Hair Simulation
(and Rendering)

Image from Final Fantasy
(Kai’s hair)

Adrien Treuille
Overview

• Project
  • Solving Linear Systems
  • Questions About the Project

• Hair
  • Real Hair
  • Hair Dynamics
  • Hair Rendering

• Course Evaluations
Overview

- Project
  - Solving Linear Systems
- Questions About the Project
- Hair
  - Real Hair
  - Hair Dynamics
- Hair Rendering
- Course Evaluations
Solving Linear Systems

- Want to solve system of the form:

\[ Ax = b \]

- \( A \) is symmetric:

\[ A^T = A \]

- \( A \) is positive-definite:

\[ x^T Ax > 0 \quad \forall x \]
// Matrix class the solver will accept
class implicitMatrix
{
  public:
    virtual void matVecMult(double x[], double b[]) = 0;
};

// Solve Ax = b for a symmetric, positive definite matrix A
double ConjGrad(int n, implicitMatrix *A, double x[], double b[],
                double epsilon,  // how low should we go?
                int    *steps);
Implicit Matrix

// Matrix class the solver will accept
class implicitMatrix
{
public:
    virtual void matVecMult(double x[], double b[]) = 0;
};

• matVecMult: a method that performs matrix multiplication
• x: the input vector
• b: the output vector
Implicit Matrix

// Solve Ax = b for a symmetric positive definite matrix A
double ConjGrad(int n, implicitMatrix *A,
                double x[], double b[],
                double epsilon,
                int *steps);

- \( n \): number of dimensions
- implicitMatrix: matrix instance
- \( x \): the output vector
- \( b \): the input vector
- \( \epsilon \): how low should we go? \((1.0^{-5})\)
- steps: inputs the max steps and outputs the actual steps
Example 1

\[
\begin{bmatrix}
2 & 0 \\
0 & 1
\end{bmatrix}
\begin{bmatrix}
x \\
1
\end{bmatrix}
\]

#include "linearSolver.h"

class A1 : public implicitMatrix {
    public:
        virtual void matVecMult(double x[], double b[]) {
            b[0] = 2 * x[0];
            b[1] = 1 * x[1];
        }
};

int main(int argc, char **argv) {
    double x[2] = {0.0, 0.0};
    double b[2] = {1.0, 1.0};
    int steps = 100;

    implicitMatrix *a1 = new A1();
    double err = ConjGrad(2, a1, x, b, 1.0e-5, &steps);
    delete a1;

    printf("Solved in %i steps with error %f.\n", steps, err);
    printf("A1 * [%f %f]^T = [%f %f]^T.\n", x[0], x[1], b[0], b[1]);

    return 0;
}

linear-solver-example@CMU-274306$ ./solve1
Solved in 1 steps with error 0.000000.
A1 * [0.500000 1.000000]^T = [1.000000 1.000000]^T.
Example 2

\[
\begin{bmatrix}
2 & 1 \\
1 & 1 \\
\end{bmatrix}
\begin{bmatrix}
x \\
\end{bmatrix}
=
\begin{bmatrix}
3 \\
4 \\
\end{bmatrix}
\]
#include "linearSolver.h"

class A2 : public implicitMatrix {
    public:
        virtual void matVecMult(double x[], double b[]) {
            b[0] = 2.0 * x[0] + 1.0 * x[1];
            b[1] = 1.0 * x[0] + 1.0 * x[1];
        }
    }

int main(int argc, char **argv) {
    double x[2] = {0.0, 0.0};
    double b[2] = {3.0, 4.0};
    int steps = 100;

    implicitMatrix *a2 = new A2();
    double err = ConjGrad(2, a2, x, b, 1.0e-5, &steps);
    delete a2;

    printf("Solved in %i steps with error %f.\n", steps, err);
    printf("a2 * [%f %f]^T = [%f %f]^T.\n", x[0], x[1], b[0], b[1]);

    return 0;
}
Why implicitMatrix?

```cpp
#include "linearSolver.h"

class A1 : public implicitMatrix {
    public:
        virtual void matVecMult(double x[], double b[]) {
            b[0] = 2 * x[0];
            b[1] = 1 * x[1];
        }
};
```

vs

```cpp
#include "linearSolver.h"

class A2 : public implicitMatrix {
    public:
        virtual void matVecMult(double x[], double b[]) {
            b[0] = 2.0 * x[0] + 1.0 * x[1];
            b[1] = 1.0 * x[0] + 1.0 * x[1];
        }
};
```

$O(n)$ vs $O(n^2)$
Example 3

\[\begin{bmatrix}
-1 & 1 \\
1 & 1 \\
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
\end{bmatrix}
= 
\begin{bmatrix}
3 \\
4 \\
\end{bmatrix}
\]

Not positive definite!

\[\begin{bmatrix}
1 & 0 \\
\end{bmatrix}
\begin{bmatrix}
-1 & 1 \\
1 & 1 \\
\end{bmatrix}
\begin{bmatrix}
1 \\
0 \\
\end{bmatrix}
= -1\]
Example 3

- What if $A$ is not symmetric or not positive-definite?

\[ Ax = b \]

- Then solve the *normal* equations:

\[ A^T A x = A^T b \]
Overview

- Project
  - Solving Linear Systems
- Questions About the Project
- Hair
  - Real Hair
  - Hair Dynamics
- Hair Rendering
- Course Evaluations
Questions?
Overview

- Project
  - Solving Linear Systems
  - Questions About the Project
- Hair
  - Real Hair
  - Hair Dynamics
  - Hair Rendering
- Course Evaluations
Real Hair: Curly

Short curly hair
Real Hair: Straight

Long smooth hair
Real Hair

• Typical human head has 150k-200k individual strands.

• Dynamics not well understood.

• Subject still open to debate.
Overview

- Project
  - Solving Linear Systems
  - Questions About the Project
- Hair
  - Real Hair
  - Hair Dynamics
- Hair Rendering
- Course Evaluations
Hair Dynamics

• Control Mesh
• Mass-Spring Systems
• Rigid Links
• Super Helices
Hair Dynamics

- Control Mesh
- Mass-Spring Systems
- Rigid Links
- Super Helices
Control Mesh

Figure 3: a) For a brush, triangle strips can be inserted between horizontally and vertically adjacent guide hairs. b) For a human scalp, triangle strips are inserted only between horizontally adjacent guide hairs.

Figure 5: Left: a sparse hair model displayed with static links. Right: a rendered image of the interpolated dense model.

A Practical Model for Hair Mutual Interactions
Johnny T. Chang, Jingyi Jin, Yizhou Yu.
Control Mesh

ha_guide_hair.avi
Hair Dynamics

- Control Mesh
- Mass-Spring Systems
- Rigid Links
- Super Helices
Recall...

Cloth and Fur
Energy Functions

Michael Kass
Hair Model

Limp hair: Just a set of springs.
Hair Model

Add body: Angular Springs

\[ E = \frac{1}{2} \sum_{i} k_i \theta_i^2 \]
Hair Model

Alternative: More Linear Springs

Difficulty:
Each spring constant affects both bending and stretching
Discretization

Make sure energy independent of sampling.

Total energy: \[ E = \frac{1}{2} k \sum (l - l_{\text{rest}})^2 \]

Stretch 100%: \[ E = \frac{1}{2} nk \left( \frac{L}{n} \right)^2 \]

Constant energy implies:

\[ k \propto n \quad \text{or} \quad k_i \propto \frac{1}{l_i} \]

Note: High sampling --> stiffness
Consider a discretized circle.

\[ E = \frac{1}{2} k \sum \theta^2 \]

Again, constant energy implies:

\[ k \propto n \quad \text{or} \quad k_i \propto \frac{1}{l_i} \]
Disadvantages

• Torsional Rigidity
• Non-stretching of the strands
Hair simulation in Rhythm and Hues
- The Chronicles of Narnia

Tae-Yong Kim
Rhythm and Hues Studios
\[ k = \infty \rightarrow \text{implicit integration?} \]

- Implicit integrator adds stability
- Loss of angular momentum
- ‘Good’ Jacobian (filter) very important

\[
\begin{align*}
\mathbf{v}^{n+1} &= \mathbf{v}^n + dt \frac{F^{n+1}}{m} \\
\mathbf{x}^{n+1} &= \mathbf{x}^n + dt \mathbf{v}^{n+1}
\end{align*}
\]
Well, how do we preserve length then?

→ use non-linear correction
non-linear post correction

\[ v^{n+1} = v^n + dt \frac{F}{m} \]

\[ x^{n+1} = x^n + dt v^{n+1} \]

\[ \tilde{x}^{n+1} = x^{n+1} + x^{corr} \]
non-linear post correction

\[ v^{n+1} = v^n + dt \frac{F}{m} \]

\[ x^{n+1} = x^n + dtv^{n+1} \]
non-linear post correction

\[ \nu^{n+1} = \nu^n + dt \frac{F}{m} \]
\[ x^{n+1} = x^n + dt \nu^{n+1} \]
\[ \tilde{x}^{n+1} = x^{n+1} + x^{corr} \]
non-linear post correction

- Post solve correction
  - Successive relaxation until convergence
  - Guaranteed length preservation
    - Cheap simulation of $k \to \infty$
non-linear post correction

How to implement?
- Cloth simulation literatures
  - Provot 1995 (position only)
  - Bridson 2002 (impulse)
- Hair-specific relaxation possible
Predictor-corrector scheme

- Implicit Filter (Predictor)

\[ v^{n+\frac{1}{2}} = v^n + \frac{dt}{2} \frac{F^{n+1}}{m} \]

\[ x^{n+1} = x^n + dt v^{n+\frac{1}{2}} \]

\[ \tilde{x}^{n+1} = x^{n+1} + x^{corr} \]

\[ \tilde{v}^{n+\frac{1}{2}} = v^{n+\frac{1}{2}} + \frac{x^{corr}}{dt} \]

- Sharpener (Corrector)

\[ v^{n+1} = \tilde{v}^{n+\frac{1}{2}} + \frac{dt}{2} \frac{F^{n+1}}{m} \]

- Implicit Filter (Predictor)
1. First pass-implicit integration

First implicit solve to get new velocity

\[ v^{n+\frac{1}{2}} = v^n + \frac{dt}{2} \frac{F^{n+1}}{m} \]
2. First pass-implicit integration

Advance position with the predicted mid-step velocity

\[ x^{n+1} = x^n + dtv^{n+\frac{1}{2}} \]

- Advance position with the predicted mid-step velocity
3. Non-linear Correction

- Apply non-linear corrector to get position (length) right

\[ \tilde{x}^{n+1} = x^{n+1} + x^{corr} \]
4. Impulse

\[ \tilde{v}^{n+\frac{1}{2}} = v^{n+\frac{1}{2}} + \frac{x^{corr}}{dt} \]

- Change velocity due to length preservation
- Velocity may be out of sync after impulse
5. Second implicit integration

\[ \vec{v}^{n+1} = \vec{v}^{n+\frac{1}{2}} + \frac{dt}{2} \frac{F^{n+1}}{m} \]

- Filters out velocity field
- Velocity field in sync again
Hair Dynamics

- Control Mesh
- Mass-Spring Systems
- Rigid Links
- Super Helices
**Featherstone Algorithm**

---

**Impulse-based Dynamic Simulation of Rigid Body Systems**

by

Brian Vincent Mirtich

If joint $i$ is prismatic,

$$
\mathbf{g}_i^T = \begin{bmatrix} 0 \\ \mathbf{u}_i \\ \frac{\mathbf{r}}{\mathbf{c}_i \times \mathbf{f}} \end{bmatrix} = \mathbf{r} \cdot \mathbf{u}_i.
$$

The right hand side is the component of the applied force along the joint axis. This force must be supported by the actuator, hence it is $Q_i$. If joint $i$ is revolute,

$$
\mathbf{g}_i^T = \begin{bmatrix} \mathbf{u}_i \\ \mathbf{u} \times \mathbf{c}_i \\ \frac{\mathbf{f}}{\mathbf{c}_i \times \mathbf{f}} \end{bmatrix} = \mathbf{r} \cdot (\mathbf{u} \times \mathbf{c}_i) + (\mathbf{r} - \mathbf{c}_i \times \mathbf{f}) \cdot \mathbf{u}_i.
$$

The right hand side reduces to $\mathbf{r} \cdot \mathbf{u}_i$, the component of the applied torque along the joint axis. This torque must be supported by the actuator, hence it is $Q_i$. □

Substituting equation (4.23) for link $i$'s spatial acceleration into (4.21) yields

$$
\mathbf{f}_i^T = \mathbf{f}_{i-1}^T \left( \mathbf{X}_i \mathbf{a}_{i-1} + \mathbf{g} \mathbf{b}_i + \mathbf{c}_i \right) + 2 \mathbf{d}_i.
$$

Premultiplying both sides by $\mathbf{g}_i^T$ and applying Lemma 7 gives

$$
Q_i = \mathbf{g}_i^T \left( \mathbf{X}_i \mathbf{a}_{i-1} + \mathbf{g} \mathbf{b}_i + \mathbf{c}_i \right) + 2 \mathbf{d}_i,
$$

from which $Q_i$ may be determined:

$$
Q_i = \mathbf{g}_i^T \left( \mathbf{X}_i \mathbf{a}_{i-1} + \mathbf{g} \mathbf{b}_i + \mathbf{c}_i \right) + 2 \mathbf{d}_i.
$$

Substituting this expression for $Q_i$ into (4.23) and rearranging gives

$$
\mathbf{f}_i^T = \left[ \mathbf{I} + \mathbf{X}_i \left( \mathbf{I} - \mathbf{X}_i \mathbf{g}_i \mathbf{b}_i \right) \right] \mathbf{a}_{i-1} + 2 \mathbf{m}_{i-1} \mathbf{g} + \mathbf{f}_{i-1}^T \left( \mathbf{g}_i - \mathbf{c}_i \right) + \mathbf{d}_i.
$$

Comparing this to the desired form (4.21),

$$
\mathbf{f}_i^T = \left[ \mathbf{I} + \mathbf{X}_i \left( \mathbf{I} - \mathbf{X}_i \mathbf{g}_i \mathbf{b}_i \right) \right] \mathbf{a}_{i-1} + 2 \mathbf{m}_{i-1} \mathbf{g} + \mathbf{f}_{i-1}^T \left( \mathbf{g}_i - \mathbf{c}_i \right) + \mathbf{d}_i.
$$

(4.28)

$$
\mathbf{2}_i = \mathbf{2}_{i-1} + \mathbf{m} \mathbf{X}_i \left( \mathbf{2}_i + \mathbf{t}_i \mathbf{c}_i \right) + \frac{\mathbf{t}_i \mathbf{b}_i}{\mathbf{g}_i} \left[ Q_i - \mathbf{c}_i \left( \mathbf{2}_i + \mathbf{t}_i \mathbf{c}_i \right) \right].
$$

(4.29)
Rigid Links

- Fewer degrees of freedom.
- Torsional forces.
- Difficult Implementation.
- Constraints Difficult.
Hair Dynamics

- Control Mesh
- Mass-Spring Systems
- Rigid Links
- Super Helices
Super Helices

Why just use straight rods?
Super Helices

\[
\mathbf{M}[s, \mathbf{q}] \cdot \ddot{\mathbf{q}} + \mathbf{K} \cdot (\mathbf{q} - \mathbf{q}^n) = \mathbf{A}[t, \mathbf{q}, \dot{\mathbf{q}}] + \int_0^L \mathbf{J}_i \mathbf{Q}[s, \mathbf{q}, t] \cdot \mathbf{F}^i(s, t) \, ds.
\]
Super Helices

Figure 1.8: Fitting $\gamma$ for a vertical oscillatory motion of a disciplined (curly hair clump. Left: comparison between the real (top) and virtual (bottom) experiments. Right: the span $\ell_A$ of the hair clump for real data is compared to the simulations for different values of $\gamma$. In this case, $\gamma = 1 \times 10^{-10} \text{ kg} \cdot \text{m}^3 \cdot \text{s}^{-1}$ gives qualitatively similar results.

Natural hair. We used the technique presented previously to fit the parameters of the Super Helix from the real manipulated hair clump. As shown in Figure 1.9 (left), our Super Helix model adequately captures the typical nonlinear behavior of hair (buckling, bending, twisting instabilities) as well as the nervousness of curly hair when submitted to high speed motion (see Figure 1.8, left). Figure 1.9 (right) shows the fast motion of a large hair, which is realistically simulated using 3 interacting Super Helices. All these experiments also allowed us to check the stability of the simulation, even for high speed motion.

Finally, Figure 1.10 demonstrates that our model convincingly captures the complex effects occurring in a full head of hair submitted to a high speed shaking motion.
Super Helices

Part 3

Animation of a full head of hair
Overview

- Project
  - Solving Linear Systems
  - Questions About the Project
- Hair
  - Real Hair
  - Hair Dynamics
  - Hair Rendering
- Course Evaluations
The Super-Helix model aims at mimicking the collective behavior of hair by using a sparse set of guide strands that define global hair motion, in a similar way to [Daldegan et al. 1993; Chang et al. 2002] and [Terzopoulos 1996], studied in the 1D case by [Nocent and Remion 2001]. A new scheme is proposed for convincingly modelling a hair assemblage from this sparse set of guide strands. Then, we proceed to the interpolation of the hair strands using the algorithm given in [Hou et al. 1998] and modify it to accommodate interactions between guide strands.

In our animations, the final hair geometry was rendered using the NURBS representation, which provides an adjustable number of degrees of freedom, namely the Dynamic NURBS model [Qin and Terzopoulos 1996]. Using NURBS representation is a key advantage over previous methods, as it allows for a well-controlled space discretization based on Lagrangian parameters.

The Super-Helix model accurately simulates the motion of hair. Images and experiments on real hair demonstrate that the Super-Helix model provides a very significant improvement over previous methods. It allows for realistic, stable, and efficient hair simulations without the need for large numbers of nodes or very small time steps. For instance, simulation of a 10 cm long naturally straight hair strand using the algorithm given in [Hou et al. 1998] remained unstable even with 200 nodes and a time step as low as 10\(\times\)10\(^{-6}\) s. The method offers a well-controlled space discretization based on Lagrangian parameters.

The cost of computing proximity of guide strands by keeping track of the local coherence of hair motion is based on a semi-interpolating scheme to generate non-simulated interactions. Hair interactions are handled efficiently by detecting collisions. Detection is processed by exploiting temporal coherence using a semi-interpolating scheme. For instance, interpolation across the right shoulder is prevented by adding a hair clump to the set of neighboring hair strands. This allows for more realistic and efficient hair animation. Note that interpolation requires more work for hair than for curly or clumpy hair.
We decided to start with the mass-spring system since we had a working code from the in-house cloth simulator. There we started by adapting the existing particle-based simulator to hair.

In our simulator, each hair would be represented by a number of nodes, each node representing the lumped mass of certain portion of hair. In practice, each CV of guide hairs created at the grooming stage was used as the mass node. Such nodes are connected by two types of springs - linear and angular springs.
Next Wednesday

- Why not simulate every single strand?
- *Jee Lee*
Conclusion

Video
Overview

• Project
  • Solving Linear Systems
  • Questions About the Project
• Hair
  • Real Hair
  • Hair Dynamics
  • Hair Rendering
• Course Evaluations
Question

- What are the salient parts of cloth that we want to simulate?
- How could we simulate cloth?
- What are the difficulties / problems with your approach?