Lecture XVI - Debugging Geometry

Best Practices in writing geometry and simulation code

Challenges in Writing Code for Geometry

- Short but rich (in math, physics, indexing tricks, hopes and dreams…)
  - Where to even begin when something goes wrong?
- Bugs can be very subtle
  - A NaN/10e-4 deviation/bad conditioning in the 1st act will fire in the next.
- Bugs are often by design rather than by mistake
  - Ill-defined systems
  - Unstable numerical practices
  - Subtle details are confusing

Ill-Defined and Illl-Posed Systems

- Problem: trying to directly solve a problem as if the solution is unique
  - Or if the problem is continuous
- Example: linear system with a null space
- Approach:
  - Did you forget something? Or add something redundant?
  - If not, add regularization
    - For instance, minimum 2-norm

Poorly-conditioned Systems

- Condition number: $\kappa(A) = \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)}$
  - ratio of max to min singular values
- Meaning: for a system $Ax = b$ small changes in b can lead to big changes in x.
  - Example: the small $\pm\epsilon$ off points in CW1 (Geometry).
- Common issue: unbalanced energies.
- Approach:
  - Are you solving something that makes sense?
  - Convert/scale the problem

Unstable/Inefficient Numerical Practices

- Comparing floating points directly
  - Solution: use tolerance (or exact numbers)
- Computing quantities with unstable functions
  - Example: using $cot(acos(e_1 \cdot e_2))$ rather than $\frac{e_1 \cdot e_2}{|e_1 \times e_2|}$
- Adequate time integration

Know What You Are Doing

- Don’t implement an algorithm before
  - You understand what the input/output are
  - You know how to construct all intermediate quantities
  - …and why they are like this.
- Thing to look out for
  - Correct dimensionality
  - Sparsity patterns
  - Matrix properties: PSD, symmetry, etc.
  -

Proper Spatial Discretization

- Geometric discretization assumptions may break when:
  - Mesh is non-manifold/non-orientable
  - Triangle shape is bad or degenerated (finite-element quality deteriorates)
  - Too little dofs (mesh is too coarse)
    - Refinement doesn’t always help! <Schwartz Lantern>
- Solution: check input mesh beforehand for assumptions
  - If needed, remesh

Face the Challenges

- Things ppl say are hard but are easy
  - Solving huge sparse linear systems
    - Pivoted Cholesky decomposition takes milliseconds.
- Easy but are hard
  - Constraints/collision handling in a simulation (most papers ignore this as if it’s trivial)
- Easy and are easy
  - As-rigid-as-possible variations
- Hard and are hard
  - Solving Navier-Stokes

Managing Expectations

- Models for simulations are very simplified
  - Linear elasticity
  - Columb friction
  - “Rigid” bodies with restitution
  - Discrete-time collisions
- More realism is commensurate with algorithmic complexity
  - Not always!

Best Practices: Index Hell

- 90% of time spent on geometry processing coding: figure out what goes where in a sparse matrix.
- How to get it less wrong:
  - Permutation matrices: have a matrix that rearranges indices wrap your operators.
    - Decouples vertex2face, vertex2tet, etc. from “per element action”.
  - Order the input to utilize block diagonal structure as much as possible
  - A lot of pen-and-paper beforehand…

Best Practices: Check Properties

- Most operators/matrices will have verifiable properties:
- Invariants & Equivariants:
  - Constant null space for Laplacian/differentials
  - Also linear precision
  - Stiffness matrix has a translation null space (sometimes rotation)
  - Angle defect sums up to χ
  - Inputs with closed-form solutions
    - Parameterization of developables
    - Reconstruction of low-order polynomial surfaces
- Verify permutation matrices copy a single non-zero input to where it should be
- Verify matrix rank and PSDness
  - For both: positivity of low eigenvalues

Best Practices: Unlocking Features Gradually

- Some algorithms/inputs can be cascaded for simplicity before unlocking the “advanced” features
  - Use it to identify where problems begin
- Examples:
  - Simulation of rigid bodies without angular velocity/inertia tensor to check if collision works
  - Start with meshes without boundary
  - or only really good meshes

Best Practices: Visualization

- Sometimes just looking at the result immediately reveals the existence of a bug
  - where it’s hard keeping track of the numbers
- #siggraph_fails

Best Practices: Start Small

- Don’t dry up for 2 hours on trying an algorithm on a mesh of 1M vertices if you don’t know it’s going to work
- 80% of bugs can be resolved with 1K vertices!