arXiv:2603.26818v1 [cs.MS] 26 Mar 2026

用于大规模相场晶体模拟的 MATLAB 多 GPU 快速傅里叶变换

Maik Punke* Marco Salvalaglio*†

2026年3月31日

摘要

我们提出了一个基于 MATLAB 的框架，可在多个图形处理器上执行二维和三维快速傅里叶变换，用于采用伪谱傅里叶方法的大规模数值模拟。该软件实现了两种互补的多 GPU 策略，既突破了单个 GPU 的内存限制，又加速了谱求解器。该方法的开发动机源于相场晶体（PFC）模型，并应用于此类模型；这类模型由十阶偏微分方程支配，需要精细的空间分辨率，且通常采用周期区域进行表述。与运行在数百个核心上的纯基于 CPU 的实现相比，我们构建的数值框架实现了显著的加速比：标准相场晶体模拟约提高六倍，多物理场扩展则最高可提高六十倍。

需求说明

基于伪谱傅里叶方法的大规模模拟 [1] 通常受限于单个图形处理器的内存容量和性能，因为这类模拟需要反复进行多维快速傅里叶变换（FFT）。为解决这一瓶颈，我们提出了两种互补的二维和三维多 GPU 快速傅里叶变换策略。

第一种策略通过域分解和 GPU 间通信，将单个高维快速傅里叶变换分布到多个图形处理器上，从而

*德累斯顿工业大学科学计算研究所，德国德累斯顿 01062
†德累斯顿工业大学德累斯顿计算材料科学中心，德国德累斯顿 01062

能够处理问题规模超出单个图形处理器内存容量的傅里叶变换。第二种策略则将不同的物理场分配给不同的图形处理器，并通过同步通信并行计算多个快速傅里叶变换。

下文将相场晶体（PFC）模型 [2, 3, 4] 作为代表性应用，凸显对此类多 GPU 能力的需求。PFC 模型能够在原子尺度上解析晶体有序结构，同时在较大的（扩散）时间尺度上演化，从而在统一的介观尺度框架内模拟弹性、缺陷、晶界和微观结构演化。捕捉这些现象通常需要相对较大的周期区域和较高的空间分辨率，这会带来巨大的内存需求，并限制单个图形处理器执行时可达到的规模。对于涉及额外变量的多物理场扩展而言，这一点更加重要，例如耦合的密度/组分、速度和/或温度场（参见例如 [5, 6]）。

伪谱傅里叶方法是 PFC 及相关模型中一种自然且广泛采用的数值方法，因为它能够高效而准确地计算其控制方程中出现的高阶空间导数和卷积算子（参见例如 [7, 8, 9, 10, 11, 12]）。然而，其性能主要受反复执行多维快速傅里叶变换的影响。由于控制 PFC动力学的偏微分方程具有非线性，每个时间步都需要进行此类计算，并且在大规模情况下，这些计算很快会成为计算瓶颈。因此，高效的多GPU快速傅里叶变换策略对于充分发挥伪谱求解器的精度和可扩展性至关重要。

作为一个典型应用案例，我们在二维和三维相场晶体（PFC）模型的数值模拟中，采用傅里叶伪谱求解器，利用所开发的多 GPU 并行策略；其中，大尺度空间区域和耦合场使得单 GPU 执行难以实现。相关算法采用 MATLAB 实现，为 GPU 加速谱模拟提供了易于使用且可扩展的基础，并首次在 MATLAB 中实现了多 GPU 快速傅里叶变换。

傅里叶伪谱方法

我们简要回顾傅里叶伪谱方法的基础知识。虽然实际实现是在二维和三维中进行的，但为简化起见，下面的讨论仅限于一维情形。我们考虑一个标量场的一般演化方程

$u \equiv u(x,\ t)$，其形式为

$$\partial_t u = \mathcal{L}u + \mathcal{N}(u), \quad x \in [0, 2\pi),\ t \geq 0, \tag{1}$$

其中，$\mathcal{L}$ 是线性微分算子，而$\mathcal{N}(u)$是一个由 u中的多项式给出的非线性项。

在周期域 $[0, 2\pi)$ 上，我们用截断傅里叶级数近似 $u(x,t)$

$$u(x,t) \approx u_N(x,t) = \sum_{k=-K}^{K} \widehat{u}_k e^{ikx}, \tag{2}$$

其中包含 $N = 2K+1$ 个模态 $\widehat{u}_k \equiv \widehat{u}(k,t)$ 和等间距配置点

$$x_j = \frac{2\pi j}{N}, \quad j = 0, \ldots, N-1. \tag{3}$$

在傅里叶伪谱方法中，通过对方程（1）的左端和右端应用傅里叶变换，可得到关于 $\widehat{u}_k$ 的含时常微分方程，写为

$$\partial_t \widehat{u}_k = \widehat{\mathcal{L}}_k \widehat{u}_k + \widehat{[\mathcal{N}(u)]_k} \qquad \forall k = -K, \ldots, K \tag{4}$$

其中，$\widehat{\mathcal{L}_k}$是关于 k 的多项式，即 $[\widehat{\partial_x^n u}]_k = (ik)^n \widehat{u}_k$； $\forall n \in \mathbb{N}$而非线性多项式项则在每个时间迭代步中于物理空间逐点计算，然后变换到傅里叶空间。对于给定的 $u(x,t)$ 和时间步长 Δt，首先通过快速傅里叶变换计算 $u(x, t+\Delta t)$、 $\widehat{u}_k(t)$ 和 $[\mathcal{N}(u)]_k(t)$，使用适当的时间积分格式求解常微分方程组（4），然后对所得的 $\widehat{u}_k(t+\Delta t)$ 进行逆快速傅里叶变换。该概念可以很容易地扩展到更高的空间维度。更多细节请参见文献 [1]。

多个图形处理器上的单次快速傅里叶变换

基于域分解和 GPU 间通信的分布式多 GPU 快速傅里叶变换已在高性能计算领域得到广泛应用；例如参见 [13, 14, 15, 16]。这些方法能够在大型 GPU 系统上实现可扩展的多维快速傅里叶变换。相比之下，针对相场晶体方程的快速傅里叶变换伪谱求解器迄今仅限于单 GPU 实现 [17] 或基于CPU的并行化 [18, 19]。据我们所知，目前尚无专为大规模三维相场晶体模拟设计的多 GPU 快速傅里叶变换实现。此外，一种通用的

多 GPU 快速傅里叶变换实现尚未在 MATLAB 中提供，且不针对相场晶体模拟等任何特定应用。

按照 [13, 14], 的策略，我们通过沿z方向进行条带分解，将大小为 $N_x \times N_y \times N_z$ 的三维快速傅里叶变换分布到 G 个图形处理器上，见图 1(a)。每个图形处理器首先执行本地二维快速傅里叶变换，然后进行点对点通信以重新分布数据，最后沿剩余维度执行一维快速傅里叶变换。完成后，每个图形处理器都持有变换后傅里叶空间数组的一部分。

我们将这一策略应用于相场晶体方程，该方程描述周期密度场 $\psi \equiv \psi$ (**x** t) [2, 3]的演化。该模型基于自由能泛函，例如，对于面心立方晶体对称性，可将其表示为

$$F[\psi] = \int_{\Omega} \left[\frac{\psi}{2} \left(\varepsilon + \mathcal{L} \right) \psi + \frac{\psi^4}{4} \right] \mathrm{d}\mathbf{r}, \tag{5}$$

其中， ε 为过冷参数，且$\mathcal{L}$ $= (1 + \nabla^2)^2(4/3 + \nabla^2)^2$ 为描述区域 Ω 内空间关联的算子。经典演化方程为

$$\partial_t \psi = \nabla^2 \frac{\delta F[\psi]}{\delta \psi} = (\varepsilon + \mathcal{L}) \nabla^2 \psi + \nabla^2 \psi^3 \tag{6}$$

该方程保留了式 (1) 的形式。我们采用傅里叶伪谱方法和半隐式时间积分 [8] 求解该方程。值得注意的是，也可以考虑其他时间积分方法 [10]。

我们对多 GPU 求解器进行了基准测试，问题规模从 750^3到 1400^3, , 与纯基于 CPU 的实现相比，加速比最高可达六倍；见图 1(a)。基准测试在德累斯顿工业大学国家高性能计算联盟中心提供的高性能计算集群中的三套系统上进行。图形处理器计算分别使用四块 NVIDIA H100 SXM5 GPU（每块配备 94 GiB HBM2e，位于 Capella 高性能计算集群）和八块 NVIDIA A100 SXM4 GPU（每块配备 40 GiB HBM2，位于半人马座阿尔法星高性能计算集群）。CPU 参考运行在一颗 Intel Xeon Platinum 8470 处理器上（100 个核心，2.00 GHz，位于 Barnard 高性能计算集群）。清单 1 展示了 MATLAB 实现。

图2(a)展示了 PFC 框架中的枝晶凝固，作为一个具有代表性的二维基准示例。

多物理场 PFC 的多个 GPU 使用

PFC 框架可轻松扩展至多物理场。举例来说，我们考虑三维空间中的流体动力学相场晶体（流体动力学 PFC）模型 [5, 20]，该模型在密度场ψ的基础上增加了介观速度场 $\mathbf{v} \equiv (\mathbf{v}_1(\mathbf{x}\ t), \mathbf{v}_2(\mathbf{x}\ t), \mathbf{v}_3(\mathbf{x}\ t)$：

$$\begin{aligned} \partial_t \psi &= \nabla^2 \left(\frac{\delta F[\psi]}{\delta \psi} \right) - \mathbf{v} \cdot \nabla \psi, \\ \rho \partial_t \mathbf{v} &= \Gamma \nabla^2 \mathbf{v} - \left\langle \psi \nabla \frac{\delta F[\psi]}{\delta \psi} \right\rangle, \end{aligned} \tag{7}$$

通过与高斯核卷积得到的 $\langle\ \cdot\ \rangle$ 局部平均

$$\langle\ \cdot\ \rangle(\mathbf{r}) = \int_\Omega \frac{(\ \cdot\ )(\mathbf{r}')}{\left(2\pi a_0^2\right)^{3/2}} \exp\left(-\frac{(\mathbf{r}-\mathbf{r}')^2}{2a_0^2} \right) d\mathbf{r}' \tag{8}$$

以及 a_0 晶格间距。请注意，得益于傅里叶变换的性质，在傅里叶谱方法中，这一操作等价于乘法。

我们将场分配到四个 GPU 上，每个设备分配一个场，并在每个时间步之后执行同步的 GPU 间通信。这一策略使我们能够处理超出单个 GPU 内存限制的问题。使用四个 NVIDIA H100 GPU 进行的基准测试表明，与此前介绍的高性能计算集群 Capella 和 Barnard 上的 CPU 执行相比，运行时间显著缩短。尤其是在单个 GPU 无法进行模拟的大型计算域中，性能提升十分显著；见图 1(b)。清单 2 展示了 MATLAB 实现。图2(b) 描绘了流体动力学 PFC 框架中的多晶粗化，作为此处展示的一个代表性三维基准测试案例。

将不同场分配到多个 GPU 上的策略，自然可以扩展到基于主傅里叶模复振幅的粗粒化表述 [21]。这些模型非常适合伪谱方法，通常需要同时演化数十个耦合的复值场；我们预计，当前的并行化策略尤其适用于这种情形。

实现片段

清单 1：多个图形处理器上的单次快速傅里叶变换

```
%%initialize and decompose into slabs
% psi ...initial density field
% psiF...Fourier transformed initial density field
% lap ...discretized laplacian
% lin ...discretized linear operator (epsilon+L^2)nabla^2

spmd %parallel GPU session (G GPUs), on each GPU perform
    for  n=1:n_timeSteps %time iteration
        %%forward fftn
        psi = fft2(psi.^3); %fft2 along first two
            dimensions
        for i=1:G %P2P communication
            psi_chunk = psi(localStart:localStart,:,:); %
                decompose each slab in a set of local
                chunks
            psi_chunk= spmdCat(psi_chunk,3,i); %stack the
                 chunks together (along third dimension)
        end
        psi = psi_chunk;
        psi = fft(psi,[],3);  %fft along third dimension

        %%semi-implicit time step update
        psiF = (psiF+dt*lap.*psi)./(1-dt*lin);

        %%backward ifftn
        psi = psiF;
        psi  = ifft(psi,[],3); %ifft along third
            dimension
        for i=1:G %P2P communication
            psi_chunk = psi(:,:,localStart:localStart);%
                decompose each slab in a set of local
                chunks
            psi_chunk= spmdCat(psi_chunk,1,i); %stack the
                 chunks together (along first dimension)
        end
        psi = psi_chunk;
        psi  = ifft2(psi); %ifft2 along first two
            dimensions
    end
end

```

```
%% stack solution
psi = cat(3, gather(psi{:}));
```

代码清单 2：多物理场 PFC 的多图形处理器用法

```
%%initialize on GPU1
% psi ...initial density field
% psiF...Fourier transformed initial density field
% v1  ...initial velocity field (first component)
% v2  ...initial velocity field (secod component)
% v3  ...initial velocity field (third component)
% lin ...discretized linear operator (epsilon+L^2)nabla^2
% k   ...discretized Fourier vector (first dimension)
% l   ...discretized Fourier vector (second dimension)
% m   ...discretized Fourier vector (third dimension)

%%initialize on GPU2...GPU4
% lap ...discretized laplacian
% k   ...discretized Fourier vector (first dimension)
% op  ...discretized linear operator (epsilon+L^2)
% cg  ... discretized convolution kernel
% additionally initialize on GPU2
% v1F ...Fourier transformed initial v1
% additionally initialize on GPU3
% v2F ...Fourier transformed initial v2
% additionally initialize on GPU4
% v3F ...Fourier transformed initial v3

spmd %parallel GPU session (4 GPUs)
    for  n=1:n_timeSteps %time iteration
        if spmdIndex==1 %GPU1 holding psi
            %%semi-implicit time step update of psi
            psiF = (psiF+dt*(lap.*fftn(psi.^3)-fftn(v1.*
                ifftn(k.*psiF)+v2.*ifftn(l.*psiF)+v3IF.*
                ifftn(m.*psiF)))./(1-dt*lin);
            psi = ifftn(psiF);
            %%send psi to GPU2...GPU4
            spmdSend(psi,[2:4],2);
            %%receive v components from GPU2...GPU4
            v1 = spmdReceive(2,4);
            v2 = spmdReceive(3,5);
            v3 = spmdReceive(4,6);

        elseif spmdIndex==2 %GPU2 holding v1
            %receive psi from GPU1
```

```
            psi = spmdReceive(1,2);
            %%fully-implicit time step update of v1
            v1F = (v1F-dt/rho0 .*cg.*fftn(psi.*ifftn(k.*(
                fftn(psi.^3) + op.*fftn(psi)))))
                 ./(1-deltatEuler/rho0.*.GammaS.*lap));
            v1 = ifftn(v1F);
            %send v1 to GPU1
            spmdSend(v1,1,4);

        elseif spmdIndex==3 %GPU3 holding v2
            %receive psi from GPU1
            psi = spmdReceive(1,2);
            %%fully-implicit time step update of v2
            v2F = (v2F-dt/rho0 .*cg.*fftn(psi.*ifftn(l.*(
                fftn(psi.^3) + op.*fftn(psi)))))
                 ./(1-deltatEuler/rho0.*.GammaS.*lap));
            v2 = ifftn(v2F);
            %send v2 to GPU1
            spmdSend(v2,1,5);

        elseif spmdIndex==4 %GPU4 holding v3
            %receive psi from GPU1
            psi = spmdReceive(1,2);
            %%fully-implicit time step update of v3
            v3F = (v3F-dt/rho0 .*cg.*fftn(psi.*ifftn(m.*(
                fftn(psi.^3) + op.*fftn(psi)))))
                 ./(1-deltatEuler/rho0.*.GammaS.*lap));
            v3 = ifftn(v3F);
            %send v3 to GPU1
            spmdSend(v3,1,6);
        end
    end
end
```

可用性

代码仓库： https://github.com/mpunke/MATLABmultiGPUFFT/**许可证：**
MIT 许可证**Zenodo 数字对象标识符：**
https://doi.org/10.5281/zenodo.18670913

致谢

我们感谢德国研究联合会（DFG，德国研究基金会）对项目（编号 447241406、493401063）的资助。我们还衷心感谢使用德累斯顿工业大学 NHR 中心的高性能计算机进行计算。该中心由联邦教育与研究部以及参与 NHR 的各州政府共同支持（www.nhr-verein.de/unsere-partner）。

参考文献

[1] Boyd J P 2001 切比雪夫与傅里叶谱方法 (CourierCorpo- ration) [2] Elder K R, Katakowski M, Haataja M and Grant M 2002 《物理评论快报》**88** 245701[3] Elder K R and Grant M 2004 《物理评论 *E*》 **70**051605[4] Emmerich H, Löwen H, Wittkowski R, Gruhn T, Tóth G I, Tegze G and Gránásy L 2012 《先进物理学》 **61**665–743[5] Skogvoll V, Salvalaglio M and Angheluta L 2022 《材料科学与工程中的建模与仿真》 **30**084002[6] Punke M, Wise S M, Voigt A and Salvalaglio M 2022 《材料科学与工程中的建模与仿真》 **30**074004[7] Cheng M and Warren J A 2008 《计算物理学杂志》 **227**6241–6248[8] Tegze G, Bansel G, Tóth G I, Pusztai T, Fan Z and Gránásy L 2009《计算物理学杂志》 **228**1612–1623[9] Cheng K, Wang C and Wise S M 2019 计算物理通讯 **261**335–1364[10] Punke M, Wise S M, Voigt A and Salvalaglio M 2023 《应用数学与力学》 **23**e202200112[11] Punke M and Salvalaglio M 2025 应用力学与工程中的计算方法 **436**117719

[12]Punke M, Milor A H and Salvalaglio M 2026 *arXiv* 预印本 *arXiv:2601.13965*[13] Pekurovsky D 2012 *SIAM* 科学计算杂志 **34**C192–C209[14] Ayala A, Tomov S, Haidar A and Dongarra J 2020 heffte：面向百亿亿级计算的高效 FFT国际计算科学会议 (施普林格) 第 262–275 页[15] Verma M, Chatterjee S, Garg G, Sharma B, Arya N, Kumar S, Saxena A and Verma M K 2023 *SN* 计算机科学 **4** 625[16] 英伟达公司 2023 cufftmp：适用于多 GPU 和多节点系统的分布式快速傅里叶变换 `https://developer.nvidia.com/cufft`[17] Hallberg H and Blixt K H 2025 《材料科学与工程中的建模与仿真》 **34**015004[18] Pinomaa T, Aho J, Suviranta J, Jreidini P, Provatas N and Laukkanen A 2024 《材料科学与工程中的建模与仿真》**32** 045002[19] Skogvoll V and Rønning J 2024 开源软件杂志 **9** 6599[20] Qiu C, Punke M, Tian Y, Han Y, Wang S, Su Y, Salvalaglio M, Pan X, Srolovitz D J and Han J 2024 《科学》 **385**980–985[21] Salvalaglio M and Elder K R 2022材料科学与工程建模与仿真**30**053001

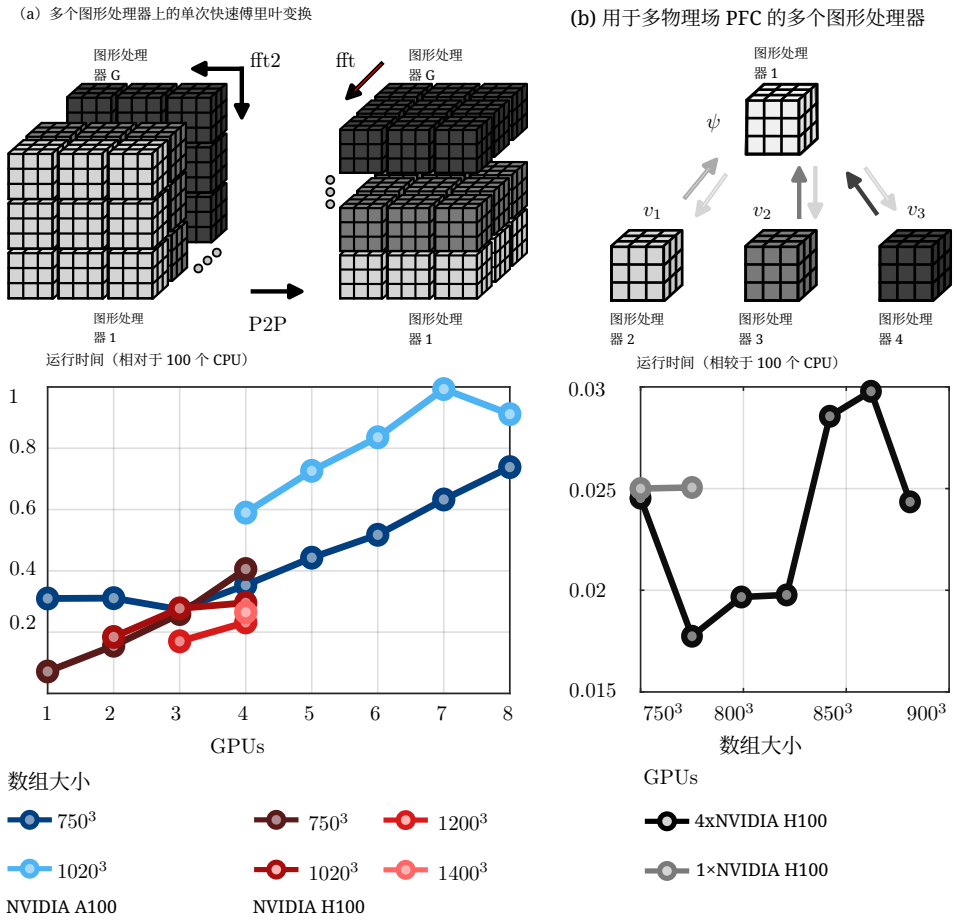

图1：(a) 基于条带分解的三维数组多 GPU 快速傅里叶变换算法示意图。数据沿 z方向分解，随后进行局部二维快速傅里叶变换、点对点通信，最后执行一维快速傅里叶变换（上图）。图中显示了 1000 个时间步的相对运行时间，并以 CPU 执行时间为基准进行归一化。加速比最高可达 6 倍；对于 750^3 ，单个图形处理器的性能最佳；由于内存限制，处理更大的区域则需要多 GPU 执行。大小为 1400^3 的数组只能在四个 H100 图形处理器上运行，其运行时间约为 CPU 执行时间的 17%（下图）。(b) 伪谱多物理场 PFC 求解器（例如流体动力学 PFC）在四个图形处理器上的分解示意图（上图）。图中显示了流体动力学 PFC 求解器的相对运行时间随问题规模的变化，表明多 GPU 执行可支持问题规模达到 900^3, 的模拟，而这在单个图形处理器上无法实现。与 CPU 实现相比，加速比最高可达 $60\times$ （下图）。

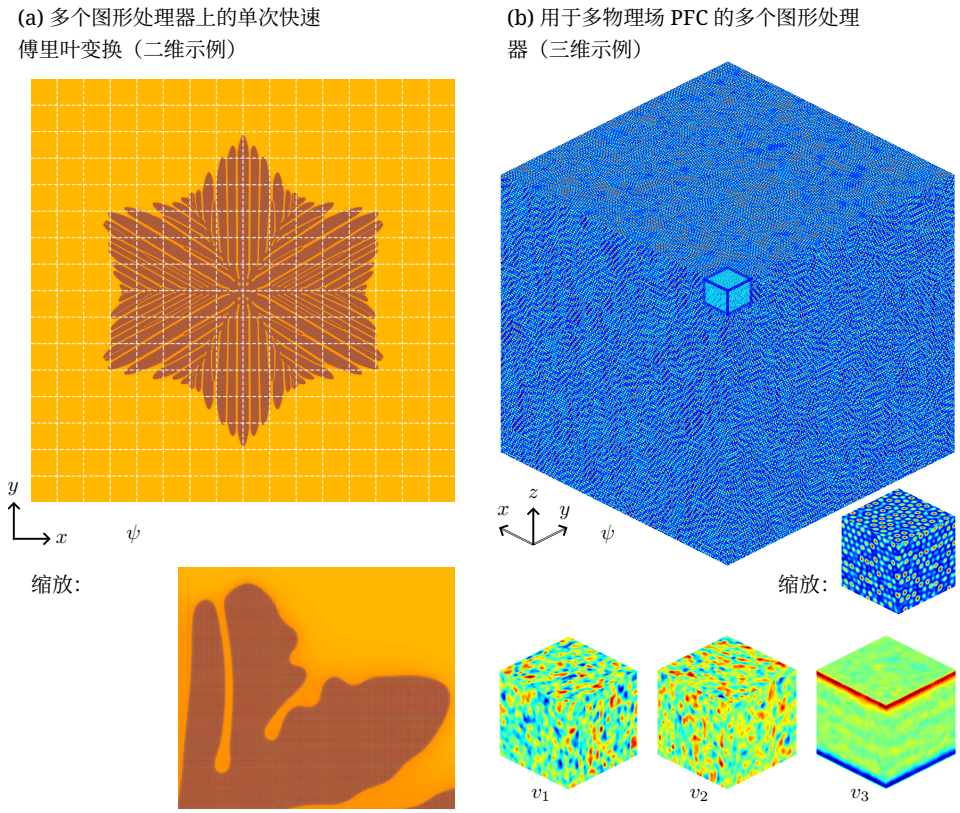

图2：多 GPU 快速傅里叶变换算法的典型大规模 PFC 基准问题：(a) 使用多图形处理器单次快速傅里叶变换实现的枝晶凝固（具有三角晶体对称性）（二维示例；计算域大小为 $5\cdot\ 10^4 \times 5\cdot\ 10^4,$，假设铝的晶格常数为 4Å 时，对应于$2.5\,\mu\mathrm{m}\ \times 2.5\ \mu\mathrm{m}$）。图中展示了密度场 ψ 及其局部放大图。白线标示分块，因为数组尺寸超过了单图绘制的限制。(b) 使用多图形处理器流体动力学相场晶体求解器模拟面心立方晶体结构的多晶粗化。图中展示了密度场 ψ （包括放大图）和速度分量$\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$。采用了 $1400 \times 1400 \times 1400$ 的网格，对应的盒子尺寸为 $40\ \mathrm{nm}\ \times 40\,\mathrm{nm}\ \times 40\,\mathrm{nm}$。材料参数和模型参数见本研究附带的代码仓库。