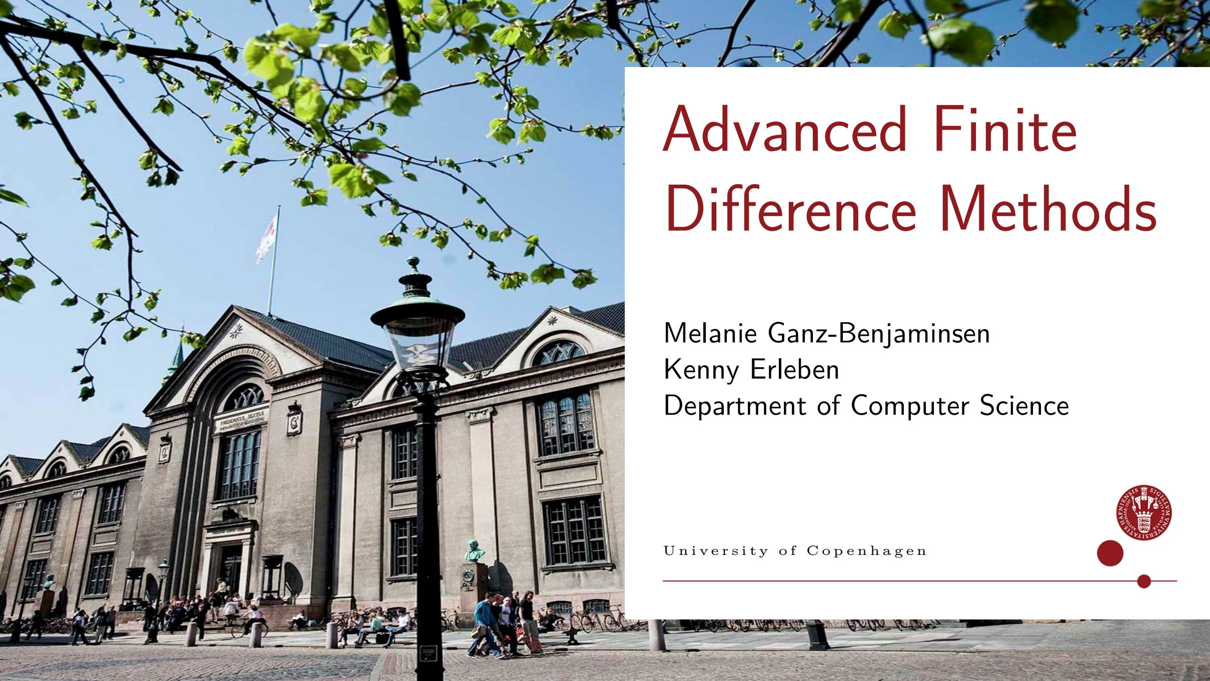

Advanced Finite Difference Methods

Melanie Ganz-Benjaminsen
Kenny Erleben
Department of Computer Science

University of Copenhagen

Advanced Topics

In the past, we have created a nice cookbook of “rules” and techniques we can apply when discretizing partial differential equations using finite difference methods.

We will now embark on more deep aspects and more elaborate examples. Central to these slides is a larger more detailed example of creating a “simulator” for computing the mean curvature flow.

Mean Curvature Flow

Mean curvature flow is given by the PDE

$$\frac{\partial \phi(\mathbf{x}, t)}{\partial t} = -\nabla_{\mathbf{x}} \cdot \frac{\nabla_{\mathbf{x}} \phi(\mathbf{x}, t)}{\| \nabla_{\mathbf{x}} \phi(\mathbf{x}, t) \|}$$

where $\phi(\mathbf{x}, t) : \mathbb{R}^n \times \mathbb{R}_+ \mapsto \mathbb{R}$ is a level set function and $\nabla_{\mathbf{x}} = \left(\frac{\partial}{\partial \mathbf{x}}\right)^T$. As shorthand

$$\frac{\partial \phi}{\partial t} = \underbrace{-2\nabla \cdot \frac{\nabla \phi}{\| \nabla \phi \|}}_{\equiv -2\kappa}$$

ϕ is usually initialized as a signed distance field of some shape. Hence negative values are inside and positive values are outside.

What is Mean Curvature Flow?

- What is curvature?
- PDE is penalizing bending so it tries to locally straighten out any bending

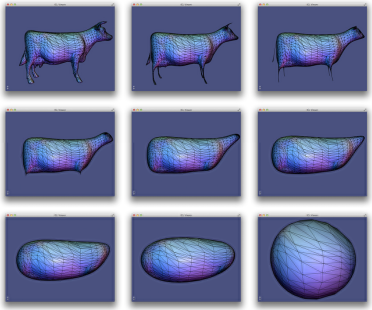

Figure: Image from `https://libigl.github.io/`

What is Mean Curvature Flow?

- Connection to signed distance fields
- What is curvature radius?

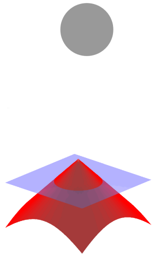

Figure: Image by Oleg Alexandrov, Public domain, via Wikimedia Commons

Our Strategy for Applying FDM

- Our objective now is to rewrite the term κ into a convenient form where we can apply rules from our FDM cook-book to deal with all spatial derivatives.
- Once we have obtained a spatial discretization we will consider how to apply FDM for the temporal derivatives.

Temporal Discretization

- Using 1st order forward difference in the time we have
$$\frac{\partial \phi}{\partial t} \approx \frac{\phi^{t+1} - \phi^{t}}{\Delta t}$$
Observe we must have $\Delta t \ll 1$.
- We substitute this into our governing equation,
$$\frac{\partial \phi}{\partial t} = \underbrace{-2\nabla \cdot \frac{\nabla \phi^{t}}{\| \nabla \phi^{t} \|}}_{\equiv -2\kappa^{t}} = 2\kappa^{t}$$
- The final update scheme becomes
$$\phi^{t+1} = \phi^{t} + \Delta t \, \kappa^{t}$$
We hide the factor 2 in $\Delta t \leftarrow 2\,\Delta t$.
- Recall that superscript denotes the time at when a term is evaluated.

Rewriting the Curvature Term

Let us return to the spatial “mean curvature” term,

$$\kappa = \nabla \cdot \frac{\nabla \phi}{\| \nabla \phi \|}$$

We will rewrite this using $\nabla \cdot (a\mathbf{v}) = \nabla a \cdot \mathbf{v} + a \nabla \cdot \mathbf{v}$ where $a \in \mathbb{R}$ and $\mathbf{v} \in \mathbb{R}^n$

$$\kappa = \nabla \phi \cdot \nabla \left(\frac{1}{\| \nabla \phi \|} \right) + \frac{1}{\| \nabla \phi \|} \nabla \cdot \nabla \phi$$

Using $\nabla \cdot \nabla \phi = \nabla^2 \phi$ and $\| \nabla \phi \| = \sqrt{\nabla \phi^T \nabla \phi}$

$$\kappa = \nabla \phi \cdot \nabla \left(\left(\nabla \phi^T \nabla \phi \right)^{-\frac{1}{2}} \right) + \frac{\nabla^2 \phi}{\| \nabla \phi \|}$$

Using the Chain Rule

So

$$\kappa = \nabla\phi \cdot \nabla \left(\left(\nabla\phi^T \nabla\phi\right)^{-\frac{1}{2}} \right) + \frac{\nabla^2\phi}{\parallel \nabla\phi \parallel}$$

Using the chain rule

$$\nabla \left(\left(\nabla\phi^T \nabla\phi\right)^{-\frac{1}{2}} \right) = -\frac{1}{2} \left(\nabla\phi^T \nabla\phi\right)^{-\frac{3}{2}} 2 \left(\nabla \left(\nabla\phi\right)\right) \nabla\phi$$

Introducing the Hessian symbol $\mathbf{H} = \left(\nabla \left(\nabla\phi\right)\right)$

$$\kappa = \frac{\mathrm{tr}\left(\mathbf{H}\right)}{\parallel \nabla\phi \parallel} - \frac{\nabla\phi^T \mathbf{H} \nabla\phi}{\parallel \nabla\phi \parallel^3}$$

where $\nabla^2\phi = \mathrm{tr}\left(\mathbf{H}\right)$

Common Denominator

Now

$$\kappa = \frac{\mathrm{tr}\left(\mathbf{H}\right)}{\parallel \nabla\phi \parallel} - \frac{\nabla\phi^{T}\mathbf{H}\nabla\phi}{\parallel \nabla\phi \parallel^{3}}$$

Becomes

$$\kappa = \frac{\nabla\phi^{T}\nabla\phi\mathrm{tr}\left(\mathbf{H}\right) - \nabla\phi^{T}\mathbf{H}\nabla\phi}{\parallel \nabla\phi \parallel^{3}}$$

Spatial Discretization in 2D

Given a 2D grid let ϕ_{ij} be the value of ϕ at node (i,j). Using central difference approximations

$$\nabla\phi_{i,j} \approx \begin{bmatrix} D_x\phi_{ij} \\ D_y\phi_{ij} \end{bmatrix} = \begin{bmatrix} \frac{\phi_{i+1,j}-\phi_{i-1,j}}{2\Delta x} \\ \frac{\phi_{i,j+1}-\phi_{i,j-1}}{2\Delta y} \end{bmatrix}$$

Trivially

$$\| \nabla\phi_{ij} \| \approx \underbrace{\sqrt{(D_x\phi_{ij})^2 + (D_y\phi_{ij})^2}}_{\equiv g_{ij}}$$

More 2D Discretization

The Hessian

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 \phi}{\partial x^2} & \frac{\partial^2 \phi}{\partial xy} \\ \frac{\partial^2 \phi}{\partial xy} & \frac{\partial^2 \phi}{\partial y^2} \end{bmatrix} \approx \begin{bmatrix} D_{xx}\phi & D_{xy}\phi \\ D_{xy}\phi & D_{yy}\phi \end{bmatrix}$$

The 2nd-order central difference approximations

$$D_{xx}\phi_{ij} = \frac{\phi_{i+1,j} - 2\phi_{i,j} + \phi_{i-1,j}}{\Delta x^2}$$

$$D_{yy}\phi_{ij} = \frac{\phi_{i,j+1} - 2\phi_{i,j} + \phi_{i,j-1}}{\Delta y^2}$$

$$D_{xy}\phi_{ij} = \frac{\phi_{i+1,j+1} - \phi_{i+1,j-1} - \phi_{i-1,j+1} + \phi_{i-1,j-1}}{4\Delta x \Delta y}$$

Putting it Together

We make the approximation of κ at node (i,j) by substitution of our FDM discretizations into

$$k \approx \kappa = \frac{\nabla\phi^T \nabla\phi \,\mathrm{tr}\left(\mathbf{H}\right) - \nabla\phi^T \,\mathbf{H}\, \nabla\phi}{\|\ \nabla\phi\ \|^3}$$

and after some cleaning up we have the approximation value k_{ij},

$$k_{ij} = \frac{\left(D_x\phi_{ij}\right)^2 D_{yy}\phi_{ij} + \left(D_y\phi_{ij}\right)^2 D_{xx}\phi_{ij} - 2D_{xy}\phi_{ij}D_x\phi_{ij}D_y\phi_{ij}}{g_{ij}^3}$$

Numerical Considerations

Looking at k_{ij} we observe

- k_{ij} could become unrealistic large or small if $g_{ij} \to 0$
- If $g_{ij} \to 0$ then we may have a division by zero
- If ϕ is a signed distance field then $g_{ij} \approx 1$ almost everywhere (at least sufficiently close to nodes where $\phi_{ij} = 0$)

Numerical Remedies (1/2)

- To avoid division by zero we redefine our discretization as

$$k_{ij} = \frac{(\cdots)}{g_{ij}^3 + \varepsilon}$$

  where the user specified constant $\varepsilon \ll 1$ is a threshold value of the same order as the numerical precision
- Alternatively if ϕ is a signed distance field we modify g_{ij}

$$g_{ij} \leftarrow \begin{cases} g_{ij} & \text{if } g_{ij} > \frac{1}{2} \\ 1 & \text{otherwise} \end{cases}$$

  In case k_{ij} at node (i,j) where $g_{ij} \to 0$ then find the closest node (k,m) of (i,j) where $g_{km} \to 1$ (is sufficiently close to 1) and replace

Numerical Remedies (2/2)

- The maximum absolute value of the mean curvature that can be represented on a regular grid is given by the conservative bound

$$\kappa_{\text{max}} = \frac{1}{\max\left(\Delta x, \Delta y\right)}$$

Hence it makes sense to clamp the discrete approximation

$$k_{ij} \leftarrow \max\left(-\kappa_{\text{max}}\,, \min\left(k_{ij}, \kappa_{\text{max}}\right)\right)$$

This avoids the problem of k_{ij} going to infinity

Recall The Temporal Discretization

Using 1st order forward difference we have

$$\frac{\partial \phi_{ij}}{\partial t} \approx \frac{\phi_{ij}^{t+1} - \phi_{ij}^{t}}{\Delta t}$$

The final update scheme becomes

$$\phi_{ij}^{t+1} = \phi_{ij}^{t} + \Delta t \, k_{ij}^{t}$$

where we need $\Delta t \ll \frac{1}{2}$ for our finite difference approximation of the time derivative to be justified. Exactly how should we select the value of Δt?

Numerics can be too fast

- When we use the final scheme one may experience spurious artifacts like noise or disintegration, for example, artifacts appearing just before a blow-up of ϕ. Such artifacts are related to instability. This means that the speed of how fast ϕ_{ij} is changing (that is the absolute value of k_{ij}) is so large that with the time-step Δt we see that ϕ_{ij} would have changed (traveled longer) more than the minimum grid size, h.
- Hence, the idea is to prevent the update scheme from using a speed of ϕ that is larger than what the grid can represent, ie $\frac{h}{\Delta t}$. One such scheme is known as a Courant—Friedrichs—Lewy (CFL) condition.

That is we would require,

$$|k_{ij}| < \frac{h}{\Delta t}, \quad \forall i, j$$

This is a necessary condition but not a sufficient one.

The CFL condition

If we define $k_{\text{max}} = \max_{i,j}\{|k_{ij}|\}$ then we can write up a global CFL condition as

$$k_{\text{max}} < \frac{h}{\Delta t}, \quad \forall i, j$$

Let us define the CFL constant $0 < C \ll 1$ then we can write

$$k_{\text{max}} = C \frac{h}{\Delta t}$$

In a 2D world, we have the grid spacings: Δx and Δy. Hence, we define $h = \min(\Delta x, \Delta y)$,

$$\Delta t = C \, \frac{h}{k_{\text{max}}}$$

The value $C = \frac{1}{2}$ is typical but often C is tuned using eye-balling.

Matrix-Free Methods for Iterative Solvers (1/8)

As seen previously a finite difference approximation leads to a coefficient equation like in this simple 1D example

$$c_{i-1}\,x_{i-1} + c_i\,x_i + c_{i+1}\,x_{i+1} = b_i$$

which can be rewritten into a stencil update rule

$$x_i \leftarrow \frac{1}{c_i}\,b_i - \frac{c_{i-1}}{c_i}\,x_{i-1} - \frac{c_{i+1}}{c_i}\,x_{i+1}$$

In many cases $c_{i-1} = c_{i+1}$ and $c_i \neq c_{i-1}$ are constants independent of i so the stencil update rule can be implemented efficiently using a compact representation

$$x_i \leftarrow \beta\,b_i - \gamma\,x_{i-1} - \gamma\,x_{i+1}$$

where $\beta = \frac{1}{c_i}$ and $\gamma = \frac{c_{i-1}}{c_i}$.

Matrix-Free Methods for Iterative Solvers (2/8)

Now we can outline the stencil update sub-routine as follows

```
function x = update_stencil_gauss_seidel(beta,gamma,b,x)
  % Apply boundary conditions to x...

   for all i in sequence do
     x(i) = beta*b(i)  - gamma*x(i-1) - gamma*x(i+1)
   end
end
```

Matrix-Free Methods for Iterative Solvers (3/8)

Or if we update all nodes in parallel

```
function y = update_stencil_Jacobi(beta,gamma,b,x)
  % Apply boundary conditions to x....

   for all i in parallel do
     y(i) = beta*b(i)  - gamma*x(i-1) - gamma*x(i+1)
   end
end
```

Matrix-Free Methods for Iterative Solvers (4/8)

One would apply the Jacobi style sub-routine like this

```
function x = Jacobi_solver(beta,gamma,b,x)
  while not converged
     y = update_stencil_Jacobi(beta,gamma,b,x)
     x = y;
  end
end
```

Matrix-Free Methods for Iterative Solvers (5/8)

A Gauss-Seidel version would be more like

```
function x = Gauss_Seidel_solver(beta,gamma,b,x)
  while not converged
     x = update_stencil_Gauss_Seidel(beta,gamma,b,x)
  end
end
```

Matrix-Free Methods for Iterative Solvers (6/8)

Observe if we make the specific choices

$$
\begin{aligned}
b &= x \\
\beta &= -c_i, \\
\gamma &= c_{i-1} \text{ or } c_{i+1}
\end{aligned}
$$

and we compute `y = update_stencil_Jacobi( -c(i) , c(i-1), x, x)` then this evaluates to

$$\mathbf{y} = \mathbf{A}\,\mathbf{x}$$

Furthermore, the Jacobi scheme is a naive parallel so it will be efficient.

Matrix-Free Methods for Iterative Solvers (7/8)

Let us define the linear operator

```
A(x) = @update_stencil_Jacobi( -c(i),c(i-1),x,x)
```

This opens up the possibility of using iterative solvers that have $\mathbf{A}\,\mathbf{x}$ matrix-vector operations as their building blocks.

This is smart because we can now compute $\mathbf{A}\,\mathbf{x}$ without assembling $\mathbf{A}$ and we can use this as a subroutine in more complex iterative solvers.

Matrix-Free Methods for Iterative Solvers (8/8)

One such method is the conjugate gradient method. Notice that there is no need to treat boundary conditions explicitly these are taken care of using stencil update rules on ghost cells as indicated in the pseudo code above.

```
function conjugate_gradient( A, x, b)
%... this is built in Matlab ...
end
```

This can be attractive as **A** often is symmetric and positive definite and may give a quadratic convergence rate compared to the linear convergence rate of the Gauss-Seidel and Jacobi style solvers.

Assignment

- Discuss what boundary conditions to apply to ϕ for a mean curvature flow problem from Page 3. Assume that ϕ is a signed distance field. (HINT: Try and sketch a signed distance map in 1D and draw a vertical line at some imaginary boundary).
- Implement a mean curvature flow simulation as outlined from Page 13 and 17. Try to make it as robust as possible.
- Assume that input ϕ is a signed distance map. Examine if this property holds throughout a simulation.
- Analyse which of the “Numerical Remedies” from Pages 15-16 that are most important to achieve a robust simulation.

Assignment

Advanced topics for self-study

- Explain upwind schemes (weak solutions)
- Walk through re-distance problem
- Walk through extrapolation problem
- Discuss how to deal with embedded surfaces (boundaries cutting cells)