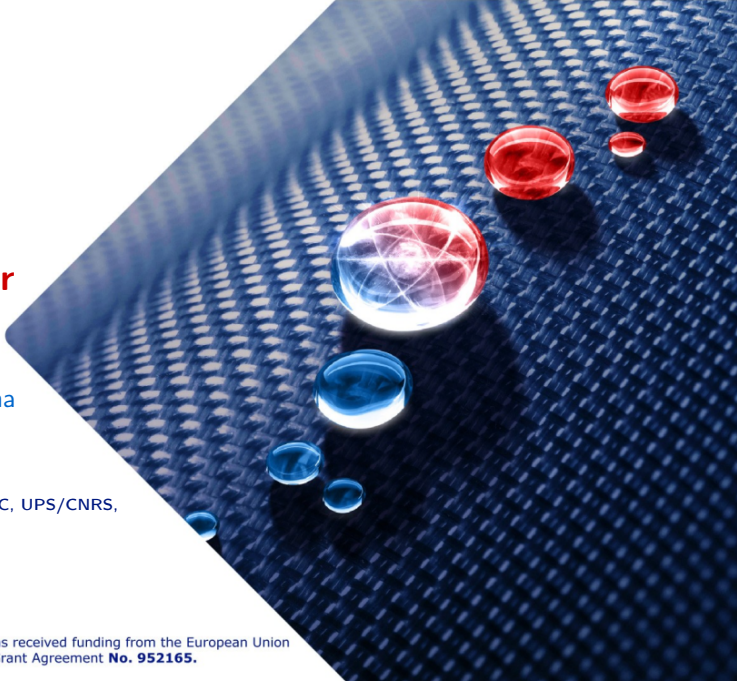
Targeting Real chemical accuracy at the EXascale

Important algorithms for CIPSI

Abdallah Ammar, Emmanuel Giner, Pierre-François Loos, Anthony Scemama

19/04/2023

Laboratoire de Chimie et Physique Quantiques, IRSAMC, UPS/CNRS, Toulouse



Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizoon 2020 research and innovation programme under Grant Agreement **No. 952165.**

Determinant-driven methods

- Integral-driven : sequential access to $\mathcal{O}(N^4)$ integrals, indirect access to vectors

```
for (i,j,k,l,integral) in all_integrals:
    pairs = find_determinant_pairs(i,j,k,l,ijkl)
    for (d1,d2) in pairs:
        do_work(d1,d2)
```

- Determinant-driven : sequential access to vectors, indirect access to integrals

```
for d1 in determinants:
  for d2 in determinants:
    i,j,k,l = get_excitation(d1,d2)
    do_work(d1,d2)
```

- Integral-driven: outer loop appears as $\mathcal{O}({N_{\mathrm{MO}}}^4)$, ignores zero integrals
- Determinant-driven: outer loops appear as $\mathcal{O}({N_{\mathrm{det}}}^2)$
- Efficient CIPSI: How to be efficient within a determinant-driven approach

Data structures

Slater-Condon's rules

$$
\begin{aligned}
\langle I|\mathcal{O}_1|I\rangle &= \sum_{i\in D}\langle\varphi_i|\mathcal{O}_1|\varphi_i\rangle \\
\langle I|\mathcal{O}_2|I\rangle &= \frac{1}{2}\sum_{(i,j)\in D}\langle\varphi_i\varphi_j|\mathcal{O}_2|\varphi_i\varphi_j\rangle - \langle\varphi_i\varphi_j|\mathcal{O}_2|\varphi_j\varphi_i\rangle \\
\langle I|\mathcal{O}_1|\hat{T}_i^j I\rangle &= \langle\varphi_i|\mathcal{O}_1|\varphi_j\rangle \\
\langle I|\mathcal{O}_2|\hat{T}_i^j I\rangle &= \sum_{k\in D}\langle\varphi_i\varphi_k|\mathcal{O}_2|\varphi_j\varphi_k\rangle - \langle\varphi_i\varphi_k|\mathcal{O}_2|\varphi_k\varphi_j\rangle \\
\langle I|\mathcal{O}_2|\hat{T}_{ik}^{jl} I\rangle &= \langle\varphi_i\varphi_k|\mathcal{O}_2|\varphi_j\varphi_l\rangle - \langle\varphi_i\varphi_k|\mathcal{O}_2|\varphi_l\varphi_j\rangle
\end{aligned}
$$

Need for functions : $f(I,J) \rightarrow (i,j,k,l,\phi)$

Double-determinant representation

A Slater determinant can be written as a Waller-Hartree double determinant

$$|I\rangle = \hat{I}\,|\rangle = -1^p \times \hat{I}_\uparrow \hat{I}_\downarrow\,|\rangle = -1^p \times \hat{I}_\uparrow\,|\rangle \otimes\ \hat{I}_\downarrow\,|\rangle$$

Storage:

- 1 determinant: one integer for $\hat{I}_\uparrow$ and one integer for $\hat{I}_\downarrow$
- Set the bit to 1 if the orbital is occupied
- $>$ 64 orbitals: N_{int} integers for $\hat{I}_\uparrow$ and for $\hat{I}_\downarrow$

Fast determinant comparisons

Bitwise operations (1 CPU cycle):

- `and`, `or`, `xor`, `shl`, `shr`: logical
- `shl`, `shr`: shift left/right
- `lzcnt`, `tzcnt` : Number of leading/trailing zero bits
- `popcnt` : Number of bits set to 1

Example: degree of excitation between $|I\rangle$ and $|J\rangle$:

```fortran
integer function degree(det_i, det_j, N_int)
  integer,   intent(in) :: N_int
  integer*8, intent(in) :: det_i(N_int,2), det_j(N_int,2)
  integer               :: two_d, i
  two_d = 0
  do i=1,N_int
     two_d = two_d + popcnt( ieor( det_i(i,1), det_j(i,1) ) ) &
                   + popcnt( ieor( det_i(i,2), det_j(i,2) ) )
  end do
  degree = rshift(two_d,1)
end function degree
```

Fast determinant comparisons

```
15 14 13 12 11 10 9  8  7  6  5  4  3  2  1  0
0000000011111111 : D1
0010010010101111 : D2
0010010001010000 : D1 xor D2
0000000001010000 : (D1 xor D2) and D1
0010010000000000 : (D1 xor D2) and D2
```

To get the orbital indices: number of leading/trailing zeros gives the positions of the 1's.

Constraints

- Integrals require a fast random access
- 8-fold permtutation symmetry $\langle ij|kl\rangle = \langle kj|il\rangle = \cdots$
- Many integrals are zero: need for a sparse data structure

Implementation

- Hash table
- $f(i,j,k,l) -> K$ gives the same K for all similar permutations
- $f(i+1,j,k,l) - f(i,j,k,l)$ is likely to be 1 : locality
- Array (cache) for 128^4 frequently used integrals

Table: Time to access integrals (in nanoseconds/integral) with different access patterns. The time to generate random numbers (measured as 67~ns/integral) was not counted in the random access results.

Access	Array	Hash table
i, j, k, l	9.72	125.79
i, j, l, k	9.72	120.64
i, k, j, l	10.29	144.65
l, k, j, i	88.62	125.79
l, k, i, j	88.62	120.64
Random	170.00	370.00

Efficient direct CI

Davidson

- Power method with preconditioning for symmetric diagonal-dominant matrices.
- Bottleneck: $\mathcal{H}|\Psi\rangle$

CIPSI

- CIPSI is not a *method* but an algorithm
- CIPSI can be seen as a refinement of Davidson's diagonalization algorithm:
  At every iteration:
  - Davidson: add *all* singles and doubles, stop at $\Delta E = 10^{-15}$
  - CIPSI: add *selected* singles and doubles, stop at $E_{PT2} = 10^{-4}$, $N_{\text{det max}}$, ...
- Everything that can be done with Davidson can be done with CIPSI: preserve symmetries (space and spin), limit degree of excitation (CISD, CISDTQ, etc), limit space (CAS), effective Hamiltonians, excited states, *etc.*

Popular misconception

Sorting is *not* $\mathcal{O}(N \log(N))$: sorting is $\mathcal{O}(N \log(M))$ (linear in N, log in M)

- A is an array of N integer values
- The bitmask is an integer with only one bit set to one (00001000)

```
void radix_sort(int* A, size_t N, int bitmask) {
    if (bitmask == 0) return;
    int left[N], right[N];
    int p=0 ; int q=0 ;
    for (int i=0 ; i<N ; i++) {
        if (A[i] & bitmask) { right[q] = A[i]; q++; }
        else                { left [p] = A[i]; p++; } }
    radix_sort(left , p, bitmask >> 1) ;
    radix_sort(right, q, bitmask >> 1) ;
    for (int i=0 ; i<p ; i++) { A[  i] = left [i] ; }
    for (int i=0 ; i<q ; i++) { A[p+i] = right[i] ; }
}
```

Double-determinant representation of Ψ

$$\Psi = \sum_{I} c_I |I\rangle = \sum_{k=1}^{N_{\text{det}}^{\uparrow}} \sum_{m=1}^{N_{\text{det}}^{\downarrow}} C_{km} D_k^{\uparrow} D_m^{\downarrow}$$

- If $D_k^{\uparrow}$ and $D_m^{\downarrow}$ are represented as N_{MO} -bit strings, this transformation can be done in $\mathcal{O}(N_{\text{det}} \times N_{\text{MO}})$ (sorting).
- Searching for same-spin excitations: looping over k or m : $\mathcal{O}(N_{\text{det}}^{\uparrow}) \sim \mathcal{O}(\sqrt{N_{\text{det}}})$

For all $I = D_k^{\uparrow} D_m^{\downarrow}$ in Ψ:

- Find indices p of $\uparrow$ singles and $\uparrow\uparrow$ doubles

$$\langle I|\mathcal{H}|\Psi\rangle = \sum_J \langle I|\mathcal{H}|J\rangle c_J = \sum_p \langle D_k^{\uparrow} D_m^{\downarrow}|\mathcal{H}|D_p^{\uparrow} D_m^{\downarrow}\rangle C_{pm}$$

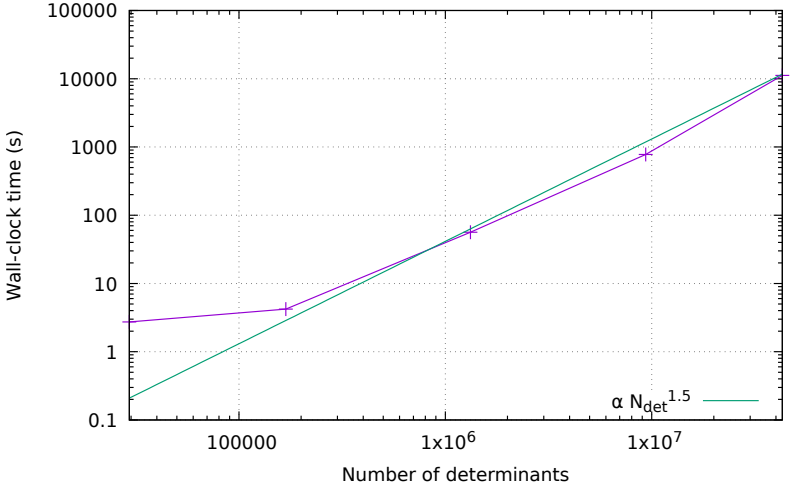
- Find indices q of $\downarrow$ singles and $\downarrow\downarrow$ doubles

$$\langle I|\mathcal{H}|\Psi\rangle = \sum_J \langle I|\mathcal{H}|J\rangle c_J = \sum_q \langle D_k^{\uparrow} D_m^{\downarrow}|\mathcal{H}|D_k^{\uparrow} D_q^{\downarrow}\rangle C_{kq}$$

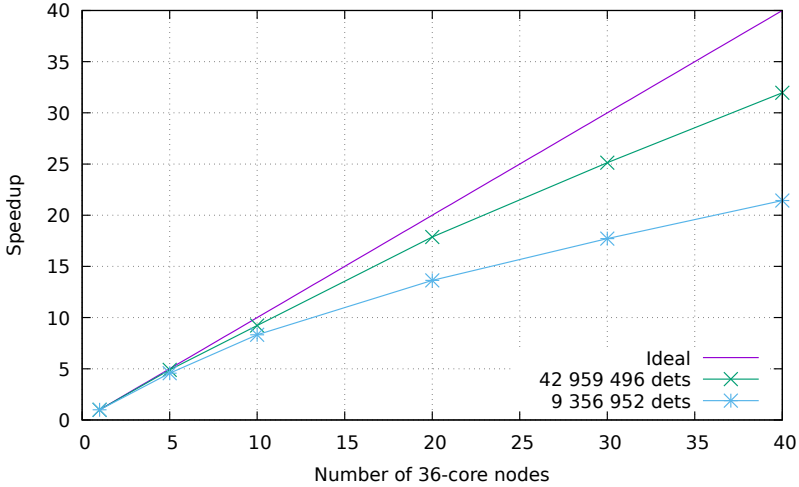
- Find indices pq of $\uparrow\downarrow$ doubles:
  - Find indices p of $\uparrow$ singles
  - Find indices q of $\downarrow$ singles

$$\langle I|\mathcal{H}|\Psi\rangle = \sum_J \langle I|\mathcal{H}|J\rangle c_J = \sum_{pq} \langle D_k^{\uparrow} D_m^{\downarrow}|\mathcal{H}|D_p^{\uparrow} D_q^{\downarrow}\rangle C_{pq}$$

Scaling with N_{det}



Parallel efficiency



Stochastic evaluation of the PT2 correction and selection

Epstein-Nesbet Second order correction

Consider a wave function Ψ expanded on an *arbitrary* set $\mathcal{D}$ of N_{det} orthonormal Slater determinants,

$$\Psi = \sum_{I \in \mathcal{D}} c_I |I\rangle, \quad E_{\text{var}} = \frac{\langle \Psi | \mathcal{H} | \Psi \rangle}{\langle \Psi | \Psi \rangle}$$

The Epstein-Nesbet 2nd order correction to the energy is

$$E_{\text{PT2}} = \sum_{\alpha \in \mathcal{A}} \frac{\langle \Psi | \mathcal{H} | \alpha \rangle \langle \alpha | \mathcal{H} | \Psi \rangle}{E_{\text{var}} - \langle \alpha | \mathcal{H} | \alpha \rangle}$$

The set $\mathcal{A}$ contains the Slater determinants

- that are not in $\mathcal{D}$
- for which $d(I, \alpha) = 1$ or 2 for at least one pair (I, α)

Formal scaling

$$E_{\mathrm{PT2}} = \sum_{\alpha \in \mathcal{A}} \frac{\langle \Psi | \mathcal{H} | \alpha \rangle \langle \alpha | \mathcal{H} | \Psi \rangle}{E_{\mathrm{var}} - \langle \alpha | \mathcal{H} | \alpha \rangle} = \sum_{\alpha \in \mathcal{A}} \frac{\left(\sum_{I \in \mathcal{D}} c_I \langle I | \mathcal{H} | \alpha \rangle \right)^2}{E_{\mathrm{var}} - \langle \alpha | \mathcal{H} | \alpha \rangle}$$

- Size of $\mathcal{A}$: size of $(\hat{T}_1 + \hat{T}_2)|\Psi\rangle$
- Number of non-zero terms : $d(I, \alpha) \leq 2 \sim N_{\mathrm{det}} \times \left[\left(N_{\mathrm{elec}}^{\uparrow} \times (N_{\mathrm{MO}} - N_{\mathrm{elec}}^{\uparrow}) \right)^2 \right]$
- Expensive

Solutions to make simulations possible

"Non-general" but *conventional* solutions:

- Partition the MO space into different classes (active, virtual, inactive, *etc*)
- Use another zeroth-order Hamiltonian (CAS-PT2, NEV-PT2)

Solutions applicable to *any* wave function:

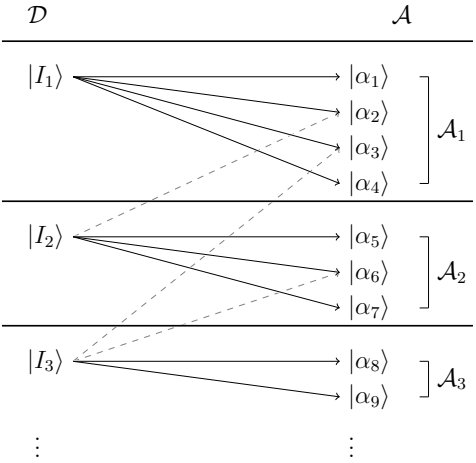
- Truncation of $\mathcal{D}$ to consider only contributions due to large c_I
  But: Truncation $\longrightarrow$ bias because E_{PT2} is a sum of same-sign values (negative).
- Algorithmic improvement
- Monte Carlo sampling in $\mathcal{A}$. Unbiased method
  But: Statistical error decreases as $\mathcal{O}\left(1/\sqrt{N_{\mathrm{samples}}}\right) \Longrightarrow$ Difficult to get $10^{-5} a.u$ precision.
- Parallelism

Central idea

- Choose an arbitrary ordering of $|I\rangle$. Natural choice:

$$w_I = \frac{{c_I}^2}{\langle\Psi|\Psi\rangle}$$

- Make *disjoint* groups $\mathcal{A}_I$ of $|\alpha\rangle$ originating from the same generator $|I\rangle$
- Each $\mathcal{A}_I$ has its own contribution ϵ_I to E_{PT2}



Central idea

$$
\begin{aligned}
E_{\mathrm{PT2}} &= \sum_{\alpha \in \mathcal{A}} \frac{(\langle \Psi | \mathcal{H} | \alpha \rangle)^2}{E_{\mathrm{var}} - \langle \alpha | \mathcal{H} | \alpha \rangle} \\
&= \sum_{I \in \mathcal{D}} \sum_{\alpha_I \in \mathcal{A}_I} \frac{(\langle \Psi | \mathcal{H} | \alpha_I \rangle)^2}{E_{\mathrm{var}} - \langle \alpha_I | \mathcal{H} | \alpha_I \rangle} \\
&= \sum_{I \in \mathcal{D}} \epsilon_I
\end{aligned}
$$

Contribution per *internal* determinant

$$
\epsilon_I = \sum_{\alpha_I \in \mathcal{A}_I} \frac{(\langle \Psi | \mathcal{H} | \alpha_I \rangle)^2}{E_{\mathrm{var}} - \langle \alpha_I | \mathcal{H} | \alpha_I \rangle}
$$

From $\mathcal{O}(N_{\text{det}}^2)$ to $\mathcal{O}(N_{\text{det}}^{3/2})$

$$\Psi = \sum_I c_I |I\rangle = \sum_{k=1}^{N_{\text{det}}^{\uparrow}} \sum_{m=1}^{N_{\text{det}}^{\downarrow}} C_{km} D_k^{\uparrow} D_m^{\downarrow}$$

- Sorting is $\mathcal{O}(N_{\text{det}})$
- $\langle I|\mathcal{H}|\alpha\rangle\langle\alpha|\mathcal{H}|J\rangle = 0$ when $d(I, J) > 4$
- Loop over $N_{\text{det}}^{\uparrow}$ determinants (rows of the C matrix)
  Remove all the rows where $d(D_k^{\uparrow}, D_{k'}^{\uparrow}) > 4$ ($\sim \mathcal{O}(\sqrt{N_{\text{det}}})$)
- Loop over $N_{\text{det}}^{\downarrow}$ determinants (columns of the C matrix)
  Remove all the columns where $d(D_m^{\downarrow}, D_{m'}^{\downarrow}) > 4$)
- The remaining number of determinants is bounded by the size of the CISDTQ space

Finer filtering

$$\epsilon_I = \sum_{\alpha \in \mathcal{A}_I} \frac{\langle \Psi'_I | \mathcal{H} | \alpha_I \rangle \langle \alpha_I | \mathcal{H} | \Psi'_I \rangle}{E_{\text{var}} - \langle \alpha_I | \mathcal{H} | \alpha_I \rangle}$$

- We know that all the $|\alpha_I\rangle$ are singles and doubles with respect to $|I\rangle$
- $|\Psi'_I\rangle$ is the projection of $|\Psi\rangle$ on the subspace of determinants in $\mathcal{D}$ which are no more than quadruply excited with respect to $|I\rangle$
- For a subset of excitations $ij \to ab$, $|\Psi'\rangle$ is filtered further with possible hole/particle constraints

$$\epsilon_I = \sum_{\alpha_I \in \mathcal{A}_I} \frac{(\langle \Psi | \mathcal{H} | \alpha_I \rangle)^2}{E_{\text{var}} - \langle \alpha_I | \mathcal{H} | \alpha_I \rangle}$$

1. $\langle \Psi | \mathcal{H} | \alpha_I \rangle = \sum_{J \geq I} c_J \langle J | \mathcal{H} | \alpha_I \rangle$
2. $\langle \alpha_I | \mathcal{H} | \alpha_I \rangle$ is always large (otherwise $|\alpha_I\rangle$ would be better in the variational space, and PT is questionable)

- $\forall I \in \mathcal{D} : \epsilon_I \leq 0$
- $|\epsilon_I|$ is expected to decrease as ${c_I}^2$
- The computational cost decreases with I

Monte Carlo formulation

$$E_{\text{PT2}} = \sum_{I \in \mathcal{D}} \epsilon_I = \sum_{I \in \mathcal{D}} p_I \frac{\epsilon_I}{p_I} = \left\langle \frac{\epsilon_I}{p_I} \right\rangle_{p_I}$$

Naive sampling

Uniform sampling:
$p_I = \frac{1}{N_{\text{det}}}$

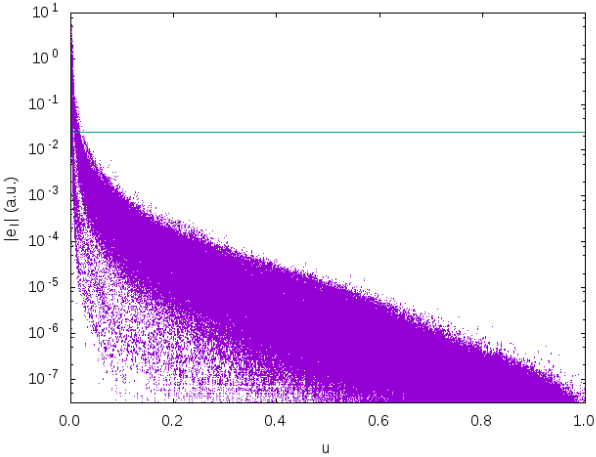


Figure: F_2, cc-pVDZ, 10^6 determinants in the variational space

Improved sampling

Sampling :
$p_I = {c_I}^2$

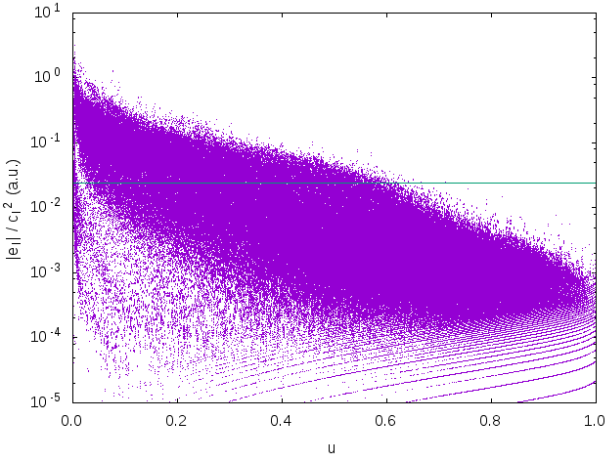


Figure: F_2, cc-pVDZ, 10^6 determinants in the variational space

Only N_{det} contributions $\epsilon_I \longrightarrow$ all ϵ_I can be stored in memory.

Lazy Evaluation (Wikipedia)

In programming language theory, *lazy evaluation*, or *call-by-need* is an evaluation strategy which delays the evaluation of an expression until its value is needed (non-strict evaluation) and which also avoids repeated evaluations (sharing).

```
def lazy_e(i):
    if not e_is_computed[i]:
        e[i] = compute_e(i)
        e_is_computed[i] = true
    return e[i]
```

Monte Carlo with Lazy Evaluation

$$E_{\mathrm{PT2}} = \sum_{I \in \mathcal{D}} \epsilon_I = \sum_{I \in \mathcal{D}} p_I \frac{\epsilon_I}{p_I} = \left\langle \frac{\epsilon_I}{p_I} \right\rangle_{p_I}$$

- Draw a generator determinant $|I\rangle$ with probability p_I
- Increment n_I, the number of evaluations of ϵ_I
- If ϵ_I is not already computed, compute it and store its value
- $E_{\mathrm{PT2}} \sim \sum_{I \in \mathcal{D}} \frac{n_I}{N_{\mathrm{samples}}} \frac{\epsilon_I}{p_I}$
- Statistical error : $\mathcal{O}\left(1/\sqrt{N_{\mathrm{samples}}}\right)$
- Lazy evaluation : Exponential acceleration (time to solution)

Monte Carlo with Lazy Evaluation

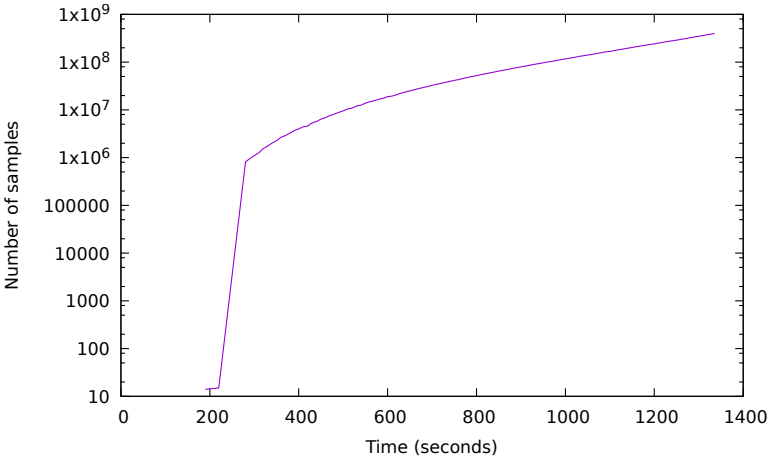


Figure: F_2, cc-pVDZ, 10^6 determinants in the variational space

Monte Carlo with Lazy Evaluation

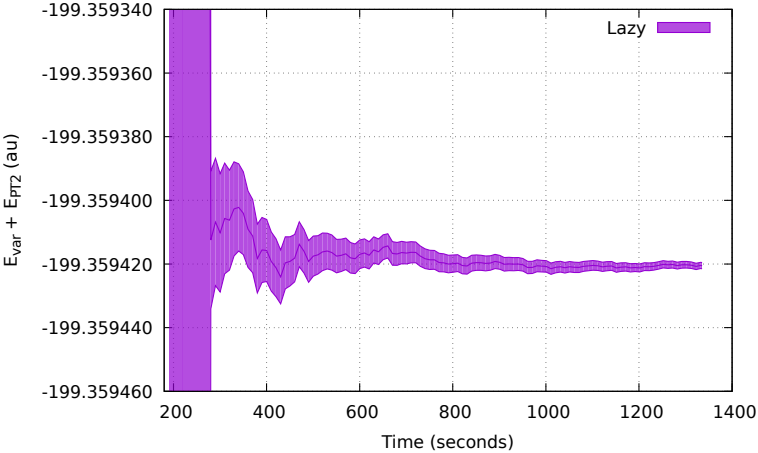


Figure: F_2, cc-pVDZ, 10^6 determinants in the variational space

Monte Carlo with Lazy Evaluation

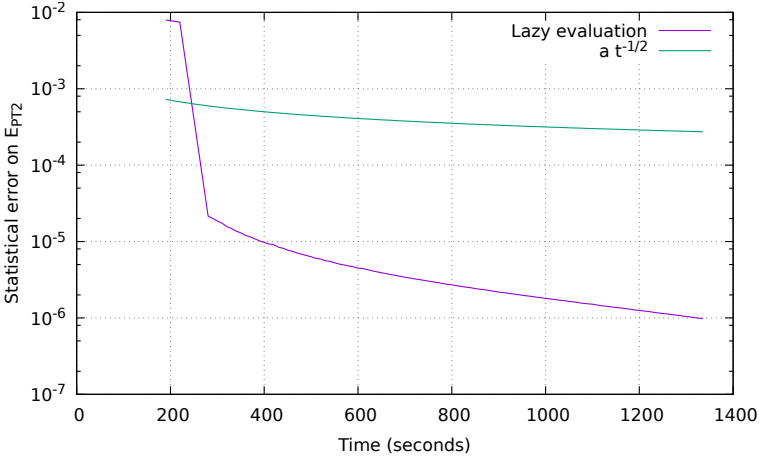


Figure: F_2, cc-pVDZ, 10^6 determinants in the variational space

Monte Carlo with Variance reduction

- Noise can be smoothed out by averaging
- Split $\mathcal{D}$ into M *equiprobable* sets : "Comb"

$$E_{\mathrm{PT2}} = \sum_{I \in \mathcal{D}} \epsilon_I = \sum_{k=1}^{M} \sum_{I_k \in \mathcal{D}_k} \epsilon_{I_k}$$

New Monte Carlo estimator

$$E_{\mathrm{PT2}} = \left\langle \frac{1}{M} \sum_{k=1}^{M} \frac{\epsilon_{I_k}}{p_{I_k}} \right\rangle_{(p_{I_1},\ldots,p_{I_M})}$$

Monte Carlo with Variance reduction

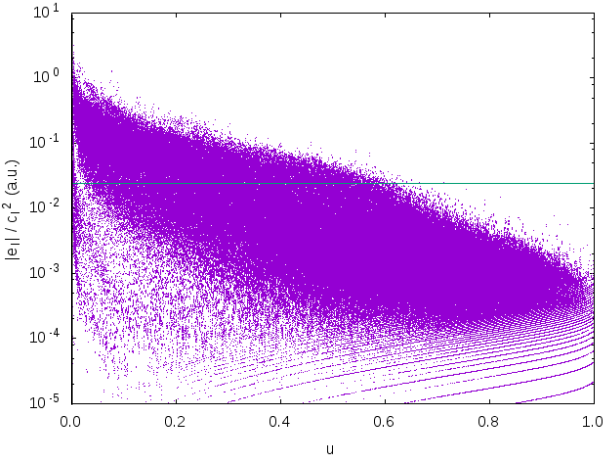


Figure: F_2, cc-pVDZ, 10^6 determinants in the variational space

Monte Carlo with Variance reduction

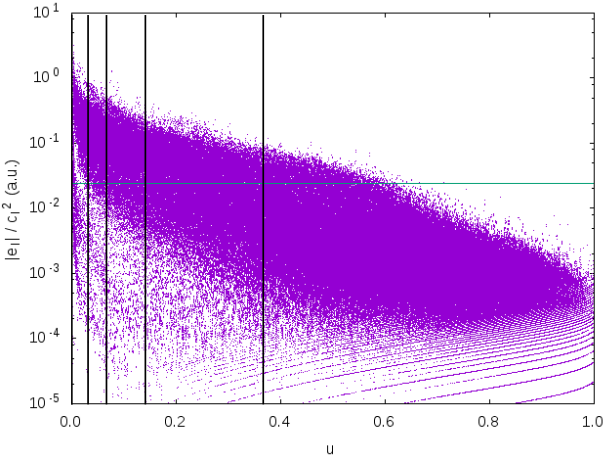


Figure: F_2, cc-pVDZ, 10^6 determinants in the variational space

Monte Carlo with Variance reduction

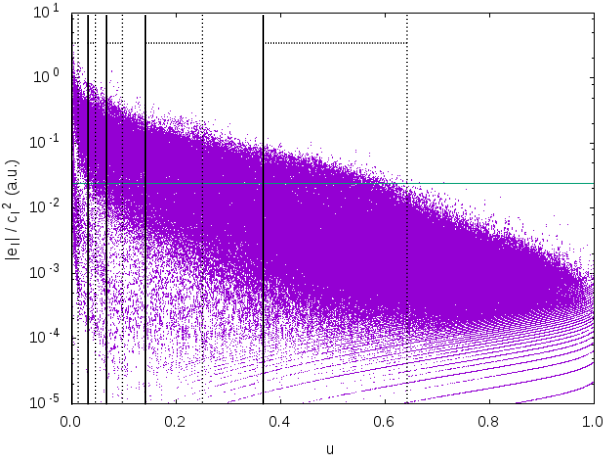


Figure: F_2, cc-pVDZ, 10^6 determinants in the variational space

Monte Carlo with Variance reduction

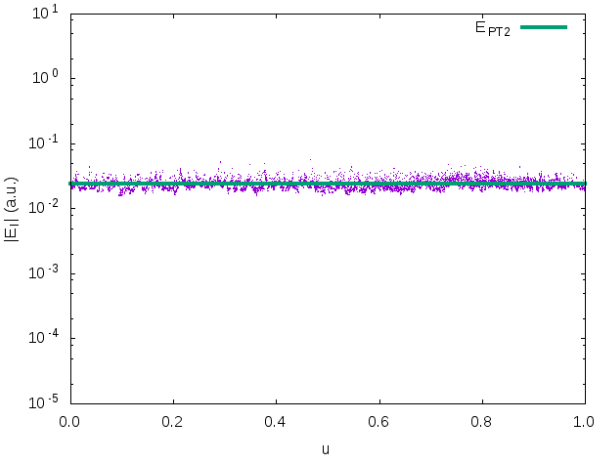


Figure: F_2, cc-pVDZ, 10^6 determinants in the variational space

Monte Carlo with Variance reduction

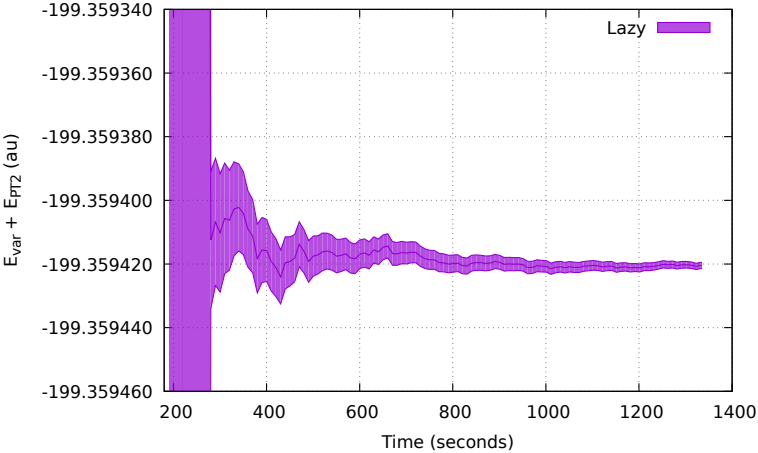


Figure: F_2, cc-pVDZ, 10^6 determinants in the variational space

Monte Carlo with Variance reduction

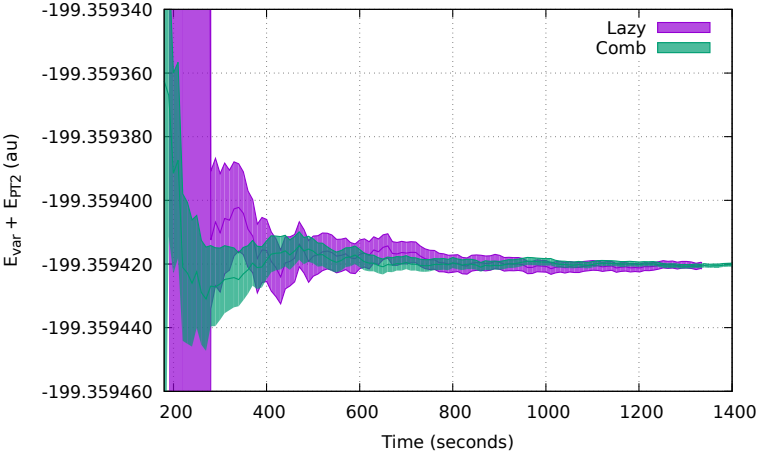


Figure: F_2, cc-pVDZ, 10^6 determinants in the variational space

Monte Carlo with Variance reduction

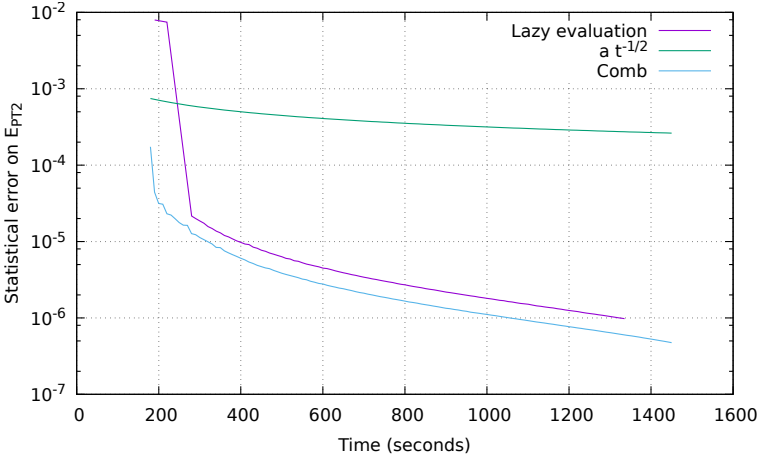


Figure: F_2, cc-pVDZ, 10^6 determinants in the variational space

Hybrid deterministic/stochastic scheme

- When all the determinants have been drawn, the *exact* E_{PT2} can be computed
- $\Longrightarrow$ The result with zero statistical error can be reached in a finite time
- In typical wave functions, 90% of the norm is on a few determinants
- Compute the few first contributions ϵ_I, and perform the MC in the rest

$$E_{\mathrm{PT2}} = \sum_{I \in \mathcal{D}_D} \epsilon_I + \left\langle \frac{1}{M} \sum_{k=1}^{M} \frac{\epsilon_{I_k}}{p_{I_k}} \right\rangle_{(p_{I \in \mathcal{D}_S})}$$

Hybrid deterministic/stochastic scheme

Make the deterministic part grow during the calculation.

At each MC step

- Draw a random number
- Find the determinants selected by the comb (increment n_I's)
- Compute the ϵ_I which have not been yet computed
- Compute deterministically the first non-computed determinant
- If a tooth of the comb is completely filled $\Longrightarrow$ Deterministic

At any time

$$E_{\mathrm{PT2}}(t) = \sum_{I \in \mathcal{D}_D(t)} \epsilon_I + \sum_{I \in \mathcal{D}_S(t)} \frac{1}{M(t)} \frac{n_I(t)}{N_{\mathrm{samples}}(t)} \frac{\epsilon_I}{p_I}$$

Hybrid deterministic/stochastic scheme

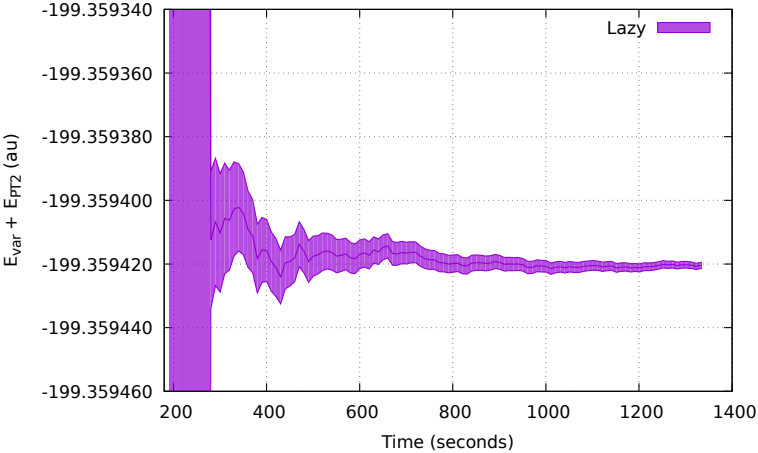


Figure: F_2, cc-pVDZ, 10^6 determinants in the variational space

Hybrid deterministic/stochastic scheme

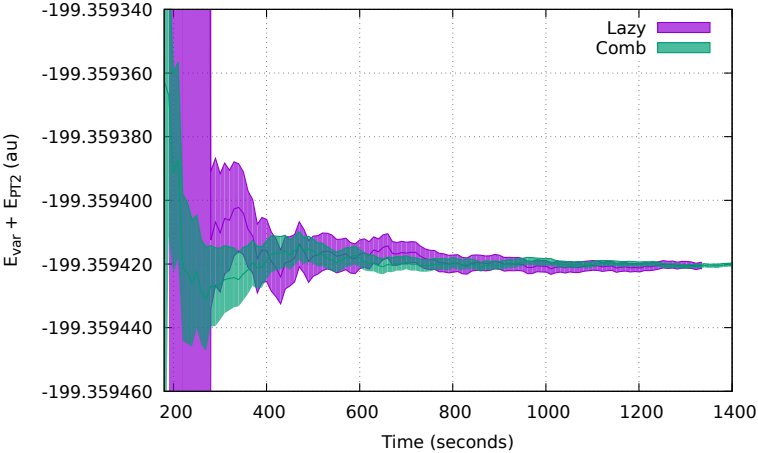


Figure: F_2, cc-pVDZ, 10^6 determinants in the variational space

Hybrid deterministic/stochastic scheme

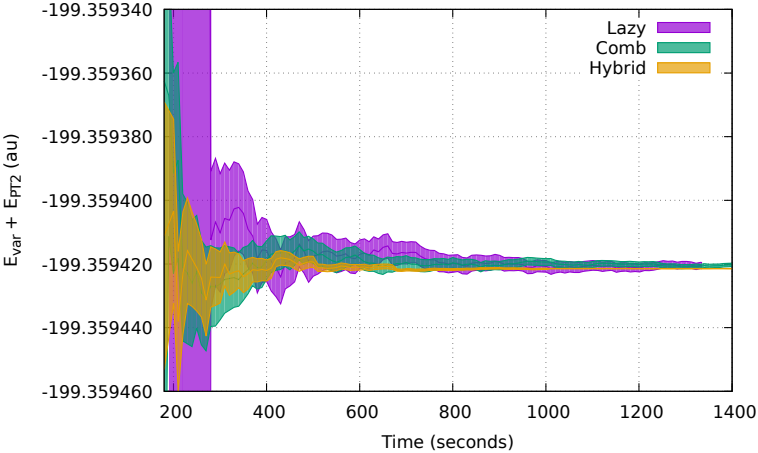


Figure: F_2, cc-pVDZ, 10^6 determinants in the variational space

Hybrid deterministic/stochastic scheme

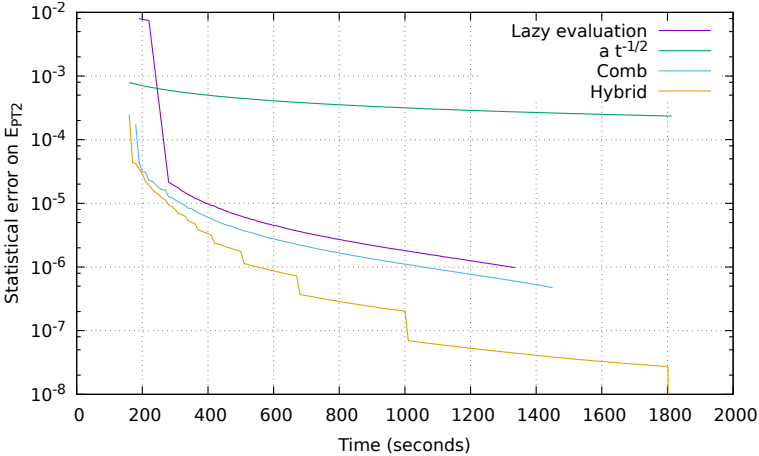
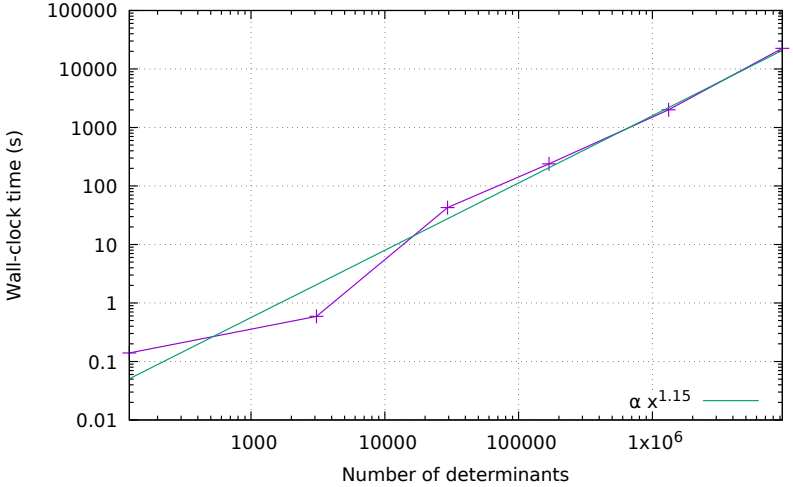


Figure: F_2, cc-pVDZ, 10^6 determinants in the variational space

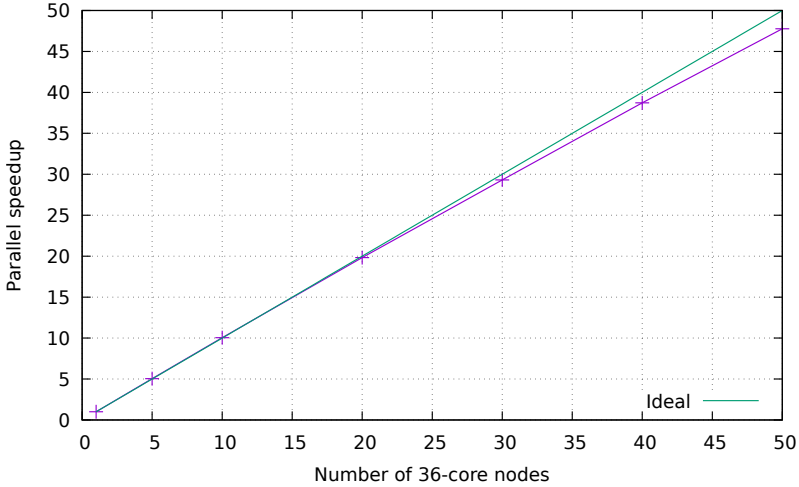
Some timings: Cr_2, $2\,10^7$ determinants, 800 cores

Basis	E_{PT2}	Wall-clock time
cc-pVDZ	$-0.068\,3(1)$	14 min
	$-0.068\,36(1)$	55 min
	$-0.068\,361(1)$	2.4 hr
	$-0.068\,360\,604$	3 hr
cc-pVTZ	$-0.124\,4(5)$	19 min
	$-0.124\,7(1)$	58 min
	$-0.124\,63(1)$	3.5 hr
	$-0.124\,642(1)$	8.7 hr
	—	$\sim$ 15 hr (estimated)
cc-pVQZ	$-0.155\,8(5)$	56 min
	$-0.155\,9(1)$	2.5 hr
	$-0.155\,95(1)$	9.0 hr
	$-0.155\,952(1)$	18.5 hr
	—	$\sim$ 29 hr (estimated)

Scaling with N_{det}



Parallel efficiency



Memory

- There is no memory bottleneck with PT2
- The $|\alpha\rangle$ determinants are never stored