – 2048 bands
If time permits

- Still in `8-gpu-256atoms`
- Run on more GPUs:
  Try 8 GPUs, 16 GPUs…
- What about 4096 bandes?
  Is it possible?
- Try to improve the convergency parameters, increase `ecut`, `nstep`, …