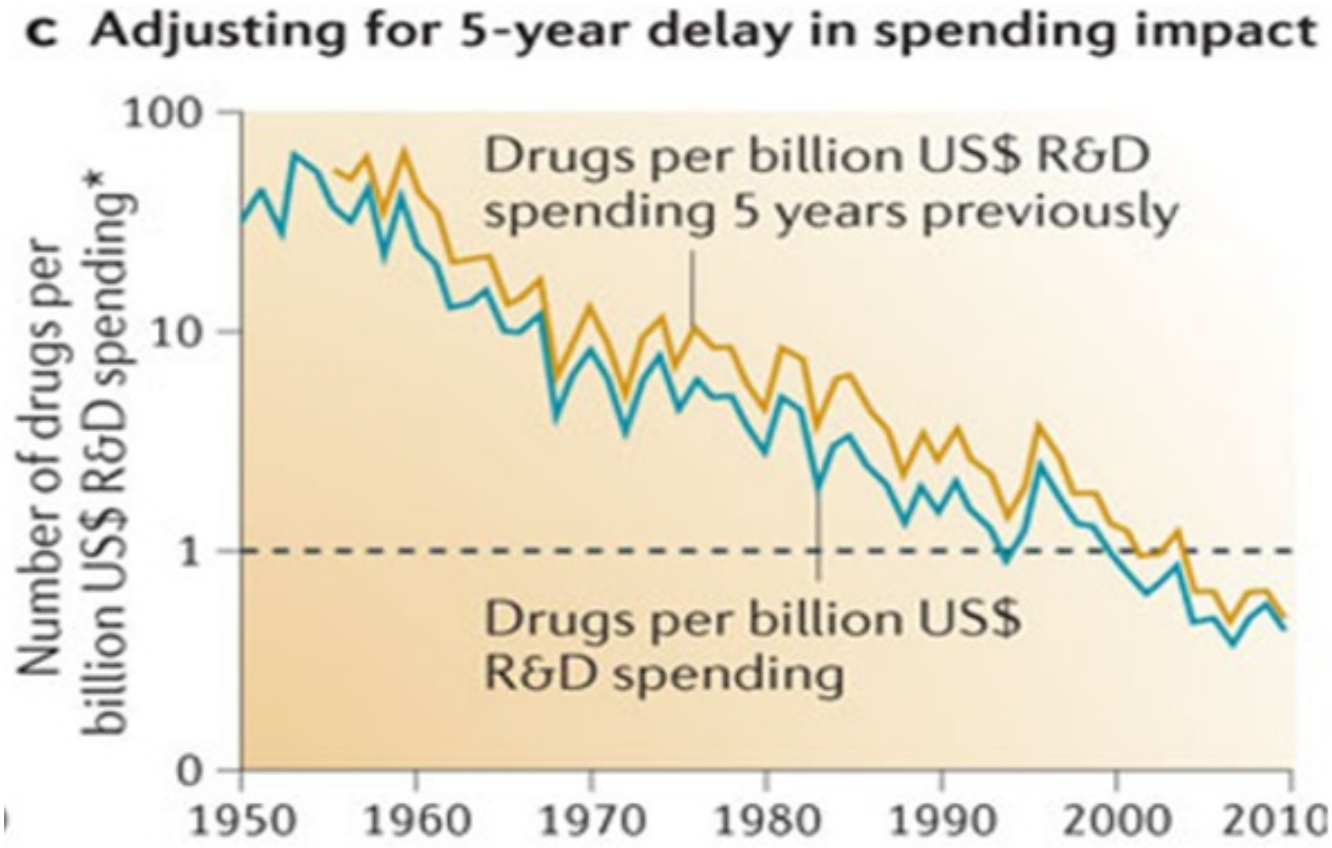
A massively parallel non-equilibrium method for fast calculations of the binding free energy in Molecular Dynamics Simulations of ligand-receptor systems.

Piero Procacci

Dipartimento di Chimica, Universitá di Firenze

Report on research activity on the CRESCO infrastructure, 2014/2015

"Eroom" Law in drug discovery

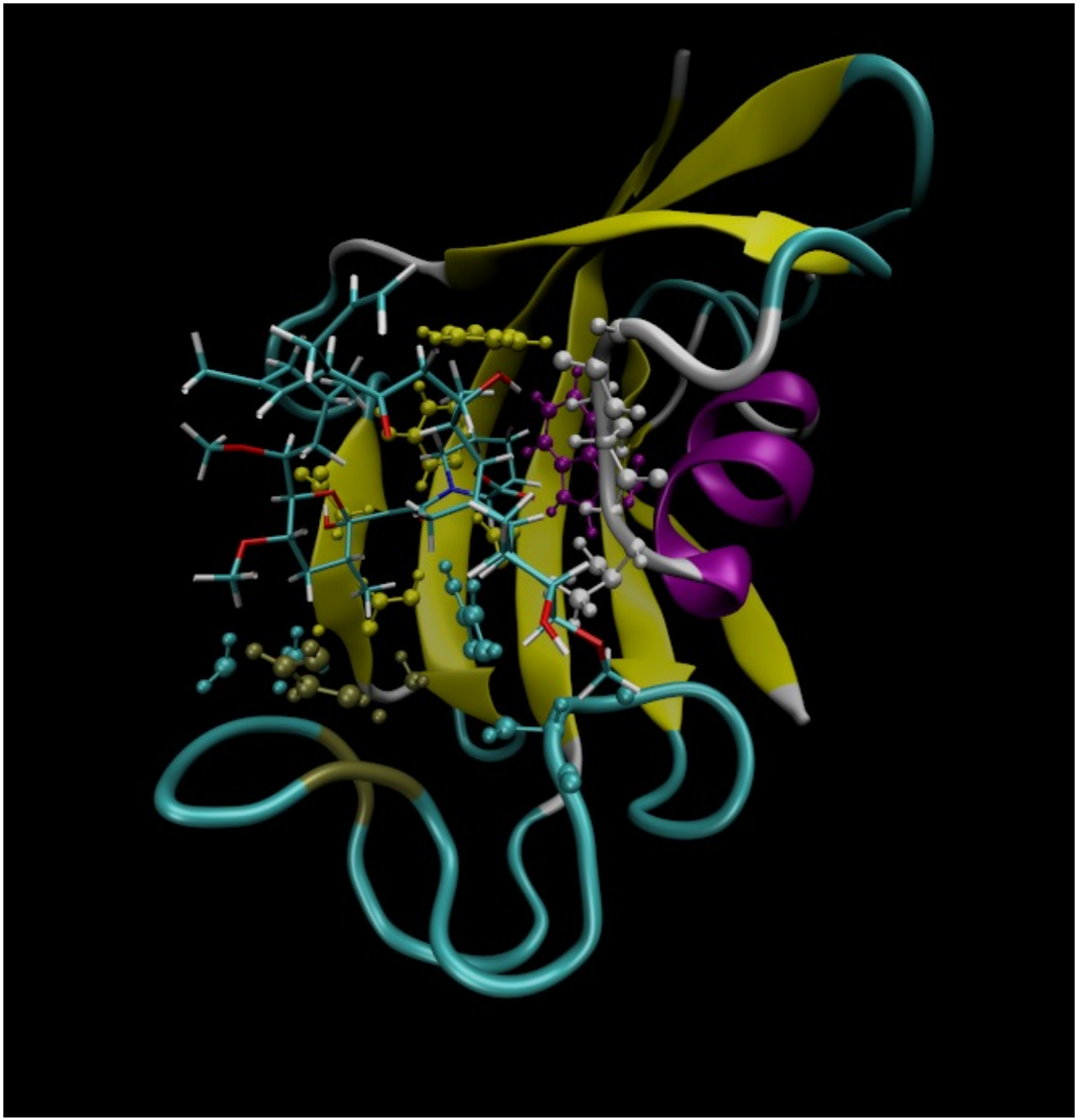


- Better than Beatles
- Tougher regulations
- Brute force HTS

Nature Reviews Drug Discovery 11, 191-200 (March 2012)

HTS: high investment, low yield.

In Silico screening



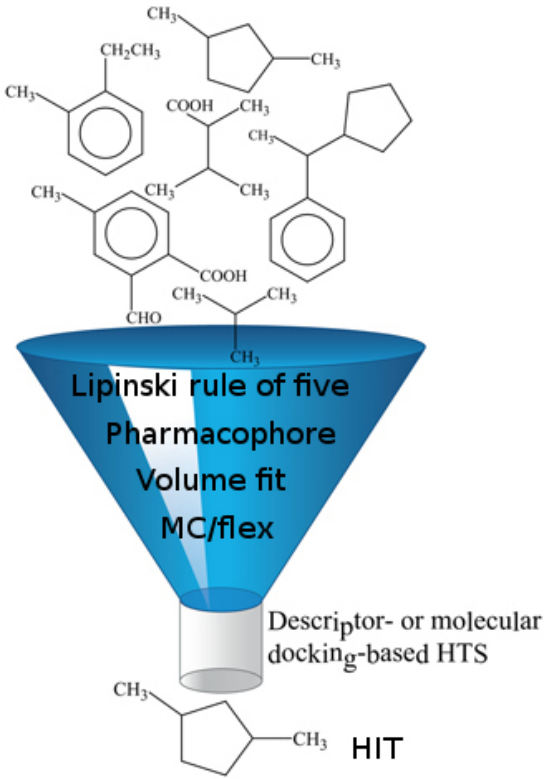
In Silico screening

$$\Delta G = < \Delta E >_{\mathbf{x}} - T < \Delta S >_{\mathbf{x}}$$

$< .. >$ = Average over the ensemble

$\mathbf{x}$ = solvent and drug receptor coordinates

HT**V**S, Molecular Docking : $\Delta G \simeq \Delta \mathbf{E}$



Expectations

In Silico screening

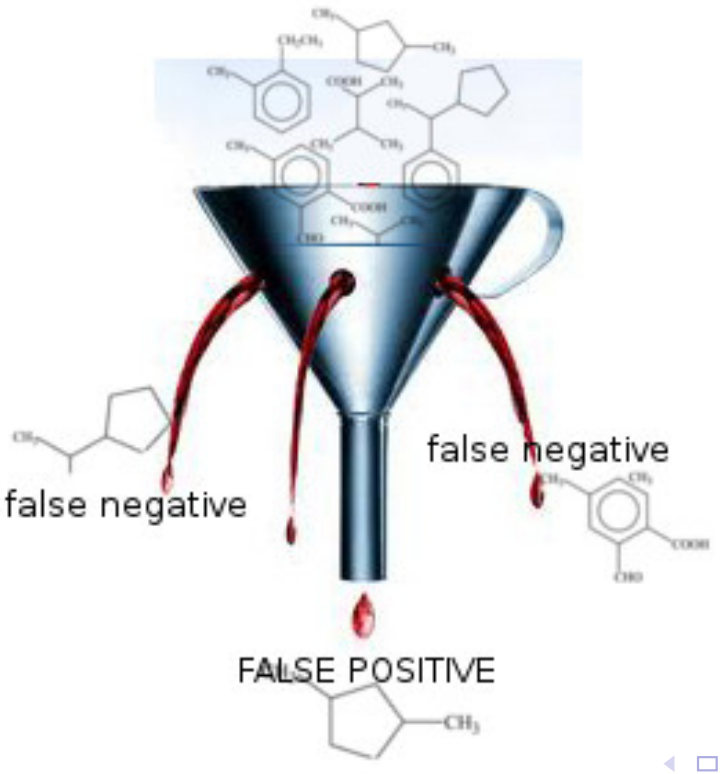
$$\Delta G = < \Delta E >_{\mathbf{x}} - T < \Delta S >_{\mathbf{x}}$$

$< .. >$ = Average over the ensemble

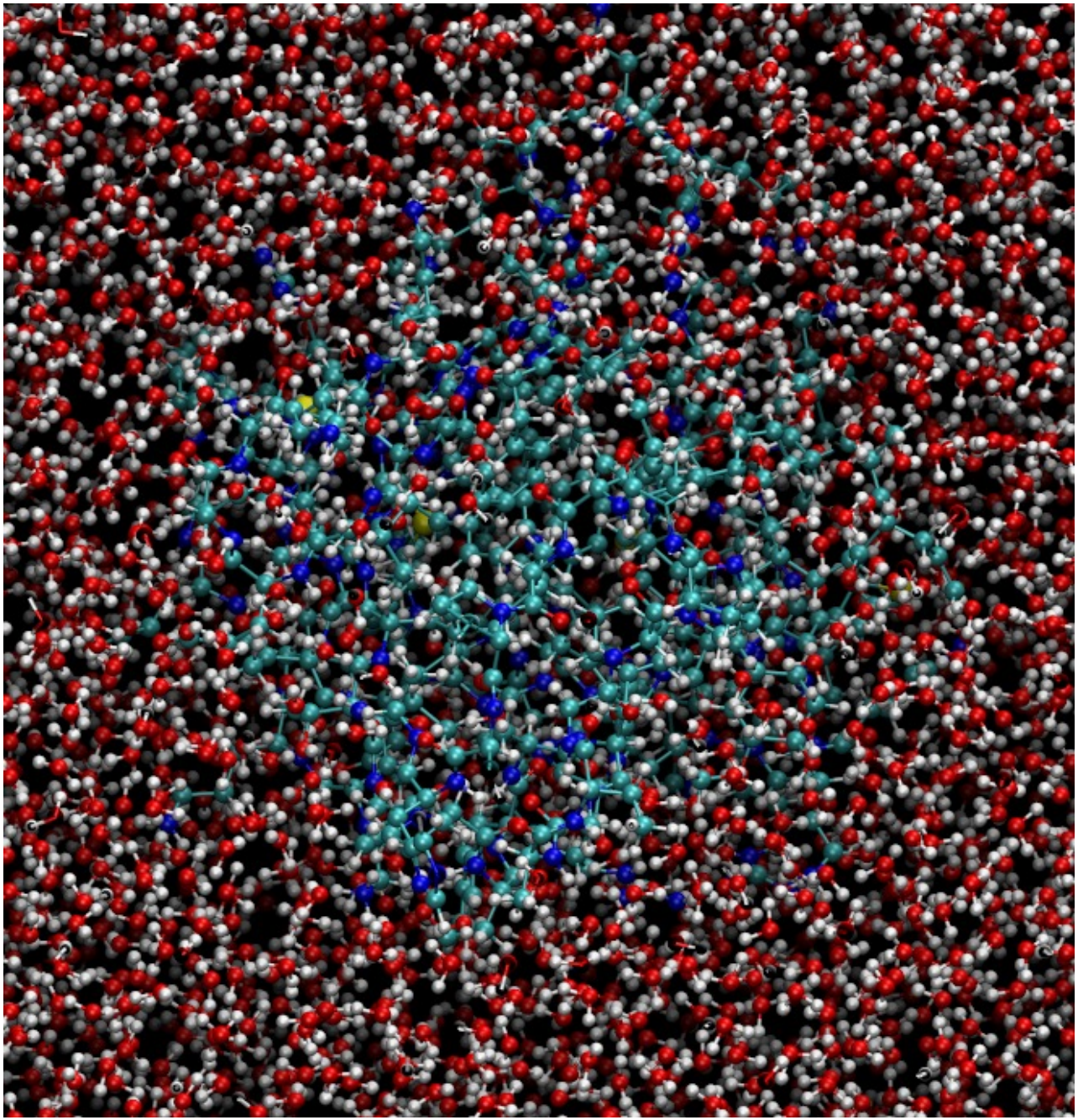
$\mathbf{x}$ = solvent and drug receptor coordinates

HT**V**S, Molecular Docking : $\Delta G \simeq \Delta \mathbf{E}$

Reality



Atomistic approach (MD)



Atomistic approach (MD)

- All particles (solute+solvent) in the MD box are considered.
- PBC are used to eliminate surface effects
- Particles interact via the FF parameterization containing a valence part and all atom-atom 2body interactions.

$$E_{pair} = \sum_{bonds} K_r (r - r_{eq})^2 + \sum_{angles} K_\theta (\theta - \theta_{eq})^2 +$$
$$\sum_{dihedrals} \frac{V_n}{2} [1 + \cos(n\phi - \gamma)] + \sum_{i<j} \left[\frac{A_{ij}}{R_{ij}^{12}} - \frac{B_{ij}}{R_{ij}^{6}} + \frac{q_i q_j}{\varepsilon R_{ij}} \right]$$

- Canonical sampling is obtained integrating a modification of Newtonian EoM

Atomistic approach: Free energy computation)

- **RAR**: Reversible Alchemical route (Jorgensen/Gilson) (free energy perturbation/thermodynamic integration)
- **HREM**: BEDAM or EDU-HREM
- **PMF**: Potential of mean force route [TI or Metadynamics]
- **PMF**: Potential of mean force route [JT, NONEQ]
- **FS-DAM**: Fast switching Double annihilation method. [JT, NONEQ]

All atomistic approaches are based on weakly communicating multiple MD trajectories (GE or JT ensemble). Modern MD algorithms for ΔG calculations are suited for double layer OpenMP/MPI parallelization on HPC infrastructures.

Task	Layer	comm.
GE or JT trajectories	MPI/distributed memory	InterNode
Forces evaluation per traj.	OpenMP/shared memory	IntraNode

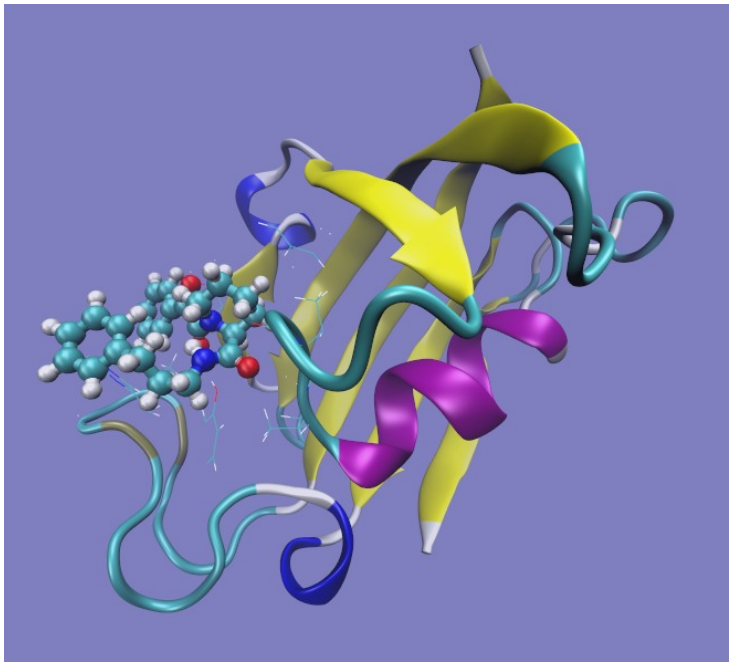
Atomistic approach (MD): the *alchemical* approach

"all that is needed is two simulations, one in which [the ligand] disappears by itself and one in which it disappears from the complex" *(W. Jorgensen, Acc.Chem.Res., more than 25 years ago).*

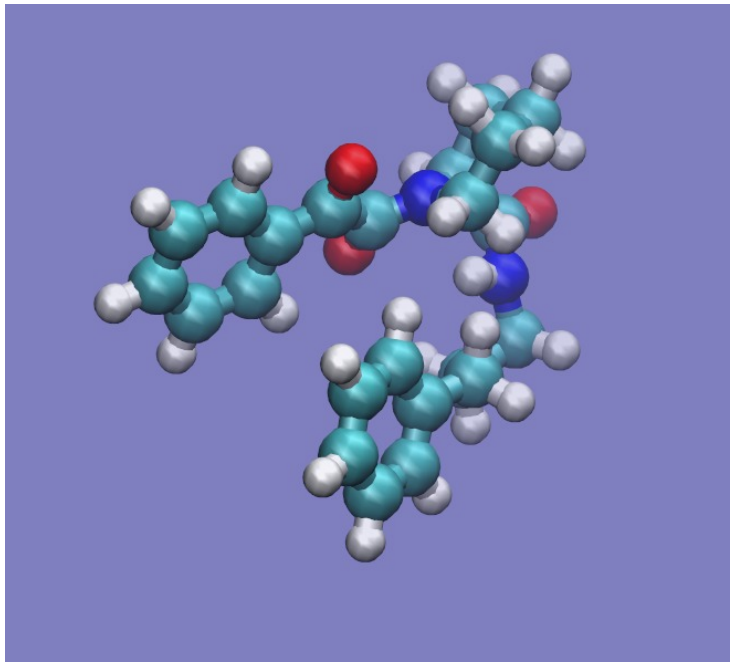
Atomistic approach (MD): the *alchemical* approach

"all that is needed is two simulations, one in which [the ligand] disappears by itself and one in which it disappears from the complex" *(W. Jorgensen, Acc.Chem.Res., more than 25 years ago).*

Process 1



Process 2

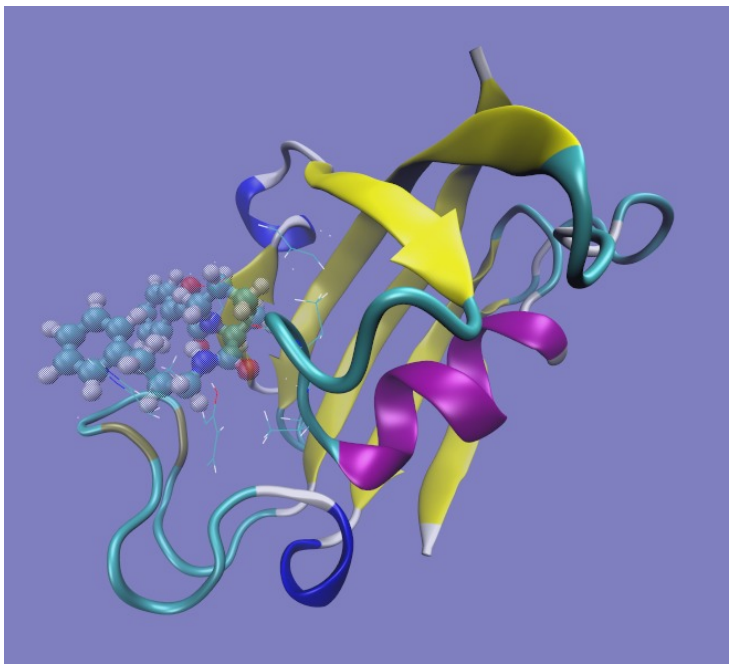


$$\lambda = 1$$

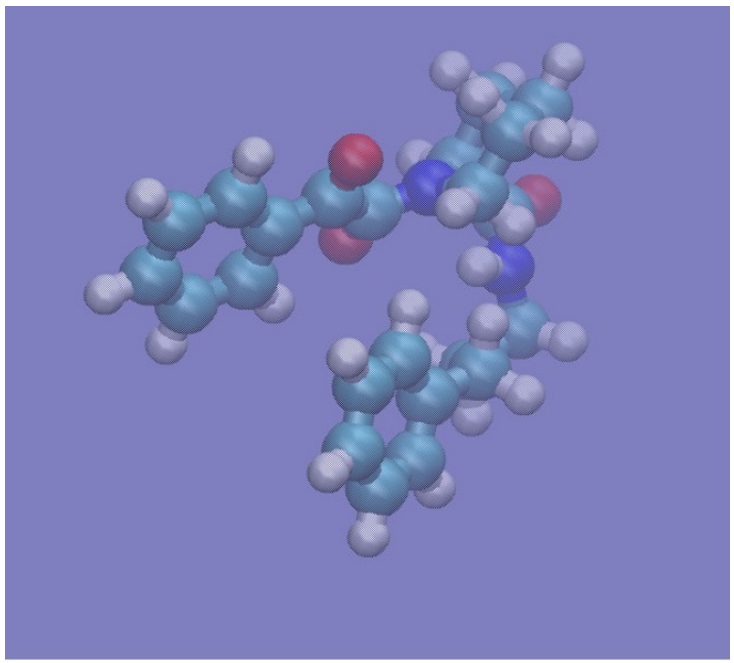
Atomistic MD: the *alchemical* approach

"all that is needed is two simulations, one in which [the ligand] disappears by itself and one in which it disappears from the complex" *(W. Jorgensen, Acc.Chem.Res., more than 25 years ago).*

Process 1



Process 2

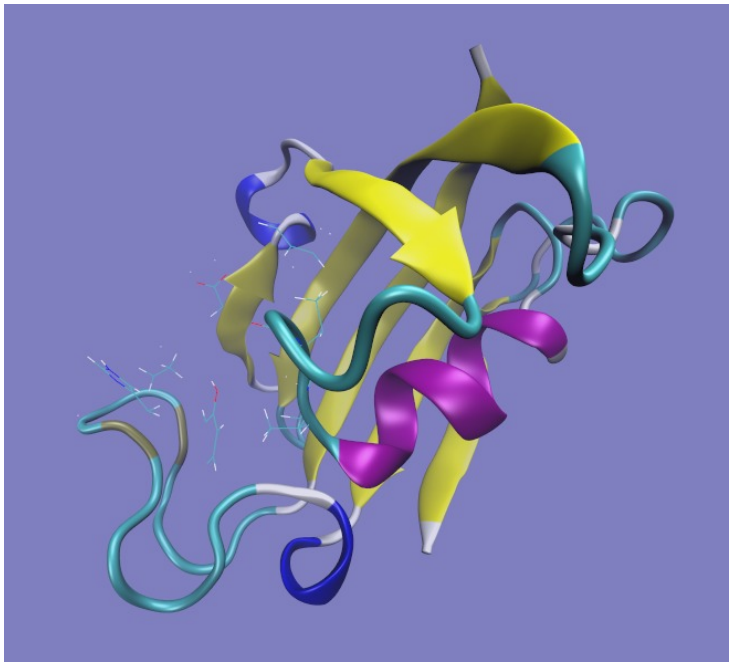


$$0 < \lambda < 1$$

Atomistic MD: the *Equilibrium* alchemical approach

"all that is needed is two simulations, one in which [the ligand] disappears by itself and one in which it disappears from the complex" *(W. Jorgensen, Acc.Chem.Res., more than 25 years ago).*

Process 1 **Process 2**



$\lambda = 0$

Equilibrium alchemical approach: HOWTO

Apply Jorgensen's statement:

$$\begin{aligned} 1) \quad \Delta G_1 &: \quad (\mathrm{RL})_{\mathrm{sol}} \rightarrow \mathrm{R}_{\mathrm{sol}} + \mathrm{L}_{\mathrm{gas}} \\ 2) \quad \Delta G_2 &: \quad \mathrm{L}_{\mathrm{sol}} \rightarrow \mathrm{L}_{\mathrm{gas}} \\ \Delta G_d = \Delta G_1 - \Delta G_2 &: \quad (\mathrm{RL})_{\mathrm{sol}} \rightarrow \mathrm{R}_{\mathrm{sol}} + \mathrm{L}_{\mathrm{sol}} \end{aligned}$$

Use thermodynamic integration to evaluate $\Delta G_1, \Delta G2$

$$W_{\mathrm{rev}} = \Delta G_{1,2}(\mathrm{sim.}) = \int_0^1 \left\langle \left(\frac{\partial \mathrm{H}_{1,2}(\lambda)}{\partial \lambda} \right) \right\rangle_\lambda \mathrm{d}\lambda$$

Implement *alchemical decoupling* by discretizing the Hamiltonian such that: $\lambda = 1$ Ligand on. $\lambda = 0$ Ligand off. $0 < \lambda < 1$ Ligand neither fully on neither fully off. Several atomistic simulations (explicit solvent, PBC) are run for each λ_i in [0,1] and the *reversible* work of the whole processes is simply obtained by appropriately summing up the contributions from the λ_i *equilibrium* simulations.

Screening on HPCplatforms

CRESCO4- 4864 Cores E5-2670 2.6 GHz 64

- **Scoring/Docking techniques**: massively parallel (each pose/ligand can be done independently) but unreliable
- **Stat-Mech based MD/atomistic approach**: reliable but Long simulations needed (10-50 ns) for each λ point/replica. Numerical integration of the EoM is a strong scaling method. FDD on infiniband: 100 atoms per domain before saturation (few hundreds of cores for typical target).
- Quest for *low communication parallel algorithms based on stat-mech* for reliable in silico screening

Non Equilibrium *fast* alchemical approach: Jarzynski

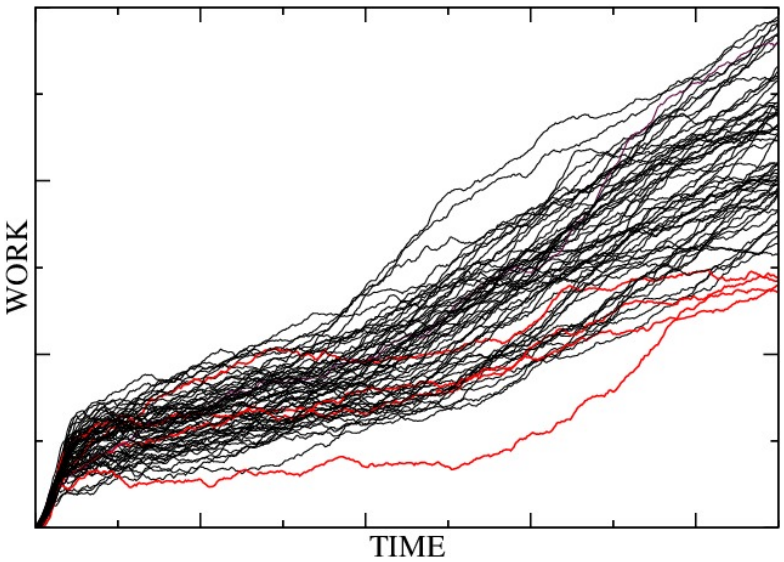
Jarzynski theorem: Phys. Rev. Lett. 1997 $e^{-\beta \Delta G} = \langle e^{-\beta W} \rangle$

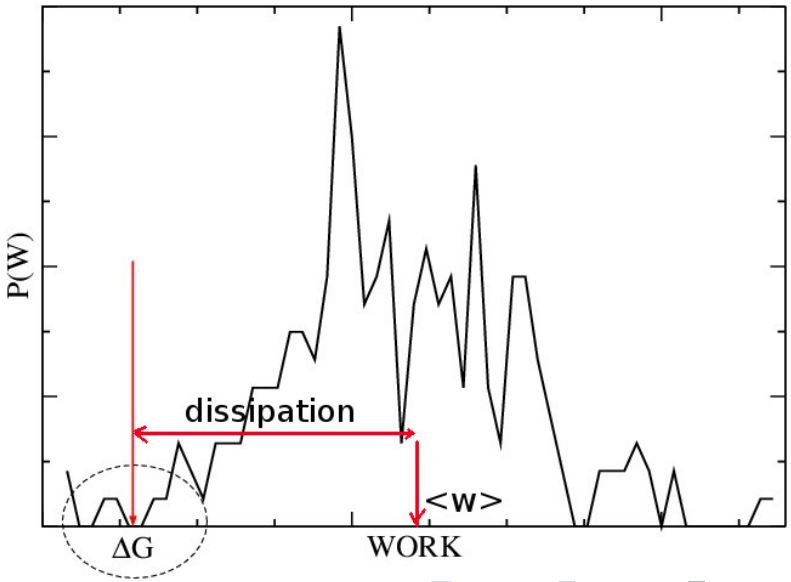
- prepare Equilibrium states in the coupled state $\lambda = 1$
- Launch N *fast* trajectories in parallel driving the system to the $\lambda = 0$ decoupled state.
- compute the histogram P(W) and evaluate ΔG via JT.
- Problem!: Exponential average depends on sampling in the tails of distributions $\rightarrow \Delta G$ biased and noisy.

Non Equilibrium *fast* alchemical approach: Jarzynski

Jarzynski theorem: Phys. Rev. Lett. 1997 $e^{-\beta \Delta G} = \langle e^{-\beta W} \rangle$

- prepare Equilibrium states in the coupled state $\lambda = 1$
- Launch N *fast* trajectories in parallel driving the system to the $\lambda = 0$ decoupled state.
- compute the histogram P(W) and evaluate ΔG via JT.
- Problem!: Exponential average depends on sampling in the tails of distributions $\rightarrow \Delta G$ biased and noisy.

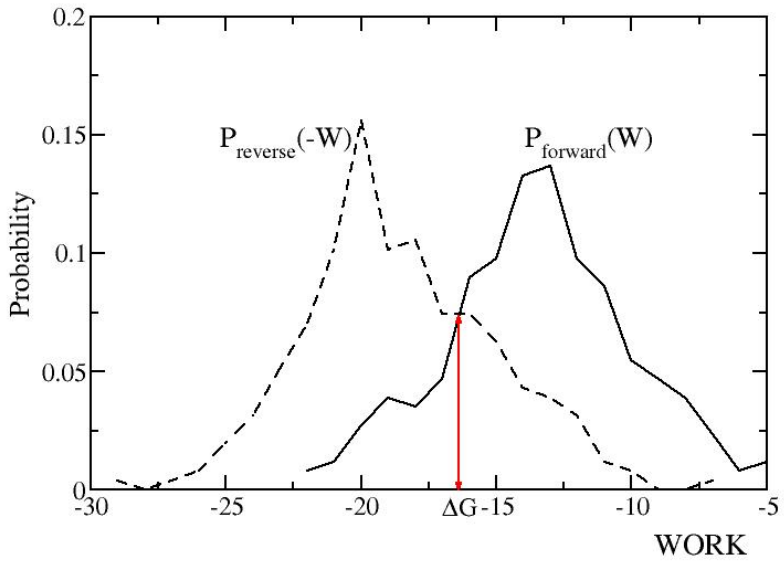




Non Equilibrium *fast* alchemical approach: Crooks

Crooks theorem: J. Stat Mech. 1998

$$\frac{P_{AB}(W)}{P_{BA}(-W)} = e^{\beta(W-\Delta G)}$$



Bennett ML formula

$$\sum_{i=1}^{n_F} \frac{1}{1+\frac{n_F}{n_R}\,\mathrm{e}^{\beta(W_i(F)-\Delta F)}} - \sum_{i=1}^{n_R} \frac{1}{1+\frac{n_R}{n_F}\,\mathrm{e}^{\beta(W_i(R)+\Delta F)}} = 0$$

- Very accurate. The creation process must be done with inverted time schedule.
- Both fully coupled $\lambda = 1$ and fully decoupled starting configurations $\lambda = 0$ must be prepared.
- The decoupled state for process 1 must be prepared with the ligand in the binding site.
- In bidirectional Crooks approach a restraint MUST be introduced .

Non Equilibrium *fast* alchemical approach: Gauss

Is there a way to use the NE *unidirectional* approach? Is a FS-DAM feasible?

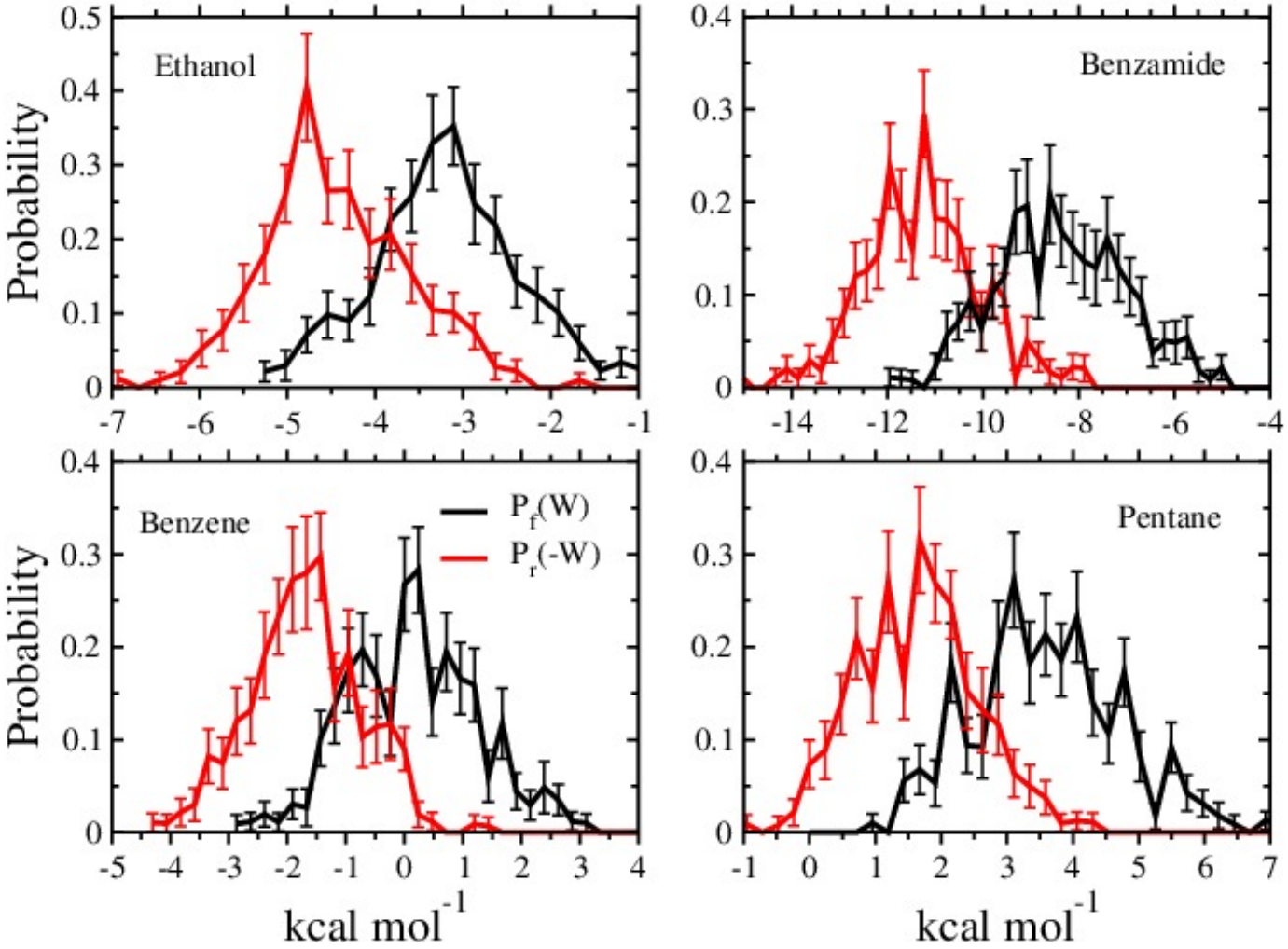
Non Equilibrium *fast* alchemical approach: Gauss

Is there a way to use the NE *unidirectional* approach? Is a FS-DAM feasible?
yes, provided that the distribution $P(W)$ for the decoupling process is GAUSSIAN. If so, then

- The decoupling $P(W)$ and the inverted re-coupling distribution $P(-W)$ should be symmetric with respect to ΔG
- The unidirectional JT reduces to (exactly) $\Delta G = \langle W \rangle - \frac{\beta\sigma^2}{2}$
- The decoupling from the bound states $(\mathrm{RL})_{\mathrm{sol}}$ can be done the no restraint.
- The Gaussian JT estimate is *unbiased* and accurate

Non Equilibrium *fast* alchemical approach: Gauss

Fast switching decoupling and re-coupling of ligand-sized molecules in water (process 2 can be inverted)



C. Cardelli, P. Procacci, J. Chem Theory Comput. 2014, 10, 2813

Non Equilibrium *fast* alchemical approach: Gauss

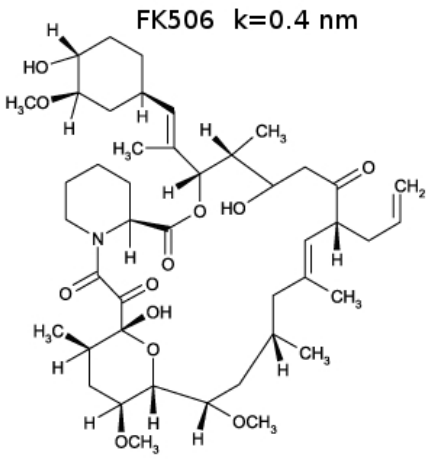
Fast switching decoupling and re-coupling of ligand-sized molecules in water (process 2 can be inverted)

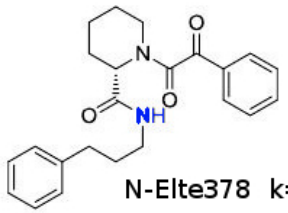
	JT Gauss(decoup.))	Crooks	JT Gauss (coupl.)
Benzene	0.87 $\pm$0.14	**0.79** $\pm$0.04	-0.91 $\pm$0.17
Benzamide	9.95 $\pm$0.24	**9.78** $\pm$0.07	-10.02 $\pm$0.22
Ethanol	3.80 $\pm$0.04	**3.80** $\pm$0.04	-3.84 $\pm$0.10
Pentane	-2.52 $\pm$0.08	**-2.56** $\pm$0.05	2.37 $\pm$0.05

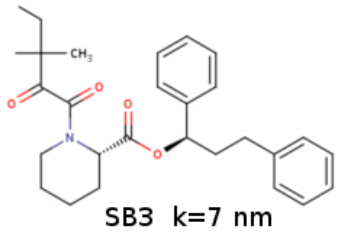
Decoupling free energies computed using unidirectional JT Gauss (decoupling), Crooks and JT Gauss (re-coupling). Errors have been evaluated using BS on 128 trajectories (Cardelli and Procacci JCTC 2014)

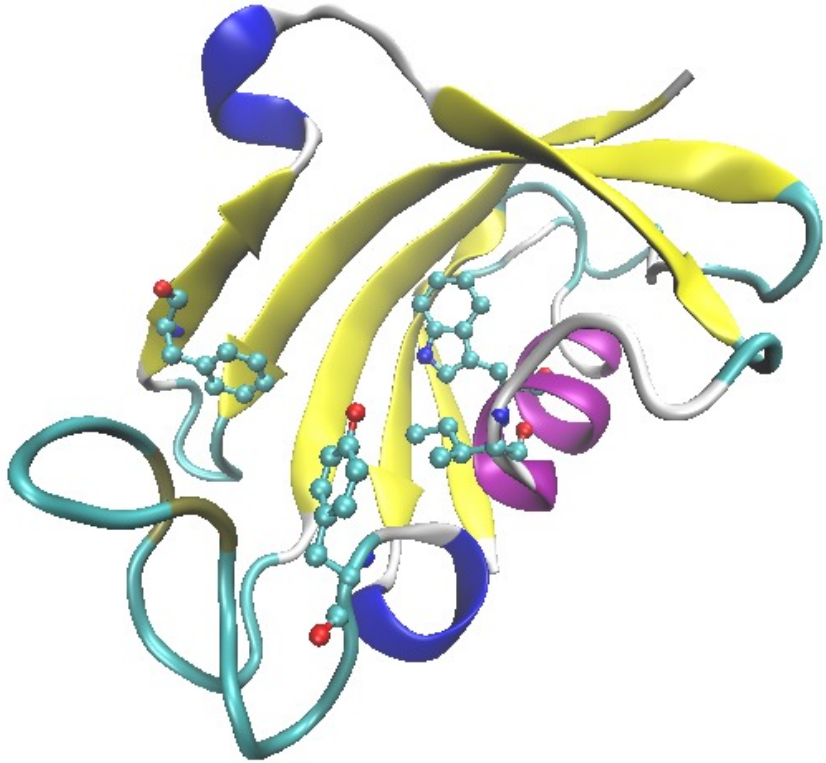
Results: FKBP12 ligands

FKBP12 ligands: 3 x (256 x 2) simulations. 180 ps (3 h on CRESCO). T=300 K, P= 1 atm, 4000 TIP3 waters in the bound state, AMBER03, PME, 5-MTS (12 fs).



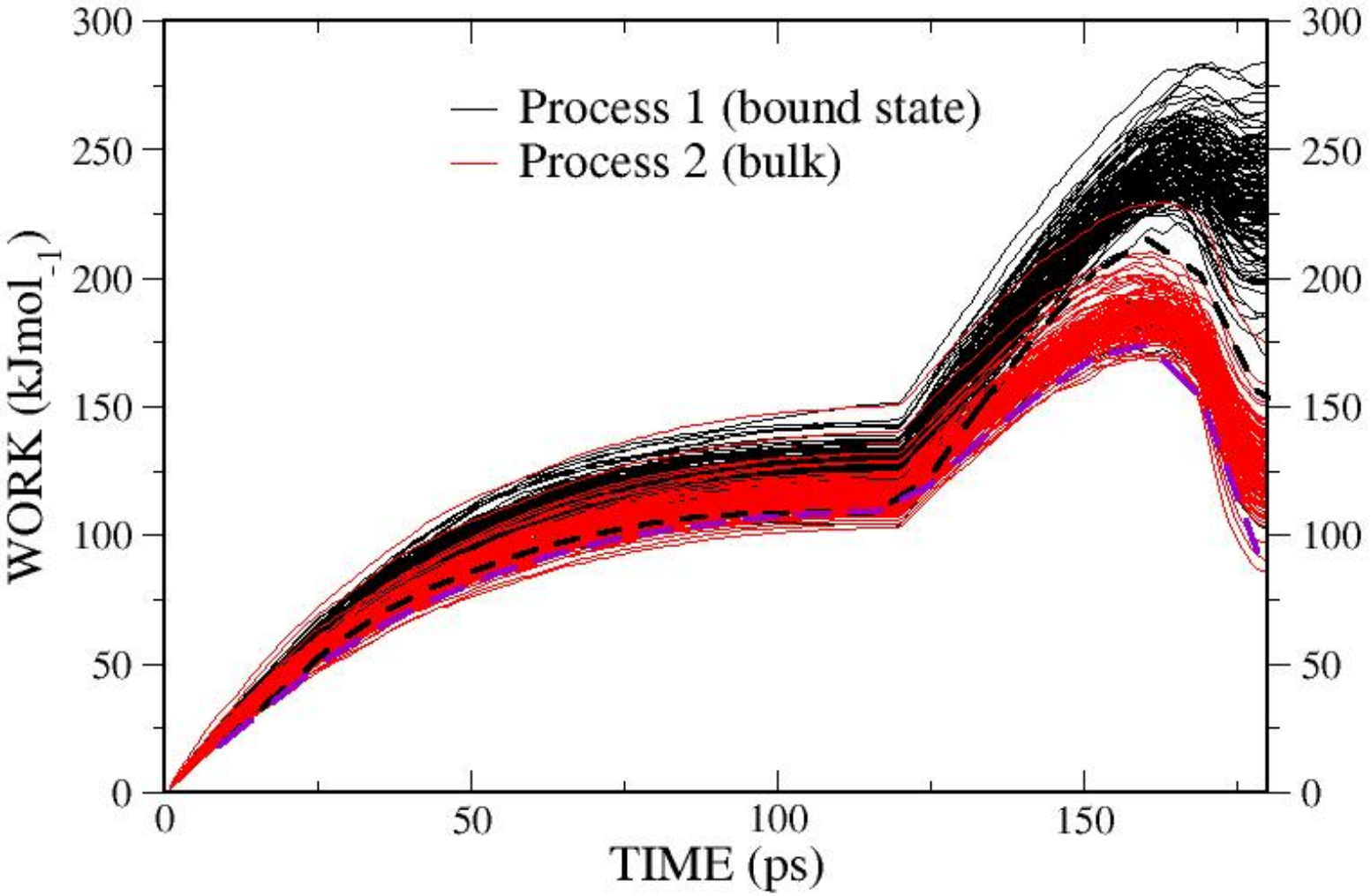






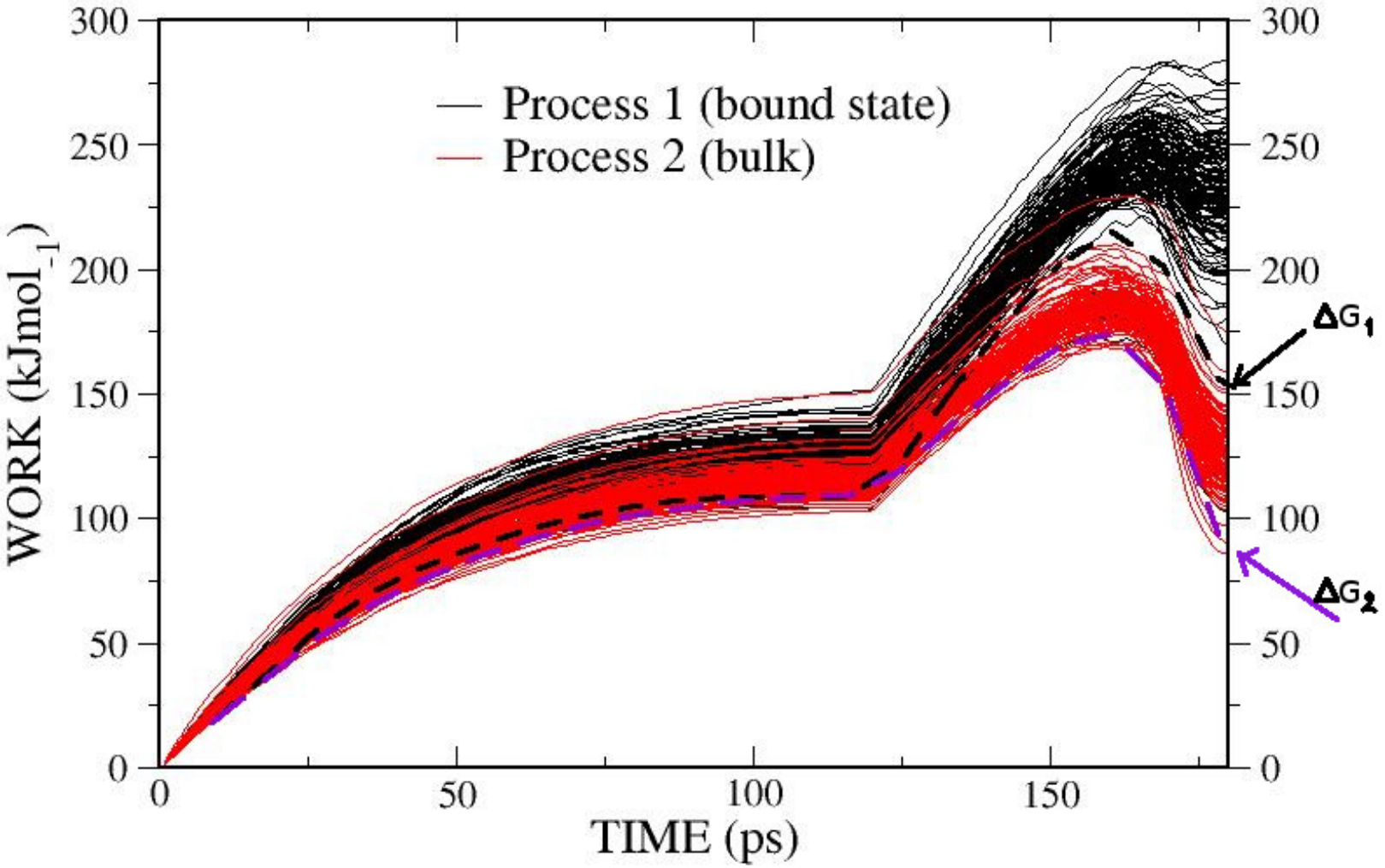
Results: FKBP12 ligands

FK506



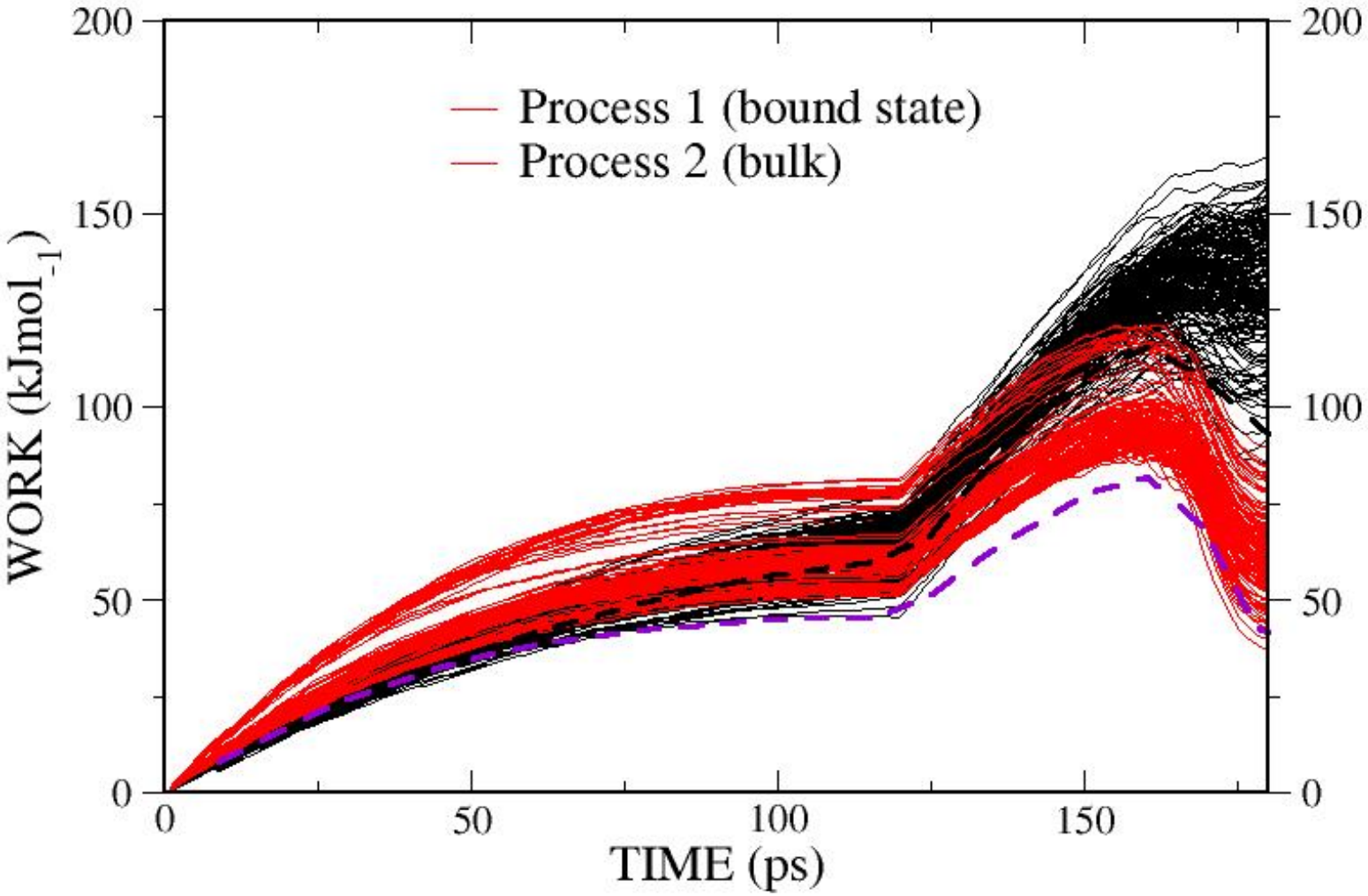
Results: FKBP12 ligands

FK506



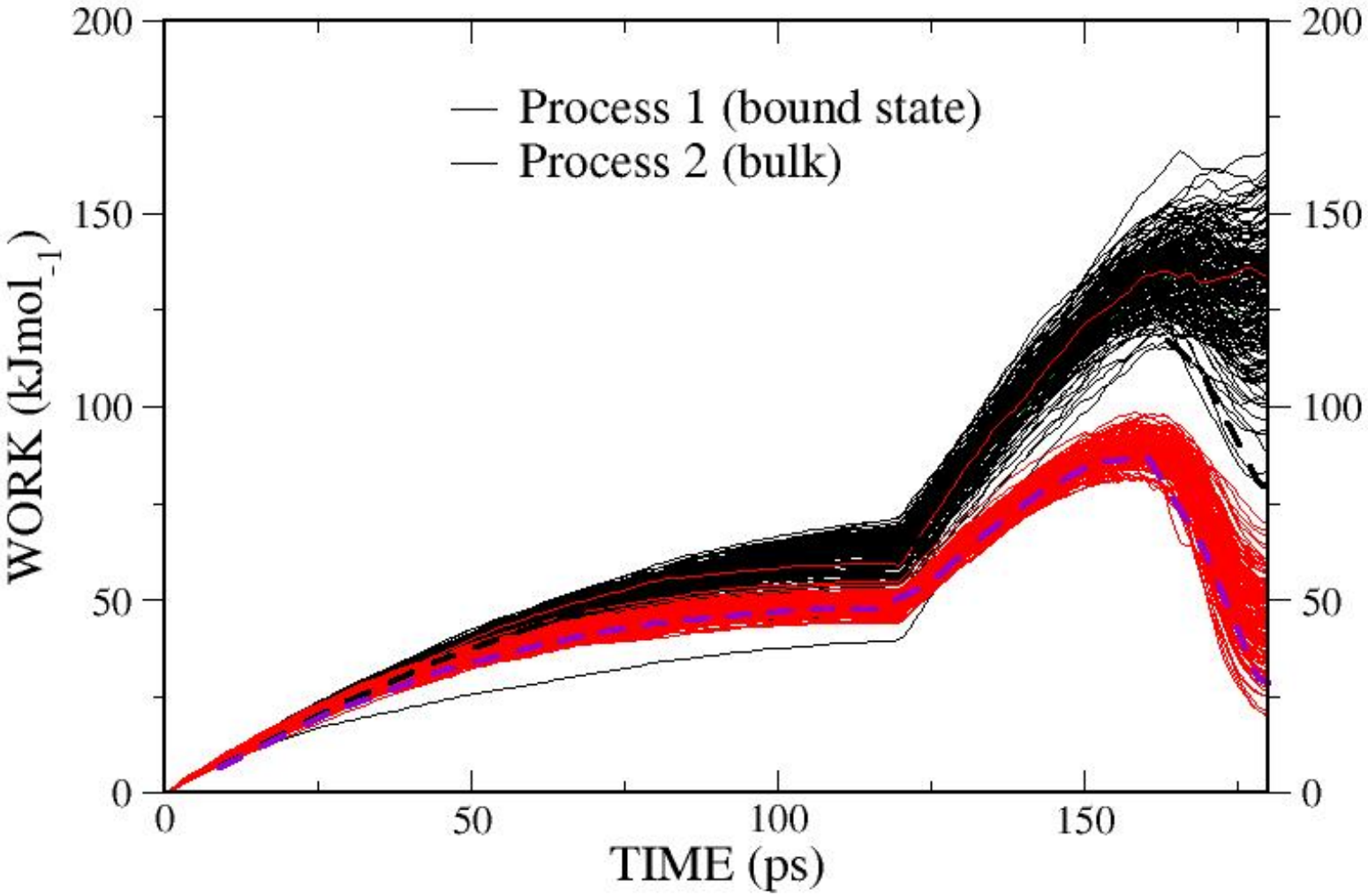
Results: FKBP12 ligands

N-Elte378 (Martina 2013, J. Med. Chem.)



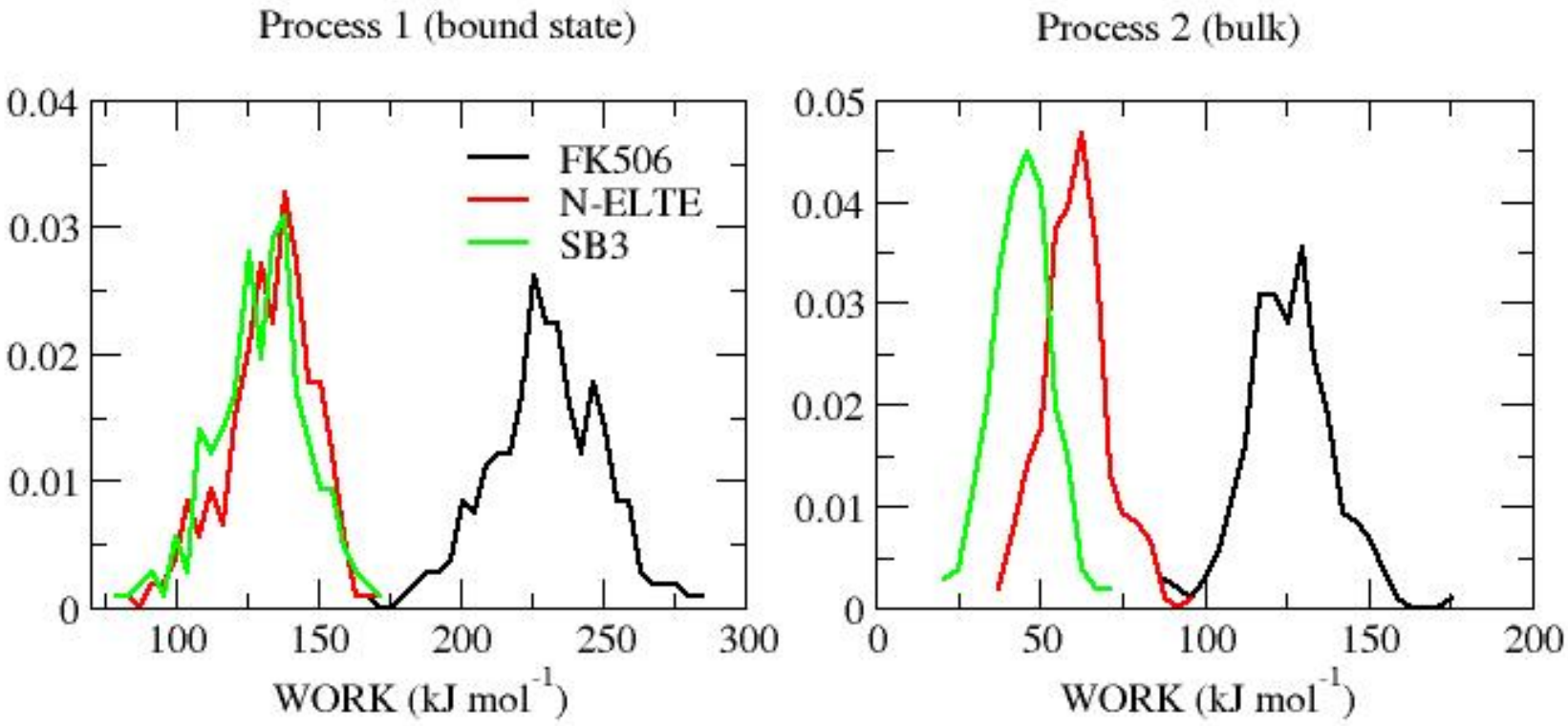
Results: FKBP12 ligands

SB3 (Holt 1994, Jacs)

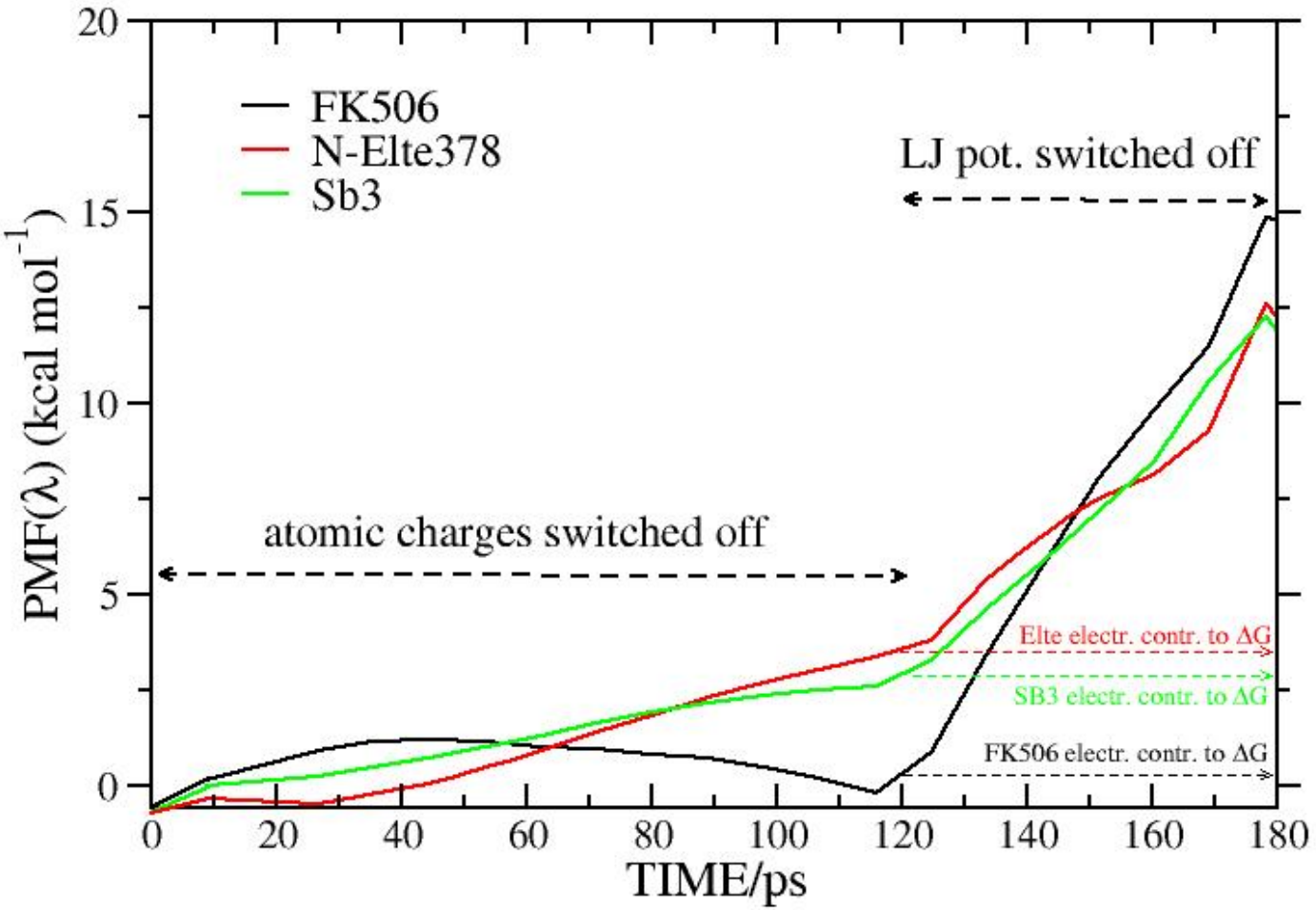


Results: FKBP12 ligands

	$< W >$ (kJ/mol)	$\sigma(kJ/mol)$	γ	η
c-FK506	54.9 ± 0.2	22.3 ± 1.2	-0.10 ± 0.09	0.08 ± 0.18
c-NELTE	31.9 ± 0.2	13.0 ± 0.8	-0.46 ± 0.09	0.02 ± 0.23
c-SB3	31.1 ± 0.2	14.3 ± 0.9	-0.24 ± 0.13	0.11 ± 0.20
s-NELTE	14.5 ± 0.1	5.3 ± 0.4	0.38 ± 0.12	0.54 ± 0.33
s-FK506	30.0 ± 0.1	9.5 ± 0.9	-0.01 ± 0.21	0.75 ± 0.29
s-SB3	10.6 ± 0.1	4.6 ± 0.2	-0.01 ± 0.13	0.19 ± 0.16



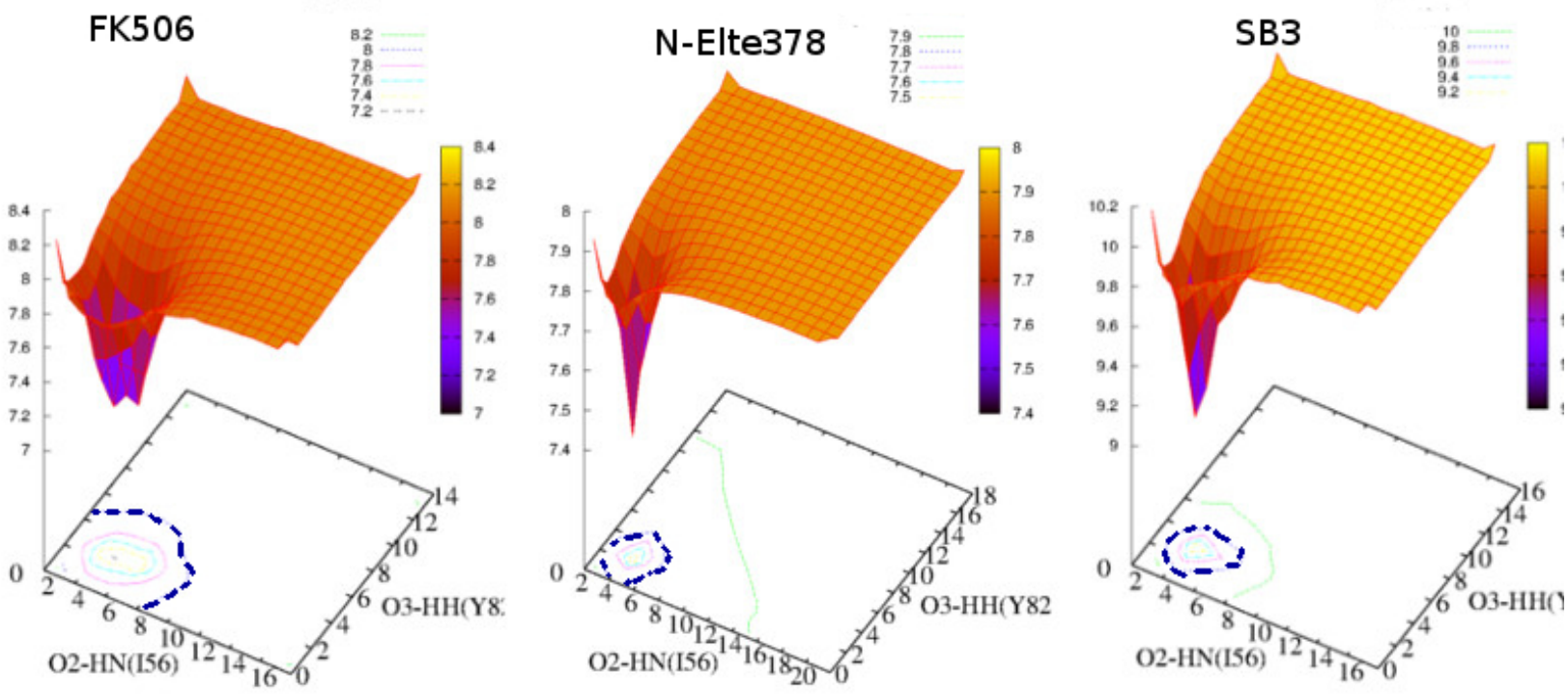
Results: FKBP12 ligands



Results: FKBP12 ligands

Volume corrections

Ligand	Volume (Å^3)	SSC (kcal mol^{-1})
FK506	512	-0.70
N-Elte378	343	-0.90
Sb3	216	-1.22



Results: FKBP12 ligands

	ΔG_1	ΔG_2	SSC	ΔG	Exp.
FK506	36.6±1.4	21.8±0.8	-0.7	*14.1* ±1.5	**13.0** ±0.8
N-Elte378	22.3±1.0	10.0±0.4	-0.9	*11.4* ±1.1	**11.8** ±0.7
Sb3	18.8±1.0	6.8±0.3	-1.2	*10.8* ±1.1	**11.0** ±0.5

Table: Standard dissociation free energies (kcal mol^{-1})

FS-DAM Double Layer parallelization: $N_{cores} = N_T \times N_t$

```
bsub -n N_cores -o lsf1.out -e lsf1.err -q cresco_32h24 ./script.sh

#!/bin/ksh
exe=/afs/enea.it/por/user/procacci/ORAC/src/Intel_MPI_OMP/orac_Linux HOSTFILE=$LSB_DJOB_HOSTFILE
mpirun --mca pls_rsh_agent "blaunch.sh" -n N_T --hostfile $HOSTFILE $exe < orac.input
```

```
program schema
  implicit none
  include 'mpif.h'     ! MPI layer
  include 'omp_lib.h'  ! OMP layer
  ....
  double precision, allocatable :: f(:),ft(:,:)  ! force arrays
!---------------------- initialize MPI layer
 CALL MPI_Init(ierr)
 CALL MPI_Comm_size(MPI_COMM_WORLD, NT, ierr)
 CALL MPI_Comm_rank(MPI_COMM_WORLD, IT, ierr)
...
! start MD loop
 do n=1,nsteps  ! START MD LOOP
!$OMP PARALLEL SHARED(f,rij,natoms,nrep) PRIVATE (iomp,itask,i,j,iun) REDUCTION(+ :FT)
    iomp=OMP_GET_THREAD_NUM()
    nforce=OMP_GET_NUM_THREADS()
    itask=1+iomp
    ! this mimicks orac force loup (work is balanced among nforce OMP threads using *nforce* as increment)
    do i=itask,natoms-1,nforce
       do j=i+1,natoms
          rij=... ! do calculation in double loop (only upper matrix
          .....
          f(i)= f(i) + rij
          ft(j)= ft(j)+ rij
       end do
    end do
!$OMP END PARALLEL
    ...
 END DO ! step loop ended
 ....
 CALL MPI_BARRIER(MPI_COMM_WORLD,ierr)
 call MPI_FINALIZE(ierr)
 STOP
end program schema
```

FS-DAM Double Layer parallelization: $N_{cores} = N_T \times N_t$

```
bsub -n N_cores -o lsf1.out -e lsf1.err -q cresco_32h24 ./script.sh

#!/bin/ksh
exe=/afs/enea.it/por/user/procacci/ORAC/src/Intel_MPI_OMP/orac_Linux HOSTFILE=$LSB_DJOB_HOSTFILE
mpirun --mca pls_rsh_agent "blaunch.sh" -n N_T --hostfile $HOSTFILE $exe < orac.input
```

```
program schema
  implicit none
  include 'mpif.h'      ! MPI layer
  include 'omp_lib.h'   ! OMP layer
  ....
  double precision, allocatable :: f(:),ft(:,:)  ! force arrays
!---------------------- initialize MPI layer
 CALL MPI_Init(ierr)
 CALL MPI_Comm_size(MPI_COMM_WORLD, NT, ierr)
 CALL MPI_Comm_rank(MPI_COMM_WORLD, IT, ierr)
...
! start MD loop
 do n=1,nsteps  ! START MD LOOP
!$OMP PARALLEL SHARED(f,rij,natoms,nrep) PRIVATE (iomp,itask,i,j,iun) REDUCTION(+ :FT)
    iomp=OMP_GET_THREAD_NUM()
    nforce=OMP_GET_NUM_THREADS()
    itask=1+iomp
    ! this mimicks orac force loup (work is balanced among nforce OMP threads using *nforce* as increment)
    do i=itask,natoms-1,nforce
       do j=i+1,natoms
          rij=... ! do calculation in double loop (only upper matrix
          .....
          f(i)= f(i) + rij
          ft(j)= ft(j)+ rij
       end do
    end do
!$OMP END PARALLEL
    ...
 END DO ! step loop ended
 ....
 CALL MPI_BARRIER(MPI_COMM_WORLD,ierr)
 call MPI_FINALIZE(ierr)
 STOP
end program schema
```

FS-DAM Double Layer parallelization: $N_{cores} = N_T \times N_t$

```
bsub -n N_cores -o lsf1.out -e lsf1.err -q cresco_32h24 ./script.sh

#!/bin/ksh
exe=/afs/enea.it/por/user/procacci/ORAC/src/Intel_MPI_OMP/orac_Linux HOSTFILE=$LSB_DJOB_HOSTFILE
mpirun --mca pls_rsh_agent "blaunch.sh" -n N_T --hostfile $HOSTFILE $exe < orac.input
```

```
program schema
  implicit none
  include 'mpif.h'     ! MPI layer
  include 'omp_lib.h'  ! OMP layer
  ....
  double precision, allocatable :: f(:),ft(:,:)  ! force arrays
!---------------------- initialize MPI layer
 CALL MPI_Init(ierr)
 CALL MPI_Comm_size(MPI_COMM_WORLD, NT, ierr)
 CALL MPI_Comm_rank(MPI_COMM_WORLD, IT, ierr)
...
! start MD loop
 do n=1,nsteps  ! START MD LOOP
!$OMP PARALLEL SHARED(f,rij,natoms,nrep) PRIVATE (iomp,itask,i,j,iun) REDUCTION(+ :FT)
    iomp=OMP_GET_THREAD_NUM()
    nforce=OMP_GET_NUM_THREADS()
    itask=1+iomp
    ! this mimicks orac force loup (work is balanced among nforce OMP threads using *nforce* as increment)
    do i=itask,natoms-1,nforce
       do j=i+1,natoms
          rij=... ! do calculation in double loop (only upper matrix
          .....
          f(i)= f(i) + rij
          ft(j)= ft(j)+ rij
       end do
    end do
!$OMP END PARALLEL
    ...
 END DO ! step loop ended
 ....
 CALL MPI_BARRIER(MPI_COMM_WORLD,ierr)
 call MPI_FINALIZE(ierr)
 STOP
end program schema
```

FS-DAM economics:

For all algorithms (RAR, PMF, EDU-HREM, BEDAM, FS-DAM) CPU time *rightarrow* scales **linearly** with *system* size.
Simulation time (S.T) *rightarrow* scales **linearly** with *ligand* size.

Below data are reported for a typical job on a 20000 atoms ligand-receptor (ligand 40 atoms) system on 256 cores on the CRESCO2 system done with the ORAC code (0.4 ns/CPU day on single core Intel Nehalem E5530 2.4 GHz).

		OpenMP		MPI				
	s.t.	N_t	p_{eff}	N_T	p_{eff}	wall	P_{eff}	$\delta\Delta G$
RAR	10 ns	4	0.99	64	0.95	6.64 d	0.95	< 1
PMF	40 ns	16	0.95	16	0.95	6.92 d	0.90	3-4
BEDAM	40 ns	16	0.95	16	0.95	6.92 d	0.90	1-3
EDU	40 ns	16	0.95	16	0.95	6.92 d	0.90	1-3
$(\lambda = 1)$	5 ns	32	0.90*	8	0.95	0.45 d	0.85	
FS	0.3 ns	2	1.00	128	1.00	0.37 d	1.00	
FS-DAM						0.82 d	0.92	< 1

* FDD MPI layer with GROMACS

Conclusions

- FS-DAM reliable, cheap and *FAST* for drug discovery.
- FS-DAM could provides the "last stage" RELIABLE (stat mech based) computational tool for drug profiling and hit-to-lead optimization with integration on in silico platform such as the Plexus Discovery module of the OIDD of Eli Lilly.
- Stat mech based computational ADME: solubility, Membranes permeability or w/oct part coefficients
- References: 1) R. B. Sandberg *et al.*, J. Chem Theory Comput. 2015 10, 423-435. and 2) C. Cardelli, P. Procacci, J. Chem Theory Comput. 2014, 10, 2813
- OMP/MPI ORAC version (still in the alpha stage) is expected by the summer on CRESCO.
- Thank you for the attention