Geometric Modeling
Summer Semester 2010

Variational Modeling
Basic Techniques · Surface Modeling · Other Applications
Overview...

Topics:

• Triangle Meshes & Multi-Resolution Representations
• Subdivision Surfaces
• Implicit Functions
• Variational Modeling
  ▪ Introduction
  ▪ Variational Framework
  ▪ Variational Function Fitting Toolkit
  ▪ Euler & Lagrange – Some More Mathematical Background
  ▪ Surface Modeling
  ▪ Other Applications
Variational Modeling

Introduction
Motivation

Surface modeling techniques we have seen so far:

• Bivariate polynomial spline patches
  ▪ Quad (tensor product) patches
  ▪ Triangular patches
• Subdivision surfaces
• Implicit functions
Motivation

Problems:

• Bivariate polynomial spline patches
  ▪ General topologies are hard to handle
  ▪ Need to adapt base mesh to user constraints (control points, boundaries etc)

• Subdivision surfaces
  ▪ More flexible than spline patches
  ▪ Problems:
    – Continuity at extraordinary vertices
    – Still need to build a base mesh

• Implicit functions
  ▪ Nice tool – but how do we construct actual surfaces?
Variational Modeling:

- Different approach:
  - We formulate smoothness properties in terms of a penalty function
  - Set additional constraints (handle points, normals, etc)
  - Then solve for the “optimal function”

- No direct manipulation of control points...
  - We use B-Splines or implicit functions as tool for the numerical representation, no direct user interaction.
  - “Meta tool”: compute control points automatically
Two Views:

In this lecture:

- Narrow view:
  - Use variational techniques for modeling shapes

- General view:
  - Short introduction / overview to variational calculus and practical techniques.
  - Application examples in geometry processing.

Applications beyond geometric modeling are important:

- Variational approaches are ubiquitous in computer graphics and computer vision
Variational Modeling
Basic Techniques
Calculus of Variation

Basic Idea:

• We look at a set of functions $f: S \rightarrow D$

• We define an “energy functional” $E: (S \rightarrow D) \rightarrow \mathbb{R}$
  - A functional assigns real numbers to functions
  - Each function gets a “score”
  - “Energy” means: the smaller the better

• We set up additional requirements (“constraints”) on $f$.
  - *Soft constraints* $\rightarrow$ violation increases energy.
  - *Hard constraints* $\rightarrow$ violation not allowed.

• We then compute the function(s) $f$ that minimize $E$. 
Calculus of Variation

Very general framework:

- A lot of problems can be directly formulated as variational problems.
- Example 1:
  - We are looking for a curve.
  - It should be as smooth as possible (energy = non-smoothness).
  - It should go through a number of points (hard constraints).
Calculus of Variation

Another example:

- **Problem:** We want to go to the moon.
- **Given:**
  - Orbits of moons, planets and star(s).
  - Flight conditions (atmosphere, gravitation of stellar bodies)
- **Unknowns:**
  - Throttle (magnitude, direction) from rocket motors (vector function)
- **Energy function:**
  - Usage of rocket fuel (the fewer the better)
  - Perhaps: Overall travel time (maybe not longer than a week)
To the moon:

- **Constraints:**
  - We want to start in Cape Canaveral (upright trajectory) and end up on the moon.
  - We do not want to hit moons or planets on our way.
  - We want to approach the moon at no more than 20 km/h relative speed upon touchdown.
  - The rocket motor has a limited range of forces it can create (not more than a certain thrust, no backward thrust)

So flying to the moon is just minimizing a functional.
(Ok, this is slightly simplified)
A Simple Example

Simple example: variational splines

- Energy:
  - We want smooth curves
  - Smooth translates to minimum curvature
  - Quadratic penalty:

\[
E(f) = \int_\text{curve} \left| \text{curvature}_f(t) \right|^2 \, dt
\]
A Simple Example

Simple example: variational splines

- Energy:
  - Problem: curvature is non-linear
  - Easier to minimize: second derivatives
  - Equivalent in case of a unit-speed parametrization (which is tricky to enforce)

\[ E(f) = \int_{\text{curve}} \left( \frac{d^2}{dt^2} f(t) \right)^2 \, dt \]
A Simple Example

Simple example: variational splines

- Constraints:
  - Hard constraints: we are given parameter values $t_1, ..., t_n$ at which we should meet control points $p_1, ..., p_n$.

  \[
  E(f) = \int_{t=t_1}^{t_n} \left[ \frac{d^2}{dt^2} f(t) \right]^2 dt
  \]

  - We already know the solution to this problem: Piecewise cubic interpolating spline.
A Simple Example

Simple example: variational splines

- More interesting: soft constraints
  - We are given parameter values $t_1, \ldots, t_n$ at which we should approximately meet control points $p_1, \ldots, p_n$.

$$E(f) = \int_{t=t_1}^{t=t_n} \left[ \frac{d^2}{dt^2} f(t) \right]^2 dt + \lambda \sum_{i=1}^{n} (f(t_i) - p_i)^2$$

- $\lambda$ controls the smoothness of the result. Large values reduce smoothness to meet the control points more precisely.
A Simple Example

Simple example: variational splines

- Soft constraints
  - We are given parameter values $t_1, ..., t_n$ at which we should approximately meet control points $p_1, ..., p_n$, up to a specific accuracy for each point.
  - We can specify the accuracy by error quadrics $Q_1, ..., Q_n$.

$$E(f) = \int_{t=t_1}^{t_n} \left[ \frac{d^2}{dt^2} f(t) \right]^2 dt + \sum_{i=1}^{n} (f(t_i) - p_i)^T Q_i (f(t_i) - p_i)$$
The rank deficient error quadric trick:

• A rank-1 matrix constraints the curve in one direction only
• Useful for point-to-surface constraints (minimize normal direction deviation, tangential motion is free)

\[ Q_i = \sigma [nn^T] \]
Numerical Treatment

Numerical computation:

- No closed form solution
- Instead:
  - Discretize (finite dimensional function space)
  - Solve for coefficients (coordinate vector in this function space)
Finite Differences

FD solution:

- Represent curve as array of k values:

<table>
<thead>
<tr>
<th>$t$</th>
<th>0</th>
<th>0.1</th>
<th>0.2</th>
<th>...</th>
<th>7.4</th>
<th>7.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y$</td>
<td>$y_0$</td>
<td>$y_1$</td>
<td>$y_2$</td>
<td>...</td>
<td>$Y_{74}$</td>
<td>$y_{75}$</td>
</tr>
</tbody>
</table>

- Unknowns are the curve points $y_1, \ldots, y_k$
Discretized Energy Function

Discretized Energy Function:

- Energy is a squared linear expression $\rightarrow$ quadratic discrete objective function
- Constraints are quadratic by construction
- We obtain a quadratic energy function that can be solved by a linear system

\[
E(f) = \int_{t=t_1}^{t_n} \left( \frac{d^2}{dt^2} f(t) \right)^2 dt + \sum_{i=1}^{n} (f(t_i) - p_i)^T Q_i (f(t_i) - p_i)
\]

\[
E^{(discr)}(f) = \sum_{i=1}^{k} \left( \frac{y_{i-1} - 2y_i + y_{i+1}}{h^2} \right)^2 + \sum_{i=1}^{n} (y_{index(t_i)} - p_i)^T Q_i (y_{index(t_i)} - p_i)
\]

(neglected here: handling boundary values)
Summary:

• Variational approaches look like this:

\[
\text{compute } \arg\min_{f \in F} E(f),
\]

\[
E(f) = E^{(\text{data})}(f) + E^{(\text{regularizer})}(f),
\]

\[
f \in F = \{f \mid f \text{ satifies hard constraints}\}
\]

• Connection to statistics:
  - Bayesian maximum a posteriori estimation
  - \( E^{(\text{data})} \) is the data likelihood (log space)
  - \( E^{(\text{regularizer})} \) is a prior distribution (log space)
Variational Toolbox
Data Fitting, Regularizer Functionals, Discretizations
Toolbox

In the following:

• We will discuss...
  ▪ ...useful standard functionals.
  ▪ ...how to model soft constraints.
  ▪ ...how to model hard constraints.
  ▪ ...how to discretize the model.

• Then snap & click your favorite custom variational modeling scheme.

• (Click & snap means: add together to a composite energy)
Functionals
Functionals

Standard Functional #1: Function norm

- Given a function $f: \mathbb{R}^m \supset \Omega \rightarrow \mathbb{R}^n$
- Minimize:
  \[ E^{(zero)}(f) = \int_{\Omega} f(x)^2 \, dx \]
- Means: the function values should not become too large
- This is often useful to include to avoid numerical problems.
  - If you have a SPD quadratic functional and add $\lambda E^{(zero)}$ the smallest eigenvalue of the discretization matrix of cannot become smaller than $\lambda$ ($\rightarrow$ condition number).
  - The system is then always solvable.
Standard Functional #2: Harmonic energy

- Given a function \( f : \mathbb{R}^m \supset \Omega \rightarrow \mathbb{R}^n \)
- Minimize:
  \[
  E^{(\text{harmonic})}(f) = \int_{\Omega} (\nabla f(\mathbf{x}))^2 \, d\mathbf{x}
  \]

- Objective: make the differences to neighboring points as small as possible
- This energy appears all the time in physics & engineering.
  - But not really what we want for smooth curves...
Harmonic Energy

Example: Heat equation

• Given a metal plate

• Hard constraints:
  ▪ A heat source
  ▪ A heat sink

• What is the final heat distribution?
  ▪ Heat flow tends to equalize temperature.
    – Stronger heat flow for larger temperature gradients.
  ▪ Gradients become as small as possible.
Harmonic Energy

Example: Harmonic energy

- Curves that minimize the harmonic energy:
  - Shortest path, a.k.a. polygons

- Two-dimensional parametric surface:

- Useful in parametrization (conformal mappings are harmonic)
**Standard Functional #3:** Thin plate spline energy

- Given a function \( f: \mathbb{R}^m \supset \Omega \rightarrow \mathbb{R}^n \)
- Minimize:

\[
E^{(TSS)}(f) = \int_\Omega \sum_{i=1}^{m} \sum_{j=1}^{m} \left( \left\| \frac{\partial^2 f(x)}{\partial x_i \partial x_j} \right\|^2 \right) dx
\]

- Objective: minimize integral second derivatives (approx. curvature)
- This energy is used all over the place in geometric modeling and geometry processing
  - Yields smooth curves & surfaces
  - A true curvature based energy is rarely used (non-quadratic)
Energies for Vector Fields

Vector fields:

- The following energies are useful for mappings from \( \mathbb{R}^n \rightarrow \mathbb{R}^n \) (e.g.: space deformations).
- Think of an object moving (over time).
- \( f(x) \) describes its deformation.
- \( f(x,t) \) describes its motion over time.

\[
\Omega \subset \mathbb{R}^n \quad \xrightarrow{f} \quad f(\Omega) \subset \mathbb{R}^n
\]
Functionals

Standard Functional #4: Green’s deformation tensor

- Given a function $f: \mathbb{R}^n \supset \Omega \rightarrow \mathbb{R}^n$
- Minimize:

$$E^{(\text{deform})}(f) = \int_{\Omega} \left\| M \left[ \nabla f^T \nabla f - I \right] \right\|_F^2 \, dx$$

- Objective: minimize metric distortion (non-identity first fundamental form)

- Basis for physically-based deformation modeling:
  - The energy is invariant under rigid transformations.
  - Bending, scaling, shearing is penalized.
  - This energy is non-quadratic (non-linear optimization required).
  - Matrix $M$ encodes material properties (often $M = I$).
Functionals

Standard Functional #5: Volume preservation

- Given a function \( \mathbf{f} : \mathbb{R}^n \supset \Omega \rightarrow \mathbb{R}^n \)
- Minimize:

\[
E^{(volume)}(\mathbf{f}) = \int_{\Omega} \left[ \det(\nabla \mathbf{f}) - 1 \right]^2 d\mathbf{x}
\]

- Objective: minimize local volume changes
- This energy tries to preserve the volume at any point.
  - Physics: Incompressible materials (for example fluids)
  - The energy is invariant under rigid transformations.
  - This energy is non-quadratic (non-linear optimization required).
  - Often used in conjunction with deformation models.
Standard Functional #6: Infinitesimal volume preservation

- Given a function $\mathbf{v}: \mathbb{R}^n \ni \Omega \rightarrow \mathbb{R}^n$
- Minimize:

$$E^{(volume)}(\mathbf{v}) = \int_{\Omega} (\text{div} \mathbf{v}(\mathbf{x}))^2 \, d\mathbf{x} = \int_{\Omega} \left( \frac{\partial}{\partial x_1} \mathbf{v}(\mathbf{x}) + \cdots + \frac{\partial}{\partial x_n} \mathbf{v}(\mathbf{x}) \right)^2 \, d\mathbf{x}$$

- Objective: minimize local volume changes in a velocity field
- Difference to the previous case:
  - The vectors are instantaneous motions ($\mathbf{v}(\mathbf{x}) = d/dt \mathbf{f}(\mathbf{x},t)$)
  - A divergence free (time dependent) vector field will not introduce volume changes
  - This functional is linear, but does not work for large (rotational) displacements.
Functionals

Standard Functionals #7 & #8: Velocity & acceleration

- Given a function $\mathbf{v}: (\mathbb{R}^n \times \mathbb{R}) \ni \Omega \rightarrow \mathbb{R}^n$
- Minimize:
  
  $$E^{(\text{velocity})}(\mathbf{f}) = \iint_{\Omega} \left( \frac{d}{dt} \mathbf{f}(\mathbf{x},t) \right)^2 \, d\mathbf{x} \, dt, \quad E^{(\text{acc})}(\mathbf{f}) = \iint_{\Omega} \left( \frac{d^2}{dt^2} \mathbf{f}(\mathbf{x},t) \right)^2 \, d\mathbf{x} \, dt$$
- Objective: minimize velocity / acceleration
- Models air resistance, inertia.
Soft Constraints
Soft Constraints

Penalty functions
- Uniform
- General quadrics
- Differential constraints

Types of soft constraints
- Point-wise constraints
- Line / area constraints

Constraint functions
- Least-squares
- M-estimators
Uniform Soft Constraints

Uniform, point-wise soft constraints:

- Given a function \( f: \mathbb{R}^n \supset \Omega \rightarrow \mathbb{R}^n \)
- Minimize:

\[
E^{(\text{constr})}(f) = \sum_{i=1}^{n} q_i (f(x_i) - y_i)^2
\]

- Constraint weights (certainty)
- Prescribed values \((x, y)_i\)
Uniform Soft Constraints

General quadratic, point-wise soft constraints:

• Given a function \( f: \mathbb{R}^n \supset \Omega \rightarrow \mathbb{R}^n \)

• Minimize:

\[
E^{(\text{constr})}(f) = \sum_{i=1}^{n} (f(x_i) - y_i)^T Q_i (f(x_i) - y_i)
\]

constraint weights (general quadratic form, non-negative)

prescribed values \((x,y)_i\)
Uniform Soft Constraints

Differential constraints:

- Given a function $f: \mathbb{R}^n \supset \Omega \rightarrow \mathbb{R}^n$
- Minimize:
  $$E^{(constr)}(f) = \sum_{i=1}^{n} (Df(x_i) - (Dy)_i)^T Q_i (Df(x_i) - (Dy)_i)$$

  - constraint weights (general quadratic form, non-negative)
  - prescribed values $(x, Dy)_i$

Differential operator: $D = \begin{pmatrix} \frac{\partial}{\partial x_{i_1,1}} & \cdots & \frac{\partial}{\partial x_{i_k,1}} \\ \vdots \\ \frac{\partial}{\partial x_{i_1,m}} & \cdots & \frac{\partial}{\partial x_{i_{km},m}} \end{pmatrix}$

This is still a quadratic constraints ($\rightarrow$ linear system).
Examples

Examples of differential constraints:

- Prescribe normal orientation of a surface
  \[ f : \mathbb{R}^2 \rightarrow \mathbb{R}^3, \quad E^{(\text{constr})}(f) = q \left( \begin{pmatrix} -\partial_u \\ -\partial_v \\ 1 \end{pmatrix} f - n \right)^2 \]

- Prescribe rotation of a deformation field
  \[ f : \mathbb{R}^3 \rightarrow \mathbb{R}^3, \quad E^{(\text{constr})}(f) = q \| \nabla f - R \|^2_F \]

- Prescribe velocity or acceleration of a particle trajectory
  \[ f : \mathbb{R}^3 \times \mathbb{R} \rightarrow \mathbb{R}^3, \quad f(x,t) = \text{pos}, \quad E^{(\text{constr})}(f) = q(x,t) \left( \ddot{f}(x,t) - a(x,t) \right)^2 \]
Line / Area Soft Constraints

Line and area constraints:

• Given a function \( f: \mathbb{R}^n \supset \Omega \rightarrow \mathbb{R}^n \)

• Minimize:

\[
E^{(\text{constr})}(f) = \int_{A \subseteq \Omega} (f(x) - y(x))^T Q(x)(f(x) - y(x))
\]

quadric error weights (may be position dependent)

prescribed values \( y(x) \) (function of position \( x \))

area \( A \subseteq \Omega \) on which the constraint is placed (line, area, volume...)

• A.k.a: “Transfinite Constraints”
Constraint Functions

Constraint Functions:

- Typically, we use quadratic constraints \( E(x) = f(x)^2 \)
  - They are easy to optimize (linear system)
  - They have a well-defined critical point (gradient vanishes)
  - However, they are very sensitive to noise
- If constraints come from measured data (e.g. 3D scanner data) instead of user interaction, quadratic constraints may cause trouble
- Alternatives:
  - \( L_1 \)-norm constraints \( E(x) = |f(x)| \) – more robust and still convex, i.e. can be optimized
  - Non-convex truncated constraints: yet more robust, but finding a global optimum can be problematic (c.f. least-squares chapter)
Discretization
Finite Element Discretization

Finite-element discretization:

- Choose a finite dimensional function space spanned by basis functions
- Compute optimum in that space only
- Finite differences (FD) is a special case for a grid of piecewise constant basis functions
- General approach:

\[
\arg\min_f E(f) \rightarrow \arg\min_\lambda E(\tilde{f}_\lambda)
\]

\[
\tilde{f}_\lambda(x) = \sum_{i=1}^{k} \lambda_i b_i(x)
\]
Finite Element Discretization

Derive a discrete equation:

• Just plug in the discrete $\tilde{f}$.
• Then minimize the it over the $\lambda$.
• For a differentiable energy function, we compute the critical point(s):

$$E(\tilde{f}_\lambda(x)) \rightarrow \min$$

$$\Rightarrow \forall i = 1...k : \frac{\partial}{\partial \lambda_i} E(\tilde{f}_\lambda(x)) = 0$$

• For quadratic functionals, this leads to a linear system.
• For non-linear functionals, we can apply Newton-optimization.
Example

(Abstract) example:

- We minimize the square integral of a differential operator.
- We have quadratic differential constraints.
- Then we obtain a quadratic optimization problem in the coefficients:
Example

(Abstract) example (cont):

\[
E(f) = \int_{\Omega} \left( D^{(1)} f(x) \right)^2 \, dx + \mu \sum_{i=1}^{n} \left( D^{(2)} f(x_i) - y_i \right)^2
\]

\[
\tilde{f}_\lambda(x) = \sum_{i=1}^{k} \lambda_i b_i(x)
\]

\[
E(\tilde{f}_\lambda) = \int_{\Omega} \left( D^{(1)} \sum_{i=1}^{k} \lambda_i b_i(x) \right)^2 \, dx + \mu \sum_{i=1}^{n} \left( D^{(2)} \sum_{i=1}^{k} \lambda_i b_i(x) - y_i \right)^2
\]

\[
= \int_{\Omega} \left( \sum_{i=1}^{k} \lambda_i \left[ D^{(1)} b_i \right](x) \right)^2 \, dx + \mu \sum_{i=1}^{n} \left( \sum_{i=1}^{k} \lambda_i D^{(2)} b_i(x) - y_i \right)^2
\]

\[
= \sum_{i=1}^{k} \sum_{j=1}^{k} \lambda_i \lambda_j \int_{\Omega} \left[ D^{(1)} b_i \right](x) \left[ D^{(1)} b_j \right](x) \, dx + \mu \sum_{i=1}^{n} \left( \sum_{i=1}^{k} \lambda_i D^{(2)} b_i(x) - y_i \right)^2
\]
Numerical Aspects
How to solve the problems?

Solving the discretized variational problem:

• Quadratic energy and quadratic constraints:
  ▪ The discretization is a quadratic function as well.
  ▪ The gradient is a linear expression.
  ▪ The matrix in this expression is symmetric.
  ▪ If the problem is well-defined, the matrix is semi-positive definite.
  ▪ It is usually very sparse (coefficients of basis functions only interact with their neighbors, as far as their support overlaps).
  ▪ We can use iterative sparse system solvers:
    – Most frequently used: conjugate gradients (needs SPD matrix). CG is available in GeoX.
How to solve the problems?

Solving the discretized variational problem:

- Non linear energy functions:
  - If the function is convex, we can get to a critical point that is the global minimum.
  - In general, we can only find a local optimum (or critical point).
  - Frequently used techniques are:
    - Newton optimization: Iteratively compute 2nd order Taylor expansions (Hessian matrix, gradient) and solve linear problems. Typically, Hessian matrices are sparse. Use conjugate gradients to solve for critical points.
    - Non-linear conjugate gradients with line search (faster than simple gradient decent).
    - In any case, we need a good initialization.
Hard Constraints
Hard Constraints:

- Sometimes, we want some properties of the solution to be met \textit{exactly} rather than \textit{approximately}.
  - Interpolation vs. approximation
  - Includes complex constraints (area constraints, differential properties etc.)

- Three options to implement hard constraints:
  - Strong soft constraints (easy, but not exact)
  - Variable elimination (exact, but limited)
  - Lagrange multipliers (most complex method)
Hard Soft Constraints

Simplest Implementation:

- Use soft constraints with a large weight

\[ E(f) = E^{\text{regularizer}}(f) + \lambda E^{\text{constraints}}(f), \text{ with } \lambda \text{ very large (say } 10^6 \text{)} \]

- This is simple to implement. But there are a few serious problems:
  - The technique is not exact (for some applications this might be not acceptable).
  - The stronger the constraints, the larger the weight. This means:
    - The condition number of the quadric matrix (condition of the Hessian in the non-linear case) becomes worse.
    - At some point, no solution is possible anymore.
    - Iterative solvers are slowed down (e.g. conjugate gradients)
**Variable Elimination**

**Idea:** Variable elimination

- We just replace variables by fixed numbers.
- Then solve the remaining system.

**Example:**

\[
f'(x_0) = h^{-1}(y_1 - 4.0)
\]

\[
f'(x_3) = h^{-1}(y_4 - y_3)
\]
Variable Elimination

Advantages:

• Exact constraints
• Conceptually simple

Problems:

• Only works for simple constraints (variable = value)
• Need to augment system (not so easy to implement generically)
• Does not work for FE methods (general basis functions)
  ▪ Values at any point are \textit{a sum} of scaled basis functions
• Does not work for complex constraints (area/integral constraints, differential constraints etc.)
Most general technique: Lagrange multipliers

- This method works for complex, composite constraints
- No problems with general basis functions (not restricted to finite difference discretizations)
- The technique is exact.
Lagrange Multipliers

Here is the idea:

• Assume we want to optimize $E(x_1, \ldots, x_n)$ subject to an implicitly formulated constraint $g(x_1, \ldots, x_n) = 0$.

• This looks like this:

\[ \nabla E = \lambda \nabla g, \quad g(x) = 0 \]
Lagrange Multipliers

Formally:

- Optimize $E(x_1, \ldots, x_n)$ subject to $g(x_1, \ldots, x_n) = 0$.
- Formally, we want:
  \[ \nabla E(x) = \lambda \nabla g(x) \text{ and } g(x) = 0 \]
- We get a local optimum for:
  \[ LG(x) = E(x) + \lambda g(x) \]
  \[ \nabla_{x,\lambda} LG(x) = 0 \]
  i.e. \[ \left( \partial_{x_1}, \ldots, \partial_{x_n}, \partial_{\lambda} \right) LG(x) = 0 \]
- A critical point of this equation satisfies both $\nabla E(x) = \lambda \nabla g(x)$ and $g(x) = 0$. 

\[ \nabla E = \lambda \nabla g \]
Example

Example: Optimizing a quadric subject to a linear equality constraint

• We want to optimize: $E(x) = x^T Ax + bx$
• Subject to: $g(x) = mx + n = 0$

We obtain:

• $LG(x) = E(x) + \lambda g(x) = x^T Ax + bx + \lambda(mx + n)$
  \[ \nabla_x (LG(x)) = 2Ax + b + \lambda m \]
  \[ \nabla_\lambda (LG(x)) = mx + n \]
• Linear system: \[
\begin{pmatrix}
2A & m \\
m & 0
\end{pmatrix}
\begin{pmatrix}
x \\ \lambda
\end{pmatrix}
=
\begin{pmatrix}
-b \\
-n
\end{pmatrix}
\]
Multiple Constraints

Multiple Constraints:

- Similar idea
- Introduce multiple “Lagrange multipliers” $\lambda$.

$$E(x) \to \min$$

subject to: $\forall i = 1 \ldots k : g_i(x) = 0$

Lagrangian objective function:

$$LG(x) = E(x) + \sum_{i=1}^{k} \lambda_i g_i(x)$$

$$\nabla_{x,\lambda} LG(x) = 0$$

i.e. $\left( \partial_{x_1}, \ldots, \partial_{x_n}, \partial_{\lambda_1}, \ldots, \partial_{\lambda_k} \right) LG(x) = 0$
Multiple Constraints

Example: Linear subspace constraints

- \( E(x) = x^T Ax + bx \) subject to \( g(x) = Mx + n = 0 \)
- \( LG(x) = E(x) + \sum_{i=1}^{n} \lambda_i g_i(x) = x^T Ax + bx + \sum_{i=1}^{n} \lambda_i (m_i x + n_i) \)

- Linear system: \( \begin{bmatrix} 2A & M^T \\ M & 0 \end{bmatrix} \begin{bmatrix} x \\ \lambda \end{bmatrix} = \begin{bmatrix} -b \\ -n \end{bmatrix} \)

- Remark: \( M \) must have full rank for this to work.
What can we do with this?

Multiple linear equality constraints:

- Constraint multiple function values, differential properties, integral values
- Area constraints: Sample at each basis function of the discretization and prescribe a value
- Need to take care:
  - We need to make sure that the constraints are linearly independent at any time

Inequality constraints:

- There are efficient quadratic programming algorithms. (Idea: turn on and off the constraints intelligently.)
The Euler Lagrange Equation
(some more math)
The Euler-Lagrange Equation

Theoretical Result:

- An integral energy minimization problem can be reduced to a differential equation.
- We look at energy functions of a specific form:

\[ f : [a, b] \to \mathbb{R} \]

\[ E(f) = \int_a^b F(x, f(x), f'(x))\,dx \]

- \( f \) is the unknown function
- \( F \) is the energy at each point \( x \) to be integrated
- \( F \) depends (at most) on the position \( x \), the function value \( f(x) \) and the first derivative \( f'(x) \).
The Euler-Lagrange Equation

Now we look for a minimum:

- Necessary condition:
  \[ \frac{d}{df} E(f) = 0 \] (critical point)
- In order to compute this:
  - Approximate \( f \) by a polygon (finite difference approximation)
  - \( f \approx ((x_1, y_1), ..., (x_n, y_n)) \)
  - Equally spaced: \( x_i - x_{i-1} = h \)

(Can be formalized more precisely using *functional derivatives*)
The Euler-Lagrange Equation

Minimum condition:

\[ E(f) = \int_{a}^{b} F(x, f(x), f'(x)) \, dx \]

\[ E(f) \approx \tilde{E}(y) = \sum_{i=2}^{n} F\left(x_i, y_i, \frac{y_i - y_{i-1}}{h}\right) \]

\[ \nabla_y \tilde{E} = \left( \partial_{y_1}, \ldots, \partial_{y_n} \right) \tilde{E} \]

\[ = \sum_{i=2}^{n} \nabla_y F\left(x_i, y_i, \frac{y_i - y_{i-1}}{h}\right) \]

\[ = \sum_{i=2}^{n} \partial_2 F\left(x_i, y_i, \frac{y_i - y_{i-1}}{h}\right) \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ h \\ \vdots \\ 1 \end{pmatrix} + \partial_3 \frac{1}{h} F\left(x_i, y_i, \frac{y_i - y_{i-1}}{h}\right) \begin{pmatrix} 0 \\ -1 \\ \vdots \\ 0 \\ 1 \end{pmatrix} \]
The Euler-Lagrange Equation

Minimum condition:

\[ \nabla_y \tilde{E} = \sum_{i=2}^{n} \left[ \partial_2 F \left( x_i, y_i, \frac{y_i - y_{i-1}}{h} \right) \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} + \partial_3 \frac{1}{h} F \left( x_i, y_i, \frac{y_i - y_{i-1}}{h} \right) \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \right] \]

ith entry:

\[ \partial_{y_i} \tilde{E} = \partial_2 F \left( x_i, y_i, \frac{y_i - y_{i-1}}{h} \right) - \frac{1}{h} \left( \partial_3 F \left( x_i, y_i, \frac{y_{i+1} - y_i}{h} \right) - \partial_3 F \left( x_i, y_i, \frac{y_i - y_{i-1}}{h} \right) \right) \]

Letting \( h \to 0 \), we obtain the continuous Euler-Lagrange differential equation:

\[ \partial_2 F(x, f(x), f'(x)) - \frac{d}{dx} \partial_3 F(x, f(x), f'(x)) = 0 \]
The Euler-Lagrange Equation

\[ \partial_2 F(x, f(x), f'(x)) - \frac{d}{dx} \partial_3 F(x, f(x), f'(x)) = 0 \]

(at every point \( x \))
Example: Harmonic Energy

\[ E(f) = \int_{a}^{b} \left( \frac{d}{dx} f(x) \right)^2 dx \]

\[ F(x, f(x), f'(x)) = f'(x)^2 \]

\[ \partial_2 F(x, f(x), f'(x)) - \frac{d}{dx} \partial_3 F(x, f(x), f'(x)) = 0 \]

\[ \Leftrightarrow 0 - \frac{d}{dx} \partial_{f(x)} f'(x)^2 = 0 \]

\[ \Leftrightarrow 0 - \frac{d}{dx} 2 \frac{d}{dx} f(x) = 0 \]

\[ \Leftrightarrow \frac{d^2}{dx^2} f(x) = 0 \]
Generalizations

Multi-dimensional version:

$$f : \mathbb{R}^d \supseteq \Omega \rightarrow \mathbb{R}$$

$$E(f) = \int_{\Omega} F(x_1, ..., x_d, f(x), \partial_{x_1} f(x), ..., \partial_{x_d} f(x)) \, dx_1 ... dx_d$$

Necessary condition for extremum:

$$\frac{\partial E}{\partial f(x)} - \sum_{i=1}^{d} \frac{d}{dx_i} \frac{\partial E}{\partial f_{x_i}} = 0$$

$$f_{x_i} := \frac{\partial}{\partial x_i} f(x)$$

This is a partial differential equation (PDE).
Example: General Harmonic energy

\[ E^{(\text{harmonic})}(f) = \int_\Omega (\nabla f(x))^2 \, dx \]

Euler Lagrange equation:

\[ \Delta f(x) = \left( \frac{\partial^2}{\partial x_1^2} f(x) + \ldots + \frac{\partial^2}{\partial x_d^2} f(x) \right) = 0 \]
Summary

Euler Lagrange Equation:

- Converts integral minimization problem into ODE or PDE.
- Gives a necessary, but not sufficient condition for extremum (critical “point”, read: function $f$).
- Application:
  - From a numerical point of view, this does not buy us much.
    - We can usually directly optimize the integral expression.
    - Similarly complex to compute (boundary value problem for a PDE vs. variational problem).
  - Analytical tool
    - Helps understanding the minimizer functions.
Surface Modeling
Applications

Variational Surface Modeling:

Two Examples:

- Parametric surfaces

- Implicit surfaces
Parametric Surfaces

Domain:

- Parametric patch: \( f: [0,1]^2 \rightarrow \mathbb{R}^3 \).
- Representation (discretization):
  - Grid of uniform tensor-product B-Splines
  - Refine by dilated functions (subdivision) until convergence
- Energy:
  - Thin-plate-spline energy
- Constraints:
  - Points (soft / hard, langrange multipliers)
  - Transfinite constraints (curves, soft constraints only)
- Numerics:
  - Quadratic objective \( \rightarrow \) solver sparse linear system
Implicit Surface

Domain:

- Implicit function: \( f: [0,1]^3 \to \mathbb{R} \).
- Representation (discretization):
  - Radial basis functions of fundamental solutions
- Energy:
  - Thin-plate-spline energy
- Constraints:
  - Points with normals (hard, variable elimination)
- Numerics:
  - Radial basis functions around points and \( \pm \) normal
  - Solve linear system for interpolation problem
  - Energy implicitly encoded in fundamental solutions
Other Applications
Variational Animation Modeling

\[ f(x, t) \] – deformation field

\[ S \] – point on urshape

\[ f(x, t) \] – deformation field
Variational Framework

\[ E(f) = E_{match}(f) + (E_{rigid} + E_{volume} + E_{accel} + E_{velocity})(f) \]

\[ E_{match}(f) = \sum_{t=1}^{T} \sum_{i=1}^{n_t} dist(d_i, f(S))^2 \]

\[ E_{rigid}(f) = \int_{V(S)} \omega_{rigid}(x) \left\| \nabla_x f(x,t)^T \nabla_x f(x,t) - I \right\|_F^2 dx \]

\[ E_{volume}(f) = \int_{V(S)} \omega_{vol}(x) \left( \left| \nabla_x f(x,t) \right| - 1 \right)^2 dx \]

\[ E_{accel}(f) = \int_{S} \omega_{acc}(x) \left( \frac{\partial^2}{\partial t^2} f(x,t) \right)^2 dx \]

\[ E_{velocity}(f) = \int_{S} \omega_{velocity}(x) \left( \frac{\partial}{\partial t} f(x,t) \right)^2 dx \]
Data Set: "Popcorn Tin"

94 frames
data: 53K points/frame
rec: 25K points/frame