Design of the `molsturm` quantum-chemistry framework

Towards method development for arbitrary basis functions

Michael F. Herbst[⋆], Andreas Dreuw[⋆], James E. Avery[†]

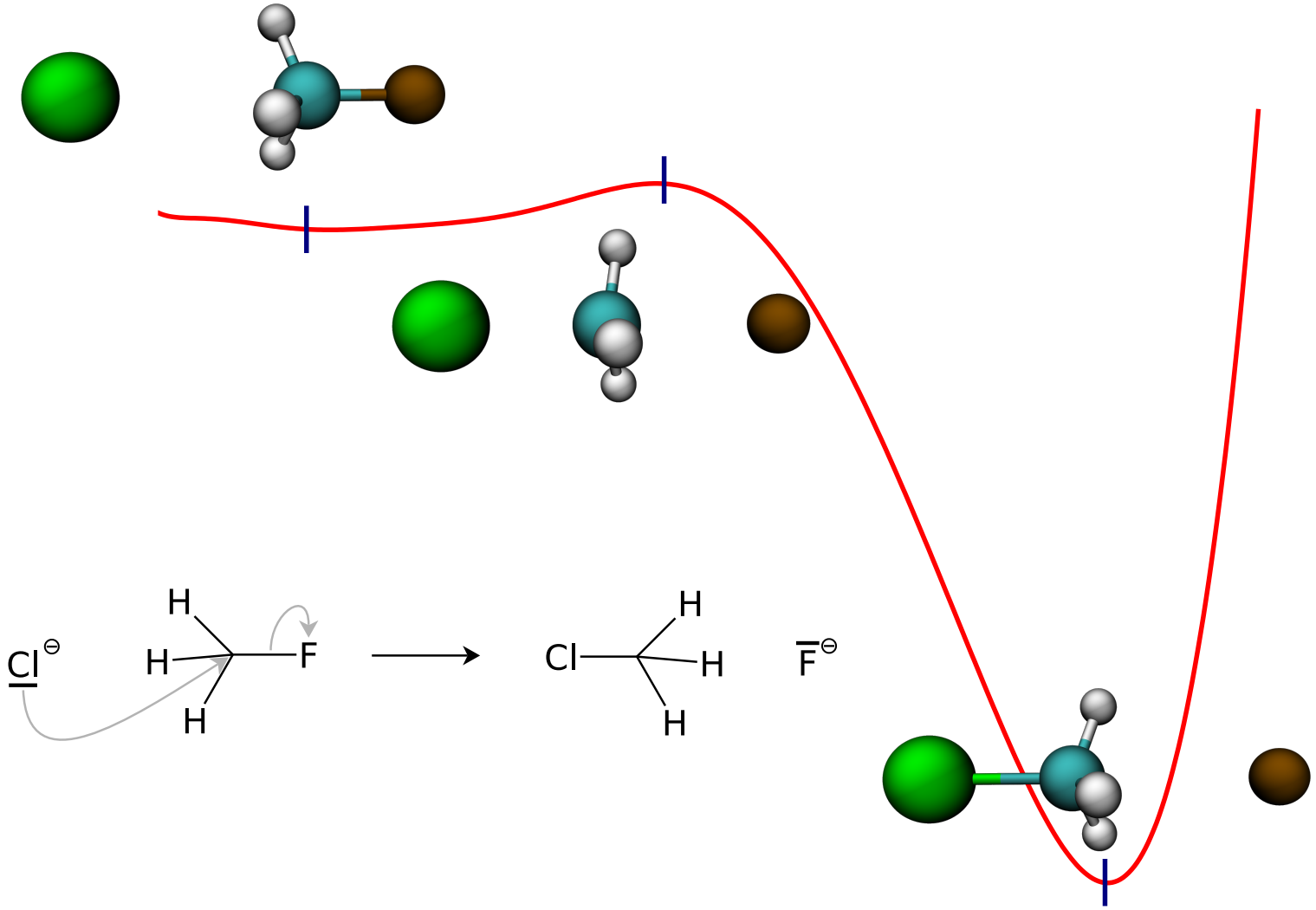
[⋆]Ruprecht-Karls-Universität Heidelberg
[†]Niels Bohr Institutet, København

23th November 2018

Contents

Contents

Describing chemistry



Electronic structure theory

- Goal: Describing chemical reactivity / properties
- Electronic Schrödinger equation:

$$\hat{\mathcal{H}}\Psi_i = E_i\Psi_i$$

- Operator $\hat{\mathcal{H}}$: Physical description of molecular system
- Most important: Ground state, i.e. E_0 and Ψ_0

Solving the Schrödinger equation: How hard can it be?

- Main ingredient: Min-max principle

$$E_0 \leq \min_{\Psi \in S} \mathcal{E}(\Psi) = \min_{\Psi \in S} \frac{\left\langle \Psi \middle| \hat{\mathcal{H}} \Psi \right\rangle}{\langle \Psi | \Psi \rangle}$$

where $S \subset Q\left(\hat{\mathcal{H}}\right) = H^1(\mathbb{R}^{3N}, \mathbb{C})$ and $L^2(\mathbb{R}^{3N}, \mathbb{C})$ inner product $\langle \cdot | \cdot \rangle$

- Discretisation: Curse of dimensionality:
  - $\langle \cdot | \cdot \rangle_N$ involves integral over $3N$-dim. space
  - Assume 2 quadrature points only
  - Chloromethane: $N = 26 \Rightarrow 2^{78} \approx 3 \cdot 10^{23}$ quadrature points

Now what?

- Need a suitable inexact model
- One starting point: Hartree-Fock approximation
  - Find best subspace

$$S = \left\{ \bigwedge_{i=1}^{N} \psi_i \,\middle|\, \psi_i \in H^1(\mathbb{R}^3, \mathbb{R}), \langle \psi_i | \psi_j \rangle_1 = \delta_{ij}, \forall\, 1 \leq i, j \leq N \right\}$$

where $\psi_i \in H^1(\mathbb{R}^3, \mathbb{R})$ are single-particle functions.

$\Rightarrow$ Minimisation problem for $\{\psi_i\}_i$

- Afterwards:
  - Approximate missing physics using minimiser $\Theta^0 = \{\psi_i^0\}_i$

$\Rightarrow$ Post-HF methods

Solving Hartree-Fock

- Euler-Lagrange equations:

$$\hat{\mathcal{F}}_{\Theta^0}\psi_i^0 = \varepsilon_i \psi_i^0 \qquad\qquad \left\langle \psi_i^0 \middle| \psi_j^0 \right\rangle_1 = \delta_{ij}$$

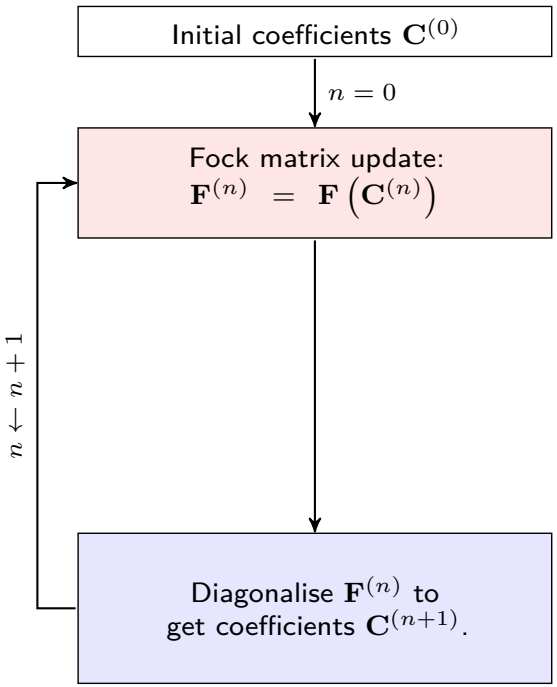
- Discretise $\hat{\mathcal{F}}_{\Theta^0}$ in a basis $\{\varphi_\mu\}_\mu$:

$$F_{\mu\nu} = \left\langle \varphi_\mu \middle| \Theta^0 \varphi_\nu \right\rangle_1$$
$$C_{\mu,i} = \langle \varphi_\mu | \psi_i \rangle_1$$

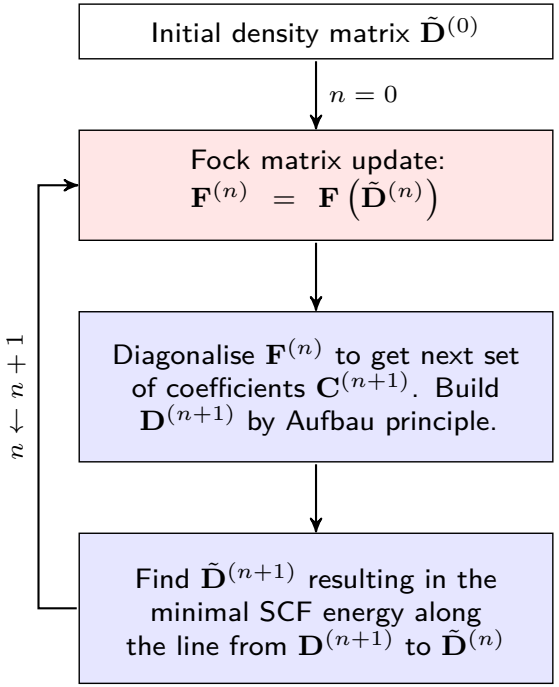
- Discretised problem:

$$\mathbf{F}[\mathbf{C}]\,\mathbf{C} = \mathbf{S}\mathbf{C}\ \operatorname{diag}(\varepsilon_1, \ldots, \varepsilon_n)$$

Self-consistent field procedure



Roothaan scheme

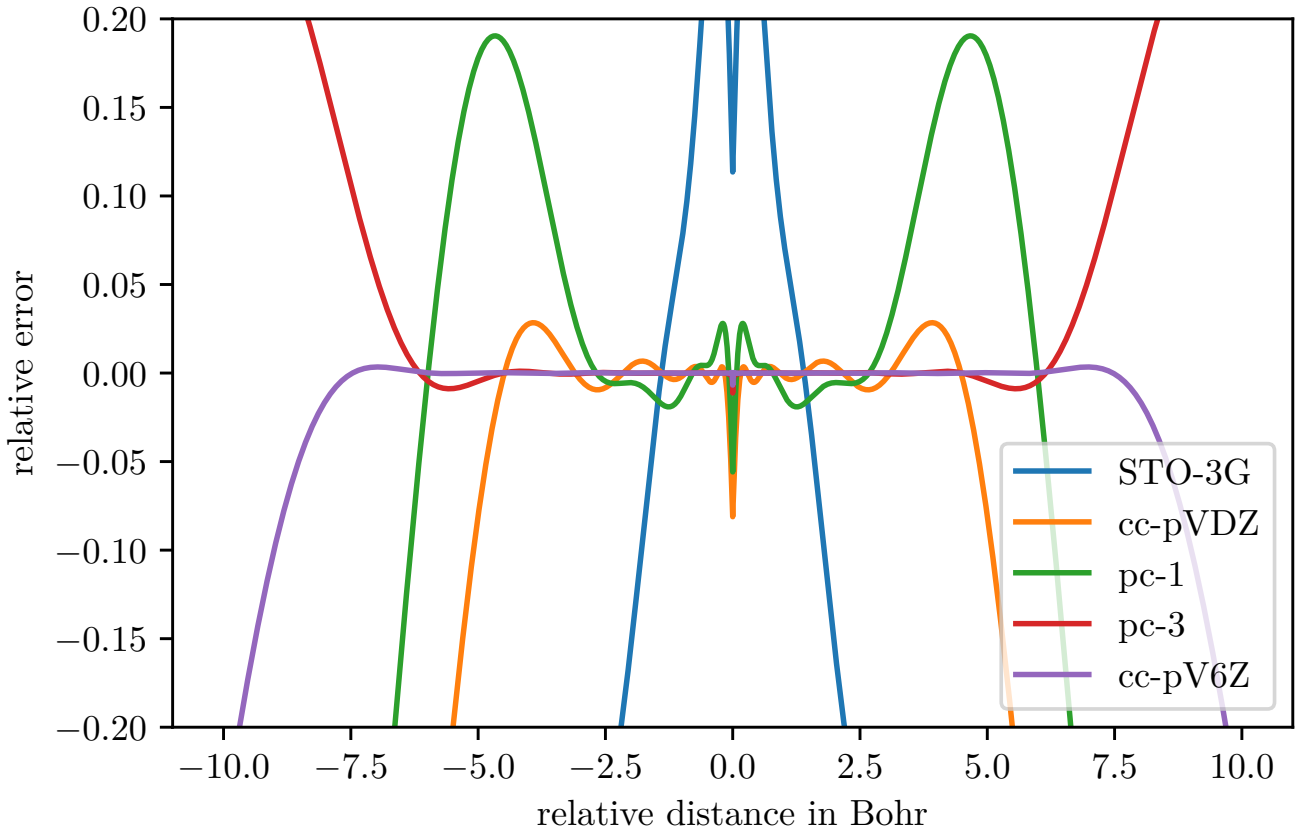


Optimal damping algorithm

Contents

The standard basis: Gaussian-type basis sets

$$\varphi_\mu^{\mathsf{GTO}}(\underline{\boldsymbol{r}}) = r^{l_\mu} \sum_i^{N_{\mathsf{contr}}} c_{\mu,i} \exp(-\alpha_{\mu,i} r^2) \cdot Y_{l_\mu}^{m_\mu}(\hat{\underline{\boldsymbol{r}}})$$



Beyond Gaussian-type orbitals

- Issues of cGTOs:
  - Representing the core region?
  - Representing the exponential decay?
  - Error estimates?
  - Black box modelling?
  - Distributed memory parallelisation?

$\Rightarrow$ Playing field to try other basis function types

Basis function types

- Gaussian-type orbitals
- Geminals

- Slater-type orbitals
- Sturmian-type orbitals

- ...

- Plane waves
- Augmented plane waves

- Wavelets
- Finite elements

- ...

Testing alternative basis function types

- Obstacle: 1 Program $\simeq$ 1 basis function type

$\Rightarrow$ Basis type often burned into existing codes

$\Rightarrow$ A new program for each basis type just to try it?

- Structure of discretised SCF problem:

$$\mathbf{FC} = \mathbf{SC}\ \mathrm{diag}(\varepsilon_1, \ldots, \varepsilon_n)$$

$\Rightarrow$ Independent of basis choice

$\Rightarrow$ It should be sufficient to swap the integral backends!

Testing alternative basis function types

- Obstacle: 1 Program $\simeq$ 1 basis function type

$\Rightarrow$ Basis type often burned into existing codes

$\Rightarrow$ A new program for each basis type just to try it?

- Structure of discretised SCF problem:

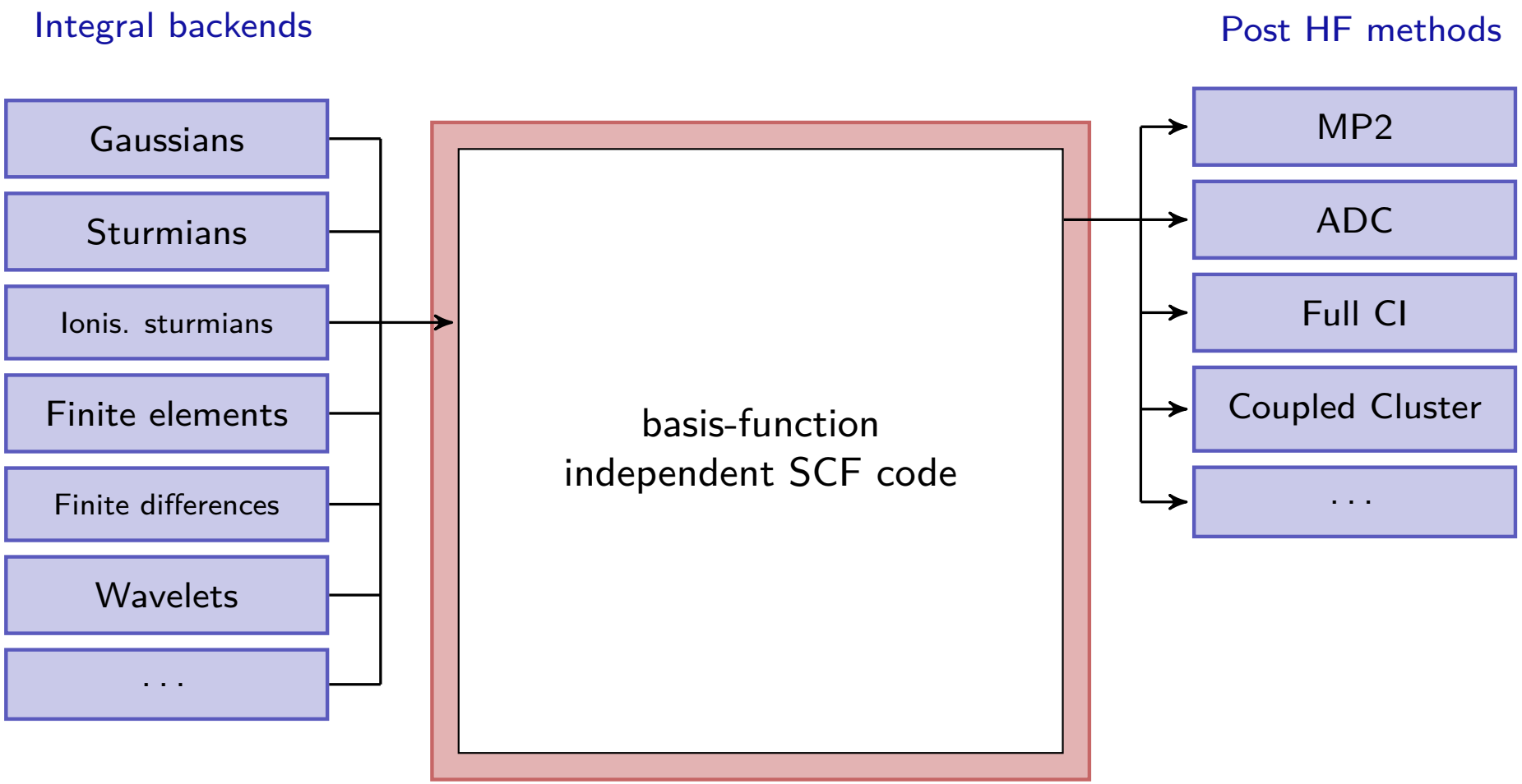
$$\mathbf{FC} = \mathbf{SC}\ \mathrm{diag}(\varepsilon_1, \ldots, \varepsilon_n)$$

$\Rightarrow$ Independent of basis choice

$\Rightarrow$ It should be sufficient to swap the integral backends!

Contents

Aims of molsturm



Achievements of molsturm

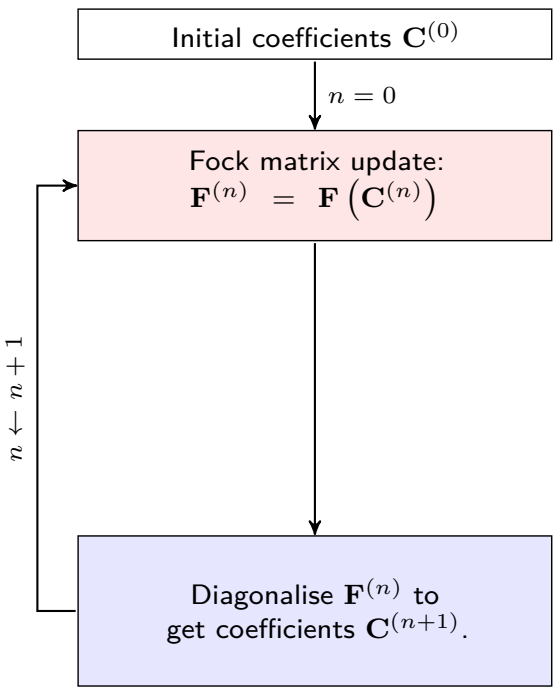
- Basis-function independent design
  - Plug and play new discretisations
  - Basis-type agnostic SCF procedure
- Easy-to-use interfaces
  - Integrate with existing code (e.g. Post-HF)
  - Avoid reinventing the wheel
  - Rapid prototyping, testing and analysis

⇒ Explore methods across basis function types[1,2]

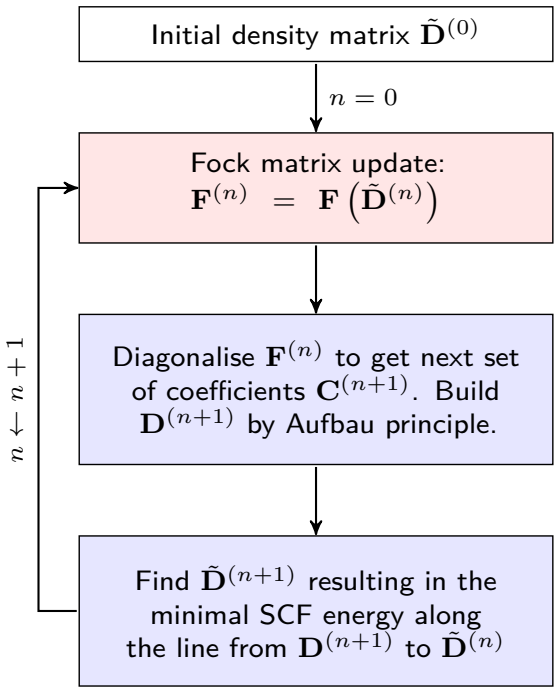
[1] M. F. Herbst, A. Dreuw and J. E. Avery. J. Chem. Phys., **149**, 84106 (2018)

[2] M. F. Herbst. Ph.D. thesis, Ruprecht-Karls-Universität Heidelberg (2018)

Two-step structure of SCF algorithms



Roothaan scheme

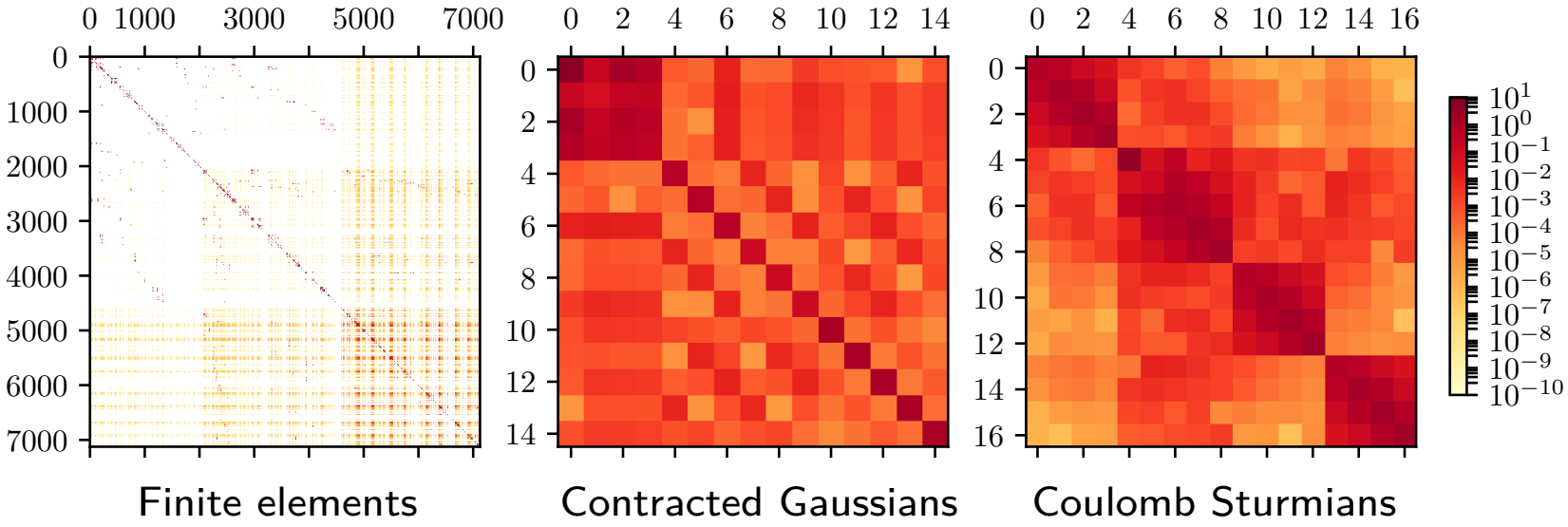


Optimal damping algorithm

- Fock update
- Coefficient update
  (density matrix update)

$\Rightarrow$ Need to be basis-type independent

Challenge: Deviating Fock matrix structures



- Required numerical procedures differ
- Details should be hidden from SCF
- Focus on HF, but our approach extends to DFT

Solution: Contraction-based methods

- Contraction-based methods
  - Avoid storing matrices
  - Employ iterative, subspace-based algorithms
  - Contraction expressions (e.g. matrix-vector products)
  - Common in Post-HF: *Working equations*

$\Rightarrow$ SCF code only needs Fock contraction

$\Rightarrow$ Hide discretisation details inside Fock object

$\Rightarrow$ Flexible to exploit discretisation-specific properties

Contraction-based methods: Flexibility

- Compare

1. $$K_{\kappa\lambda} = \sum_{\mu\nu,i} \langle \kappa\nu || \mu\lambda \rangle \, C_{\mu,i} C_{\nu,i}$$
2. $$y_\kappa = \sum_\lambda K_{\kappa\lambda} x_\lambda$$

with directly

$$y_\kappa = \sum_{\lambda\mu\nu,i} \langle \kappa\nu || \mu\lambda \rangle \, C_{\mu,i} C_{\nu,i} x_\lambda$$

- Reordering terms
- Exploit known symmetries in x_λ, $\langle \kappa\nu || \mu\lambda \rangle$
- Exploit index selection rules
- $\mathbf{K}$ like a matrix with state $\mathbf{C}$

Contraction-based methods: Overview

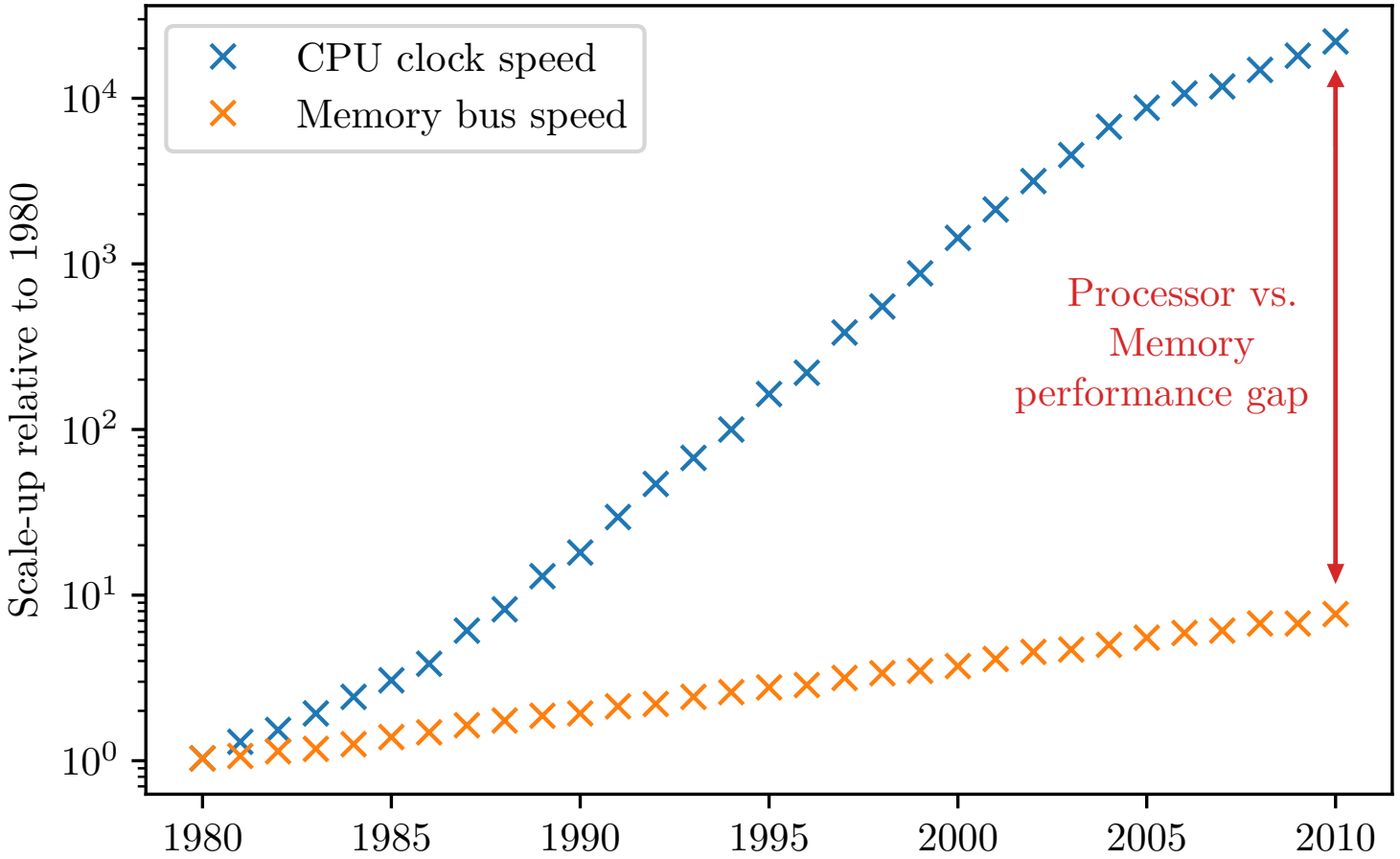
Advantages

- Maximum flexibility at the point of evaluation
- Parallelisation easier
  - ⇒ Less data management
  - ⇒ Easier modularisation
- Hardware trends are in favour

Disadvantages

- Matrices more intuitive than contraction-functions
- More computations
  - ⇒ Need efficient contraction schemes for the contraction
  - ⇒ Algorithms more complex

Contraction-based methods: Overview



Data from `https://dave.cheney.net/2014/06/07/five-things-that-make-go-fast`

Contraction-based methods: Overview

Storage layer	Latency /ns	FLOPs
L1 cache	0.5	13
L2 cache	7	180
Main memory	100	2600
SSD read	$1.5 \cdot 10^4$	$4 \cdot 10^5$
HDD read	$1 \cdot 10^7$	$3 \cdot 10^8$

Data from
`https://people.eecs.berkeley.edu/~rcs/research/interactive_latency.html`
FLOPs for a Sandy Bridge 3.2GHz CPU with perfect pipelining

Contraction-based methods: Overview

Advantages

- Maximum flexibility at the point of evaluation
- Parallelisation easier
  - ⇒ Less data management
  - ⇒ Easier modularisation
- Hardware trends are in favour

Disadvantages

- Matrices more intuitive than contraction-functions
- More computations
  - ⇒ Need efficient contraction schemes for the contraction
  - ⇒ Algorithms more complex

Lazy matrices

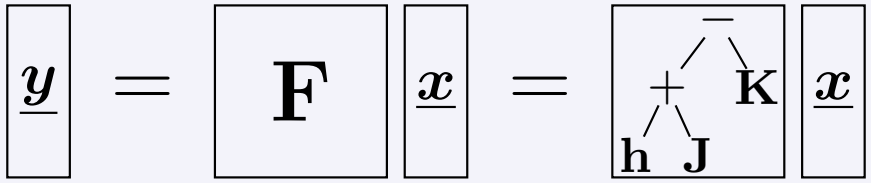
- Contraction expressions dressed as a matrix (physical intuition)
- Build and pass Fock expression tree to SCF
- Lazy evaluation:

$$\mathbf{F} = \mathbf{h} + \mathbf{J} - \mathbf{K}$$

$$\vdots$$

Iterative solver

$$\underline{\boldsymbol{y}} = \mathbf{F}\underline{\boldsymbol{x}}$$



- Idea: Integral back end provides lazy matrix terms

Contraction-based, two-step SCF

Fock expression	Lazy matrix, sum of integral terms
Coefficient update	Iterative solvers
Fock update	Replace coefficients in expression tree

Achieve basis-function independence:

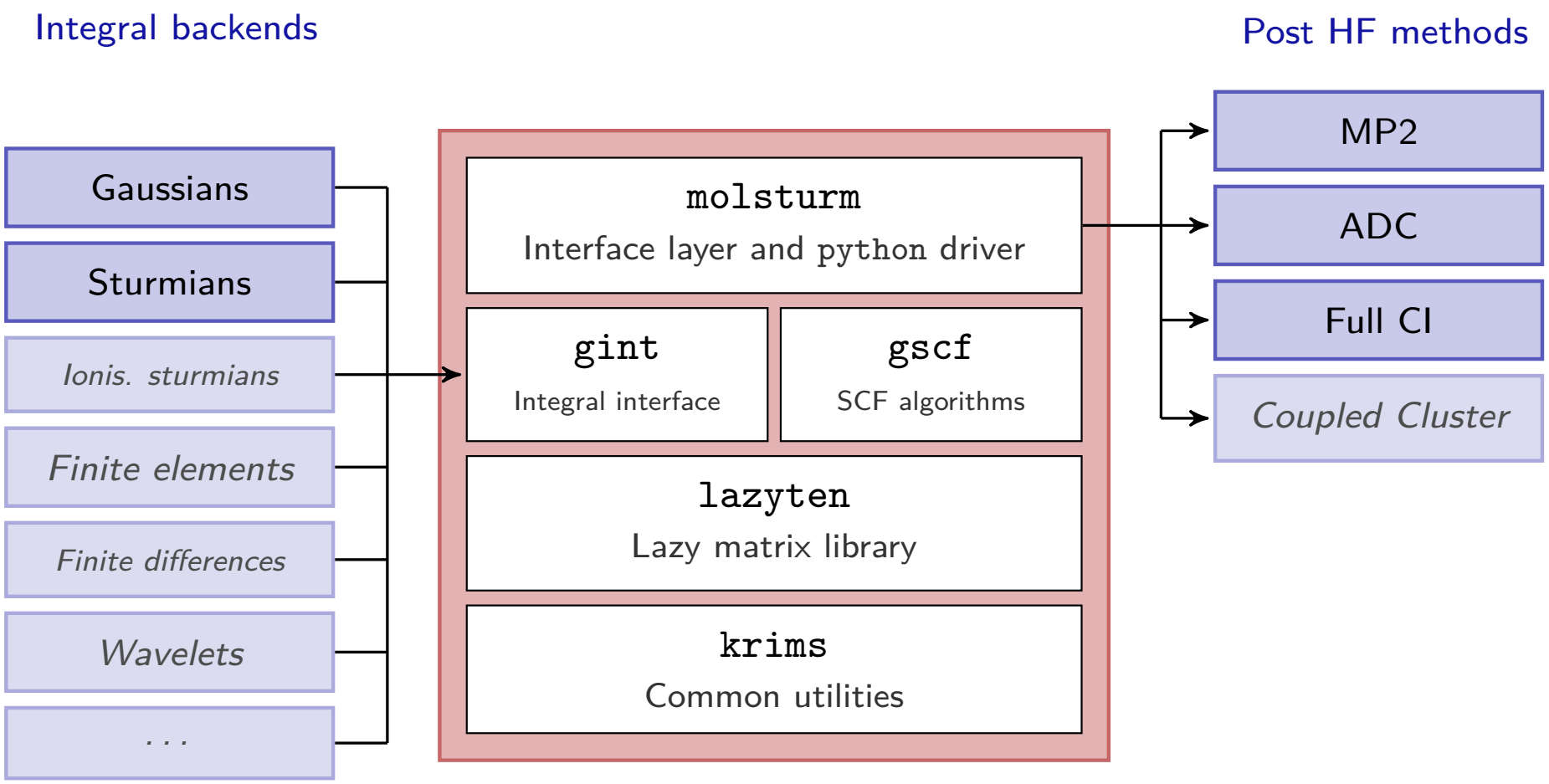
- Lazy matrices: Abstraction between integrals and SCF
- Integral back end: Controls evaluation of contractions

⇒ Decides integral data production and consumption

⇒ Transparent to SCF

⇒ May exploit discretisation-specific properties

molsturm structure



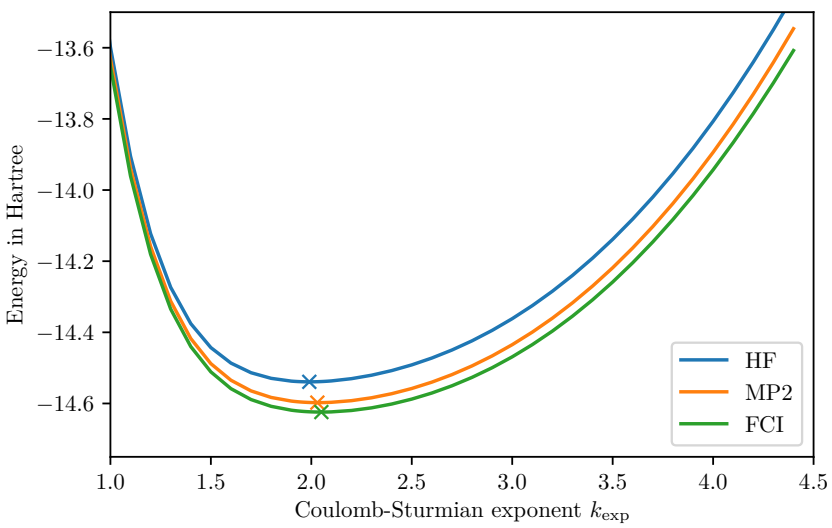
molsturm interface: CCD residual (parts)

$$
\begin{aligned}
r_{ij}^{ab} = &-\frac{1}{2}\sum_{mnef}\langle mn||ef\rangle\, t_{mn}^{af}\, t_{ij}^{eb} + \frac{1}{2}\sum_{mnef}\langle mn||ef\rangle\, t_{mn}^{bf}\, t_{ij}^{ea} - \frac{1}{2}\sum_{mnef}\langle mn||ef\rangle\, t_{in}^{ef}\, t_{mj}^{ab} \\
&+\frac{1}{2}\sum_{mnef}\langle mn||ef\rangle\, t_{jn}^{ef}\, t_{mi}^{ab} + \frac{1}{4}\sum_{mnef}\langle mn||ef\rangle\, t_{mn}^{ab}\, t_{ij}^{ef} + \frac{1}{2}\sum_{mnef}\langle mn||ef\rangle\, t_{im}^{ae}\, t_{jn}^{bf} \\
&-\frac{1}{2}\sum_{mnef}\langle mn||ef\rangle\, t_{jm}^{ae}\, t_{in}^{bf} - \frac{1}{2}\sum_{mnef}\langle mn||ef\rangle\, t_{im}^{be}\, t_{jn}^{af} + \frac{1}{2}\sum_{mnef}\langle mn||ef\rangle\, t_{jm}^{be}\, t_{in}^{af}
\end{aligned}
$$

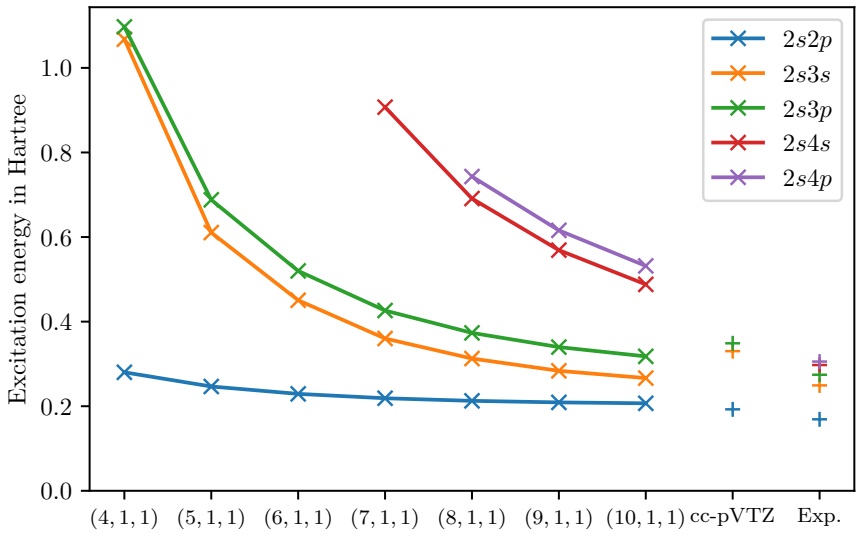
```
eri_phys = state.eri.transpose((0, 2, 1, 3))
eri = eri_phys - eri_phys.transpose((1, 0, 2, 3))
res = \
    - 0.5  * einsum("mnef,manf,iejb->iajb", eri.block("oovv"), t2, t2) \
    + 0.5  * einsum("mnef,mbnf,ieja->iajb", eri.block("oovv"), t2, t2) \
    - 0.5  * einsum("mnef,ienf,majb->iajb", eri.block("oovv"), t2, t2) \
    + 0.5  * einsum("mnef,jenf,maib->iajb", eri.block("oovv"), t2, t2) \
    + 0.25 * einsum("mnef,manb,iejf->iajb", eri.block("oovv"), t2, t2) \
    + 0.5  * einsum("mnef,iame,jbnf->iajb", eri.block("oovv"), t2, t2) \
    - 0.5  * einsum("mnef,jame,ibnf->iajb", eri.block("oovv"), t2, t2) \
    - 0.5  * einsum("mnef,ibme,janf->iajb", eri.block("oovv"), t2, t2) \
    + 0.5  * einsum("mnef,jbme,ianf->iajb", eri.block("oovv"), t2, t2)
```

molsturm interface: Linked codes

Coulomb-Sturmian based MP2 and FCI



Coulomb-Sturmian and Gaussian based ADC(2)



- FCI from `pyscf`[1]
- Coulomb Sturmians from `sturmint`[2]

- ADC(2) from `adcman`[3]
- Gaussians from `libint`[4] or `libcint`[5]

[1] Q. Sun et al. WIREs Comput Mol Sci, **8**, e1340 (2017).

[2] J. E. Avery and M. F. Herbst. `https://molsturm.org/sturmint` (2018)

[3] M. Wormit et al. Mol. Phys., **112**, 774 (2014).

[4] E. Valeyev et al. *evaleev/libint: 2.3.1* (2017).

[5] Q. Sun. J. Comput. Chem., **36**, 1664 (2015)

Contents

Coulomb Sturmians

- Iso-energetic solutions φ_{nlm} to hydrogen-like equation[1]

$$\left(-\frac{1}{2}\Delta - \beta_n \frac{Z}{r}\right)\varphi_{nlm}(\underline{\boldsymbol{r}}) = E\varphi_{nlm}(\underline{\boldsymbol{r}})$$

- Scaling factor β_n chosen to uniform energy:

$$\beta_n = \frac{kn}{Z} \quad \Rightarrow \quad E = -\frac{k^2}{2}$$

- φ_{nlm} look like hydrogenic orbitals with $\frac{Z}{n}$ replaced by k
- Radial part R_{nl} satisfies

$$\left(-\frac{1}{2r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial}{\partial r}\right) + \frac{l(l+1)}{2r^2} - \frac{nk}{r} - E\right)R_{nl}(r) = 0.$$

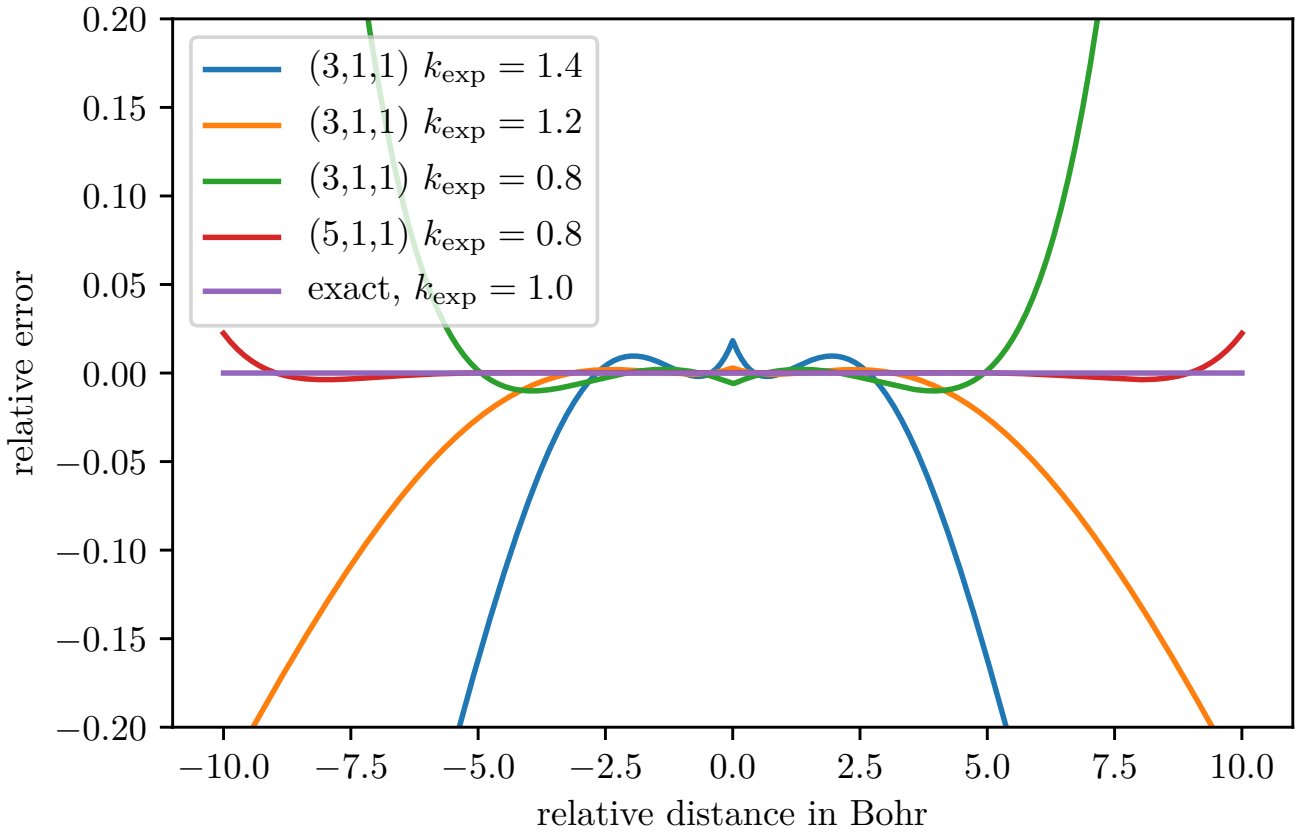
$\Rightarrow$ Sturm-Liouville equation[2]

[1] H. Shull and P.-O. Löwdin. J. Chem. Phys., **30**, 617 (1959)

[2] M. Rotenberg. Ann. Phys., **19**, 262 (1962)

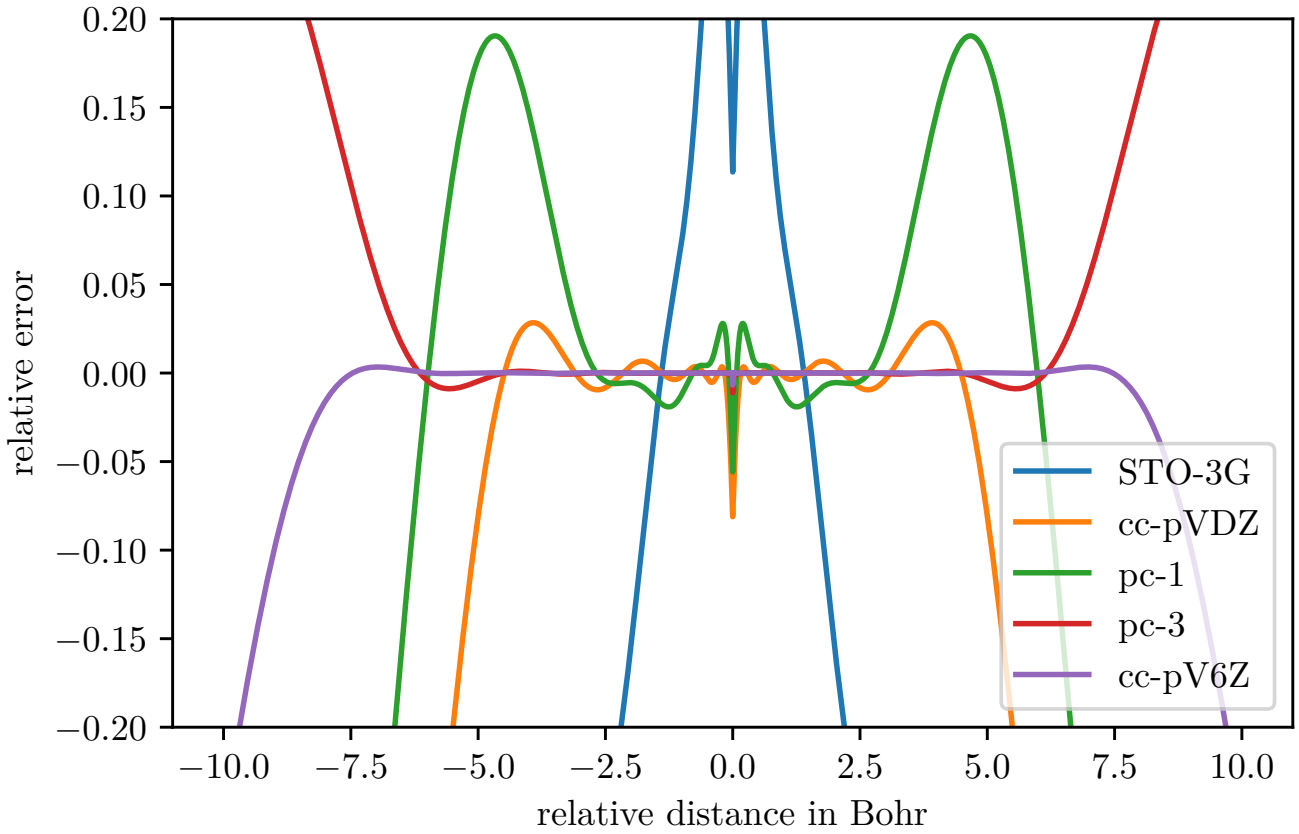
Coulomb-Sturmians vs Gaussians (for hydrogen)

$$\varphi_\mu^{\mathsf{CS}}(\underline{\boldsymbol{r}}) = P_{nl}(r)\exp(-kr)\cdot Y_l^m(\hat{\underline{\boldsymbol{r}}})$$



Coulomb-Sturmians vs Gaussians (for hydrogen)

$$\varphi_\mu^{\mathsf{GTO}}(\underline{\boldsymbol{r}}) = r^{l_\mu} \sum_i^{N_{\mathsf{contr}}} c_{\mu,i} \exp(-\alpha_{\mu,i} r^2) \cdot Y_{l_\mu}^{m_\mu}(\hat{\underline{\boldsymbol{r}}})$$



Construction of CS basis sets

- All CS basis functions share common exponent k
- Index restrictions by CS equations:

$$n > 0 \qquad 0 \leq l < n \qquad -l \leq m \leq l$$

- Basis set construction:[1] Additional restrictions

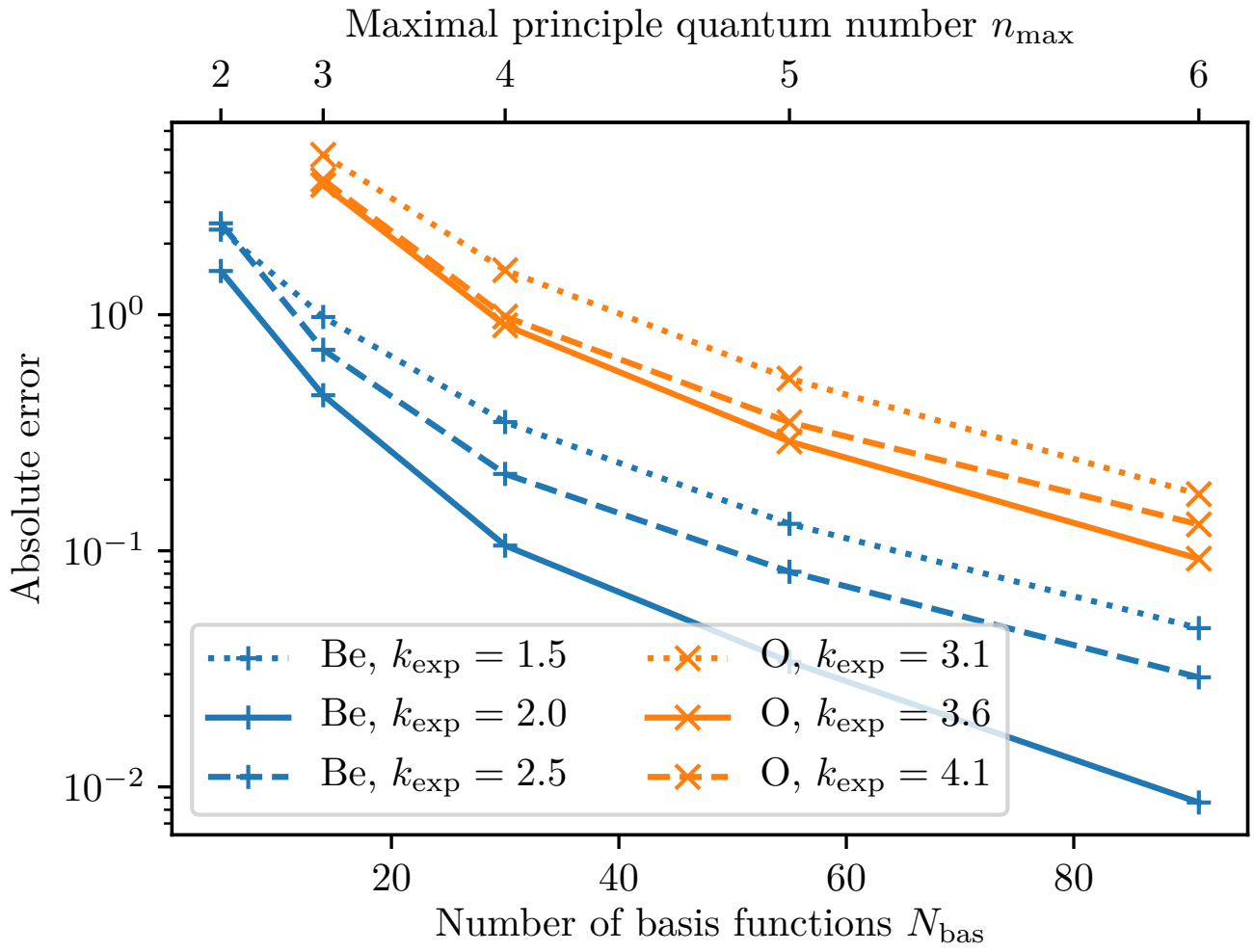
$$n \leq n_{\text{max}} \qquad l \leq l_{\text{max}} \qquad m \leq m_{\text{max}}$$

- Restriction by m_{max}, sometimes l_{max} neglected

$\Rightarrow$ Physically motivated

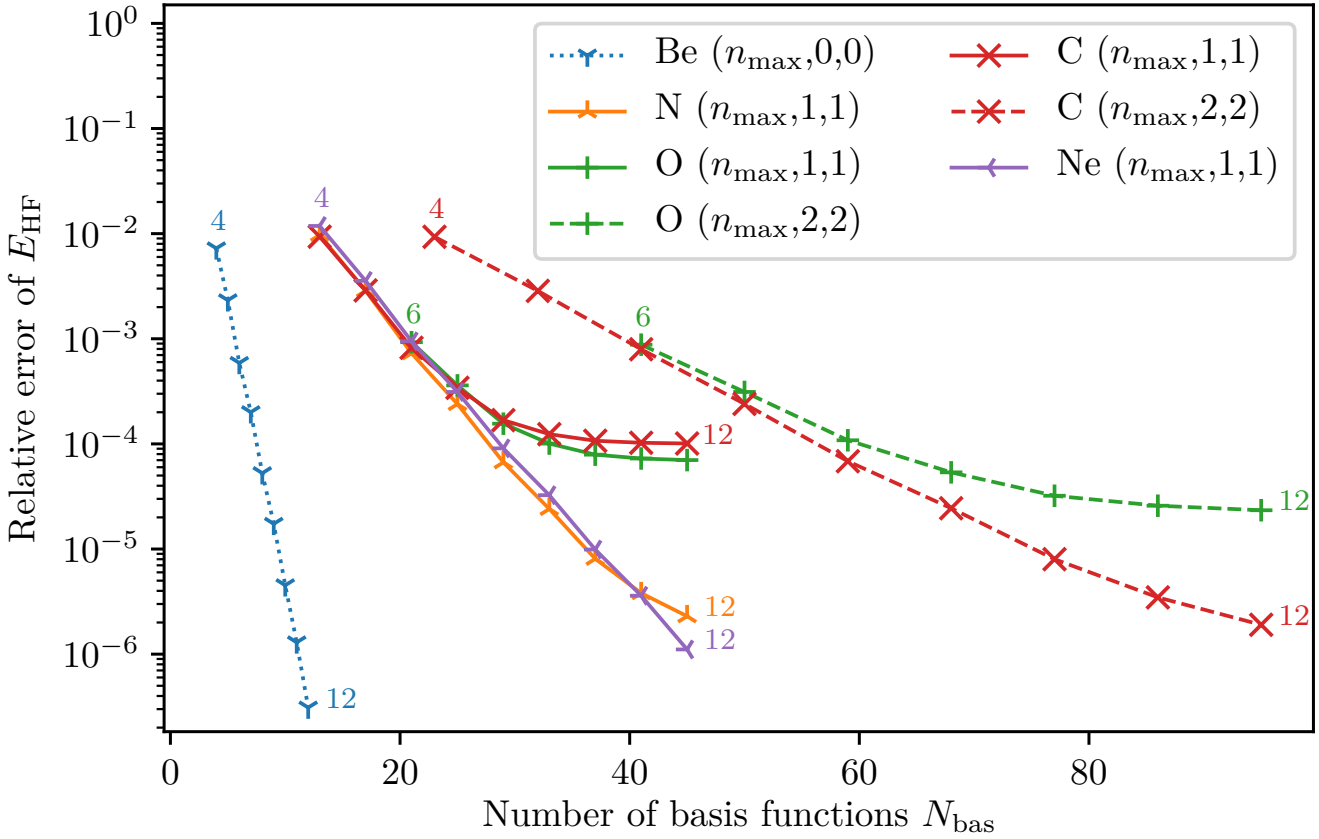
[1] M. F. Herbst, J. E. Avery and A. Dreuw (2018). arXiv: 1811.05777

Energy convergence[1]



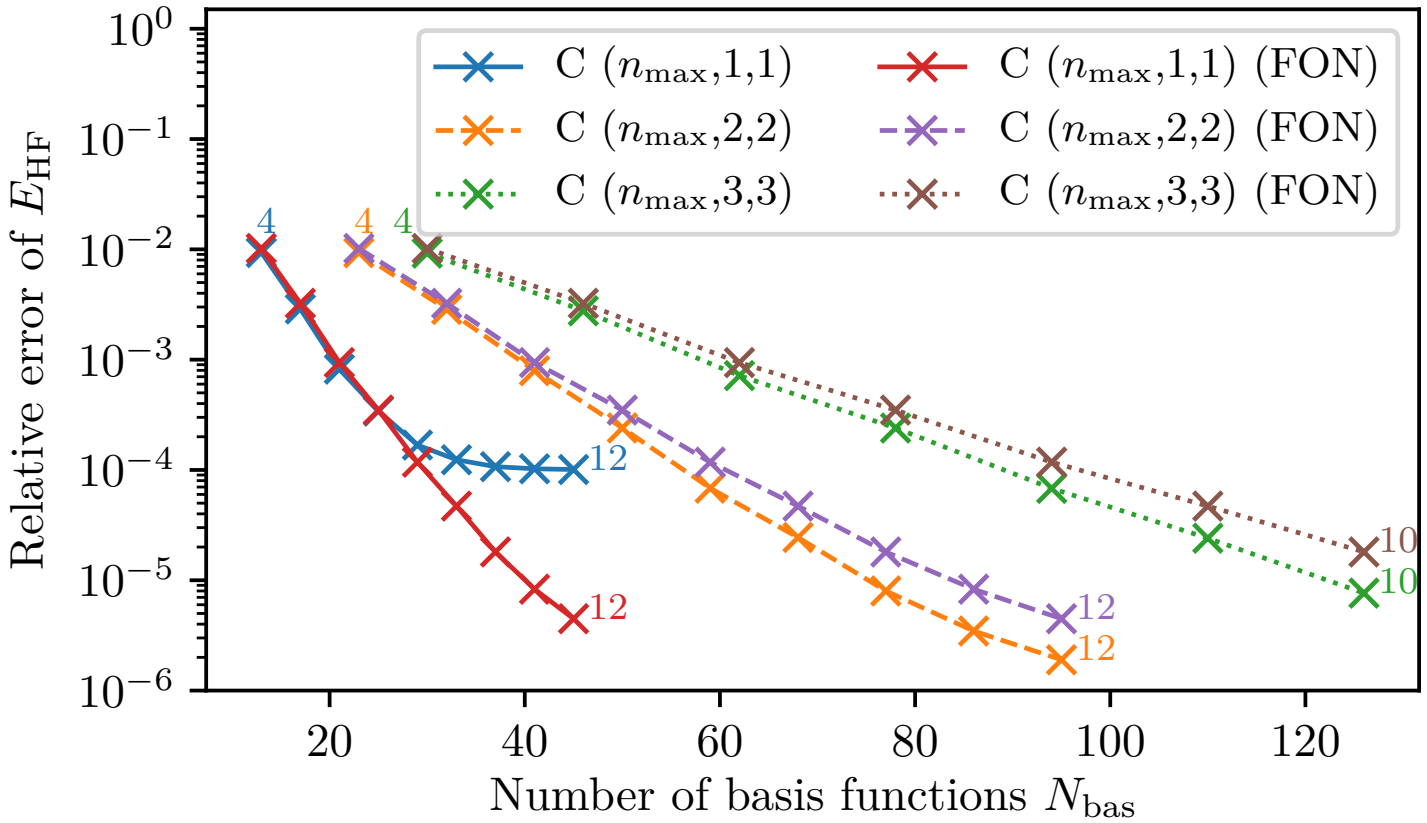
[1] M. F. Herbst, J. E. Avery and A. Dreuw (2018). arXiv: 1811.05777

Basis progressions: Energy versus basis size[1]



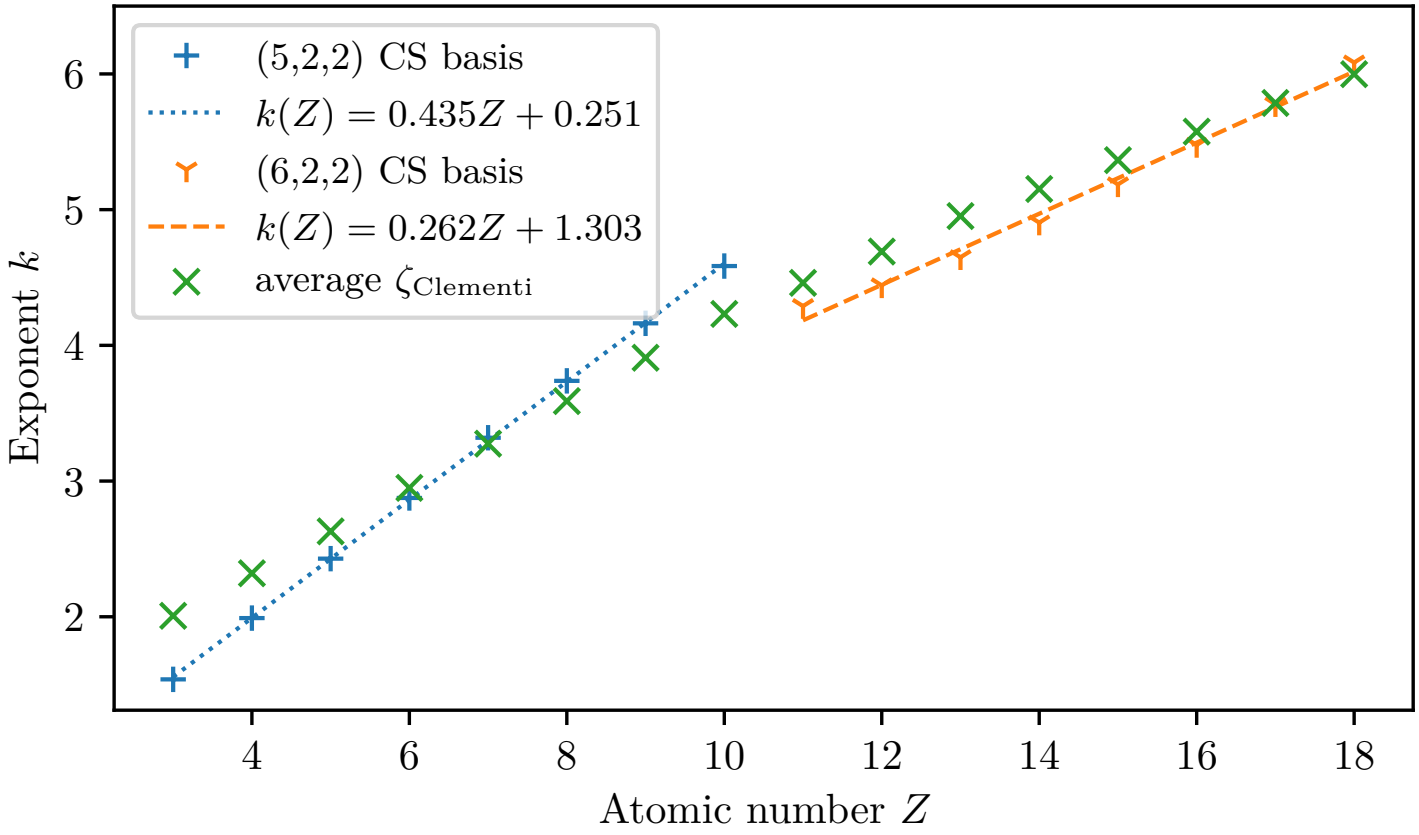
[1] M. F. Herbst, J. E. Avery and A. Dreuw (2018). arXiv: 1811.05777

Basis progressions: Carbon[1]



[1] M. F. Herbst, J. E. Avery and A. Dreuw (2018). arXiv: 1811.05777

Optimal exponent versus Clementi exponents[1]



[1] M. F. Herbst, J. E. Avery and A. Dreuw (2018). arXiv: 1811.05777

molsturm[2] in essence

- Contraction-based, basis-function independent SCF
- Mediator: Integral libraries $\leftrightarrow$ Post-HF
- Modular and light-weight structure

$\Rightarrow$ Make use of what exists (Post-HF or python)

- Plug-and-play new basis-function types
- Allow apples-to-apples comparison
- Rapid prototyping of new methods

- Application: Convergence of Coulomb-Sturmians basis sets[1]

[1] M. F. Herbst, J. E. Avery and A. Dreuw (2018). arXiv: 1811.05777

[2] M. F. Herbst, A. Dreuw and J. E. Avery. J. Chem. Phys., **149**, 84106 (2018)

Acknowledgements

James Avery

Andreas Dreuw

Adrian Dempwolff

Guido Kanschat

Maximilian Scheurer

UNIVERSITÄT HEIDELBERG ZUKUNFT SEIT 1386

IWR Interdisciplinary Center for Scientific Computing

HGS MathComp

DAAD

Questions?

Papers: M. F. Herbst, A. Dreuw and J. E. Avery. J. Chem. Phys., **149**, 84106 (2018)

M. F. Herbst, J. E. Avery and A. Dreuw (2018). arXiv: 1811.05777

Thesis: `https://michael-herbst.com/phd-thesis.html`

Code: `https://molsturm.org`

Email: `michael.herbst@iwr.uni-heidelberg.de`

Blog: `https://michael-herbst.com/blog`

Lazy matrix evaluation

- Actual expression in source code

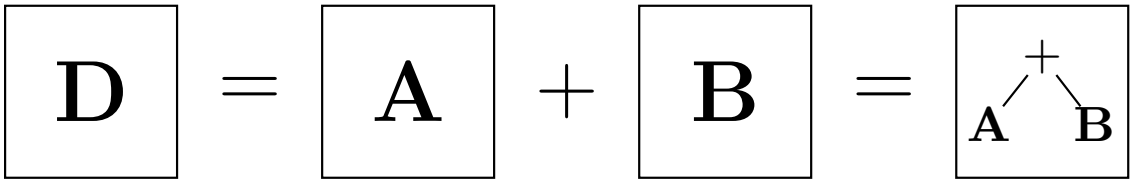
$$\mathbf{D} = \mathbf{A} + \mathbf{B},$$
$$\mathbf{E} = \mathbf{D}\mathbf{C},$$
$$\underline{\boldsymbol{y}} = \mathbf{E}\underline{\boldsymbol{x}},$$

Lazy matrix evaluation

- Actual expression in source code

$$\mathbf{D} = \mathbf{A} + \mathbf{B},$$
$$\mathbf{E} = \mathbf{D}\mathbf{C},$$
$$\underline{\boldsymbol{y}} = \mathbf{E}\underline{\boldsymbol{x}},$$

- Performed operation



Lazy matrix evaluation

- Actual expression in source code

$$\mathbf{D} = \mathbf{A} + \mathbf{B},$$
$$\mathbf{E} = \mathbf{D}\mathbf{C},$$
$$\underline{\boldsymbol{y}} = \mathbf{E}\underline{\boldsymbol{x}},$$

- Performed operation

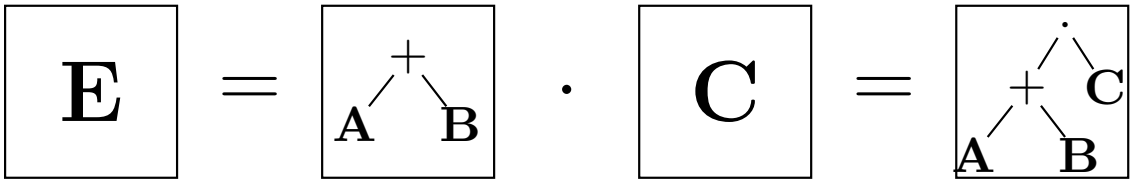
$$\boxed{\mathbf{E}} = \boxed{\mathbf{D}} \cdot \boxed{\mathbf{C}}$$

Lazy matrix evaluation

- Actual expression in source code

$$\mathbf{D} = \mathbf{A} + \mathbf{B},$$
$$\mathbf{E} = \mathbf{D}\mathbf{C},$$
$$\underline{\boldsymbol{y}} = \mathbf{E}\underline{\boldsymbol{x}},$$

- Performed operation



Lazy matrix evaluation

- Actual expression in source code

$$\mathbf{D} = \mathbf{A} + \mathbf{B},$$
$$\mathbf{E} = \mathbf{D}\mathbf{C},$$
$$\underline{\boldsymbol{y}} = \mathbf{E}\underline{\boldsymbol{x}},$$

- Performed operation

$$\boxed{\underline{\boldsymbol{y}}} = \boxed{\mathbf{E}}\,\boxed{\underline{\boldsymbol{x}}} = \boxed{\begin{tikzcd} & \cdot \arrow[dl, dash] \arrow[dr, dash] & \\ \mathbf{A} & + \arrow[l, dash] \arrow[r, dash] & \mathbf{C} \\ & \mathbf{B} \arrow[u, dash] & \end{tikzcd}}\,\boxed{\underline{\boldsymbol{x}}} = (\mathbf{A} + \mathbf{B})\,\mathbf{C}\,\underline{\boldsymbol{x}}$$

lazyten: Lazy matrix library