Motivation

We want to solve Schrödinger's equation:

$$-\frac{1}{2}\nabla^2\psi(\mathbf{r}) + V(\mathbf{r})\psi(\mathbf{r}) = \varepsilon\psi(\mathbf{r})$$

- Kinetic energy operator is diagonal in momentum-space

- Potential operator is diagonal in real-space

$\Rightarrow$ need to be able to switch between momentum- and real-space representations
i.e. perform Fourier transforms.

Fourier series

- Bloch's theorem:

$$\psi(\mathbf{r}) = \exp(\mathrm{i}\mathbf{k}\cdot\mathbf{r})u_{\mathbf{k}}(\mathbf{r})$$

where $u_{\mathbf{k}}(\mathbf{r})$ is cell-periodic i.e. $u_{\mathbf{k}}(\mathbf{r}+\mathbf{R}) = u_{\mathbf{k}}(\mathbf{r})$ for any lattice vector $\mathbf{R}$.

⇒ Expand $u_{\mathbf{k}}(\mathbf{r})$ as a Fourier series:

$$u_{\mathbf{k}}(\mathbf{r}) = \sum_{\mathbf{G}} c_{\mathbf{k}}(\mathbf{G})\exp(\mathrm{i}\mathbf{G}\cdot\mathbf{r})$$

where $\mathbf{G}$ denotes a reciprocal lattice vector.

- Fourier inversion theorem gives:

$$c_{\mathbf{k}}(\mathbf{G}) = \frac{1}{V_{\mathrm{cell}}}\int_{\mathrm{cell}} \mathrm{d}\mathbf{r}\, u_{\mathbf{k}}(\mathbf{r})\exp(-\mathrm{i}\mathbf{G}\cdot\mathbf{r})$$

Discrete Fourier transforms

- In practice, we sample $u_{\mathbf{k}}(\mathbf{r})$ discretely on a uniform grid of N points $\{\mathbf{r}_n\}$:

$$\begin{aligned} u_{\mathbf{k}}(\mathbf{r}_n) &= \sum_{\mathbf{G}} c_{\mathbf{k}}(\mathbf{G}) \exp(\mathrm{i}\mathbf{G} \cdot \mathbf{r}_n) \\ c_{\mathbf{k}}(\mathbf{G}) &= \frac{1}{N} \sum_{\mathbf{r}_n} u_{\mathbf{k}}(\mathbf{r}_n) \exp(-\mathrm{i}\mathbf{G} \cdot \mathbf{r}_n) \end{aligned}$$

- The sum over reciprocal lattice vectors $\mathbf{G}$ now only runs over those below the Nyquist frequency determined by the real-space grid.

- Although these results are an approximation to the continuous Fourier series, the above inversion theorem is *exact*.

Slow Fourier Transforms

Consider a general 1D Fourier transform relating two vectors of length n:

- $\{x_k; 0 \leq k < n\}$ contains the values in real-space
- $\{X_k; 0 \leq k < n\}$ contains the frequency components

$$X_k = \sum_{j=0}^{n-1} \exp(-2\pi \mathrm{i} kj/n) x_j$$

This is just a matrix-vector multiplication $X_k = F_{kj} x_j$

- $F_{kj} = [\exp(-2\pi \mathrm{i}/n)]^{kj} = \omega_n^{kj}$
- A straightforward implementation requires $\mathcal{O}(n^2)$ operations.

Danielson-Lanczos Lemma

G. C. Danielson and C. Lanczos, "Some improvements in practical Fourier analysis and their application to X-ray scattering from liquids", *J. Franklin Inst.* **233**, 365 (1942).

For even $n = 2m$:

$$\begin{aligned} X_k &= \sum_{j=0,\ \text{even}}^{n-1} \omega_n^{kj} x_j + \sum_{j=0,\ \text{odd}}^{n-1} \omega_n^{kj} x_j \\ &= \sum_{j'=0}^{m-1} \omega_n^{k(2j')} x_{2j'} + \sum_{j'=0}^{m-1} \omega_n^{k(2j'+1)} x_{2j'+1} \\ &= \sum_{j'=0}^{m-1} \omega_m^{kj'} x_{j'}^{\text{even}} + \omega_n^k \sum_{j'=0}^{m-1} \omega_m^{kj'} x_{j'}^{\text{odd}} \end{aligned}$$

since $\omega_n^{2k} = \omega_{n/2}^k$ and writing $x_k^{\text{even}} = x_{2k}$ and $x_k^{\text{odd}} = x_{2k+1}$.

Fast Fourier Transforms

J. W. Cooley and J. W. Tukey, “An algorithm for the machine calculation of complex Fourier series”, *Math. Comput.* **19**, 297 (1965).

- The Danielson-Lanczos lemma enables us to write a discrete Fourier transform (DFT) of length n as a combination of two DFTs of length $n/2$.

- If n is a power of 2, we may apply this lemma recursively until we require $\log_2 n$ trivial DFTs of length 1.

- The cost is therefore $\mathcal{O}(n \log_2 n)$ instead of $\mathcal{O}(n^2)$ which for $n \sim 10\,000$ is roughly $1\,000$ times faster.

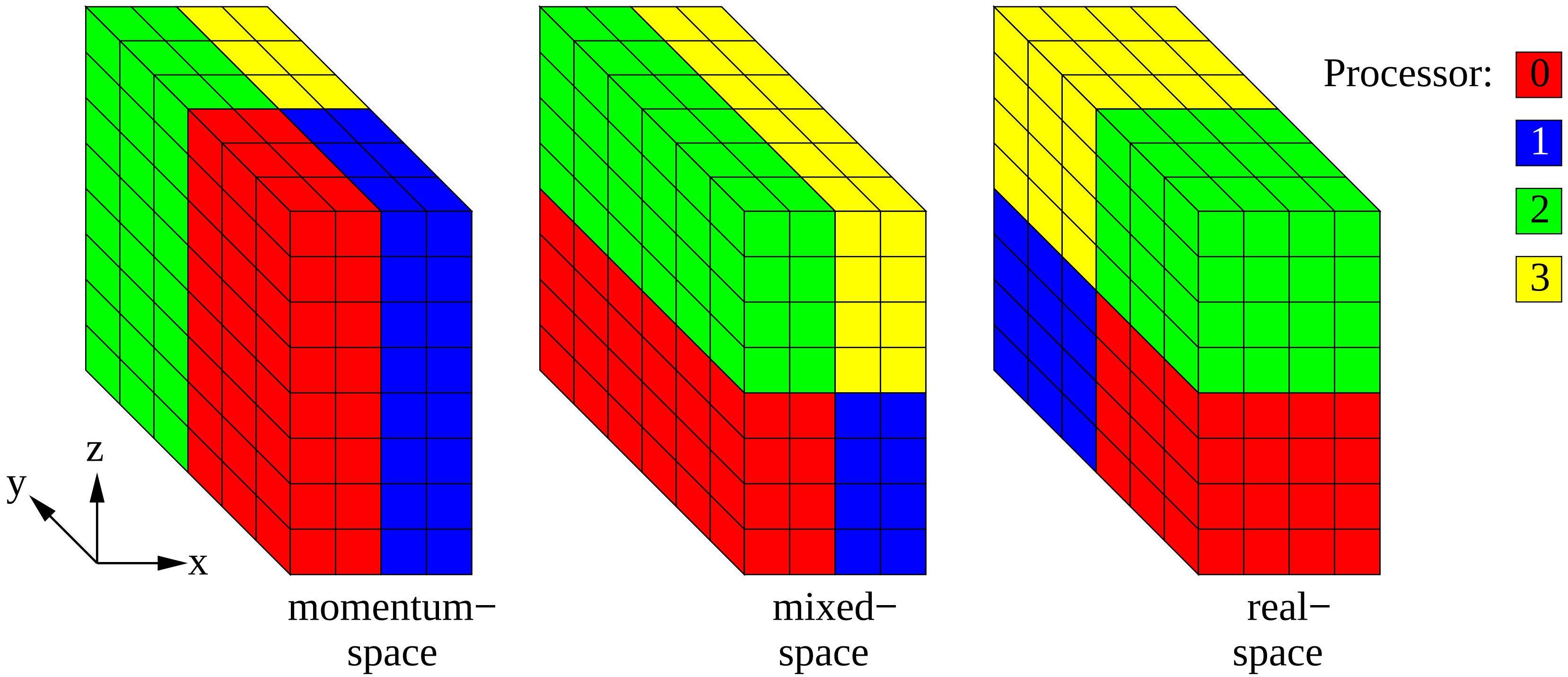
- This result can be generalised for the case when n contains prime factors other than 2.

3D Fourier Transforms

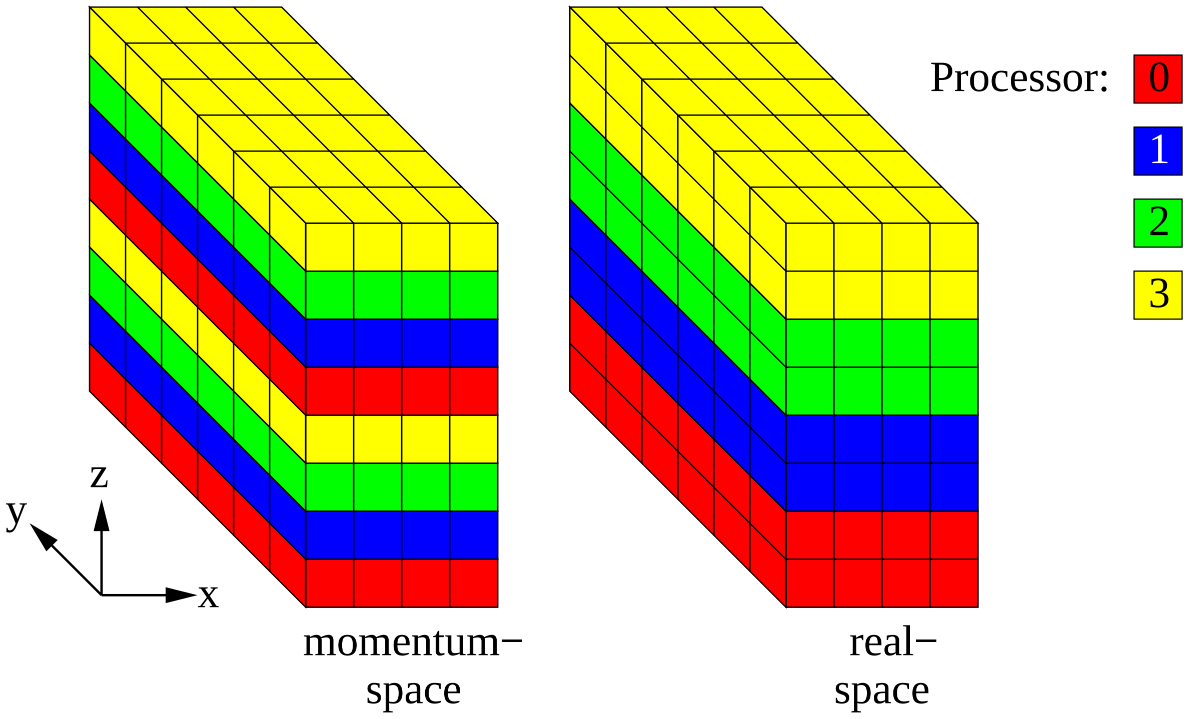
$$X_{k_x k_y k_z} = \sum_{j_x=0}^{n_x-1} \omega_{n_x}^{k_x j_x} \sum_{j_y=0}^{n_y-1} \omega_{n_y}^{k_y j_y} \sum_{j_z=0}^{n_z-1} \omega_{n_z}^{k_z j_z} \, x_{j_x j_y j_z}$$

- The product of three 1D DFTs.
- The 1D DFTs commute with each other.
- Different stages of the 1D DFTs may also be interlaced.

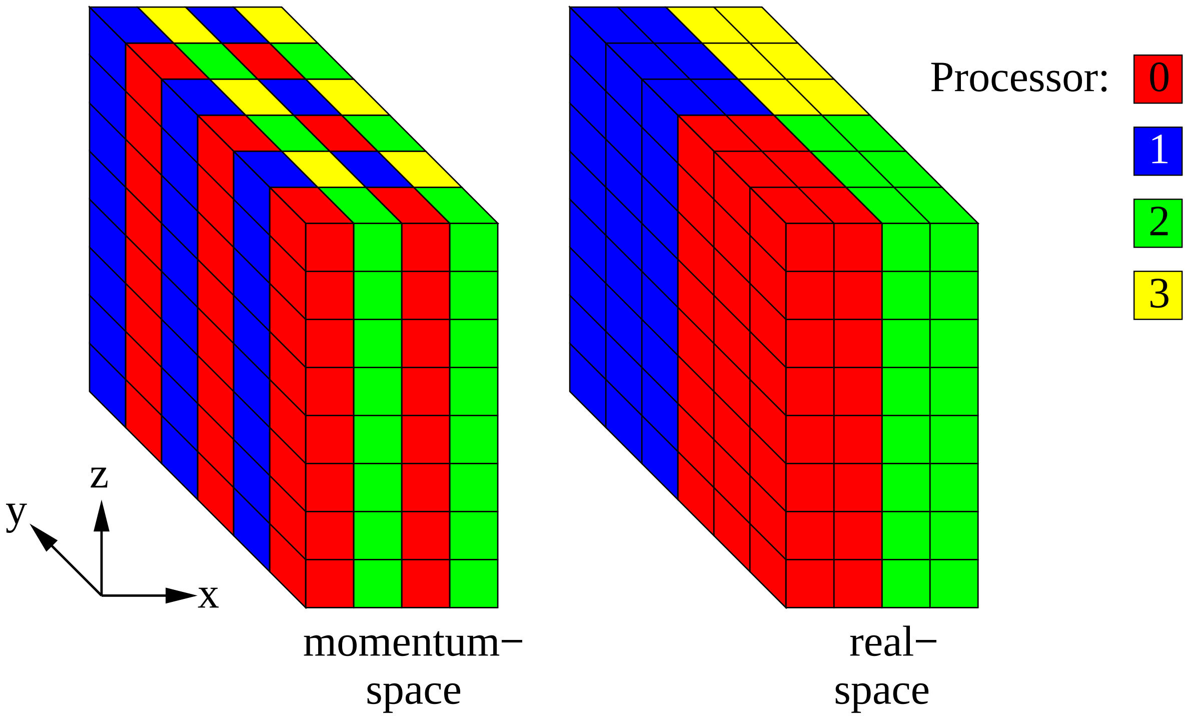
Traditional Parallel Fast Fourier Transforms



New Parallel Fast Fourier Transforms



Alternative Distribution



Cost comparison

- Computational costs are identical.
- Communication patterns between nodes are very different:
  - Traditional method has two transposition phases in which each node communicates with every other node.
  - New method has $\log_2 n$ phases in which only pairs of nodes communicate.

Cost modelling

The time cost to transfer a data packet between two nodes consists of two parts:

- A fixed overhead or *latency*, α, which is independent of the amount of data sent.
- The time to transmit the data between the nodes, which depends upon the *bandwidth* of the connection, β, and the size of the data packet.

Notation:

- $n = n_x n_y n_z$ is the full FFT grid size.
- The number of nodes is N.

Cost for traditional method

We can get each node to communicate with every other node in $N-1$ stages in which $N/2$ pairs of nodes simultaneously exchange packets of size nu/N^2 where u is the size of a single data element (16 bytes for double precision complex).

$$\begin{aligned} \tau_{\text{trad}} &= 2(N-1)\left[\alpha + \frac{nu}{\beta N^2}\right] \\ &\approx 2\left[\alpha N + \frac{nu}{\beta N}\right] \end{aligned}$$

Cost for new method

There are now $\log_2 N$ pairwise communication phases in which packets of size nu/N are exchanged.

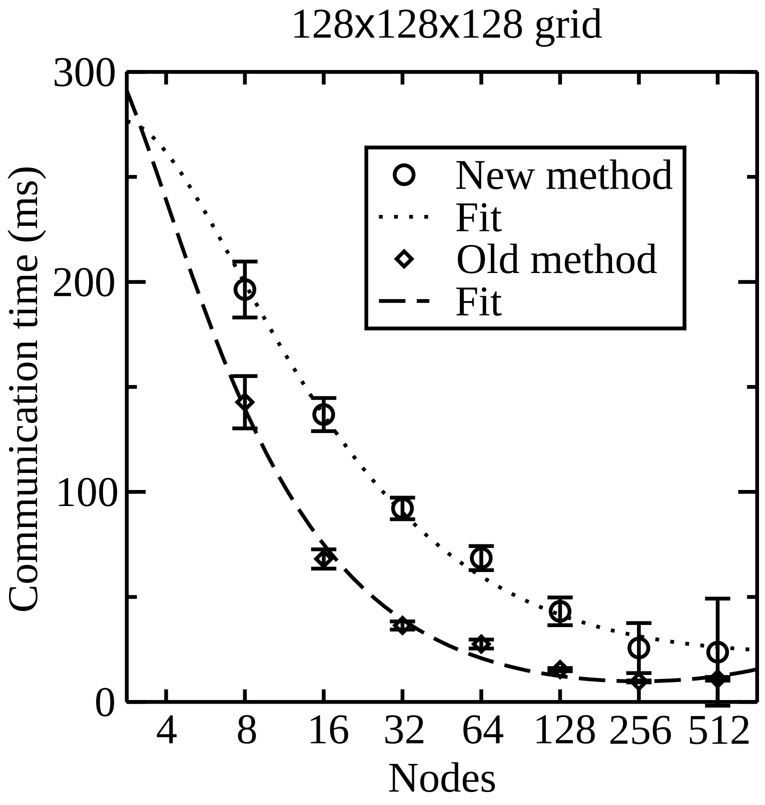
$$\tau_{\mathrm{new}} = \log_2 N \left[\alpha + \frac{nu}{\beta N}\right]$$

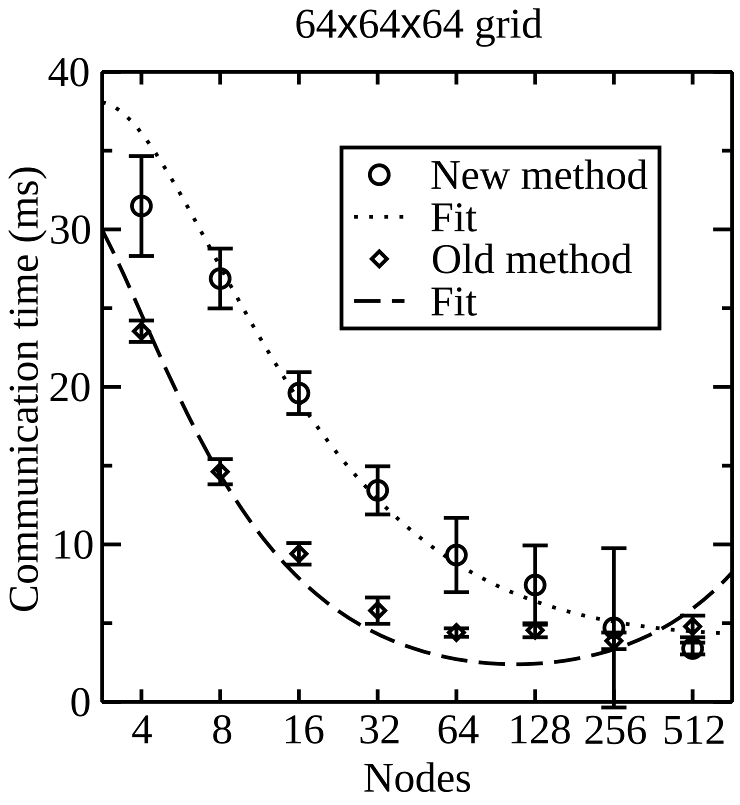
- Fewer packets exchanged ⇒ lower latency cost.
- Larger packets exchanged ⇒ higher transmission cost.

Expect the new method to be advantageous in the limits of small n and large N.

Exact cross-over depends upon the machine: the product $\alpha\beta$ determines the packet size which costs as much in latency as transmission to send.

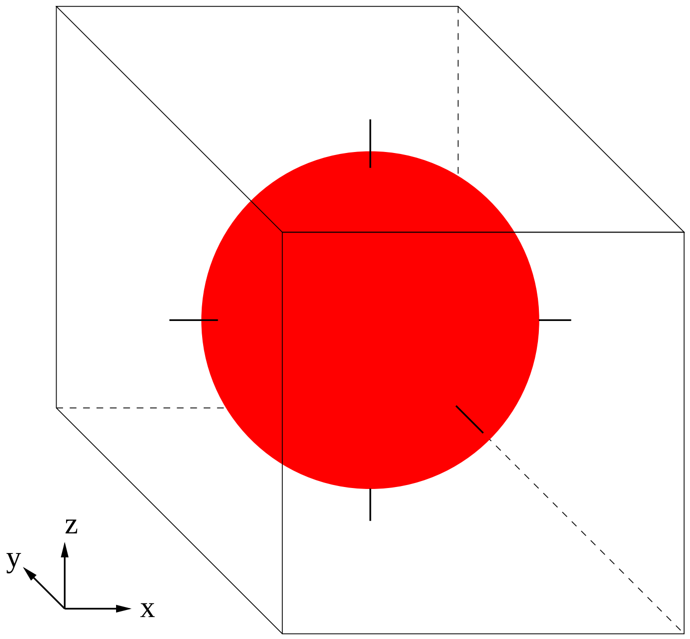
Results





CRAY T3E-1200E; $\alpha \approx 300\mu\mathrm{s}$ and $\beta \approx 8.7\mathrm{Mb\ s}^{-1}$.

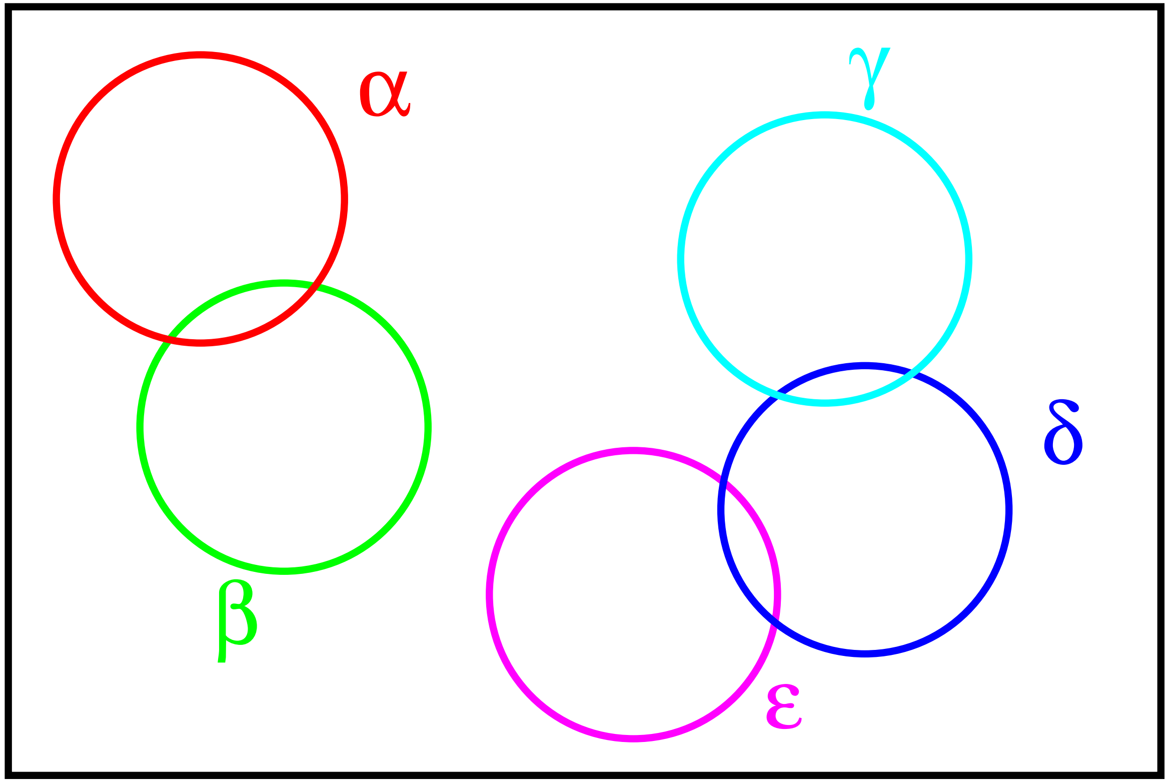
Load balancing



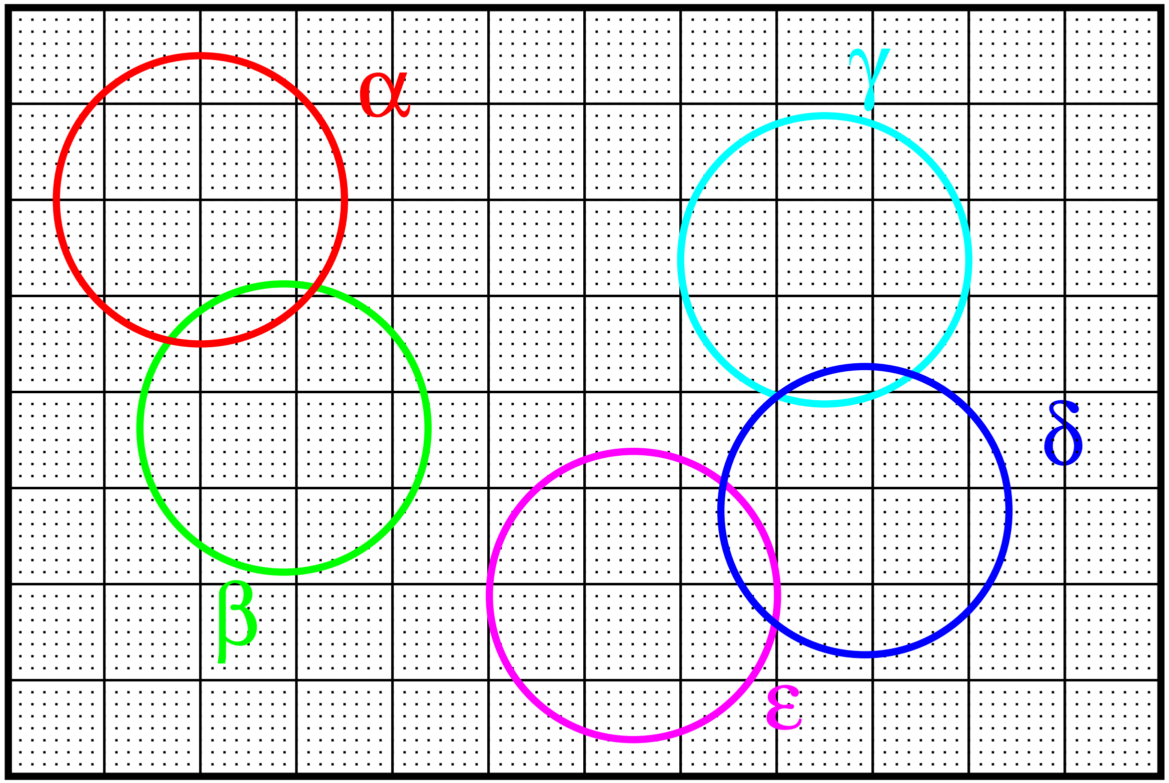
Cutoff sphere in momentum-space.

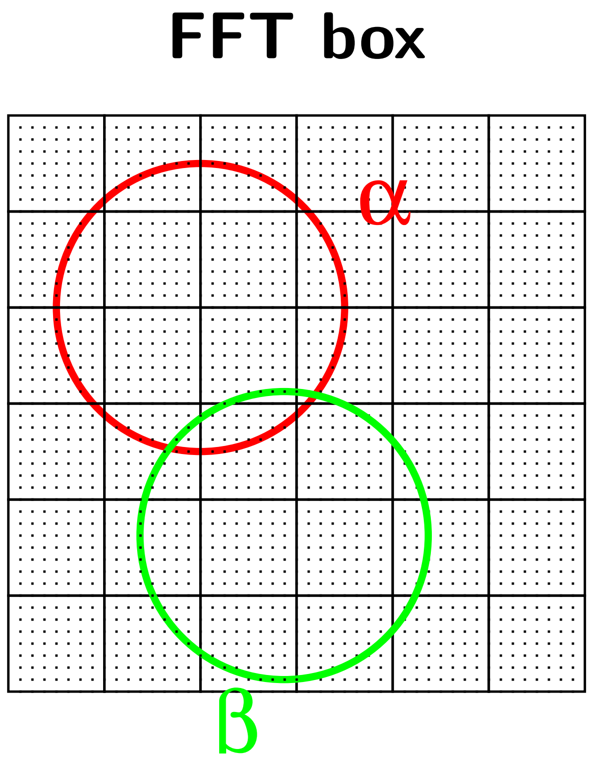
Application to $O(N)$ methods

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



Subcells




FFT box
α
β

Advantages of the new FFT method

- Exploits localization in real-space: no need to do the initial 3D FFT on subcells not overlapping a function in the FFT box.

- Can store the intermediate stage of the FFT in the same amount of memory as in real-space e.g. for calculating the density:

$$n(\mathbf{r}) = 2\sum_{\alpha\beta} \phi_\alpha(\mathbf{r}) K^{\alpha\beta} \phi_\beta(\mathbf{r})$$

where the functions $\{\phi_\alpha(\mathbf{r})\}$ must be Fourier interpolated before their product is calculated.