Experiments with the Parareal algorithm

A. Roy,G. Zérah

Département de Physique Théorique et Appliquée, Commissariat à l'Energie Atomique,

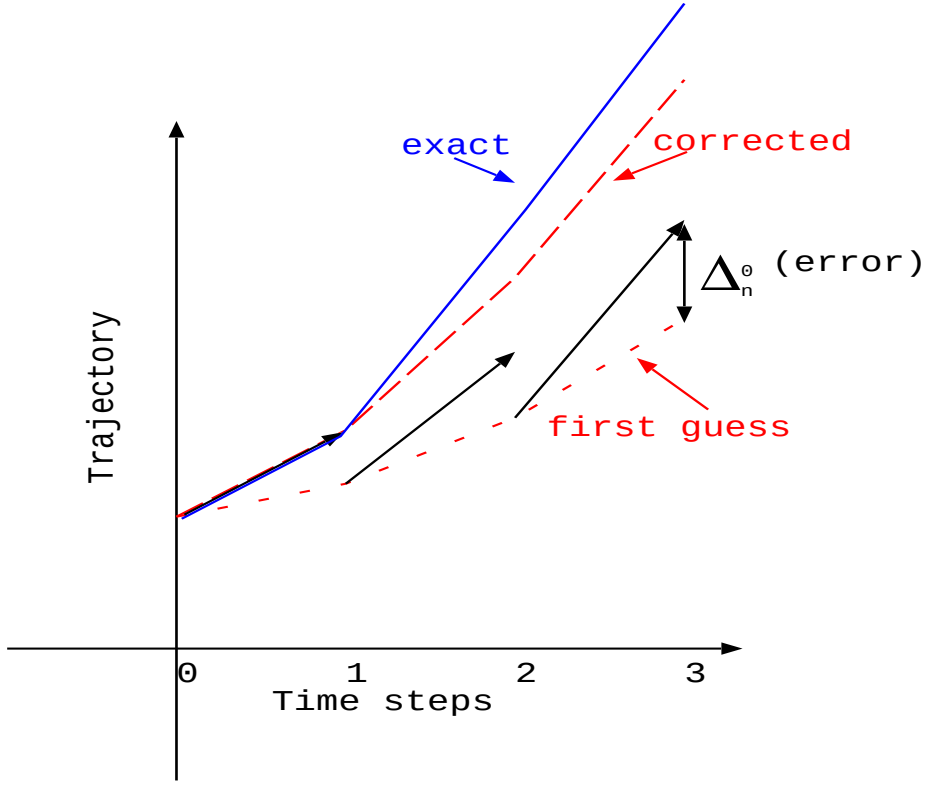
CEA-DAM Ile de France, B.P. 12, 91680 Bruyères le Châtel, France

Triying to use parallel machines for MI

- Rewrite the equation of motion symbolically as $\frac{du}{dt} = \Lambda(u)$, discretize it using a timestep ΔT.
- If u_n is our position in phase space after n steps, we introduce the propagator: $F_{\Delta T}(u_n) = u_{n+1}$
- $F_{\Delta T}$ describes merely the way we go from one configuration to the next.
- One could introduce, for any set v_n, $J = \sum_i (F_{\Delta T}(v_n) - v_{n+1})^2$, and try to minimize it in parallel.
- Here, we follow another route (parareel algorithm) more akin to a feedback mechanism.

The parareal algorithm (in graphics)

- Use two force fields as a predictor-corrector



The first step is exact

The parareal algorithm (in equations)

- Use two force fields as a predictor-corrector.
- First field:

$$\begin{aligned} u^0_{n+1} &= G_{\Delta T}(u^0_n) \\ u^0_0 &= u_0 \end{aligned}$$

- $G_{\Delta T}(u^0_n)$ is the coarse propagator.
- $F_{\Delta T}(u^0_n)$ is the fine propagator.
- The error is: $\Delta^0_n = F_{\Delta T}(u^0_n) - G_{\Delta T}(u^0_n)$
- Naturally, $F_{\Delta T}(u^0_n)$ can be computed in parallel

The feedback process(Lions et al 2001)

- Propagate a second time using the coarse propagator *corrected* by Δ_n^0

$$u_{n+1}^1 = G_{\Delta T}(u_n^1) + \Delta_{n+1}^0$$

- The process can be iterated, and this defines our successive trajectories, denoted by u_n^k

$$\begin{aligned} u_{n+1}^{k+1} &= G_{\Delta T}(u_n^{k+1}) + F_{\Delta T}(u_n^k) - G_{\Delta T}(u_n^k) \\ u_0^k &= u_0 \end{aligned}$$

- Expensive $F_{\Delta T}$ in // on N processors
- Efficient if $k_{\text{conv}} \ll N$ and $G_{\Delta T} \ll F_{\Delta T}$

A simple analysis

Why should it work?

- Case of a simple linear system
- Propagators are multiplication of u by F and G. The parareal formula reads now:

$$u_{n+1}^{k+1} = G_{\Delta T} u_n^{k+1} + (F_{\Delta T} - G_{\Delta T}) u_n^k$$

If $F_{\Delta T}$ and $G_{\Delta T}$ commute:

$$u_{n+1}^{k+1} = F_{\Delta T}^{n+1} u_0 - \sum_{p=k+1}^{n+1} \binom{n+1}{p} (F_{\Delta T} - G_{\Delta T})^p G_{\Delta T}^{n+1-p} u_0$$

Parallelisation

- We merely need to transfer coordinates and velocity to all processors
- A new MPI group is defined, which allows compatibility with k-point parallelisation
- A high level routine (pstate) has been written, which call gstate, changing the dtset parameters
- We need a routine to transform $\vec{r}_n, \vec{v}_n$ in $\vec{r}_{n+1}, \vec{v}_{n+1}$
- This requires, two steps in the Verlet algorithm (quite expensive)

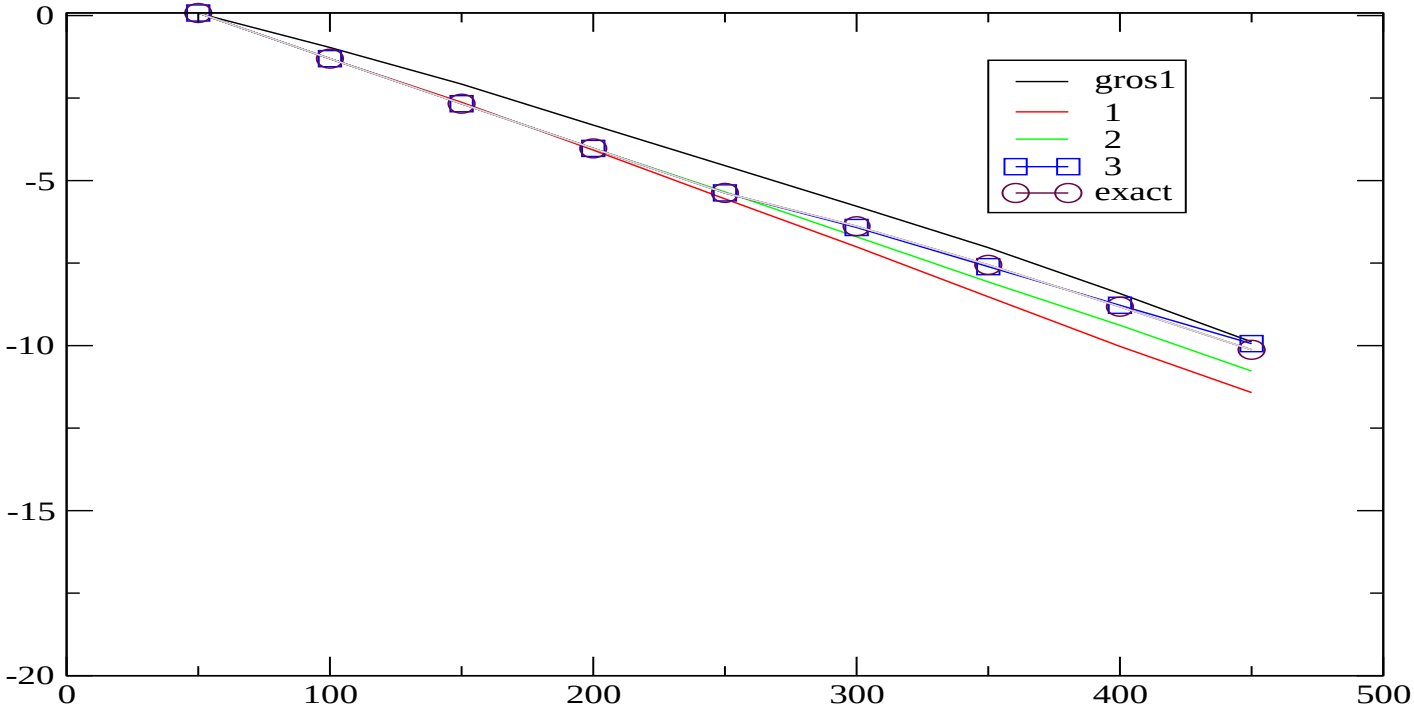
Coupling TF and ab initio

- Simulations of Helium
- Coarse integrator: Thomas Fermi (using a soft local potential)

$$C\rho(r)^{2/3} + v_h(\rho) + v_{xc}(\rho) + v_{ps}(r) = \mu$$

Results

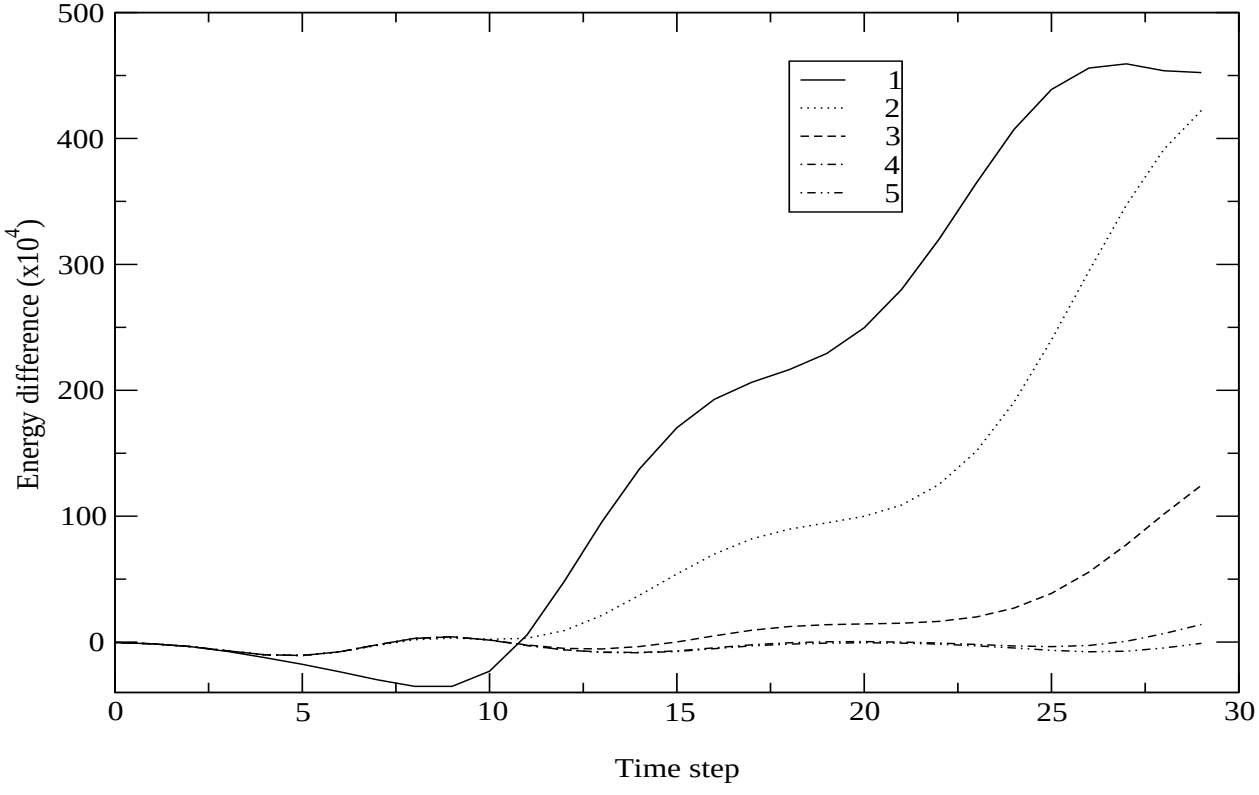
- Convergence of Trajectory



- For $k = 3$, we are converged
- The speedup is about 1.5 (9/3/2) on 9 processors. Still quite modest.

Results

- For comparison, in Al, using a classical potential as a predictor



- For $k = 4$, we are converged

Conclusion

- The use of Thomas-Fermi as a predictor has the advantage of not requiring a classical potential
- But the gains of the parareal method remain less important than using a classical potential as a predictor
- Still other methods needed to achieve higher speed in MD

THANK YOU!