PCSE Lecture 12

PDEs: Setting the Stage for Parallel Computations

Cyrus Proctor
Victor Eijkhout

April 21, 2015

Partial Differential Equations

Space-constraint phenomena:
BVP (Boundary Value Problem), usually Partial Differential Equations

Time-evolving plus space-constraint phenomena:
IBVP (Initial Value Problem), usually Partial Differential Equations

Boundary Value Problems

Boundary value problems

Consider Poisson's equation in 1D

$$u''(x) = f(x, u, u')$$

for $x \in [a, b]$ where $u(a) = u_a$, $u(b) = u_b$
and, later in 2D:

$$-u_{xx}(\bar{x}) - u_{yy}(\bar{x}) = f(\bar{x})$$

for $x \in \Omega = [0, 1]^2$ with $u(\bar{x}) = u_0$ on $\delta\Omega$.

Approximation of 2nd order derivatives

Taylor series (write h for δx):

$$u(x+h) = u(x) + u'(x)h + u''(x)\frac{h^2}{2!} + u'''(x)\frac{h^3}{3!} + u^{(4)}(x)\frac{h^4}{4!} + u^{(5)}(x)\frac{h^5}{5!} + \cdots$$

and

$$u(x-h) = u(x) - u'(x)h + u''(x)\frac{h^2}{2!} - u'''(x)\frac{h^3}{3!} + u^{(4)}(x)\frac{h^4}{4!} - u^{(5)}(x)\frac{h^5}{5!} + \cdots$$

Add the two equations together:

$$u(x+h) + u(x-h) = 2u(x) + u''(x)h^2 + u^{(4)}(x)\frac{h^4}{12} + \cdots$$

so

$$u''(x) = \frac{u(x+h) - 2u(x) + u(x-h)}{h^2} - u^{(4)}(x)\frac{h^4}{12} + \cdots$$

Numerical scheme:

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

(2nd order PDEs are very common!)

This leads to linear algebra

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

Equally spaced points on $[0,1]$: $x_k = kh$ where $h = 1/n$, then

$$-u_{k-1}+2u_k-u_{k+1}=h^2\,f(x_k,u_k,u_k') \quad \text{for } k=1,\ldots,n-1$$

Written as matrix equation:

$$\begin{pmatrix} 2 & -1 & & \\ -1 & 2 & -1 & \\ & \ddots & \ddots & \ddots \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \end{pmatrix} = \begin{pmatrix} h^2 f_1 + u_0 \\ h^2 f_2 \\ \vdots \end{pmatrix}$$

Matrix properties

- Very sparse, banded (tridiagonal in 1D)
- Symmetric (only because 2nd order problem)
- Sign pattern: positive diagonal, nonpositive off-diagonal (true for many second order methods)
- Positive definite (just like the continuous problem)

Initial Boundary Value Problems

Initial Boundary value problem

Heat conduction in a rod $T(x,t)$ for $x \in [a,b]$, $t > 0$:

$$\frac{\partial}{\partial t} T(x,t) - \alpha \frac{\partial^2}{\partial x^2} T(x,t) = q(x,t)$$

- Initial condition: $T(x,0) = T_0(x)$
- Boundary conditions: $T(a,t) = T_a(t)$, $T(b,t) = T_b(t)$
- Material property: $\alpha > 0$ is thermal diffusivity
- Forcing function: $q(x,t)$ is externally applied heating.

The equation $u''(x) = f$ above is the steady state.

Discretization

Space discretization: $x_0 = a$, $x_n = b$, $x_{j+1} = x_j + \Delta x$
Time discretization: $t_0 = 0$, $t_{k+1} = t_k + \Delta t$
Let T_j^k approximate $T(x_j, t_k)$

Space:

$$\frac{\partial}{\partial t} T(x_j, t) - \alpha \frac{T(x_{j-1}, t) - 2T(x_j, t) + T(x_{j+1}, t)}{\Delta x^2} = q(x_j, t)$$

Explicit time stepping:

$$\frac{T_j^{k+1} - T_j^k}{\Delta t} - \alpha \frac{T_{j-1}^k - 2T_j^k + T_{j+1}^k}{\Delta x^2} = q_j^k$$

Implicit time stepping:

$$\frac{T_j^{k+1} - T_j^k}{\Delta t} - \alpha \frac{T_{j-1}^{k+1} - 2T_j^{k+1} + T_{j+1}^{k+1}}{\Delta x^2} = q_j^{k+1}$$

Computational form: explicit

$$T_j^{k+1} = T_j^k + \frac{\alpha \Delta t}{\Delta x^2}(T_{j-1}^k - 2T_j^k + T_{j+1}^k) + \Delta t q_j^k$$

This has an explicit form (under-tilde used to denote a column vector):

$$\underset{\sim}{T}^{k+1} = \left(I + \frac{\alpha \Delta t}{\Delta x^2} \right) \underset{\sim}{T}^k + \Delta t \underset{\sim}{q}^k$$

$$\underset{\sim}{T}^k = \begin{pmatrix} T_1^k \\ \vdots \\ T_n^k \end{pmatrix}; I = \begin{pmatrix} 1 & 0 & 0 & \\ 0 & 1 & 0 & \\ & \ddots & \ddots & \ddots \end{pmatrix}$$

Computational form: implicit

$$T_j^{k+1} - \frac{\alpha\Delta t}{\Delta x^2}(T_{j-1}^{k+1} - 2T_j^{k+1} + T_{j+1}^{k+1}) = T_j^k + \Delta t q_j^k$$

This has an implicit form:

$$\left(I - \frac{\alpha\Delta t}{\Delta x^2}K\right)\underset{\sim}{T}^{k+1} = \underset{\sim}{T}^k + \Delta t \underset{\sim}{q}^k$$

where

$$K = \begin{pmatrix} 2 & -1 & & \\ -1 & 2 & -1 & \\ & \ddots & \ddots & \ddots \end{pmatrix}$$

Needs to solve a linear system in every time step

Von Neumann Stability Analysis

Stability of explicit scheme

Let $q \equiv 0$; assume $T_j^k = \beta^k e^{i\ell x_j}$; for stability we require $|\beta| < 1$:

$$\begin{aligned}
T_j^{k+1} &= T_j^k + \frac{\alpha\Delta t}{\Delta x^2}(T_{j-1}^k - 2T_j^k + T_{j+1}^k) \\
\Rightarrow \beta^{k+1} e^{i\ell x_j} &= \beta^k e^{i\ell x_j} + \frac{\alpha\Delta t}{\Delta x^2}(\beta^k e^{i\ell x_{j-1}} - 2\beta^k e^{i\ell x_j} + \beta^k e^{i\ell x_{j+1}}) \\
\Rightarrow \beta \equiv \frac{\beta^{k+1}}{\beta^k} &= 1 + 2\frac{\alpha\Delta t}{\Delta x^2}[\frac{1}{2}(e^{i\ell\Delta x} + e^{-\ell\Delta x}) - 1](\text{Euler's Formula}) \\
&= 1 + 2\frac{\alpha\Delta t}{\Delta x^2}(\cos(\ell\Delta x) - 1)
\end{aligned}$$

$$\beta \equiv \frac{\beta^{k+1}}{\beta^k} = 1 + 2\frac{\alpha\Delta t}{\Delta x^2}(\cos(\ell\Delta x) - 1)$$

To get $|\beta| < 1$ means $\beta < 1$ and $-\beta < 1$; i.e.

$$1 + 2\frac{\alpha\Delta t}{\Delta x^2}(\cos(\ell\Delta x) - 1) < 1$$

and

$$-(1 + 2\frac{\alpha\Delta t}{\Delta x^2}(\cos(\ell\Delta x) - 1)) < 1$$

- $2\frac{\alpha\Delta t}{\Delta x^2}(\cos(\ell\Delta x) - 1) < 0$: automatic b/c $sin^2(\frac{\ell\Delta x}{2}) \equiv \frac{(1-cos(\ell\Delta x))}{2}$
- $2\frac{\alpha\Delta t}{\Delta x^2}(\cos(\ell\Delta x) - 1) > -2$: needs $2\frac{\alpha\Delta t}{\Delta x^2} < 1$, that is

$$\Delta t < \frac{\Delta x^2}{2\alpha}$$

Conditionally stable. Restriction on size of time steps as spatial mesh shrinks.

Stability of implicit scheme

$$\begin{aligned}
T_j^{k+1} - \frac{\alpha\Delta t}{\Delta x^2}(T_{j-1}^{k+1} - 2T_j^{k+1} + T_{j+1}^{k+1}) &= T_j^k \\
\Rightarrow \beta^{k+1}e^{i\ell x_j} - \frac{\alpha\Delta t}{\Delta x^2}(\beta^{k+1}e^{i\ell x_{j-1}} - 2\beta^{k+1}e^{i\ell x_j} + \beta^{k+1}e^{i\ell x_{j+1}}) &= \beta^k e^{i\ell x_j}
\end{aligned}$$

$$\begin{aligned}
\Rightarrow \beta^{-1} &= 1 + 2\frac{\alpha\Delta t}{\Delta x^2}(1-\cos(\ell\Delta x)) \\
\beta &= \frac{1}{1+2\frac{\alpha\Delta t}{\Delta x^2}(1-\cos(\ell\Delta x))}
\end{aligned}$$

Noting that $1-\cos(\ell\Delta x) > 0$, the condition $|\beta| < 1$ always satisfied: Unconditionally stable.

Matrix properties for implicit scheme

- Same structure as in BVP
- Need to solve one “BVP” per time step
- 1D would be tridiagonal
- 2D would be 5-banded
- 3D would be 7-banded
- Symmetric (only because 2nd order problem)
- Sign pattern: positive diagonal, nonpositive off-diagonal (true for many second order methods)
- Positive definite (just like the continuous problem)

Linear System Solution Techniques

Two different approaches

Solve $Ax = b$

Direct methods:

- Deterministic
- Exact up to machine precision
- Expensive (in time and space)

Iterative methods:

- Only approximate
- Cheaper in space and (possibly) time
- Convergence not guaranteed

Iterative methods

In a basic form, choose any starting x^0 and repeat

$$x^{k+1} = Bx^k + c$$

until some absolute tolerance

$$\|x^{k+1} - x^k\|_2 < \epsilon$$

or until some relative tolerance

$$\frac{\|x^{k+1} - x^k\|_2}{\|x^k\|} < \epsilon$$

is met.

Example of iterative solution

Example system

$$\begin{pmatrix} 10 & 0 & 1 \\ 1/2 & 7 & 1 \\ 1 & 0 & 6 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 21 \\ 9 \\ 8 \end{pmatrix}$$

with solution $(2, 1, 1)$.

Suppose you know (physics) that solution components are roughly the same size, and observe the dominant size of the diagonal, then

$$\begin{pmatrix} 10 & & \\ & 7 & \\ & & 6 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 21 \\ 9 \\ 8 \end{pmatrix}$$

might be a good approximation: solution $(2.1, 9/7, 8/6)$.

Iterative example

Example system

$$\begin{pmatrix} 10 & 0 & 1 \\ 1/2 & 7 & 1 \\ 1 & 0 & 6 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 21 \\ 9 \\ 8 \end{pmatrix}$$

with solution $(2, 1, 1)$.

Also "easy" to solve:

$$\begin{pmatrix} 10 & & \\ 1/2 & 7 & \\ 1 & 0 & 6 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 21 \\ 9 \\ 8 \end{pmatrix}$$

with solution $(2.1, 7.95/7, 5.9/6)$.

Abstract presentation

- To solve $Ax = b$ by direct solution too expensive
- Multiplying by A is feasible
- Suppose $K \approx A$ and solving $Kx = b$ is possible
- Define $Kx_0 = b$
- Error is then $e_0 = x_0 - x$
- Our original system would be $A(x_0 - e_0) = b$
- Distribute $Ax_0 - Ae_0 = b$
- Then $Ae_0 = Ax_0 - b = r_0$ where r_0 is our residual
- $Ae_0 = r_0$ still too expensive to solve
- Instead, solve $K\tilde{e}_0 = r_0$
- Define $x_1 \equiv x_0 - \tilde{e}_0$
- And $e_1 = x_1 - x$, $r_1 = Ax_1 - b$, $K\tilde{e}_1 = r_1$
- Et cetera

Choice of K

- The closer K is to A, the faster the convergence.
- Diagonal and lower triangular choice mentioned above: let

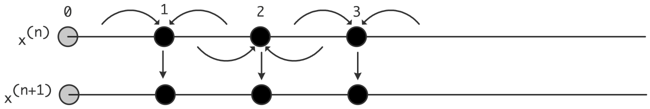
  $$A = D_A + L_A + U_A$$

  be a matrix splitting into diagonal, lower triangular, upper triangular part, then

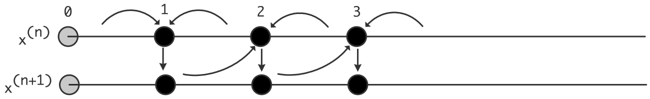
The "classics":

- Jacobi method: $K = D_A$ (diagonal part),
- Gauss-Seidel method: $K = D_A + L_A$ (lower triangle, including diagonal)
- Successive Over Relaxation (SOR) method: $K = \omega D_A + L_A$

Jacobi method:



Gauss-Seidel and/or SOR method:



Choice of K through incomplete LU

- Inspiration from direct methods: let $K = LU \approx A$

Gauss elimination:

```
for k,i,j:
   a[i,j] = a[i,j] - a[i,k] * a[k,j] / a[k,k]
```

Incomplete variant:

```
for k,i,j:
  if a[i,j] not zero:
    a[i,j] = a[i,j] - a[i,k] * a[k,j] / a[k,k]
```

$\Rightarrow$ sparsity of $L + U$ the same as of A (no fill-in!)

- The above cases are Stationary Iterative Methods where B and c of

$$x^{k+1} = Bx^k + c$$

  do not depend on the iteration count
- Older, simpler, easy to implement, but usually not as effective as newer methods
- Many, newer, and most popular solution methods today are the Krylov subspace methods
- Some of the best known Krylov subspace methods are Arnoldi, Lanczos, CG (conjugate gradient), IDR(s) (induced dimension reduction), GMRES (generalized minimum residual), BiCGSTAB (biconjugate gradient stabilized), QMR (quasi minimal residual), TFQMR (transpose-free QMR), and MINRES (minimal residual)

Sparse matrix-vector product

Sparse matrix storage

Matrices above have many zeros: n^2 elements but only $O(n)$ nonzeros. Big waste of space to store this as square array.

Matrix is called 'sparse' if there are enough zeros to make specialized storage feasible.

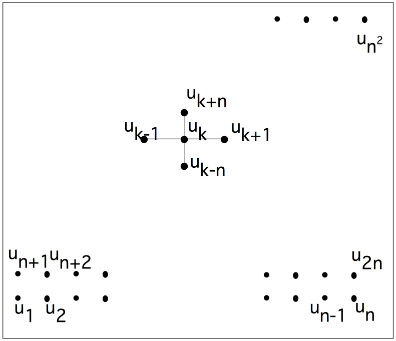
Sparse matrix in 2D case

Sparse matrices so far were tridiagonal: only in 1D case.

Two-dimensional: $-u_{xx} - u_{yy} = f$ on unit square $[0,1]^2$

Difference equation:

$$4u(x,y) - u(x+h,y) - u(x-h,y) - u(x,y+h) - u(x,y-h) = h^2 f(x,y)$$

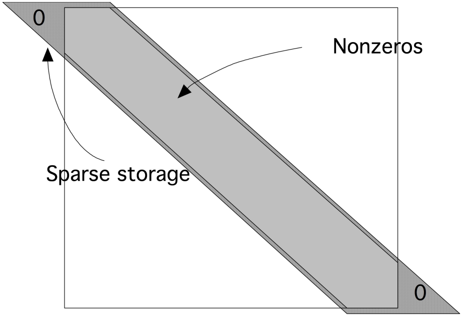


Sparse matrix from 2D equation

$$\left(\begin{array}{ccccc|ccccc|cc}
4 & -1 & & & \emptyset & -1 & & & & \emptyset & & \\
-1 & 4 & 1 & & & & -1 & & & & & \\
 & \ddots & \ddots & \ddots & & & & \ddots & & & & \\
 & & \ddots & \ddots & -1 & & & & \ddots & & & \\
\emptyset & & & -1 & 4 & \emptyset & & & & -1 & & \\
\hline
-1 & & & & \emptyset & 4 & -1 & & & & -1 & \\
 & -1 & & & & -1 & 4 & -1 & & & & -1 \\
 & \uparrow & \ddots & & & \uparrow & \uparrow & \uparrow & & & & \uparrow \\
 & k-n & & & & k-1 & k & k+1 & & -1 & & k+n \\
 & & & & -1 & & & & -1 & 4 & & \\
\hline
 & & & & & \ddots & & & & & \ddots &
\end{array}\right)$$

Storage by diagonals

The most economical storage scheme for a banded matrix (like in our ex. 1/2D BVP) would store the elements consecutively. It's actually more convenient to waste a few storage locations:



For our simple case:
Fortran: 1D: $A(n,-1:1)$; 2D: $A(n,-2:2)$; $A(i,j)\sim A_{i,i-j}$
C: 1D: $A[2][n]$; 2D: $A[4][n]$; $A[i][j]\sim A_{b+i-j,i}$ $b=$dimension

Using the Fortran notation above:
For the matrix-vector product, the main diagonal is used as:

$$y_i \leftarrow y_i + A_{ii} x_i$$

The first superdiagonal is used as:

$$y_i \leftarrow y_i + A_{ii+1} x_{i+1}$$

and the first subdiagonal as:

$$y_i \leftarrow y_i + A_{ii-1} x_{i-1}$$

The whole matrix vector product can be executed in just three vector operations of length n (or $n-1$) instead of n inner products of length 3 (or 2)

Compressed row storage

Consider an example of a sparse matrix:

$$A = \begin{pmatrix} 10 & 0 & 0 & 0 & -2 & 0 \\ 3 & 9 & 0 & 0 & 0 & 3 \\ 0 & 7 & 8 & 7 & 0 & 0 \\ 3 & 0 & 8 & 7 & 5 & 0 \\ 0 & 8 & 0 & 9 & 9 & 13 \\ 0 & 4 & 0 & 0 & 2 & -1 \end{pmatrix} .$$

- No simple band structure
- Or, perhaps, the number of nonzero diagonals becomes too large
- Column indices are stored in int array
- Store pointers to where a column starts

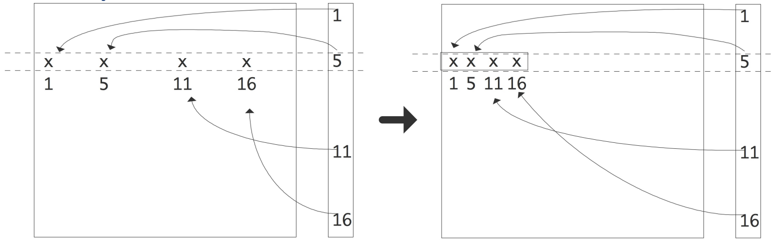
val	10	-2	3	9	3	7	8	7	$3 \cdots 9$	13	4	2	-1
col_ind	0	4	0	1	5	1	2	3	$0 \cdots 4$	5	1	4	5

row_ptr	0	2	5	8	12	16	19

.

Compressed row storage

Pictorially:



Sparse matrix operations

Most common operation: matrix-vector product $y = Ax$

```
for (row=0; row<nrows; row++) {
   s = 0;
   for (icol=ptr[row]; icol<ptr[row+1]; icol++) {
      int col = ind[icol];
      s += a[aptr] * x[col];
      aptr++;
   }
   y[row] = s;
}
```

Operations with changes to the nonzero structure are much harder!

Indirect addressing of `x` gives low spatial and temporal locality.

References

- Victor Eijkhout, “Introduction to High Performance Scientific Computing”
- Victor Eijkhout, “Parallel Computing for Science and Engineering”
- Randall LeVeque “Finite Difference Methods for Ordinary and Partial Differential Equations”
- Victor Ryaben’kii “A theoretical Introduction to Numerical Analysis”