

Plane-Wave-Pseudopotential technicalities

CECAM Tutorial on *Simulating Matter at the Nanoscale...*, 13 Nov. 2006

Fourier Transform of a periodic function

Let us consider a 1-d example: a periodically repeated function $f(x)$, with period L . Its Fourier components:

$$\tilde{f}(q) = \frac{1}{L} \int f(x) e^{-iqx} dx$$

are nonzero over an infinite set of discrete values of q :

$$q_n = n \frac{2\pi}{L}, \quad -\infty < n < \infty$$

The Fourier components decay to 0 for large q . The inverse Fourier transform has the form

$$f(x) = \sum_n \tilde{f}(q_n) e^{iq_n x} = \sum_n \tilde{f}_n e^{in(2\pi x/L)}$$

Discrete Fourier Transform

Our functions are defined over a discrete but finite grid of q . How are they represented in x space? If we repeat periodically also the q grid, we will automatically have a discrete grid of x as well:

$$f(x) \rightarrow f_m = f(x_m), \quad x_m = m \frac{L}{N}, \quad m = 0, \dots, N - 1$$
$$\tilde{f}(q) \rightarrow \tilde{f}_n = \tilde{f}(q_n), \quad q_n = n \frac{2\pi}{L}, \quad n = 0, \dots, N - 1$$

The Discrete Fourier Transform can be rewritten as

$$f_m = \sum_{n=0}^{N-1} \tilde{f}_n e^{i(2\pi nm/N)} \quad (x - \text{space})$$
$$\tilde{f}_n = \frac{1}{N} \sum_{m=0}^{N-1} f_m e^{-i(2\pi nm/N)} \quad (q - \text{space})$$

Discrete Fourier Transform in 3d

Generalization of the Discrete Fourier Transform to 3 dimensions:

$$\mathbf{G} = n'_1 \mathbf{G}_1 + n'_2 \mathbf{G}_2 + n'_3 \mathbf{G}_3$$

with $n_1 = 0, \dots, N_1 - 1$, $n_2 = 0, \dots, N_2 - 1$, $n_3 = 0, \dots, N_3 - 1$, and n'_1, n'_2, n'_3 are n_1, n_2, n_3 refolded so that they are centered around the origin (remember: the \mathbf{G} -space grid is also periodic!). $\mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_3$ are the primitive translations of the unit cell of the reciprocal lattice.

$$\mathbf{r} = m_1 \frac{\mathbf{R}_1}{N_1} + m_2 \frac{\mathbf{R}_2}{N_2} + m_3 \frac{\mathbf{R}_3}{N_3}$$

with $m_1 = 0, \dots, N_1 - 1$, $m_2 = 0, \dots, N_2 - 1$, $m_3 = 0, \dots, N_3 - 1$. $\mathbf{R}_1, \mathbf{R}_2, \mathbf{R}_3$ are the primitive translations of the unit cell. This grid spans the unit cell.

Discrete Fourier Transform in 3d

Original Fourier transform:

$$f(\mathbf{r}) = \sum_{\mathbf{G}} \tilde{f}(\mathbf{G}) e^{i\mathbf{G}\cdot\mathbf{r}} \rightarrow f(m_1, m_2, m_3)$$

$$\tilde{f}(\mathbf{G}) = \frac{1}{\Omega} \int f(\mathbf{r}) e^{-i\mathbf{G}\cdot\mathbf{r}} d\mathbf{r} \rightarrow \tilde{f}(n_1, n_2, n_3)$$

Discretized Fourier Transform:

$$f(m_1, m_2, m_3) = \sum_{n_1, n_2, n_3} \tilde{f}(n_1, n_2, n_3) e^{i(2\pi n_1 m_1 / N_1)} e^{i(2\pi n_2 m_2 / N_2)} e^{i(2\pi n_3 m_3 / N_3)}$$

$$\tilde{f}(n_1, n_2, n_3) = \frac{1}{N_1 N_2 N_3} \sum_{m_1, m_2, m_3} \tilde{f}(m_1, m_2, m_3) e^{-i(2\pi n_1 m_1 / N_1)} e^{-i(2\pi n_2 m_2 / N_2)} e^{-i(2\pi n_3 m_3 / N_3)}$$

Remember that $\mathbf{G}_i \cdot \mathbf{R}_j = 2\pi\delta_{ij}$.

PW-PP calculations and Discrete Fourier Transform

PW expansion:

$$\psi_i(\mathbf{r}) = \frac{1}{V} \sum_{\mathbf{G}} c_{\mathbf{k}+\mathbf{G}} e^{i(\mathbf{k}+\mathbf{G})\cdot\mathbf{r}}, \quad \frac{\hbar^2}{2m} |\mathbf{k} + \mathbf{G}|^2 \leq E_{cut}$$

Which grid in \mathbf{G} -space? Need to calculate the charge density:

$$n(\mathbf{G}') = \sum_{\mathbf{G}} \sum_{i,\mathbf{k}} f_{i,\mathbf{k}} c_{i,\mathbf{k}+\mathbf{G}}^* c_{i,\mathbf{k}+\mathbf{G}+\mathbf{G}'}$$

Fourier components \mathbf{G}' with $\max(|\mathbf{G}'|) = 2\max(|\mathbf{G}|)$ appear.

Or we need the product of the potential time a wavefunction:

$$(V\psi)(\mathbf{G}) = \sum_{\mathbf{G}'} V(\mathbf{G} - \mathbf{G}') c_{i,\mathbf{k}+\mathbf{G}'}$$

Again, $\max(|\mathbf{G} - \mathbf{G}'|) = 2\max(|\mathbf{G}|)$. We need a kinetic energy cutoff for the Fourier components of the charge and potentials that is four times larger as the cutoff for the PW basis set:

$$\frac{\hbar^2}{2m} |\mathbf{G}|^2 \leq 4E_{cut}$$

In practice such condition may occasionally be relaxed.

Important: for *ultrasoft* pseudopotentials, a different (larger) cutoff for augmentation charges may be needed !

Fourier Transform grid

$$\mathbf{G} = n'_1 \mathbf{G}_1 + n'_2 \mathbf{G}_2 + n'_3 \mathbf{G}_3$$

with $n_1 = 0, \dots, N_1 - 1$, $n_2 = 0, \dots, N_2 - 1$, $n_3 = 0, \dots, N_3 - 1$. This grid must be big enough to include all \mathbf{G} -vectors up to a cutoff

$$\frac{\hbar^2}{2m} |\mathbf{G}|^2 \leq 4E_{cut}$$

and NOT up to the cutoff of the PW basis set! In general, the grid will also contain “useless” Fourier components (beyond the above-mentioned cutoff, so that $n(\mathbf{G}) = 0$, $V(\mathbf{G}) = 0$ etc.)

Fast Fourier Transform

Computational cost of a conventional Fourier Transform of order n : $T_{CPU} = \mathcal{O}(n^2)$.

Computational cost of a Fast Fourier Transform of order n : $T_{CPU} = \mathcal{O}(n \log n)$.

Difference: enormous in practical applications.

Advantages of the use of FFT in PW-PP calculations: enormous, especially in conjunction with iterative techniques and of the “dual-space” technique

Iterative matrix diagonalization

The solution of $(H - \epsilon)\psi_i = 0$ for a large $N \times N$ matrix costs $T_{CPU} = \mathcal{O}(N^3)$. Too much for most applications.

We actually need only the lowest occupied M eigenvectors, with $M \ll N$ (N is the number of PWs).

Iterative diagonalization: based on iterative refinement of a trial solution. Refinement is stopped when the reached accuracy is deemed sufficient. Example: Davidson algorithm.

Very convenient in conjunction with SCF iteration:

- high accuracy not needed in the first iterations
- starting trial wavefunctions available from previous iteration

Iterative matrix diagonalization 2

Basic ingredients: evaluation of products $\phi_i = (H - \epsilon_i)\psi_i$ on trial wavefunctions ψ_i . Same ingredient used in direct minimization, Car-Parrinello, etc.

If the product is calculated as a matrix-vector product: $T_{CPU} = \mathcal{O}(N^2)$ for a single product, $T_{CPU} = \mathcal{O}(MN^2)$ in total. Still much better than conventional diagonalization. But:

- if N becomes large, storing H becomes unpractical, if not impossible;
- much CPU could be spared if an economical way of calculating $H\psi$ was available

Solution: treat H as an operator, taking advantage of FFTs. There is no longer any need to store H as a matrix.

Dual space technique

$$H\psi \equiv (T + \hat{V}_{NL} + V_{loc} + V_H + V_{xc})\psi$$

$(T\psi)$: easy in **G**-space, $T_{CPU} = \mathcal{O}(N)$

$(V_{loc} + V_H + V_{xc})\psi$: easy in **r**-space, $T_{CPU} = \mathcal{O}(N)$

$(\hat{V}_{NL}\psi)$: easy in **G**-space (also in **r**-space) if \hat{V} is written in separable form
 $T_{CPU} = \mathcal{O}(mN)$, m = number of projectors

FFT is used to jump from real to reciprocal space. Operations are performed where it is easier.