Recap of Mathematics

Frank Marsman

February 17, 2017

1 Linear Algebra

1.1 Definitions

- An $m \times n$ matrix A has m rows and n columns, where a_{ij} denotes the element in the ith row and jth column.
- *Unitary matrix*; a matrix who's conjugate transpose is also its inverse.
- *Hermitian matrix*; a matrix that is equal to its conjugate transpose.
- *Positive-semidefinite*; a matrix $\mathbf{M}$ for which $x^*\mathbf{M}x \geq 0$, $\forall x \in \mathbb{C}$. Positive-definite means that the product is always greater than 0.

1.2 Eigendecomposition of matrix

Let $\mathbf{A}$ be a square $N \times N$ matrix having N linearly independent eigenvectors q_i. The eigendecomposition of $\mathbf{A}$ is the factorization

$$\mathbf{A} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{-1},$$

where $\mathbf{Q}$ has the eigenvectors as columns and $\mathbf{\Lambda}$ is a diagonal matrix containing the eigenvalues; $\Lambda_{ii} = \lambda_i$.

1.3 Singular value decomposition of a matrix

The singular value decomposition (SVD) is a generalization of the eigendecomposition of a matrix. The SVD of an $m \times n$ matrix $\mathbf{M}$ is the factorization

$$\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*,$$

where $\mathbf{U}$ ($m \times m$) and $\mathbf{V}$ ($n \times n$) are two unitary matrices and $\mathbf{\Sigma}$ is an $m \times n$ rectangular diagonal matrix. The columns of $\mathbf{U}$ and $\mathbf{V}$ are the left- and right-singular vectors of $\mathbf{M}$ and the diagonal entries of $\mathbf{\Sigma}$ are its singular values.

The left-singular vectors of $\mathbf{M}$ are the orthonormal eigenvectors of $\mathbf{M}\mathbf{M}^*$ and the right-singular those of $\mathbf{M}^*\mathbf{M}$. The non-zero singular values are the square roots of the non-zero eigenvalues of both $\mathbf{M}\mathbf{M}^*$ and $\mathbf{M}^*\mathbf{M}$.

Interpretation; if $\mathbf{M}$ is square, real and with positive determinant, then $\mathbf{U}$, $\mathbf{\Sigma}$ and $\mathbf{V}$ are also square and represent geometrical transformations. The outer two matrices are rotation matrices and the middle one a scaling matrix. Thus $\mathbf{U\Sigma V}$ represents a composition of a rotation or reflection, followed by a scaling, followed by another rotation or reflection.

1.4 Polar decomposition of a matrix

The polar decomposition of a matrix is a factorization analogous to the polar form of a nonzero complex number $z = re^{i\theta}$. The polar decomposition of a square (complex) matrix $\mathbf{A}$ is the factorization

$$\mathbf{A} = \mathbf{U}\mathbf{P},$$

where $\mathbf{P}$ is a positive-semidefinite Hermitian matrix and $\mathbf{U}$ a unitary matrix. This decomposition always exists and if $\mathbf{A}$ is invertible, it is unique.

2 Projective Dynamics Paper

Let us here summarize the PBD paper. We consider a mesh consisting of m vertices with positions and velocities $\mathbf{q}, \mathbf{v} \in \mathbb{R}^{m\times 3}$. System evolves in time; $t_1, t_2, \ldots$. At time t_n the system is defined as $\{\mathbf{q}_n, \mathbf{v}_n\}$. Also, we have external and internal forces $\mathbf{f}_{\text{int}}$ and $\mathbf{f}_{\text{ext}}$, where

$$\mathbf{f}_{\text{int}}(\mathbf{q}) = -\sum_i \nabla W_i(\mathbf{q}),$$

where $W_i(\mathbf{q})$ is a scalar potential energy function. Positions are updated using the rule

$$\mathbf{q}_{n+1} = \mathbf{q}_n + h\mathbf{v}_{n+1}.$$

Velocity is updated as

$$\mathbf{v}_{n+1} = \mathbf{v}_n + h\mathbf{a}_{n+1} = \mathbf{v}_n + h\mathbf{M}^{-1}\Big(\mathbf{f}_{\text{int}}(\mathbf{q}_{n+1}) + \mathbf{f}_{\text{ext}}\Big),$$

where $\mathbf{M}$ is the *mass matrix*. This is a diagonal matrix containing the masses of the vertices on the diagonal. Masses can for instance depend on the size of the adjacent faces or something.

From the above equations we can derive

$$\mathbf{M}\big(\mathbf{q}_{n+1} - \mathbf{q}_n - h\mathbf{v}_n\big) = h^2\big(\mathbf{f}_{\text{int}}(\mathbf{q}_{n+1}) + \mathbf{f}_{\text{ext}}\big). \tag{1}$$

Finding a $\mathbf{q}_{n+1}$ that satisfies this equality is an optimization problem;

$$\min_{\mathbf{q}_{n+1}} \left\{ \frac{1}{2h^2} \left\| \mathbf{M}^{\frac{1}{2}} \left(\mathbf{q}_{n+1} - s_n \right) \right\|_F^2 + \sum_i W_i \left(\mathbf{q}_{n+1} \right) \right\}, \tag{2}$$

where $s_n \equiv \mathbf{q}_n + h\mathbf{v}_n + h^2\mathbf{M}^{-1}\mathbf{f}_{\text{ext}}$. This minimization is a compromise between the *momentum potential* (left part) and the elastic potential (sum on the right).

The elastic potentials

The idea is now that we use elastic potentials of a specific form;

$$W_i(\mathbf{q}, \mathbf{p}) = \frac{w_i}{2} \left\| \mathbf{A}_i\mathbf{q} - \mathbf{B}_i\mathbf{p} \right\|_F^2 + \delta_{\mathbf{C_i}}(\mathbf{p}),$$

where $\delta_{\mathbf{C_i}}(\mathbf{p})$ is zero if $\mathbf{p}$ lays inside the constrain set and infinite otherwise. Minimizing W_i thus corresponds to finding a point in the constrain set closest to $\mathbf{q}$.

2.1 The algorithm

Using our new potentials, we can rewrite the optimization problem (Eq. 2):

$$\min_{\mathbf{q}} \left\{ \frac{1}{2h^2} \left\| \mathbf{M}^{\frac{1}{2}} \left(\mathbf{q} - s_n \right) \right\|_F^2 + \sum_i \left[\frac{w_i}{2} \left\| \mathbf{A}_i\mathbf{S}_i\mathbf{q} - \mathbf{B}_i\mathbf{p}_i \right\|_F^2 + \delta_{\mathbf{C_i}}(\mathbf{p}_i) \right] \right\}, \tag{3}$$

where $\mathbf{S}_i$ is a constant selection matrix selecting only vertices involved in the ith constraint. Now we can finally start finding the $\mathbf{q}$ that minimizes Eq. (3). We do this in two parts, local and global. We repeat the local/global step, finding a better $\mathbf{q}$ after each iteration.

Local solve

We first minimize Eq. (3) over the auxiliary variables $\mathbf{p}_i$. So for each elastic potential, we compute (keeping $\mathbf{q}$ constant)

$$\min_{\mathbf{p}_i} \left\{ \frac{w_i}{2} \left\| \mathbf{A}_i\mathbf{S}_i\mathbf{q} - \mathbf{B}_i\mathbf{p}_i \right\|_F^2 + \delta_{\mathbf{C_i}}(\mathbf{p}_i) \right\}.$$

Global solve

We now have updated the $\mathbf{p}_i$'s such that Eq. (3) is minimized. We can now find a better value of $\mathbf{q}$ because the new $\mathbf{p}_i$'s change the optimization equation (the optimum $\mathbf{q}$ depends on the values $\mathbf{p}_i$). So we find our new $\mathbf{q}$ by solving

$$\left(\frac{\mathbf{M}}{h^2} + \sum_i w_i \mathbf{S}_i^T \mathbf{A}_i^T \mathbf{A}_i \mathbf{S}_i \right) \mathbf{q} = \frac{\mathbf{M}}{h^2} s_n + \sum_i w_i \mathbf{S}_i^T \mathbf{A}_i^T \mathbf{B}_i \mathbf{p}_i.$$

The factor on the left of $\mathbf{q}$ is a matrix that is constant. Therefore we can prefactor this matrix in order to solve this system repeatedly in an efficient way. Only the right hand side changes after each local solve, because the $\mathbf{p}_i$'s are updated during the local solve.

Derivation of global solve

Let's see if we can derive the global solve equation from Eq. (3). Let us first rewrite the two parts in this equation.

$$\begin{aligned}
\left\|\mathbf{M}^{\frac{1}{2}}(\mathbf{q}-s_n)\right\|_F^2 &= \operatorname{tr}\left((\mathbf{q}-s_n)^T(\mathbf{M}^{\frac{1}{2}})^T\mathbf{M}^{\frac{1}{2}}(\mathbf{q}-s_n)\right)\\
&= \operatorname{tr}\left((\mathbf{q}-s_n)^T\mathbf{M}(\mathbf{q}-s_n)\right)\\
&= \operatorname{tr}\left(\mathbf{q}^T\mathbf{M}\mathbf{q}-\mathbf{q}^T\mathbf{M}s_n-s_n^T\mathbf{M}\mathbf{q}+s_n^T\mathbf{M}s_n\right)\\
&= \operatorname{tr}(\mathbf{q}^T\mathbf{M}\mathbf{q})-\operatorname{tr}(\mathbf{q}^T\mathbf{M}s_n)-\operatorname{tr}(s_n^T\mathbf{M}\mathbf{q})+\operatorname{tr}(s_n^T\mathbf{M}s_n)\\
&= \mathbf{q}^T\mathbf{M}\mathbf{q}-2s_n^T\mathbf{M}\mathbf{q}+s_n^T\mathbf{M}s_n\\
&= \mathbf{q}^T\mathbf{M}\mathbf{q}-2\mathbf{q}^T\mathbf{M}s_n+s_n^T\mathbf{M}s_n\\
\|\mathbf{A}_i\mathbf{S}_i\mathbf{q}-\mathbf{B}_i\mathbf{p}_i\|_F^2 &= \operatorname{tr}\left(\left(\mathbf{q}^T\mathbf{S}_i^T\mathbf{A}_i^T-\mathbf{p}_i^T\mathbf{B}_i^T\right)\left(\mathbf{A}_i\mathbf{S}_i\mathbf{q}-\mathbf{B}_i\mathbf{p}_i\right)\right)\\
&= \mathbf{q}^T\mathbf{S}_i^T\mathbf{A}_i^T\mathbf{A}_i\mathbf{S}_i\mathbf{q}+\mathbf{p}_i^T\mathbf{B}_i^T\mathbf{B}_i\mathbf{p}_i-2\mathbf{q}^T\mathbf{S}_i^T\mathbf{A}_i^T\mathbf{B}_i\mathbf{p}_i
\end{aligned}$$

Let us now minimize Eq. (3).

$$\begin{aligned}
0 &= \frac{1}{2h^2}\left\|\mathbf{M}^{\frac{1}{2}}(\mathbf{q}-s_n)\right\|_F^2+\sum_i\left[\frac{w_i}{2}\|\mathbf{A}_i\mathbf{S}_i\mathbf{q}-\mathbf{B}_i\mathbf{p}_i\|_F^2\right]\\
&= \frac{1}{2h^2}(\mathbf{q}^T\mathbf{M}\mathbf{q}-2\mathbf{q}^T\mathbf{M}s_n+s_n^T\mathbf{M}s_n)\\
&\quad+\sum_i\frac{w_i}{2}(\mathbf{q}^T\mathbf{S}_i^T\mathbf{A}_i^T\mathbf{A}_i\mathbf{S}_i\mathbf{q}+\mathbf{p}_i^T\mathbf{B}_i^T\mathbf{B}_i\mathbf{p}_i-2\mathbf{q}^T\mathbf{S}_i^T\mathbf{A}_i^T\mathbf{B}_i\mathbf{p}_i)\\
&\Rightarrow \frac{1}{2h^2}(\mathbf{q}^T\mathbf{M}\mathbf{q}+s_n^T\mathbf{M}s_n)+\sum_i\frac{w_i}{2}(\mathbf{q}^T\mathbf{S}_i^T\mathbf{A}_i^T\mathbf{A}_i\mathbf{S}_i\mathbf{q}+\mathbf{p}_i^T\mathbf{B}_i^T\mathbf{B}_i\mathbf{p}_i)\\
&= \frac{1}{h^2}\mathbf{q}^T\mathbf{M}s_n+\sum_i w_i\mathbf{q}^T\mathbf{S}_i^T\mathbf{A}_i^T\mathbf{B}_i\mathbf{p}_i.
\end{aligned}$$

Ah, we have to take the derivative. We then lose the two terms that do not contain $\mathbf{q}^T$, and we obtain

$$\left(\frac{\mathbf{M}}{h^2}+\sum_i w_i\mathbf{S}_i^T\mathbf{A}_i^T\mathbf{A}_i\mathbf{S}_i\right)\mathbf{q}=\frac{\mathbf{M}}{h^2}s_n+\sum_i w_i\mathbf{S}_i^T\mathbf{A}_i^T\mathbf{B}_i\mathbf{p}_i.$$

2.2 Working things out using specific potential functions

We will now define some potential functions that can be actually used.

2.2.1 Strain

We can define strain as the local deviation from a rigid motion. We will define the strain for each triangle in the mesh. So we will have a potential for each triangle. We will now work in $\mathbb{R}^2$ For a given triangle with area A, we define the strain potential as

$$W(\mathbf{q}, \mathbf{T}) \equiv \frac{w}{2} A \left\| \mathbf{X_f} \mathbf{X_g}^{-1} - \mathbf{T} \right\|_F^2 ,$$

where $\mathbf{X_f} = [\mathbf{e}_1 \ \mathbf{e}_2] \in \mathbb{R}^{2\times 2}$ contains the triangle edges of the current configuration (defined by $\mathbf{q}$) and $\mathbf{X_g}$ contains the edges of the rest (initial) configuration. Apparently, the rotation $\mathbf{T}$ minimizing this potential transforms the current triangle to match the rest configuration the best. Let us test this. If the current triangle is in the same configuration, we have $\mathbf{X_f} = \mathbf{X_g} \Rightarrow \mathbf{X_f}\mathbf{X_g}^{-1} = \mathbf{I}$. This seems correct; in this case W is minimized when $\mathbf{T} = \mathbf{I}$, which makes sense when the triangle already matches perfectly. In general, the potential is zero if $\mathbf{X_f} = \mathbf{T}\mathbf{X_g}$, which makes sense. If the current triangle is obtained by applying a rotation to the original one, we have no strain.

3 Solutions to problems discussed with Klaus

3.1 Strain

We have an original triangle $A = (\mathbf{e}_1 \ \mathbf{e}_2)$ that changes to triangle B. We want to define the strain for this deformation, where a rotation does not creates strain. We thus need to find a rotation R that we apply to A that minimizes the difference with B. If the difference is zero then B is just a rotated version of A and we have no strain. We have

$$E = \min_{R \in SO(2)} \|B - RA\|_F^2 ,$$

which we should be able to write in the following form

$$E(\mathbf{x}) = \mathbf{x}^T C \mathbf{x} + \mathbf{b}^T \mathbf{x} + c.$$

Let’s start;

$$\begin{aligned}\|B-RA\|_F^2 &= \operatorname{tr}\Big((B-RA)^T(B-RA)\Big)\\ &= \operatorname{tr}\Big((B^T-A^TR^T)(B-RA)\Big)\\ &= \operatorname{tr}\Big(B^TB-B^TRA-A^TR^TB+A^TR^TRA\Big)\\ &= \operatorname{tr}\Big(B^TB-B^TRA-A^TR^TB+A^TA\Big)\\ &= \operatorname{tr}(B^TB)-2\operatorname{tr}(B^TRA)+\operatorname{tr}(A^TA),\end{aligned}$$

since $B^TRA=(A^TR^TB)^T$ and $\operatorname{tr}(X)=\operatorname{tr}(X^T)$ for all square matrices X. We use the rule $\operatorname{tr}(X^TY)=\operatorname{vec}(Y)^T\operatorname{vec}(X)$ to further solve this equation;

$$\begin{aligned}\operatorname{tr}(B^TB) &= \operatorname{vec}(B)^T\operatorname{vec}(B),\\ \operatorname{tr}(B^TRA) &= \operatorname{vec}(RA)^T\operatorname{vec}(B),\\ \operatorname{tr}(A^TA) &= \operatorname{vec}(A)^T\operatorname{vec}(A).\end{aligned}$$

We now have

$$\begin{aligned}\|B-RA\|_F^2 &= \operatorname{tr}(B^TB)-2\operatorname{tr}(B^TRA)+\operatorname{tr}(A^TA)\\ &= \operatorname{vec}(B)^T\operatorname{vec}(B)-2\operatorname{vec}(RA)^T\operatorname{vec}(B)+\operatorname{vec}(A)^T\operatorname{vec}(A)\\ &= \mathbf{x}^T\mathbf{x}+\mathbf{b}^T\mathbf{x}+c, \qquad (4)\end{aligned}$$

where $\mathbf{x}=\operatorname{vec}(B)=\binom{\mathbf{e}_1}{\mathbf{e}_2}$, $\mathbf{b}=-2\operatorname{vec}(RA)$ and $c=\operatorname{vec}(A)^T\operatorname{vec}(A)$.

3.2 Minimizing the energy

We want to minimize $\|B-RA\|_F^2$ by finding the best rotation matrix R. Let us write A, B and R in matrix form;

$$A=\begin{bmatrix}x_1 & x_2\\ y_1 & y_2\end{bmatrix},\quad B=\begin{bmatrix}\tilde{x}_1 & \tilde{x}_2\\ \tilde{y}_1 & \tilde{y}_2\end{bmatrix},\quad R=\begin{bmatrix}\cos\theta & \sin\theta\\ -\sin\theta & \cos\theta\end{bmatrix}.$$

We have

$$RA=\begin{bmatrix}\cos\theta & \sin\theta\\ -\sin\theta & \cos\theta\end{bmatrix}\begin{bmatrix}x_1 & x_2\\ y_1 & y_2\end{bmatrix}=\begin{bmatrix}x_1\cos\theta+y_1\sin\theta & x_2\cos\theta+y_2\sin\theta\\ -x_1\sin\theta+y_1\cos\theta & -x_2\sin\theta+y_2\cos\theta\end{bmatrix}.$$

Now we can write out the vector form of the matices;

$$\operatorname{vec}(A)=\begin{bmatrix}x_1\\ y_1\\ x_2\\ y_2\end{bmatrix},\quad \operatorname{vec}(B)=\begin{bmatrix}\tilde{x}_1\\ \tilde{y}_1\\ \tilde{x}_2\\ \tilde{y}_2\end{bmatrix},\quad \operatorname{vec}(RA)=\begin{bmatrix}x_1\cos\theta+y_1\sin\theta\\ -x_1\sin\theta+y_1\cos\theta\\ x_2\cos\theta+y_2\sin\theta\\ -x_2\sin\theta+y_2\cos\theta\end{bmatrix}.$$

We are getting close to rewriting Eq. (4);

$$\begin{aligned}
\mathbf{x}^T\mathbf{x} &= \operatorname{vec}(B)^T \operatorname{vec}(B) = \tilde{x}_1^2 + \tilde{y}_1^2 + \tilde{x}_2^2 + \tilde{y}_2^2 \\
c &= \operatorname{vec}(A)^T \operatorname{vec}(A) = x_1^2 + y_1^2 + x_2^2 + y_2^2 \\
\mathbf{b}^T\mathbf{x} &= -2\operatorname{vec}(RA)^T \operatorname{vec}(B) \\
&= -2\Big(\tilde{x}_1(x_1\cos\theta + y_1\sin\theta) + \tilde{y}_1(-x_1\sin\theta + y_1\cos\theta) \\
&\quad + \tilde{x}_2(x_2\cos\theta + y_2\sin\theta) + \tilde{y}_2(-x_2\sin\theta + y_2\cos\theta)\Big) \\
&= -2\cos\theta\cdot(x_1\tilde{x}_1 + y_1\tilde{y}_1 + x_2\tilde{x}_2 + y_2\tilde{y}_2) \\
&\quad - 2\sin\theta\cdot(y_1\tilde{x}_1 + y_2\tilde{x}_2 - x_1\tilde{y}_1 - x_2\tilde{y}_2) \qquad (5)
\end{aligned}$$

Since Eq. (4) represents $\|B - RA\|_F^2$, it must always be ≥ 0. Also, both $\mathbf{x}^T\mathbf{x}$ and c are always ≥ 0. So in order to minimize $\|B - RA\|_F^2$, we must minimize $\mathbf{b}^T\mathbf{x}$, or Eq (5). This equation is of the form

$$\big(\mathbf{b}^T\mathbf{x}\big)(\theta) = a\cos\theta + b\sin\theta, \qquad (6)$$

where $a = -2(x_1\tilde{x}_1 + y_1\tilde{y}_1 + x_2\tilde{x}_2 + y_2\tilde{y}_2)$ and $b = -2(y_1\tilde{x}_1 + y_2\tilde{x}_2 - x_1\tilde{y}_1 - x_2\tilde{y}_2)$ We have to find the θ that minimizes this sum;

$$\begin{aligned}
0 &= \frac{\partial\mathbf{b}^T\mathbf{x}}{\partial\theta} = -a\sin\theta + b\cos\theta \\
&\Rightarrow a\sin\theta = b\cos\theta \\
&\Rightarrow \frac{b}{a} = \frac{\sin\theta}{\cos\theta} = \tan\theta \\
&\Rightarrow \theta = \arctan\frac{b}{a} = \arctan\frac{y_1\tilde{x}_1 + y_2\tilde{x}_2 - x_1\tilde{y}_1 - x_2\tilde{y}_2}{x_1\tilde{x}_1 + y_1\tilde{y}_1 + x_2\tilde{x}_2 + y_2\tilde{y}_2}. \qquad (7)
\end{aligned}$$

So, if everything is correct, we now have an expression for the value of θ that defines the rotation matrix R that minimizes $\|B - RA\|_F^2$. It is now time to implement this in my program and see what happens.

4 Geometric Modelling recap

4.1 Definitions

Let us recap some concepts.

Linear polynomials on meshes

We can define functions on a mesh using *Langrange basis functions*, which are linear polynomials on a mesh ϕ_i which are zero on each vertex except at vertex i. Vertices are denoted by v_i. A linear polynomial in a triangle has a value $u(p)$ at point p inside the triangle T given by

$$u(p) = \sum_{i=1}^{3} u_i \phi_i(p),$$

where the u_i are the values of u at the vertices.

Gradients of linear polynomials on meshes

For the gradient of u we have

$$\nabla u(p) = \sum_{i=1}^{3} u_i \nabla \phi_i(p) = \frac{1}{2 \cdot \text{area}(T)} \sum_{i=1}^{3} u_i R^{90^\circ} e_i,$$

where $R^{90^\circ} e_i$ is the vector representing the edge opposite to vertex i, rotated by 90 degrees. We can write this in matrix form:

$$\nabla u(p) = \frac{1}{2 \cdot \text{area}(T)} [u_1 \; u_2 \; u_3]^T [R^{90^\circ} e_1 \; R^{90^\circ} e_2 \; R^{90^\circ} e_3].$$

Polynomials and gradients for whole mesh

Ok, let's redo the above for the total mesh M containing n vertices. The space of all continuous linear polynomials on M is called S_h. For any polynomial $u \in S_h$ we have

$$u(x) = \sum_{i=1}^{n} u_i \phi_i(x).$$

The gradient of any function in S_h is a linear map $G : S_h \to V_h$. We can write G as a matrix of $3\#T$ rows and $\#V$ columns, assembled from the elementary matrices

$$\frac{1}{2 \cdot \text{area}(T)} [R^{90^\circ} e_1 \; R^{90^\circ} e_2 \; R^{90^\circ} e_3].$$

Summary of matrices used for meshes

M is diagonal $n \times n$ matrix containing A_{p_i}'s on diagonal; a third of sum of areas of triangles adjacent to p_i. We can define L^2 scalar product of two functions u, v as

$$\int_M u(x)v(x)\ dx \approx u^T M v.$$

M_V is $3m \times 3m$ matrix containing areas of all m triangles on diagonal (each area A_{T_i} is repeated 3 times. Inner product of two vectorfields V, W on M is given by

$$\int_M \langle V(x), W(x) \rangle\ dx = V^T M_V W.$$

G was the gradient matrix, that maps functions to vector fields. We have

$$S \equiv G^T M_V G.$$

Dirichlet energy of a function u is

$$E_D(u) = \int_M \langle \nabla u, \nabla u \rangle\ dx = \frac{1}{2} u^T S u = \frac{1}{2} u^T G^T M_V G u.$$

We define the Laplace matrix L as

$$L \equiv M^{-1} S = M^{-1} G^T M_V G.$$

4.2 Deformation-based editing

Denote by $x \in S_h^3$ the map that maps each vertex to its positions in $\mathbb{R}^3$ and by $u \in S_h^3$ a displacement of the surface. A deformation energy measures the energy in a deformation; $E : S_h^3 \to \mathbb{R}$.

Dirichlet energy of deformation; $E_D(u) = \frac{1}{2} u^T S u$. Laplace-based energy;

$$E_L(u) = \frac{1}{2} u^T L^T M L u = \frac{1}{2} u^T S M^{-1} M M^{-1} S u = \frac{1}{2} u^T S M^{-1} S u.$$

4.3 Assignment

After a meeting with Klaus we thought it's a good idea to read the assignment about gradients again. In the assignment, the user can deform a mesh by applying a 3×3 matrix to the gradients of the embedding. We define 3 functions x_x, x_y, x_z on the mesh M. The values of these functions are the vertex coordinates. So one function x_x has the x coordinate of vertex i at the position of this vertex, idem for other coordinates. We then compute three gradients (vector fields on M); g_x, g_y and g_z are the gradients of the corresponding functions. We then multiply each gradient of each triangle by a user defined matrix, obtaining three new gradients $\tilde{g}_x, \tilde{g}_y$ and $\tilde{g}_z$.

We then compute the new coordinate functions $\tilde{x}_x, \tilde{x}_y, \tilde{x}_z$ such that their gradients match the new gradients;

$$G^T M_V G \tilde{x}_x = G^T M_V \tilde{g}_x,$$

idem for y, z. Let's solve. We have $S \equiv G^T M_V G$, and S is square, symmetric (and Hermitian) and positive semi defenite. So we have

$$S\tilde{x} = G^T M_V \tilde{g}.$$

We can solve this by factorizing (Cholensky); $S = LL^*$. We have $S\tilde{x} = LL^*\tilde{x} = G^T M_V \tilde{g}$. Then we define $y = L^*\tilde{x}$, so $Ly = G^T M_V \tilde{g}$. Then we find y by forward substitution and then find $\tilde{x}$ by solving $L^*\tilde{x} = y$ using back substitution.

5 Operators and Matrices

5.1 Frobenius norm

The *Frobenius norm* $\|A\|_F$ of a matrix A is given by

$$\|A\|_F \equiv \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2} = \sqrt{\operatorname{tr}(A^*A)}.$$

5.2 Transpose of matrix

Let us review the properties of the transpose operation. Transposing a matrix 'flips' it around its diagonal. We have

$$(A+B)^T = A^T + B^T,$$

$$(AB)^T = B^T A^T,$$

$$(cA)^T = cA^T.$$

If a is orthogonal, then $A^T = A^{-1}$.

5.3 Trace of matrix

The trace of a square matrix is the sum of the elements on the main diagonal. Some properties of this operation are

$$\operatorname{tr}(A+B) = \operatorname{tr}(A) + \operatorname{tr}(B),$$

$$\operatorname{tr}(cA) = c\operatorname{tr}(A),$$

$$\operatorname{tr}(A^T) = \operatorname{tr}(A),$$

$$\operatorname{tr}(X^TY) = \operatorname{vec}(Y)^T\operatorname{vec}(X),$$

where $\operatorname{vec}(X)$ is the vectorization of X; it is a column vector containing all columns of X, stacked on top of each other.

5.4 Mass Matrix

The mass matrix is a diagonal matrix M containing the area's around the vertices on the diagonal. Area A_p of a vertex can be defined in multiple ways. One way is to define it as one third of the total area of all faces connected to p.

5.5 Laplace operator

The Laplace operator is given by the divergence of the gradient of a function on Euclidean space;

$$\Delta f \equiv \nabla^2 f = \nabla \cdot \nabla f = \sum_{i=1}^{n} \frac{\partial^2 f}{\partial x_i^2}.$$

6 Simple implementation in 2D

Time do get real. Let's work out all the details and write out an exact implementation. We will start with a simple potential. We'll model all edges as springs connecting two particles (vertices). We have m vertices and F edges. Our position and velocity vectors are

$$\mathbf{q}_n \in \mathbb{R}^{2m} = \begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ \vdots \\ x_m \\ y_m \end{bmatrix}, \quad \mathbf{v}_n \in \mathbb{R}^{2m} = \begin{bmatrix} \dot{x}_1 \\ \dot{y}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_m \\ \dot{y}_m \end{bmatrix}.$$

Also, for practicality, we denote vertex i by $\mathbf{w}_i$:

$$\mathbf{w}_i \equiv \begin{bmatrix} x_i \\ y_i \end{bmatrix}.$$

We do not use any external force. The mass matrix of our system will be simple at first. If it gives weird behavior, we'll change it. It contains the masses of the vertices on the diagonal. Since we are in 2D, we repeat each mass twice;

$$\mathbf{M} = \begin{bmatrix} M_1 & 0 & \dots & \dots & 0 \\ 0 & M_1 & 0 & \dots & 0 \\ \vdots & 0 & M_2 & \dots & 0 \\ \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & 0 & M_m \end{bmatrix} = \mathbf{I}_{2m}.$$

6.1 The potentials

Now let's work out the potentials W_i. Every edge $e_i = \{a_i, b_i\}, i \in [1, E]$ behaves like a spring with rest length l_i. It connects vertices a_i and b_i, where, for simplicity $b_i > a_i$. We obtain their positions from the position vector $\mathbf{q}$ using a selection matrix

$\mathbf{S}_i$, which is obtained by taking the identity matrix $\mathbf{I}_{2m}$ and removing all rows except row $2a_i$ and $2a_i - 1$ and the same for b_i.

$$\mathbf{S}_i = \begin{bmatrix} 0 & \dots & 0 & 1 & 0 & 0 & \dots & \dots & \dots & 0 \\ 0 & \dots & 0 & 0 & 1 & 0 & \dots & \dots & \dots & 0 \\ 0 & \dots & \dots & \dots & \dots & 0 & 1 & 0 & \dots & 0 \\ 0 & \dots & \dots & \dots & \dots & 0 & 0 & 1 & \dots & 0 \end{bmatrix}.$$

So all elements of $\mathbf{S}_i$ are zero except the element in row 1 and column $2a_i - 1$, row 2 and column $2a_i$, row 3 and column $2b_i - 1$ and row 4 and column $2b_i$. We have

$$\mathbf{S}_i\mathbf{q} \in \mathbb{R}^4 = \begin{bmatrix} x_{a_i} \\ y_{a_i} \\ x_{b_i} \\ y_{b_i} \end{bmatrix} = \begin{bmatrix} \mathbf{w}_{a_i} \\ \mathbf{w}_{b_i} \end{bmatrix}.$$

For our constrain manifold, we define

$$\mathcal{C}_i \equiv \left\{ \mathbf{p} \in \mathbb{R}^4 = \begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \end{bmatrix} : (x_1 - x_2)^2 + (y_1 - y_2)^2 = l_i^2 \right\}.$$

And our elastic potential;

$$W_i(\mathbf{q}, \mathbf{p}) = \frac{w_i}{2} \|\mathbf{S}_i\mathbf{q} - \mathbf{p}\|_F^2 + \delta_{\mathcal{C}_i}(\mathbf{p}).$$

So that

$$W_i(\mathbf{q}) = \min_{\mathbf{p}} \left\{ \frac{w_i}{2} \|\mathbf{S}_i\mathbf{q} - \mathbf{p}\|_F^2 + \delta_{\mathcal{C}_i}(\mathbf{p}) \right\}.$$

Let us now find the $\mathbf{p}_i$'s minimizing the W_i's. We write $\mathbf{p}_i$ as

$$\mathbf{p}_i = \begin{bmatrix} \mathbf{w}'_{a_i}(i) \\ \mathbf{w}'_{b_i}(i). \end{bmatrix} = \begin{bmatrix} x'_{a_i}(i) \\ y'_{a_i}(i) \\ x'_{b_i}(i) \\ y'_{b_i}(i) \end{bmatrix}.$$

The x' and y' are the 'new' positions of the vertices, dictated by potential W_i. Now let's find $\mathbf{w}'_{a_i}(i)$ and $\mathbf{w}'_{b_i}(i)$:

$$\mathbf{w}'_{a_i}(i) = \mathbf{w}_{a_i} + \frac{1}{2}\Delta l_i \cdot \hat{\mathbf{r}}_i \quad \text{and} \quad \mathbf{w}'_{b_i}(i) = \mathbf{w}_{b_i} - \frac{1}{2}\Delta l_i \cdot \hat{\mathbf{r}}_i,$$

where

$$\mathbf{r}_i \equiv \mathbf{w}_{b_i} - \mathbf{w}_{a_i}$$

and

$$\Delta l_i \equiv l_i - |\mathbf{r}_i|.$$

We now have defined the $\mathbf{p}_i$'s in terms of the current vertex positions.

6.1.1 Using different metric for spring potentials

After a discussion with Klaus, we decided that the current definition of the spring potential doesn't do our technique justice. The reason is that we now have $\mathbf{A}_i = \mathbf{B}_i = \mathbf{I}$. This results in our system matrix being diagonal, which is easy to solve by any solver, not just a Cholesky method. So let us define another metric for our spring system. First we define the constraint manifold $\mathcal{C}$:

$$\mathcal{C}_i \equiv \left\{ \mathbf{p} \in \mathbb{R}^2 = \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} : (\Delta x)^2 + (\Delta y)^2 = l_i^2 \right\}.$$

To clarify; our constraint manifold contains all difference vectors (vector from one vertex on the spring to the other) of length l_i. However, we do not have direct access to the difference vector, but only to the actual vertex positions;

$$\mathbf{S}_i\mathbf{q} \in \mathbb{R}^4 = \begin{bmatrix} x_{a_i} \\ y_{a_i} \\ x_{b_i} \\ y_{b_i} \end{bmatrix}.$$

We need a matrix $\mathbf{A}_i$ that maps $\mathbf{S}_i\mathbf{q}$ to a vector pointing from vertex a to b;

$$\mathbf{A}_i\mathbf{S}_i\mathbf{q} = \begin{bmatrix} x_{b_i} - x_{a_i} \\ y_{b_i} - y_{a_i} \end{bmatrix} = \begin{bmatrix} \Delta x_i \\ \Delta y_i \end{bmatrix} \Rightarrow \mathbf{A}_i = \begin{bmatrix} -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}.$$

Note that $\mathbf{A}_i$ is the same for all spring systems. We can now redefine our potential

$$\begin{aligned} W_i(\mathbf{q}, \mathbf{p}) &= \frac{w_i}{2} \left\| \mathbf{A}_i\mathbf{S}_i\mathbf{q} - \mathbf{p} \right\|_F^2 + \delta_{\mathcal{C}_i}(\mathbf{p}) \\ &= \frac{w_i}{2} \left\| \begin{bmatrix} \Delta x_i \\ \Delta y_i \end{bmatrix} - \begin{bmatrix} p_x \\ p_y \end{bmatrix} \right\|_F^2 + \delta_{\mathcal{C}_i}(\mathbf{p}) = \frac{w_i}{2} \left\| \begin{bmatrix} \Delta x_i - p_x \\ \Delta y_i - p_y \end{bmatrix} \right\|_F^2 + \delta_{\mathcal{C}_i}(\mathbf{p}) \\ &= \frac{w_i}{2} \Big((\Delta x_i - p_x)^2 + (\Delta y_i - p_y)^2 \Big) + \delta_{\mathcal{C}_i}(\mathbf{p}). \end{aligned}$$

Let us look at how to find the $\mathbf{p}$ that minimizes W_i. First note that $\mathbf{p}$ must lay on the constraint manifold. This guarantees us that $\mathbf{p}$ must lay somewhere on a circle of radius l_i centered at the origin. Minimizing W_i comes down to minimizing the distance between the point $(\Delta x_i, \Delta y_i)$ and $\mathbf{p}$. One can easily see that $\mathbf{p}$ must be the point where the line through the origin and $(\Delta x_i, \Delta y_i)$ intersects the circle of radius l_i centered at the origin. Note that there are two such points, only one of which is correct. We find the correct point by multiplying $(\Delta x_i, \Delta y_i)$ by a factor such that the length of the resulting vector equals l_i;

$$\mathbf{p}_i = \frac{l_i}{\sqrt{(\Delta x_i)^2 + (\Delta y_i)^2}} \begin{bmatrix} \Delta x_i \\ \Delta y_i \end{bmatrix} = \frac{l_i}{\sqrt{\|\mathbf{A}_i\mathbf{S}_i\mathbf{q}\|^2}} \mathbf{A}_i\mathbf{S}_i\mathbf{q}.$$

Since $\mathbf{A}$ is no longer the identity matrix, our (constant) system matrix changes. The new system we have to solve is

$$\left(\frac{\mathbf{M}}{h^2} + \sum_i w_i \mathbf{S}_i^T \mathbf{A}_i^T \mathbf{A}_i \mathbf{S}_i\right)\mathbf{q} = \frac{\mathbf{M}}{h^2} s_n + \sum_i w_i \mathbf{S}_i^T \mathbf{A}_i^T \mathbf{p}_i.$$

Let us now work out some parts of this equation first. Let us look at some matrix products. First let's look at $\mathbf{A}_i\mathbf{S}_i$, which is a $(2 \times 2m)$ matrix:

$$\begin{aligned}\mathbf{A}_i\mathbf{S}_i &= \begin{bmatrix}-1 & 0 & 1 & 0\\ 0 & -1 & 0 & 1\end{bmatrix}\begin{bmatrix}0 & \dots & 0 & 1 & 0 & 0 & \dots & \dots & \dots & 0\\ 0 & \dots & 0 & 0 & 1 & 0 & \dots & \dots & \dots & 0\\ 0 & \dots & \dots & \dots & \dots & 0 & 1 & 0 & \dots & 0\\ 0 & \dots & \dots & \dots & \dots & 0 & 0 & 1 & \dots & 0\end{bmatrix}\\ &= \begin{bmatrix}0 & \dots & -1 & 0 & \dots & 1 & 0 & \dots & 0\\ 0 & \dots & 0 & -1 & \dots & 0 & 1 & \dots & 0\end{bmatrix}.\end{aligned}$$

The product $\mathbf{S}_i^T\mathbf{A}_i^T\mathbf{A}_i\mathbf{S}_i$ is a $(2m \times 2m)$ matrix that has zero's everywhere except for 8 points. On the diagonal, we have ones at positions $2a, 2a+1, 2b, 2b+1$ (a and b are indices of vertices of spring). In column $2a$ and row $2b$ we have a -1, as well as in col $2a+1$ and row $2b+1$. The same is true if we swap col/row (the matrix is symmetric). Now let's look at $\mathbf{S}_i^T\mathbf{A}_i^T\mathbf{p}_i$. It is a column vector having $2m$ elements.

$$\mathbf{S}_i^T\mathbf{A}_i^T\mathbf{p}_i = \begin{bmatrix}0\\ \vdots\\ -p_x\\ -p_y\\ \vdots\\ p_x\\ p_y\\ \vdots\\ 0\end{bmatrix}.$$

First nonzero element is in row $2a$ and the third in row $2b$.

6.2 Computing $\mathbf{q}_{n+1}$ using global/local solve

First we perform the local solve by finding the $\mathbf{p}_i$'s from the current $\mathbf{q}$. We then update $\mathbf{q}$ during the global step by solving

$$\left(\frac{\mathbf{M}}{h^2} + \sum_i w_i \mathbf{S}_i^T \mathbf{S}_i\right)\mathbf{q} = \frac{\mathbf{M}}{h^2} s_n + \sum_i w_i \mathbf{S}_i^T \mathbf{p}_i,$$

where

$$s_n \in \mathbb{R}^{2m} = \mathbf{q}_n + h\mathbf{v}_n.$$

Let's look at some terms to see what they do. We have

$$\mathbf{S}_i^T \mathbf{p}_i \in \mathbb{R}^{2m} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ x'_{a_i}(i) \\ y'_{a_i}(i) \\ 0 \\ \vdots \\ 0 \\ x'_{b_i}(i) \\ y'_{b_i}(i) \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \mathbf{w}'_{a_i}(i) \\ 0 \\ \vdots \\ 0 \\ \mathbf{w}'_{b_i}(i) \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

These terms are vectors containing 'desired' new vertex positions. Note that a vertex can occur in multiple potentials and thus in multiple $\mathbf{p}_i$'s. For this reason, we define

$$\mathcal{W}_k \equiv \left\{ i \in [1, E] : k \in e_i \right\},$$

in other words, $\mathcal{W}_k$ contains the indices of the edges (and thus potentials) that contain vertex k. We now define

$$\mathbf{w}'_k \in \mathbb{R}^2 \equiv \sum_{i \in \mathcal{W}_k} w_i \mathbf{w}'_k(i).$$

This definition allows us to rewrite the sum

$$\sum_i w_i \mathbf{S}_i^T \mathbf{p}_i = \begin{bmatrix} \mathbf{w}'_1 \\ \mathbf{w}'_2 \\ \vdots \\ \mathbf{w}'_E \end{bmatrix} \in \mathbb{R}^{2m}.$$

So we now know that

$$\overbrace{\frac{\mathbf{M}}{h^2} s_n}^{\text{momentum}} + \underbrace{\sum_i w_i \mathbf{S}_i^T \mathbf{p}_i}_{\text{internal forces}}$$

is a vector in $\mathbb{R}^2$. Now let's look at $\mathbf{S}_i^T \mathbf{S}_i$. It is an $2m \times 2m$ matrix in which all elements are zero except four on the diagonal. The $2a_i$th element on the diagonal is 1, as well as elements $2a_i - 1$, $2b_i$ and $2b_i - 1$. This makes

$$\sum_i w_i \mathbf{S}_i^T \mathbf{S}_i = \begin{bmatrix} \sum\limits_{i \in \mathcal{W}_1} w_i & & & \\ & \sum\limits_{i \in \mathcal{W}_1} w_i & & \mathbf{0} \\ & & \ddots & \\ & \mathbf{0} & & \ddots & \\ & & & & \sum\limits_{i \in \mathcal{W}_m} w_i \end{bmatrix}.$$

Now we can write

$$\left(\frac{\mathbf{M}}{h^2} + \sum_i w_i \mathbf{S}_i^T \mathbf{S}_i \right) \mathbf{q} = \begin{bmatrix} \left(\frac{M_1}{h^2} + \sum\limits_{i \in \mathcal{W}_1} w_i \right) x_1 \\ \left(\frac{M_1}{h^2} + \sum\limits_{i \in \mathcal{W}_1} w_i \right) y_1 \\ \left(\frac{M_2}{h^2} + \sum\limits_{i \in \mathcal{W}_2} w_i \right) x_2 \\ \vdots \\ \left(\frac{M_m}{h^2} + \sum\limits_{i \in \mathcal{W}_m} w_i \right) y_m \end{bmatrix}.$$

Okay. We know everything now. We can write a program. Let's write some pseudo code to make it more clear.

6.3 Code

```
class Sim {
  matrix $\mathbf{M}$;
  vector $\mathbf{q}$;
  vector <vector> $\mathbf{P}$; // auxiliary variables ($\mathbf{P}_i = \mathbf{p}_i$)

  void UpdatePs( ); // updates auxiliary variables $\mathbf{p}_i$'s
  void NextStep( ); // compute next $\mathbf{q}$
}
```

The local solve:

```
void UpdatePs( ) {
  for $i \in [1, E]$ {
    $l$ = $l_i$;
    $a$ = $a_i$;
    $b$ = $b_i$;
    $\mathbf{r}$ = $\mathbf{w}_b - \mathbf{w}_a$; // vector pointing from $a$ to $b$
    $\Delta l$ = $|r| - l$;
    $\hat{\mathbf{r}}$ = $\mathbf{r} \div |\mathbf{r}|$;
    $\mathbf{w}{*}_a$ = $\mathbf{w}_a$ + $\frac{1}{2}\Delta l \hat{\mathbf{r}}$;
    $\mathbf{w}{*}_b$ = $\mathbf{w}_b$ - $\frac{1}{2}\Delta l \hat{\mathbf{r}}$;
    $\mathbf{P}[i]$ = $[\mathbf{w}{*}_a, \mathbf{w}{*}_b]$;
  } // for
} // UpdatePs
```