Introducing ONETEP

Part II - Efficient implementation of a parallel code

Chris-Kriton Skylaris

Peter D. Haynes

Arash A. Mostofi

Mike C. Payne

Electronic Structure Discussion Group

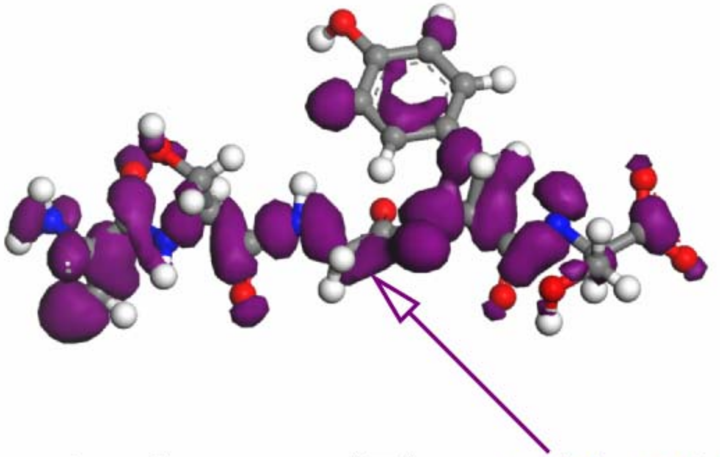
Theory of Condensed Matter, Cavendish Laboratory

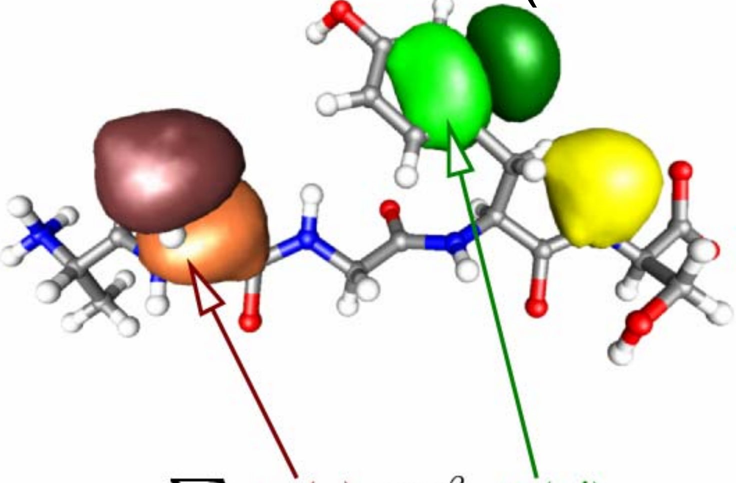
Cambridge, 19 May 2004

ONETEP: A density-matrix linear-scaling DFT method

Molecular orbitals (MOs)

Non-orthogonal Generalised Wannier Functions (NGWFs)



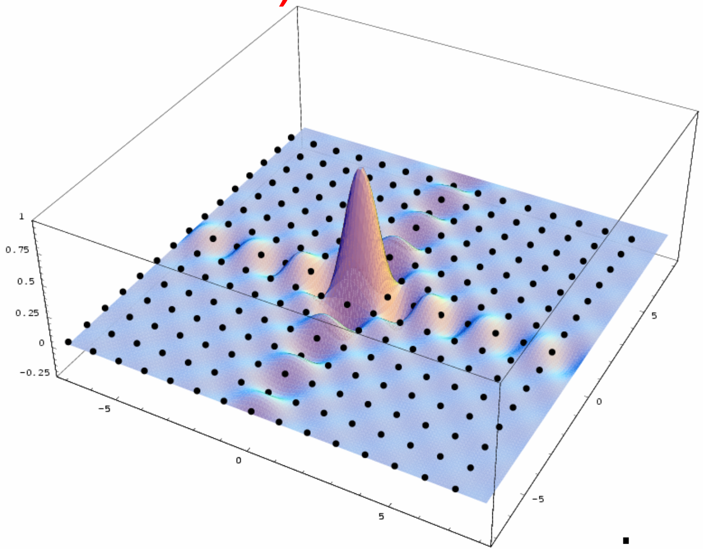


$$\rho(\mathbf{r},\mathbf{r}') = \sum_n f_n\, \psi_n(\mathbf{r})\, \psi_n^*(\mathbf{r}') = \sum_{\alpha\beta} \phi_\alpha(\mathbf{r})\, K^{\alpha\beta}\, \phi_\beta^*(\mathbf{r}')$$

- Optimise non–orthogonal localised functions $\{\phi_\alpha(\mathbf{r})\}$ instead of orthogonal extended wavefunctions $\{\psi_n(\mathbf{r})\}$ $\Big\}$ linear scaling
- Aim: to achieve the same accuracy as traditional plane–wave methods

PSINC basis set (=plane waves) for the NGWFs

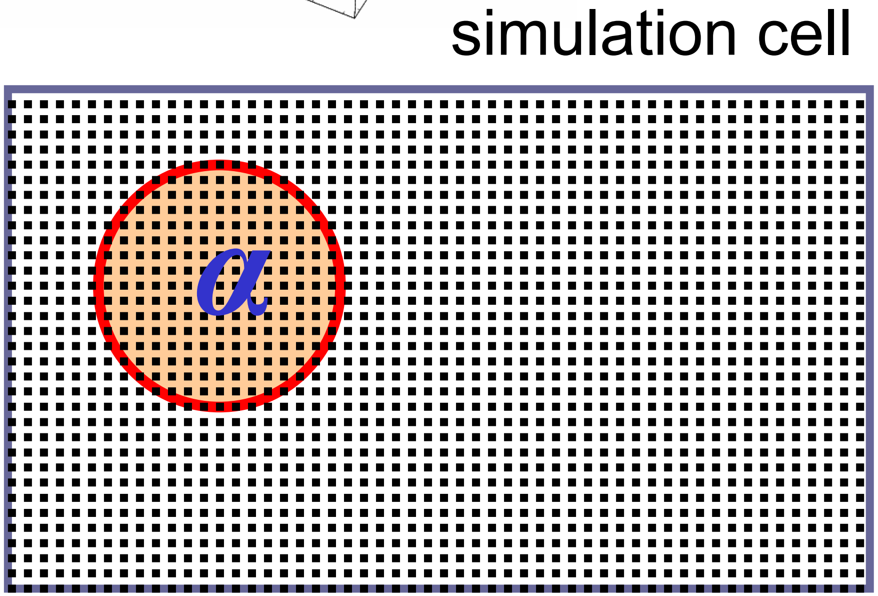
$$D(\mathbf{r}) = \frac{1}{N} \sum_{\mathbf{G}}^{\mathbf{G}_{max}} e^{i\mathbf{G}\cdot\mathbf{r}}$$



$$\phi_\alpha(\mathbf{r}) = \sum_{m}^{\text{sim. cell}} D_m(\mathbf{r}) C_{m\alpha}$$

$$= \sum_{m}^{\text{sim. cell}} D(\mathbf{r} - \mathbf{r}_m) C_{m\alpha}$$

$$C_{m\alpha} = 0 \text{ if } m \notin \text{sphere of } \alpha$$



DFT always computationally demanding – ONETEP O(N) scheme should take full advantage of parallel computers

$$E[n] = E[\{K^{\alpha\beta}\}, \{\phi_\alpha(\mathbf{r})\}] = E[\{K^{\alpha\beta}\}, \{C_{m\alpha}\}]$$

ONETEP two-nested-loop CG optimisation scheme

minimise $F[\{C_{m\alpha}\}]$ w.r.t. $\{C_{m\alpha}\}$

$$n(\mathbf{r}) = \sum_{\alpha\beta} \phi_\alpha(\mathbf{r}) K^{\alpha\beta} \phi^*_\beta(\mathbf{r})$$

$F[\{C_{m\alpha}\}] =$ minimise $E[\{K^{\alpha\beta}\}, \{C_{m\alpha}\}]$ w.r.t. $\{K^{\alpha\beta}\}$

keep the $\{C_{m\alpha}\}$ fixed

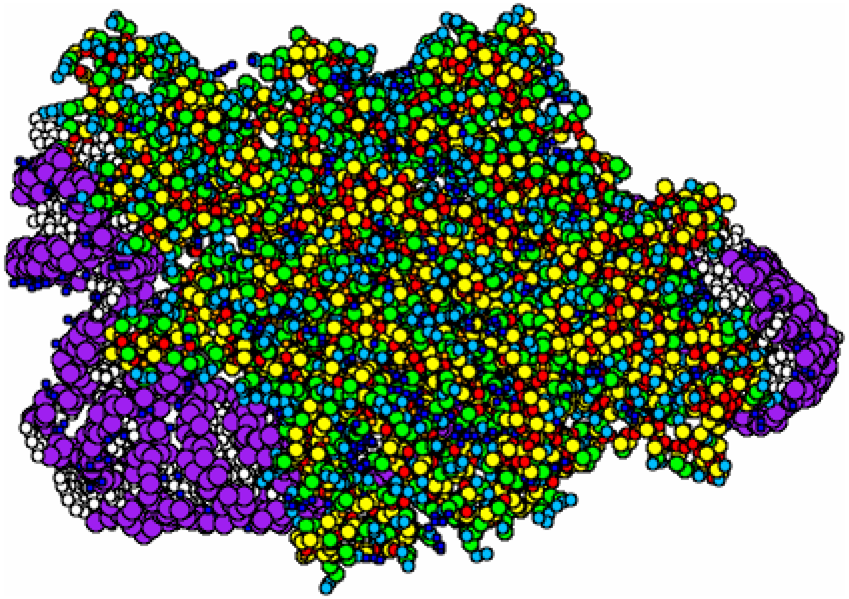
$$\frac{\partial E}{\partial K^{\alpha\beta}} \propto H_{\alpha\beta} = \langle \phi_\alpha | \hat{H} | \phi_\beta \rangle$$

$$\frac{\partial F}{\partial C_{m\alpha}} \propto [\hat{H}\phi_\beta](\mathbf{r}_m) K^{\beta\alpha}$$

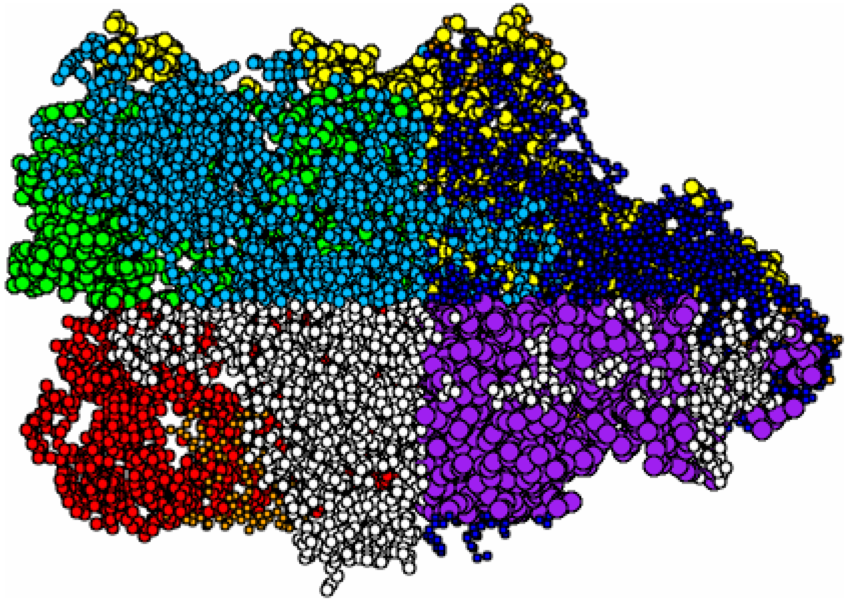
Parallel implementation using the Message Passing Interface (MPI) paradigm – each processor runs its own copy of the program with its own data

Parallelisation of data: Distribution of atoms (and NGWFs) to processors according to a space-filling curve

without SF curve



with SF curve

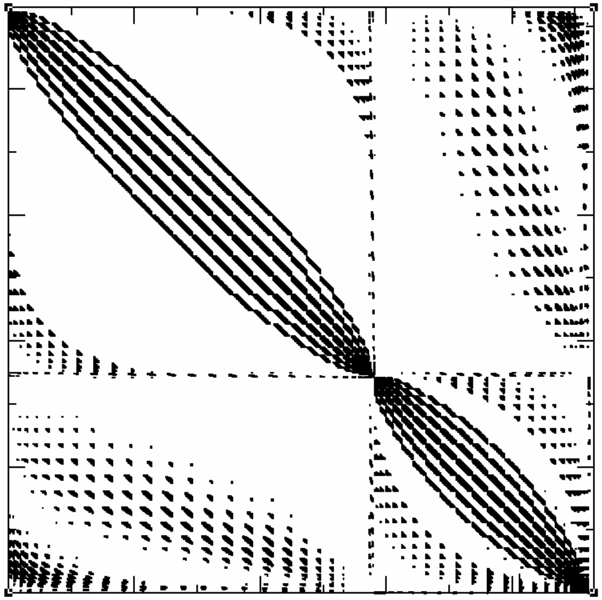


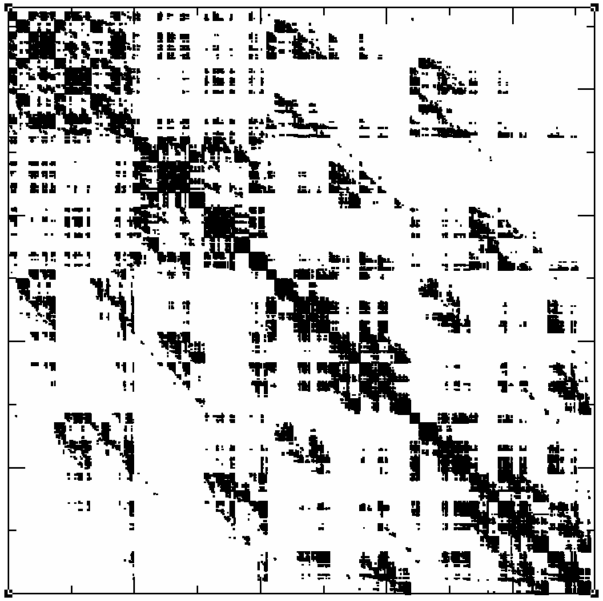
Effect on sparsity pattern of Hamiltonian matrix

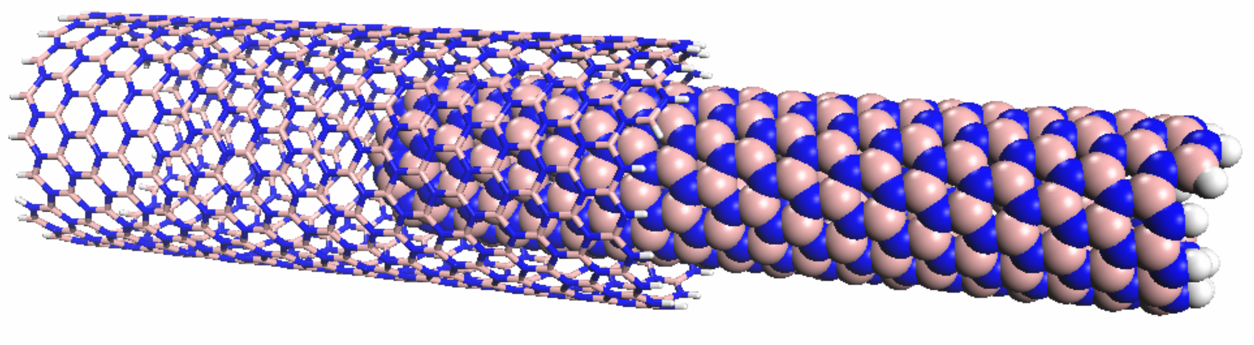
without SF curve

4650 x 4650

with SF curve





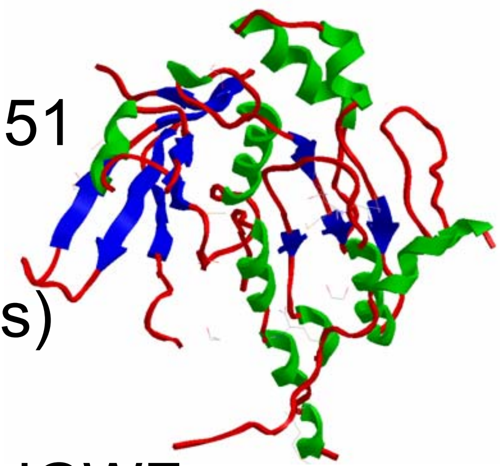


“Small” sparse matrices!

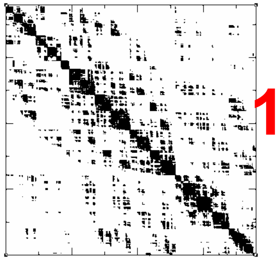
$$\rho(\mathbf{r},\mathbf{r}') = \sum_{\alpha\beta} \phi_\alpha(\mathbf{r}) K^{\alpha\beta} \phi^*_\beta(\mathbf{r}')$$

BRC4-RAD51 complex

(3000 atoms)

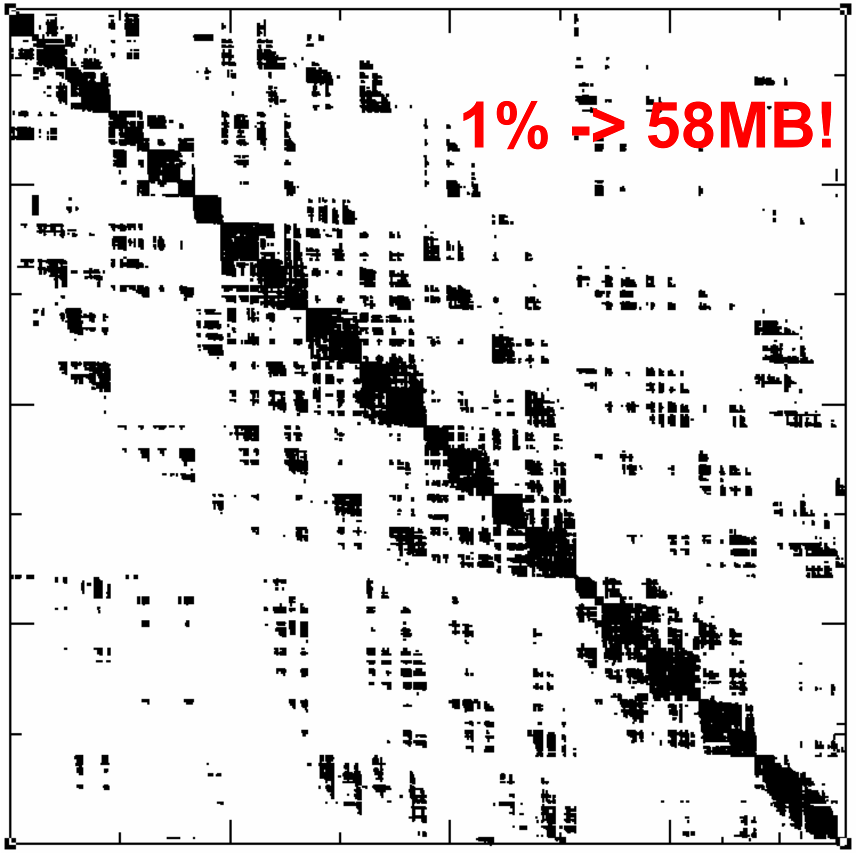


ONETEP NGWFs:
7600 x 7600



1% -> 4.4 MB!

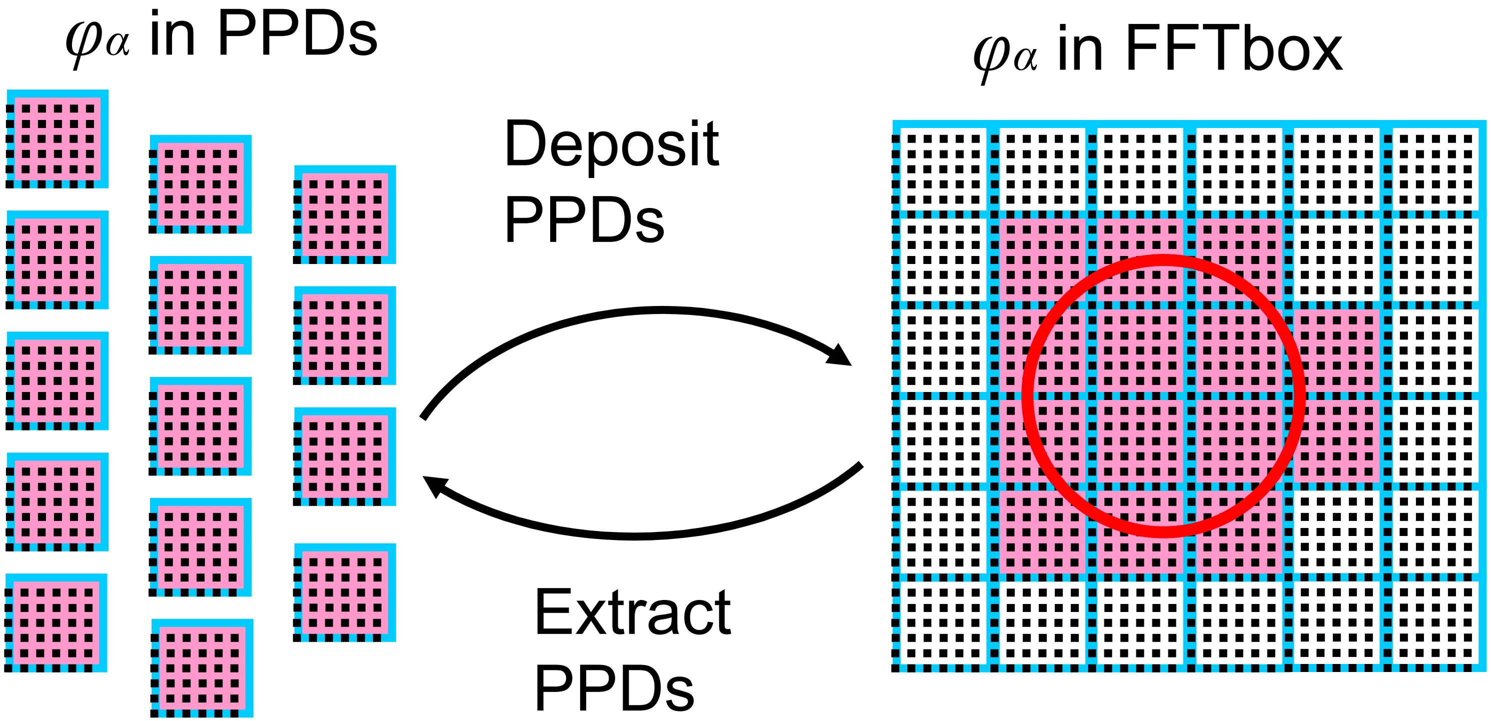
Atomic orbital “DZP” basis:
27500 x 27500



1% -> 58MB!

Can scale up to 100 processors (~5000 atoms) without data-parallel matrices!

PPD & FFTbox representation of functions



φ_α in PPDs

Deposit PPDs

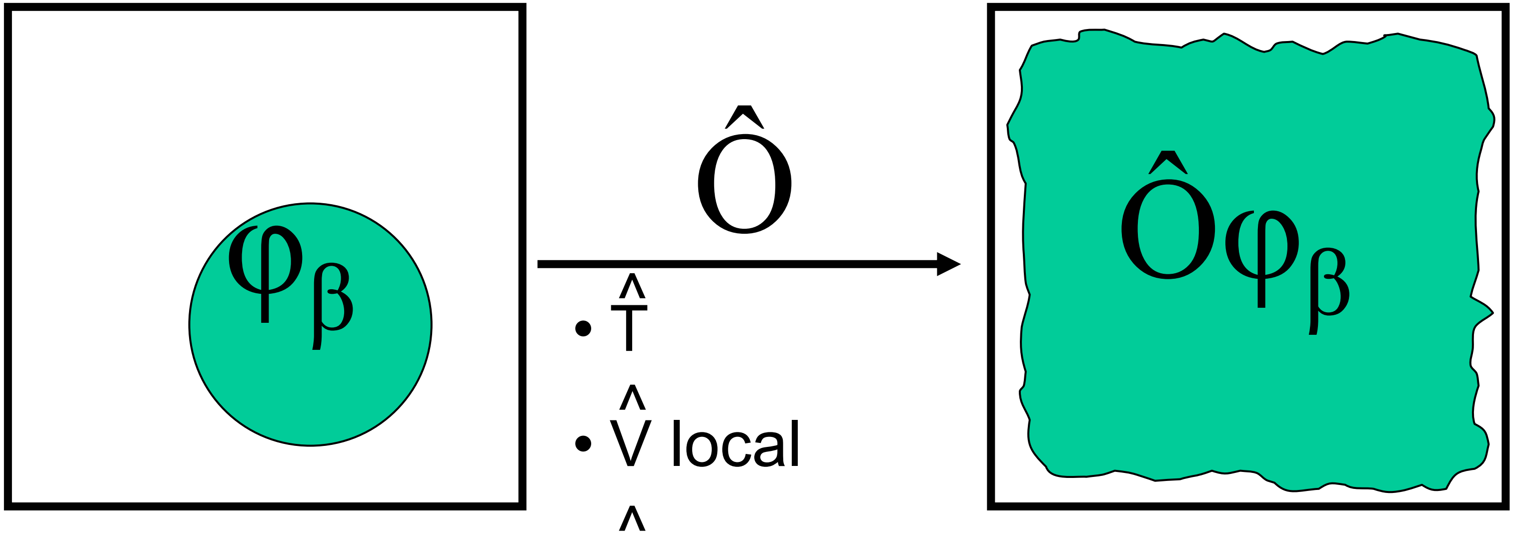
Extract PPDs

φ_α in FFTbox

- Compact storage in 1D arrays
- Fast communication

- QM operators
- Sums, products, interpolation

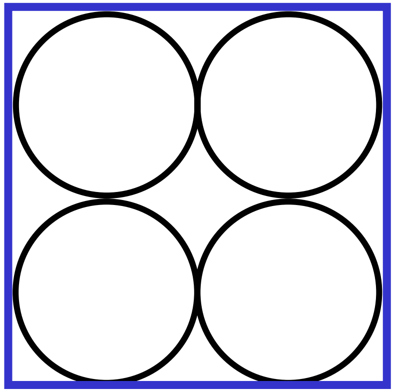
Quantum mechanical operators delocalise to the whole volume of the FFTbox. **Computationally expensive!**



- $\hat{T}$
- $\hat{V}$ local
- $\hat{V}$ non-local
- Interpolation to 2Gmax PSINC basis (fine grid)

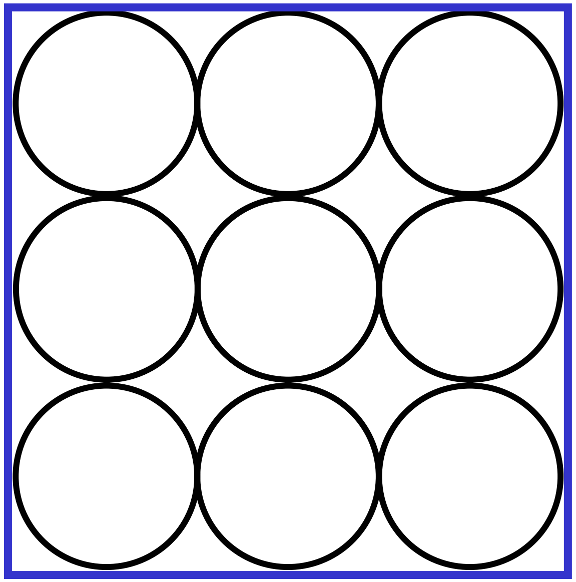
Calculation of $\hat{O}\varphi_\beta$ **for each** φ_α is linear-scaling but not ideal (large prefactor)!

There is a way to reduce the prefactor: Enlarge FFTbox!



Smallest FFT box ensuring:

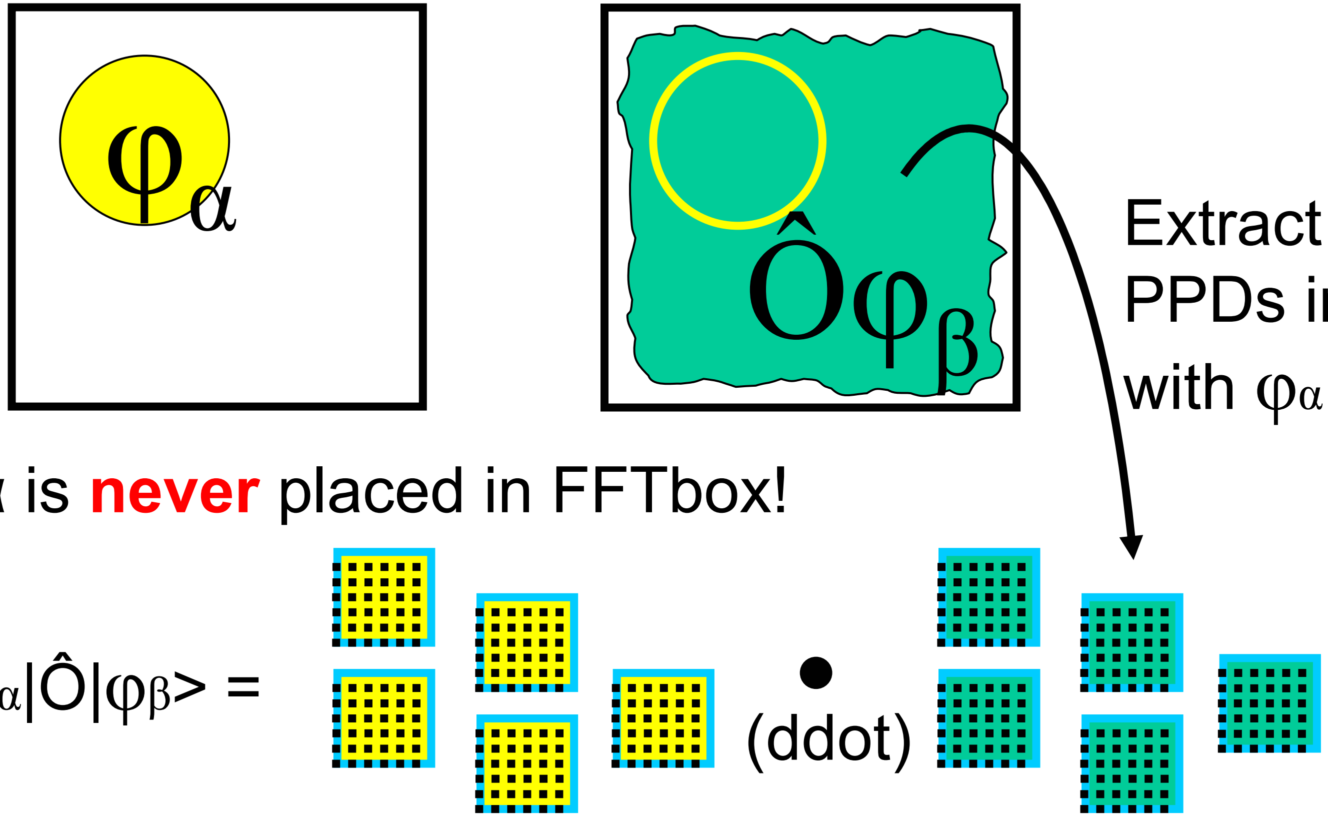
- Hermiticity
- Uniform representation for each operator
- **But**: φ_β needs to be re-centred for every single φ_α which overlaps with it!



Same advantages plus:

- φ_β does not need to be re-centred. $\hat{O}\varphi_\beta$ needs to be calculated **only once**!

Efficient calculation of integrals $<\varphi_\alpha|\hat{O}|\varphi_\beta>$



Extract **only** PPDs in common with φ_α

φ_α is **never** placed in FFTbox!

$<\varphi_\alpha|\hat{O}|\varphi_\beta> =$ (ddot)

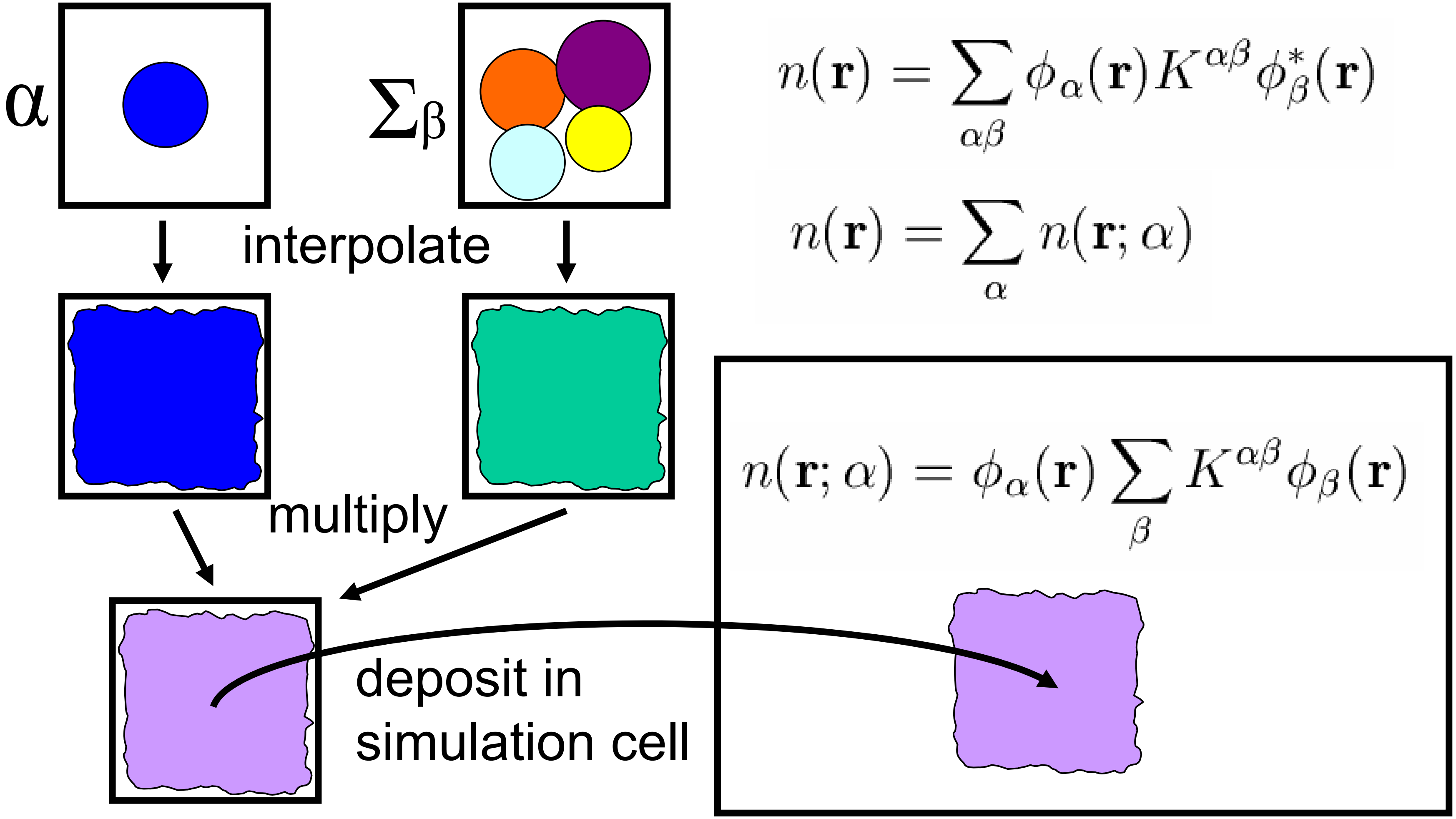
dot product of (small) 1D arrays

Keep **$\hat{O}\varphi_\beta$ FFTbox in memory and re-use** for $<\varphi_\beta|\hat{O}|\varphi_\beta>$, $<\varphi_\gamma|\hat{O}|\varphi_\beta>$, etc.

Efficient calculation of the charge density, *n(r)*

$$n(\mathbf{r}) = \sum_{\alpha\beta} \phi_\alpha(\mathbf{r}) K^{\alpha\beta} \phi^*_\beta(\mathbf{r})$$

$$n(\mathbf{r}) = \sum_{\alpha} n(\mathbf{r};\alpha)$$



α

Σ_β

interpolate

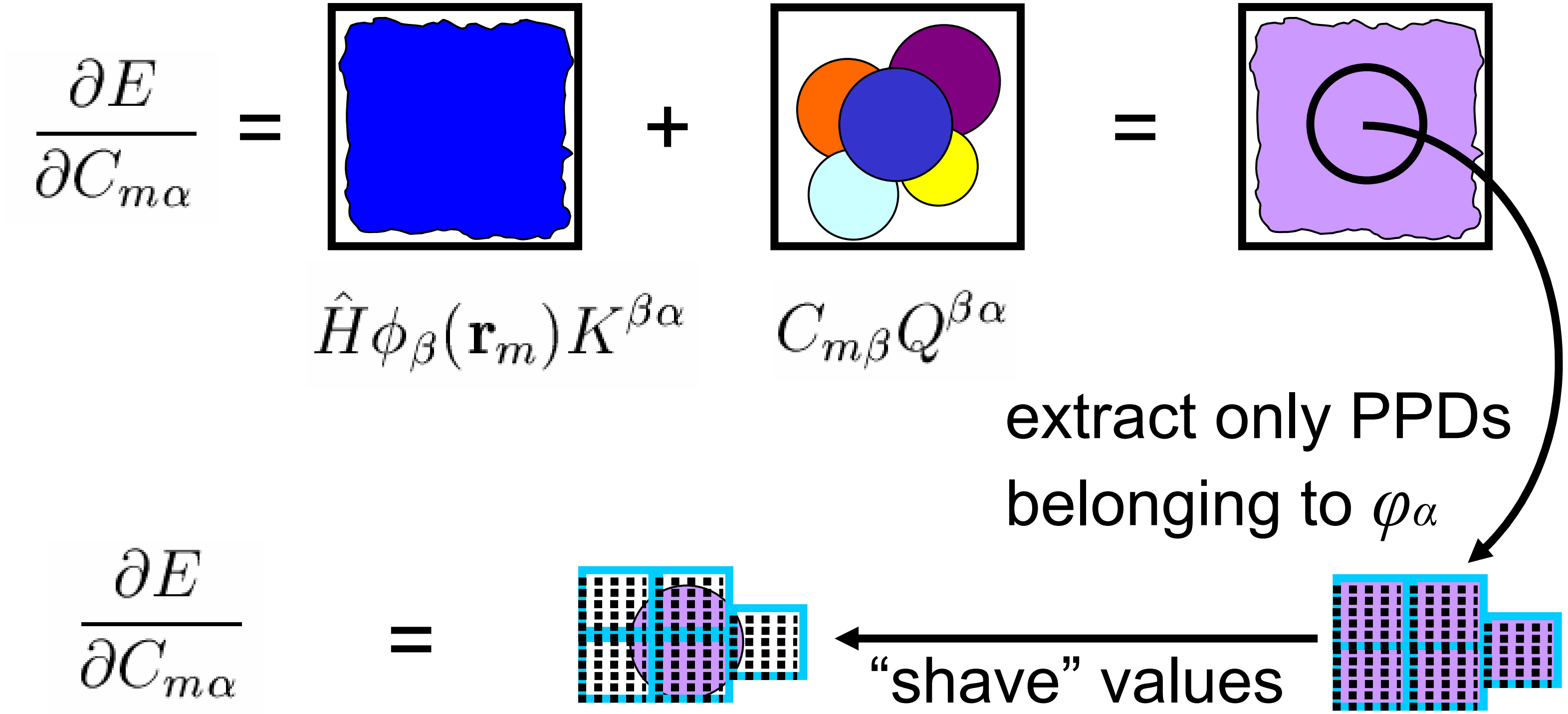
multiply

deposit in simulation cell

$$n(\mathbf{r};\alpha) = \phi_\alpha(\mathbf{r}) \sum_{\beta} K^{\alpha\beta} \phi_\beta(\mathbf{r})$$

Only two interpolations per ***n(r;α)***, **independent of num β**!

Efficient calculation of the NGWF gradient



$$\frac{\partial E}{\partial C_{m\alpha}} = \qquad + \qquad =$$

$\hat{H}\phi_\beta(\mathbf{r}_m)K^{\beta\alpha}$ $\quad C_{m\beta}Q^{\beta\alpha}$

extract only PPDs belonging to φ_α

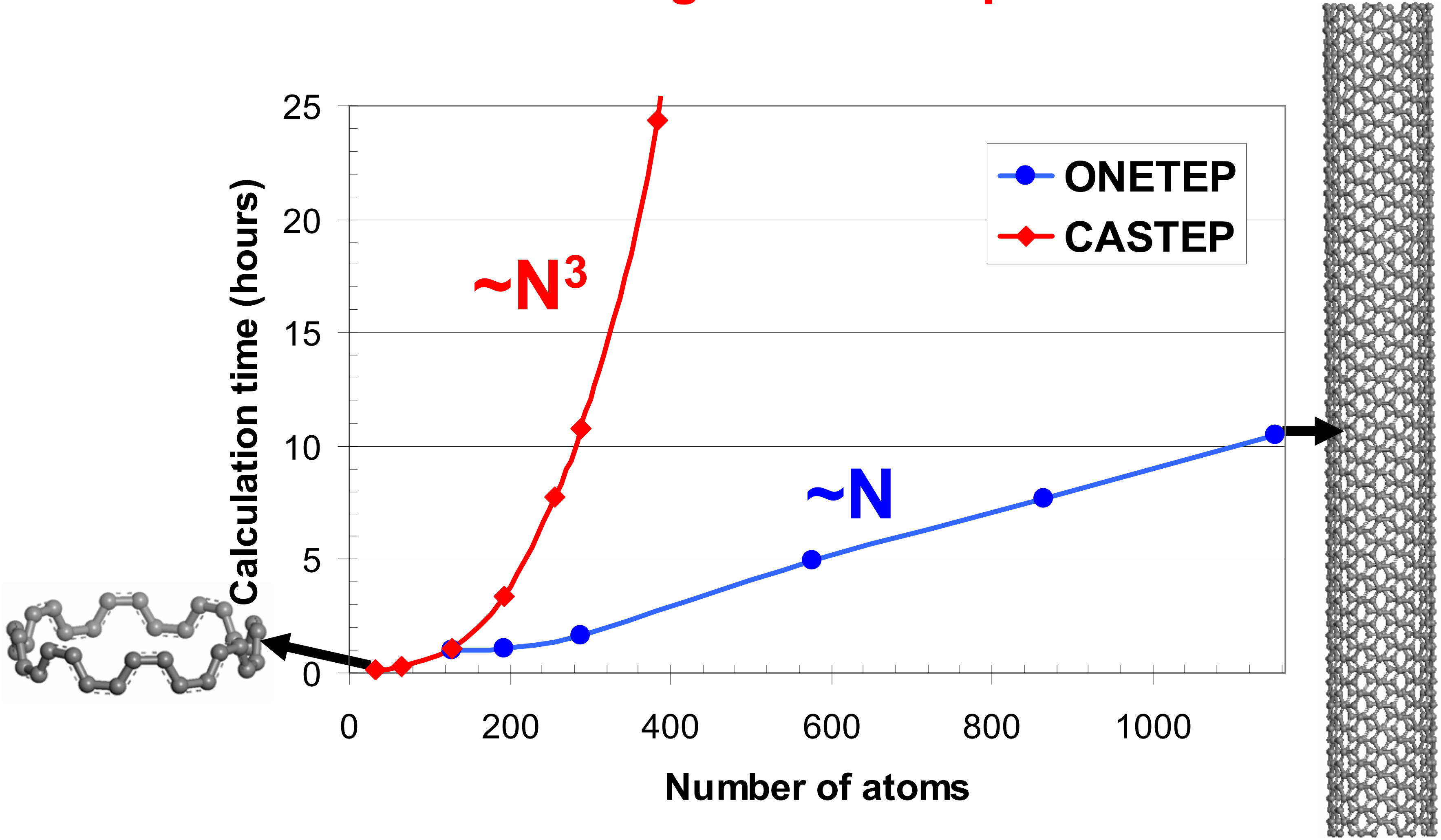
$$\frac{\partial E}{\partial C_{m\alpha}} =$$

“shave” values outside region of φ_α

(restricted to region of φα - suitable for updating φα in conjugate gradients optimisation)

Only one application of Ĥ per φ_α!

Linear-scaling with small prefactor

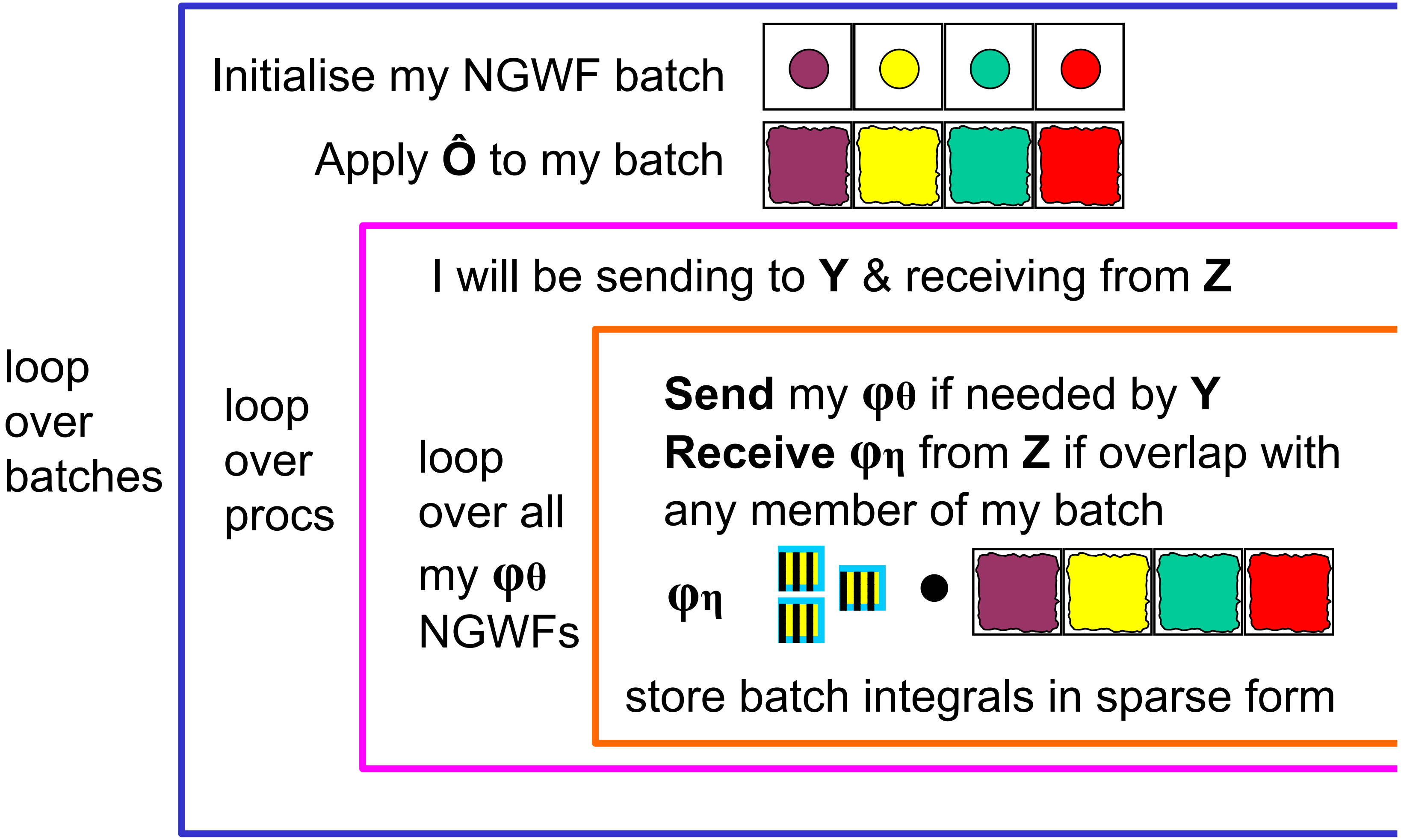


Communication model

Functions on different processors which overlap need to be communicated during computation

- Use only **point-to-point** communication
- Keep functions to be sent in buffers and interleave communication with computation by using **non-blocking sends**
- Only use **PPD representation** during communication
- Keep in memory **batches of $\hat{O}_{\varphi\beta}$** to minimise communication
- **Send only if** there is an overlap with current batch of functions of receiving processor
- Work with **processor-processor blocks** of functions/matrices – do $N_{\text{processor}}$ supersteps

Communication model **simplest case – evaluation of integrals**: outline from the viewpoint of processor **X**



loop over batches

Initialise my NGWF batch

Apply **Ô** to my batch

loop over procs

I will be sending to **Y** & receiving from **Z**

loop over all my $\varphi\theta$ NGWFs

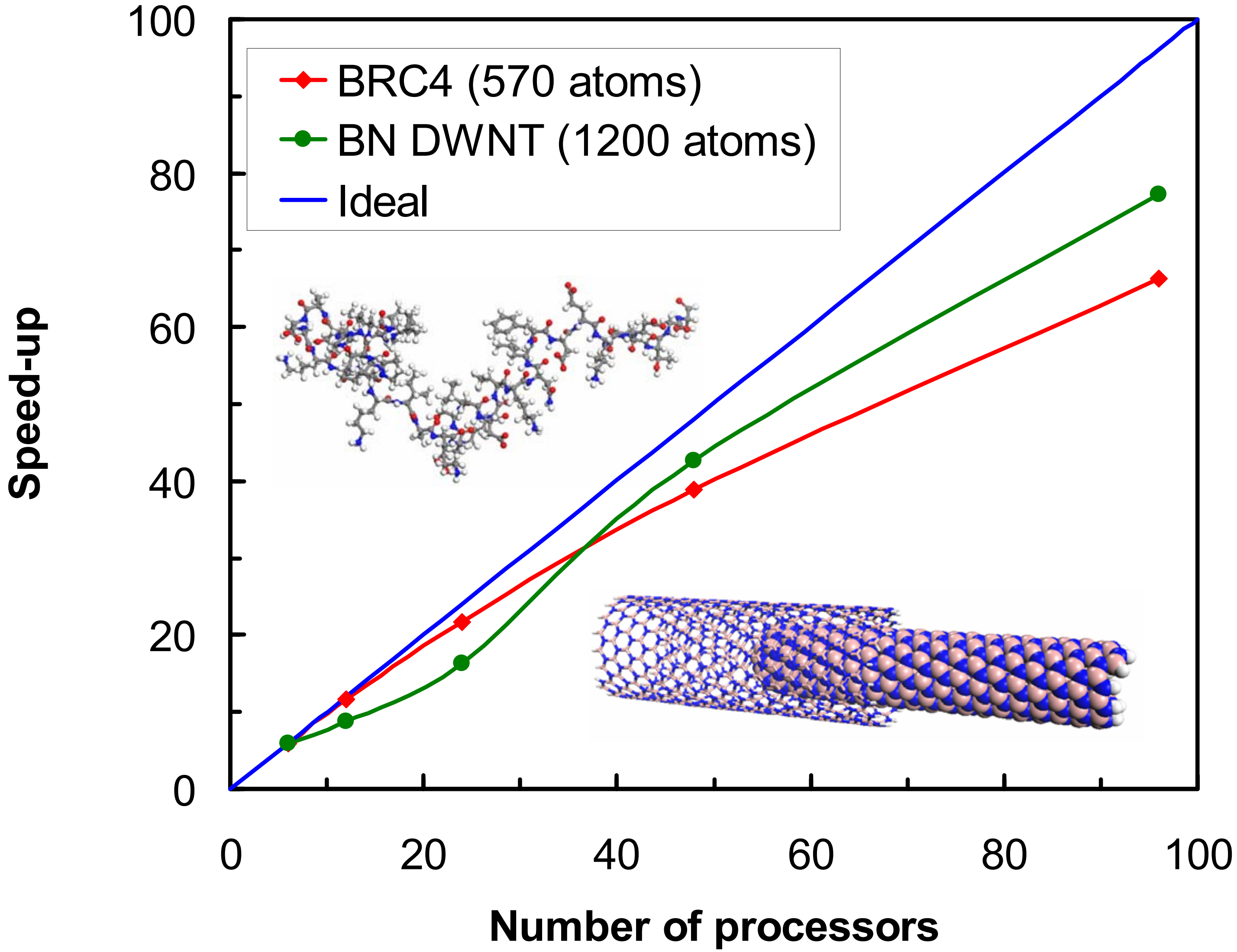
Send my $\varphi\theta$ if needed by **Y**

Receive $\varphi\eta$ from **Z** if overlap with any member of my batch

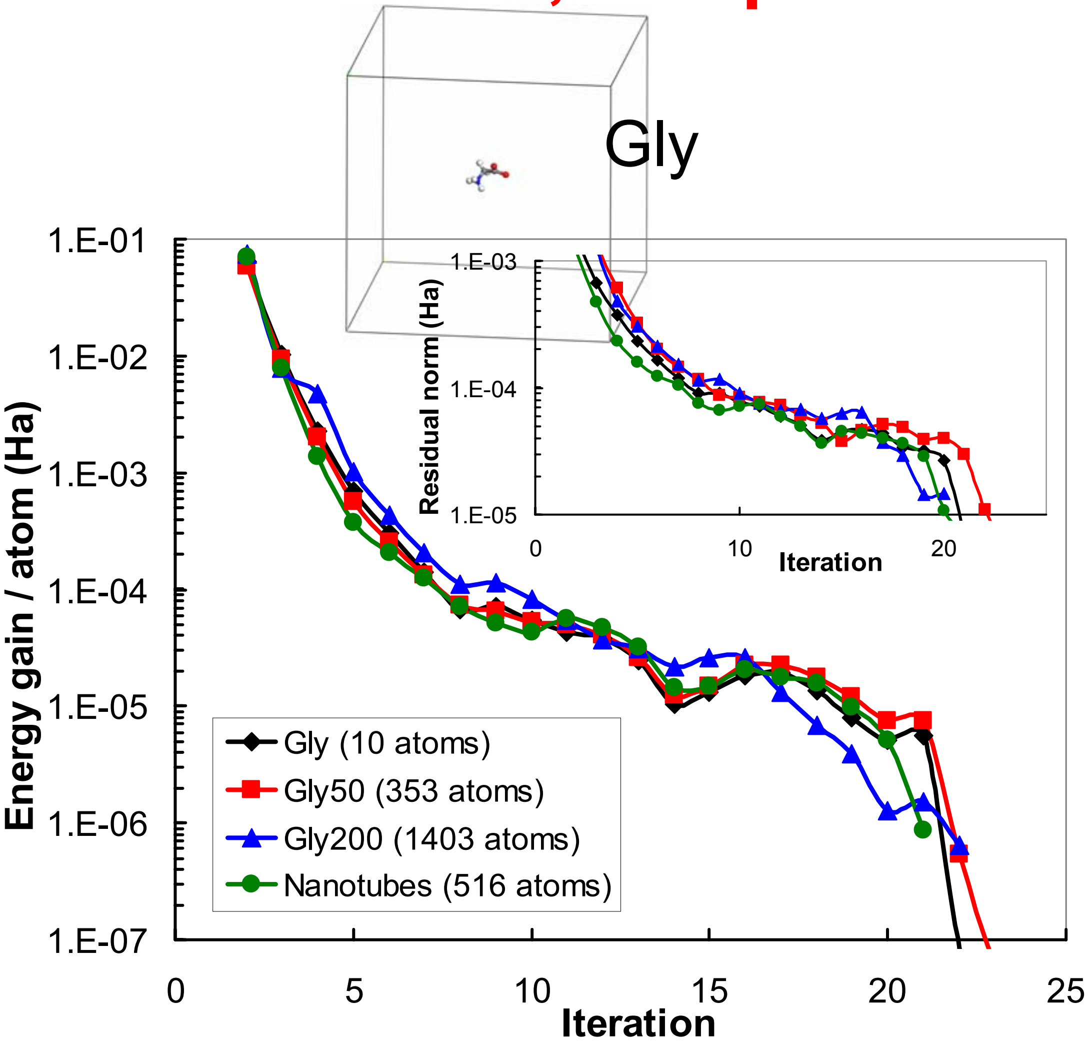
$\varphi\eta$ ●

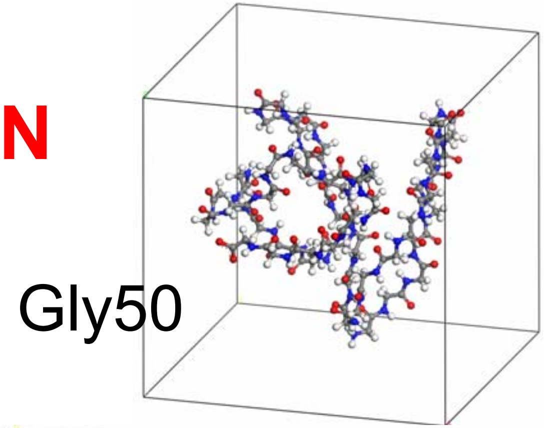
store batch integrals in sparse form

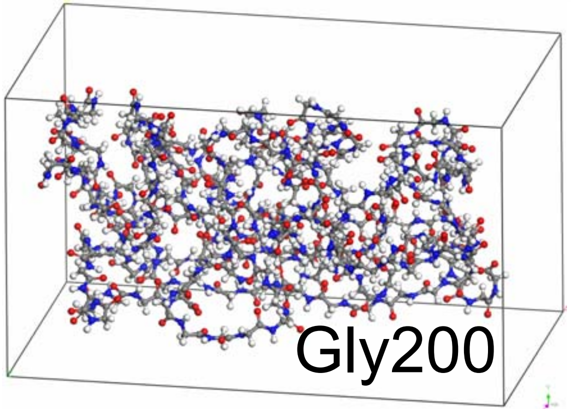
Speedup with increasing number of processors

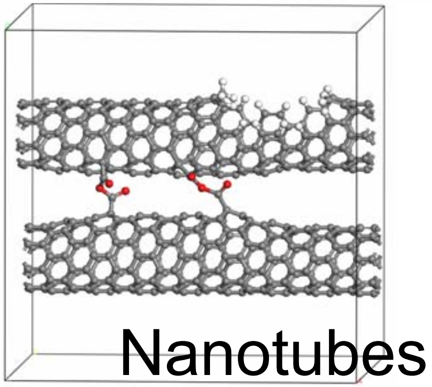


True linear-scaling: Number of iterations small, independent of N









Chemical accuracy, no Basis Set Superposition Error

Basis sets, number of localised functions

CASTEP

Plane-waves 1292 eV

ONETEP

Plane-waves 1292 eV

H: 1 NGWF

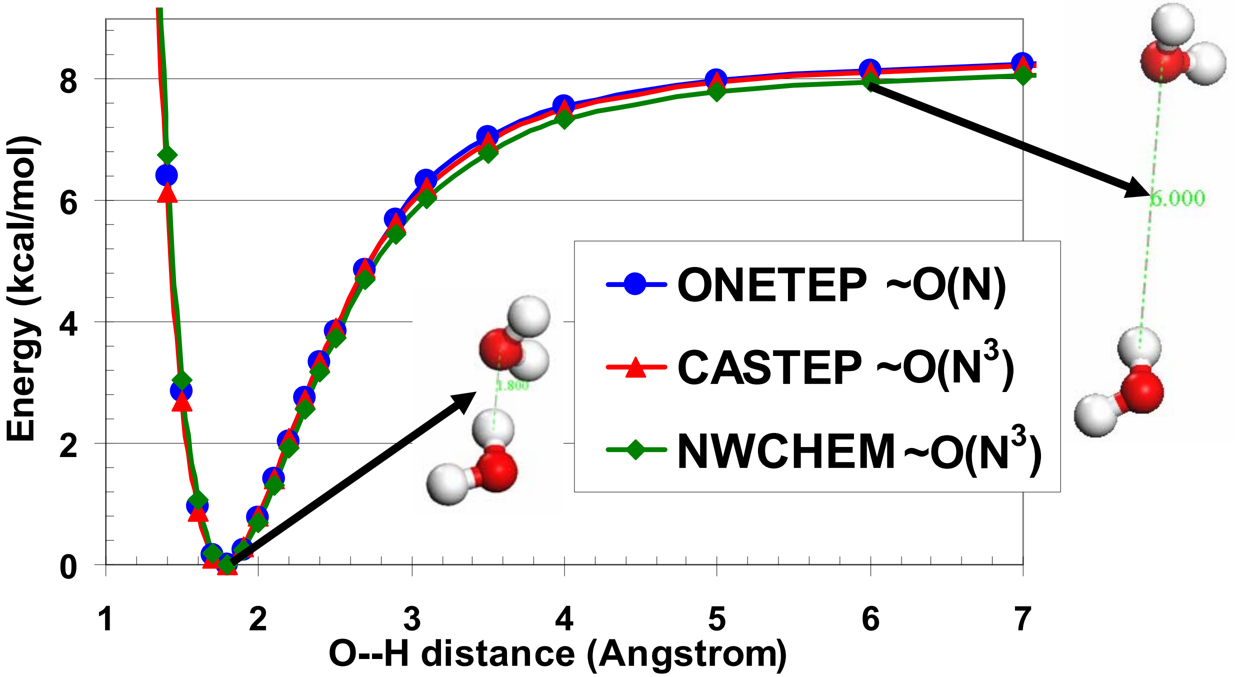
O: 4 NGWFs

NWCHEM

cc-pVTZ+diffuse

H: 25 contracted Gaussians

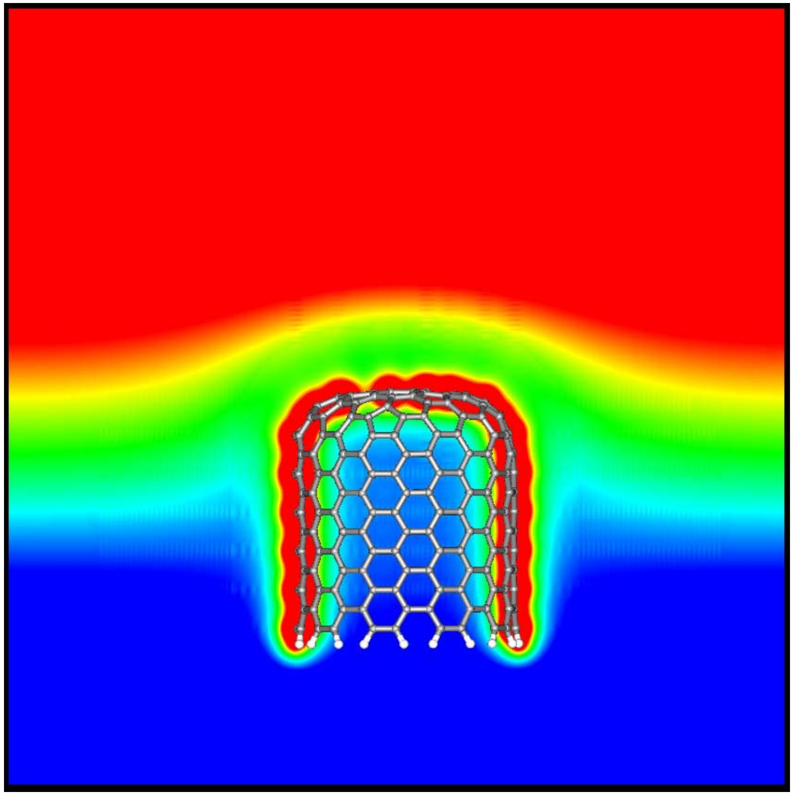
O: 55 contracted Gaussians



Simulation cell size cost is O(0) in number of atoms. Can do plane-wave calculations in huge simulation cells!

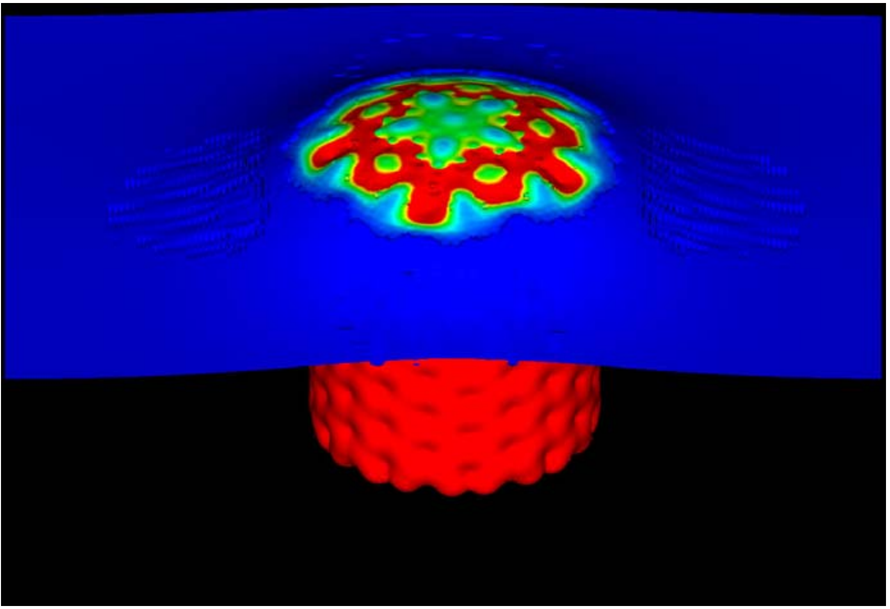
ONETEP calculations of carbon nanotube tip in uniform external electric field (C.-K. Skylaris & G. Csányi et al.)

Local potential



50Å x 50Å x 50Å simulation cell

Charge density on local potential iso-surface near Fermi level



Chris-Kriton Skylaris. ESDG 19/5/2004. TCM group, Cavendish Laboratory, University of Cambridge.

Conclusions

- ONETEP – linear-scaling total energy code – takes full advantage of parallel computers
- Accuracy equivalent to conventional cubic-scaling Plane-wave / Gaussian DFT codes
- Low prefactor – breakeven with cubic-scaling codes in the region of a few hundred atoms
- Work in progress: data-parallelisation of simulation cell – even larger simulation cells
- A whole new level of large scale first principles simulations from condensed matter physics to biology now possible - see next week’s instalment which will be brought to you by Peter D. Haynes