

**Dan Morris**  
**Notes on ArtDefo**

What is ArtDefo?

- Mechanism for modeling a deformable object, we all know what the big picture is
- I'm going to basically use BEM today when I mean ArtDefo

What do they present in this paper?

- Overview and motivation
- Background calculus
- Implementation
- Optimization

I'm going to replace the last section with "things I don't understand yet..." for this discussion...

## Intro

If I said “go implement BEM for me”, what would the inputs and outputs of your code be?

- Inputs
  - Material properties (more detail later)
  - Geometry (more detail later)
  - Known forces (generally not known)
  - Known displacements (haptic device position)
- Outputs
  - Computed forces (used for haptic feedback, the rest are typically zero)
  - Computed displacements (used for rendering)
- Note that conceptually this method allows each point to have *either* a force or a displacement specified, but they don't provide complete formulae for the case in which you specify non-zero forces, and it's not clear how complicated that case would be. Basically they present the typical haptic-probe-touching-an-object case.

What's nice about BEM? I.e., BEM beats x because blah... the x's on the table are LEM, FEM, and m-s

- Fast when not too many boundary conditions change (relative to FEM)
  - What does that mean?
  - Not too many known p's are converted into known u's
- Small number of elements relative to the size of the volume (relative to FEM)
- Nominally simple to implement (relative to FEM)
- Easily represent global properties (improvement over m-s)
- Static, doesn't explode if you fall behind (relative to m-s)
- Truly volumetric (relative to LEM, m-s)
- Control / constraints can theoretically go both ways (although they rule this out in a tiny footnote for their implementation)
  - A quick review of control mods :
    - An object under admittance control accepts forces and does the right thing geometrically
    - An object under impedance control gets deformed geometrically and gives you back forces
    - LEM, for example, is only admittance controlled... you can't tell element x to be at a certain position without re-inverting a big slow matrix

What are the limitations of BEM?

- Need to re-invert matrices a lot (unlike LEM, m-s)
- Matrices are  $O(n^2)$  (unlike LEM, m-s)
- Can pre-compute the hard work (I don't know what the deal is with FEM in this respect)
- Homogeneous materials only
- Linear elastic assumption
- No stresses and strains inside the body without more math (unlike FEM)

## Background calculus / modeling (boundary integral formulation)

What is Navier's equation?

- Tells us that for a homogenous material, the world works out such that if forces are just applied on the surface, you can find out what's going on at the rest of the surface without understanding everything inside. Very much not obvious, but very cool.
- Sets a constraint that something must be equal to zero. This is where the "=" comes from. Always important to find that.

What parameters does Navier's equation need?

- $\nu$  = The ratio of (a) the transverse to (b) the longitudinal normal strains in a strain field produced by a uniaxial stress along the longitudinal axis of an object.
  - So this does *not* measure how "stiff" an object is. The question is answers is "I pushed 5cm along this axis, what did this other axis do?" And maybe I had to push like hella-crazy hard to get there...
  - What does it mean if  $\nu$  is :
    - 0 : completely compressible, doesn't feel that it needs to expand in one direction to accommodate change in another
      - sponge, cork
    - 0.5 : completely incompressible, expands to exactly compensate for volume changes
      - Interestingly, although they say they can model this, they have an obvious division by zero (in the  $p_{ij}$  fundamental solution) in this case
    - $<0$  : Actually gets *wider* if I pull on the long axis... the only example I found was an "unwinding filament" object that someone made to flex their materials science skillz.
- $G$  : stress / strain, like all other moduli
  - What is stress?
    - Engineering word for force
  - What is strain?
    - Engineering word for displacement

What do we know about Navier's equation?

- Of course the math itself is crazy, but the world has already figured out Navier's equation pretty well
- What are these u,p equations? The "fundamental solutions"?
- The u,p equations tell me for any points (x,y) on the (continuous!) surface of an object what will happen at y if I push (p) or displace (u) x
- Relevant to know that they came – complete with notation – from this book. The point is that they didn't invent this, and good ideas can come from books, as long as they're too hard for most people to understand.

## Implementation (the BEM)

- What is the geometry? What is 'n'?
  - The number of *surface* patches, which we'll assume are triangles
  - What's the deal with 6.1: "constant element case"
  - They say "complete formulae for constant boundary elements are provided in the appendix"... what does that mean?
  - What would "linear boundary elements" be?
  - What are the implications for implementation?
- So we have an integral equation, which tells us that the integral of some function is equal to some other integral. How does one go from a surface integral equation to a discrete equation?
  - Sum the integrals over each triangle
  - Now we have more integrals...how do we perform the integral over each triangle?
    - I wish I knew... it seems to me that this is trivial for the constant elements case, but they don't constraint their math to this case.
    - Basically you know the values at some points using the magic u and p formulae they provide
  - Now we really have some matrix equations, in the form  $Hu = Gp$ . How do we solve this? This is the interesting linear algebra that they don't talk about...
  - Move unknowns to get  $Ax = b$ ... they actually store  $A^{-1}$ , since it will speed up iterations later on.

## Optimization

- SMW – a way of quickly computing the inverse of a matrix when you already know the inverse of a similar matrix
- Subdivision surfaces

## Things I don't understand, the places I futz with in my code :

- Good values for G and v
- In their gii computation, they refer to “two integrals”, what are they?
- What is  $n_i$  in their p formula?  $N_x$  or  $N_y$ ?
- Why do they say “direction cosines” instead of “components of the normal”?
- What is the interpolation matrix? Isn't it identity for constant elements, and isn't integration just multiplication by area?
- What is a good way to maintain the triangulation for an object with constant elements? I use simple objects (explain Matlab simulation)

My program that doesn't work does the following things :

- tessellates a cube
- finds the centroid of each triangle
- computes the u and p matrices
- “integrates” to make the H and G matrices
  - does all the nasty gii computation
- reorganizes G and H to make  $Ax = b$
- solves
- computes the convex hull of the “bulged”, undisplaced points for rendering
- applies that convex hull to the undisplaced and displaced points
- renders the mess