

## Lecture 12: Nonlinear PDEs in 2d and sparsity

COMP5930M Scientific Computation

# Today

Model problem

Numerical solution

Approximation in space

Nonlinear system

Sparse matrix storage

Summary

Next

## 2d nonlinear diffusion

Find  $u(x, y, t)$  satisfying

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x} \left( c(u) \frac{\partial u}{\partial x} \right) + \frac{\partial}{\partial y} \left( c(u) \frac{\partial u}{\partial y} \right) + g(u, x, y)$$

- ▶ The PDE is defined on spatial region  $(x, y) \in \Omega \subset \mathbb{R}^2$
- ▶ Initial condition  $u(x, y, 0)$  needs to be given
- ▶ The solution  $u(x, y, t)$  is known on the boundary  $\partial\Omega$
- ▶  $g(u, x, y)$  is **the source function** that can depend on  $u$  but not its derivatives

## Application in image processing

Let  $u_0(x, y)$  denote the intensity of a noisy grayscale image at pixel  $(x, y)$

**Perona-Malik equation:** find  $u(x, y, t)$  s.t.

$$\begin{aligned}\frac{\partial u}{\partial t} &= \frac{\partial}{\partial x} \left( c(u) \frac{\partial u}{\partial x} \right) + \frac{\partial}{\partial y} \left( c(u) \frac{\partial u}{\partial y} \right), & t > 0 \\ u(x, y, 0) &= u_0(x, y), & t = 0.\end{aligned}$$

where  $c(u) = \frac{1}{1 + \left( \frac{\|\nabla u\|}{K} \right)^2}$  is an anisotropic diffusion term

Removes noise from images without smearing boundaries:

[https://www.youtube.com/watch?v=J5mZD40V\\_VU](https://www.youtube.com/watch?v=J5mZD40V_VU)

## Numerical approximation

We consider only **spatial discretisation** here

- ▶ Approximate in space
  - ▶ Fully discrete nonlinear system
- ▶ Solution algorithm
  - ▶ Numerical solution with Newton's Method

## Spatial approximation

- ▶ Assume our domain  $\Omega$  is rectangular
- ▶ Define an  $n \times m$  uniform grid with spacing

$$\Delta x = \frac{X_2 - X_1}{n - 1}, \quad \Delta y = \frac{Y_2 - Y_1}{m - 1}$$

- ▶ A point (node)  $p_{ij}$  has coordinates  $(x_i, y_j)$  where

$$x_i = X_1 + (i - 1)\Delta x, \quad y_j = Y_1 + (j - 1)\Delta y$$

## Solution data mapping

- ▶ Our discrete problem has  $(n - 2)(m - 2) = N$  unknowns (assuming boundary data is known)
- ▶ ie. our nonlinear system has  **$N$  equations**
- ▶ Our solution algorithm stores a **vector  $\mathbf{U}$**  but our discrete data is a **matrix  $u_{ij}$**
- ▶ We require a unique mapping between these two representations

## Numbering convention

- ▶ We can define a row-based numbering system as

$$U_k \equiv u_{ij}, \quad k = (j-2)(n-2) + (i-1) \\ i = 2, \dots, n-1, \quad j = 2, \dots, m-1$$

- ▶ Could alternatively use column-based numbering



## A compact FDM approximation

$$\begin{aligned}
 \frac{\partial}{\partial x} \left( c(u) \frac{\partial u}{\partial x} \right) &\approx \frac{(c(u) \frac{\partial u}{\partial x})_{i+\frac{1}{2}} - (c(u) \frac{\partial u}{\partial x})_{i-\frac{1}{2}}}{\Delta x} \\
 &\approx \frac{c(u_{i+\frac{1}{2}}) \left( \frac{u_{i+1} - u_i}{\Delta x} \right) - c(u_{i-\frac{1}{2}}) \left( \frac{u_i - u_{i-1}}{\Delta x} \right)}{\Delta x} \\
 &= \frac{c(u_{i+\frac{1}{2}}) (u_{i+1} - u_i) - c(u_{i-\frac{1}{2}}) (u_i - u_{i-1})}{\Delta x^2}
 \end{aligned}$$

- Requires only  $u_{i-1}, u_i, u_{i+1}$
- Reduces to the second order derivative for constant  $c(u)$

## Computing $c(u_{i+\frac{1}{2}})$

Nonlinear  $c(u)$  leads to two obvious alternatives

$$c(u_{i+\frac{1}{2}}) = \frac{c(u_i) + c(u_{i+1})}{2}$$

$$c(u_{i+\frac{1}{2}}) = c\left(\frac{u_i + u_{i+1}}{2}\right)$$

- ▶ Formally of the same accuracy
- ▶ Will produce different nonlinear equations and Jacobian matrix

## Semi-discrete 2d system

$$\begin{aligned}
 \dot{u}_i = & \frac{c(u_{i+\frac{1}{2}j})(u_{i+1j} - u_{ij}) - c(u_{i-\frac{1}{2}j})(u_{ij} - u_{i-1j})}{\Delta x^2} \\
 & + \frac{c(u_{ij+\frac{1}{2}})(u_{ij+1} - u_{ij}) - c(u_{ij-\frac{1}{2}})(u_{ij} - u_{ij-1})}{\Delta y^2} \\
 & + g(u_{ij}, x_i, y_j)
 \end{aligned}$$

- Sparse
- Only 5 values required for each equation
- But not pentadiagonal

## Sparse storage schemes

- ▶ We require **compact storage schemes** for large sparse matrices
  - ▶ Only store non-zero values
- ▶ Our algorithms will have to work with data stored in this format
- ▶ Common alternatives:
  - ▶ Coordinate format
  - ▶ Row-major or column-major format

## Coordinate format

- ▶ For every non-zero item store indices  $i, j$  and value  $a_{ij}$
- ▶ Requires 3 vectors for storage of size  $nz$
- ▶ No ordering implied

## Row-major format

- ▶ For every row  $i$  of the matrix store:
  - ▶ The column index  $j$  and value  $a_{ij}$
  - ▶ The location  $l_i$  in that list of the start of each row
- ▶ Requires 2 vectors of size  $nz$  and 1 of size  $N$
- ▶ Ordering is implied (at the row level)

## Example of sparse row-major format

Sparse Matrix

10	0	0	0	-2
3	9	0	0	0
0	7	8	7	0
3	0	8	7	5
0	8	0	9	13

Row pointer array

0	2	4	7	11	14
---	---	---	---	----	----

Column indices array

0	4	0	1	1	2	3	0	2	3	4	1	3	4
---	---	---	---	---	---	---	---	---	---	---	---	---	---

Values array

10	-2	3	9	7	8	7	3	8	7	5	8	9	13
----	----	---	---	---	---	---	---	---	---	---	---	---	----

A sparse matrix and its corresponding CSR row pointer, column indices and values arrays

## Column-major format (MATLAB)

- ▶ For every column  $j$  of the matrix store:
  - ▶ The row index  $i$  and value  $a_{ij}$
  - ▶ The location  $l_j$  in that list of the start of each column
- ▶ Requires 2 vectors of size  $nz$  and 1 of size  $N$
- ▶ Ordering is implied (at the column level)



## Unrolling matrices into vectors in MATLAB

```
>> A=[1 2 3; 4 5 6; 7 8 9]
```

```
A =
```

1	2	3
4	5	6
7	8	9

```
>> A(:)'
```

```
ans =
```

1	4	7	2	5	8	3	6	9
---	---	---	---	---	---	---	---	---

## Summary

- ▶ Discretisation of 2d problems leads to sparse Jacobian...
- ▶ ...but bandwidth in general larger than tridiagonal
- ▶ Sparse matrices can be stored efficiently using  $2 \times nnz$  vectors (row- or column-major format)
- ▶ Problem is how to find out the sparsity pattern of the Jacobian (more details in tutorial)