

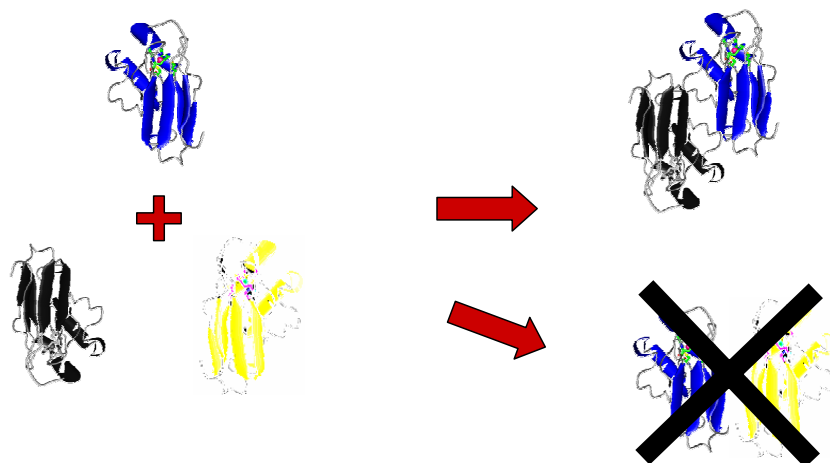
Project *PSI-Wat*

ab initio simulations of *protein/surface* interactions mediated by Water

Stefano Corni, Arrigo Calzolari, Rosa di Felice (CNR-INFM S3)
Giancarlo Cicero (Politecnico di Torino, IT)
Alessandra Catellani (CNR-IMEM, Parma IT)
Carlo Cavazzoni (CINECA, Bologna IT)

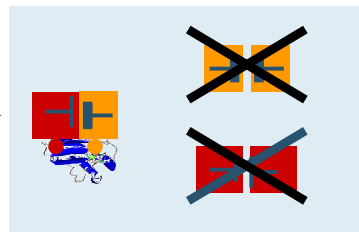
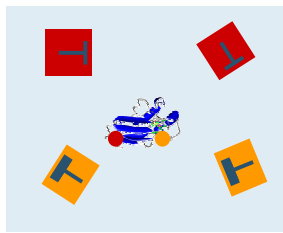
Proteins on inorganic surfaces

Exploiting the intrinsic capabilities of proteins



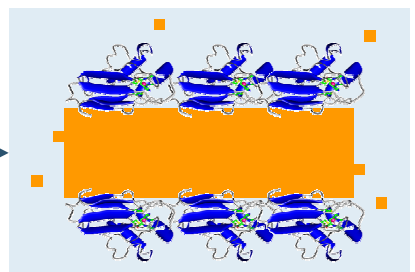
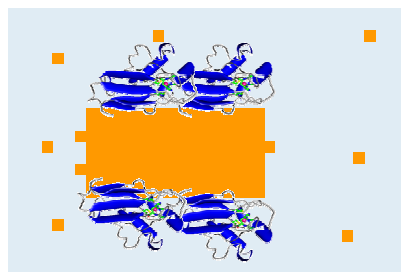
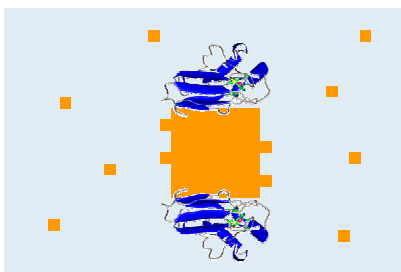
- Proteins can **recognize** other proteins/molecules

Examples of possible applications

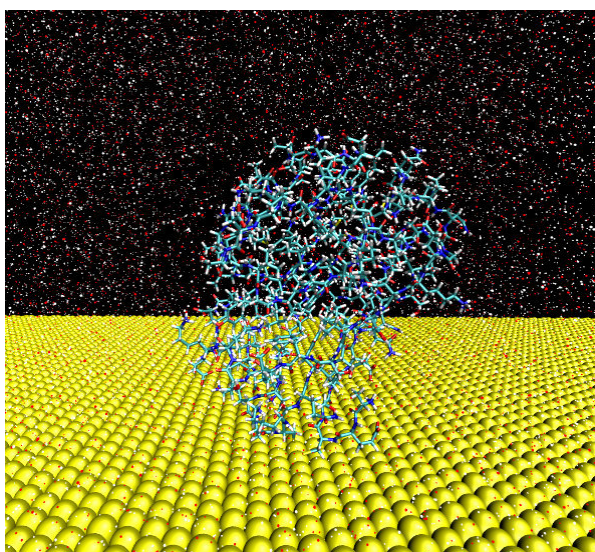


• Use proteins to guide the self-assembly of inorganic nanodevices (**proteins as a smart glue**)

• Use protein to **control the growth** of inorganic materials, e.g., to obtain nanocrystals with new shapes



Complexity of the systems

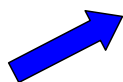
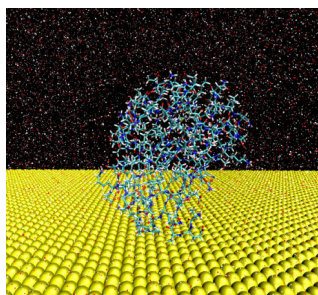


They represent a **computational challenge**:

- Very **large** systems (10^3 - 10^4 atoms)
 - Protein + Inorganic Surface + Solvent
- Require **long** simulation **time**
 - Instant quantities not useful → averaging
 - Interesting dynamics on long time-scale
- Involve **interactions** of **different origins**
 - Chemical bonds, Coulomb, dispersion...
 - Different methods most suitable for different portions

Computational strategies

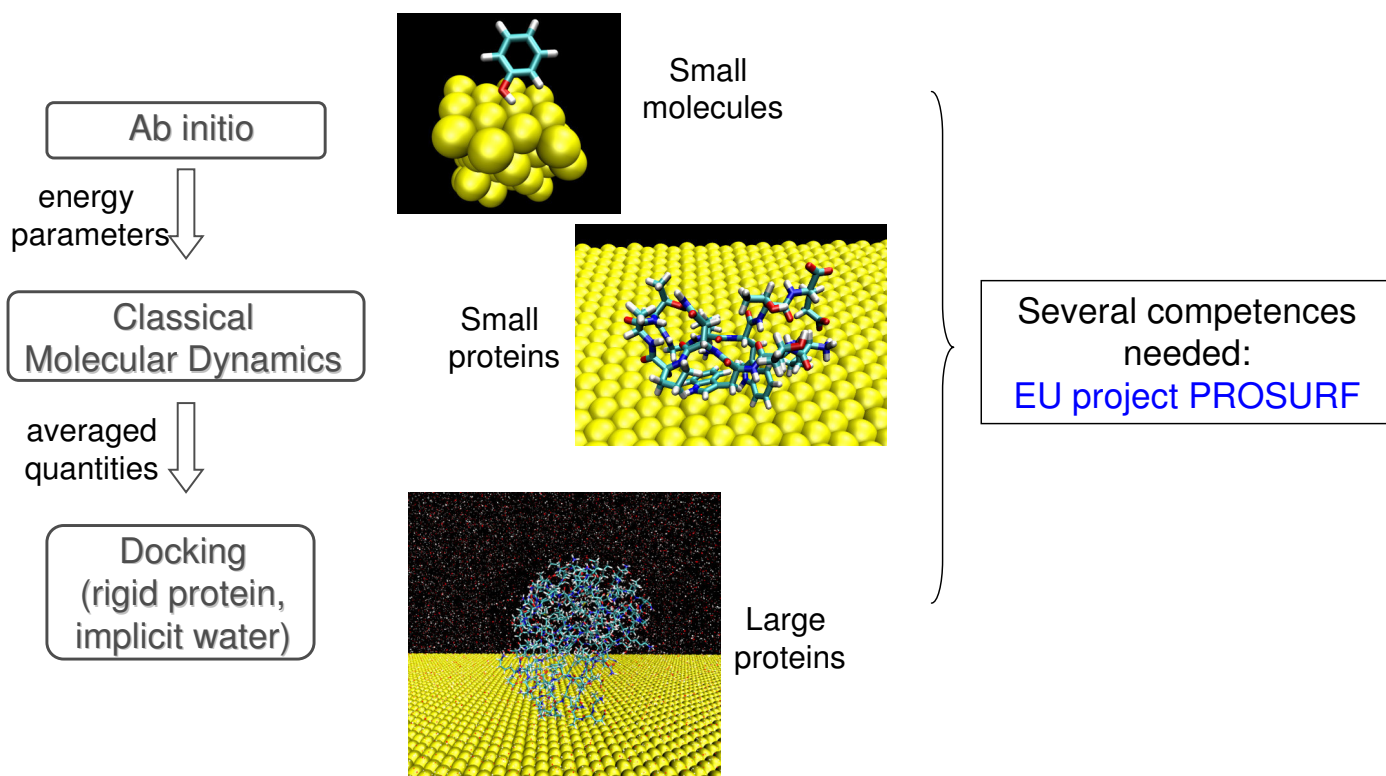
Complexity of the system



Multiscale modelling
relatively general but
long term

**Exploiting extreme
supercomputer power**
feasible for some important cases

Multiscale modelling

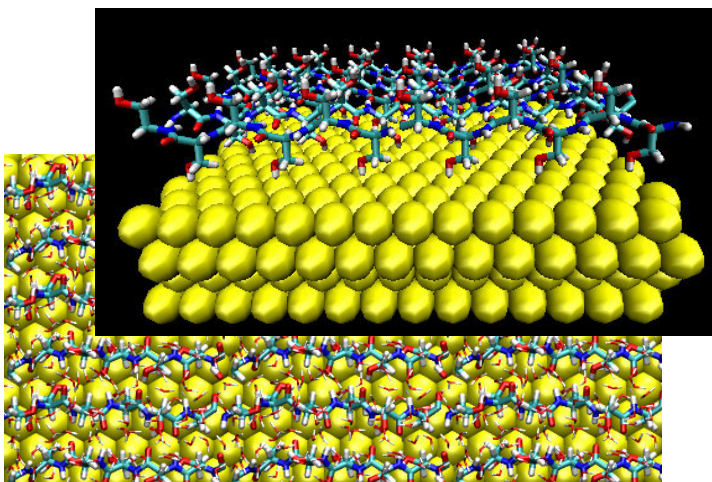


Exploiting extreme supercomputer power

Protein+Surface+Water **all** ab initio!

However:

- Exceptional calculation
- Requires very good scalability of the code
- Small model proteins, max. tens of ps



20 ps requires ~350 000 cpu h on IBM SP5
(~60 days with 256 cpus and perfect scalability)



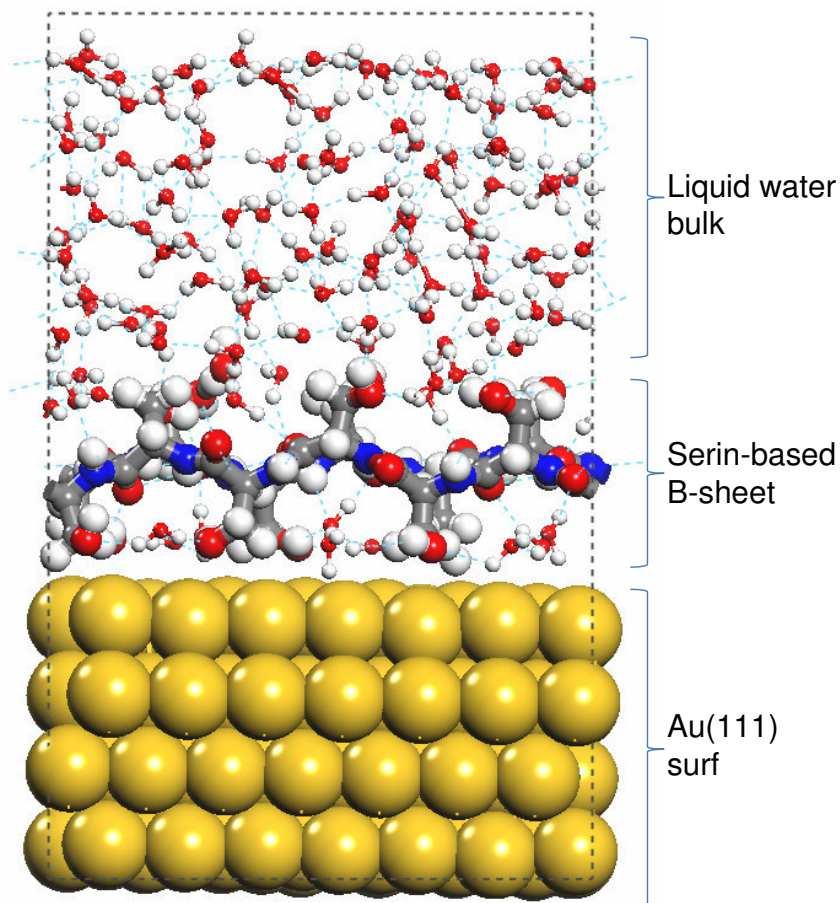
Require huge computational resources:
EU initiative DEISA

PSI-Wat project

System

Open questions:

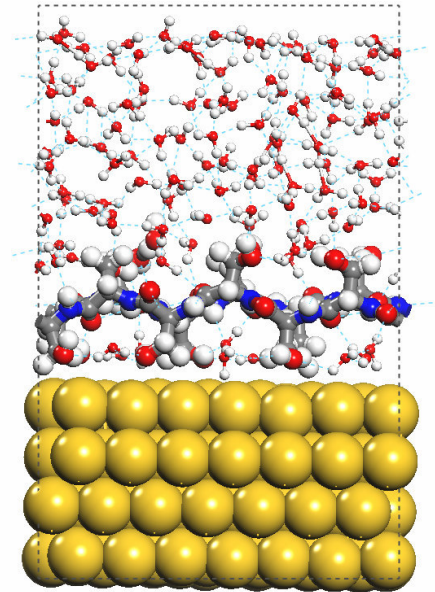
- How is the protein/surface interaction?
- Does the protein bind on surface?
- What is the role of the water?
- Is the nanostructured Au surface hydrophobic or hydrophilic?



System

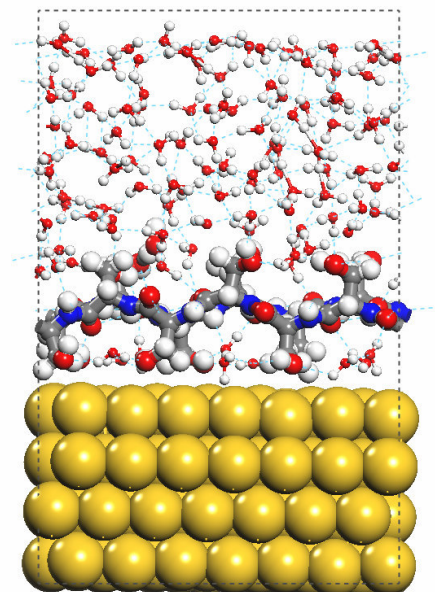
Simulation details

- *ab initio* Car-Parrinello MD
- Cell (10.59 x 20.529 x 32.656) Å³
- #atoms = 587
 - 4 layers of Au(111) → (112 Au)
 - periodic b-sheet → 130 (36 C + 12 N + 23 O + 59 H)
 - 115 heavy water molecules → (115 O + 230 D)
- # electrons = 2552
- XC=PBE, USPP, plane waves
- Ecut = 25 Ry, ecutrho = 200 Ry
- Kpt mesh = Gamma
- Fictitious electronic mass $\mu=450$ au
- Simulation time step $\delta t=1.69$ fs
- 12 ps of thermal equilibration at T= 400K (Nose' thermostat)



COMPUTATIONAL DETAILS

- CODE = cp.x (quantum espresso package)
- Running machine = *MareNostrum* (Barcelona Supercomputing Center)
- Total CPU time = 350000 CPU hours.
- Total task per run = 200 CPUs
- Array leading dimensions = (88, 176, 280)
- Total memory allocation = 100Gbyte

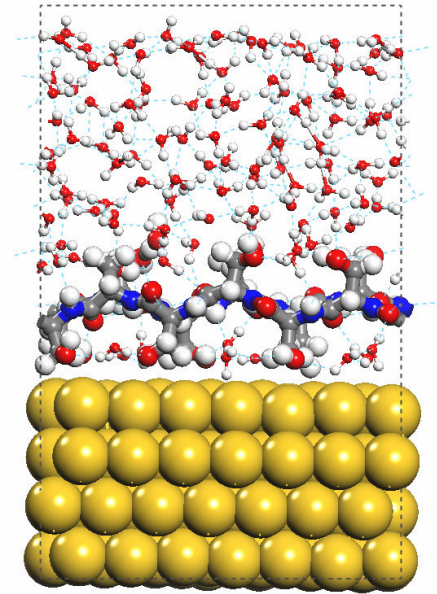


RESULTS (ASAP)

WORK IN PROGRESS

SIMULATION STILL RUNNING

(~6 ps of molecular dynamics at 400K)



Enabling of CP code of QE

- Task Group
- Ortho
- memory allocation
- Profiling and Optimizations

Force and Charge loop:

```
do i = 1, nb
  compute parallel fft
end do
```

the parallelization is limited by the number of plane in the fft on a grid NR1xNR2xNR3 there is little gain to use more than NR3 proc

relevant parameter: -ntg XX

to be specified on the execution line, XX being the number of Groups no more limited to BG/L, very good results also on Cray XT3

Force and Charge loop:

```
redistribute band
do i = 1, nb, ngroup
  compute ngroup fft at the same time
end do
```

we can scale up to NR3 x NGROUP processor.

This cost an additional ALLTOALL and memory (NGROUP times the size of the charge)

Ortho Matrix Multiplication

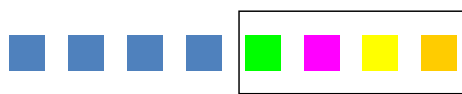
ortho was not parallelized in CP, because compared to FFT was much less expansive for small/medium size system.

The problem is that ortho scale as the Nb^3 (Nb number of bands) For large system this become quickly dominant

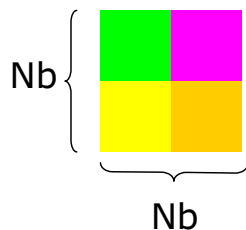
Now ortho is parallelized using a fast parallel algorithm for matrix multiplication, know as Cannon algorithm (see Module/ptoolkit.f90).

Limitation: matrix should be distributed on a number of proc which is a perfect square ($NP \times NP$).

This is not a limitation on the number of procs, the parallelization is performed on a subset of the number of processors (ortho group)



in a run with 8 processors,
ortho group has 4 procrs (2x2)



Matrixes are block distributed to the ortho group.

In this case is possible to use a mixed parallelization
MPI+OpenMP using SMP library

Now matrix diagonalization uses a different data distribution (matrix rows are cyclically distributed to proc), we are rewriting the algorithm to let the diagonalization to work on block distributed data (more efficient)

*Relevant parameter: ortho_para = XX
in electrons namelist, XX is the number of proc to be used in the ortho group. If not specified the code will try to get the optimal size of the group*

Memory Consideration

There are array dimensioned: Nuber of Atoms * Number of Bands
replicated on all node.

For large systems this quickly saturate the memory of nodes.

They have been distributed, adding communication when
required.