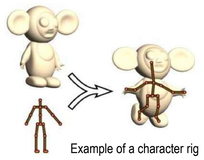
Parameter Estimation for Constitutive Models of Deformable Objects

Reema Al-Aifari, Laura Bufford, Linda Farczadi, Brittan Farmer

Institute for Pure and Applied Mathematics, UCLA

Introduction

In computer animation, characters are deformable objects that can be controlled by animators through rigs. Physical simulations are used as a supplementary tool. Our goal is to fit parameters of a given constitutive model to a rigged character so that the resulting physical simulation matches the behavior of the rig.



Example of a character rig

Specifically, we solved an inverse problem to fit material parameters to hand-animated target data.

This project presents a solution to this problem using two different approaches: non-linear least squares fitting, implemented using the Gauss-Newton algorithm, and TV regularization, implemented using the Split Bregman algorithm.

Problem Statement

The general problem can be stated as: given target data x^t and a parametric constitutive model, determine the set of material and other environmental parameters λ which minimize the energy functional

$$E(\lambda) = \min_{\lambda} \|x(\lambda) - x^t\|^2. \qquad (1)$$

In our case the target data x^t consists of a vector of positions in 1-D or higher dimensions and x(λ) can be obtained by solving a system of partial differential equations. The minimization problem can be solved using numerical methods.

Non-Linear Least Squares Approach

The minimization problem (1) is a non-linear least squares problem. In order to solve it we can use a method known as the Gauss-Newton algorithm. First, the minimization functional is approximated using a quadratic approximation

$$E(\lambda + d\lambda) \approx ||x(\lambda) + \frac{\partial x(\lambda)}{\partial \lambda} d\lambda - x^t||^2.$$

Then this approximation can be minimized using an iterative optimization method. Starting from an initial guess λ_0 the solution of the (k+1)-th step is

$$\lambda_{k+1} = \lambda_k + d\lambda,$$

where dλ is the least squares solution of

$$-\frac{\partial x(\lambda_k)}{\partial \lambda} d\lambda = x(\lambda_k) - x^t.$$

This is equivalent to solving the normal equation

$$J_k^T J_k d\lambda = J_k (x(\lambda_k) - x^t),$$

where $J_k = \frac{\partial x(\lambda_k)}{\partial \lambda}$ denotes the Jacobian for the k-th iteration.

Total Variation Minimization Approach

TV regularization tackles the problem of ill-posedness encountered with the Gauss-Newton approach by adding a regularization term to the minimization problem.

One choice is the total variation (TV) which is given by

$$TV(\lambda) = \int_\Omega |\nabla \lambda(X)| \, dX.$$

The minimization functional is then given by

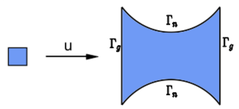
$$\begin{aligned} E(\lambda) &= TV(\lambda) + \frac{\mu}{2}\|x(\lambda) - x^t\|^2 \\ &= \int_\Omega |\nabla \lambda| dX + \frac{\mu}{2}\|x(\lambda) - x^t\|^2, \end{aligned}$$

where μ is a tuning parameter that takes a predetermined value. TV regularization ensures that our optimal parameters will be piecewise constant. In computer animation, characters are often composed of discontinuous sections such as skin, muscle or bone, hence piecewise constant parameters are desirable.

The regularization complicates the minimization problem. The Split Bregman algorithm allows us to decouple the L^1 and L^2 portions of our equation for E(λ). This reduces the L^1 regularization problem to a series of unconstrained optimization problems performed in each iterative step.

The 2-D Problem

In order to describe the 2-D problem, we first considered a square with side length s < 1 and a function k that gives the distribution of the stiffness of this square. We stretch the square such that the left and right boundaries are the left and right edges of the unit square. The lower and upper boundaries are set to be free. If we begin with a square with side length 0 *(zero rest length area)*, the deformation u can be described by the Poisson equation



$$\nabla \cdot [k(X)\nabla(u(X))] = 0 \ \forall X \in \Omega.$$

Including the boundary conditions, the complete PDE system is given by

$$\begin{aligned} \nabla \cdot [k(\mathbf{X})\nabla(u(\mathbf{X}))] &= 0 && \forall\, \mathbf{X} \in \Omega, \\ u(\mathbf{X}) &= g(\mathbf{X}) && \forall\, \mathbf{X} \in \Gamma_g, \\ k(\mathbf{X})\frac{\partial u}{\partial X_2}(\mathbf{X}) &= 0 && \forall\, \mathbf{X} = (X_1, X_2) \in \Gamma_n, \end{aligned}$$

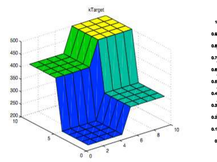
for a given function g.

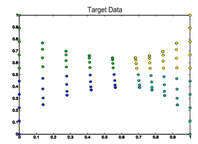
This system can be solved numerically by using the finite difference method. Therefore, we discretize Ω by applying a uniform grid with N^2 grid points. Replacing the derivatives by finite differences and incorporating both Dirichlet and Neumann boundary conditions yields a linear system of equations

$$\begin{aligned} A\underline{v} &= f_1, \\ A\underline{w} &= f_2. \end{aligned}$$

The matrix A is sparse and the vectors v and w correspond to the discretized solutions. The algorithm for solving the forward problem is needed to generate target data.

The inverse problem can be solved with either Gauss-Newton or Split Bregman. We applied both methods to fitting target data generated by piecewise constant parameter functions. We ran those tests to compare the two methods and to use this information for the 3-D problem.



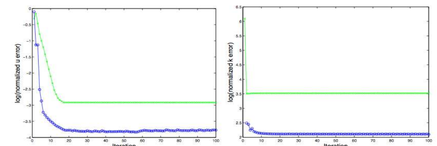


Left: A piecewise constant k distribution. Right: The corresponding spatial configuration.

Here, the normal equation in each step of Gauss-Newton was solved with the CG method which was stopped according to the discrepancy principle. This is necessary because the equation is ill-conditioned.

Gauss-Newton yielded faster convergence but Split Bregman resulted in better approximations of the piecewise constant parameter distribution which is due to the TV-term.

Especially when testing for noisy target data, we noticed a high sensitivity of the TV-method to the Split Bregman parameters.



Error vs. Iterations for noisy target data

For sparse target data, both algorithms approximate the sparse data very well, but do not result in a good accuracy for the dense data.

	Gauss-Newton	Split Bregman
$\|u_{error}\|_\infty$	0.0527	0.0342
$\|u_{error}\|_2$	0.298	0.146
$\|k_{error}\|_\infty$	492.58	185.06
$\|k_{error}\|_2$	3545.80	841.33

Sparse target data

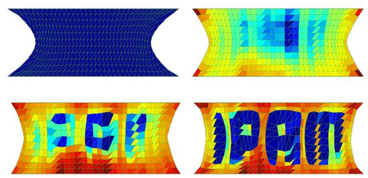
PhysBAM Software for Tests in 2-D

PhysBAM is a C++ library used by both Pixar and Disney for physical simulations, so it is a natural choice for implementing our parameter estimation techniques. This software utilizes the object-oriented aspects of C++, defining data structures as classes, which are created with constructor functions.

PhysBAM performs simulations using the theory of continuum mechanics.

We first implemented a non-linear least squares parameter estimation. In order to solve our minimization problem, we used the Levenberg-Marquardt algorithm which interpolates between the Gauss-Newton algorithm and the method of gradient descent.

We ran different tests in 2-D, one of which was to try to set up a surface and spell out the letters IPAM by using different stiffness values for letters and the background.



Parameter estimation for IPAM pattern after 1 iteration (top left), 5 iterations (top right), 10 iterations (bottom left) and 40 iterations (bottom right).

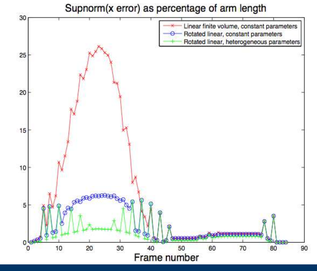
PhysBAM Software for 3-D Tests

Our sponsor provided us with test data for a 3-D animated arm. The animation shows how the arm bends inward and the bicep expands. After that, the arm returns to its undeformed state before it fully extends and the tricep expands.

Once the parameters were found by a physics simulation, we used them in a physical simulation of the arm. To animate from frame to frame, the boundary conditions were changed and the object was evolved to quasistatic equilibrium. After each frame, the particle positions were reset to the actual animated positions before simulating to the next frame. Using a linear constitutive model, the modeled arm flattened out as it bent. To fix this problem, we used a rotated linear constitutive model.

Frame 24 of the hand-animation (left) and of a physical simulation using the linear finite volume constitutive model (middle) and the rotated linear finite constitutive model (right).

The figure on the right shows the error of the different simulations compared to the original animation.



Conclusion

In exploring the inverse problem of parameter fitting we tested the Gauss-Newton and Split Bregman in one, two and three dimensions. For the 1-D spring problem we thoroughly tested both the Gauss-Newton and Split Bregman algorithms to ensure they were appropriate for this type of problem. We also looked at 2-D examples using the Poisson equation which served as a proof of concept of the Split Bregman algorithm in 2-D. Using the PhysBAM library we used the algorithms to solve the inverse problem of parameter fitting successfully.

We did this using a coarser grid for the parameter distribution so it was not necessary to solve for separate parameters for each tetrahedron, but we were still able to obtain visually pleasing results. From this it appears it is unnecessary to solve for separate parameters for every tetrahedron. This then is a way to reduce the size of the problem to make it more computationally manageable.

Acknowledgements

This research was part of the RIPS 2009 program at IPAM and was sponsored by Pixar and Walt Disney Animation Studios. We would like to thank IPAM and our sponsors, in particular our academic mentor Aleka McAdams and our sponsoring mentors Andrew Selle and John Anderson.