On compact finite differences for the Poisson equation

Peter Arbenz, IT4I Ostrava/ETH Zurich

Talk at TU Ostrava, September 3, 2020

Outline

Introduction

1D Poisson problems

2D Poisson problems

3D Poisson problems

Conclusions

Motivation from beam dynamics

1. Vlasov-Poisson formulation for particle evolution

- In physical devices like accelerators $10^9 \ldots 10^{14}$ (or more) charged particles are accelerated in electric fields.
- Instead of computing with individual particles one considers particle density $f(\mathbf{x}, \mathbf{v}, t)$ in phase space (position-velocity $(\mathbf{x}, \mathbf{v})$ space).
- *Vlasov equation* describes the evolving particle density

$$\frac{df}{dt} = \partial_t f + \mathbf{v} \cdot \nabla_{\mathbf{x}} f + \frac{q}{m_0}(\mathbf{E} + \mathbf{v} \times \mathbf{B}) \cdot \nabla_{\mathbf{v}} f = 0,$$

  where $\mathbf{E}$ and $\mathbf{B}$ are electric and magnetic fields, respectively.
- The charged particles are 'pushed' by Newton's law

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{v}, \quad \frac{d\mathbf{v}(t)}{dt} = \frac{q}{m_0}(\mathbf{E} + \mathbf{v} \times \mathbf{B}).$$

Motivation from beam dynamics (cont.)

- The determination of **E** and **B** is done in the co-moving Lorentz frame where $\hat{\mathbf{B}} \approx \mathbf{0}$ and

$$\hat{\mathbf{E}} = -\nabla\hat{\phi},$$

where the electrostatic potential $\hat{\phi}$ is the solution of the *Poisson problem*

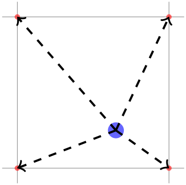
$$-\Delta\hat{\phi}(\mathbf{x}) = \frac{\hat{\rho}(\mathbf{x})}{\varepsilon_0}, \tag{1}$$

equipped with appropriate boundary conditions.

- The charge densities ρ is proportional to the particle density.

Motivation from beam dynamics (cont.)

2. Particle-in-cell (PIC) method in N-body Simulations



- Interpolate individual particle charges to a rectangular grid
- Discretize the Poisson equation by finite differences on the rectangular grid
- This leads to a system of linear equations

$$\mathbf{A}\mathbf{x} = \mathbf{b}. \tag{2}$$

  $\mathbf{b}$ denotes the interpolated charge densities at the mesh points.

- Solve the Poisson equation on the mesh in a Lorentz frame
- $\mathcal{O}(n \log n)$ operations needed provided that the domain is rectangular.

Purpose of the talk

- Poisson equation on rectangular domains often solved by finite differences (5-point stencil).
  Ditto in 3D with the 7-point stencil.
- These methods converge with $\mathcal{O}(h^2)$ in the mesh width h.
- Higher orders of accuracy requires bigger stencils or more brain.
- Higher orders of accuracy lead to (much) smaller linear systems of equations for the same accuracy.
- We discuss how to get fourth order compact finite difference schemes.
- Emphasis is on rectangular grids and on fast (FFT-based) Poisson solvers.

References

1. L. Collatz. *Numerische Behandlung von Differentialgleichungen*. Springer, Berlin-Heidelberg, 1951. ($\longrightarrow$ *Mehrstellenmethode*)
2. R. J. LeVeque. *Finite Difference Methods for Ordinary and Partial Differential Equations*. SIAM, 2007.
3. W. F. Spotz and G. F. Carey. A high-order compact formulation for the 3D Poisson equation. *Numer. Methods Partial Differ. Equations*, 12:235–243, 1996.
4. S. O. Settle, C. C. Douglas, I. Kim, and D. Sheen. On the derivation of highest-order compact finite difference schemes for the one- and two-dimensional Poisson equation with Dirichlet boundary conditions. *SIAM J. Numer. Anal.*, 51:2470–2490, 2013.
5. E. Deriaz. Compact finite difference schemes of arbitrary order for the Poisson equation in arbitrary dimensions. *BIT Numer. Math.*, 60:199–233, 2020.

The 1D case: problem statement

- Interval $I = (0, a)$
- Poisson equation:

$$-u''(x) = f(x), \quad 0 < x < a, \qquad u(0) = u(a) = 0.$$

- Equidistant mesh $0 = x_0 < x_1 < \cdots < x_n < x_{n+1} = a$.
- Mesh width $h = x_j - x_{j-1} = a/(n+1)$.
- Approximation $u_j \approx u(x_j)$.
- Approximate Poisson equation by

$$\frac{-u_{j-1} + 2u_j - u_{j+1}}{h^2} = f(x_j), \qquad 1 \le j \le n. \tag{3}$$

The 1D case: linear system

The n equations in (3) can be collected in matrix equation

$$\frac{1}{h^2}\boldsymbol{T}_n\boldsymbol{u} = \frac{1}{h^2}\begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix}\begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{n-1} \\ u_n \end{bmatrix} = \begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_{n-1}) \\ f(x_n) \end{bmatrix} = \boldsymbol{f}.$$

$\boldsymbol{T}_n \in \mathbb{R}^{n\times n}$ has the spectral decomposition

$$\boldsymbol{T}_n = \boldsymbol{Q}_n\boldsymbol{\Lambda}_n\boldsymbol{Q}_n^T, \qquad (4)$$

with diagonal $\boldsymbol{\Lambda}_n$

$$\boldsymbol{\Lambda}_n = \mathrm{diag}(\lambda_1^{(n)},\ldots,\lambda_n^{(n)}), \qquad \lambda_k^{(n)} = 4\sin^2\frac{k\pi}{2(n+1)}. \qquad (5)$$

The 1D case: linear system (cont.)

$\boldsymbol{Q}_n$ is orthogonal, i.e., $\boldsymbol{Q}_n^{-1} = \boldsymbol{Q}_n^T$, with elements

$$q_{jk} = \left(\frac{2}{n+1}\right)^{1/2} \sin \frac{jk\pi}{n+1}.$$

Multiplying with $\boldsymbol{Q}_n$ or $\boldsymbol{Q}_n^T$ is related to the Fourier transform.

If n is chosen properly then the Fast Sine Transform ($\sim$Fast Fourier Transform) can be employed to solve (3).

This does not make much sense in the 1D case, since the direct solution (Gaussian elimination) costs only $\mathcal{O}(n)$ flops.

The 1D case: local truncation error

The local truncation error is obtained by plugging the exact solution in the FD formula,

$$\frac{-u(x-h)+2u(x)-u(x+h)}{h^2}-f(x)=\tau(x;h)$$

The 1D case: local truncation error (cont.)

A Taylor series expansion gives

$$\begin{aligned}
u(x_{j-1}) - 2u(x_j) + u(x_{j+1}) \\
&= u(x_j) - hu'(x_j) + \frac{h^2}{2}u''(x_j) - \frac{h^3}{6}u'''(x_j) + \frac{h^4}{24}u''''(x_j) + \cdots \\
&- 2u(x_j) \\
&+ u(x_j) + hu'(x_j) + \frac{h^2}{2}u''(x_j) + \frac{h^3}{6}u'''(x_j) + \frac{h^4}{24}u''''(x_j) + \cdots \\
&= h^2 u''(x_j) + \frac{h^4}{12}u''''(x_j) + \mathcal{O}(h^6)
\end{aligned}$$

The 1D case: local truncation error (cont.)

A Taylor series expansion gives

$$
\begin{aligned}
&u(x_{j-1}) - 2u(x_j) + u(x_{j+1}) \\
&\qquad = h^2 u''(x_j) + \frac{h^4}{12} u''''(x_j) + \mathcal{O}(h^6)
\end{aligned}
$$

or, using $-u''(x) = f(x)$,

$$
\frac{-u(x_{j-1}) + 2u(x_j) - u(x_{j+1})}{h^2} = f(x_j) \underbrace{- \frac{h^2}{12} u''''(x_j) + \mathcal{O}(h^4)}_{\tau(x_j)} \tag{6}
$$

The 1D case: local truncation error (cont.)

A Taylor series expansion gives

$$u(x_{j-1}) - 2u(x_j) + u(x_{j+1}) \\ = h^2 u''(x_j) + \frac{h^4}{12} u''''(x_j) + \mathcal{O}(h^6)$$

or, using $-u''(x) = f(x)$,

$$\frac{-u(x_{j-1}) + 2u(x_j) - u(x_{j+1})}{h^2} = f(x_j) \underbrace{- \frac{h^2}{12} u''''(x_j) + \mathcal{O}(h^4)}_{\tau(x_j)} \quad (6)$$

The 1D case: global error

$$\frac{-u_{j-1} + 2u_j - u_{j+1}}{h^2} = f(x_j). \tag{7}$$

$$\frac{-u(x_{j-1}) + 2u(x_j) - u(x_{j+1})}{h^2} = -u''(x_j) + \tau(x_j). \tag{8}$$

Subtracting (7) from (8) we get for the error $e(x_j) = u(x_j) - u_j$

$$h^{-2}\boldsymbol{T}_n\boldsymbol{e} = \boldsymbol{\tau}.$$

So, the L_2-error behaves like the local truncation error since

$$\|h^2\boldsymbol{T}_n^{-1}\|_2 < \frac{a^2}{8} \quad \text{for all } n.$$

$8/a^2$ is a lower bound for the smallest eigenvalue of $h^{-2}\boldsymbol{T}_n$.

The 1D case: Improving accuracy

1. Use longer stencil

$$\frac{1}{12h^2}(-u_{j-2} + 16u_{j-1} - 30u_j + 16u_{j+1} - u_{j+2}) = f(x_j)$$

2.

The 1D case: Improving accuracy

1. Use longer stencil

$$\frac{1}{12h^2}(-u_{j-2} + 16u_{j-1} - 30u_j + 16u_{j+1} - u_{j+2}) = f(x_j)$$

2. Closer look at truncation error

$$\frac{-u(x_{j-1}) + 2u(x_j) - u(x_{j+1})}{h^2} = -u''(x_j) + \tau(x_j).$$

The 1D case: Improving accuracy

1. Use longer stencil

$$\frac{1}{12h^2}(-u_{j-2} + 16u_{j-1} - 30u_j + 16u_{j+1} - u_{j+2}) = f(x_j)$$

2. Closer look at truncation error

$$\frac{-u(x_{j-1}) + 2u(x_j) - u(x_{j+1})}{h^2} = -u''(x_j) - \frac{h^2}{12}u''''(x_j) + \mathcal{O}(h^4)$$

The 1D case: Improving accuracy

1. Use longer stencil

$$\frac{1}{12h^2}(-u_{j-2}+16u_{j-1}-30u_j+16u_{j+1}-u_{j+2})=f(x_j)$$

2. Closer look at truncation error

$$\frac{-u(x_{j-1})+2u(x_j)-u(x_{j+1})}{h^2}=-u''(x_j)-\frac{h^2}{12}u''''(x_j)+\mathcal{O}(h^4)$$

Replace finite difference stencil by

$$\frac{-u_{j-1}+2u_j-u_{j+1}}{h^2}=f(x_j)+\frac{h^2}{12}f''(x_j) \tag{9}$$

The 1D case: Improving accuracy

1. Use longer stencil

$$\frac{1}{12h^2}(-u_{j-2} + 16u_{j-1} - 30u_j + 16u_{j+1} - u_{j+2}) = f(x_j)$$

2. Closer look at truncation error

$$\frac{-u(x_{j-1}) + 2u(x_j) - u(x_{j+1})}{h^2} = -u''(x_j) - \frac{h^2}{12}u''''(x_j) + \mathcal{O}(h^4)$$

Replace finite difference stencil by

$$\frac{-u_{j-1} + 2u_j - u_{j+1}}{h^2} = f(x_j) + \frac{h^2}{12}f''(x_j) \tag{9}$$

or

$$\frac{-u_{j-1} + 2u_j - u_{j+1}}{h^2} = f(x_j) + \frac{1}{12}(f(x_{j-1}) - 2f(x_j) + f(x_{j+1})) \tag{10}$$

The 1D case: Matlab demo

`generate_convergence_plot1D`

The 2D case: problem statement

- Rectangle $\Omega = (0, a_x) \times (0, a_y)$
- Poisson equation

$$-\nabla^2 u(x,y) = f(x,y) \quad \text{in } \Omega, \qquad u = 0 \text{ on } \partial\Omega. \tag{11}$$

- Rectangular mesh: $n_x + 2 \times n_y + 2$ grid points (incl. boundary)
- Mesh widths: $h_x = a_x/(n_x+1)$ and $h_y = a_y/(n_y+1)$
- 5-point stencil is most used approximation of the Laplacian
- Approximation $u_{ij} \approx u(x_i, y_j)$
- Approximate Poisson equation by

$$\frac{-u_{i-1,j} + 2u_{ij} - u_{i+1,j}}{h_x^2} + \frac{-u_{i,j-1} + 2u_{ij} - u_{i,j+1}}{h_y^2} = f(x_i, y_j) \tag{12}$$

for $0 < i \le n_x, 0 < j \le n_y$.

The 2D case: stencil

Often, the discretized Poisson equation is displayed as a *stencil*

$$-\nabla_5^2 u(x,y) = \begin{array}{ccc} & -\frac{1}{h_y^2} & \\ -\frac{1}{h_x^2} & \frac{2}{h_x^2}+\frac{2}{h_y^2} & -\frac{1}{h_x^2} \\ & -\frac{1}{h_y^2} & \end{array} \odot u(x,y) = f(x,y)$$

which shows nicely the five involved grid points with their weights.

The 2D case: linear system

Collect the $u_{ij}/f(x_i, y_j)$ in a vector $\boldsymbol{u}, \boldsymbol{f} \in \mathbb{R}^{n_x n_y}$.

The $n_x n_y$ equations in (12) can be collected in matrix form

$$\left(\frac{1}{h_x^2}\boldsymbol{I}_{n_y} \otimes \boldsymbol{T}_{n_x} + \frac{1}{h_y^2}\boldsymbol{T}_{n_y} \otimes \boldsymbol{I}_{n_x}\right)\boldsymbol{u} = \boldsymbol{f}, \tag{13}$$

where $\otimes$ denotes Kronecker product. Using the spectral decomposition (4) of $\boldsymbol{T}_n$, (13) can be written as

$$(\boldsymbol{Q}_{n_y} \otimes \boldsymbol{Q}_{n_x})(\frac{1}{h_x^2}\boldsymbol{I}_{n_y} \otimes \boldsymbol{\Lambda}_{n_x} + \frac{1}{h_y^2}\boldsymbol{\Lambda}_{n_y} \otimes \boldsymbol{I}_{n_x})(\boldsymbol{Q}_{n_y}^T \otimes \boldsymbol{Q}_{n_x}^T)\boldsymbol{u} = \boldsymbol{f}. \tag{14}$$

Matrix in the middle is diagonal.

With $n = n_x n_y$, (14) can be solved with $\mathcal{O}(n \log n)$ flops, if FFT is applicable.

The 2D case: truncation error

Local truncation error for 5-point stencil is

$$-\nabla_5^2 u(x,y) - f(x,y) = -\frac{h_x^2}{12}\partial_x^4 u(x,y) - \frac{h_y^2}{12}\partial_y^4 u(x,y) + \mathcal{O}(h_x^4 + h_y^4).$$

Can we do better in 2D as well?

The 2D case: improving accuracy

Define a 9-point (compact) stencil

$$\nabla_9^2 u_{i,j} \equiv \nabla_5^2 u_{i,j} + \frac{1}{12}\Big(4u_{i,j} - 2(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}) + u_{i+1,j+1} + u_{i-1,j+1} + u_{i+1,j-1} + u_{i-1,j-1}\Big)\left(\frac{1}{h_x^2} + \frac{1}{h_y^2}\right).$$

For the local truncation error of the Poisson equation we get

$$-\nabla_9^2 u(x,y) - f(x,y) = -\frac{h_x^2}{12}\left(\partial_x^4 u(x,y) + \partial_x^2\partial_y^2 u(x,y)\right) - \frac{h_y^2}{12}\left(\partial_x^2\partial_y^2 u(x,y) + \partial_y^4 u(x,y)\right) + \mathcal{O}((h_x^2 + h_y^2)^2),$$

which does not look like an improvement w.r.t. the 5-pt stencil.

The 2D case: improving accuracy (cont.)

BUT

$$
\begin{aligned}
-\nabla_9^2 u(x,y) - f(x,y) = &-\frac{h_x^2}{12}\left(\partial_x^4 u(x,y) + \partial_x^2\partial_y^2 u(x,y)\right) \\
&-\frac{h_y^2}{12}\left(\partial_x^2\partial_y^2 u(x,y) + \partial_y^4 u(x,y)\right) + \mathcal{O}((h_x^2 + h_y^2)^2)
\end{aligned}
$$

The 2D case: improving accuracy (cont.)

BUT

$$
\begin{aligned}
-\nabla_9^2 u(x,y) - f(x,y) = &- \frac{h_x^2}{12}\left(\partial_x^4 u(x,y) + \partial_x^2\partial_y^2 u(x,y)\right) \\
&- \frac{h_y^2}{12}\left(\partial_x^2\partial_y^2 u(x,y) + \partial_y^4 u(x,y)\right) + \mathcal{O}((h_x^2 + h_y^2)^2) \\
= &- \frac{h_x^2}{12}\left(\partial_x^2(\partial_x^2 u(x,y) + \partial_y^2 u(x,y))\right) \\
&- \frac{h_y^2}{12}\left(\partial_y^2(\partial_x^2 u(x,y) + \partial_y^2 u(x,y))\right) + \cdots
\end{aligned}
$$

The 2D case: improving accuracy (cont.)

BUT

$$
\begin{aligned}
-\nabla_9^2 u(x,y) - f(x,y) =& -\frac{h_x^2}{12}\left(\partial_x^4 u(x,y) + \partial_x^2\partial_y^2 u(x,y)\right) \\
& -\frac{h_y^2}{12}\left(\partial_x^2\partial_y^2 u(x,y) + \partial_y^4 u(x,y)\right) + \mathcal{O}((h_x^2+h_y^2)^2) \\
=& -\frac{h_x^2}{12}\left(\partial_x^2(\partial_x^2 u(x,y) + \partial_y^2 u(x,y))\right) \\
& -\frac{h_y^2}{12}\left(\partial_y^2(\partial_x^2 u(x,y) + \partial_y^2 u(x,y))\right) + \cdots \\
=& -\frac{h_x^2}{12}\,\partial_x^2\nabla^2 u(x,y) - \frac{h_y^2}{12}\,\partial_y^2\nabla^2 u(x,y) + \cdots
\end{aligned}
$$

The 2D case: improving accuracy (cont.)

BUT

$$
\begin{aligned}
-\nabla_9^2 u(x,y) - f(x,y) = &-\frac{h_x^2}{12}\left(\partial_x^4 u(x,y) + \partial_x^2\partial_y^2 u(x,y)\right) \\
&-\frac{h_y^2}{12}\left(\partial_x^2\partial_y^2 u(x,y) + \partial_y^4 u(x,y)\right) + \mathcal{O}((h_x^2+h_y^2)^2) \\
= &-\frac{h_x^2}{12}\left(\partial_x^2(\partial_x^2 u(x,y) + \partial_y^2 u(x,y))\right) \\
&-\frac{h_y^2}{12}\left(\partial_y^2(\partial_x^2 u(x,y) + \partial_y^2 u(x,y))\right) + \cdots \\
= &-\frac{h_x^2}{12}\,\partial_x^2\nabla^2 u(x,y) - \frac{h_y^2}{12}\,\partial_y^2\nabla^2 u(x,y) + \cdots \\
= &\frac{h_x^2}{12}\,\partial_x^2 f(x,y) + \frac{h_y^2}{12}\,\partial_y^2 f(x,y) + \mathcal{O}((h_x^2+h_y^2)^2)
\end{aligned}
$$

The 2D case: improving accuracy (cont.)

If the second derivatives of f not available or too expensive to compute, replace them by finite differences:

$$\frac{1}{12}\times\begin{bmatrix} -\frac{1}{h_x^2}-\frac{1}{h_y^2} & \frac{2}{h_x^2}-\frac{10}{h_y^2} & -\frac{1}{h_x^2}-\frac{1}{h_y^2} \\ \frac{2}{h_y^2}-\frac{10}{h_x^2} & \frac{20}{h_x^2}+\frac{20}{h_y^2} & \frac{2}{h_y^2}-\frac{10}{h_x^2} \\ -\frac{1}{h_x^2}-\frac{1}{h_y^2} & \frac{2}{h_x^2}-\frac{10}{h_y^2} & -\frac{1}{h_x^2}-\frac{1}{h_y^2} \end{bmatrix}\odot u(x,y)=\frac{1}{12}\times\begin{bmatrix} & 1 & \\ 1 & 8 & 1 \\ & 1 & \end{bmatrix}\odot f(x,y).$$

A fourth order local truncation error is the best one can get in 2D by compact FD (Settle et al. SINUM 2013).

The 2D case: improving accuracy (cont.)

Truncation error with 4-th order terms exposed:

$$
\begin{aligned}
\tau(x,y) = &- \nabla_9^2 u(x,y) - f(x,y) - \frac{h_x^2}{12}\partial_x^2 f(x,y) - \frac{h_y^2}{12}\partial_y^2 f(x,y) = \\
&- \frac{h_x^4}{720}\left(2\partial_x^6 u(x,y) + 5\partial_x^4\partial_y^2 u(x,y)\right) \\
&- \frac{h_x^2 h_y^2}{144}\left(\partial_x^4\partial_y^2 u(x,y) + \partial_x^2\partial_y^4 u(x,y)\right) \\
&- \frac{h_y^4}{720}\left(5\partial_x^2\partial_y^4 u(x,y) + 2\partial_y^6 u(x,y)\right) \\
&+ \mathcal{O}((h_x^2 + h_y^2)^3).
\end{aligned}
$$

If grid is square ($h = h_x = h_y$) then the fourth order term can be expressed as $\frac{h^4}{360}(\nabla^4 f + 2\partial_x^2\partial_y^2 f)$.

The 2D case: linear system for compact FD

The matrix form of the stencil before is

$$\left(\frac{1}{h_x^2}\boldsymbol{I}_{n_y}\otimes\boldsymbol{T}_{n_x}+\frac{1}{h_y^2}\boldsymbol{T}_{n_y}\otimes\boldsymbol{I}_{n_x}-\frac{1}{12}\left(\frac{1}{h_x^2}+\frac{1}{h_y^2}\right)\boldsymbol{T}_{n_y}\otimes\boldsymbol{T}_{n_x}\right)\boldsymbol{u}$$
$$=\left(\boldsymbol{I}-\frac{1}{12}(\boldsymbol{I}_{n_y}\otimes\boldsymbol{T}_{n_x}+\boldsymbol{T}_{n_y}\otimes\boldsymbol{I}_{n_x})\right)\boldsymbol{f}$$

Using the spectral decompositions of the matrices $\boldsymbol{T}_{n_x}$, $\boldsymbol{T}_{n_y}$ gives

$$\boldsymbol{u}=(\boldsymbol{Q}_{n_y}\otimes\boldsymbol{Q}_{n_x})\left(h_y^2\boldsymbol{I}_{n_y}\otimes\boldsymbol{\Lambda}_{n_x}+h_x^2\boldsymbol{\Lambda}_{n_y}\otimes\boldsymbol{I}_{n_x}-\frac{h_x^2+h_y^2}{12}\boldsymbol{\Lambda}_{n_y}\otimes\boldsymbol{\Lambda}_{n_x}\right)^{-1}$$
$$\times\, h_x^2h_y^2\left(\boldsymbol{I}-\frac{1}{12}(\boldsymbol{I}_{n_y}\otimes\boldsymbol{\Lambda}_{n_x}+\boldsymbol{\Lambda}_{n_y}\otimes\boldsymbol{I}_{n_x})\right)(\boldsymbol{Q}_{n_y}^T\otimes\boldsymbol{Q}_{n_x}^T)\boldsymbol{f}$$

In the middle there is again a diagonal matrix.

The 2D case: numerical example

$$a_x = 0.9, \quad a_y = 1.1,$$

$$u(x,y) = \sin(\pi x/a_x)\sin(3\pi y/a_y).$$

$$f(x,y) = -\nabla^2 u(x,y) = \pi^2\left(\frac{1}{a_x^2}+\frac{9}{a_y^2}\right)\sin\frac{\pi x}{a_x}\cdot\sin\frac{3\pi y}{a_y}.$$

In the Matlab code the approximation error is plotted versus the mesh width $h \sim 1/n$. The norm of the error is computed as

$$\|\boldsymbol{e}\| = \sqrt{\frac{1}{n_x n_y}\sum_{i=1}^{n_x}\sum_{j=1}^{n_y}|u_{i,j}-u(x_i,y_j)|^2}.$$

In this example we have $n = n_x = n_y$.

The 2D case: Matlab demo

`generate_convergence_plot2D`

The 3D case: problem statement

- Cuboid $\Omega = (0, a_x) \times (0, a_y) \times (0, a_z)$
- Poisson equation

$$-\nabla^2 u(x, y, z) = f(x, y, z) \quad \text{in } \Omega, \qquad u = 0 \text{ on } \partial\Omega. \quad (15)$$

- Rectangular mesh: $(n_x+2) \times (n_y+2) \times (n_z+2)$ grid points
- Mesh widths: h_x, h_y, h_z
- 7-point stencil is standard approximation of the Laplacian
- Approximation $u_{ij} \approx u(x_i, y_j)$
- In interior $n_x n_y n_z$ grid points approximate Poisson eq. by

$$\frac{-u_{i-1,j,k} + 2u_{ijk} - u_{i+1,j,k}}{h_x^2} + \frac{-u_{i,j-1,k} + 2u_{ijk} - u_{i,j+1,k}}{h_y^2} + \frac{-u_{i,j,k-1} + 2u_{ijk} - u_{i,j,k+1}}{h_z^2} = f(x_i, y_j, z_k)$$

The 3D case: linear system for the 7-point stencil

Collect values u_{ijk}, $f(x_i, y_j, z_k)$ in vectors $\boldsymbol{u}, \boldsymbol{f} \in \mathbb{R}^{n_x n_y n_z}$, similarly as in the 2D case. Then, the matrix form of above equations is

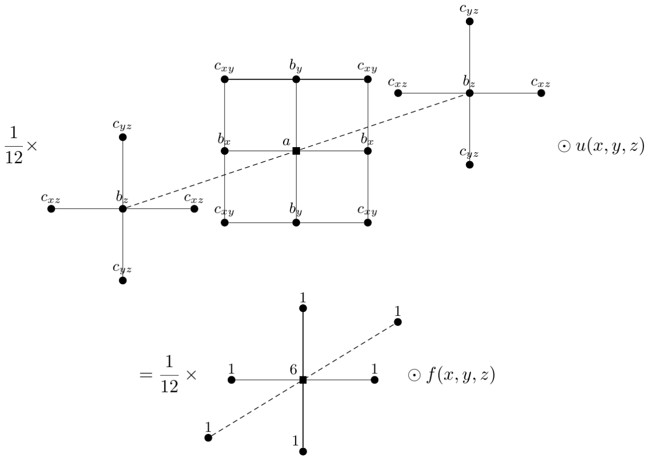
$$\left(\frac{1}{h_x^2}\boldsymbol{I}_{n_z} \otimes \boldsymbol{I}_{n_y} \otimes \boldsymbol{T}_{n_x} + \frac{1}{h_y^2}\boldsymbol{I}_{n_z} \otimes \boldsymbol{T}_{n_y} \otimes \boldsymbol{I}_{n_x} + \frac{1}{h_z^2}\boldsymbol{T}_{n_z} \otimes \boldsymbol{I}_{n_y} \otimes \boldsymbol{I}_{n_x}\right)\boldsymbol{u} = \boldsymbol{f}.$$

Using the spectral decomposition of the $\boldsymbol{T}$'s this becomes

$$(\boldsymbol{Q}_{n_z} \otimes \boldsymbol{Q}_{n_y} \otimes \boldsymbol{Q}_{n_x}) \left(\frac{1}{h_x^2}\boldsymbol{I}_{n_z} \otimes \boldsymbol{I}_{n_y} \otimes \boldsymbol{\Lambda}_{n_x} + \frac{1}{h_y^2}\boldsymbol{I}_{n_z} \otimes \boldsymbol{\Lambda}_{n_y} \otimes \boldsymbol{I}_{n_x} + \frac{1}{h_z^2}\boldsymbol{\Lambda}_{n_z} \otimes \boldsymbol{I}_{n_y} \otimes \boldsymbol{I}_{n_x}\right) (\boldsymbol{Q}_{n_z}^T \otimes \boldsymbol{Q}_{n_y}^T \otimes \boldsymbol{Q}_{n_x}^T)\, \boldsymbol{u} = \boldsymbol{f}.$$

The diagonal matrix in the middle can be precomputed.

The 3D case: linear system for 4th order 19-point stencil



Cf. Spotz&Carey

The 3D case: linear system for 4th order 19-point stencil (cont.)

The matrix form of this stencil is

$$\Big(\frac{1}{h_x^2}\boldsymbol{I}_{n_z}\otimes\boldsymbol{I}_{n_y}\otimes\boldsymbol{T}_{n_x}+\frac{1}{h_y^2}\boldsymbol{I}_{n_z}\otimes\boldsymbol{T}_{n_y}\otimes\boldsymbol{I}_{n_x}+\frac{1}{h_z^2}\boldsymbol{T}_{n_z}\otimes\boldsymbol{I}_{n_y}\otimes\boldsymbol{I}_{n_x}$$
$$-\frac{1}{12}\left(\frac{1}{h_x^2}+\frac{1}{h_y^2}\right)\boldsymbol{I}_{n_z}\otimes\boldsymbol{T}_{n_y}\otimes\boldsymbol{T}_{n_x}-\frac{1}{12}\left(\frac{1}{h_x^2}+\frac{1}{h_z^2}\right)\boldsymbol{T}_{n_z}\otimes\boldsymbol{I}_{n_y}\otimes\boldsymbol{T}_{n}$$
$$-\frac{1}{12}\left(\frac{1}{h_y^2}+\frac{1}{h_z^2}\right)\boldsymbol{T}_{n_z}\otimes\boldsymbol{T}_{n_y}\otimes\boldsymbol{I}_{n_x}\Big)\boldsymbol{u}$$
$$=\left(\boldsymbol{I}-\frac{1}{12}(\boldsymbol{I}_{n_z}\otimes\boldsymbol{I}_{n_y}\otimes\boldsymbol{T}_{n_x}+\boldsymbol{I}_{n_z}\otimes\boldsymbol{T}_{n_y}\otimes\boldsymbol{I}_{n_x}+\boldsymbol{T}_{n_z}\otimes\boldsymbol{I}_{n_y}\otimes\boldsymbol{I}_{n_x})\right)\boldsymbol{f}.$$

Remark: Spotz&Carey also give a $\mathcal{O}(h^6)$ 27-pt stencil for the Laplacian that does not lead to a compact stencil for the Poisson equation, though.

The 3D case: numerical example

$$a_x = 1.1, \quad a_y = 1.0, \quad a_z = 0.9,$$

$$f(x,y,z) = \pi^2 \left(\frac{1}{a_x^2} + \frac{9}{a_y^2} + \frac{25}{a_z^2} \right) \sin(\frac{\pi x}{a_x}) \sin(\frac{3\pi y}{a_y}) \sin(\frac{5\pi z}{a_z}).$$

$$u(x,y,z) = \sin(\pi x/a_x) \sin(3\pi y/a_y) \sin(5\pi z/a_z).$$

in the Matlab code the approximation error is plotted versus the mesh width $h \sim 1/n$. The norm of the error is computed as

$$\|\boldsymbol{e}\| = \sqrt{\frac{1}{n_x n_y n_z} \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} \sum_{k=1}^{n_z} |u_{i,j,k} - u(x_i, y_j, z_k)|^2}.$$

In this example we have $n = n_x = n_y = n_z$.

The 3D case: Matlab demo

```
generate_convergence_plot3D
```

Conclusions

- High-order methods can generate accurate solutions on coarse grids
- Solutions have to be smooth enough
- Matrices get denser as order increases, but we use its spectral decomposition and FFT
- Class of operators is limited, but Laplacian is fine
- In 3D 6th order is possible but the stencil for the right-hand side is not compact anymore
- To use compact FD inside other software, the (input) data has to be accurate