Fast Stencil Computations using Fast Fourier Transforms

Zafar Ahmad, Rezaul Chowdhury, Rathish Das, Pramod Ganapathi, Aaron Gregory, Yimin Zhu

Stencil Computations

- Stencil: pattern to compute value of a cell at a timestep based on values of nearby cells in previous timesteps
- Stencil Computation: applying a stencil to a grid for a number of timesteps

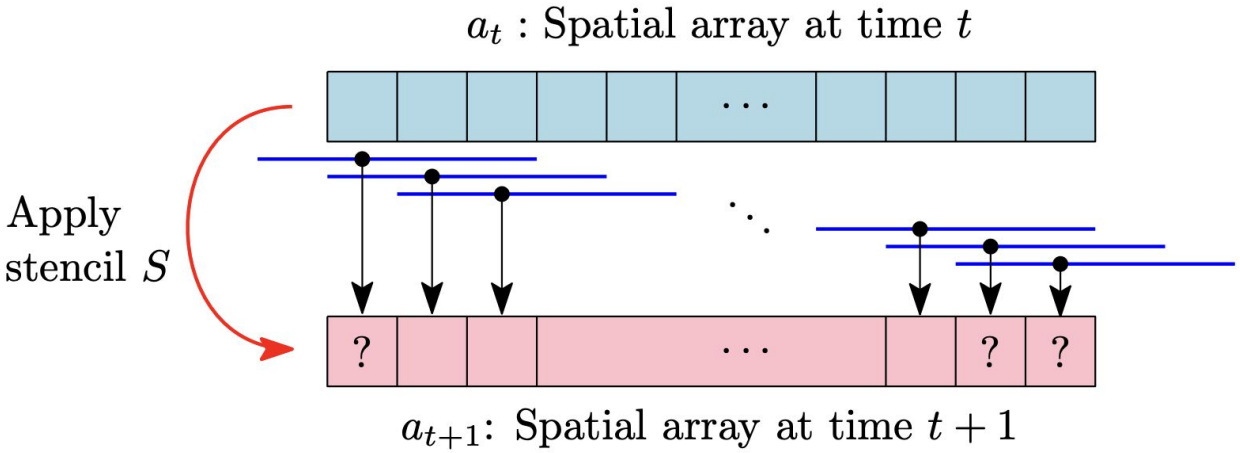
Used for modeling the evolution of physical systems (e.g. fluid dynamics, meteorology, cellular automata)

Problem Specification

Input data: array $a_0[...]$ of size N, linear stencil S such that $a_{t+1} = Sa_t$

Boundary conditions can be periodic or aperiodic

Goal: compute $a_T = S^T a_0$ (i.e. the state after T timesteps)



Related Work

Direct Solvers

- Looping Algorithms: iteratively compute each timestep
- Tiled Looping: make better use of cache locality
- Divide and Conquer: cache-oblivious (e.g. trapezoidal decomposition)

Krylov Subspace Methods

- Compute successively better approximations
- Tradeoff between runtime and accuracy

Limitations of Existing Work

Direct Solvers

- Nonoptimal computational complexity (all O(NT) work)
- No support for implicit stencils

Krylov Subspace Methods

- Manual analysis for convergence
- Specialized for particular problems
- Inexact solutions

Key Contribution

o(NT)-work direct solver based on DFT preconditioning on a Krylov method

- Supports homogeneous linear stencils on vector-valued fields
- Does not support nonlinear or inhomogeneous stencils
    - Can break homogeneity either spatially or temporally

Periodic Boundary Conditions

- 1-D case: $a_t[i] = a_t[i+n]$
- Generally, indices can be computed modulo the size in given dimension
- All values can be computed by direct application of the stencil

Circulant Periodic Stencils

- Linear stencil is a linear mapping, so can write as NxN matrix S
- Periodicity means cyclic permutation preserves relative ordering
- Stencil only affected by relative ordering, so applying cyclic permutation and then stencil results in the same as vice versa → S is circulant
- Only need to store 1st column of S

Circulant S

$$\begin{bmatrix} s_0 & s_{N-1} & \cdots & s_2 & s_1 \\ s_1 & s_0 & s_{N-1} & & s_2 \\ \vdots & & & \ddots & \vdots \\ \vdots & s_1 & s_0 & & \\ & & \ddots & \ddots & \\ s_{N-2} & & \ddots & \ddots & s_{N-1} \\ s_{N-1} & s_{N-2} & \cdots & s_1 & s_0 \end{bmatrix}$$

Properties of Circulant Stencil

- Convolution Theorem states can diagonalize $\mathcal{F}S\mathcal{F}^{-1} = \Lambda$
- Repeated exponentiation of diagonalizable matrix $(\mathcal{F}S\mathcal{F}^{-1})^T = \Lambda^T$
- Let x be the initial data a_0 applied with the Fourier transform, then

$$a_T = S^T a_0 = \mathcal{F}^{-1}\Lambda^T x$$

1-D Explicit Stencil Algorithm

1. FFT - compute FFT of stencil and input data

$$\mathcal{F}S\mathcal{F}^{-1} \qquad \mathcal{F}a_0$$

2. Repeated Squaring - log-time exponentiation of diagonal matrix

$$\mathcal{F}S^T\mathcal{F}^{-1}$$

3. Elementwise Product - apply transformed stencil to input

$$\mathcal{F}a_T = (\mathcal{F}S^T\mathcal{F}^{-1})(\mathcal{F}a_0)$$

4. Inverse FFT - transform result back to spatial domain

$$\mathcal{F}^{-1}\mathcal{F}a_T$$

Work and Span

1. FFT — O(N log N) work, O(log N log log N) span
2. Repeated Squaring — O(N log T) work, O(log N + log T) span
3. Elementwise Product — O(N) work, O(log N) span
4. Inverse FFT — O(N log N) work, O(log N log log N) span

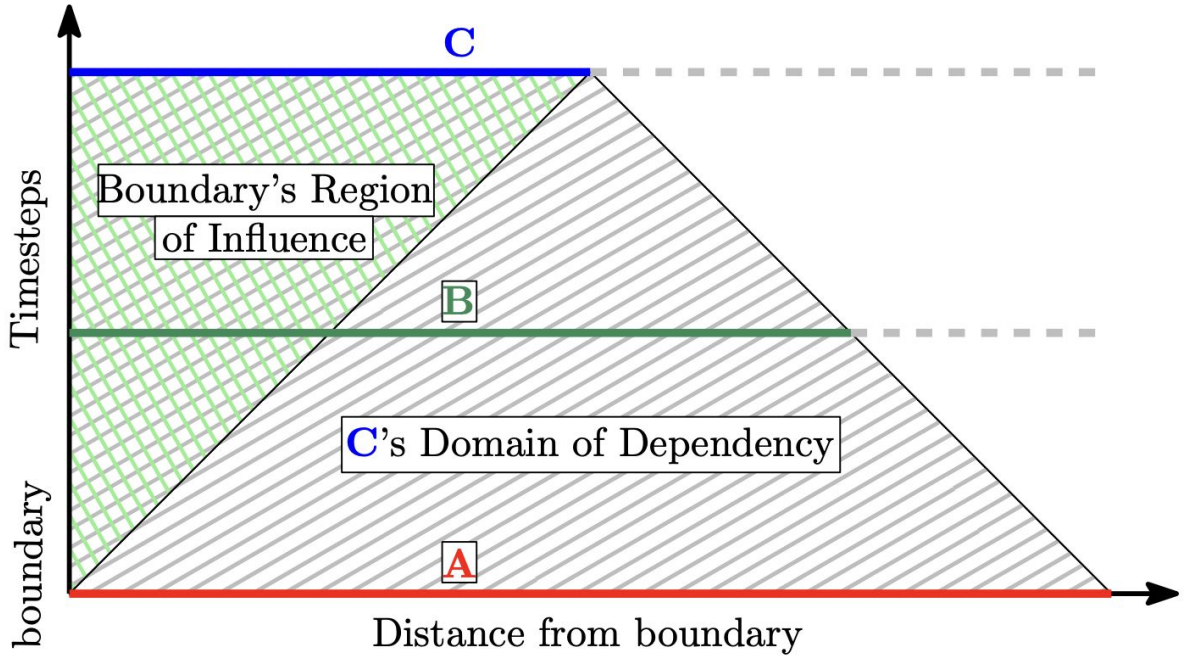
Overall: **O(N log(NT)) work, O(log N log log N + log T) span**

Generalizations

- Multi-Dimensional Stencils
  - Index into stencil and grid data with d-dimensional vector instead of integer
- Implicit Stencils
  - Stencils that depend on field data from timestep being computed
  - Compute pseudoinverse after diagonalizing implicit part of stencil to make explicit
- Vector-Valued Fields
  - Stencil for each pair of indices between input and output vectors
  - Obtain equivalent stencils for longer timesteps by sequential squaring of matrix of stencils

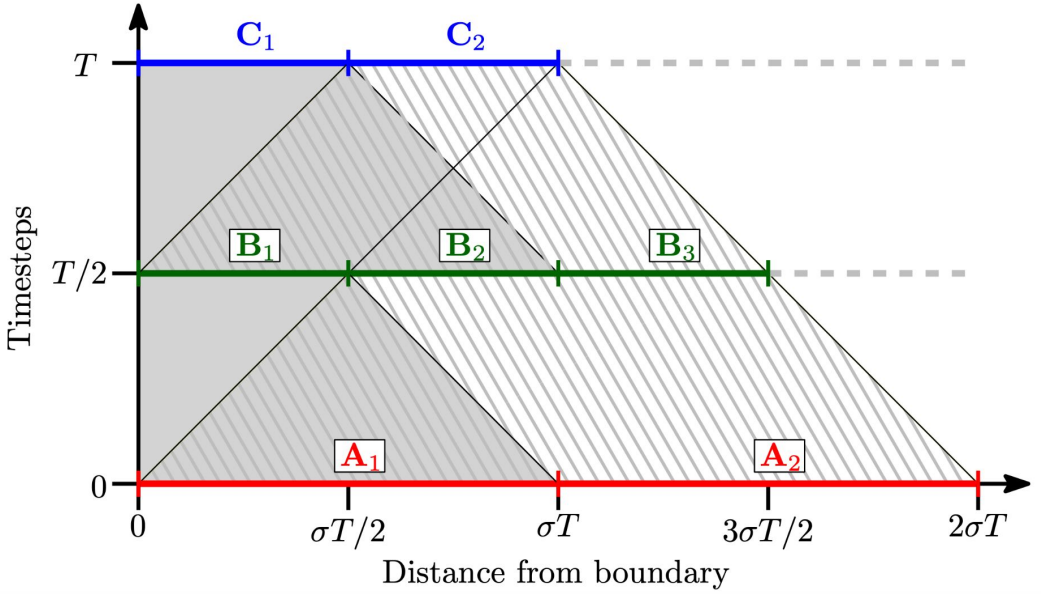
Aperiodic Boundaries

Examples: Dirichlet and Von Neumann



Correcting Boundary Region of Influence

- Recursive trapezoidal decomposition
- Striped regions computed with periodic algorithm
- Dark regions solved with sequential divide and conquer
- Base case uses standard looping algorithm



Work and Span

Work of *Recursive Boundary*

- Recurrence: W(T) = 2W(T/2) + O(bT log bT)
  - Base case: O(b)
- Solving gives *W(T) = O(bT log(bT) log(T))*

Span of *Recursive Boundary*

- S(T) = max{2S(T/2), O(1) log(T) log log(T)} + O(log b)
- Solving gives *S(T) = O(T log b)*

Overall **O(bT log(bT) log(T) + N log(N)) work**
O(T log(b) + log(N) log log(N)) span

*Where b is size of boundary

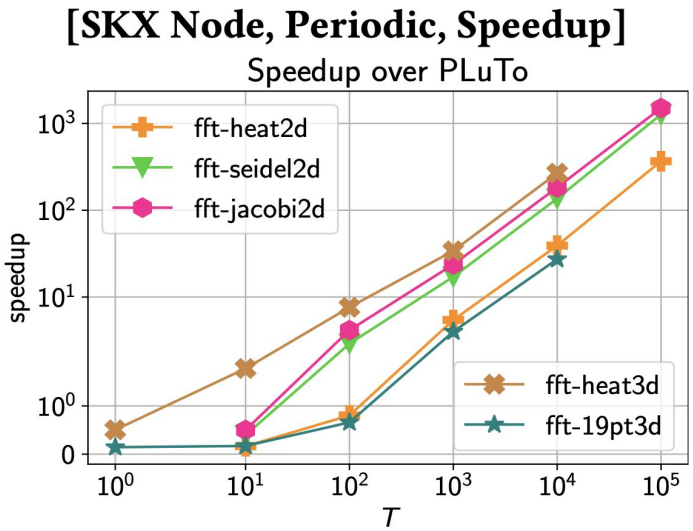
Experimental Setup

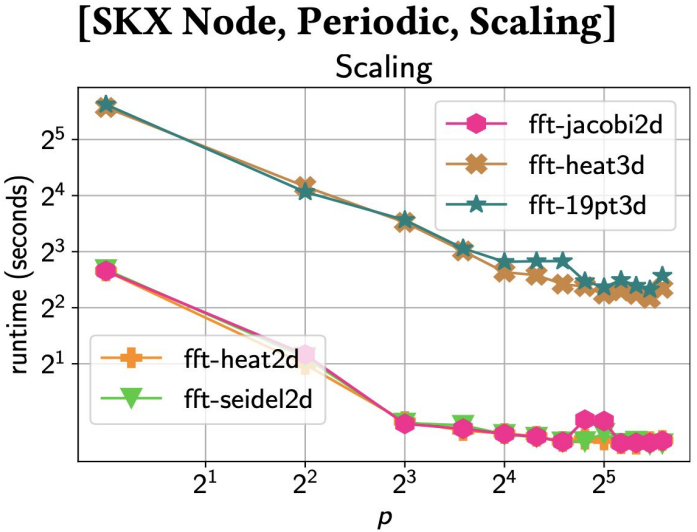
- Run on 68-core KNL and 48-core SKX nodes
- Baseline SoTA tiled looping code generator (PLuTo)
- Benchmark problems in 1, 2, and 3 dimensions with both periodic and aperiodic boundaries
  - N = 1.6M, 8Kx8K, and 800x800x800 for 1, 2, 3 dims. respectively

Benchmark	heat1d	heat2d	seidel2d	jacobi2d	heat3d	19pt3d
Stencil points (α)	3pts	5pts	9pts	25pts	7pts	19pts
Stencil radius (σ)	1	1	1	2	1	2

Periodic Stencil Algorithms

- Theoretical speedup over looping algorithms is O(T / log T)
- In practice, do observe near linear speedup as N is held constant
- Due to memory bounded FFT, do not expect to scale well





Aperiodic Stencil Algorithms

- Their algorithm was at least 8.5, 2.3, and 1.3 times faster on KNL than PLuTo generated code for 1, 2, and 3 dimensions respectively
- Corresponding figures were 5.2, 1.7, and 1.3 times faster on SKX
- Theoretical bounds predict a speedup of O(N^(1/d) / (log(TN^(1-1/d)) log T)) over PLuTo code, so for fixed N, increases in T will lead to worse speedups

Numerical Accuracy

- Compared max relative error to naive looping code
- No significant loss of accuracy

Stencil	Grid size	Timesteps	Our algorithm	Looping code
heat1d	$1,000$	10^6	5.71632×10^{-6}	5.71637×10^{-6}
heat2d	500×500	2.5×10^5	2.73253×10^{-5}	2.73253×10^{-5}
heat3d	$200 \times 200 \times 200$	4×10^4	1.72981×10^{-4}	1.72981×10^{-4}

Future Work

- Extension to nonlinear stencils
- Low-span algorithms for aperiodic stencils
- Approximate algorithms for inhomogeneous stencils

Evaluation

Strengths

- Very significant speedup for periodic boundaries
- Novel approach improving both theoretical bounds and practical performance

Weaknesses

- Aperiodic unable to take full advantage of their periodic approach, needing to compute essentially iteratively for larger T
- Low scalability beyond 32 threads