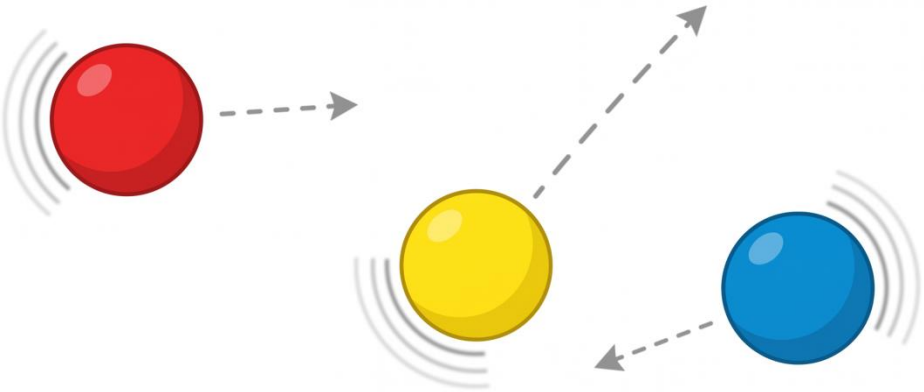
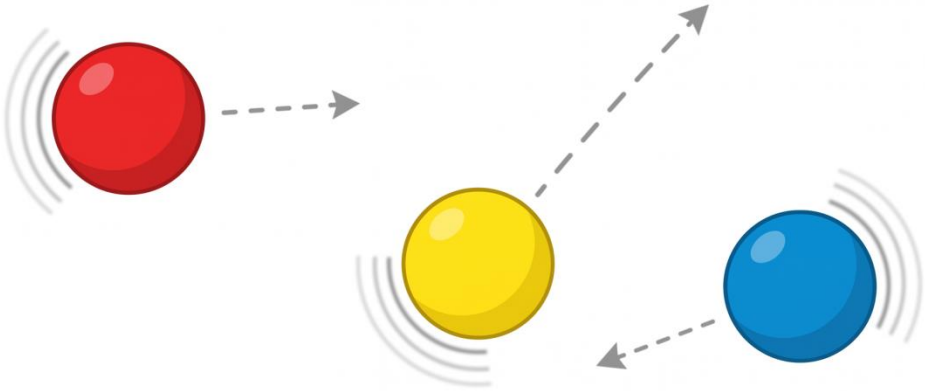
Beyond Verlet:
Unlocking a 2x Speedup in Molecular Dynamics with Novel Time Integrators



Martin Brehm* and Christian Dreßler

* Paderborn University

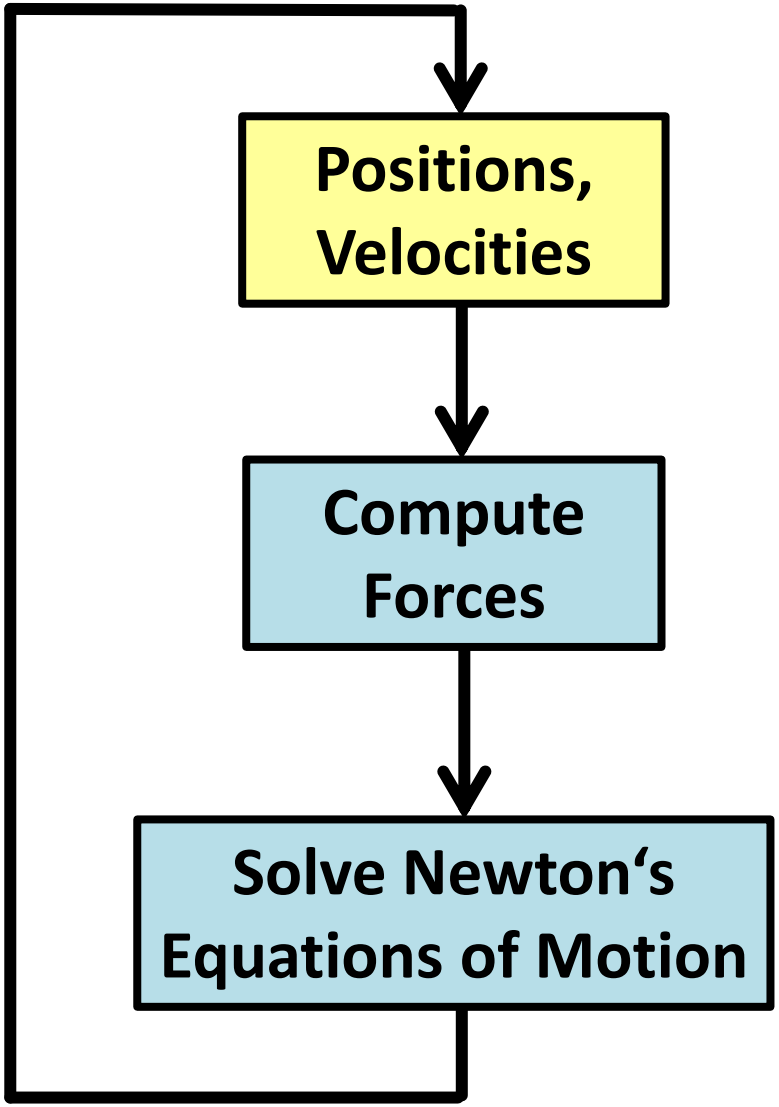
https://brehm-research.de/integrator



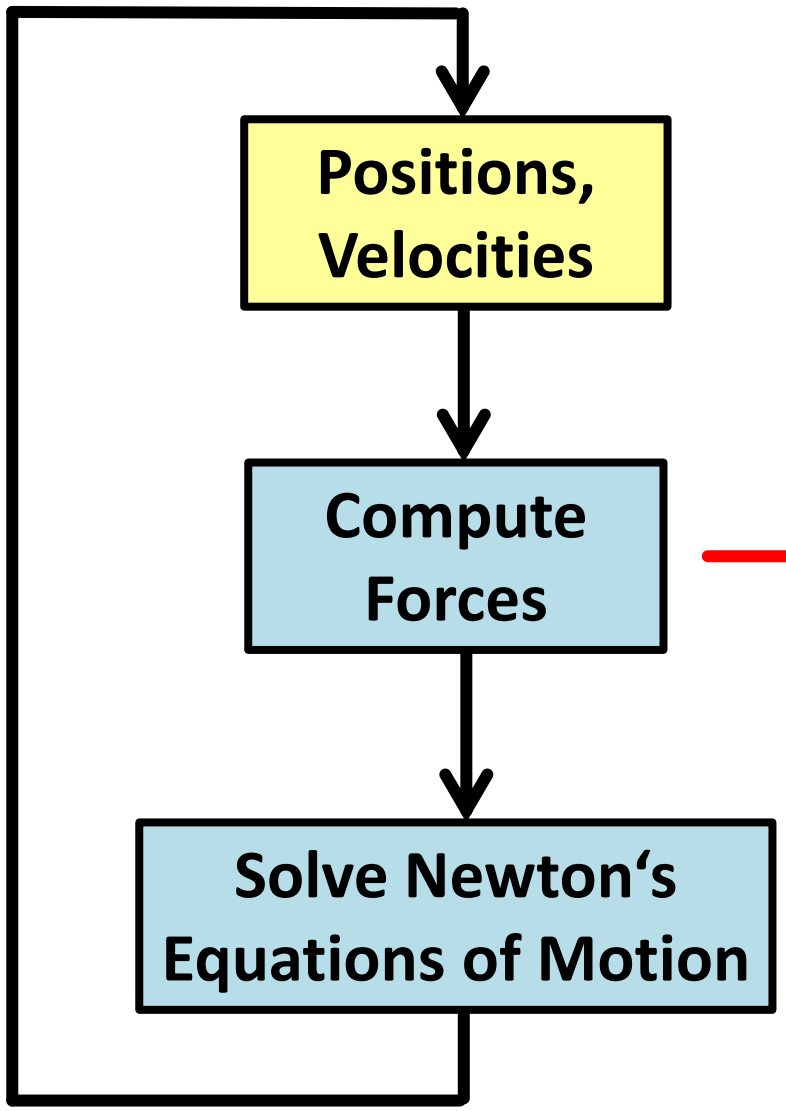
Slides have been uploaded for public access:

https://brehm-research.de/integrator

Molecular Dynamics



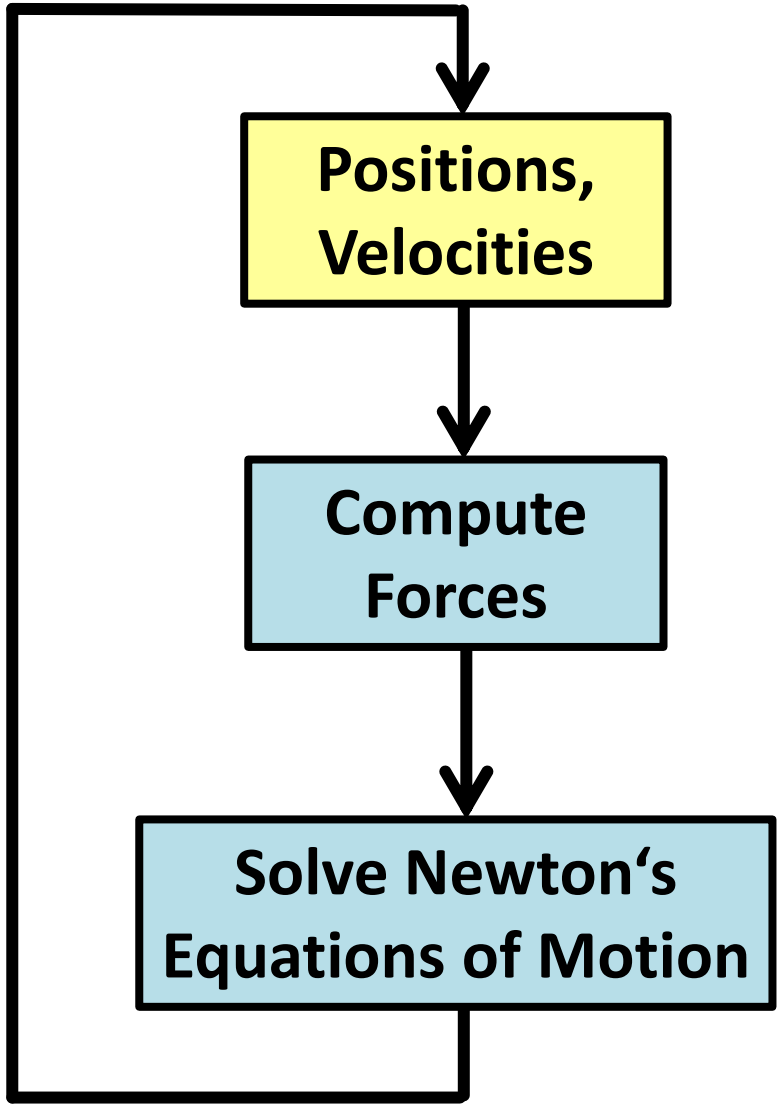
Molecular Dynamics



Has evolved a lot:

- Force field MD
- Polarizable force fields
- *ab initio* MD
- Machine learning IPs
- Langevin Dynamics
- Thermostats *(NHC, CSVR, ...)*
- Enhanced Sampling *(Umbrella Sampling, Metadynamics, OPES, eABF, ...)*

Molecular Dynamics



Has not evolved at all:

Verlet / Velocity verlet / Leapfrog

*(all three are **identical**)*

Time Integration Algorithms

$$\frac{\mathrm{d}q}{\mathrm{d}t} = \frac{\partial \mathcal{H}(q,p)}{\partial p} =: u(p),$$

$$\frac{\mathrm{d}p}{\mathrm{d}t} = -\frac{\partial \mathcal{H}(q,p)}{\partial q} =: w(q)$$

„Canonical Equations“
from Hamilton formalism

→ Set of ODEs which we need to solve

Time Integration Algorithms

$$\frac{\mathrm{d}q}{\mathrm{d}t} = \frac{\partial \mathcal{H}(q,p)}{\partial p} =: u(p),$$
$$\frac{\mathrm{d}p}{\mathrm{d}t} = -\frac{\partial \mathcal{H}(q,p)}{\partial q} =: w(q)$$

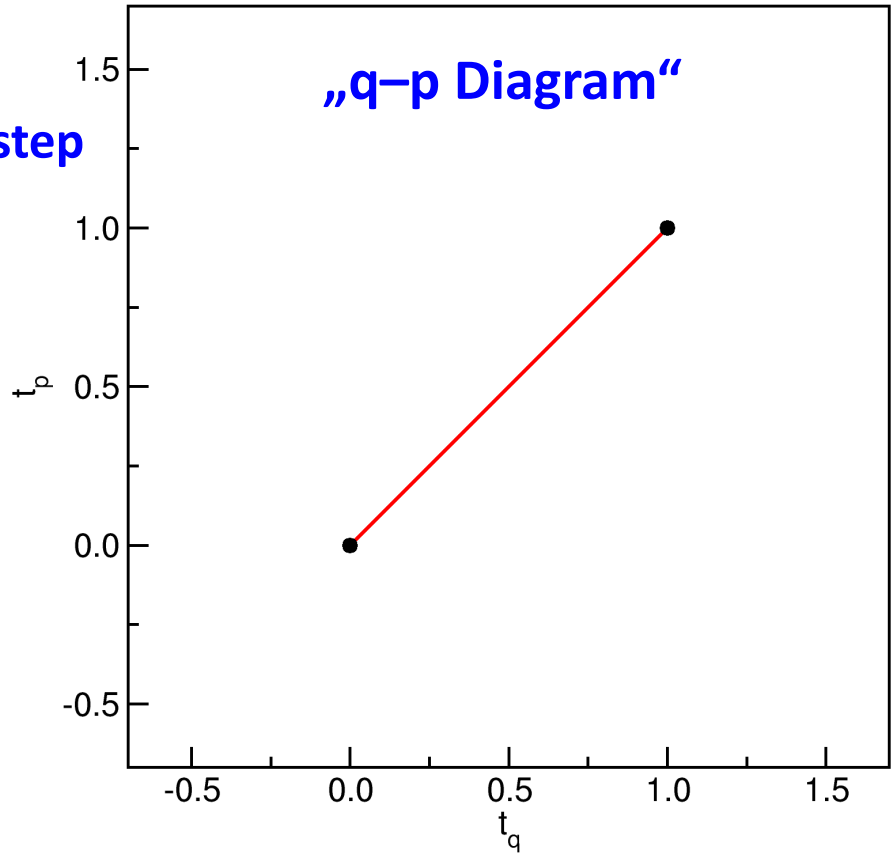
„Canonical Equations“
from Hamilton formalism

→ Set of ODEs which we need to solve

Explicit Euler Method:

$$q_{i+1} := q_i + h\, u(p_i),$$
$$p_{i+1} := p_i + h\, w(q_i)$$

h **= Δt is the finite time step**



Time Integration Algorithms

$$\frac{\mathrm{d}q}{\mathrm{d}t} = \frac{\partial \mathcal{H}(q,p)}{\partial p} =: u(p),$$
$$\frac{\mathrm{d}p}{\mathrm{d}t} = -\frac{\partial \mathcal{H}(q,p)}{\partial q} =: w(q)$$

„Canonical Equations“
from Hamilton formalism

→ Set of ODEs which we need to solve

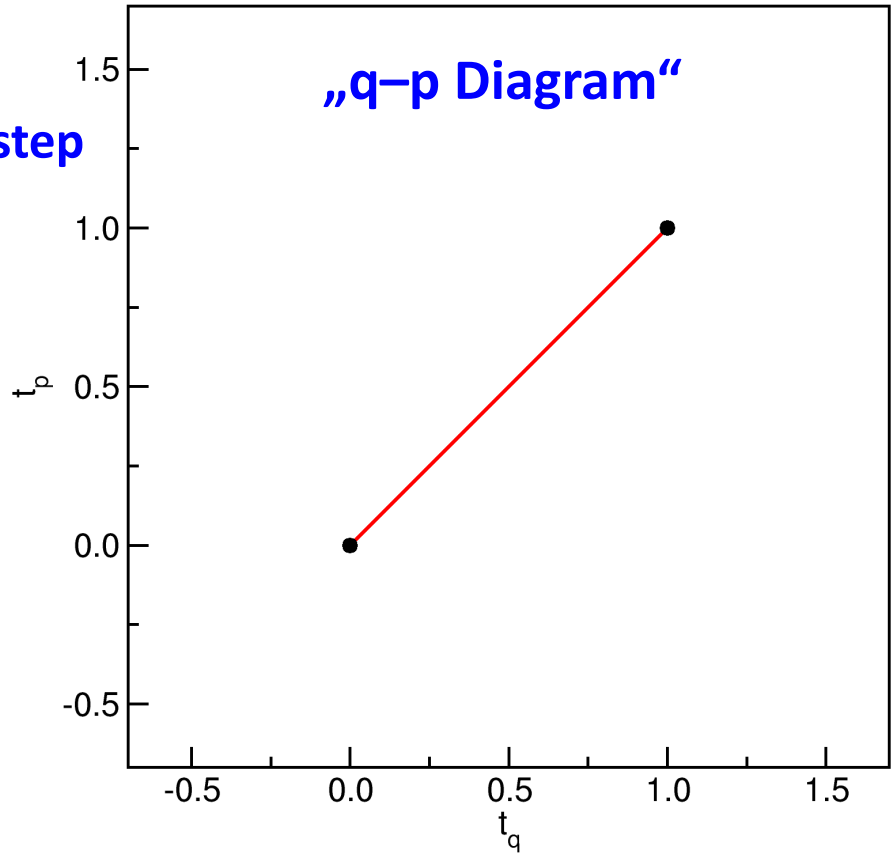
Explicit Euler Method:

$$q_{i+1} := q_i + h\,u(p_i),$$
$$p_{i+1} := p_i + h\,w(q_i)$$

h **= Δt is the finite time step**

Rather bad performance, not used...

Total energy not conserved



Time Integration Algorithms

$$\frac{\mathrm{d}q}{\mathrm{d}t} = \frac{\partial \mathcal{H}(q,p)}{\partial p} =: u(p),$$
$$\frac{\mathrm{d}p}{\mathrm{d}t} = -\frac{\partial \mathcal{H}(q,p)}{\partial q} =: w(q)$$

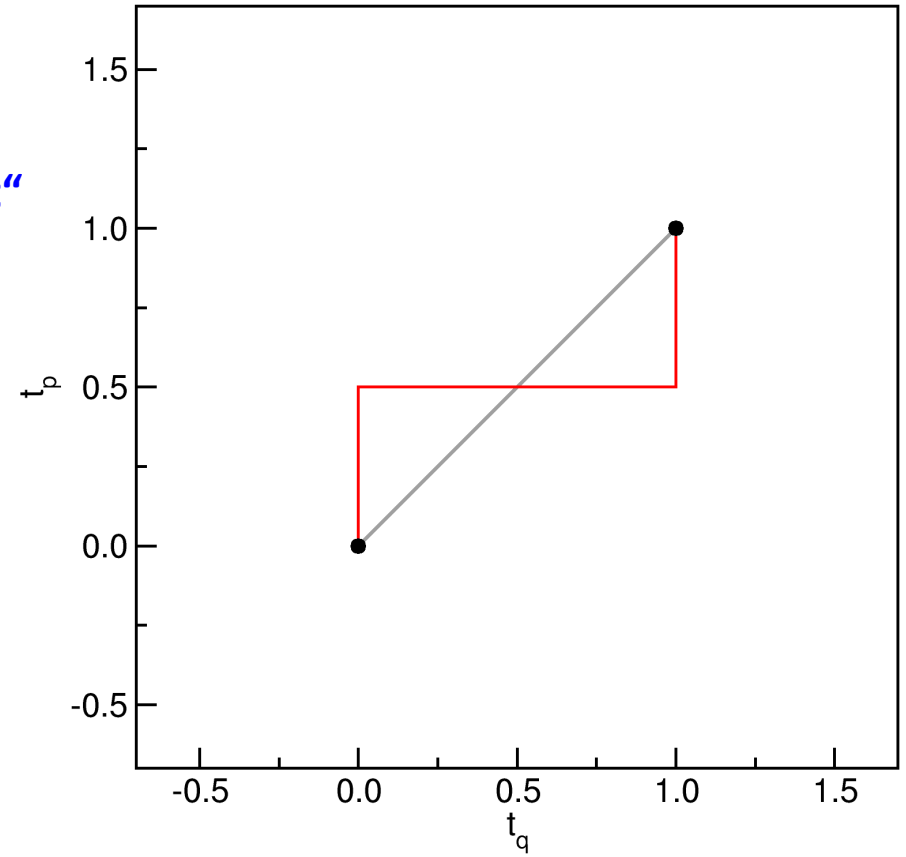
„Canonical Equations“
from Hamilton formalism

→ Set of ODEs which we need to solve

Störmer–Verlet Method:

$$p := p + \frac{1}{2} h\, w(q),$$
$$q := q + h\, u(p),$$
$$p := p + \frac{1}{2} h\, w(q)$$

„Verlet“
„Velocity Verlet“
„Leapfrog“
(all identical)



Time Integration Algorithms

$$\frac{\mathrm{d}q}{\mathrm{d}t} = \frac{\partial \mathcal{H}(q,p)}{\partial p} =: u(p),$$
$$\frac{\mathrm{d}p}{\mathrm{d}t} = -\frac{\partial \mathcal{H}(q,p)}{\partial q} =: w(q)$$

„Canonical Equations“
from Hamilton formalism

→ Set of ODEs which we need to solve

Störmer–Verlet Method:

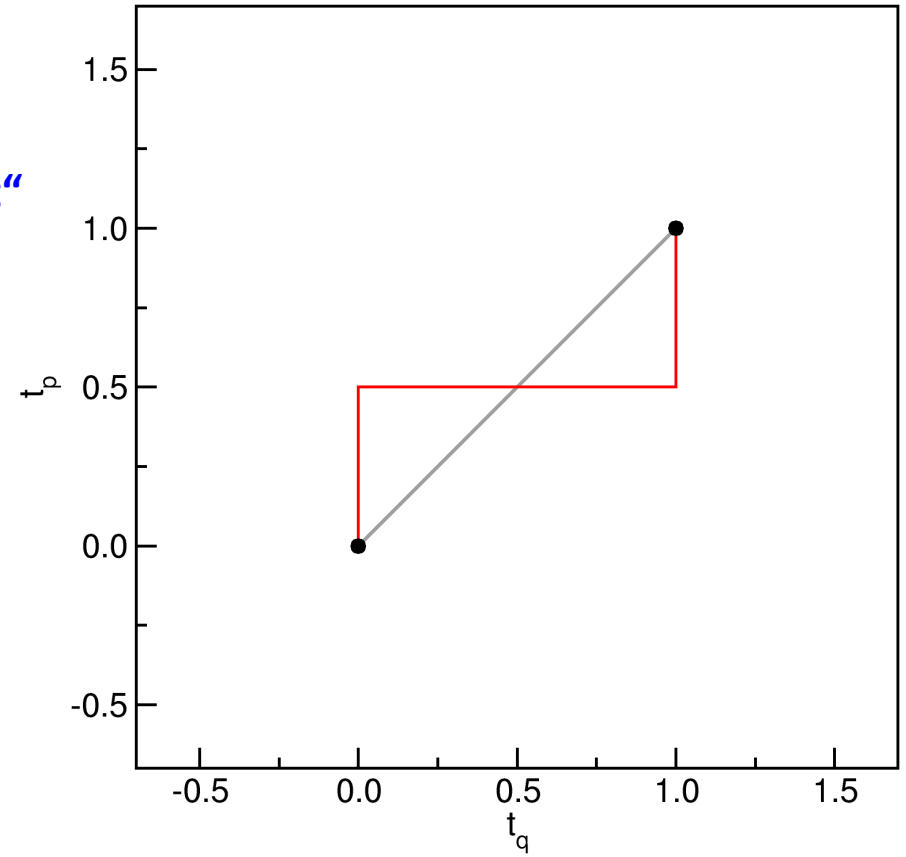
$$p := p + \frac{1}{2} h\, w(q),$$
$$q := q + h\, u(p),$$
$$p := p + \frac{1}{2} h\, w(q)$$

„Verlet“
„Velocity Verlet“
„Leapfrog“
(all identical)

Standard method
(used by „everybody“)

Good energy conservation

2nd order



Time Integration Algorithms

$$\frac{\mathrm{d}q}{\mathrm{d}t} = \frac{\partial \mathcal{H}(q,p)}{\partial p} =: u(p),$$
$$\frac{\mathrm{d}p}{\mathrm{d}t} = -\frac{\partial \mathcal{H}(q,p)}{\partial q} =: w(q)$$

„Canonical Equations“
from Hamilton formalism

→ Set of ODEs which we need to solve

Störmer–Verlet Method:

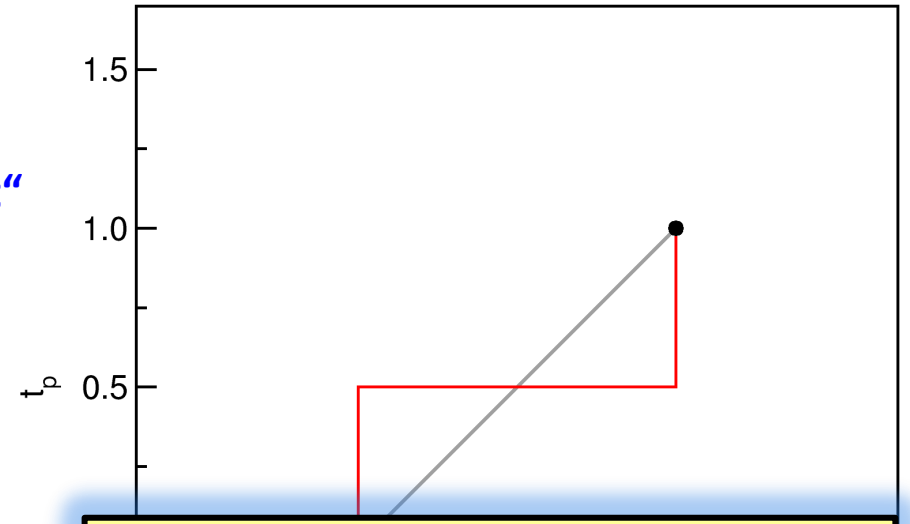
$$p := p + \frac{1}{2} h\, w(q),$$
$$q := q + h\, u(p),$$
$$p := p + \frac{1}{2} h\, w(q)$$

„Verlet“
„Velocity Verlet“
„Leapfrog“
(all identical)

Standard method
(used by „everybody“)

Good energy conservation

2nd order



„Xth order“ means:

If timestep $h = \Delta t$ is reduced by factor 2, energy conservation is improved by factor 2^X.

Time Integration Algorithms

$$\frac{\mathrm{d}q}{\mathrm{d}t} = \frac{\partial \mathcal{H}(q,p)}{\partial p} =: u(p),$$
$$\frac{\mathrm{d}p}{\mathrm{d}t} = -\frac{\partial \mathcal{H}(q,p)}{\partial q} =: w(q)$$

„Canonical Equations“
from Hamilton formalism

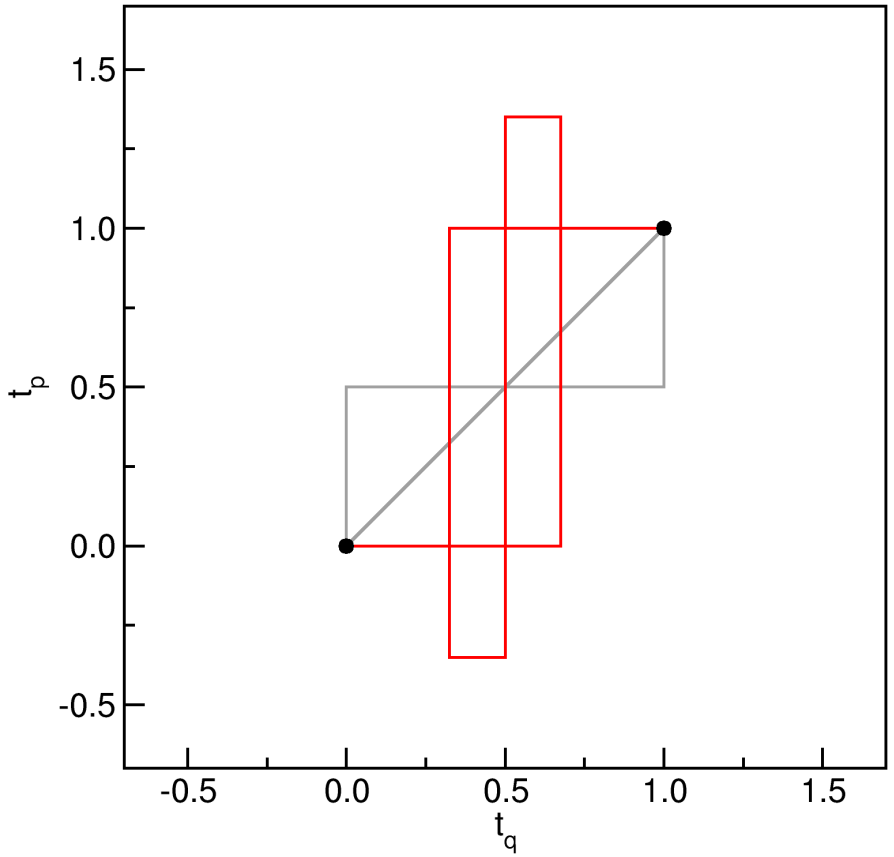
→ Set of ODEs which we need to solve

Yoshida‘s 4th Order Method:

$$p := p + a_1 h\, w(q),$$
$$q := q + b_1 h\, u(p),$$
$$\ldots$$
$$p := p + a_s h\, w(q),$$
$$q := q + b_s h\, u(p),$$

„Splitting Method“

b_n	a_n
0	0.6756035959798288
1.351207191959658	–0.1756035959798288
–1.702414383919315	–0.1756035959798288
1.351207191959658	0.6756035959798288



Time Integration Algorithms

$$\frac{\mathrm{d}q}{\mathrm{d}t} = \frac{\partial \mathcal{H}(q,p)}{\partial p} =: u(p),$$
$$\frac{\mathrm{d}p}{\mathrm{d}t} = -\frac{\partial \mathcal{H}(q,p)}{\partial q} =: w(q)$$

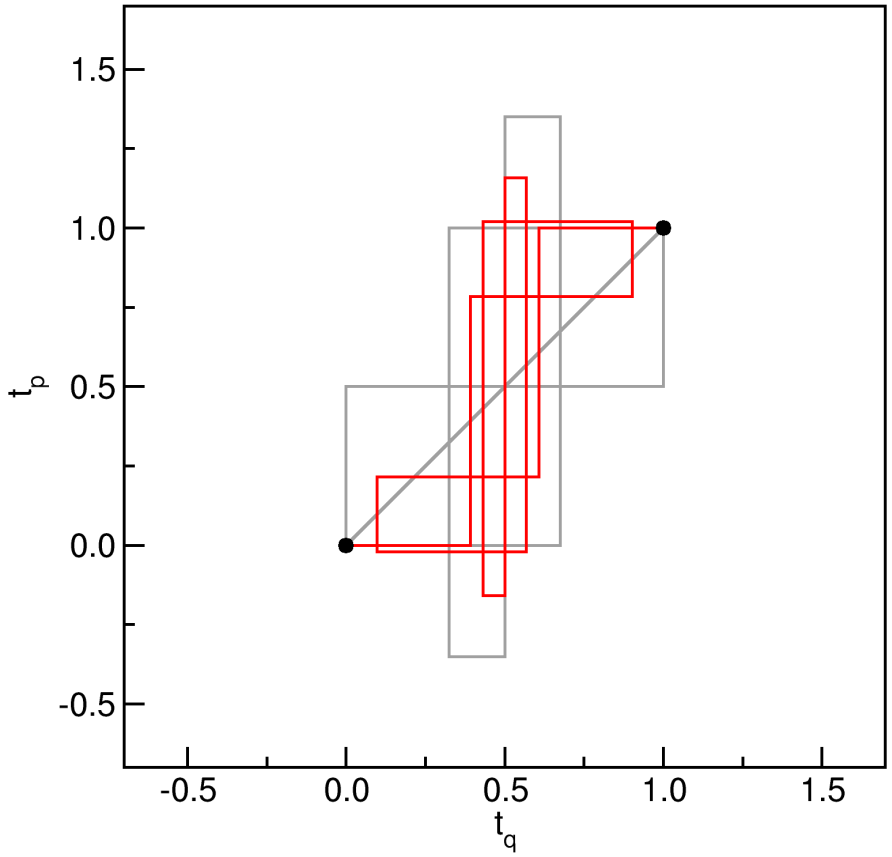
„Canonical Equations“
from Hamilton formalism

→ Set of ODEs which we need to solve

Yoshida‘s 6th Order Method:

$$p := p + a_1\, h\, w(q),$$
$$q := q + b_1\, h\, u(p),$$
$$\ldots$$
$$p := p + a_s\, h\, w(q),$$
$$q := q + b_s\, h\, u(p),$$

b_n	a_n
0	0.3922568052387800
0.784513610477560	0.5100434119184585
0.235573213359357	–0.4710533854097565
–1.177679984178870	0.0687531682525180
1.315186320683906	0.0687531682525180
–1.177679984178870	–0.4710533854097565
0.235573213359357	0.5100434119184585
0.784513610477560	0.3922568052387800



Time Integration Algorithms

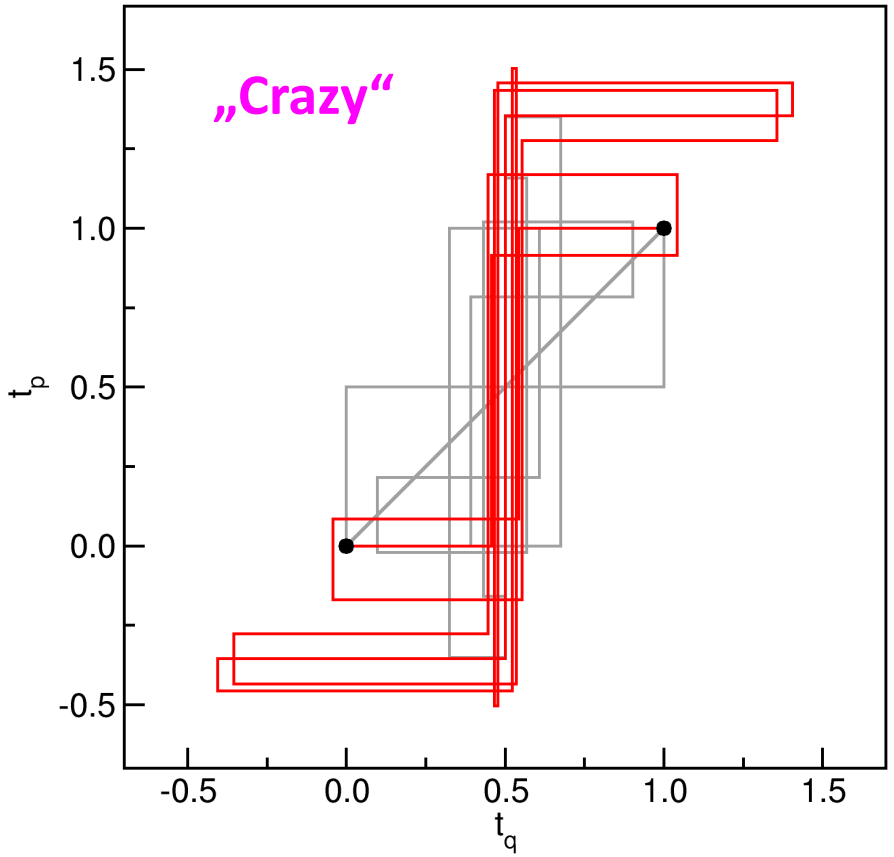
$$\frac{\mathrm{d}q}{\mathrm{d}t} = \frac{\partial \mathcal{H}(q,p)}{\partial p} =: u(p),$$
$$\frac{\mathrm{d}p}{\mathrm{d}t} = -\frac{\partial \mathcal{H}(q,p)}{\partial q} =: w(q)$$

„Canonical Equations“
from Hamilton formalism

→ Set of ODEs which we need to solve

Yoshida‘s 8th Order Method:

b_n	a_n
0	0.4574221231148700
0.914844246229740	0.5842687913979845
0.253693336566229	–0.5955794501471255
–1.444852236860480	–0.8015464361143615
–0.158240635368243	0.8899492511272585
1.938139137622760	–0.0112355476763650
–1.960610232975490	–0.9289051917917525
0.102799849391985	0.9056264600894915
1.708453070786998	0.9056264600894915
0.102799849391985	–0.9289051917917525
–1.960610232975490	–0.0112355476763650
1.938139137622760	0.8899492511272585
–0.158240635368243	–0.8015464361143615
–1.444852236860480	–0.5955794501471255
0.253693336566229	0.5842687913979845
0.914844246229740	0.4574221231148700



Practical MD Performance

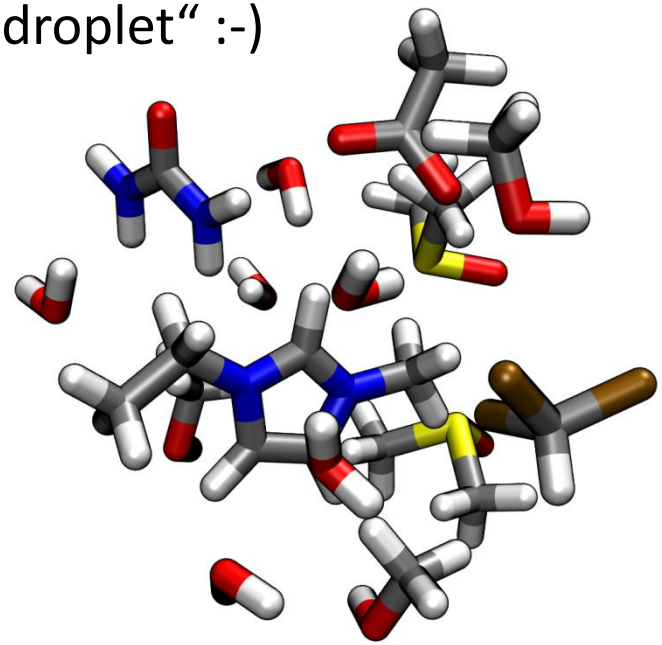
MB wrote his mathematics Diploma thesis
on symplectic integration algorithms...

→ Idea to benchmark those for MD simulations → **better efficiency?**

Benchmark system: „Magic droplet“ :-)

Contains ≈ 100 atoms:

- Water
- Methanol
- [EMIm][OAc]
- DMSO
- Urea
- Chloroform



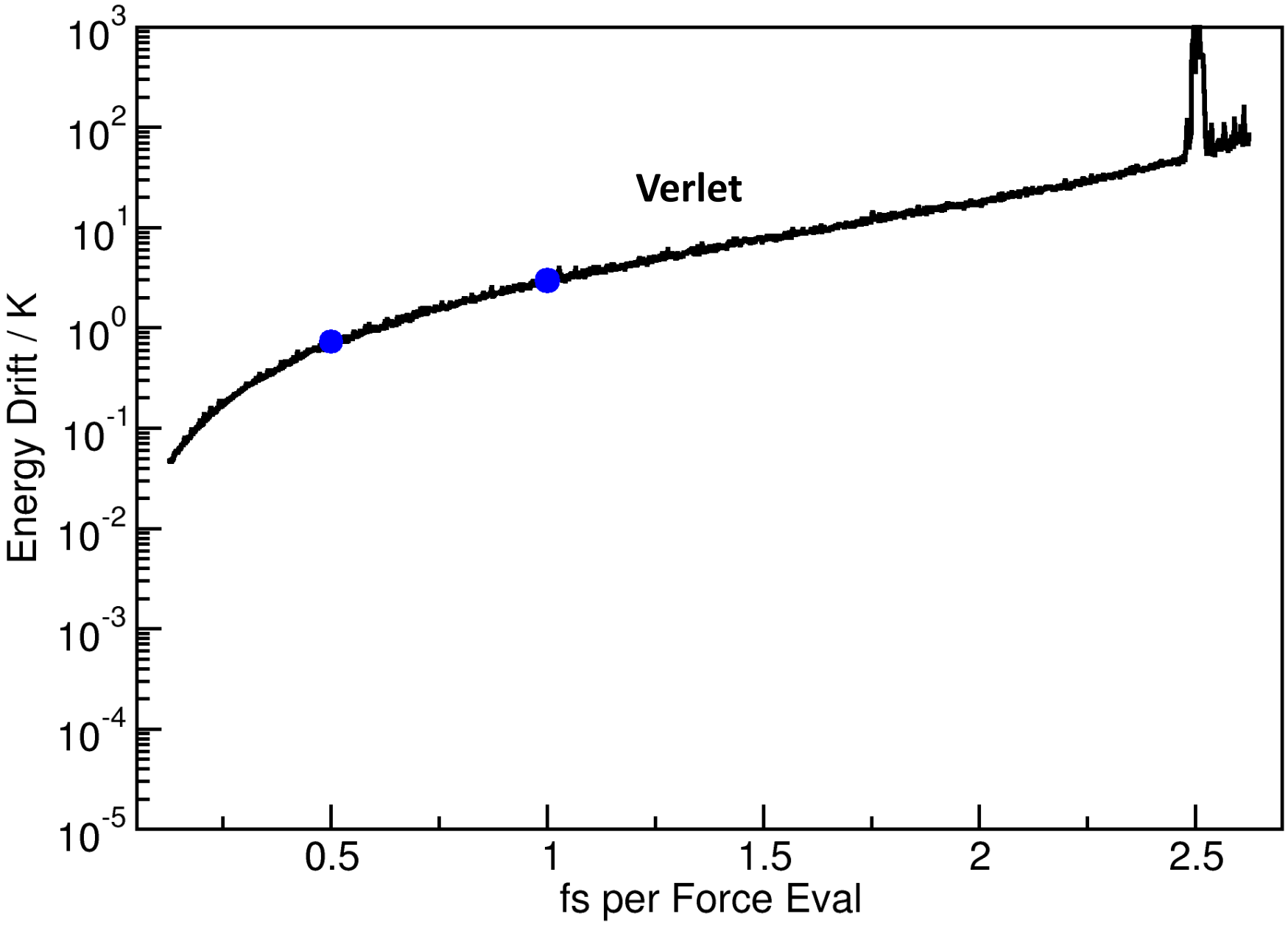
Do simple force field MD (OPLS-AA) for 6 ps per run,
monitor **maximum energy deviation**.

Highest frequency:

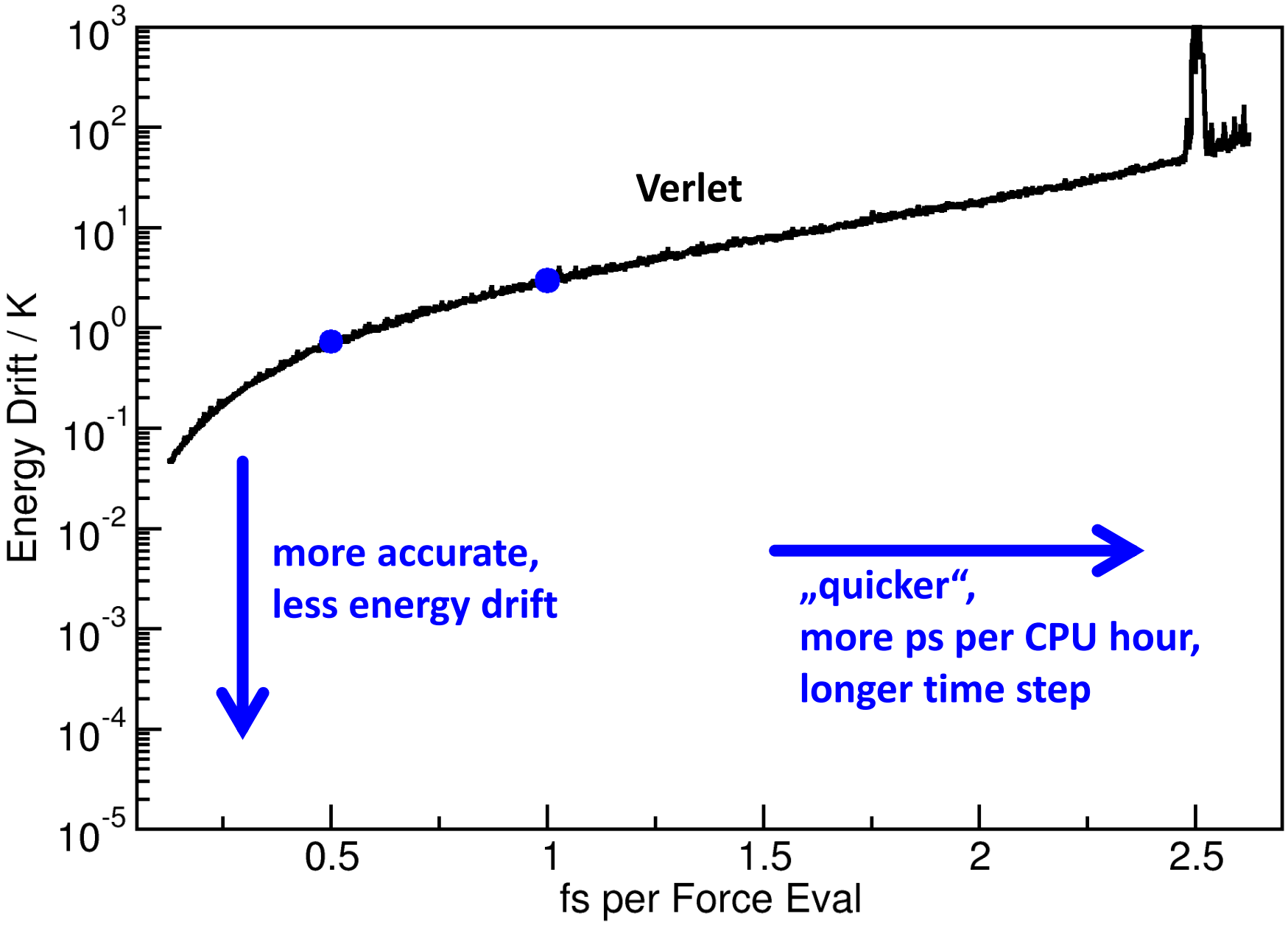
O-H stretching,
3706 cm^{-1}

no constraints
(all flexible)

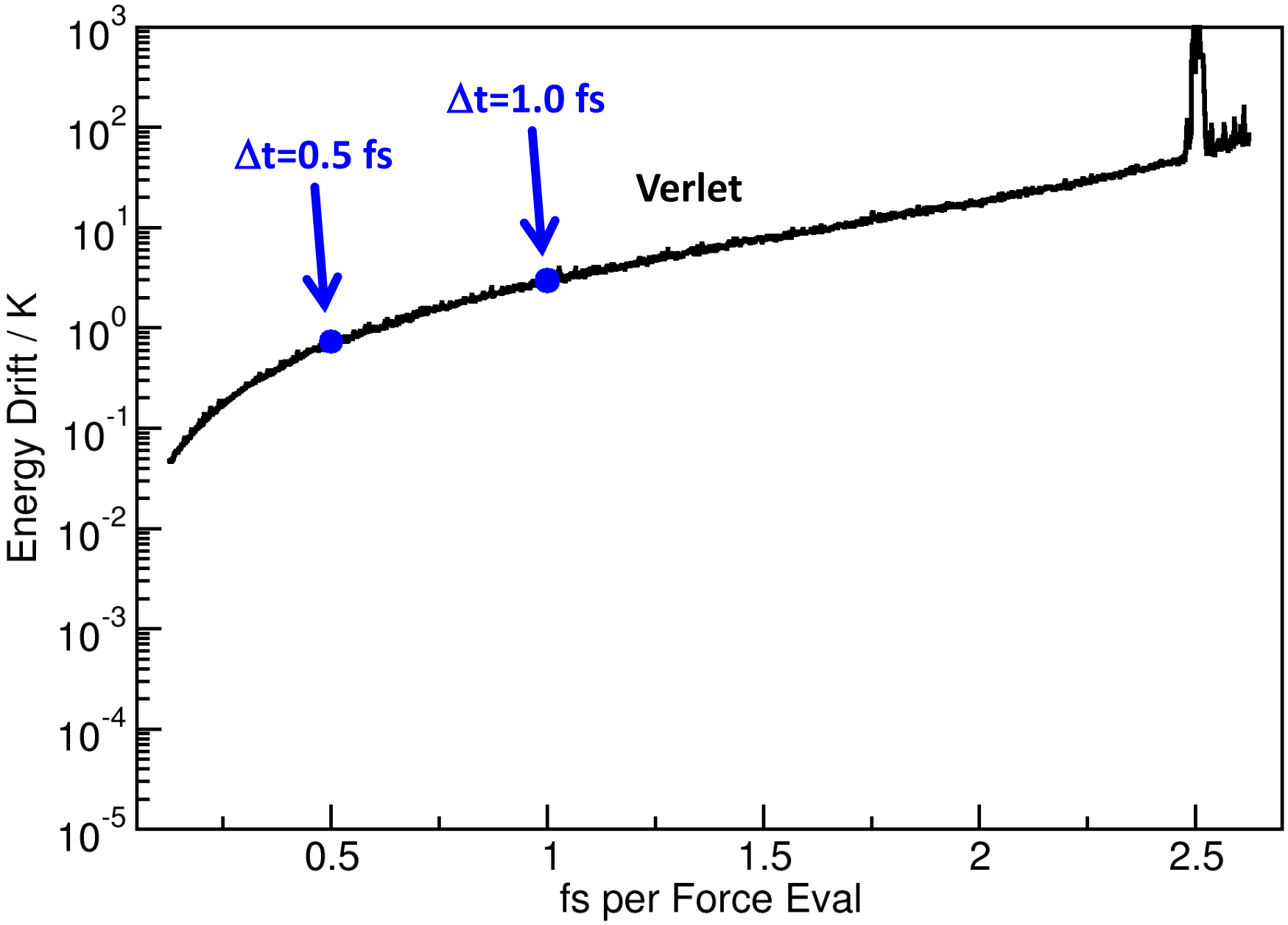
Practical MD Performance



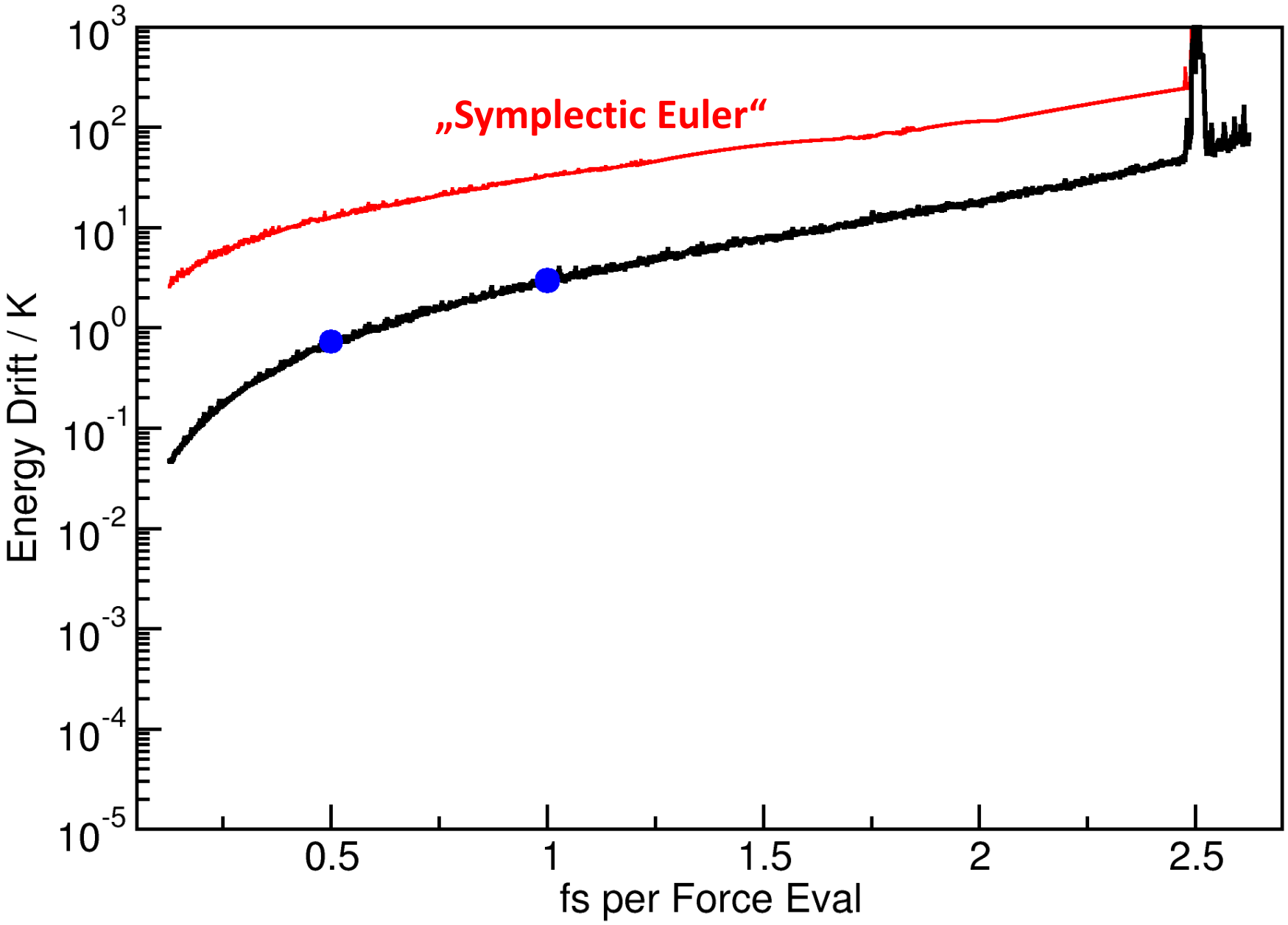
Practical MD Performance



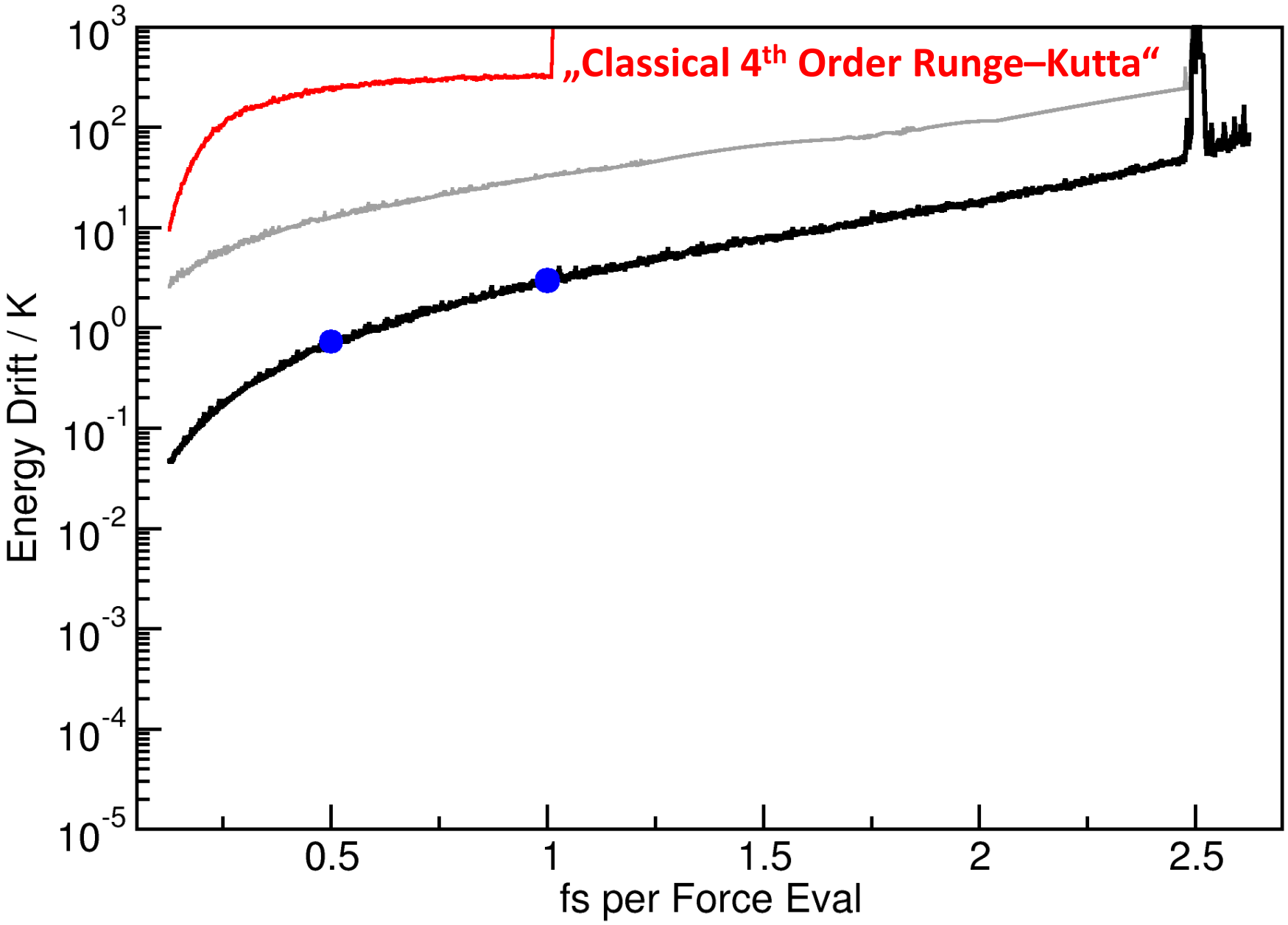
Practical MD Performance



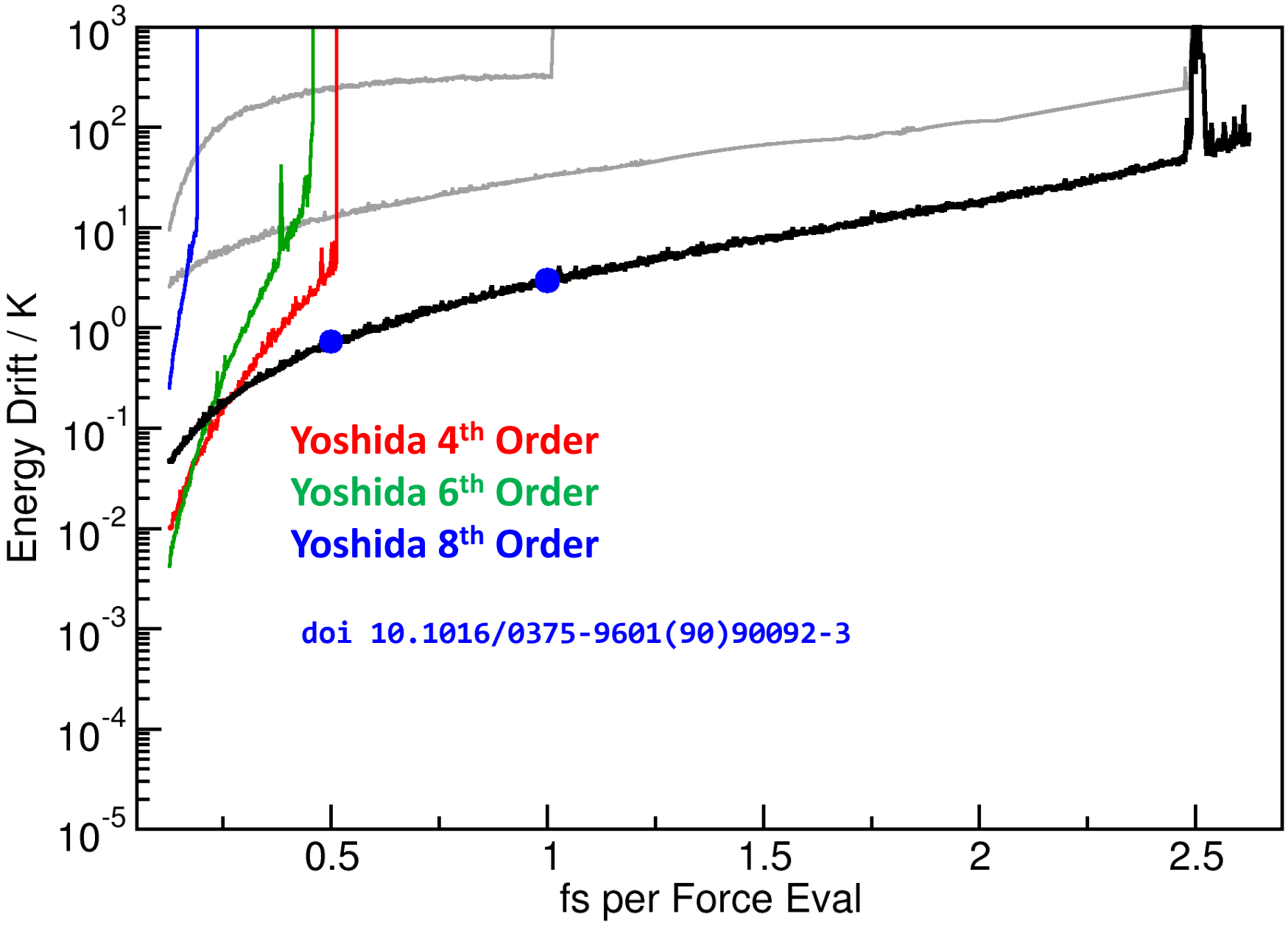
Practical MD Performance



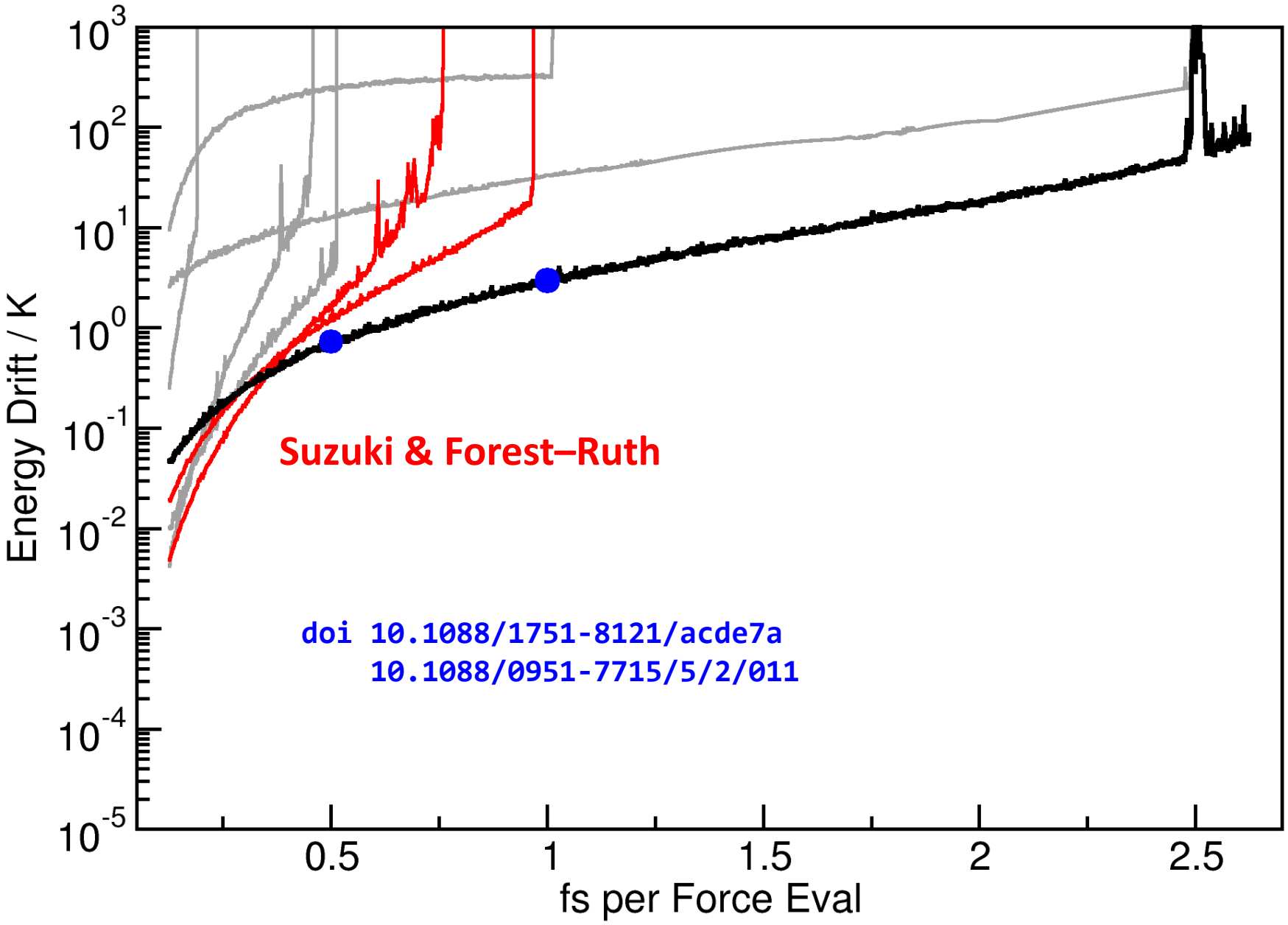
Practical MD Performance



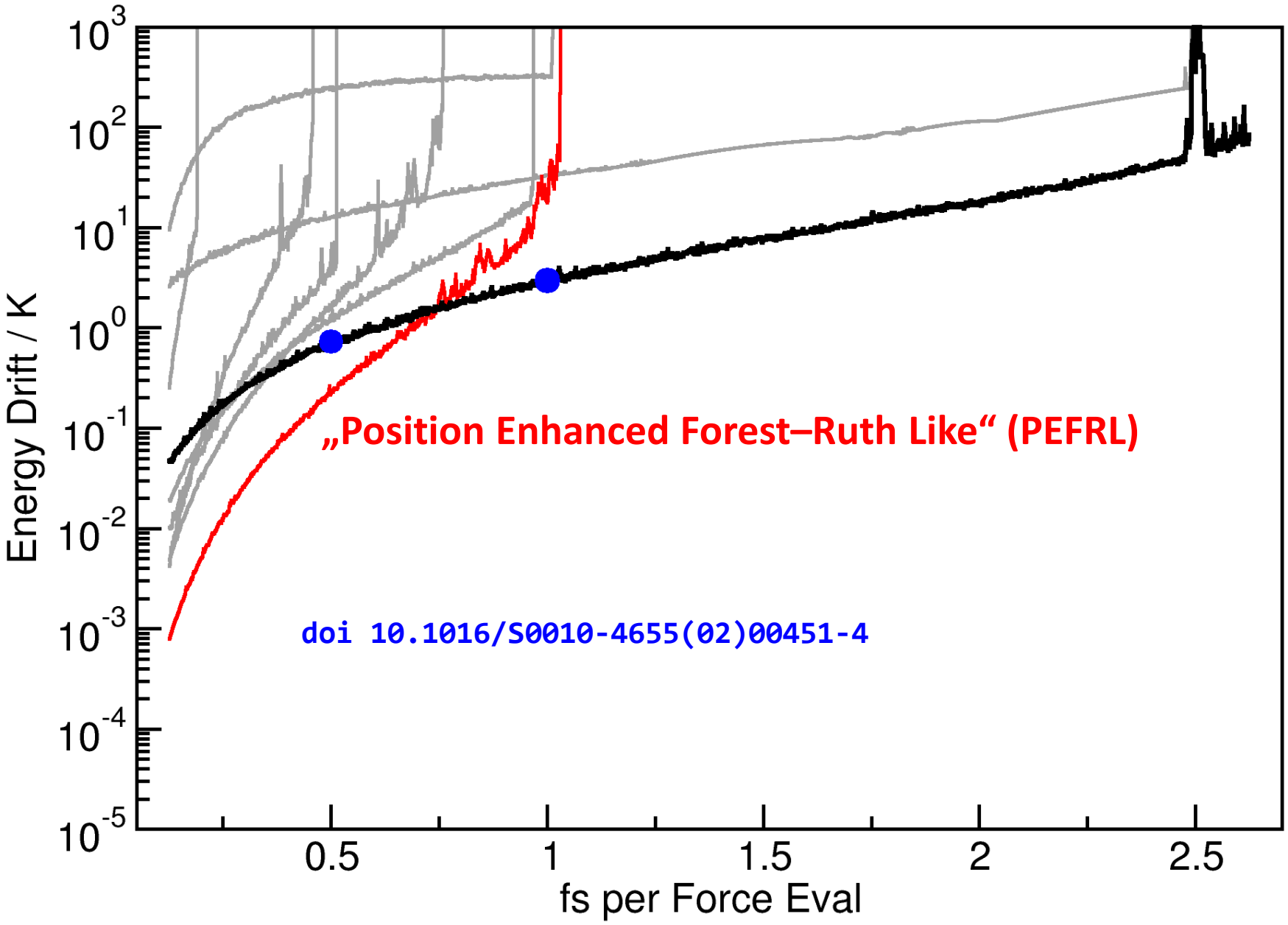
Practical MD Performance



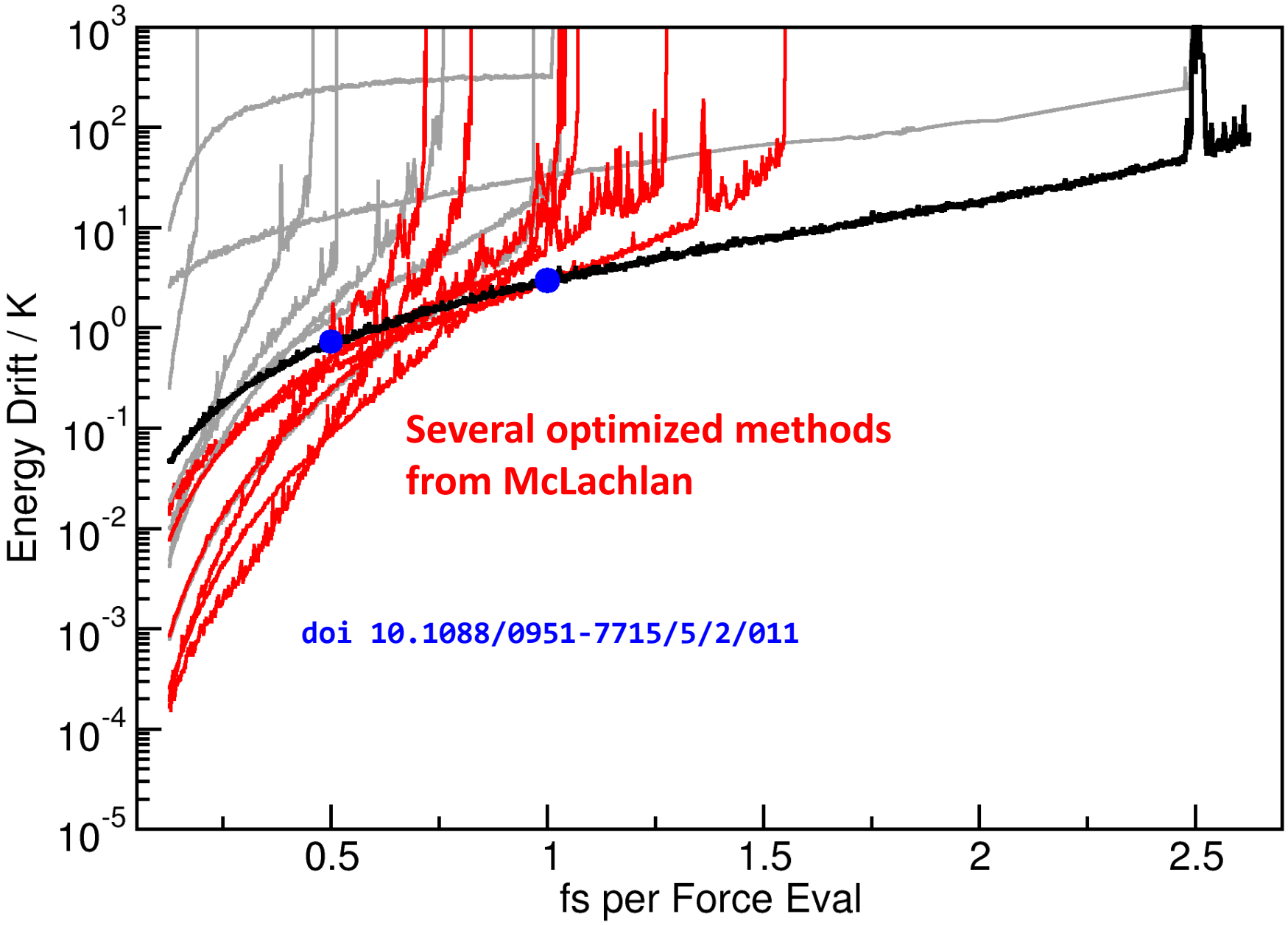
Practical MD Performance



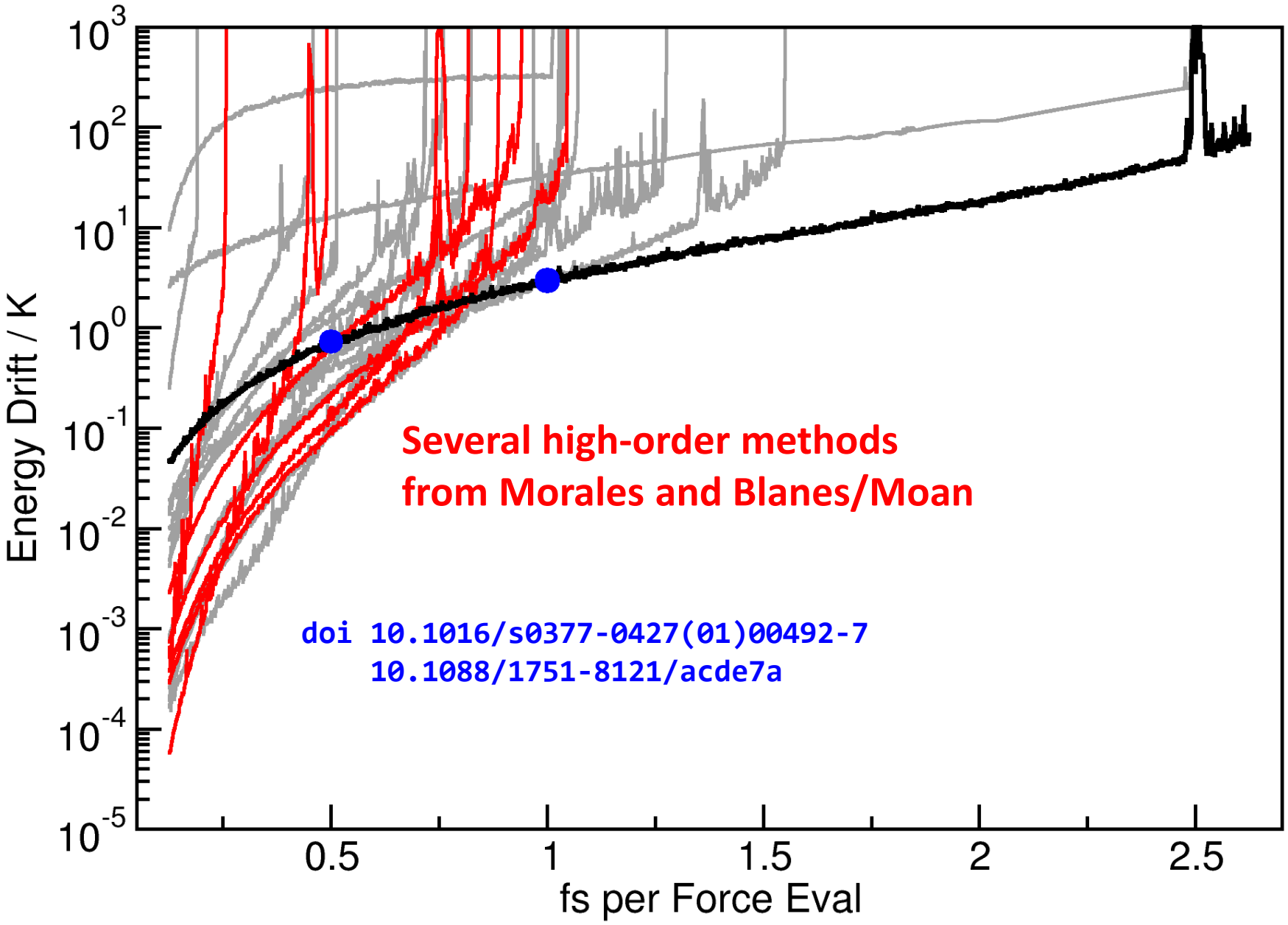
Practical MD Performance



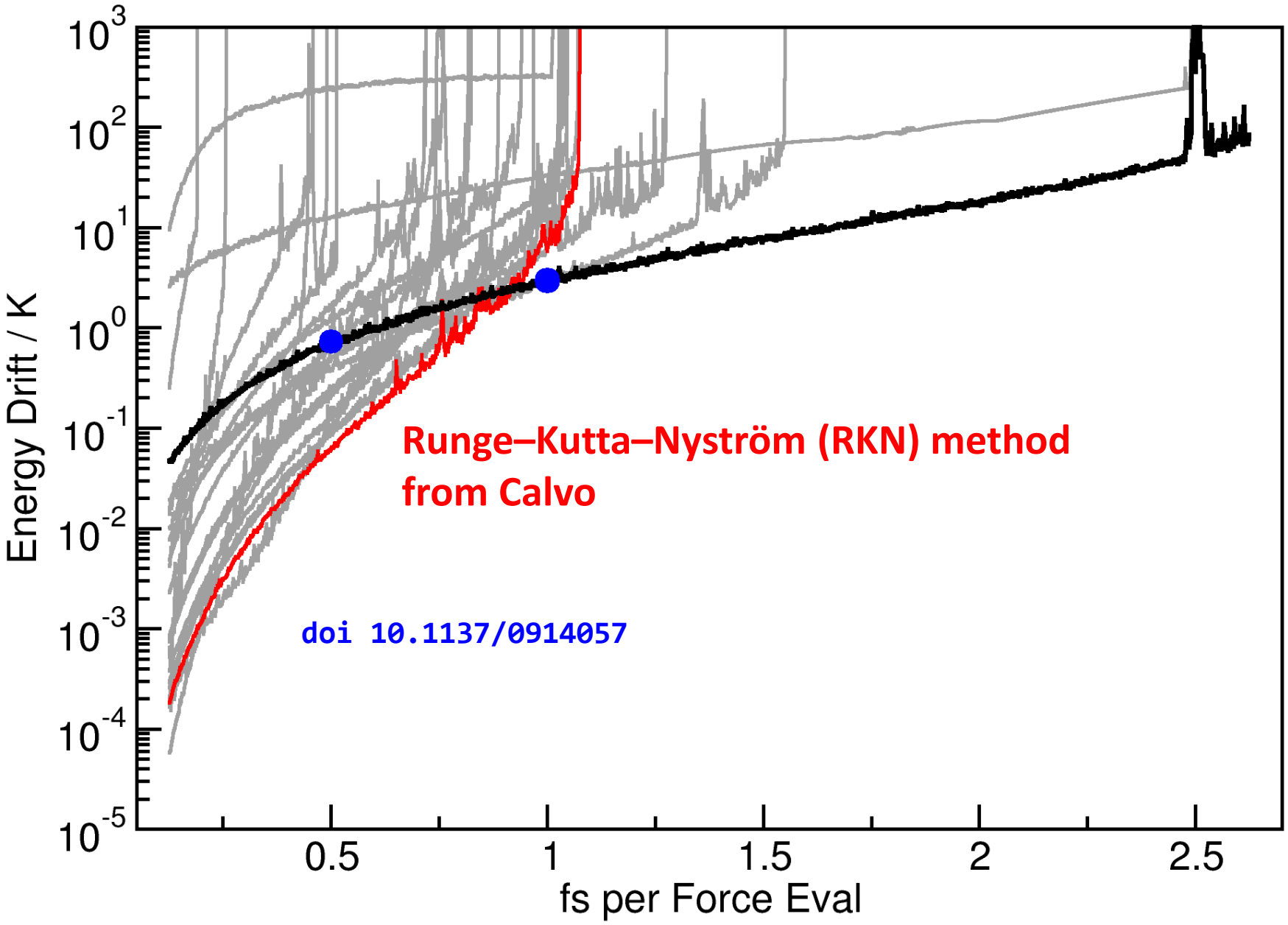
Practical MD Performance



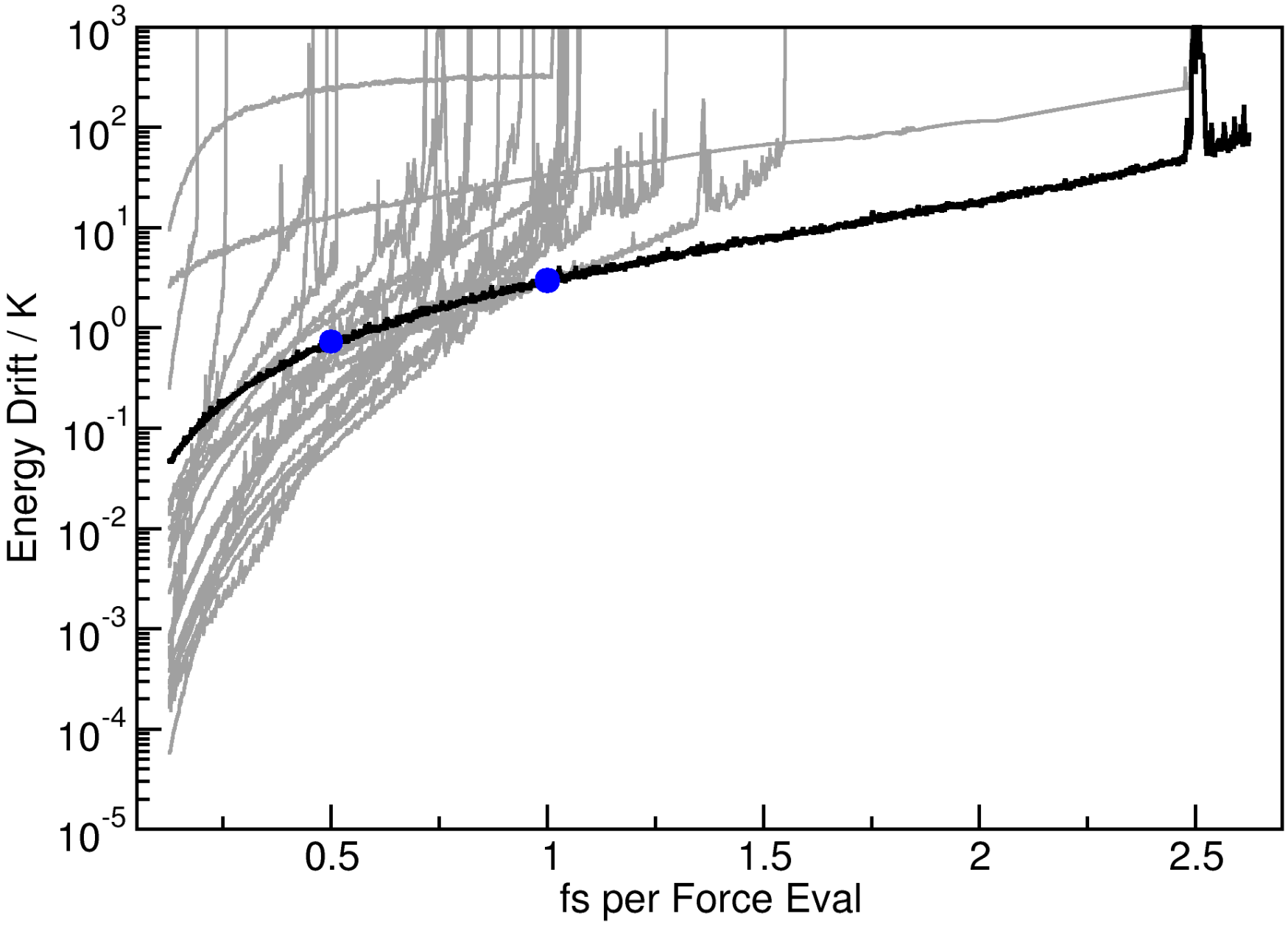
Practical MD Performance



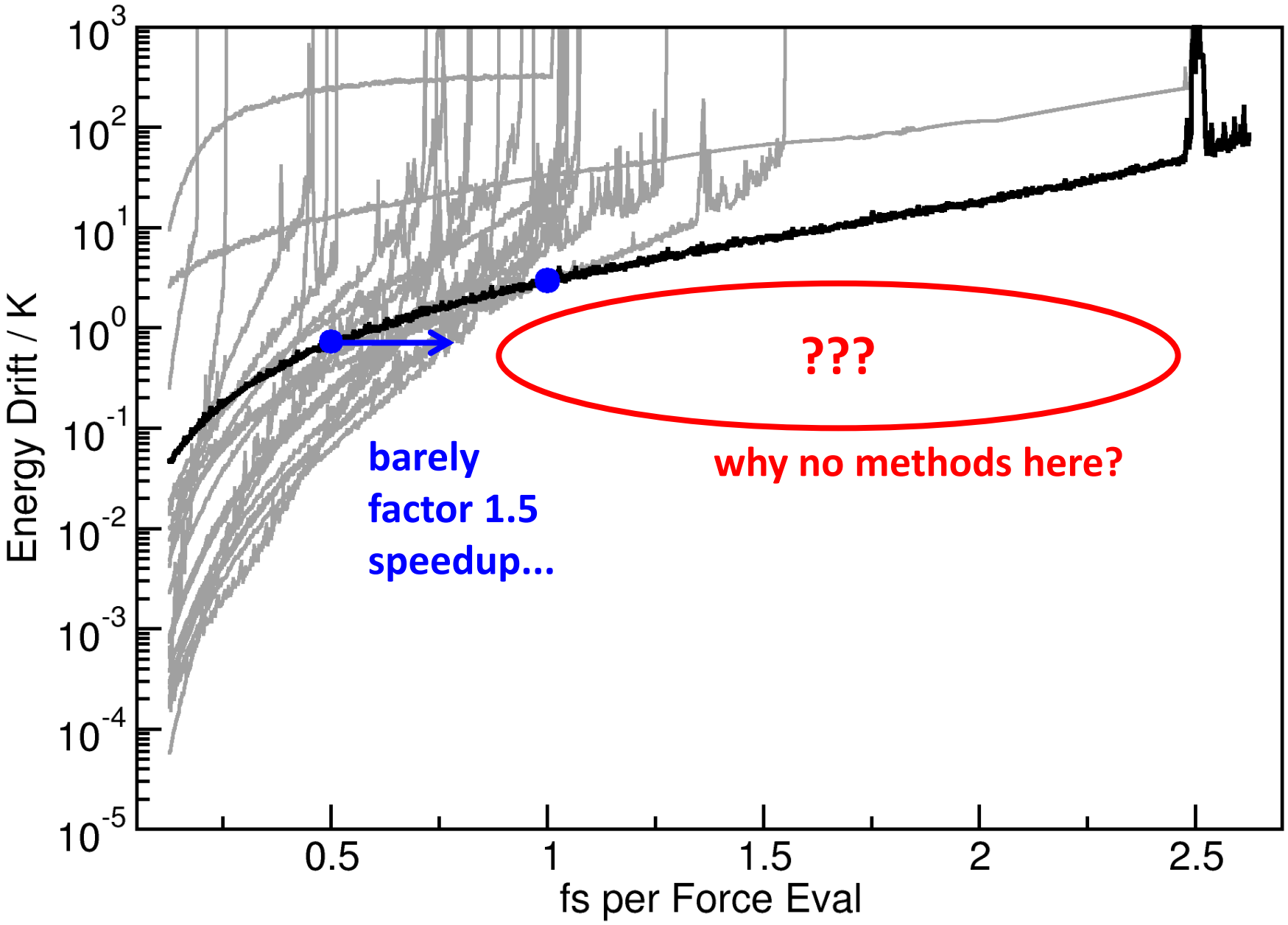
Practical MD Performance



Practical MD Performance



Practical MD Performance



Conclusions I

- Existing literature intgeration methods **barely** outperform Verlet
- They are designed to have a **high order**...
- But a **high order is detrimental** when larger steps are desired!
- Most promising is a **Runge–Kutta–Nyström** (RKN) method...

→ maybe try to optimize those further?

Runge–Kutta–Nyström Methods

$$y_{i+1} := y_i + hz_i + h^2 \sum_{l=1}^{s} \beta_l k_l,$$

$$z_{i+1} := z_i + h \sum_{l=1}^{s} b_l k_l,$$

$$k_l := f\left(t + c_l h,\ \ y_i + c_l h z_i + h^2 \sum_{m=1}^{s} a_{lm} k_m,\ \ z_i + h \sum_{m=1}^{s} \hat{a}_{lm} k_m \right)$$

c_1	a_{11}	$\cdots$	a_{1s}
$\vdots$	$\vdots$	$\ddots$	$\vdots$
c_s	a_{s1}	$\cdots$	a_{ss}
	b_1	$\cdots$	b_s
	β_1	$\cdots$	β_s

Butcher Tableau

→ they are „beasts“...

Many intermediate results required

For 8 stages: 88 parameters!

Runge–Kutta–Nyström Methods

MB‘s math diploma thesis:

„Every symplectic RKN method is equivalent to a splitting method“

→ these are a lot simpler :-)

$$
\begin{aligned}
p &:= p + a_1\, h\, w(q), \\
q &:= q + b_1\, h\, u(p), \\
&\dots \\
p &:= p + a_s\, h\, w(q), \\
q &:= q + b_s\, h\, u(p),
\end{aligned}
$$

Just consecutive p and q updates with lengths a_s and b_s

b_n	a_n
b_1	a_1
b_2	a_2
b_3	a_3
b_4	a_4
b_5	a_4
b_4	a_3
b_3	a_2
b_2	a_1
b_1	0

8-stage symmetric method
→ 9 parameters

Runge–Kutta–Nyström Methods

MB‘s math diploma thesis:

„Every symplectic RKN method is equivalent to a splitting method“

→ these are a lot simpler :-)

$$
\begin{aligned}
p &:= p + a_1\, h\, w(q), \\
q &:= q + b_1\, h\, u(p), \\
&\dots \\
p &:= p + a_s\, h\, w(q), \\
q &:= q + b_s\, h\, u(p),
\end{aligned}
$$

b_n	a_n
b_1	a_1
b_2	a_2
b_3	a_3
b_4	a_4
b_5	a_4
b_4	a_3
b_3	a_2
b_2	a_1
b_1	0

8-stage symmetric method
→ 9 parameters

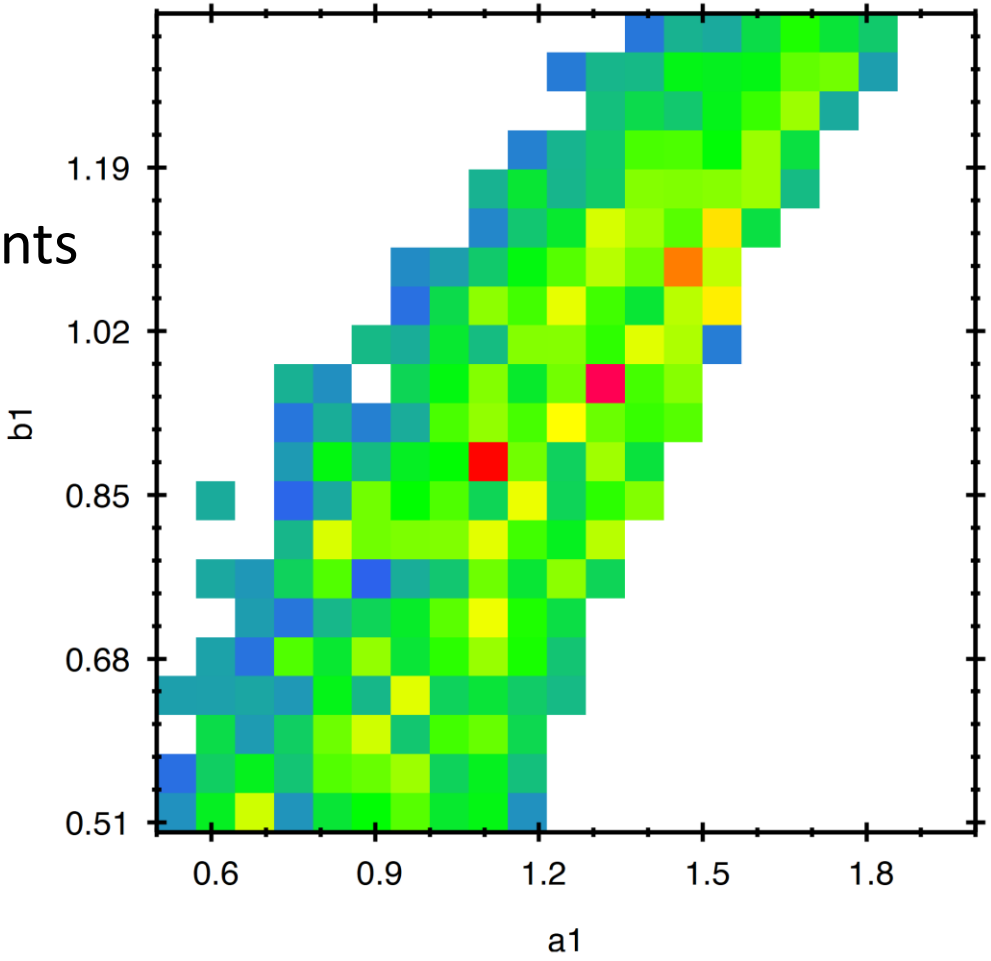
Consistency Conditions:

$$\sum_{i=1}^{s} a_i = 1 \qquad \sum_{i=1}^{s} b_i = 1$$

→ only 7 parameters remain

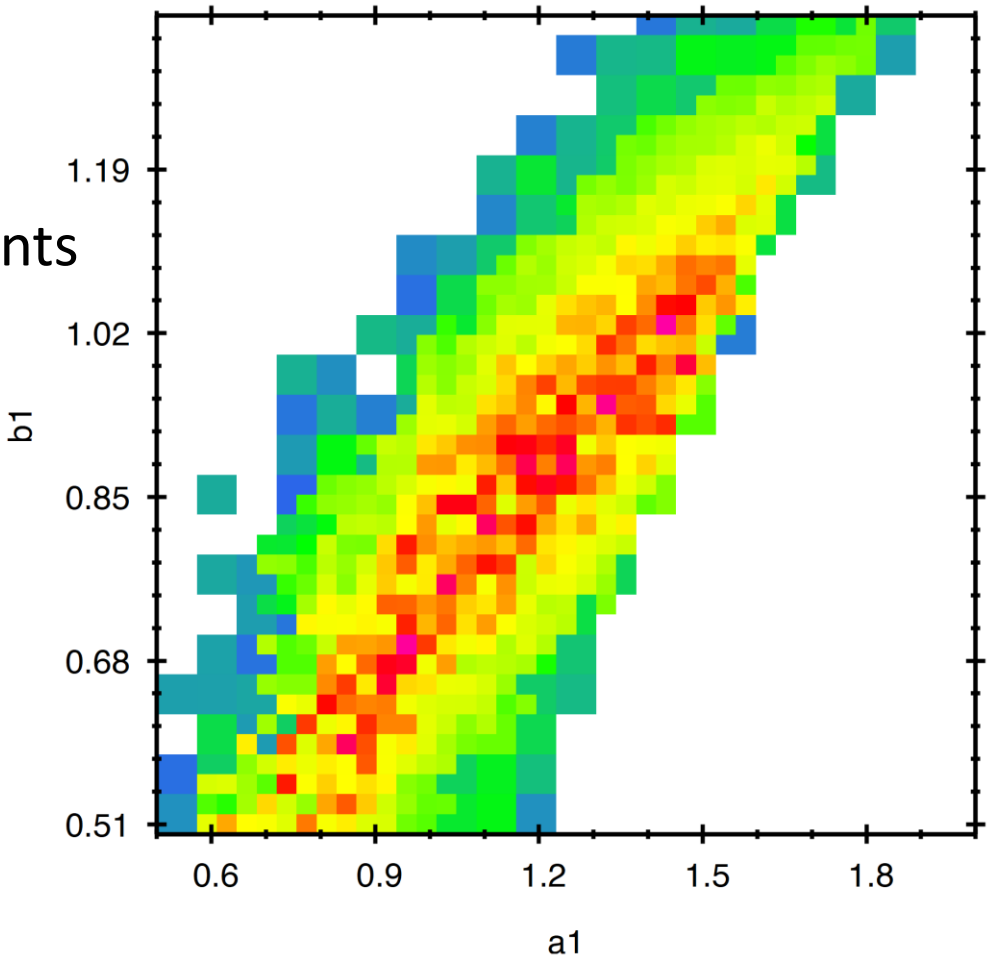
Numerical Optimization

- 7-dim. optimization problem, many local minima...
- We used an adaptive grid approach
- Start with $21^7 \approx 1.8 \cdot 10^9$ grid points



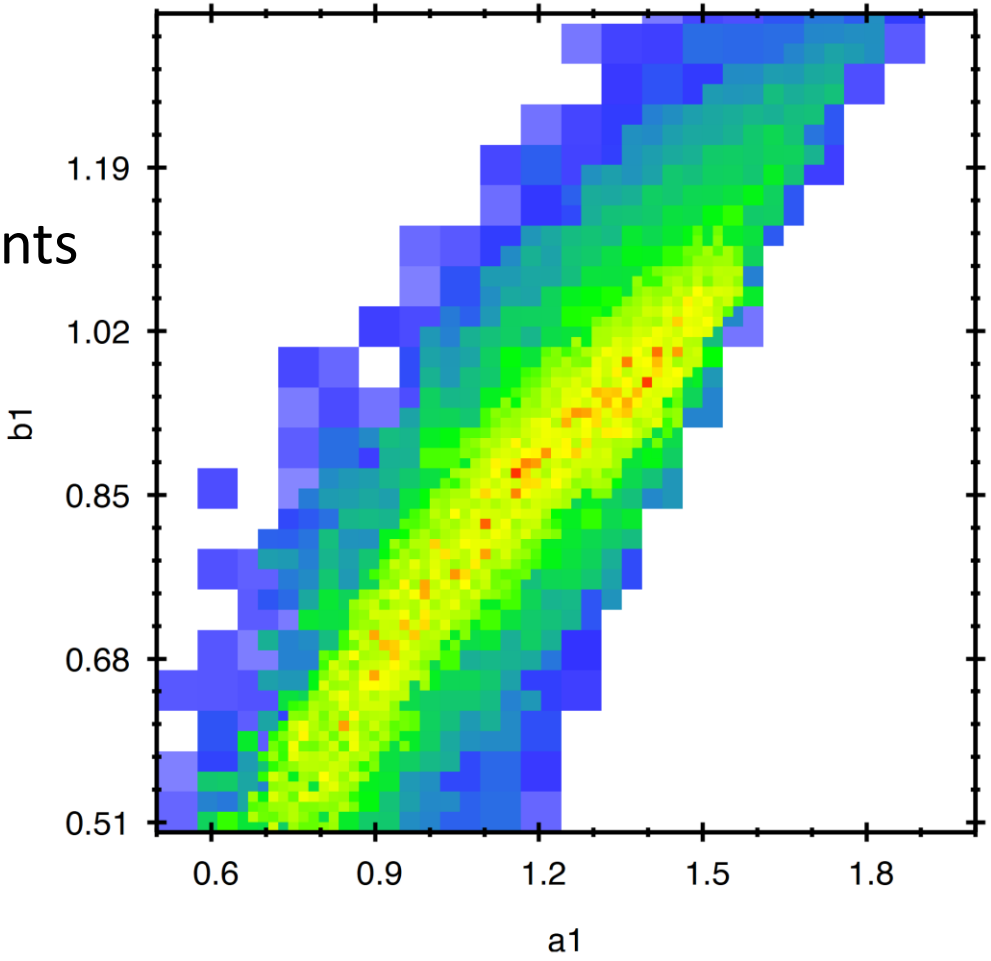
Numerical Optimization

- 7-dim. optimization problem, many local minima...
- We used an adaptive grid approach
- Start with $21^7 \approx 1.8 \cdot 10^9$ grid points
- Pick best 5000 points, run refined grid *(factor 2)*



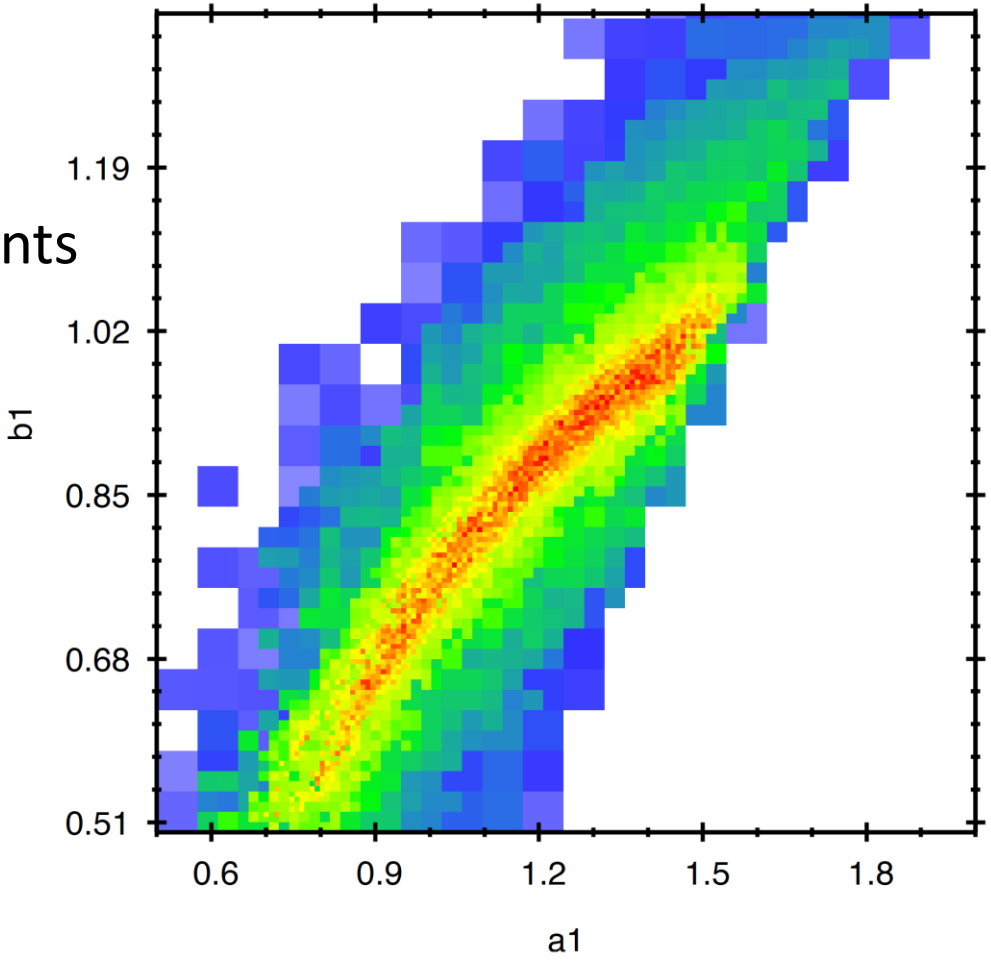
Numerical Optimization

- 7-dim. optimization problem, many local minima...
- We used an adaptive grid approach
- Start with $21^7 \approx 1.8 \cdot 10^9$ grid points
- Pick best 5000 points, run refined grid *(factor 2)*
- Continue until convergence



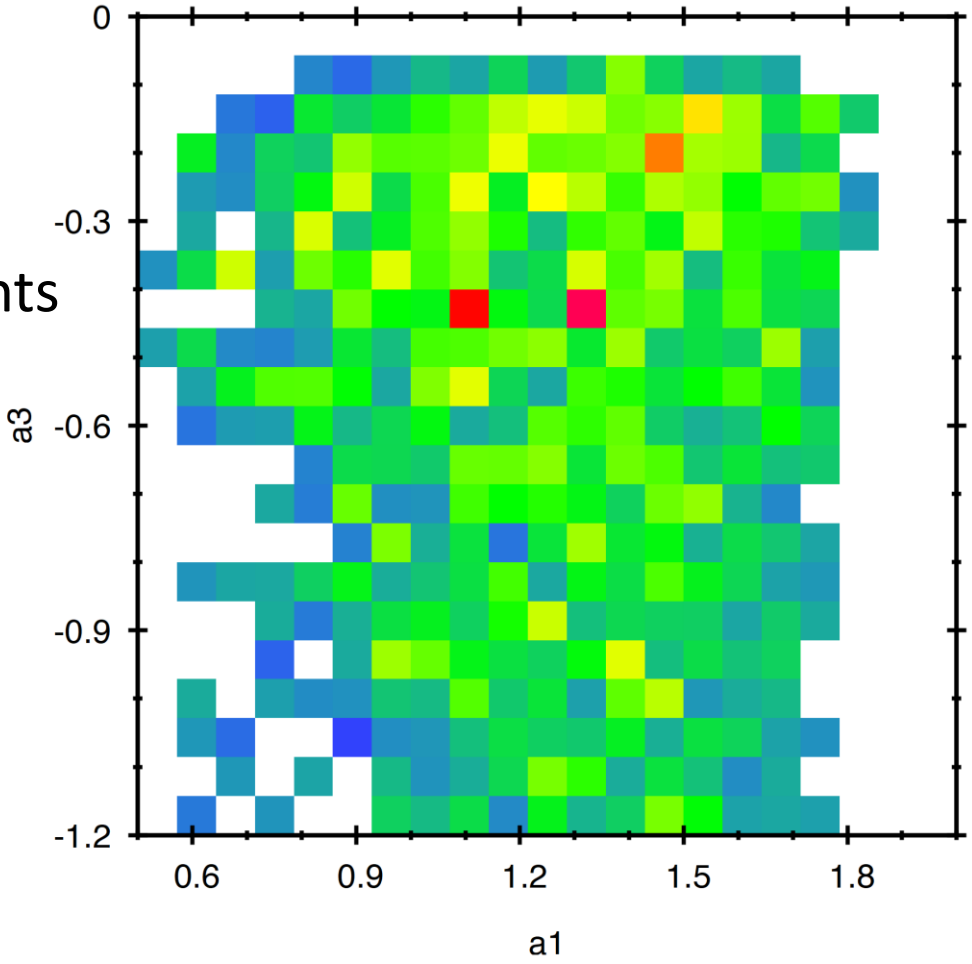
Numerical Optimization

- 7-dim. optimization problem, many local minima...
- We used an adaptive grid approach
- Start with $21^7 \approx 1.8 \cdot 10^9$ grid points
- Pick best 5000 points, run refined grid *(factor 2)*
- Continue until convergence



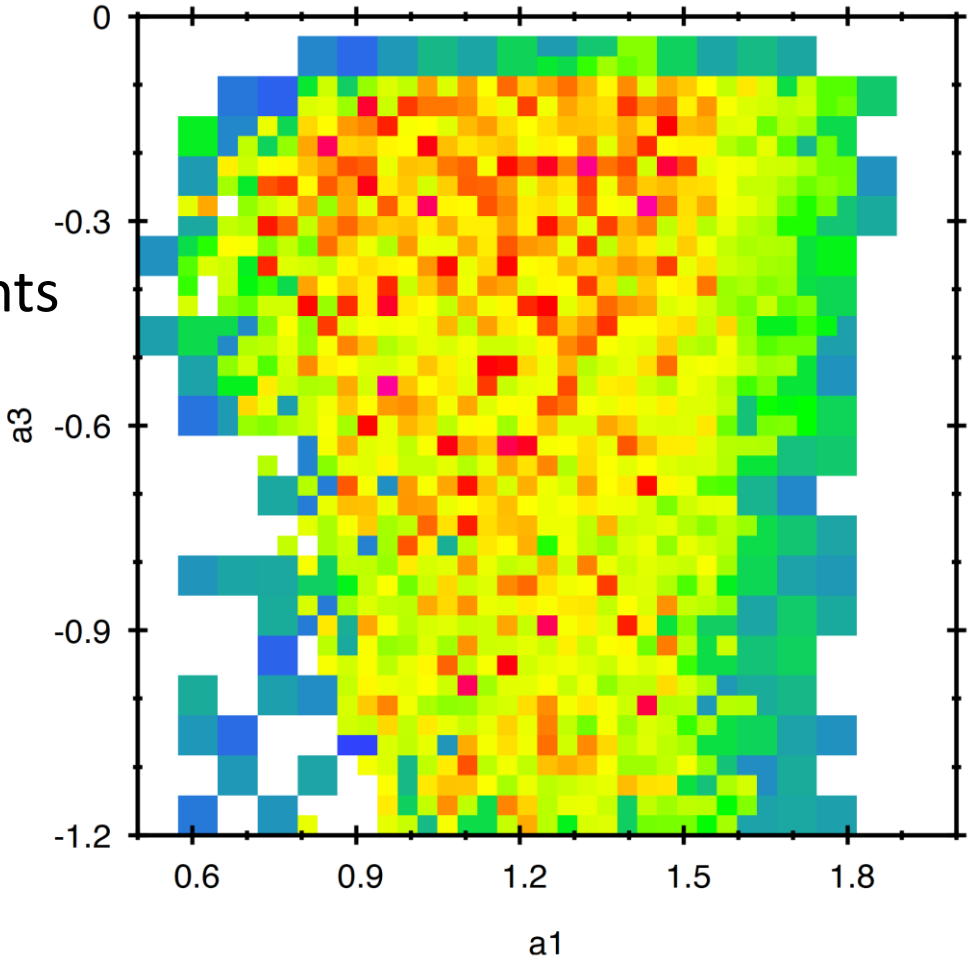
Numerical Optimization

- 7-dim. optimization problem, many local minima...
- We used an adaptive grid approach
- Start with $21^7 \approx 1.8 \cdot 10^9$ grid points
- Pick best 5000 points, run refined grid *(factor 2)*
- Continue until convergence



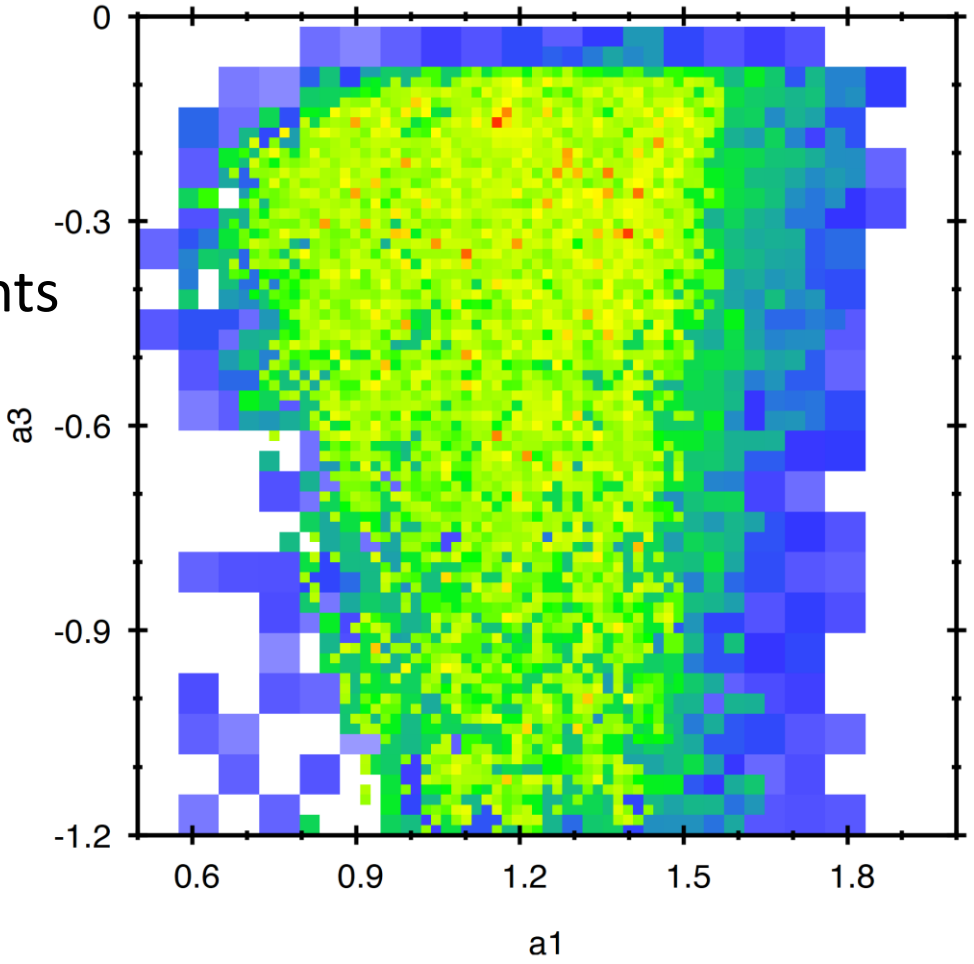
Numerical Optimization

- 7-dim. optimization problem, many local minima...
- We used an adaptive grid approach
- Start with $21^7 \approx 1.8 \cdot 10^9$ grid points
- Pick best 5000 points, run refined grid *(factor 2)*
- Continue until convergence



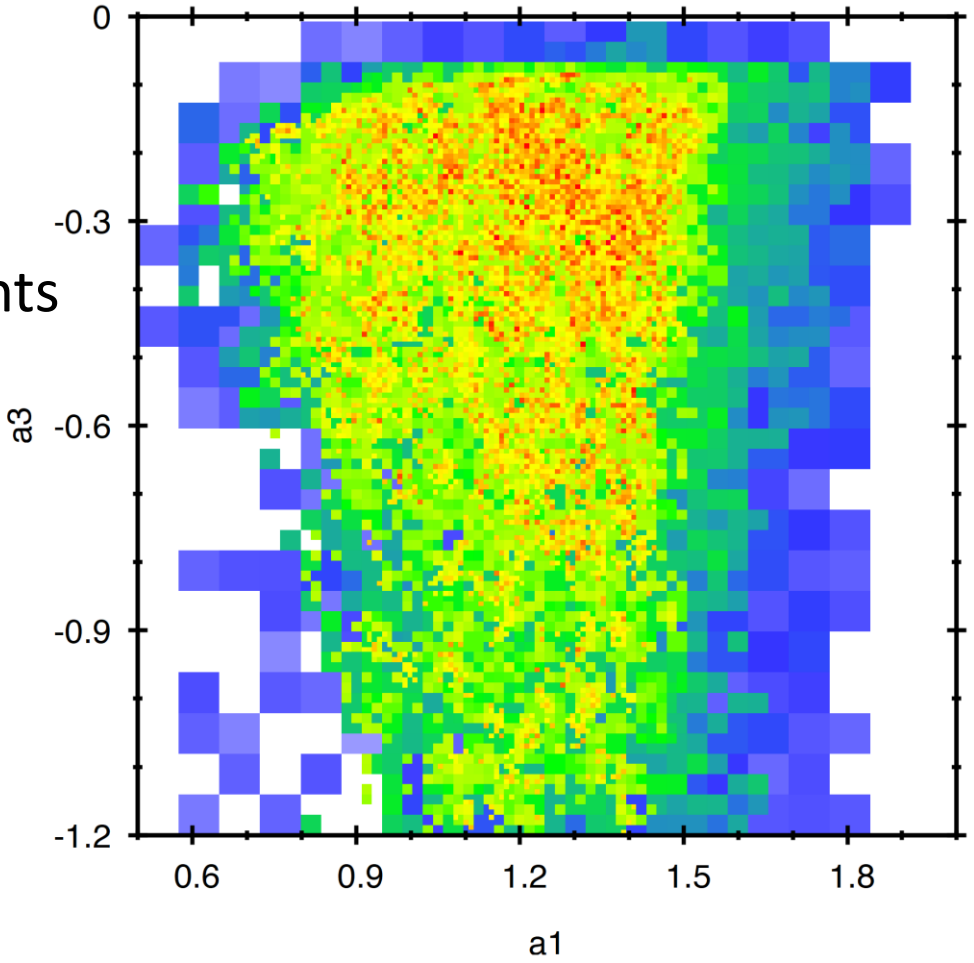
Numerical Optimization

- 7-dim. optimization problem, many local minima...
- We used an adaptive grid approach
- Start with $21^7 \approx 1.8 \cdot 10^9$ grid points
- Pick best 5000 points, run refined grid *(factor 2)*
- Continue until convergence



Numerical Optimization

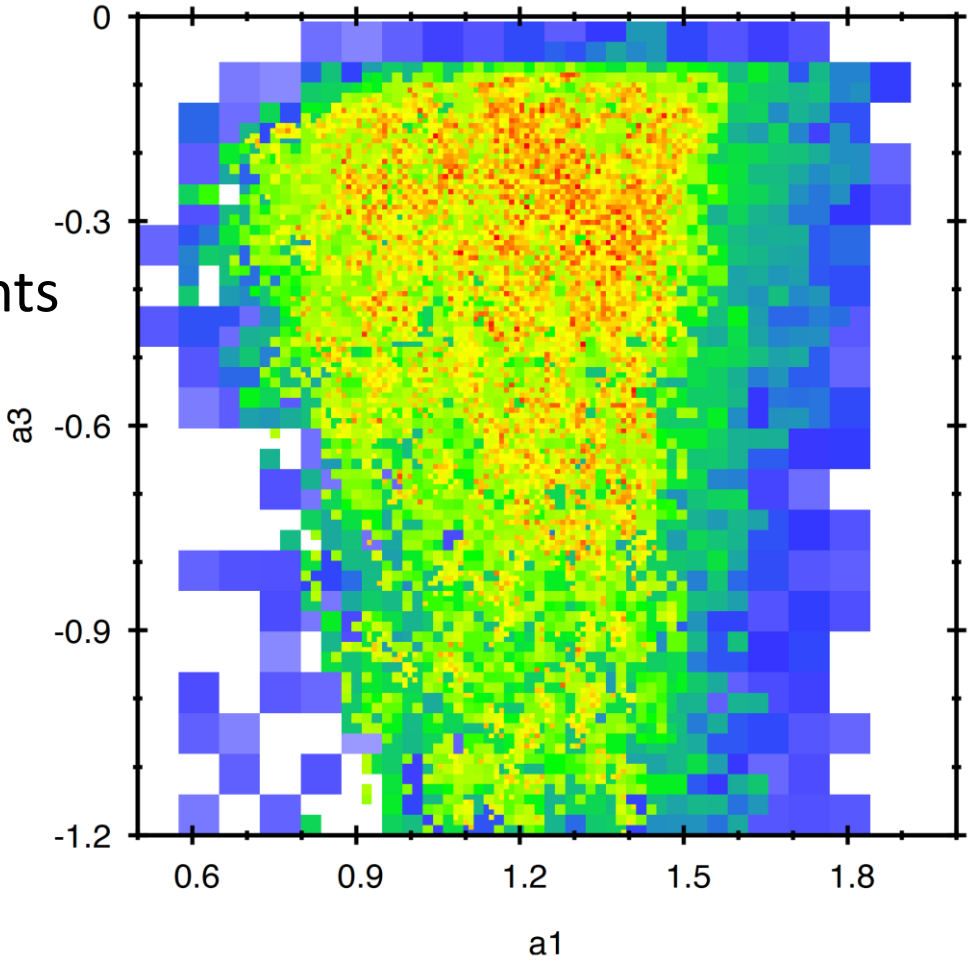
- 7-dim. optimization problem, many local minima...
- We used an adaptive grid approach
- Start with $21^7 \approx 1.8 \cdot 10^9$ grid points
- Pick best 5000 points, run refined grid *(factor 2)*
- Continue until convergence



Numerical Optimization

- 7-dim. optimization problem, many local minima...
- We used an adaptive grid approach
- Start with $21^7 \approx 1.8 \cdot 10^9$ grid points
- Pick best 5000 points, run refined grid *(factor 2)*
- Continue until convergence
- Each grid point: One integration method
- Run 200 timestep lengths and 100 initial configurations

→ 20‘000 MD runs per point



Numerical Optimization

$1.8 \cdot 10^9$ grid points x 200 timestep lengths x 100 initial configs.

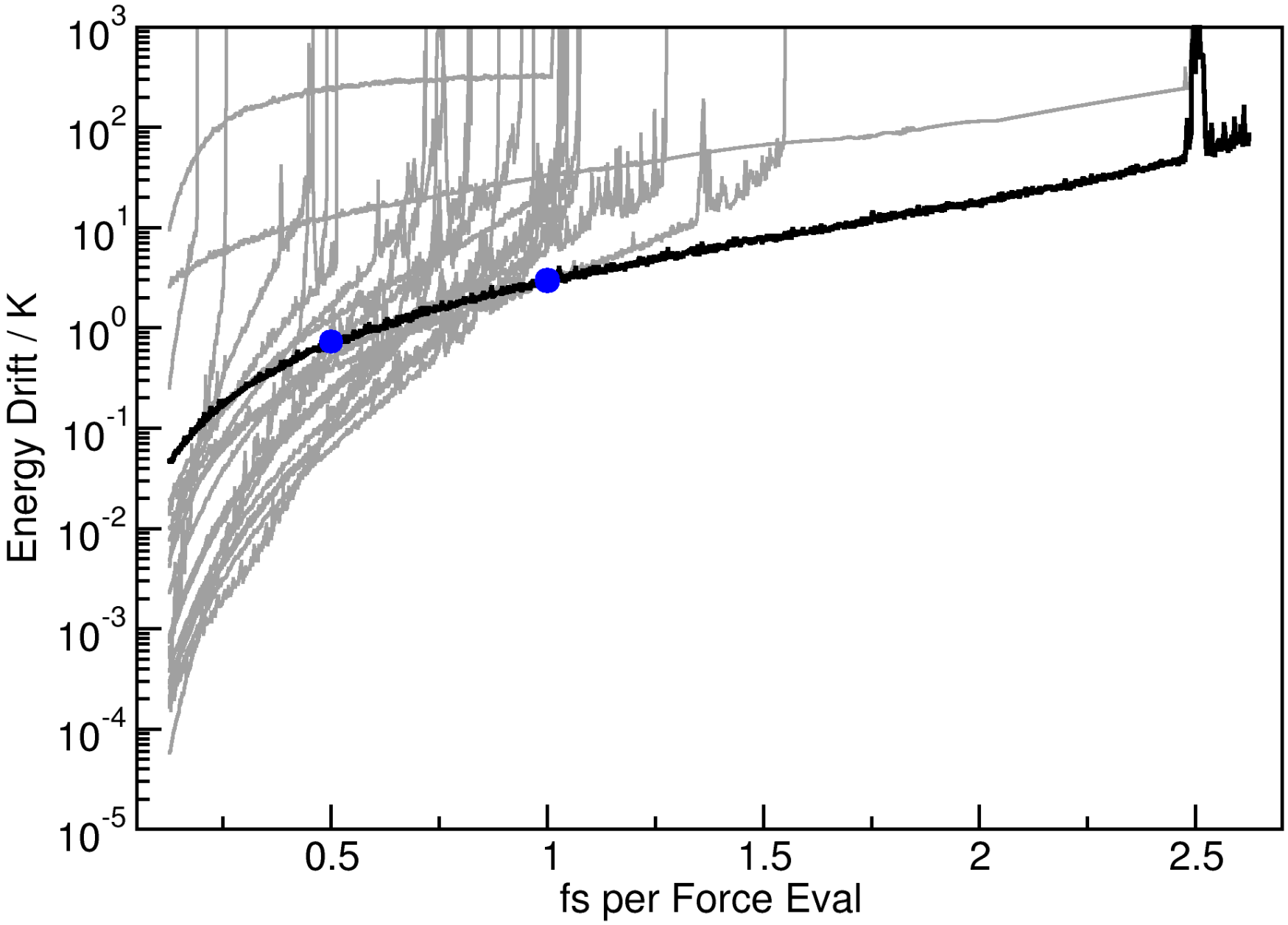
$= 3.6 \cdot 10^9$ MD runs *(6 ps each)* → 200 seconds (!) of total MD time

Not possible. Tricks needed...

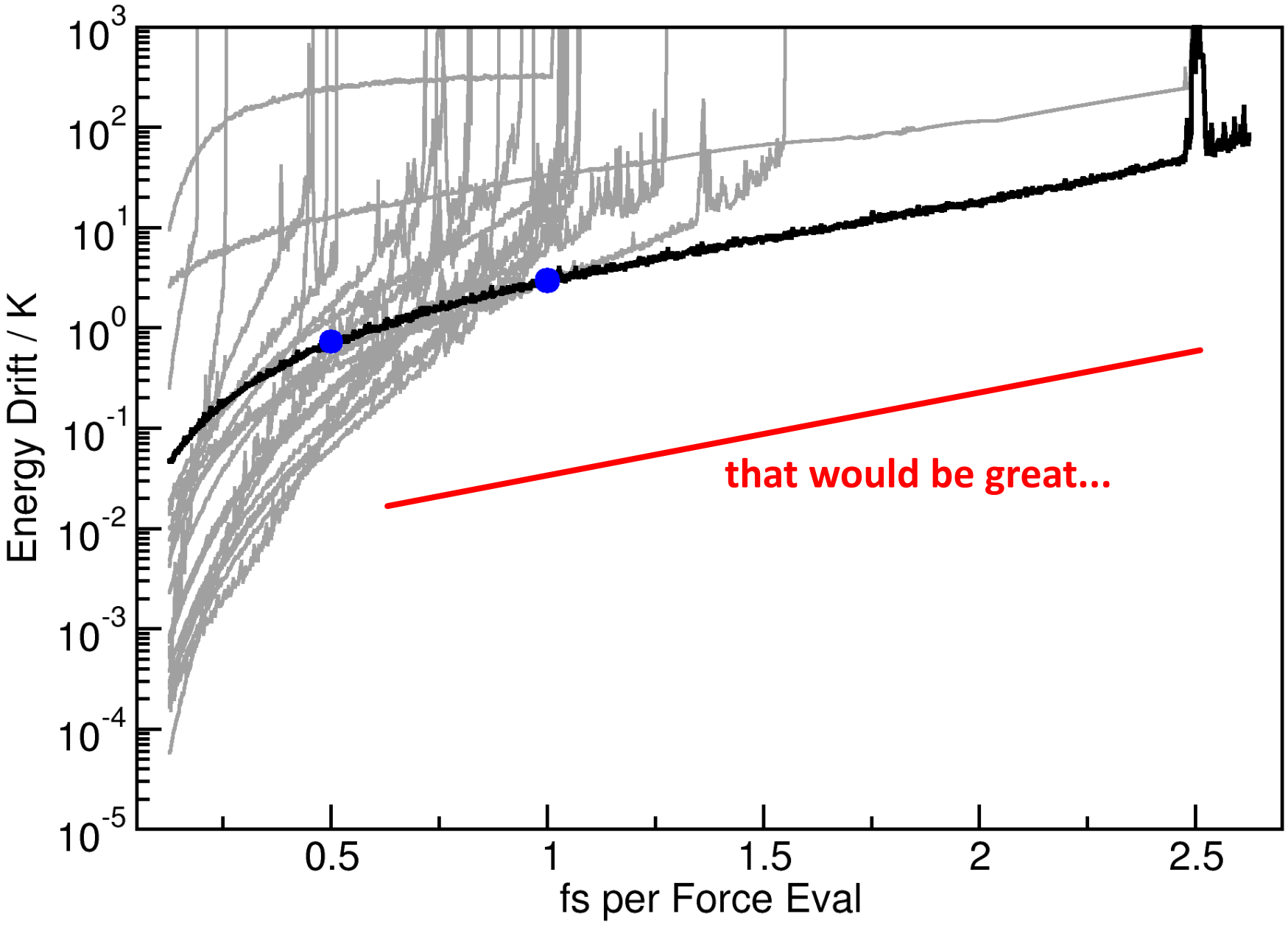
Abort each run if energy deviation becomes larger than some *(dynamically adjusted)* threshold

We used around 7 million CPU core hours for this project

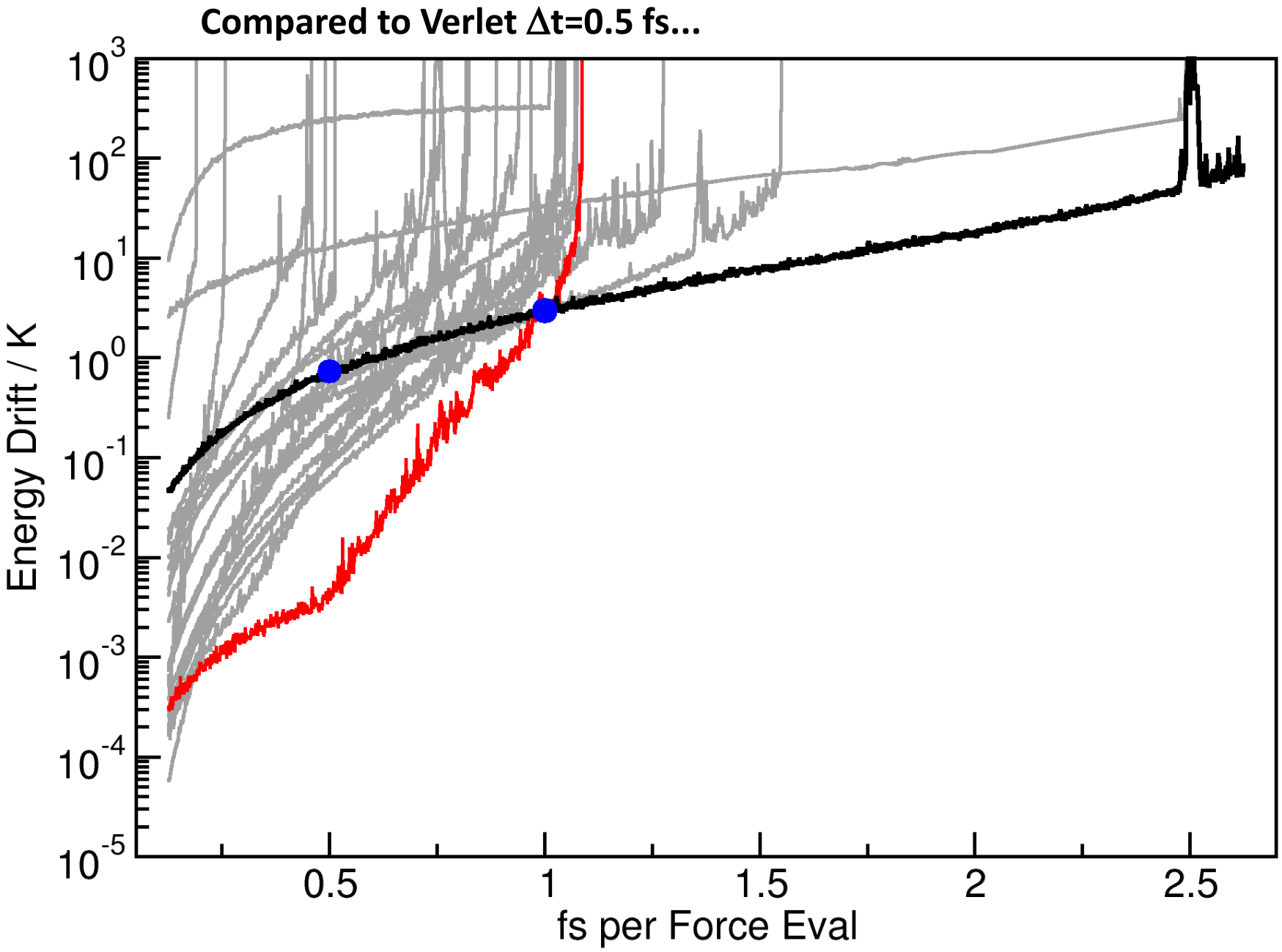
Practical MD Performance – Results



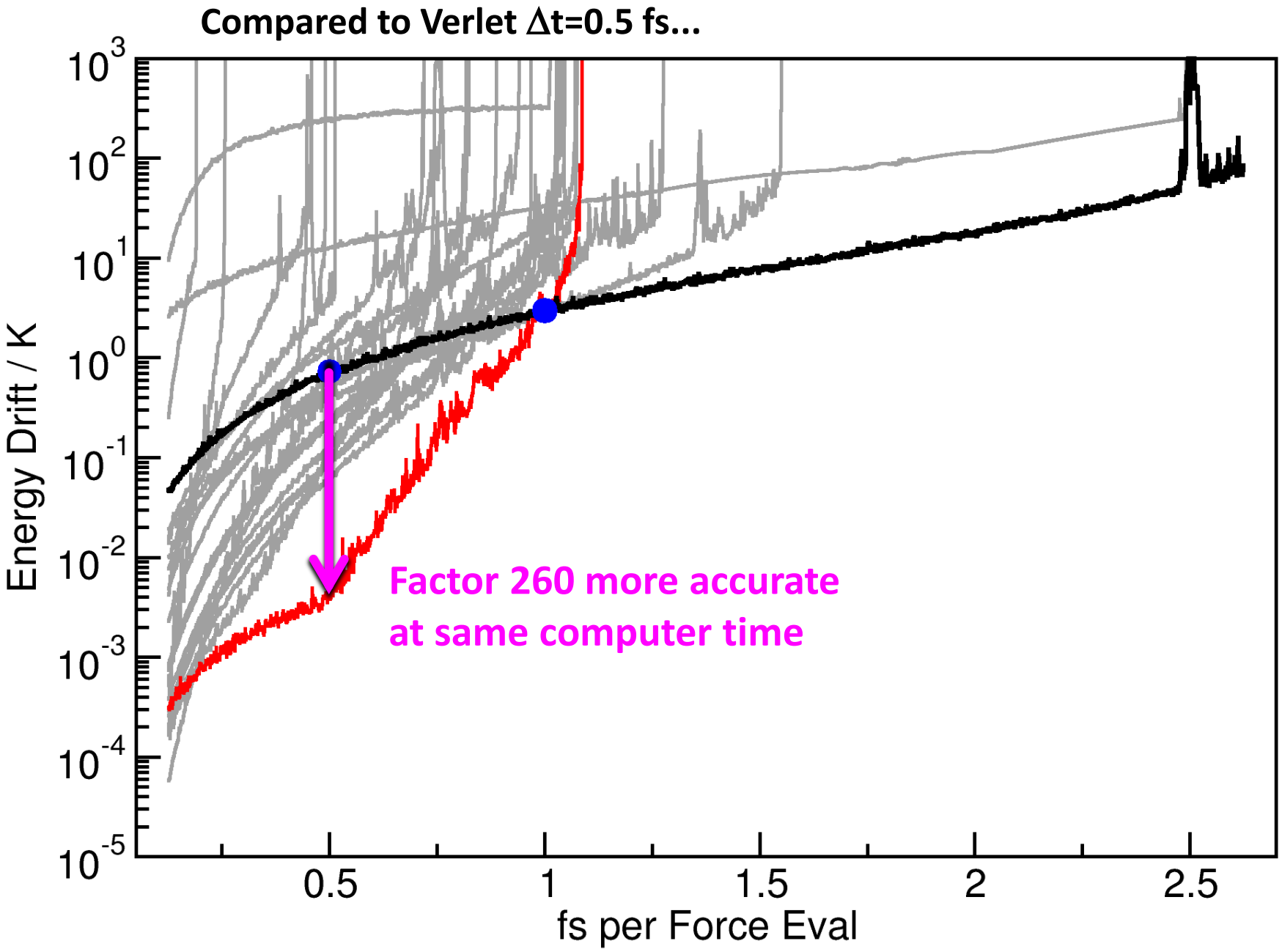
Practical MD Performance – Results



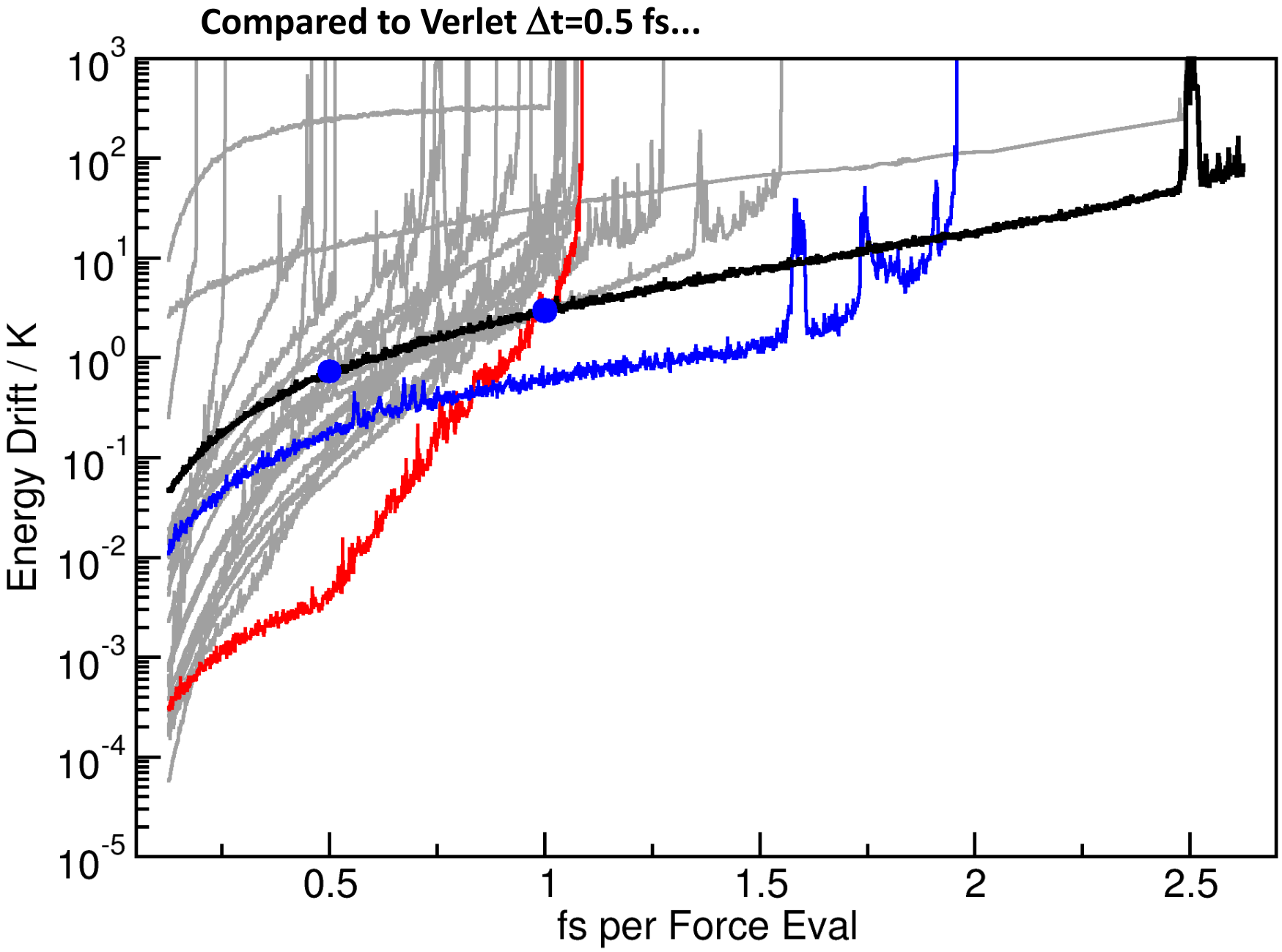
Practical MD Performance – Results



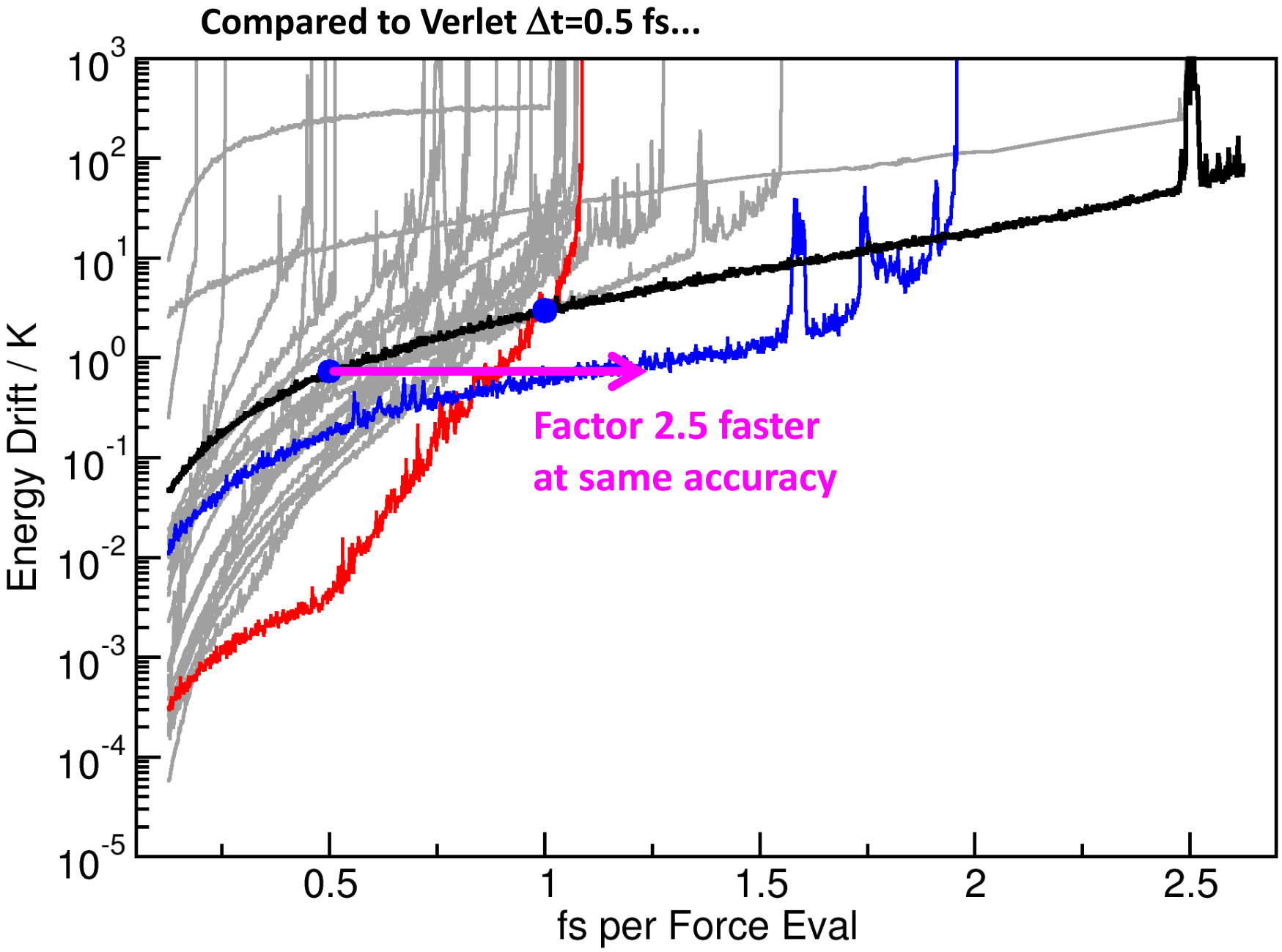
Practical MD Performance – Results



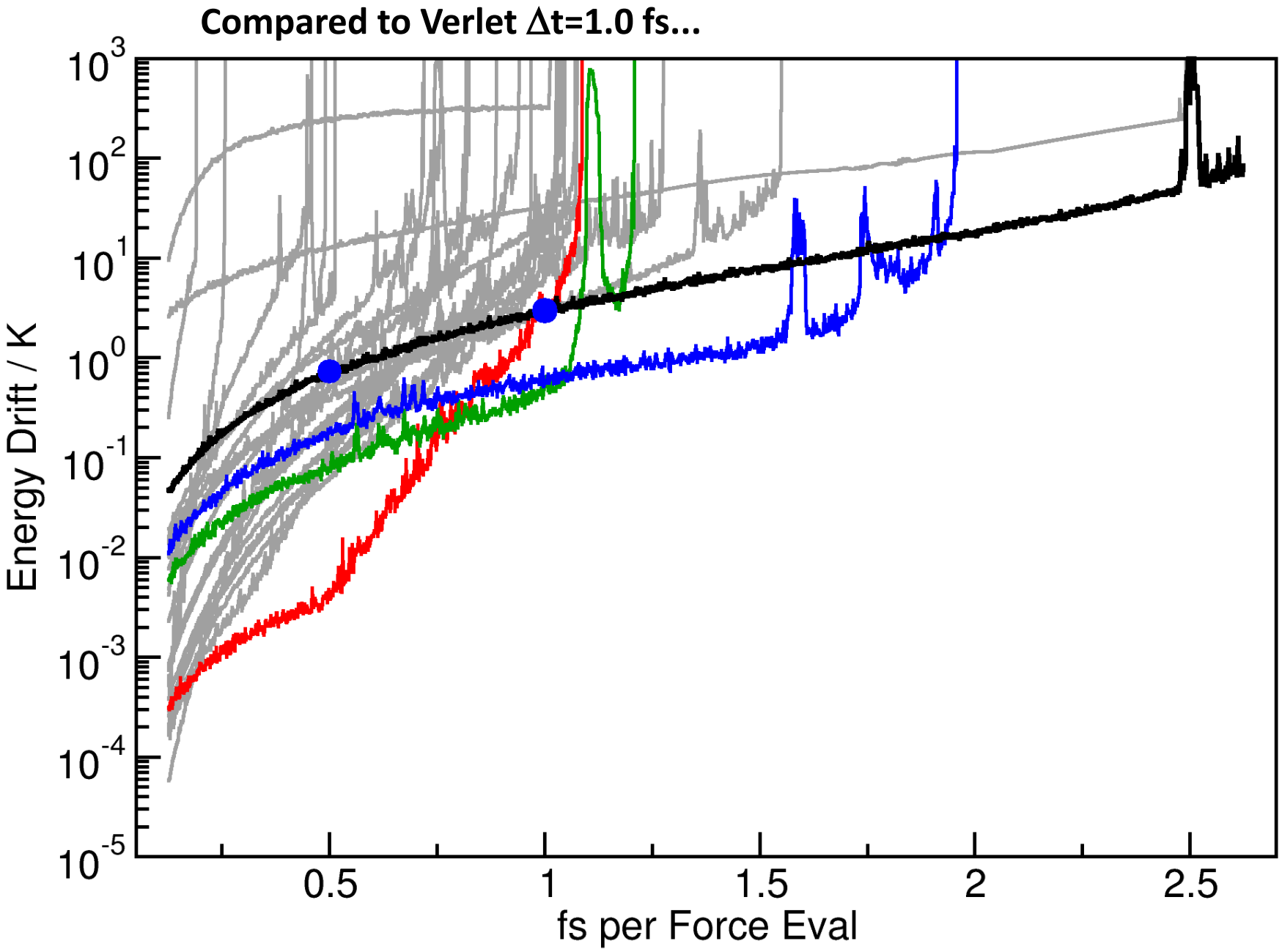
Practical MD Performance – Results



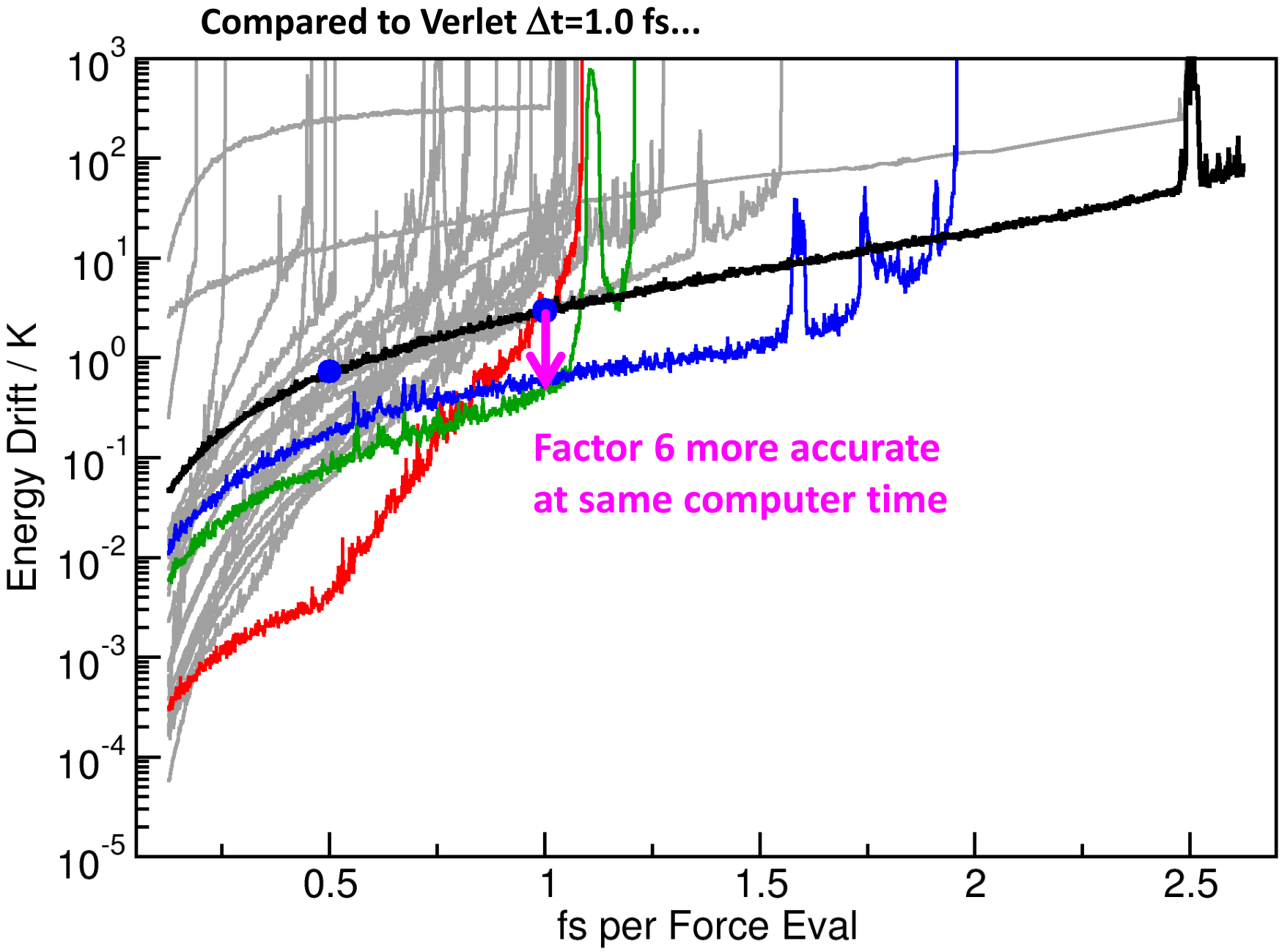
Practical MD Performance – Results



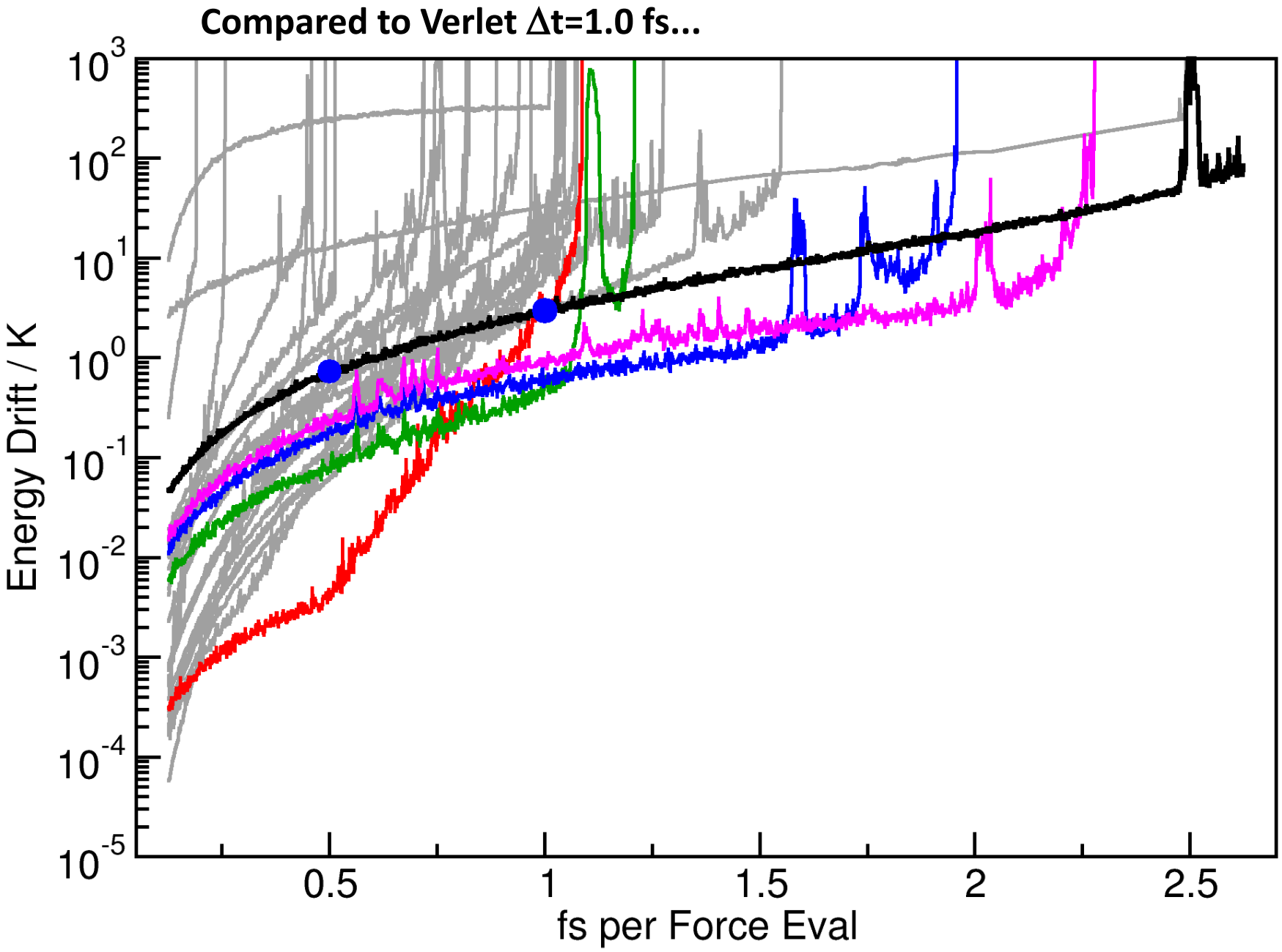
Practical MD Performance – Results



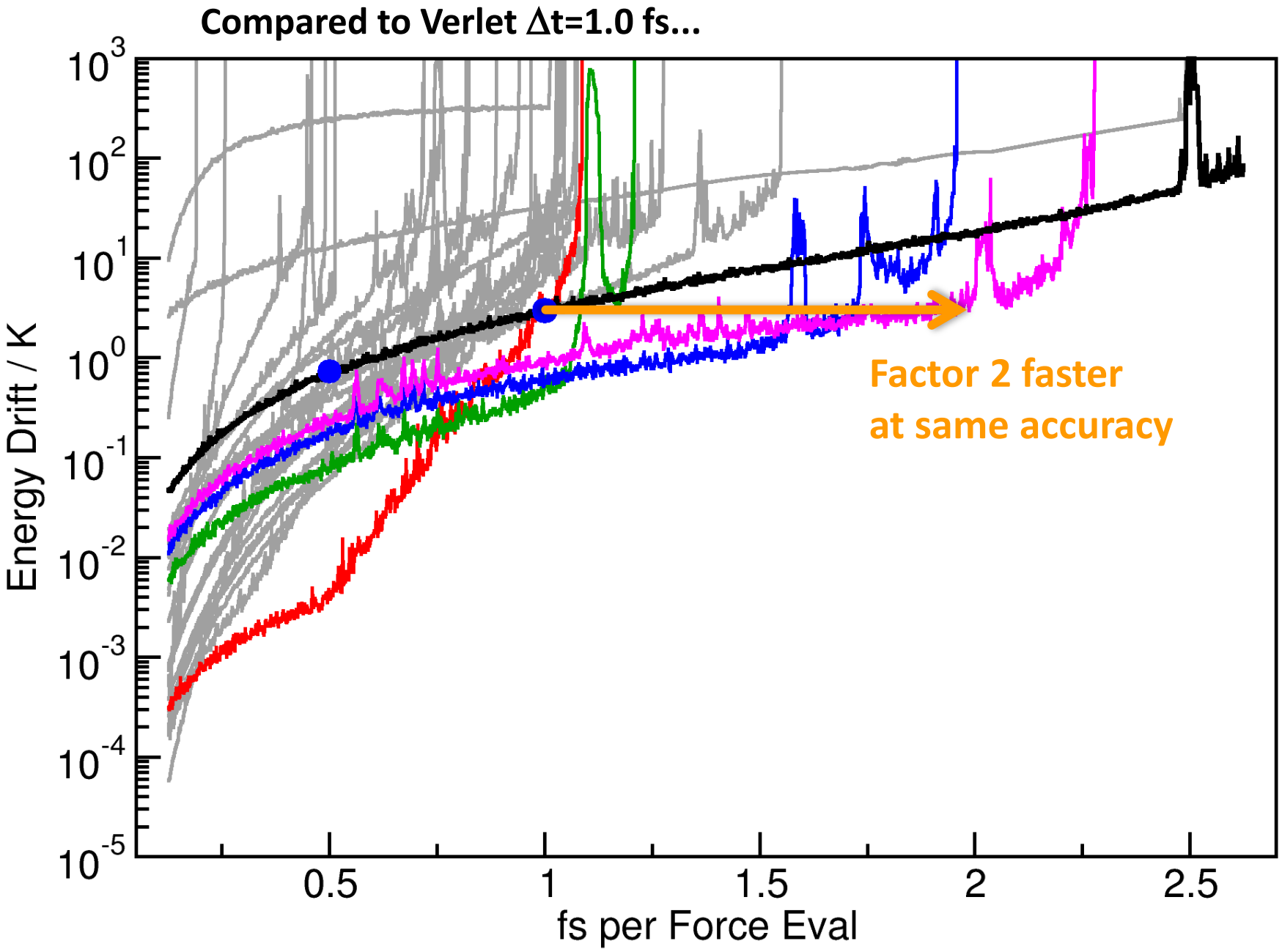
Practical MD Performance – Results



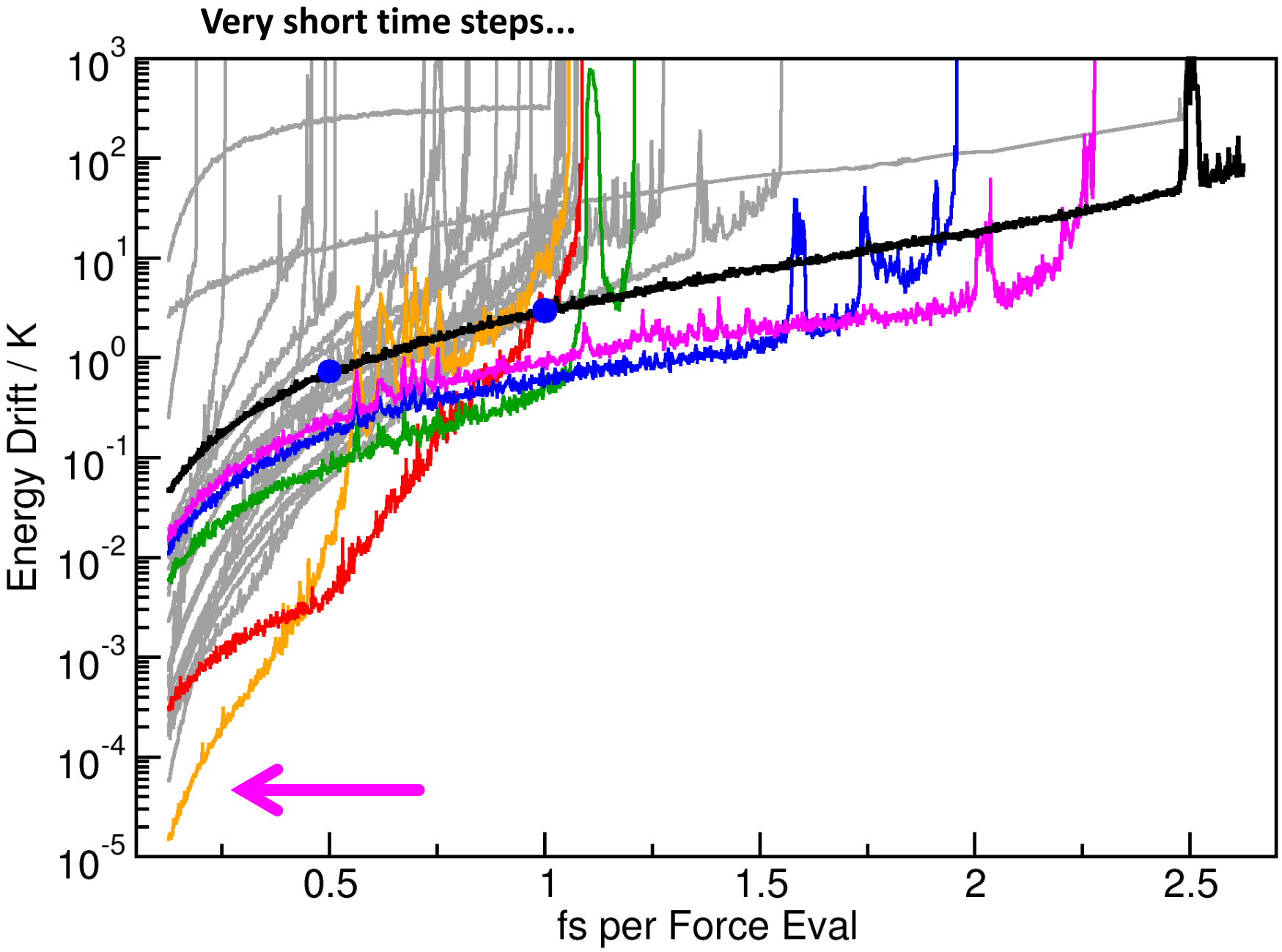
Practical MD Performance – Results



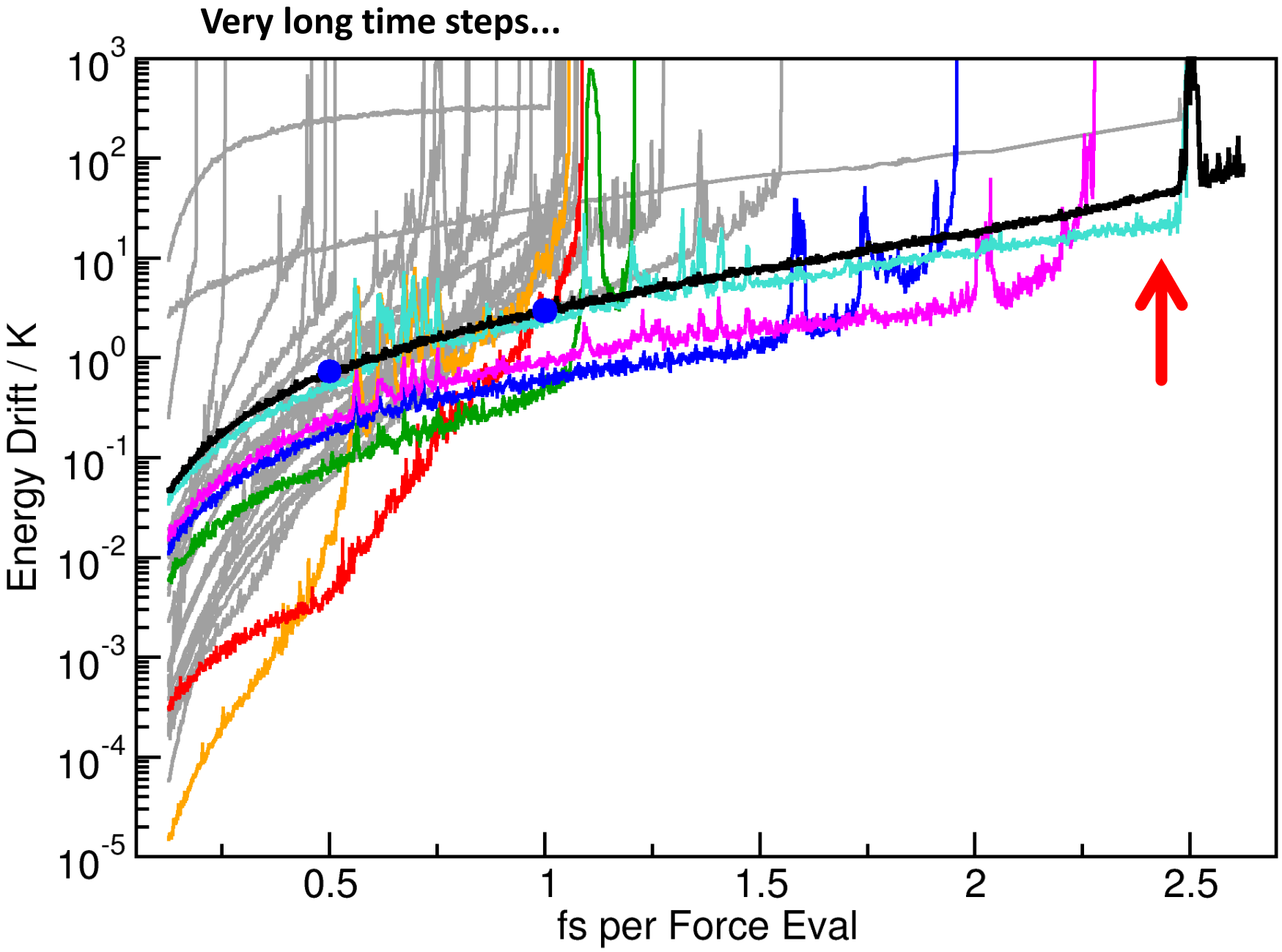
Practical MD Performance – Results



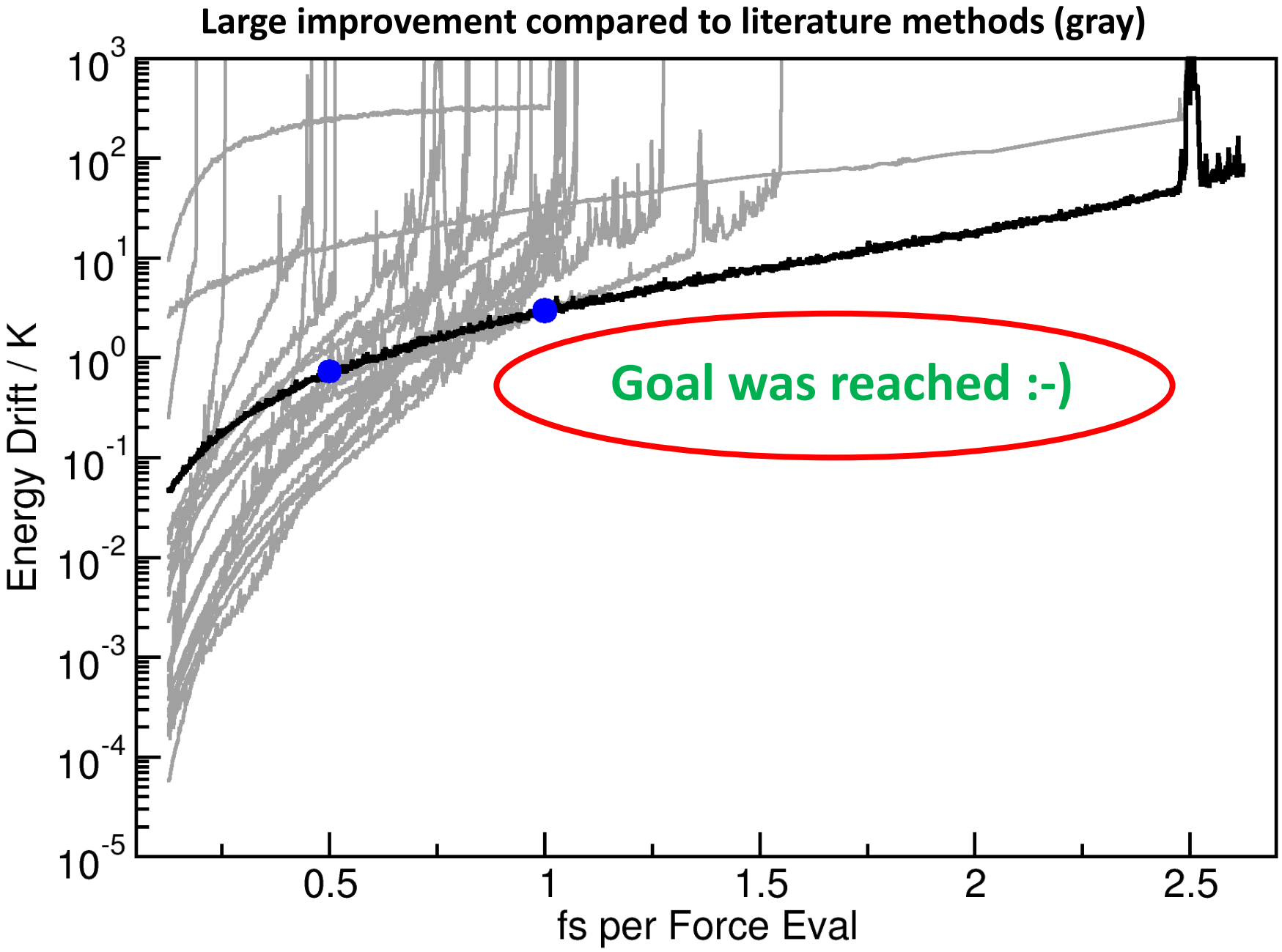
Practical MD Performance – Results



Practical MD Performance – Results

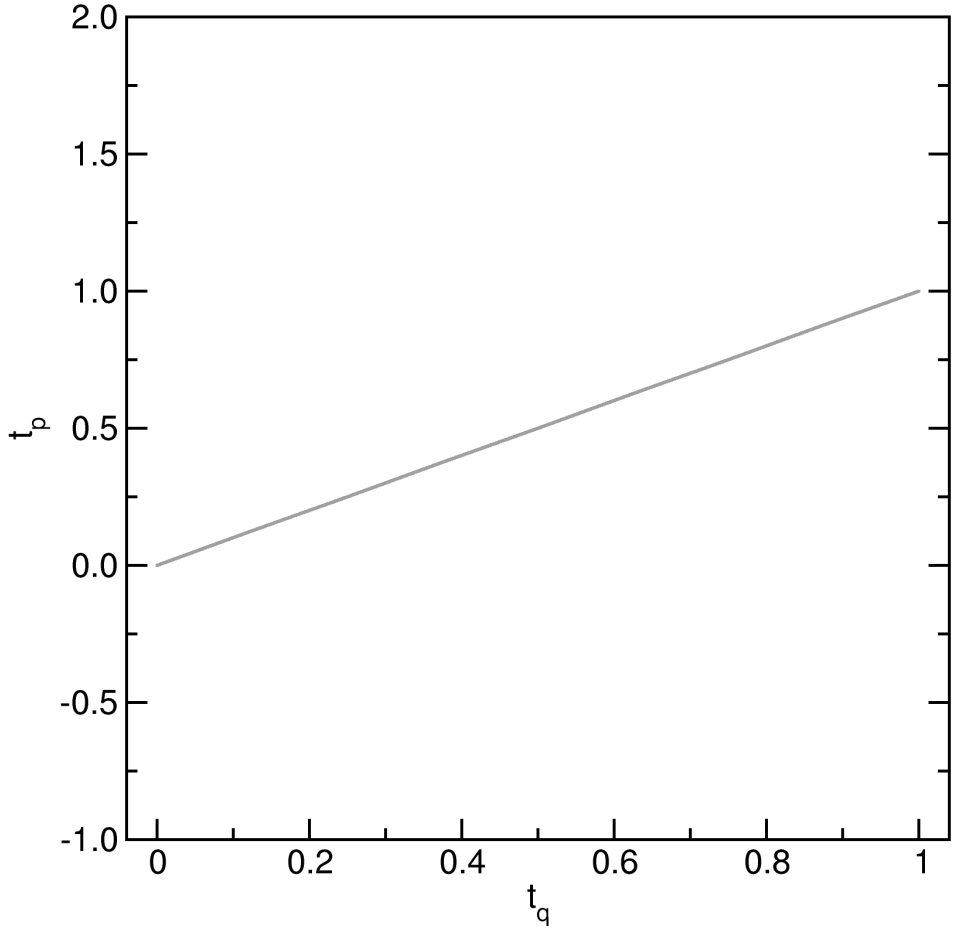


Practical MD Performance – Results

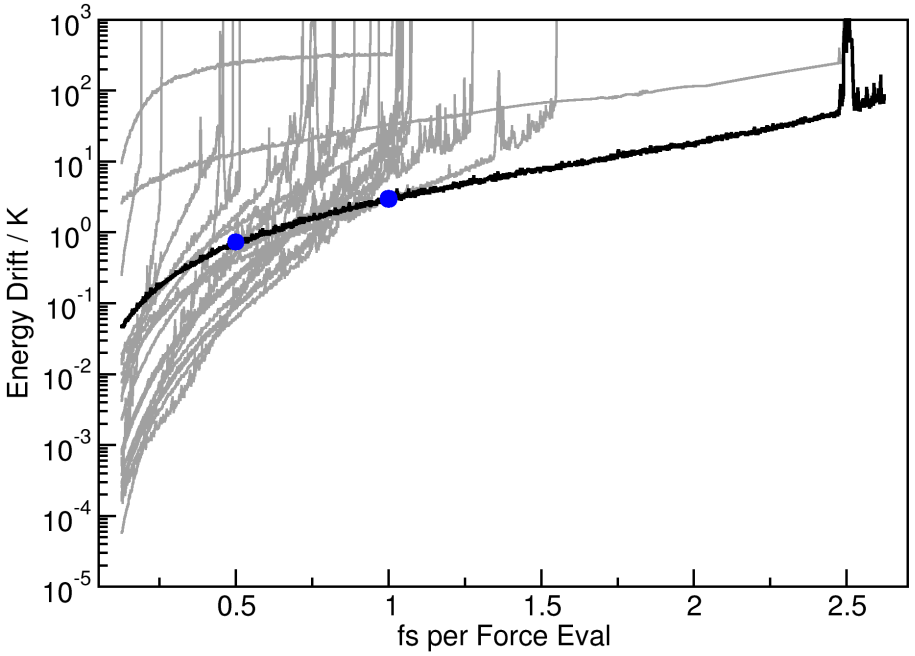


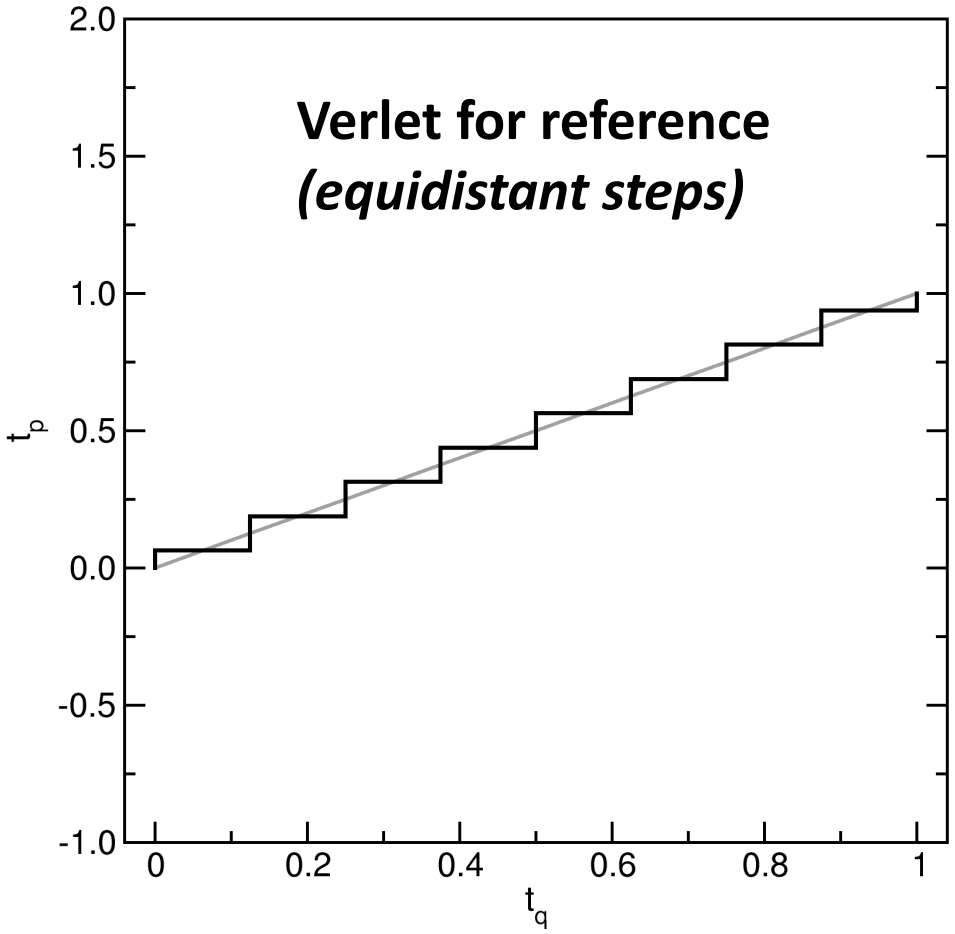
q–p Diagrams of new Methods

Do they look as crazy as Yoshida 8th order?

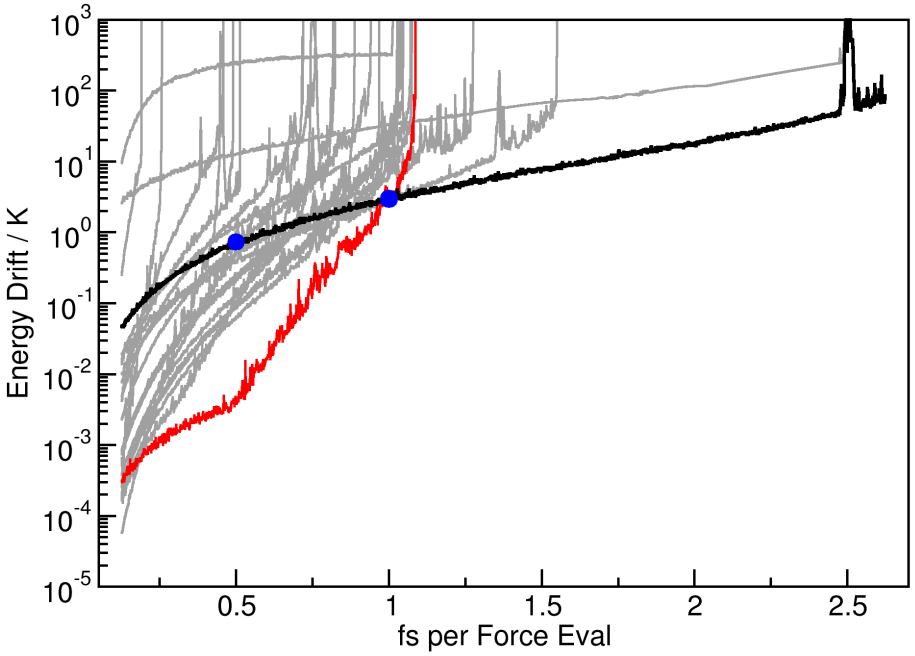


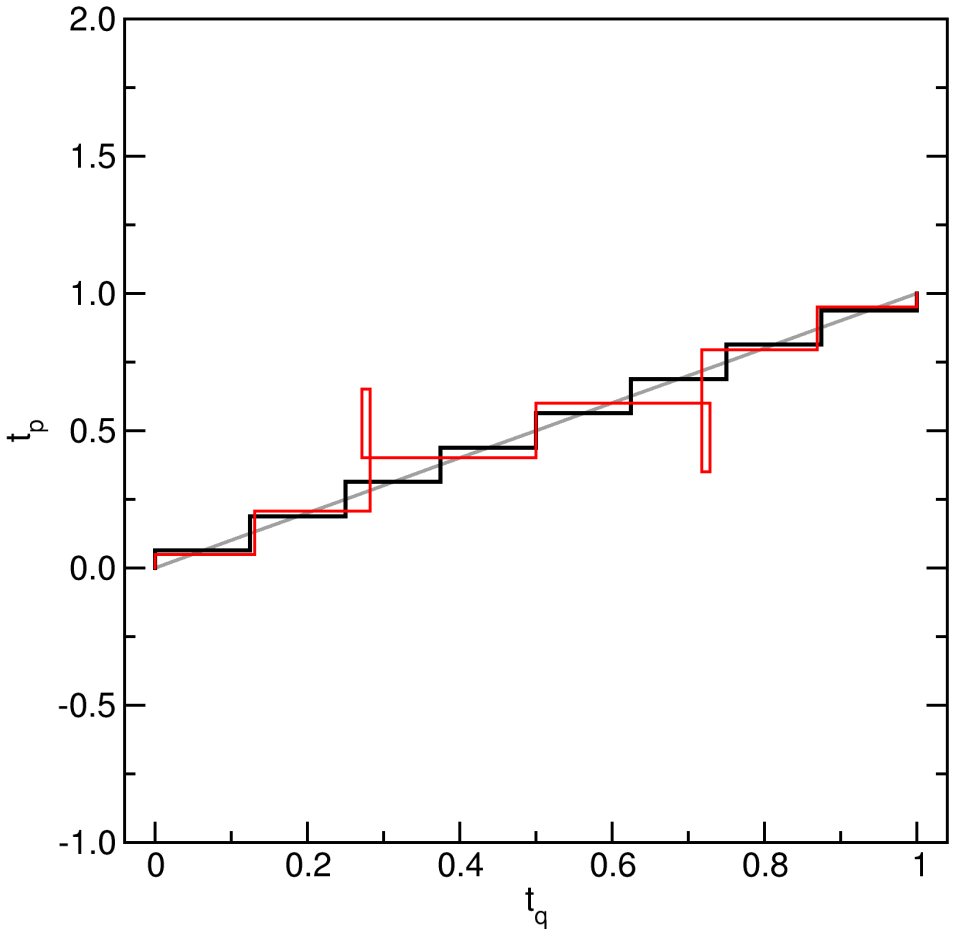
q–p Diagrams of new Methods



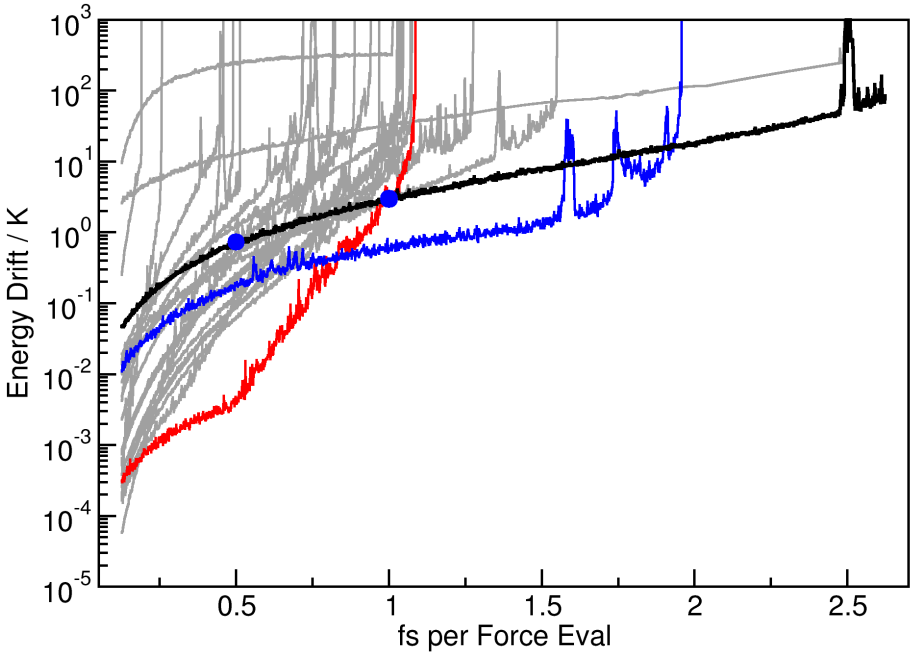


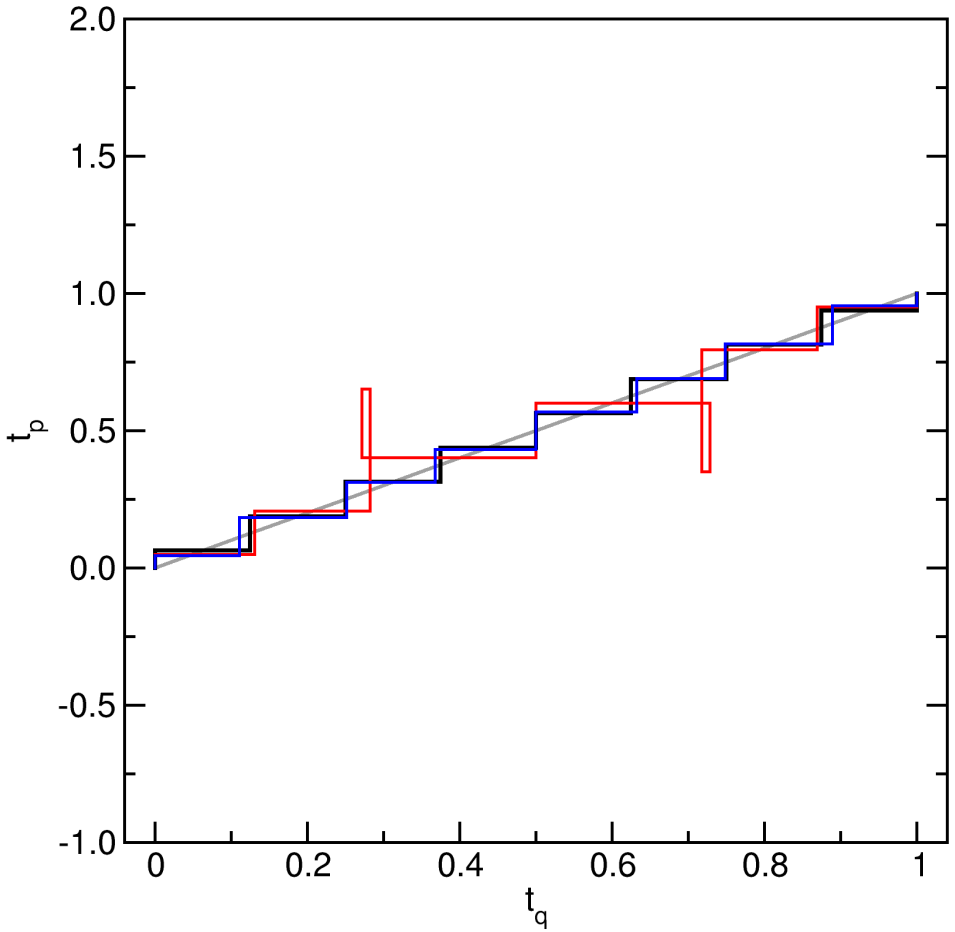
q–p Diagrams of new Methods



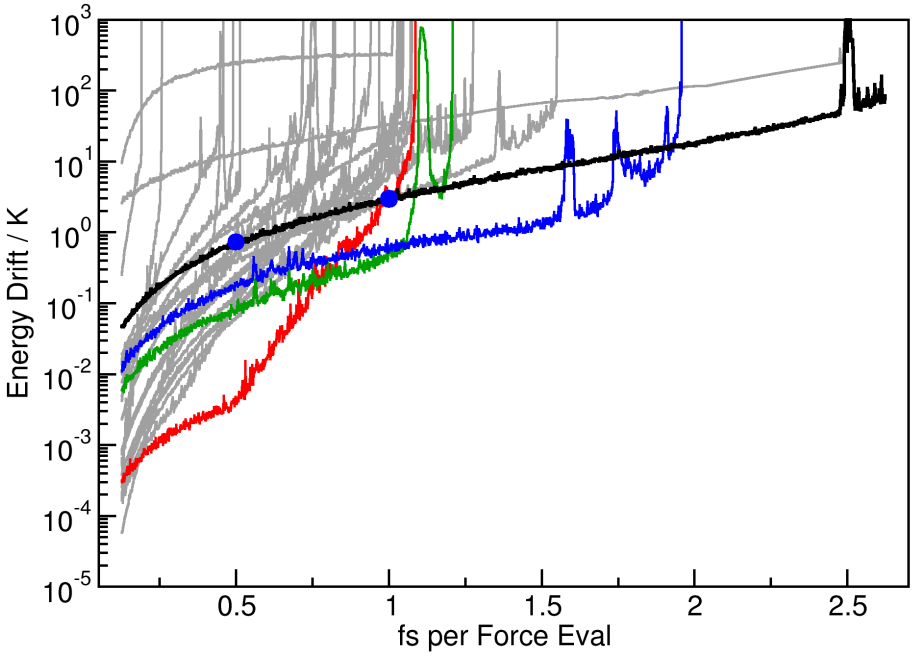


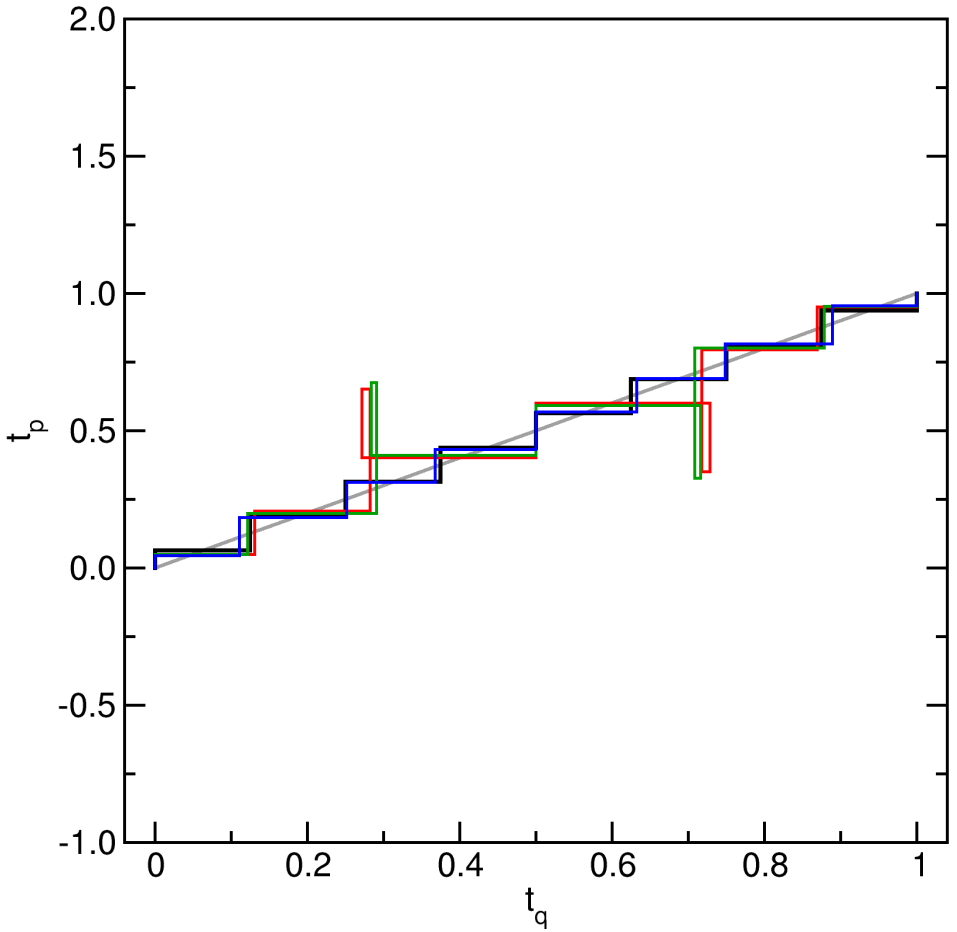
q–p Diagrams of new Methods



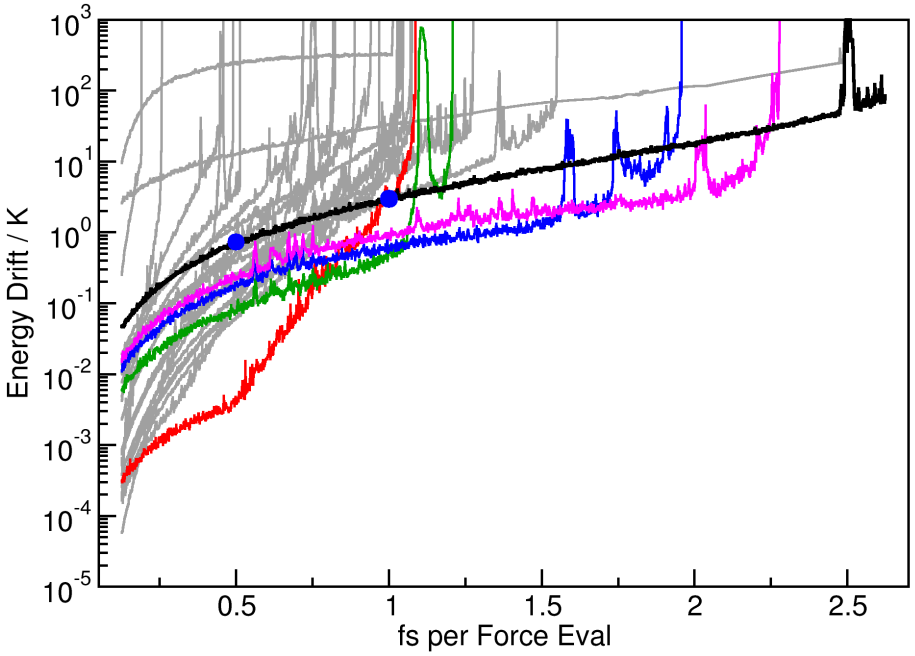


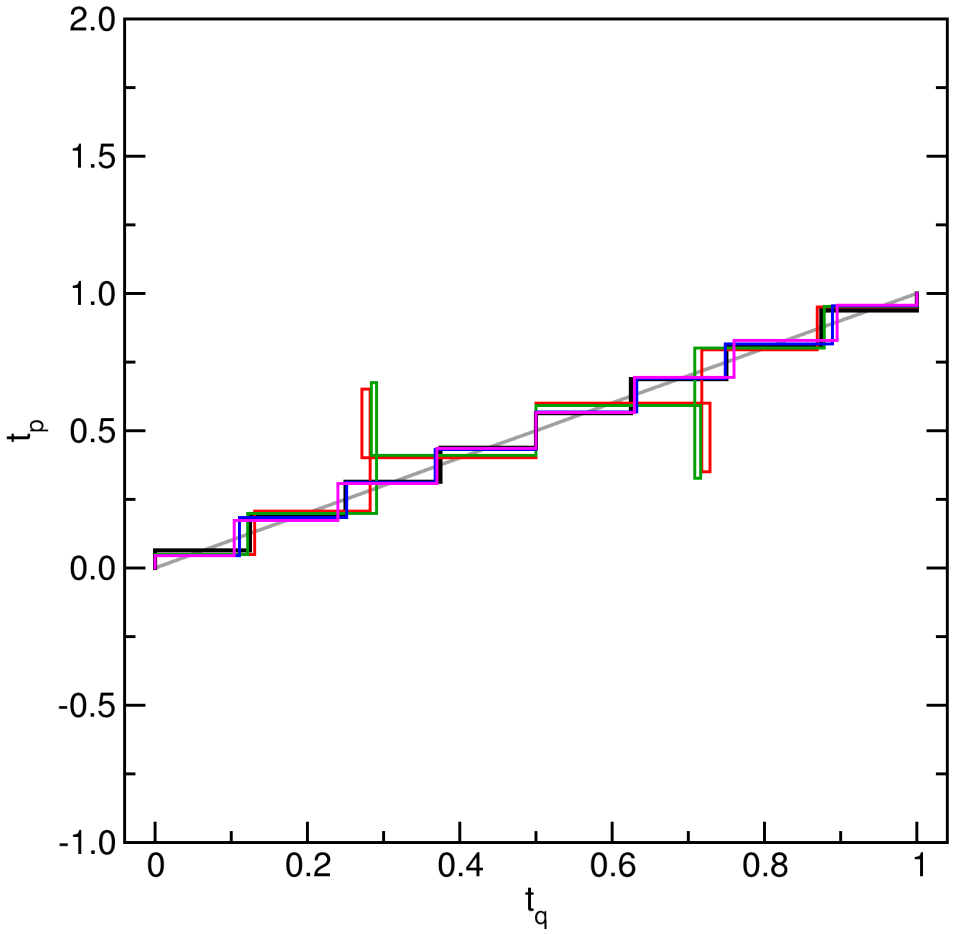
q–p Diagrams of new Methods



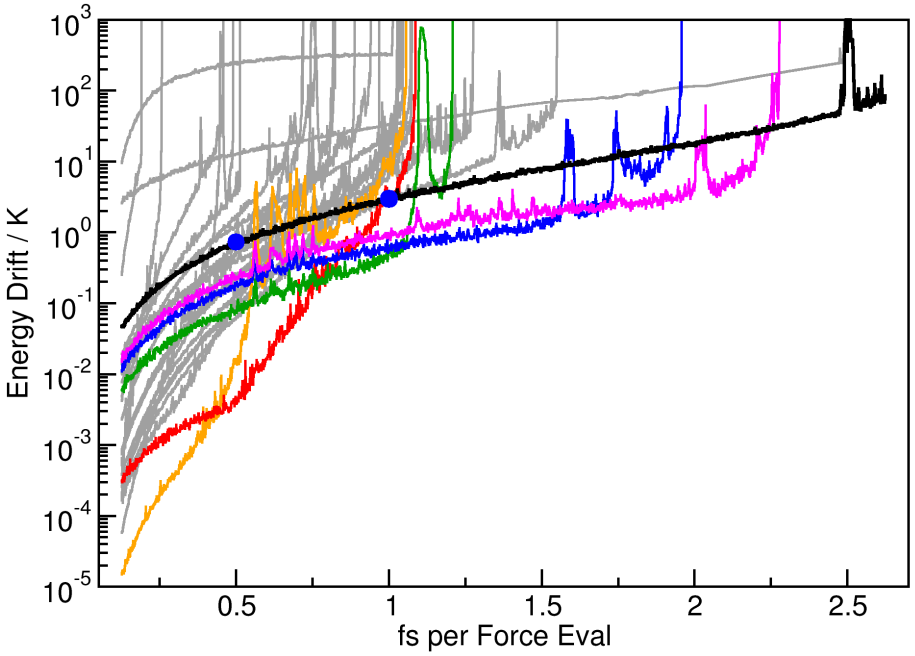


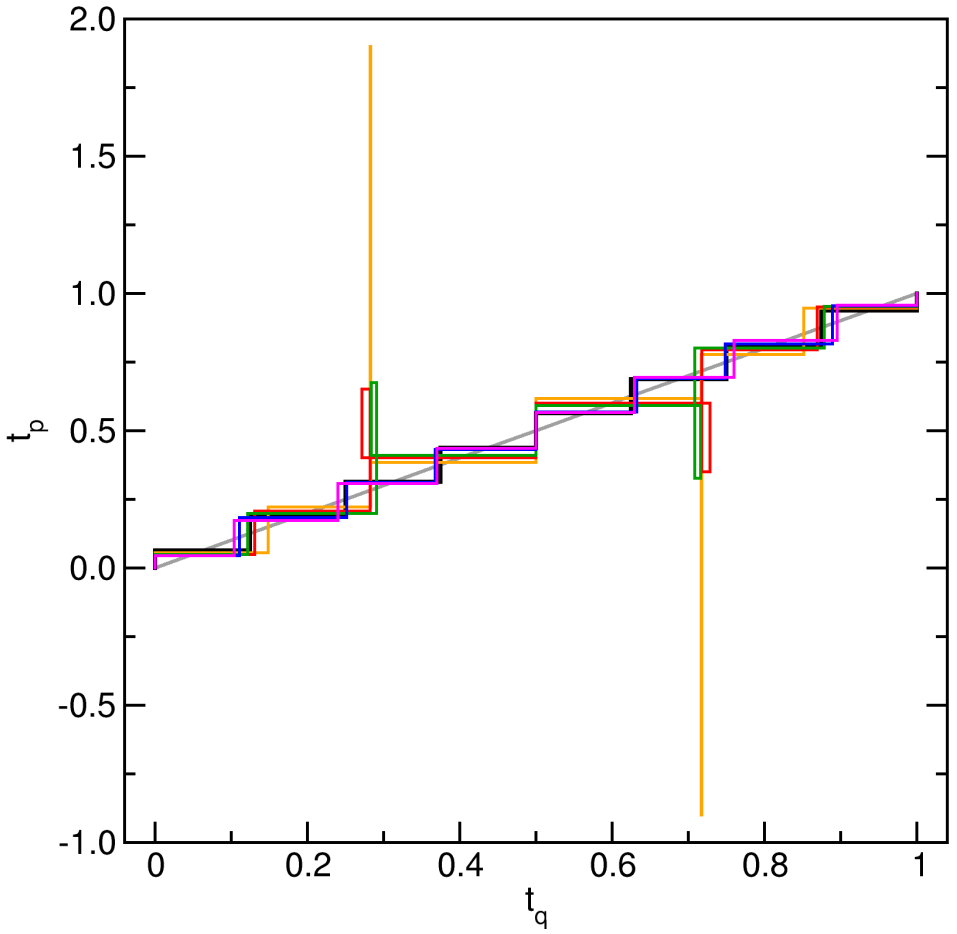
q–p Diagrams of new Methods



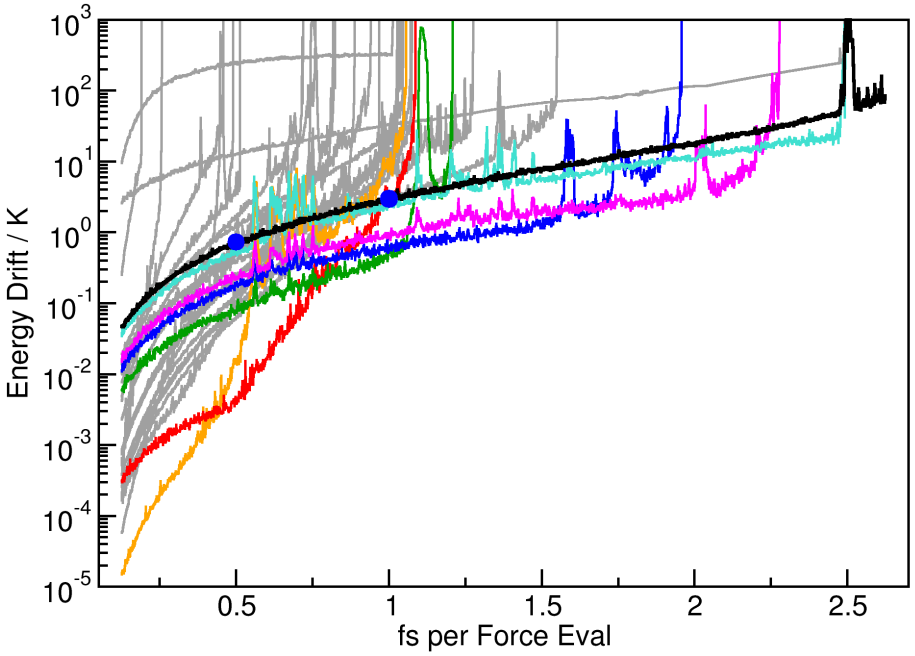


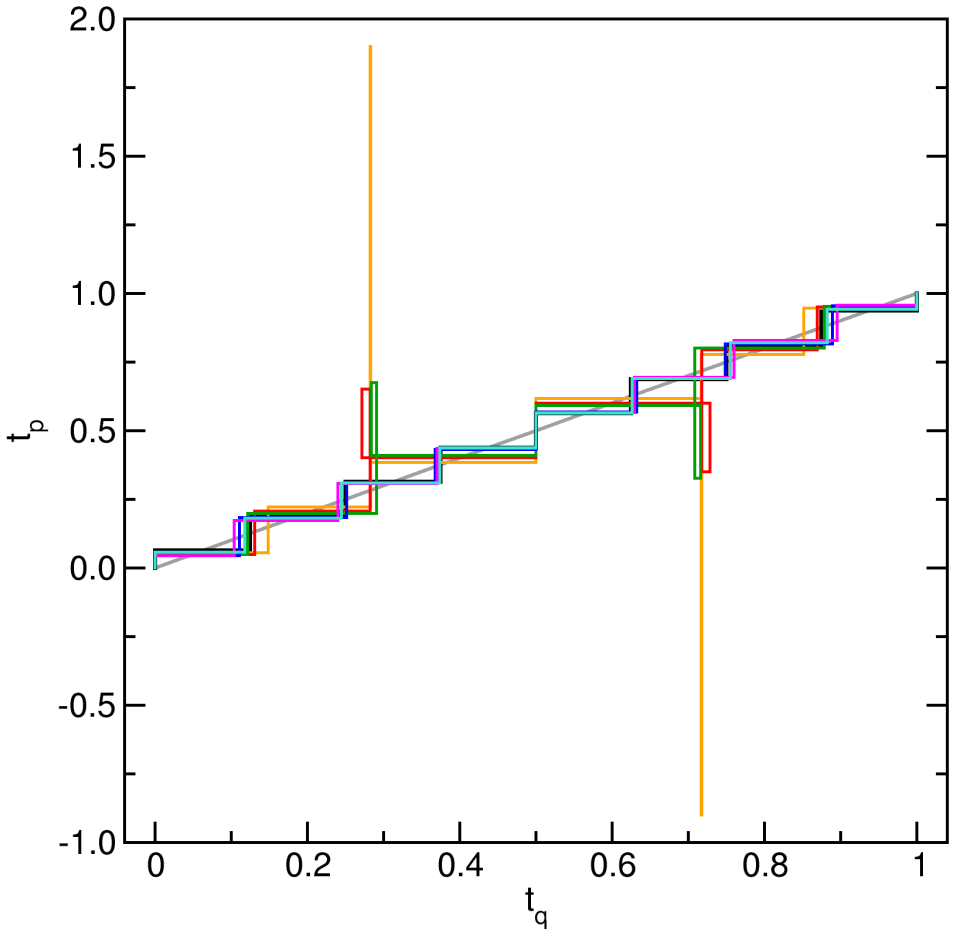
q–p Diagrams of new Methods





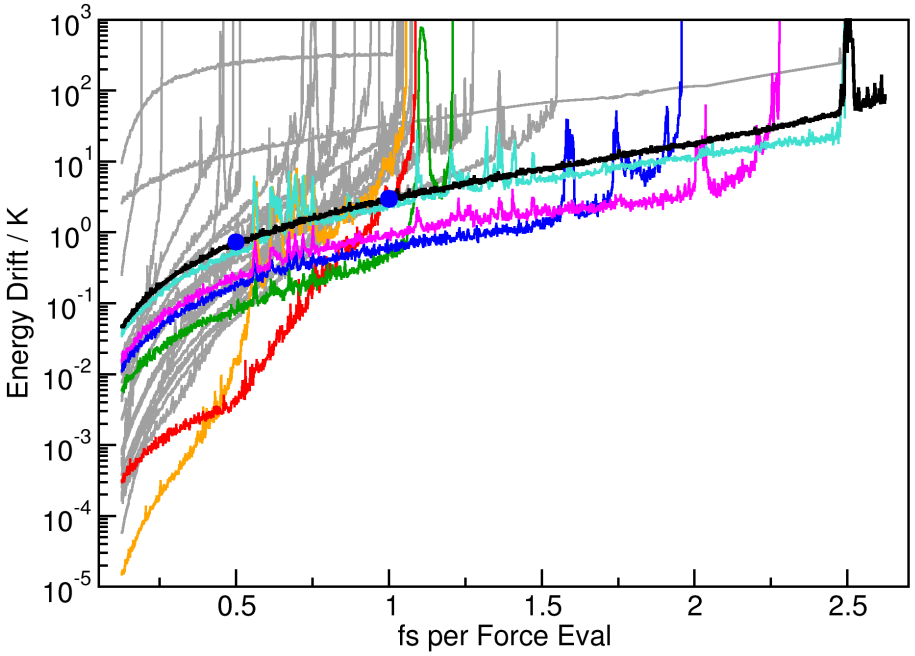
q–p Diagrams of new Methods

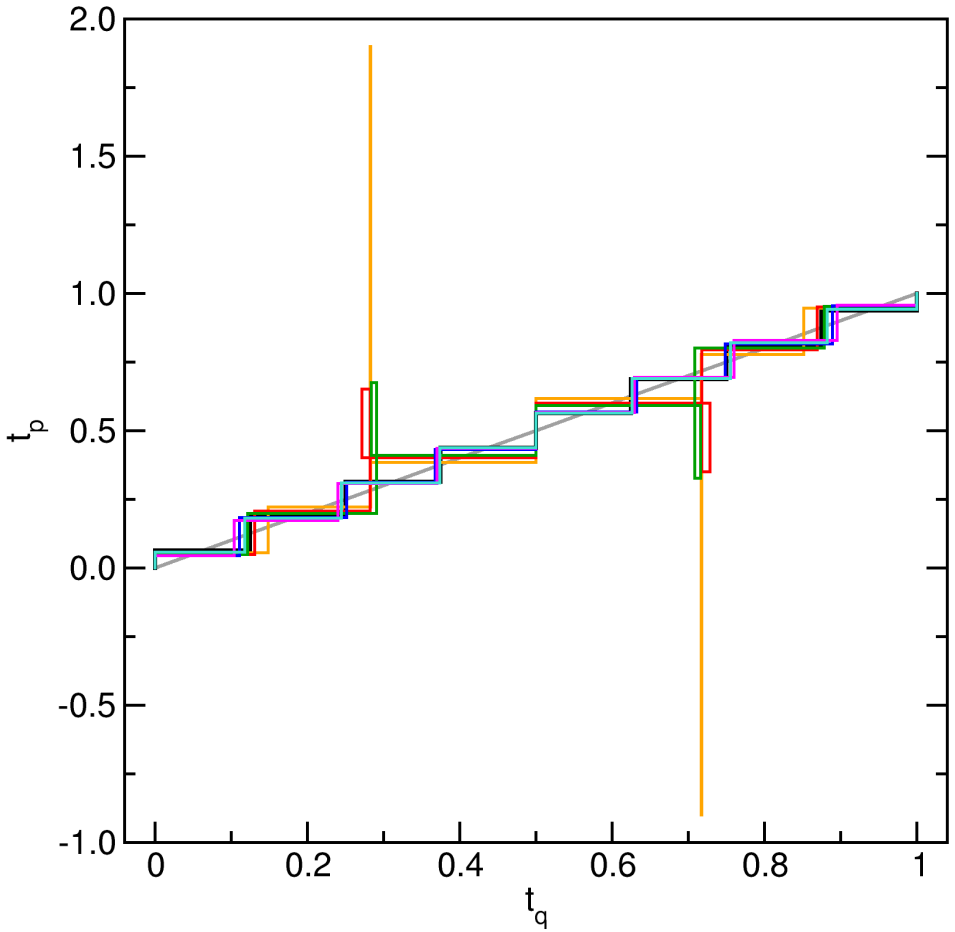




q–p Diagrams of new Methods

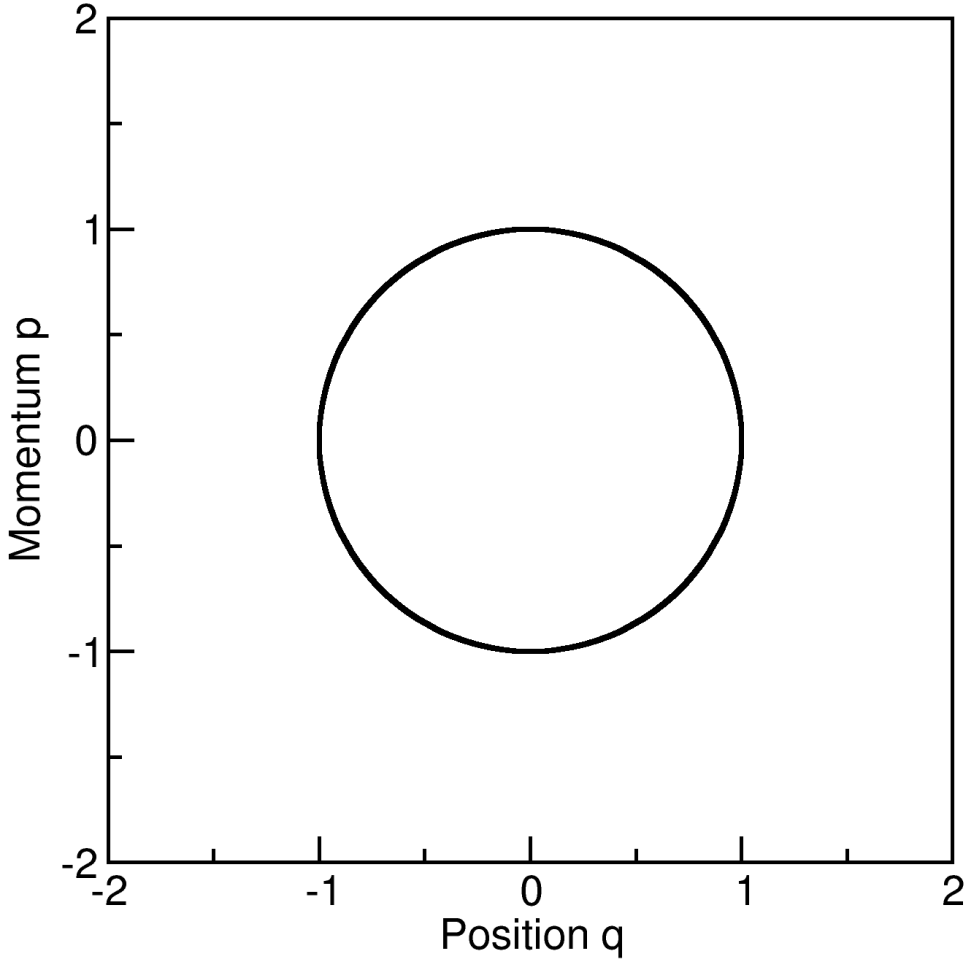
→ they look rather similar to Verlet
(much less „crazy“ than Yoshida etc.)

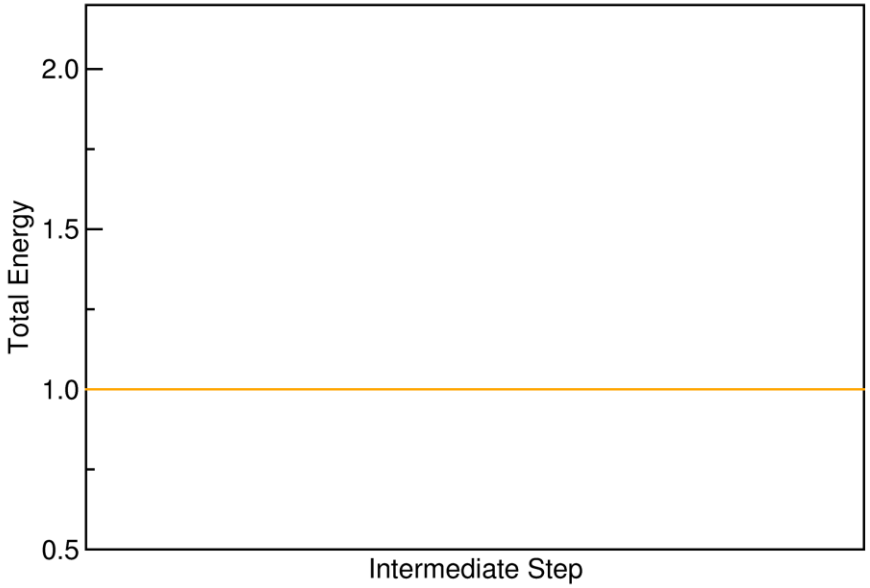




Phase Space Portraits

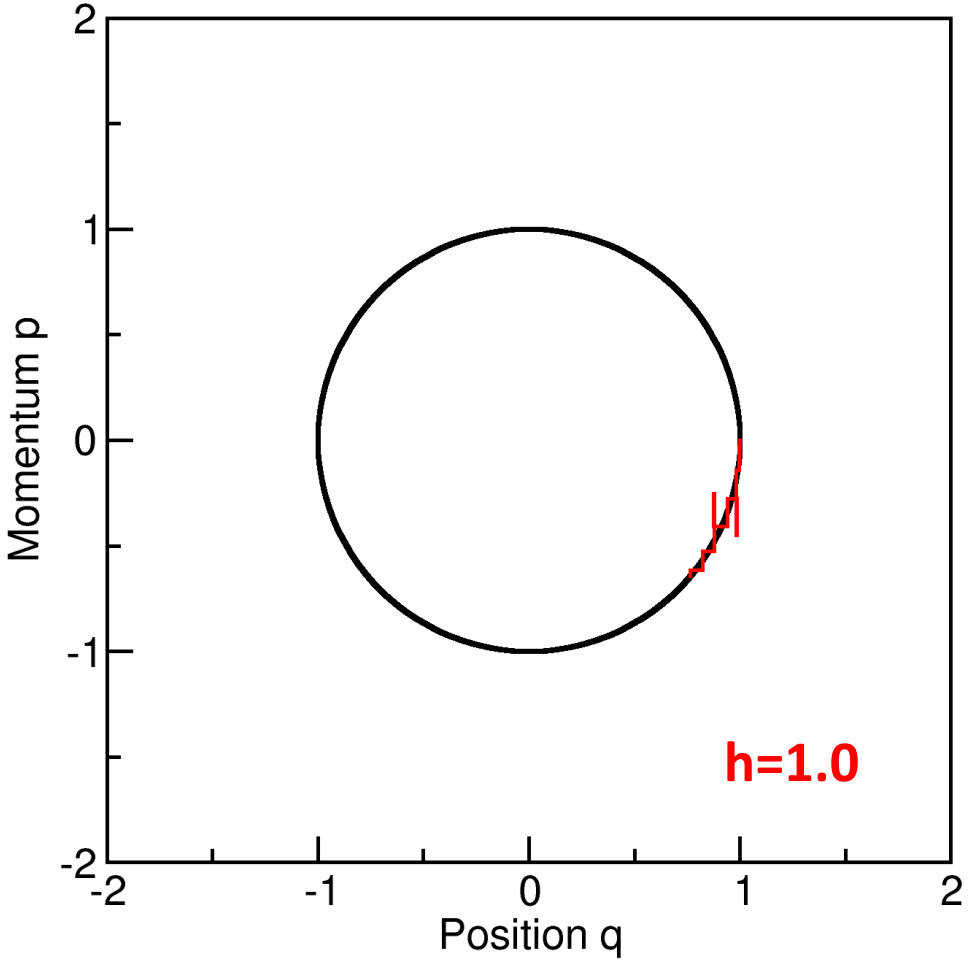
Applying one of our integrators to the
harmonic oscillator with different time step lengths...

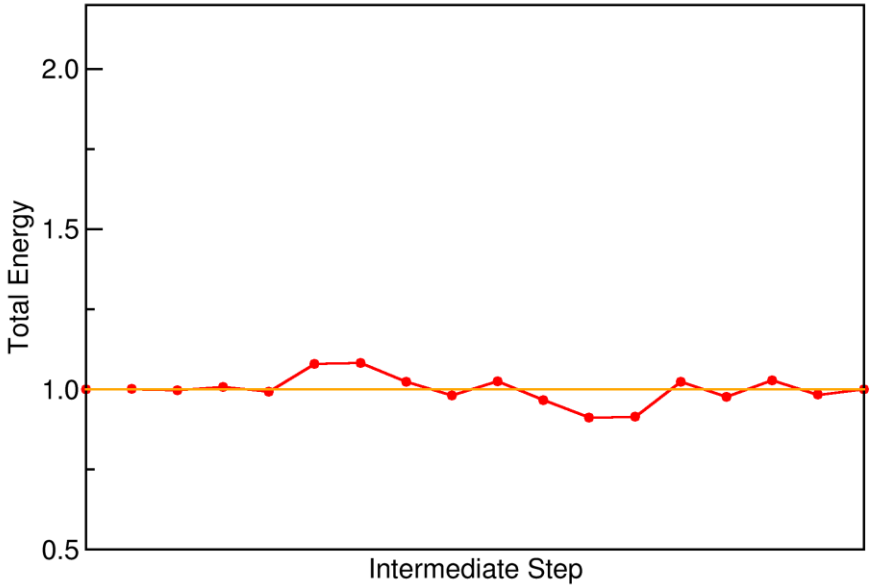




Phase Space Portraits

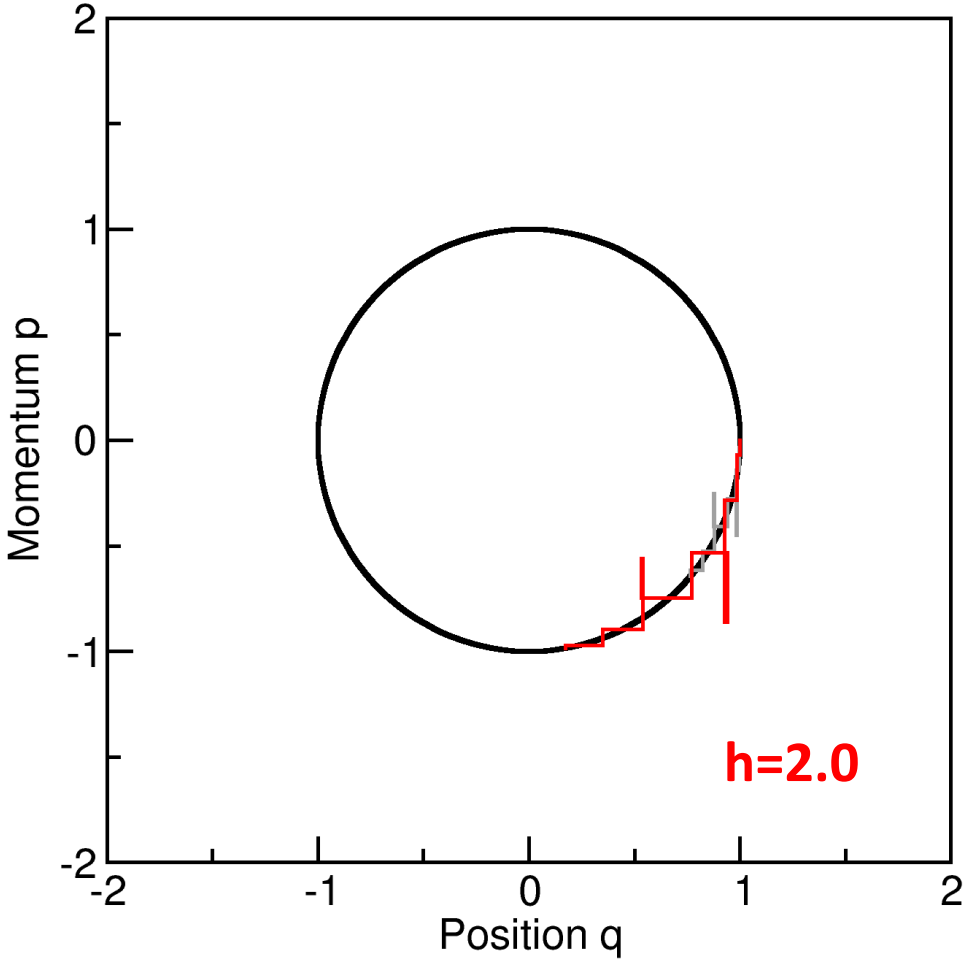
Applying one of our integrators to the
harmonic oscillator with different time step lengths...

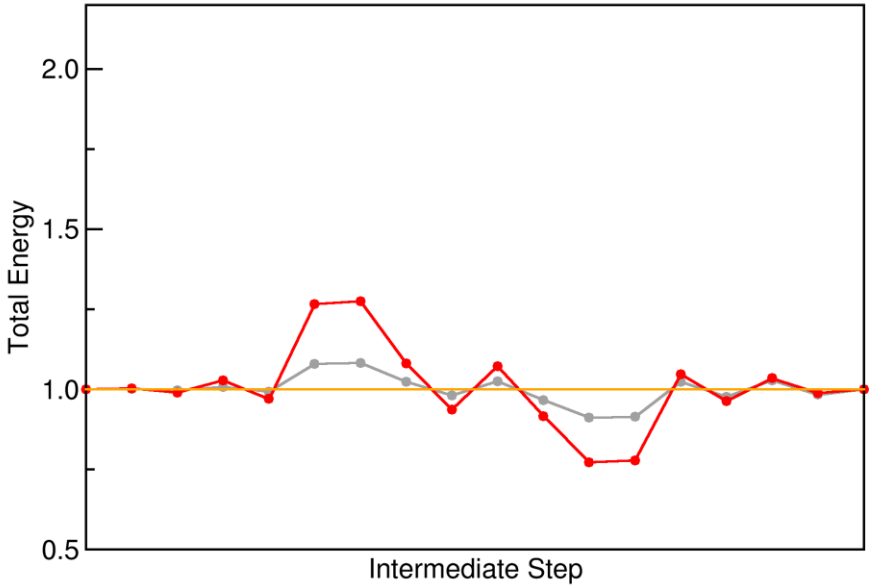




Phase Space Portraits

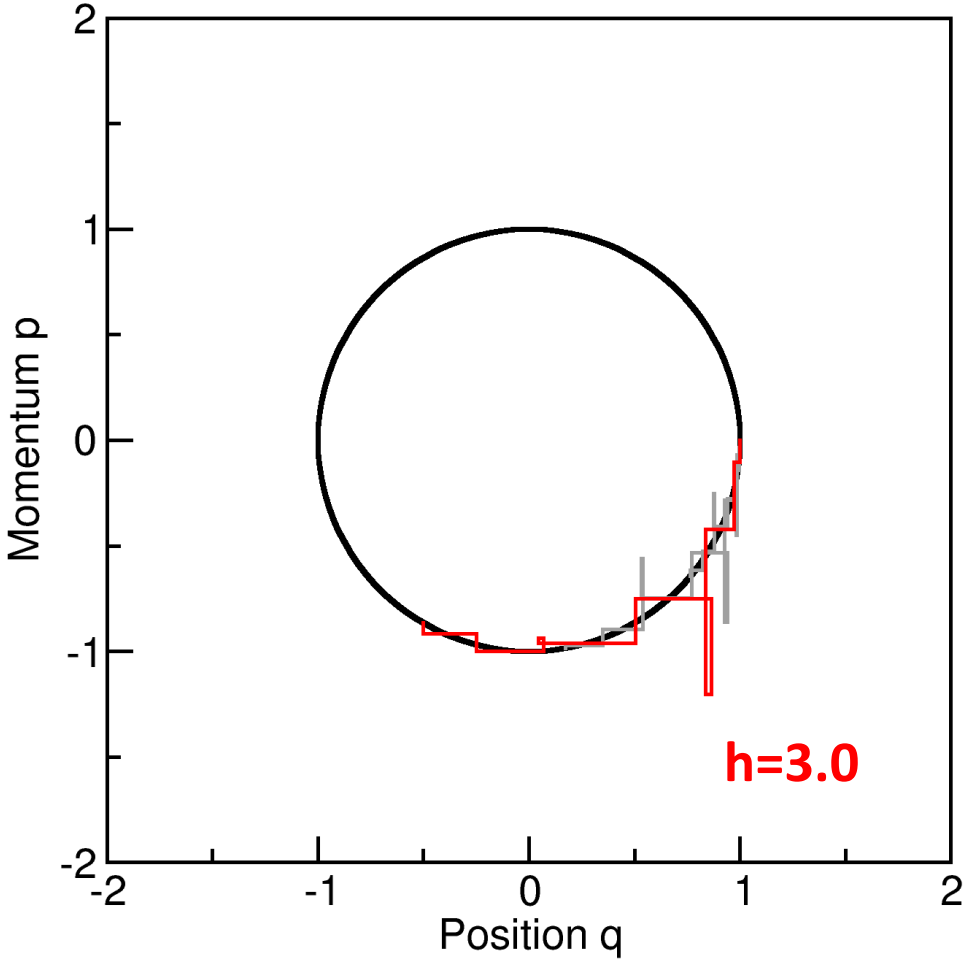
Applying one of our integrators to the
harmonic oscillator with different time step lengths...

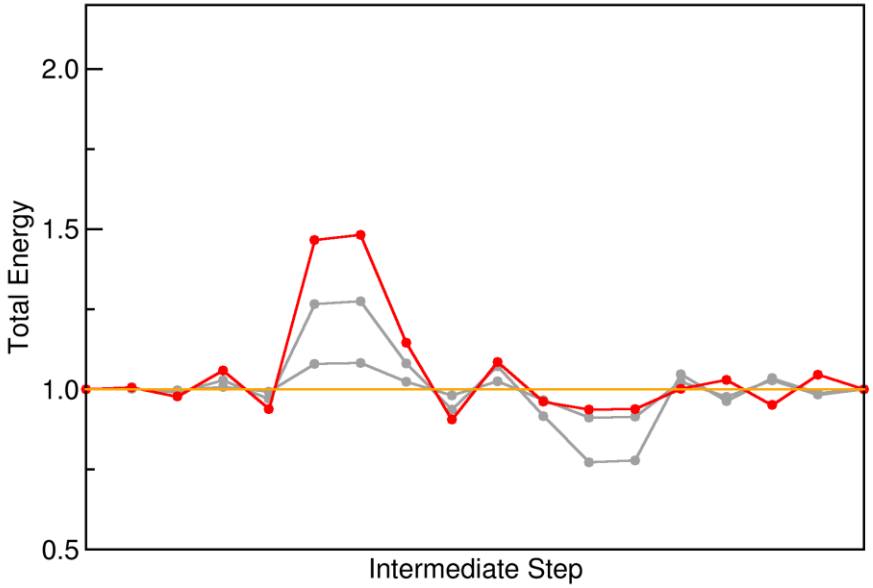




Phase Space Portraits

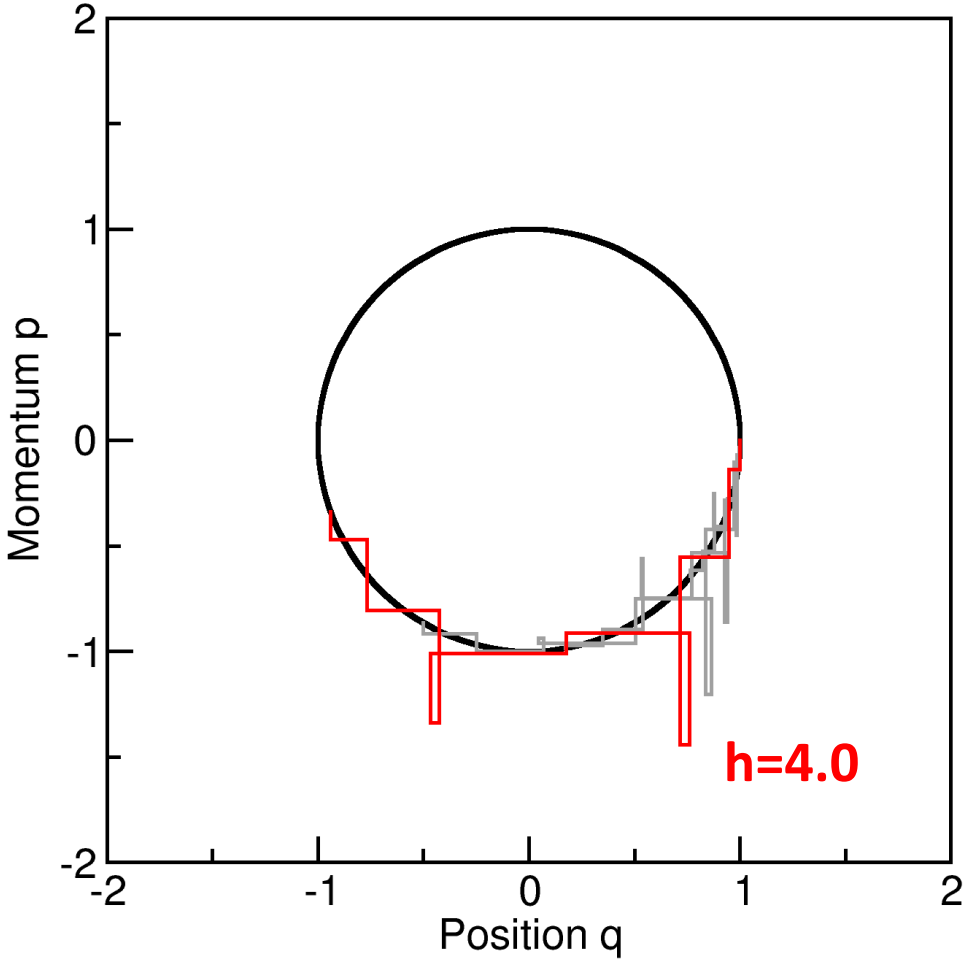
Applying one of our integrators to the
harmonic oscillator with different time step lengths...

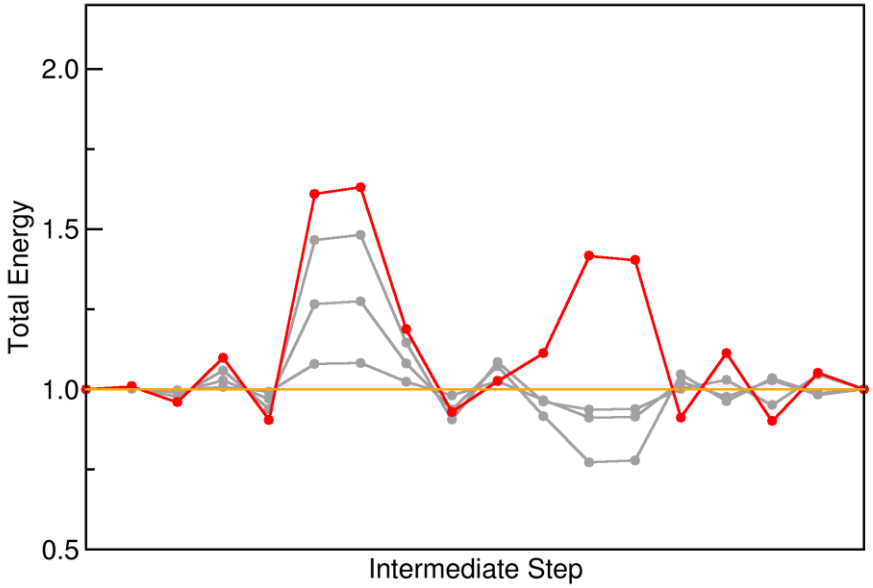




Phase Space Portraits

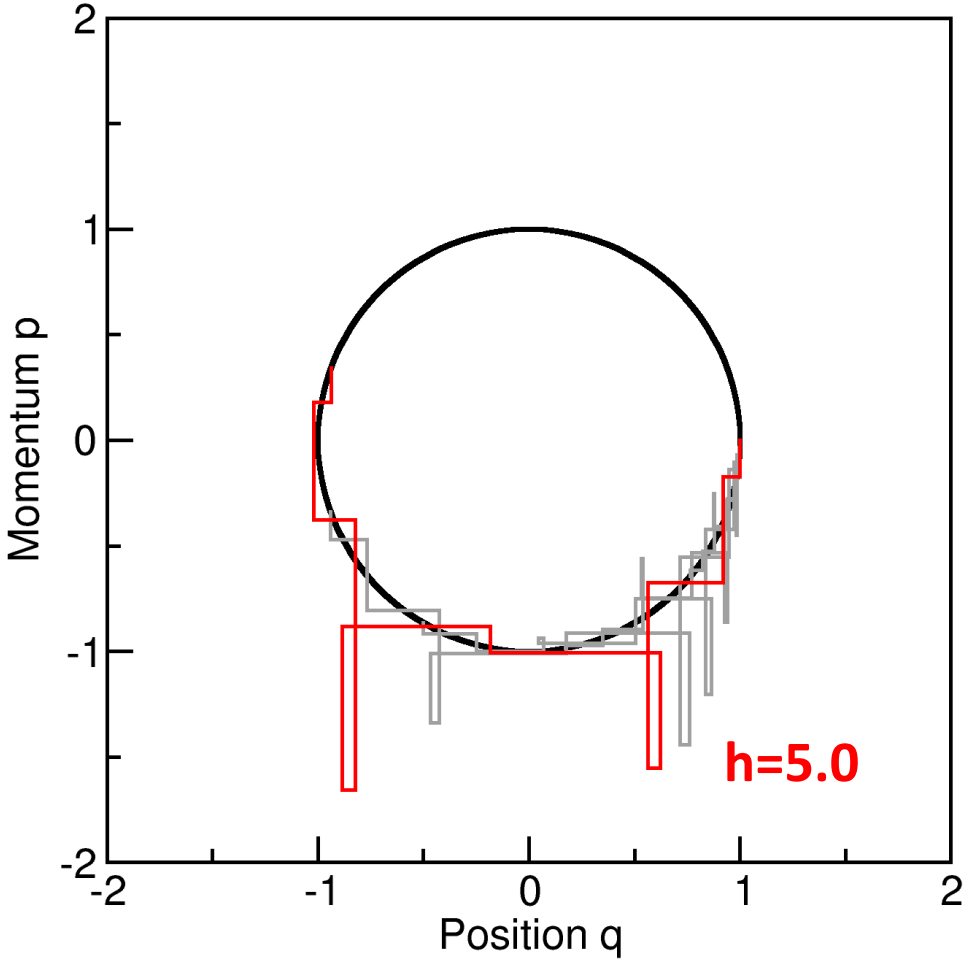
Applying one of our integrators to the
harmonic oscillator with different time step lengths...

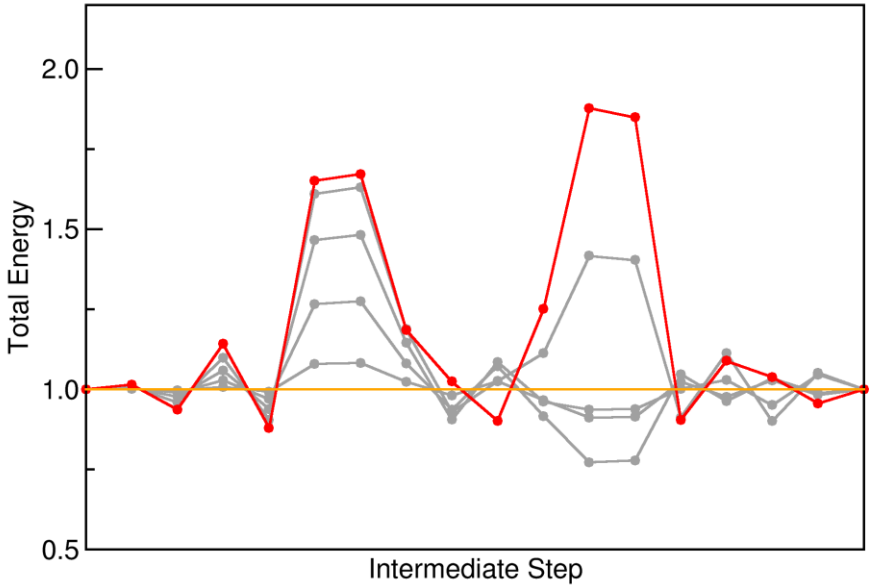




Phase Space Portraits

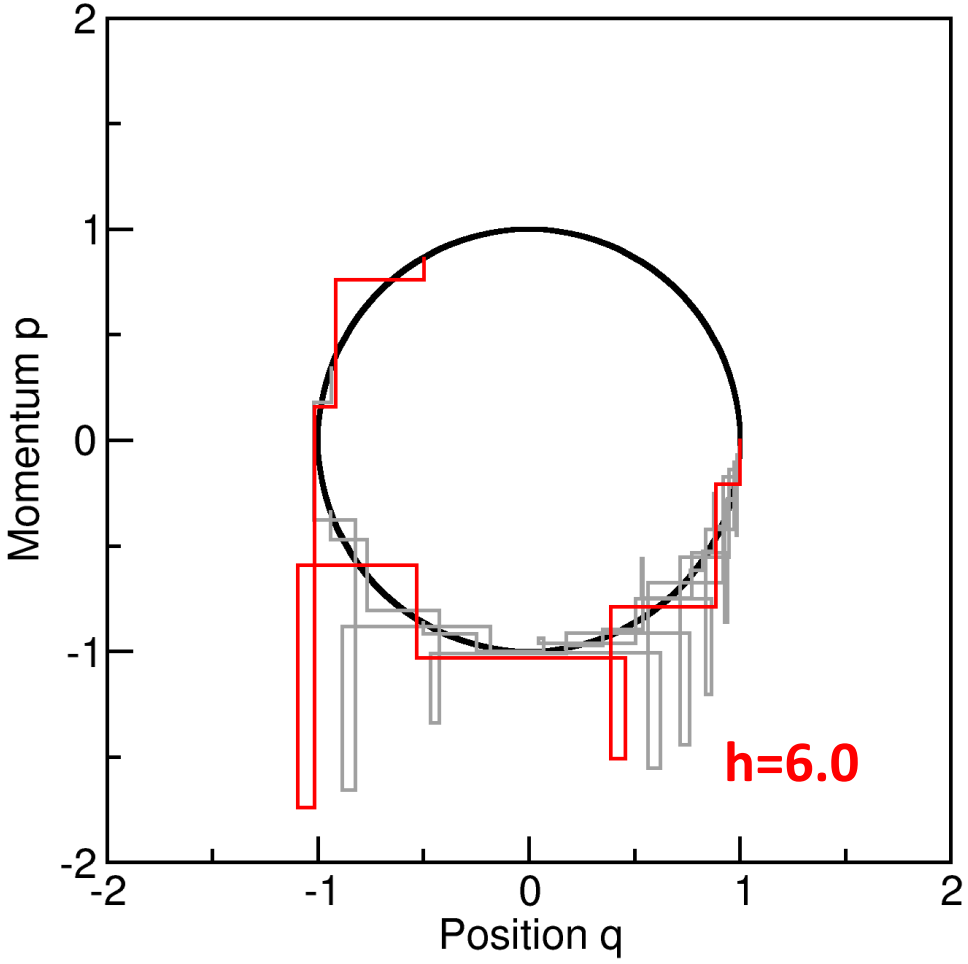
Applying one of our integrators to the
harmonic oscillator with different time step lengths...

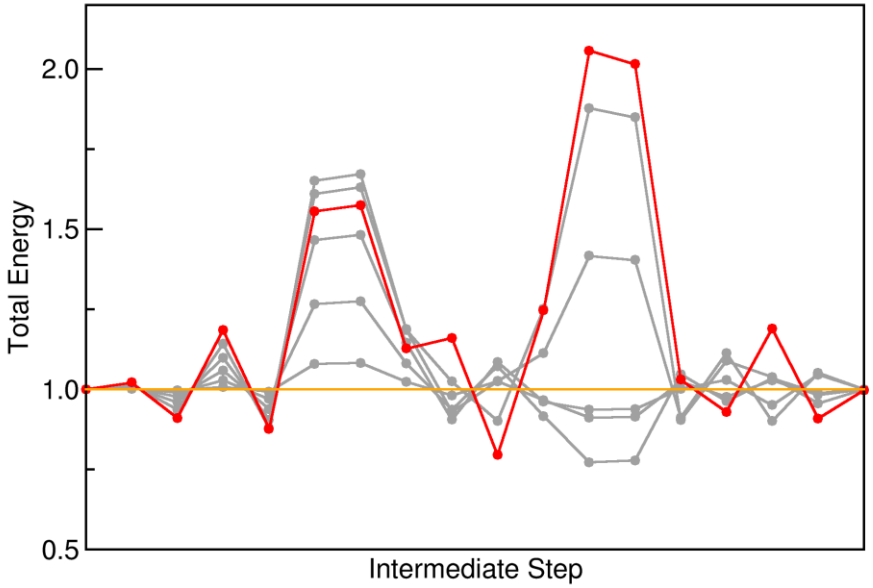




Phase Space Portraits

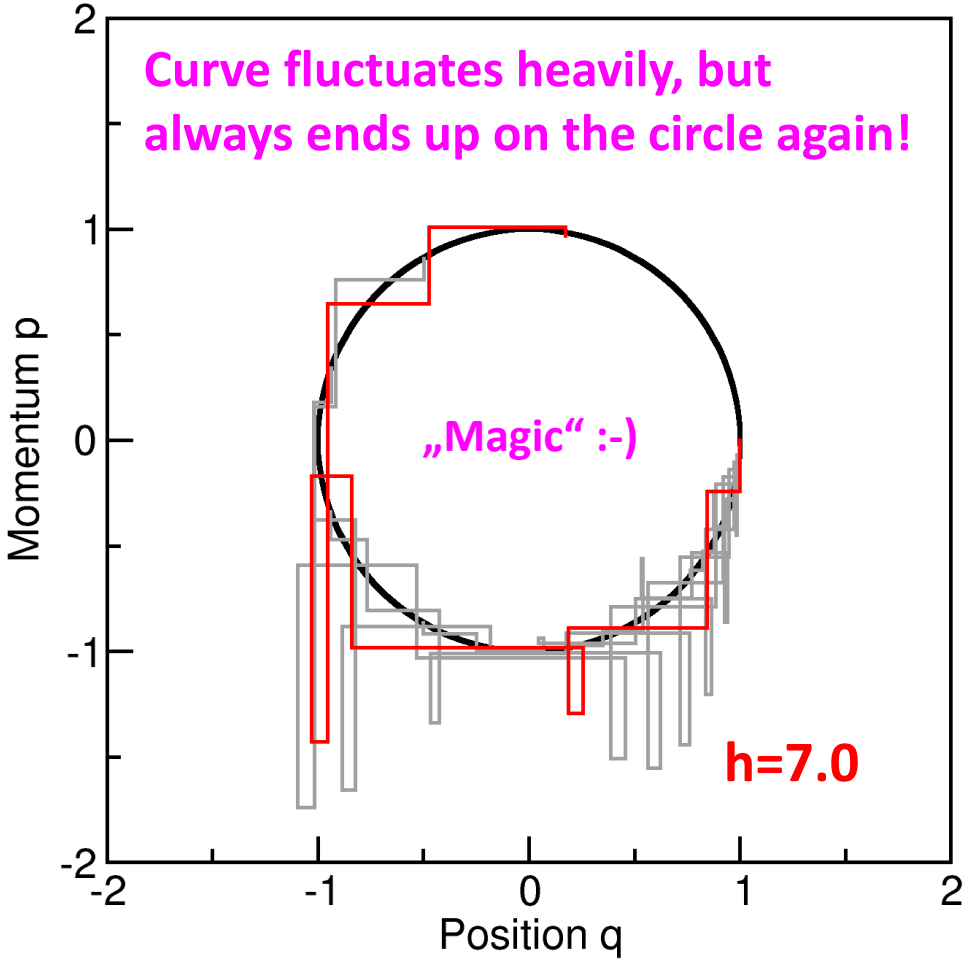
Applying one of our integrators to the
harmonic oscillator with different time step lengths...

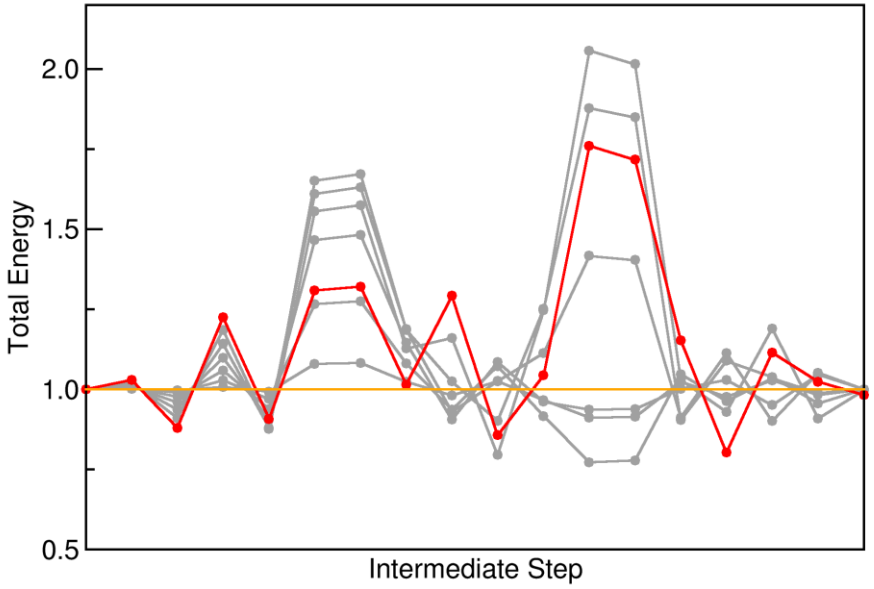




Phase Space Portraits

Applying one of our integrators to the
harmonic oscillator with different time step lengths...





How to Implement

- Our methods are all 8-stage symmetric symplectic Runge–Kutta–Nyström (RKN) approaches
- MB‘s math diploma thesis: These are equivalent to splitting methods, so that **implementation is extremely straight-forward**:

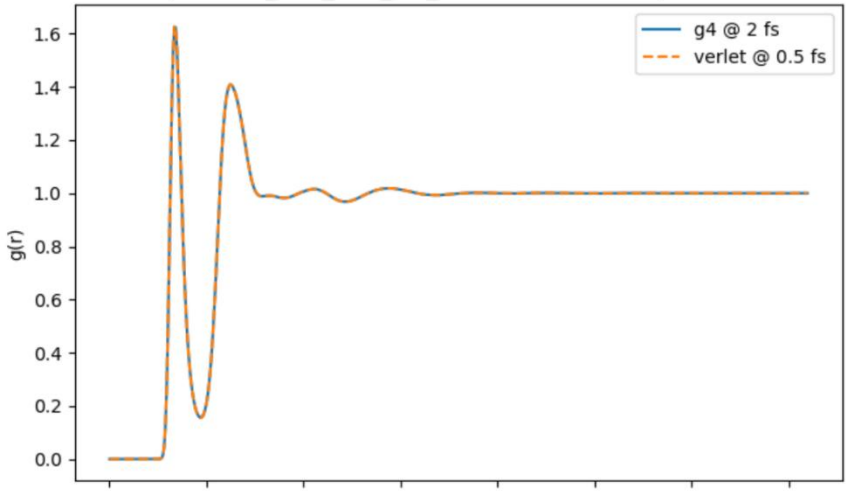
$$
\begin{aligned}
p &:= p + a_1\, h\, w(q),\\
q &:= q + b_1\, h\, u(p),\\
&\ldots\\
p &:= p + a_s\, h\, w(q),\\
q &:= q + b_s\, h\, u(p),
\end{aligned}
$$

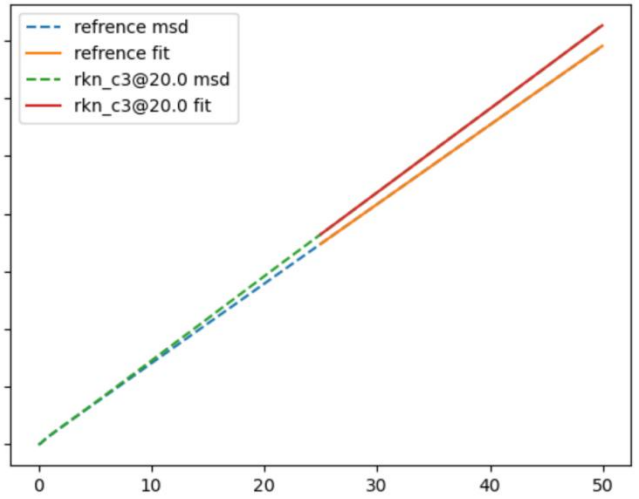
- Implementation in any MD software should be simple
- Works directly „on the variables“, no temporary storage needed
- 8 force calculations per MD step

Benchmarking

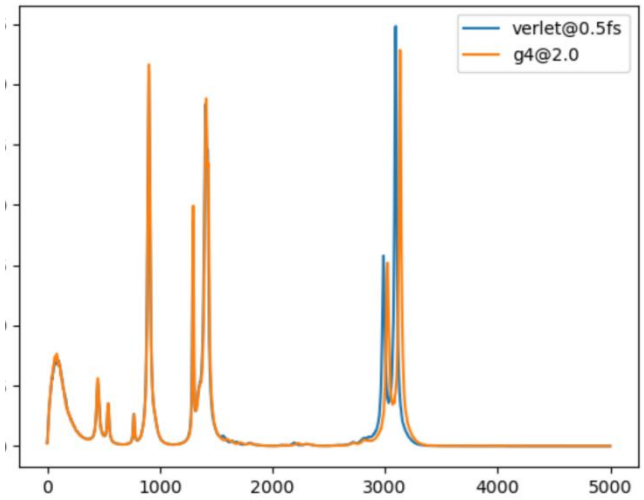
Extensive benchmarking is currently ongoing

(different systems, RDFs, MSDs, power spectra, ...)





Initial results look very good!



Conclusions II

- We optimized six different integration methods to increase efficiency in practical MD simulations
- These are all 8-stage symmetric symplectic Runge–Kutta–Nyström approaches
- **Performance gain** *(compared to Verlet Δt=0.5 fs):*
  - **Factor 2.5 faster** at same accuracy
  - **Factor 260 more accurate** at same computer cost
- Implementation is very straight-forward
- Extensive benchmarking *(structure / dynamics)* is ongoing
- Compatible with **all flavors of MD** *(force field, AIMD, ML)*
- Pilot implementation in **LAMMPS** and **CP2k** soon available

https://brehm-research.de/integrator

Acknowledgment

Our Posters:
P156 Omid
P137 Hamza
P170 Vishwas

AK Brehm

Christian Dreßler
(TU Ilmenau)

https://brehm-research.de/integrator