

Physics-based Animation: A Mathematical Perspective

Based on Ladislav Kavan's CS6660 with annotations from wxgopher

June 17, 2019

Contents

1	Classical Mechanics	2
1.1	Basics: a harmonic oscillator	2
2	Time Integration	2
3	Optimization	2
3.1	Implicit Newmark	2
3.2	Optimization Problems	2
3.2.1	Solving unconstrained problems	3
3.2.2	Newton's method	3
3.3	Numerical Linear Algebra	4
3.3.1	Direct solvers	4
3.3.2	Iterative solvers	5
4	Elastic Materials and Finite Element Simulation	6
5	Fluids and Partial Differential Equations	7
6	Something more...	7
7	Papers and books	7

This lecture note is based on Dr. Ladislav Kavan's Physics-based Animation course CS6660, originally taught at the University of Utah. I also try to include some supplementary materials from other resources.

Warning: Although as I strive to make this material useful, there are certain bugs, use this material at your own risk. I would also be grateful to hear [feedbacks](#).

.....

1 Classical Mechanics

1.1 Basics: a harmonic oscillator

2 Time Integration

3 Optimization

3.1 Implicit Newmark

$$x_{n+1} = x_n + \frac{h}{2}(v_n + v_{n+1}), \quad (1a)$$

$$v_{n+1} = v_n + \frac{h}{2}(f_n + f_{n+1}), \quad (1b)$$

where x and v are positions and velocities, and $f_n \equiv f(x_n)$. Thus we have

$$x_{n+1} - x_n - hv_n = \frac{h^2}{4}(f_n + f_{n+1}). \quad (2)$$

Let $y = x_n + hnv_n + \frac{h^2}{4}f_n$, $x_{n+1} = x$, note that $\nabla E = -f$, then (2) becomes

$$x - y = \frac{h^2}{4}f \Rightarrow x - y + \frac{h^2}{4}f\nabla E = 0. \quad (3)$$

Let $g(x) = \frac{1}{2}\|x - y\|^2 + \frac{h^2}{4}E$, then solving g equals to solve x for

$$\min_x g(x). \quad (4)$$

3.2 Optimization Problems

Problem formulation:

$$\min_x g(x), \quad x \in R^n, \quad g \in R. \quad (5)$$

Optimization problems can be categorized into constrained or unconstrained problems, or convex or non-convex problems.

Theorem 1. For a convex problem (where both objective and feasible set are convex), if the objective is C^2 , then the Hessian $H \succeq 0$, and the local minimum is the global minimum.

Definition 1. A **linear programming (LP)** is a problem with linear objective and linear equality or inequality constraints. A **quadratic programming (QP)** is the same as LP except with a quadratic objective.

Note 1. Convex QP has polynomial time solver but non-convex QP is NP-hard.

Example 1. A non-convex QP:

$$\min_x \frac{1}{2} \|Ax\|^2, \quad (6a)$$

$$s.t. \quad \|x\|_2 = 1. \quad (6b)$$

Note 2. Software package for solving non-convex problem: IpOPT, KNITRO, or [NEOS-Guide](#)

3.2.1 Solving unconstrained problems

There are two ways to solve an unconstrained problem: descent method or trust-region method.

Descent method refers to pick a descent direction d and do an exact or inexact line search (LS) to determine descent distance αd .

Definition 2. Descent direction: $\forall \alpha \in (0, \alpha_0), \alpha_0 > 0, g(x + \alpha d) < g(x)$.

Definition 3. Exact LS refers to solving the following problem:

$$\arg \min_{\alpha > 0} g(x + \alpha d).$$

Backtracking LS

3.2.2 Newton's method

A usual descent direction is called **gradient descent** (GD), denoted by $d = -\nabla g$. It doesn't work well sometime, for example:

Example 2.

$$\min_{x_1, x_2} g(x_1, x_2) = \frac{1}{2}(x_1^2 + \gamma x_2^2), \quad \gamma > 0. \quad (7)$$

If we do exact LS, we get

$$x_1^k = \gamma \left(\frac{\gamma - 1}{\gamma + 1} \right)^k, \quad (8a)$$

$$x_2^k = \left(-\frac{\gamma - 1}{\gamma + 1} \right)^k. \quad (8b)$$

If gamma becomes huge, we can tell GD converges slowly (as two GD directions are perpendicular).

Newton's method refers to do a quadratic approximation to problem (5), given by

$$g(x + p) = g(x) + \nabla g^T(x)p + \frac{1}{2}p^T \nabla^2 g(x)p + O(\|p\|^2) \equiv \tilde{g}(x) + O. \quad (9)$$

Thus we have

$$\frac{\partial \tilde{g}}{\partial p} = 0 \iff \nabla g + \nabla^2 g p = 0.$$

The **Newton descent direction** is given by

$$p = -(\nabla^2 g)^{-1} \nabla g. \quad (10)$$

Note 3. Newton's method is affine invariant. Let $x = Ty$ with T being a nonsingular matrix, so problem (5) is equivalent to problem

$$\min_y \tilde{g}(y). \quad (11)$$

We also have

$$\nabla \tilde{g} = T^2 \nabla g, \quad \nabla^2 \tilde{g} = T^T \nabla^2 g T.$$

Newton direction Δy of \tilde{g} and Δx of g is given by

$$\Delta y = -T^{-1} \nabla^2 g \nabla g = T^{-1} \Delta x.$$

Definition 4. Newton decrement: substitute Newton descent direction (10) into (9), we get the **Newton decrement**

$$\lambda^2 = g - \frac{1}{2} \nabla g^T (\nabla^2 g)^{-1} \nabla g. \quad (12)$$

A stopping criteria for Newton's method would be

$$\frac{\lambda^2}{2} \leq \epsilon.$$

Note that Newton decrement is also affine invariant.

Convergence analysis of the Newton's method is given as follows:

Theorem 2. Kantorovich: if g is strictly convex and $\nabla^2 g$ is Lipschitz continuous, then $\exists \epsilon > 0, \eta > 0, \gamma > 0$, s.t.

Damped phase I If $\|\nabla g\| \geq \eta$, then $\|g^{k+1}\| \leq \|g^k\| - \gamma$;

Quadratic phase II If $\|\nabla g\| < \eta$, then $\|g^{k+1}\| \leq c \|g^k\|^2$.

Some cases where Newton's method doesn't work well:

1. If g is highly nonlinear, the damped phase will be long;
2. If g is non-convex, solution might not converge to a minimum.

In practical implementations, the Hessian might not always be positive definite. A possible Hessian modification is:

Note 4. Hessian modification: Consider $A = \nabla^2 g + cI$, where $c > 0$. It is shown that if $\nabla^2 g = Q\Lambda Q^T$ ($\nabla^2 g$ is symmetric, hence it is always diagonalizable), then $A = Q(\Lambda + cI)Q^T$, hence A is positive definite given sufficiently large $c > 0$.

Solving Newton's direction (10) refers to solving a linear system. We introduce several techniques to tackle this.

3.3 Numerical Linear Algebra

We consider solving a linear system $Ax = b$.

3.3.1 Direct solvers

We can prefactor A into $A = UV^T$, with U and V being low rank matrices.

Common factorization methods are

LU If A is nonsingular, then there exist L, P, U , where they are lower-triangular matrix, permutation matrix (hence orthogonal), and upper-triangular matrix, respectively, s.t. $A = PLU$.

Sparse LU $A = P_1 L U P_2$, where P_1 and P_2 are permutations to utilize sparse information of A .

Cholesky For symmetric positive-definite matrix, we have $A = LL^T$ with unique L . Note that for positive semi-definite matrix, Cholesky is not unique.

LDLT

3.3.2 Iterative solvers

Iterative solvers includes:

1. Classical method: Jacobian, Gauss-Seidel, SOR (super over-relaxation), etc.;
2. Krylov subspace method: CG (conjugate gradient)¹, GMRES (generalized minimal residual), etc.;
3. multigrid method.

Classical method Suppose P is easy to invert, let $A = P + A - P$, then

$$Ax = b \Leftrightarrow Px = (P - A)x + b.$$

$P = \text{diag}(A)$ gives us Jacobian method, and it's GPU-friendly; $P = \text{lower}(A)$ gives us GS method; $P = \text{diag}(A) + w \cdot \text{lower}(A)$ gives us SOR method where $w < 2$.

The residual term follows

$$r_k \equiv A(x^* - x_k) = b - Ax_k \Leftrightarrow r_{k+1} = (I - P^{-1}A)r_k.$$

To make classical method converge, we need $\rho(I - P^{-1}A) < 1$, where $\rho(\cdot)$ is the spectrum radius.

Krylov subspace method Assuming $A^T = A$ and $A \succeq 0$, then

$$\arg \min_x g(x) = \frac{1}{2}x^T Ax - x^T b \Leftrightarrow \text{solve } Ax = b. \quad (13)$$

Let $\nabla g_k = Ax_k - b = -r_k$ ² as a descent direction, the line search for (13) is as follows:

1. Solve α from $\arg \min_{\alpha} (\frac{1}{2}(x + \alpha r)^T A(x + \alpha r) - (x + \alpha r)^T b) \equiv \frac{1}{2}\|x + \alpha r\|_A^2 - (x + \alpha r)^T b$ by letting $\frac{\partial E}{\partial \alpha} = 0$, where E is the objective;
2. α follows $\alpha = -\frac{\|r\|^2}{\|r\|_A^2}$;
3. $x_{k+1} = x_k + \alpha_k r_k$.

This is the gist of CG from gradient descent.

CG Assuming solving problem (13).

Definition 5. A-conjugate A set of vectors $\{p_i\}$ are A-conjugate iff $\forall i, j, i \neq j, p_i^T A p_j = 0$.

The CG algorithm is as follows:

1. We define $r_k = Ax_k - b$;
2. Let $p_1 = r_1$ ³;
3. $p_k = r_k + \beta_k p_{k-1}, \beta_k = \frac{r_k^T A p_{k-1}}{p_{k-1}^T A p_{k-1}}$;
4. Do LS along direction p_k , we have $\alpha_k = \frac{r_k^T p_k}{p_k^T A p_k}$;
5. $x_{k+1} = x_k + \alpha_k p_k$.

¹A good reference on CG is [6].

² Ax is computed on-fly, we don't need the whole A .

³ r_1 is usually chosen as a GD step

Preconditioning CG If A is ill-conditioned, we consider adding preconditioner M to A , s.t. $\text{cond}(MA) < \text{cond}(A)$.

The Jacobian preconditioner is given by $M = \text{diag}(A)^{-1}$. The incomplete Cholesky is given by pre-factoring A into $L_a L_a^T$ where L_a is an approximation of L of the Cholesky decomposition of A given by $A = LL^T$.

4 Elastic Materials and Finite Element Simulation

Consider kinetic energy

$$k = \frac{1}{2} v^T M v.$$

Assuming unit mass, we have

$$k = \frac{1}{2} \int_{\Omega} \rho(p) \|v(p)\|^2 dp, \quad (14)$$

where $p \in \Omega$ is the material point. We discretize Ω , and for every node p_i we define piece-wise shape function $s_i : \Omega \mapsto [0, 1]$, s.t. $\sum_i s_i = 1$. For every region E_t , we also define

$$r_t(p) = \begin{cases} 1 & p \in E_t, \\ 0 & \text{otherwise.} \end{cases}$$

We have

$$v(p) = \sum_i v_i s_i(p), \quad (15a)$$

$$\rho(p) = \sum_t \rho_t r_t(p). \quad (15b)$$

The kinetic energy follows

$$k = \frac{1}{2} \int_{\Omega} \sum_t \rho_t r_t \|v(p)\|^2 dp \quad (16a)$$

$$= \frac{1}{2} \sum_t \rho_t \int_{\Omega} r_t(p) \|v(p)\|^2 dp \quad (16b)$$

$$= \frac{1}{2} \sum_t \rho_t \int_{E_t} \|v\|^2 dp \quad (16c)$$

$$= \frac{1}{2} \sum_t \rho_t \sum_{i,j} v_i^T v_j \int_{E_t} s_i s_j dp. \quad (16d)$$

$$(16e)$$

We define

$$\int_{E_t} s_i s_j dp = s_{i,j}^t, \quad (17)$$

for different methods of discretization, we have

Triangal element (2-dimension)

$$s_{i,j}^t = \begin{cases} \frac{\text{Area}(t)}{6} & i = j, \\ \frac{\text{Area}(t)}{12} & i \neq j. \end{cases}$$

Tetrahderal element (3-dimension)

$$s_{i,j}^t = \begin{cases} \frac{\text{Area}(t)}{10} & i = j, \\ \frac{\text{Area}(t)}{20} & i \neq j. \end{cases}$$

Let's consider practical implementation of FEM. From discussions above, we have

$$\int_{E_t} \|v(p)\|^2 dp = \sum_{i,j} v_i^T v_j \int_{E_t} s_i s_j dp \quad (18a)$$

$$= \sum_{i,j} v_i^T v_j s_{i,j}^t \quad (18b)$$

$$= V^T S_t V, \quad (18c)$$

where $V = (v_1, v_2, \dots, v_n)^T$, and

$$S_t = \begin{pmatrix} s_{1,1}^t \cdot I_3 & \cdots & s_{1,n}^t \cdot I_3 \\ \vdots & \ddots & \vdots \\ s_{n,1}^t \cdot I_3 & \cdots & s_{n,n}^t \cdot I_3 \end{pmatrix}.$$

5 Fluids and Partial Differential Equations

6 Something more...

7 Papers and books

Numerical optimization can refer to [2] [4].

[1] is a great reference for continuum mechanics and FEM.

An awesome book on fluids simulation is [3].

Vector calculus can be found in [5].

References

- [1] Javier Bonet and Richard D Wood. *Nonlinear continuum mechanics for finite element analysis*. Cambridge university press, 1997.
- [2] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.
- [3] Robert Bridson. *Fluid simulation for computer graphics*. AK Peters/CRC Press, 2015.
- [4] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.
- [5] Kaare Brandt Petersen, Michael Syskind Pedersen, et al. The matrix cookbook. *Technical University of Denmark*, 7(15):510, 2008.
- [6] Jonathan Richard Shewchuk et al. An introduction to the conjugate gradient method without the agonizing pain, 1994.