Solving Partial Differential Equations Numerically

Miklós Bergou

with: Gary Miller, David Cardoze, Todd Phillips, Mark Olah

Overview

- What are partial differential equations?
- How do we solve them? (Example)
- Numerical integration
- Doing this quickly

Partial Differential Equations

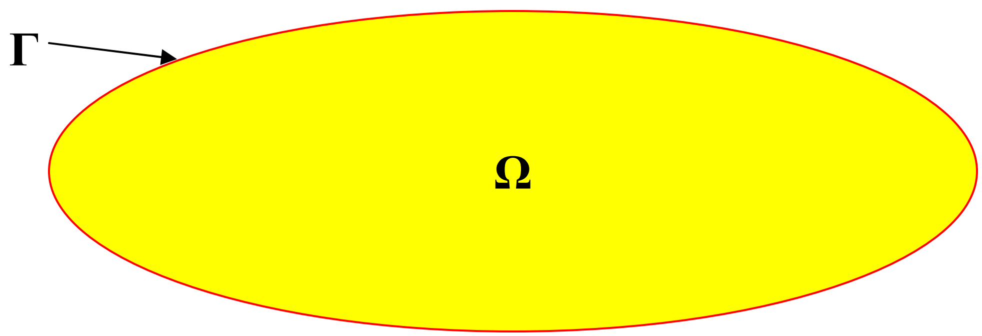
- Equation involving functions and their partial derivatives
- Example: Wave Equation

$$\frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} + \frac{\partial^2 \psi}{\partial z^2} = \frac{1}{v^2} \frac{\partial^2 \psi}{\partial t^2}$$

- We wish to know ψ, which is function of many variables
- Typically, no analytical solution possible

Problem Domain

- Want to solve problem for specific domain
- Ω: bounded open domain in space R^n
- Γ: boundary of Ω
- If domain 2-D, we have following:



Navier-Stokes Equation

- Model of incompressible fluid flow
- Governed by equation:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} - \nu \nabla^2 \mathbf{u} = -\frac{\nabla P}{\rho} + \mathbf{a} \qquad \text{in } \Omega \times (0, \mathrm{T}]$$

$$\nabla \cdot \mathbf{u} = 0 \qquad \text{in } \Omega \times (0, \mathrm{T}]$$

$$\mathbf{u} = 0 \qquad \text{on } \Gamma \times (0, \mathrm{T}]$$

$$\mathbf{u}(\mathbf{x}) = \mathbf{u}^0 \qquad \text{at } t = 0$$

u: fluid velocity

P: pressure

ρ: mass density

ν: dynamic viscosity

a: acceleration due to external force

Simplifying Navier-Stokes: Just Stokes

- Assume steady flow:

$$
\begin{aligned}
&(\mathbf{u}\cdot\nabla)\mathbf{u} - \nu\nabla^2\mathbf{u} = -\frac{\nabla P}{\rho} + \mathbf{a} && \text{in } \Omega \\
&\nabla\cdot\mathbf{u} = 0 && \text{in } \Omega \\
&\mathbf{u} = 0 && \text{on } \Gamma
\end{aligned}
$$

- Neglect convection:

$$
\begin{aligned}
&-\nu\nabla^2\mathbf{u} = -\frac{\nabla P}{\rho} + \mathbf{a} && \text{in } \Omega \\
&\nabla\cdot\mathbf{u} = 0 && \text{in } \Omega \\
&\mathbf{u} = 0 && \text{on } \Gamma
\end{aligned}
$$

Solving Stokes Equation

$$\begin{array}{ll} -\nu\nabla^2\mathbf{u} = \mathbf{f} & \text{in } \Omega \\ \nabla\cdot\mathbf{u} = 0 & \text{in } \Omega \\ \mathbf{u} = 0 & \text{on } \Gamma \end{array}$$

- Space of possible functions V that could be solutions to velocity $\mathbf{u}$
- Choose any $\mathbf{v} \in V$, multiply both sides of Stokes Equation and integrate, resulting in:

$$-\nu\int_{\Omega}\mathbf{v}\cdot\nabla^2\mathbf{u}\,d\mathbf{x} = \int_{\Omega}\mathbf{v}\cdot\mathbf{f}\,d\mathbf{x}$$

Solving Stokes Equation 2

- Apply Green's Theorem and boundary conditions to obtain:

$$\nu \int_{\Omega} \nabla \mathbf{u} \cdot \nabla \mathbf{v} d\mathbf{x} = \int_{\Omega} \mathbf{f} \cdot \mathbf{v} d\mathbf{x}$$

- For notational convenience, this is written as:

$$\langle \mathbf{u}, \mathbf{v} \rangle = (\mathbf{f}, \mathbf{v})$$

Discretize the domain

- Create mesh by partitioning domain into finite elements (curved Bezier triangles)
- Create subspace V_h of V of piecewise polynomial functions with basis function defined at each node

$$\varphi(\mathbf{x}) = \text{set of basis functions}$$

$$\mathbf{v}_h(\mathbf{x}) \in V_h \Rightarrow \mathbf{v}_h(\mathbf{x}) = \sum_{i=1}^{M} \eta_i \varphi_i(\mathbf{x})$$

Linear Equations

- Since $V_h \subset V$, $\mathbf{u}_h \in V$ so we can say that in order for $\mathbf{u}_h$ to be a solution to Stokes equation, we need

$$\langle \mathbf{u}_h, \mathbf{v} \rangle = (\mathbf{f}, \mathbf{v}) \qquad \forall \mathbf{v} \in V_h$$

- In particular, must be true for basis functions, so writing $\mathbf{u}_h$ in terms of basis functions, we have:

$$\sum_{i=1}^{M} \eta_i \langle \varphi_i, \varphi_j \rangle = (\mathbf{f}, \varphi_j) \qquad j = 1, \ldots, M$$

Solution to Problem

$$\mathbf{A}\boldsymbol{\eta} = \mathbf{b}$$

$$\mathbf{A}_{ij} = \langle \varphi_i, \varphi_j \rangle$$

$$\boldsymbol{\eta} = [\eta_1, \ldots, \eta_M]^T$$

$$\mathbf{b} = [(\mathbf{f}, \varphi_1), \ldots, (\mathbf{f}, \varphi_M)]^T$$

- Knowing coefficients η_i means we know solution $\mathbf{u}_h$
- This is a system of M linear equations with M unknowns, can be solved with various numerical methods

Integration

- Computing stiffness matrix and load vector requires lots of integrals –
  $\langle \varphi_i, \varphi_j \rangle, \quad (\mathbf{f}, \varphi_i)$
- These are done numerically via quadratures:

$$\int_0^1 f(x)dx = \sum_{i=1}^{n} w_i f(x_i)$$

$$\int_0^1 \int_0^{1-y} f(x,y)dxdy = \sum_{i=1}^{n} w_i f\left(x_i, y_i\right)$$

- Choose Gauss points and weights to give high order accuracy

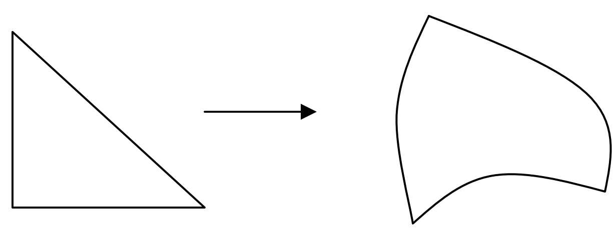
Full Navier-Stokes Equation

- Requires four types of integrals

$$\int_\Omega f \cdot g \qquad \int_\Omega \nabla f \cdot \nabla g \qquad \int_\Omega f \cdot \frac{\partial g}{\partial \alpha} \qquad \int_\Omega \frac{\partial f}{\partial \alpha} \cdot \frac{\partial g}{\partial \beta}$$

- f and g are basis functions on curved triangles
- Map from K_2 simplex to curved triangles:

$$F : K_2 \rightarrow Bezier$$



Mapping K_2 to Bezier

- Mapping requires 6 control points, defined by

$$F = \sum_{i=1}^{6} c_i b_i$$

- b_i's are Bezier basis function (polynomials)
- Jacobian is defined in standard way

$$J = \begin{bmatrix} \partial F_x / \partial x & \partial F_x / \partial y \\ \partial F_y / \partial x & \partial F_y / \partial y \end{bmatrix}$$

Integrals on K_2

$$\int_\Omega f\cdot g \qquad \int_\Omega \nabla f\cdot\nabla g \qquad \int_\Omega f\cdot\frac{\partial g}{\partial\alpha} \qquad \int_\Omega \frac{\partial f}{\partial\alpha}\cdot\frac{\partial g}{\partial\beta}$$

$$\int_{K_2}\phi\cdot\psi\cdot\left|\det J\right| \qquad \int_{K_2} J^{-T}\nabla\phi\cdot J^{-T}\nabla\psi\cdot\left|\det J\right| \qquad \int_{K_2}\phi\cdot\frac{\partial\psi}{\partial\alpha}\cdot J_\alpha^{-1}\cdot\left|\det J\right| \qquad \int_{K_2}\left(\frac{\partial\psi}{\partial\alpha}\cdot J_\alpha^{-1}\right)\cdot\left(\frac{\partial\psi}{\partial\beta}\cdot J_\beta^{-1}\right)\cdot\left|\det J\right|$$

Need for speed

- Each type of integral is done on each triangle, for each combination of basis functions
- After each time step, mesh moves (Lagrangian), so integrals need to be done for each timestep
- Speed and accuracy are required

Speedup: Cacheing

- Cache K_2 basis functions, so only Jacobian needs to be evaluated for each element
- Basis functions cached at Gauss points, so functions are essentially just arrays of values
- Cache Jacobian for each element as well, since it is reused in every integral
- Large speedup versus recomputation each time

Idea: Expand integrals

- Jacobian determinant can be written in terms of Bezier basis functions with coefficients given by control points

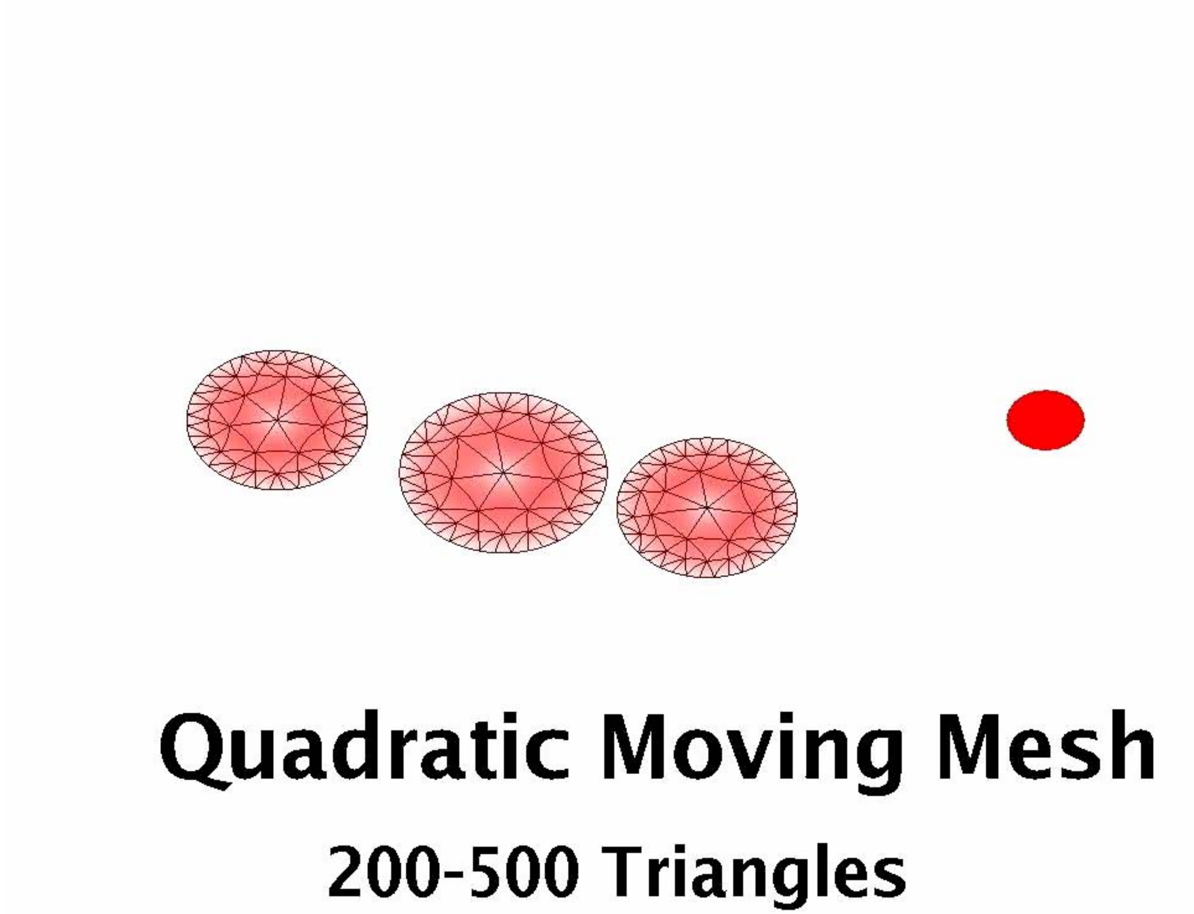
$$\int_{\Omega} f \cdot g = \int_{K_2} \phi \cdot \psi \left|\det J\right| = \sum_{i=1}^{6} n_i \int_{K_2} \phi \cdot \psi \cdot b_i$$

- Integrals in this sum no longer depend on control points
- Precompute integrals, compute coefficients only

Reusing old values

- After each time step, large portions of mesh unchanged
- If Jacobian of element is “close enough” to old value, reuse old integrals
- Speedup depends on measure of “close enough” and how much mesh changes

Example movie



Questions?