MA5233 Computational Mathematics

Lecture 6: Sparse Matrices

Simon Etter

National University of Singapore

2019/2020

Sparse Matrices

Example

```
julia> n = 100_000
       A = Tridiagonal(
           fill(-1.0,n-1),
           fill( 2.0,n),
           fill(-1.0,n-1)
       )
       b = rand(100_000)
       @time A \ b;
  0.011402 seconds (...)

julia> n = 10_000
       A = rand(n,n)
       b = rand(n)
       @time A \ b;
  8.071135 seconds (...)
```

First matrix is 10x bigger, yet `A\b` is roughly 1000x faster.
How is this possible?

Sparse Matrices

Outline

- LU factorisation of triangular matrices
- Poisson equation in two dimensions
- General sparse matrix formats
- Sparse matrices in Julia
- Eigenvalues and -vectors of two-dimensional Laplacian.

Sparse Matrices

Theorem

LU factorisation of tridiagonal matrix is tridiagonal.

Proof sketch.

$$\begin{pmatrix} x & x & \\ x & x & x \\ & x & x \end{pmatrix} = \begin{pmatrix} 1 & & \\ x & 1 & \\ & & 1 \end{pmatrix} \begin{pmatrix} x & x & \\ & x & x \\ & x & x \end{pmatrix}$$

$$= \begin{pmatrix} 1 & & \\ x & 1 & \\ & x & 1 \end{pmatrix} \begin{pmatrix} x & x & \\ & x & x \\ & & x \end{pmatrix}$$

Corollary

LU factorisation of tridiagonal matrix takes $\mathcal{O}(n)$ FLOP instead of $\mathcal{O}(n^3)$.

Corollary of corollary

One-dimensional Poisson equation can be solved very efficiently!

Sparse Matrices

Discretising the two-dimensional Poisson equation

Functions:

- Introduce mesh $\Omega_n \times \Omega_n$ where

$$\Omega_n := \left\{ x_k := \tfrac{k}{n+1} \mid k = 0, \ldots, n+1 \right\}.$$

- Replace function $f(x, y)$, with vector of point-values $f(x_{k_1}, x_{k_2})$.
- Use *lexicographical ordering* to arrange these point values as vector:

$$f_{k_1 + n\,(k_2 - 1)} := f(x_{k_1}, x_{k_2}).$$

Example for lexicographical ordering

1	5	9	13
2	6	10	14
3	7	11	15
4	8	12	16

Sparse Matrices

Discretising the two-dimensional Poisson equation

Derivatives:

$$\frac{\partial^2 u}{\partial x^2} \quad \longrightarrow \quad (n+1)^2 \left(u_{i+1} - 2u_i + u_{i-1} \right)$$

$$\frac{\partial^2 u}{\partial y^2} \quad \longrightarrow \quad (n+1)^2 \left(u_{i+n} - 2u_i + u_{i-n} \right)$$

Laplacian now becomes

$$\Delta_n\big(i_1 + n\,(i_2 - 1), j_1 + n\,(j_2 - 1)\big) = \ldots$$

$$= (n+1)^2 \begin{cases} -4 & \text{if } |i_1 - j_1| = 0 \text{ and } |i_2 - j_2| = 0, \\ 1 & \text{if } |i_1 - j_1| = 0 \text{ and } |i_2 - j_2| = 1, \\ 1 & \text{if } |i_1 - j_1| = 1 \text{ and } |i_2 - j_2| = 0, \\ 0 & \text{otherwise} \end{cases}$$

Sparse Matrices

Two-dimensional Laplacian matrix

$$\Delta_n \propto \begin{pmatrix} \begin{bmatrix} -4 & 1 & & \\ 1 & \ddots & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & -4 \end{bmatrix} & \begin{bmatrix} 1 & & \\ & \ddots & \\ & & \ddots & \\ & & & 1 \end{bmatrix} & & \\ \begin{bmatrix} 1 & & \\ & \ddots & \\ & & \ddots & \\ & & & 1 \end{bmatrix} & \ddots & \ddots & \\ & \ddots & \ddots & \begin{bmatrix} 1 & & \\ & \ddots & \\ & & \ddots & \\ & & & 1 \end{bmatrix} \\ & & \begin{bmatrix} 1 & & \\ & \ddots & \\ & & \ddots & \\ & & & 1 \end{bmatrix} & \begin{bmatrix} -4 & 1 & & \\ 1 & \ddots & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & -4 \end{bmatrix} \end{pmatrix}$$

Sparse Matrices

Observation

Two-dimensional Laplacian is no longer tridiagonal.

This will complicate both the data structures and algorithms.

Common data structures for sparse matrices

- Coordinate list
- Compressed sparse column (CSC)

Sparse Matrices

Coordinate list format

Three vectors `i,j,v` of length `nnz` (number of nonzeros) such that

```
A = zeros(n,n)
for k = 1:nnz
    A[i[k],j[k]] = v[k]
end
```

Example

$$A = \begin{pmatrix} 0.2 & & \\ 0.6 & & 0.7 \\ & 0.6 & 1.0 \end{pmatrix} \quad \longrightarrow \quad \begin{array}{l} i = (\; 1 \quad 2 \quad 3 \quad 2 \quad 3 \;)^T \\ j = (\; 1 \quad 1 \quad 2 \quad 3 \quad 3 \;)^T \\ v = (\; 0.2 \quad 0.6 \quad 0.6 \quad 0.7 \quad 1.0 \;)^T \end{array}$$

Properties

- Convenient to assemble sparse matrix.
- A bit wasteful since j contains many repeated entries.

Sparse Matrices

Compressed sparse column (CSC) format

Three vectors `p,i,v` with

`length(p) == n+1`, `length(i) == length(v) == nnz`

such that

```
A = zeros(n,n)
for j = 1:n
    for k = p[j]:p[j+1]-1
        A[i[k],j] = v[k]
    end
end
```

CSC is the format most commonly used in practice.

Example

$$A = \begin{pmatrix} 0.2 & & \\ 0.6 & & 0.7 \\ & 0.6 & 1.0 \end{pmatrix} \longrightarrow \begin{array}{l} p = (\ 1 \quad 3 \quad 4 \quad 6\ )^T \\ i = (\ 1 \quad 2 \quad 3 \quad 2 \quad 3\ )^T \\ v = (\ 0.2 \quad 0.6 \quad 0.6 \quad 0.7 \quad 1.0\ )^T \end{array}$$

Sparse Matrices

Sparse matrices in Julia

- Sparse matrix tools are provided by the `SparseArrays` package.
  Type `using SparseArrays` before calling any of the functions listed below.
- Assemble a sparse matrix: `sparse(i,j,v)`
  See section on coordinate lists above for meaning of `i,j,v`.
- Extract `i,j,v` from sparse matrix `A`: `i,j,v = findnz(A)`
- Sparse identity matrix: `sparse(I, (n,n))`
- Sparse matrix of zeros: `spzeros(n,n)`
- Convert to full matrix: `Matrix(A)`

Sparse Matrices

Assembling the 2d Laplacian matrix, the tedious way

See `laplacian_2d_tedious()` in `6_sparse_matrices.jl`.

Assembling the 2d Laplacian matrix, the clever way

2d Laplacian operator is given by $\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$.

Let us imitate this in the discrete case: $\Delta_n = \Delta_n^{(2,1)} + \Delta_n^{(2,2)}$ where

$$\begin{aligned}
\Delta_n^{(2,1)}\big(i_1 + n\,(i_2 - 1), j_1 + n\,(j_2 - 1)\big) &= \dots \\
&= (n+1)^2 \begin{cases} -2 & \text{if } |i_1 - j_1| = 0 \text{ and } |i_2 - j_2| = 0, \\ 1 & \text{if } |i_1 - j_1| = 1 \text{ and } |i_2 - j_2| = 0, \\ 0 & \text{otherwise,} \end{cases} \\
\Delta_n^{(2,2)}\big(i_1 + n\,(i_2 - 1), j_1 + n\,(j_2 - 1)\big) &= \dots \\
&= (n+1)^2 \begin{cases} -2 & \text{if } |i_1 - j_1| = 0 \text{ and } |i_2 - j_2| = 0, \\ 1 & \text{if } |i_1 - j_1| = 0 \text{ and } |i_2 - j_2| = 1, \\ 0 & \text{otherwise.} \end{cases}
\end{aligned}$$

Sparse Matrices

$$\Delta_n^{(2,1)} \propto \begin{pmatrix} \begin{bmatrix} -2 & 1 & & \\ 1 & \ddots & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & -2 \end{bmatrix} & & \\ & \ddots & \\ & & \begin{bmatrix} -2 & 1 & & \\ 1 & \ddots & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & -2 \end{bmatrix} \end{pmatrix}$$

Sparse Matrices

$$\Delta_n^{(2,2)} \propto \begin{pmatrix} \begin{bmatrix} -2 & & \\ & \ddots & \\ & & -2 \end{bmatrix} & \begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{bmatrix} & & \\ \begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{bmatrix} & \ddots & \ddots & \\ & \ddots & \ddots & \begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{bmatrix} \\ & & \begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{bmatrix} & \begin{bmatrix} -2 & & \\ & \ddots & \\ & & -2 \end{bmatrix} \end{pmatrix}$$

Sparse Matrices

Kronecker product of matrices

$$A \otimes B := \begin{pmatrix} A[1,1]\,B & \cdots & A[1,n]\,B \\ \vdots & \ddots & \vdots \\ A[n,1]\,B & \cdots & A[n,n]\,B \end{pmatrix}$$

2d Laplacian $\Delta_n^{(2)}$ can be expressed in terms of 1d $\Delta_n^{(1)}$ as

$$\Delta_n^{(2)} = \Delta_n^{(1)} \otimes I + I \otimes \Delta_n^{(1)}.$$

See `laplacian_2d_clever()` in `6_sparse_matrices.jl`
for how to do this in Julia.

General rule

If A, B are "one-dimensional" operators, then $A \otimes B$ is the "two-dimensional" operator which applies A in one dimension and B in the other dimension.

Be careful about which operator applies to which dimension.

Sparse Matrices

Kronecker product of vectors

$$a \otimes b := \begin{pmatrix} a[1]\, b \\ \vdots \\ a[n]\, b \end{pmatrix}$$

Theorem

$$(A \otimes B)\,(a \otimes b) = (Aa) \otimes (Bb)$$

Proof. Straightforward but tedious computations.

Sparse Matrices

Eigenvalues and -vectors of 2d Laplacian

Assume λ_ℓ, u_ℓ are eigenpairs of $\Delta_n^{(1)}$.
Then, eigenpairs of $\Delta_n^{(2)}$ are given by

$$\lambda_{\ell_1,\ell_2} := \lambda_{\ell_1} + \lambda_{\ell_2}, \qquad u_{\ell_1,\ell_2} := u_{\ell_1} \otimes u_{\ell_2}.$$

Proof.

$$\begin{aligned}
\Delta_n^{(2)} u_{\ell_1,\ell_2} &= \left(\Delta_n^{(1)} \otimes I + I \otimes \Delta_n^{(1)}\right)\left(u_{\ell_1} \otimes u_{\ell_2}\right) \\
&= \left(\Delta_n^{(1)} u_{\ell_1}\right) \otimes u_{\ell_2} + u_{\ell_1} \otimes \left(\Delta_n^{(1)} u_{\ell_2}\right) \\
&= \lambda_{\ell_1}\, u_{\ell_1} \otimes u_{\ell_2} + \lambda_{\ell_2}\, u_{\ell_1} \otimes u_{\ell_2} \\
&= \left(\lambda_{\ell_1} + \lambda_{\ell_2}\right) u_{\ell_1} \otimes u_{\ell_2} \\
&= \left(\lambda_{\ell_1} + \lambda_{\ell_2}\right) u_{\ell_1,\ell_2}.
\end{aligned}$$