Siggraph 2007 course notes

Practical Least-Squares for Computer Graphics

Fred Pighin
Industrial Light and Magic

J.P. Lewis
Stanford University

Abstract. The course presents an overview of the least-squares technique and its variants. A wide range of problems in computer graphics can be solved using the least-squares technique (LS). Many graphics problems can be seen as finding the best set of parameters for a model given some data. For instance, a surface can be determined using data and smoothness penalties, a trajectory can be predicted using previous information, joint angles can be determined from end effector positions, etc. All these problems and many others can be formulated as minimizing the sum of squares of the residuals between some features in the model and the data.

Despite this apparent versatility, solving problems in the least-squares sense can produce poor results. This occurs when the nature of the problem error does not match the assumptions of the least-squares method. The course explains these assumptions and show how to circumvent some of them to apply LS to a wider range of problem.

The focus of the course is to provide a practical understanding of the techniques. Each technique will be explained using the simple example of fitting a line through data, and then illustrated through its use in one or more computer graphics papers.

Prerequisites. The attendee is expected to have had an introductory course to computer graphics and some basic knowledge in linear algebra at the level of OpenGL transforms.

Updates and Slides. The latest version of these notes and the associated slides are located at `http://graphics.stanford.edu/~jplewis/lscourse`. Please download the version from that directory – it may have fixes and other improvements.

Contents

1 Introduction

1.1 Motivation

The course presents an overview of the least-squares technique and its variants. A wide range of problems in computer graphics can be solved using the least-squares technique (LS). Many graphics problems can be seen as finding the best set of parameters for a model given some data. For instance, a surface can be determined using data and smoothness penalties, a trajectory can be predicted using previous information, joint angles can be determined from end effector positions, etc. All these problems and many others can be formulated as minimizing the sum of squares of the residuals between some features in the model and the data.

Despite this apparent versatility, solving in the least-squares sense sometime produce unwanted or disappointing results. This is often due to a misunderstanding of the assumption underlying LS. These assumptions are stated in the Gauss-Markov theorem. The course will explain these assumptions and show how to circumvent some of them to apply LS to a wider range of problems. For instance, outliers within the data can severely bias a least-squares estimate but robust techniques, such as least-median-of-squares, can give more reliable results while still minimizing a least-squares criterion. We will show that better results can often be obtained by slightly modifying the error function. We will also discuss the techniques used to minimize these errors and their computational cost.

The goal of these notes is to provide a practical understanding of LS techniques. Rather than limiting ourselves to mathematical descriptions, we provide mathematical justifications but also describe their practical domain of application and properties with examples specifically drawn from the computer graphics literature. We will show and justify the choice of technique appropriate for each class of problem.

These notes are not meant to be complete or precise presentation of the least-squares techniques. For mathematical references on least-squares technique and linear algebra, we recommend the book by Trefethen and Bau [35] as well as the one by Ake Björck [2].

Finally, in choosing the papers to illustrate the various techniques, our goal was to touch as many fields as possible (animation, modeling, image processing, etc). We could not within these notes do justice to all related papers.

1.2 History

The least-squares method was first invented and developed by three of the foremost mathematicians of the eighteenth and nineteenth centuries: Johann Carl Friedrich Gauss (1777 - 1855), Adrien Marie Legendre (1752 - 1833), and Pierre Simon Laplace (1749 - 1827).

The technique was first applied to predict the motion of comets. In 1801 an Italian astronomer, Giuseppe Piazzi, discovered an asteroid and tracked its movement for a period of 40 days. A number of scientists attempted to predict its subsequent movement based on this data. Of these, only Gauss' least-squares based calculation was precise enough to allow another astronomer to re-locate the asteroid later in the year. Gauss's basic least-squares approach had already been worked out in 1795 at the age of 18. He did not publish the least-squares method until 1809 however, in his book on celestial mechanics.

The French Adrien Marie Legendre independently developed the same method in 1805 and published it in 1806. Legendre and Gauss both claimed priority and engaged in a bitter dispute. Neither Gauss nor Legendre gave any proof of the technique, although Gauss did state that least-squares is preferred if errors are distributed "normally" (i.e., with a Gaussian distribution). Laplace first provided a proof in terms of minimizing the expectation of the error in 1812. In 1829 Gauss produced an early version of the Gauss-Markov theorem. The theorem says that the best linear unbiased estimator is the least-squares estimate (this statement will be explained below).

In the mid to late 1800's, Francis Galton used the technique to study the heritability of size. His work laid down the foundation of correlation and regression analysis.

1.3 Outline

Following this introduction, we have broken these notes into several sections.

In Section 2, we introduce the linear least-squares technique using linear regression. We also discuss the basic algebra and geometry associated with least-squares. In Section 3, we discuss the optimality of ordinary least-squares technique using estimation theory and statistics. Then Sections 4 to 8 describe the variants of the linear least-squares technique. First, in Section 4, we study different ways to adapt ordinary least-squares to more general error distributions. Then, in Section 5, we cover techniques that are robust to samples with large errors (i.e. outliers). Following, we describe several techniques for constraining the solution to some subset. In Section 4, we present various techniques for handling ill-conditioned problems. Finally, we finish this overview by describing some of the most common technique for solving non-linear least-squares problems in Section 8.

1.4 Notations

In these notes we have used the following notations:

$a_{i,j}$	scalars
$\boldsymbol{b}$	vectors
$\boldsymbol{A}$	matrices
$\tilde{f}$	approximation of f
$\hat{\boldsymbol{x}}$	estimate of $\boldsymbol{x}$

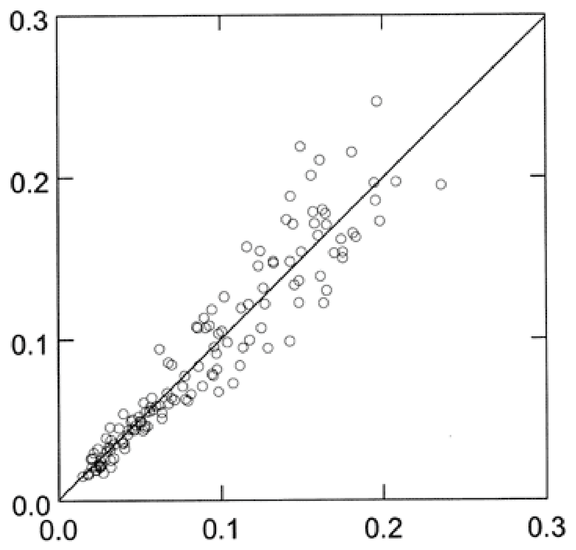

Figure 1: A one-dimensional regression. The observations are displayed as a 2-dimensional scatter plot.

2 Ordinary least-squares

2.1 Linear regression

There are many ways to justify and explain the concept of the linear least-squares solution. We'll start from one of the simplest problems, that of one-dimensional line fitting, and later move on to more general situations.

The 1-dimensional case. Imagine we are trying to find a relationship between two variables a and b (e.g. the temperature and pressure of a gas). We assume that there exists a linear relationship between the two such that:

$$b = xa.$$

That is to say we would like to predict the value of b given a value of a (perhaps because it is much easier to measure a than b). Since there is a notion of dependency between a and b, a is called the independent variable and b the dependent variable. Also, since the previous relationship might not be exact for all pairs of values, we prefer to write:

$$\hat{b} = xa,$$

where $\hat{b}$ (pronounced "b hat") is the prediction of b given a. It is a prediction in the sense that $\hat{b}$ is a view of b that depends on the (linear) model "xa" that we defined.

Now given a set of observations $\{a_i, b_i\}$ (e.g. measurements of temperature and pressure of a gas in a lab), we can estimate a linear relationship between the variables a and b. One way to look for this relationship is to require that our linear model matches (as best as possible) the observations. To do this, it seems natural to wish that the discrepancy, or *residual*, between the model and the data be as small as possible. The questions is then: how do we measure this discrepancy? The least-squares criteria measures the discrepancy by adding the squared residual at each observation, i.e.:

$$e = \sum_{i=1}^{n} (b_i - \hat{b}_i)^2 = \sum_{i=1}^{n} (b_i - xa_i)^2,$$

where n is the number of observations and $b_i - xa_i$ is the residual for observation i. Thus we are looking for the value of x that minimizes the summed squared residuals, e. This problem is often specified by the minimum of the error function:

$$\min_{a} \sum_{i=1}^{n} (b_i - xa_i)^2,$$

or the value of x that minimizes

$$\arg\min_{x} \sum_{i=1}^{n} (b_i - xa_i)^2.$$

Using the following notations:

$$\boldsymbol{b} = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix} \text{ and } \boldsymbol{a} = \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix},$$

we can rewrite the error function using linear algebra as:

$$e = (\boldsymbol{b} - x\boldsymbol{a})^t (\boldsymbol{b} - x\boldsymbol{a}) = \|\boldsymbol{b} - x\boldsymbol{a}\|_2^2,$$

where the subscript 2 refers to the Euclidian or 2-norm $\|\boldsymbol{v}\|_2 = \sqrt{\sum_i v_i^2}$.

Figure 1 illustrates the concept of a one-dimensional linear regression. The result of the regression is a line in the $a - b$ plane that goes through the origin.

The multi-dimensional case. Let us now imagine the slightly more complicated case where b depends on m independent variables $\{a_j\}$. In a similar way, we would like to find a linear relationship between b and the $\{a_j\}$ such that:

$$\hat{b} = x_1 a_1 + \ldots + x_m a_m.$$

or

$$\hat{b} = \sum_{j=1}^{m} x_j a_j.$$

As previously, if we dispose from set of observations $\{b_i, \{a_{i,j}\}\}$ we can look for the linear relationship that minimizes the sum of squared residuals:

$$e = \sum_{i=1}^{n} (b_i - \hat{b}_i)^2 = \sum_{i=1}^{n} (b_i - \sum_{j=1}^{m} x_j a_{i,j})^2.$$

We can rewrite this using matrix notation as:

$$e = \|\boldsymbol{b} - \sum_{j=1}^{m} \boldsymbol{a}_j x_j\|_2^2 = \|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}\|_2^2.$$

In the first step we replace the outer sum by the squared norm of the vector $b - \sum_{j=1}^{m} \boldsymbol{a}_j x_j$, where the vector $\boldsymbol{a}_j$ has for coordinates $a_{i,j}$. The second step replaces the inner sum by a matrix multiplication where $\boldsymbol{A}$ is the matrix whose column vectors are $\{\boldsymbol{a}_j\}$.

2.2 Geometry of ordinary least-squares and the normal equation

As discussed in the previous section, we are looking for the least squares solution; in other words we are looking for the value of $\boldsymbol{x}$ that minimizes

$$e(\boldsymbol{x}) = \|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}\|_2^2.$$

From algebra we recall that a minimum occurs when the first derivative is zero and the second derivative is positive (which is a mathematical way of saying that the function should "look like" a convex bowl at the minima). An analogous statement holds in the multidimensional case, where the first derivative of a function $f(\boldsymbol{x})$ is the gradient, $\nabla f(\boldsymbol{x})$, a row vector, and the second derivative, the Hessian, is a matrix that we will denote as $H_f(\boldsymbol{x})$.

A simple statement of a multidimensional minimum, at $\boldsymbol{x}$, is then

$$\nabla f(\boldsymbol{x}) = 0 \text{ and } H_f(\boldsymbol{x}) \text{ is positive semidefinite.}$$

To discuss this, let us find the second order derivative of the error function:

$$e(\boldsymbol{x}) = \|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}\|_2^2 = (\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x})^T(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}) = \boldsymbol{x}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} - \boldsymbol{x}^T\boldsymbol{A}^T\boldsymbol{b} - \boldsymbol{b}^T\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}^T\boldsymbol{b}.$$

Differentiating[1] with respect to x gives

$$\nabla e(\boldsymbol{x}) = 2\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} - 2\boldsymbol{A}^T\boldsymbol{b}.$$

[1]We here use the matrix differentiation formulas: $\frac{d}{dp}(\boldsymbol{M}\boldsymbol{p}) = \boldsymbol{M}$, $\frac{d}{d\boldsymbol{p}}(\boldsymbol{p}T\boldsymbol{M}\boldsymbol{p}) = \boldsymbol{p}^T(\boldsymbol{M}^T + \boldsymbol{M})$, and $\frac{d}{d\boldsymbol{p}}(\boldsymbol{p}^T\boldsymbol{q}) = \frac{d}{d\boldsymbol{p}}(\boldsymbol{q}^T\boldsymbol{p}) = \boldsymbol{q}^T$.

And differentiating once more gives

$$H_e(\boldsymbol{x}) = 2\boldsymbol{A}^T\boldsymbol{A}.$$

Let us examine the second property first. For a matrix $\boldsymbol{M}$ to be positive semi-definite the product $\boldsymbol{x}^T\boldsymbol{M}\boldsymbol{x}$ needs to be non-negative for all $\boldsymbol{x}$. By construction the product $\boldsymbol{x}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x}$ involving the matrix $\boldsymbol{M} = \boldsymbol{A}^T\boldsymbol{A}$ is always non-negative. To see this notice that $x^T\boldsymbol{A}^T\boldsymbol{A}x = (\boldsymbol{A}x)^2$ and this is just $\|\boldsymbol{A}x\|_2^2$, which is a length, which is non-negative by definition.

The first condition above yields the equation

$$\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^T\boldsymbol{b}.$$

This is the *normal equation*. Much of what can be done at this stage depends on the properties of the matrix $\boldsymbol{A}^T\boldsymbol{A}$. If $\boldsymbol{A}^T\boldsymbol{A}$ is invertible (or non-singular) then the normal equation gives a single solution. If however it is not invertible then there is a subspace of solutions. A square matrix is invertible if it has full rank, in other words its column and row vectors form linearly independent systems of vectors. For a square matrix like $\boldsymbol{A}^T\boldsymbol{A}$, we have the following relationships:

$$\text{rank }(\boldsymbol{A}^T\boldsymbol{A}) = \text{rank }(\boldsymbol{A}) \leq min(n,m).$$

In case of a regression, n is the number of independent variables and m is the number of observations. For $\boldsymbol{A}^T\boldsymbol{A}$ to have full rank (i.e. rank $(\boldsymbol{A}^T\boldsymbol{A}) = n$), requires having a number of independent (in the linear sense) observations that is greater or equal than the number of independent variables. This just tells us that we need at least as many observations as there are variables describing the observed system or, from a different perspective, that we need at least as many constraints as there are degrees of freedom in the model.

Let us examine these two scenarios in more detail.

$\boldsymbol{A}^T\boldsymbol{A}$ is non-singular. In this case the least-squares solution is unique and is $\boldsymbol{x} = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{b}$ (the matrix $(\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T$ is called a pseudo-inverse of $\boldsymbol{A}$).

As we noted before the solution $\boldsymbol{x}$ are the coordinates of $\boldsymbol{b}$ in the column space of $\boldsymbol{A}$. In the original basis the coordinates of this vector are $\boldsymbol{A}(\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{b}$. The matrix $\boldsymbol{P} = \boldsymbol{A}(\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T$ is important. There are two notable things about this matrix, first it is symmetric (i.e. $\boldsymbol{P}^T = \boldsymbol{P}$), second it is idempotent (i.e. $\boldsymbol{P}\boldsymbol{P} = \boldsymbol{P}$). In linear algebra idempotent matrices represent projection operators, and when the matrix is also symmetric it represents an orthogonal projection. Given our previous discussion, it is clear that $\boldsymbol{P}$ represents nothing other than the orthogonal projector onto the column space of $\boldsymbol{A}$. Another way to see this is to notice that the projection $\boldsymbol{P}\boldsymbol{b}$ $(= \boldsymbol{A}\boldsymbol{x})$ is orthogonal to the residual $\boldsymbol{r} = \boldsymbol{b} - \boldsymbol{P}\boldsymbol{b}$ for all $\boldsymbol{x}$.

Figure 2 illustrates this least-squares geometry. The least-squares projector $\boldsymbol{P}$ decomposes $\boldsymbol{b}$ into two orthogonal vectors $\boldsymbol{P}\boldsymbol{b}$ and $\boldsymbol{b} - \boldsymbol{P}\boldsymbol{b}$.

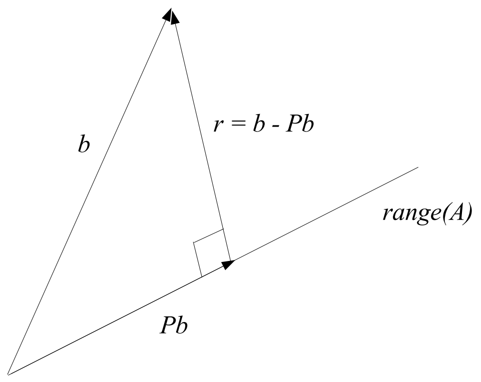

Figure 2: The least-squares geometry. The least-squares solution is the coordinates of the projection of $\boldsymbol{b}$ on to the range of $\boldsymbol{A}$. The residual $\boldsymbol{r} = \boldsymbol{b} - \boldsymbol{Pb}$ is always orthogonal to the projection.

A reverse argument can be made to provide a geometric justification for the least-squares solution. We can start from constraining the residual to be orthogonal to $range(\boldsymbol{A})$ and end up with the normal equation:

$$\begin{array}{ll} & \boldsymbol{r} \text{ orthogonal to } range(\boldsymbol{A}) \\ \Leftrightarrow & \boldsymbol{a}_i^T \boldsymbol{r} = 0, \text{ for all i} \\ \Leftrightarrow & \boldsymbol{A}^T \boldsymbol{r} = 0 \\ \Leftrightarrow & \boldsymbol{A}^T (\boldsymbol{b} - \boldsymbol{Ax}) = 0 \\ \Leftrightarrow & \boldsymbol{A}^T \boldsymbol{Ax} = \boldsymbol{A}^T \boldsymbol{b} \end{array}$$

$\boldsymbol{A}^T\boldsymbol{A}$ is singular. If $\boldsymbol{A}^T\boldsymbol{A}$ is singular then there exists a “null space” of vectors $\boldsymbol{z}$ for which $\boldsymbol{A}^T\boldsymbol{Az} = 0$ (also called the kernel of $\boldsymbol{A}^T\boldsymbol{A}$). This means that for a solution $\boldsymbol{x}$, $\boldsymbol{x} + \boldsymbol{z}$ is also a solution since $\boldsymbol{A}^T\boldsymbol{A}(\boldsymbol{x}+\boldsymbol{z}) = \boldsymbol{A}^T\boldsymbol{Ax} + \boldsymbol{A}^T\boldsymbol{Az} = \boldsymbol{A}^T\boldsymbol{Ax} = \boldsymbol{b}$. We will see later that it is possible to add constraints to in order to get a unique solution.

2.3 Ordinary least-squares and system of equations

Solving linear system of equations and the ordinary least-squares method are closely related techniques. It is also important to remember that they are different.

Following our notations, a linear system of equations can be represented as:

$$\boldsymbol{Ax} = \boldsymbol{b},$$

where $\boldsymbol{A}$ (an $m \times n$ matrix) and $\boldsymbol{b}$ (an m-dimensional vector) are given and $\boldsymbol{x}$ (an n-dimensional vector) is the unknown vector. By writing $\boldsymbol{Ax} = \boldsymbol{b}$, we specify $\boldsymbol{x}$ as being the coordinates of $\boldsymbol{b}$ in the system defined by the column vectors of $\boldsymbol{A}$. To see this notice that we can rewrite it as:

$$\boldsymbol{Ax} = \begin{bmatrix} a_{1,1} & \dots & a_{1,m} \\ \vdots & & \vdots \\ a_{n,1} & \dots & a_{n,m} \end{bmatrix} \boldsymbol{x} = \begin{bmatrix} a_{1,1} \\ \vdots \\ a_{n,1} \end{bmatrix} x_1 + \ldots + \begin{bmatrix} a_{1,m} \\ \vdots \\ a_{n,m} \end{bmatrix} x_m = \sum_{j=1}^{m} \boldsymbol{a}_j x_j = \boldsymbol{b},$$

where the $\boldsymbol{a}_i$ are the column vectors of the matrix $\boldsymbol{A}$. Another way to express this is to say that the vector $\boldsymbol{Ax}$ belongs to $range(\boldsymbol{A})$, where $range(\boldsymbol{A})$ is the vector space spanned by the colum vectors of the matrix $\boldsymbol{A}$.

In the case where $\boldsymbol{A}$ is square and has full rank then $\boldsymbol{A}$ is invertible and its columns form a basis of $\Re^m$. In this case, the decomposition of $\boldsymbol{b}$ onto the column space of $\boldsymbol{A}$ is unique and is $\boldsymbol{A}^{-1}\boldsymbol{b}$. By solving the system, we just performed a change of basis.

If however $\boldsymbol{A}$ is not square or singular, we need a criteria to define how $\boldsymbol{b}$ gets decomposed (or projected) onto the column space of $\boldsymbol{A}$. If we use the least-squares criteria, $\min_{\boldsymbol{x}} \|\boldsymbol{b} - \boldsymbol{Ax}\|_2^2$, then indeed we are solving a least-squares problem. However, one could use a different criteria (e.g. $\min_{\boldsymbol{x}} \|\boldsymbol{b} - \boldsymbol{Ax}\|_1$) that would require a different technique.

2.4 Examples

2.4.1 Example: mesh reconstruction and optimization

Least-squares problems with quadratic constraint arise in the field of mesh reconstruction (i.e. construction of a mesh from a set of samples) and mesh optimization (i.e. improving a mesh for a better distribution of its vertices).

The *Least-squares meshes* technique [34] reconstructs a mesh based on a sparse set of control points and a given planar mesh. The smoothness of the reconstructed mesh at a vertex $\boldsymbol{v}_i$ is enforced by the condition:

$$d_i \boldsymbol{v}_i = \sum_{(i,j) \in E} \boldsymbol{v}_j,$$

where d_i is the valence of vertex i and E is the set of edges. We can aggregate these conditions using a matrix representation. Let us call $\boldsymbol{x}$, $\boldsymbol{y}$, and $\boldsymbol{z}$ the vectors representing the coordinates of all vertices, and define the matrix $\boldsymbol{L} = (L_{i,j})$ such that:

$$L_{i,j} = \begin{cases} 1 & \text{if } i = j \\ -\frac{1}{d_i} & \text{if } (i,j) \in E \\ 0 & \text{otherwise.} \end{cases}$$

then the smoothness constraints can be written as:

$$\boldsymbol{Lx} = \boldsymbol{0}, \boldsymbol{Ly} = \boldsymbol{0}, \boldsymbol{Lz} = \boldsymbol{0}.$$

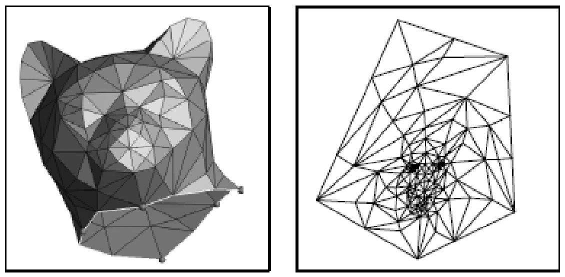
(a) (b)

Figure 3: Harmonic map computation. The mesh (b) is mapped onto a 2-d patch (b) using a least-squares computation (borrowed from [9]

.

The control vertices impose another set of constraints on the resulting mesh. If C is the set of indices of the constrained vertices and, for $s \in C$, $\boldsymbol{v}'_s = (x'_s, y'_s, z'_s)$ is the desired position of vertex $\boldsymbol{v}_s = (x_s, y_s, z_s)$ then the the x-coordinate of the vertices solution is:

$$\min_{\boldsymbol{x}} ||\boldsymbol{Lx}||^2 + \sum_{s \in C} (x_s - x'_s)^2$$

The y and z coordinates are found similarly. The above problem is a linear least-squares problem. It is sparse and its matrix has full rank.

The goal of the *Laplacian mesh optimization* [29] technique is to improve or optimize the quality of a mesh by relocating its vertices. This can be formulated in a way that is very similar to the previous problem. Additional terms can be introduced for instance to limit the motion in directions perpendicular to the mesh. Also sharp features can be preserved by reducing the weight of the Laplacian constraint of high curvature vertices.

2.4.2 Example: computation of harmonic maps

The technique of harmonic maps has found applications in computer graphics such as texture-mapping [19], remeshing [9] and surface mapping [38]. It is used to compute mapping between 3-dimensional meshes and 2-dimensional patches.

By definition an harmonic map $\phi : M \rightarrow N$ between two manifolds M and N is the minimum of the energy function:

$$E(\phi) = \int_M e(\phi) dM,$$

where $e(\phi)$ is the energy density of the map ϕ at a point. In computer graphics $E(\phi)$ is discretized as:

$$\tilde{E}(\phi) = \frac{1}{2} \sum_{(i,j) \in Edges} k_{i,j} ||\phi(\boldsymbol{v}_i) - \phi(\boldsymbol{v}_j)||^2,$$

where $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ are two vertices on a edge and the $\{k_{i,j}\}$ are spring constants defined in [9] as:

$$k_{i,j} = (L^2_{i,k_1} + L^2_{j,k_1} - L^2_{i,j})/A_{i,j,k_1} + (L^2_{i,k_2} + L^2_{j,k_2} - L^2_{i,j})/A_{i,j,k_2}$$

where the symbols L and A are used for edge length and triangle area respectively and (i, j, k_1) and (i, j, k_2) are the triangles that share the edge (i, j). Minimizing $\tilde{E}(\phi)$ with a set of boundary conditions is solved as a linear least-squares problem.

Figure 3 illustrates an harmonic map computed with this technique. Note that the open base of the 3-d mesh was mapped onto the boundary of the 2-d polygons and used as constraints.

3 Least-squares and optimality

Generally speaking, the goal of the least-squares technique is to estimate an unknown quantity given some data. In the case of the linear least-squares, it is assumed that the relationship between the unknown and the data is linear and that we measure error/loss by summing the squares of the error for each data point. So, in using least-squares, we are playing the following guessing game: given some data that we could gather, what is the value of the unknown quantity that we want to know? But before starting to guess, it would be useful to know how much we can rely on our guessing procedure. This can be learned from probability and estimation theory. In what follows, we formalize these notions with a theoretical detour to gain an understanding of the properties of the least-squares guess.

In the context of estimation theory, the least-squares technique is called an estimator. An estimator is a function, $\hat{X}$, that takes measured/sampled data, b, as input and produces an estimate of the parameters, $\hat{X}(b)$. Estimators are designed to exhibit optimality. Simply put, an estimator is optimal if it extracts all available information in the sampled data.

From a probabilistic point of view, both the data and the parameters are random variables. This can reflect two different things. If the problem involves measured data, then it reflects that our knowledge of these quantities is uncertain (e.g. noise in the measurements). However, quite often in computer graphics, the data is known precisely. In such context, the data is not a set of observations but rather a set of constraints that we want to fit our model to. The variance on the data can then be interpreted as how tight we want each constraint to be. In this case, the use of probabilities might seem inappropriate. Without getting into a philosophical discussion, we will take the bayesian stance of using probabilities as coherence constraints on a given assumption (linearity).

More specifically for least-squares, we make the assumption that the error at each data point, e_i, is a random variable. Recall that:

$$e_i = b_i - a_{i,1}x_1 - \ldots - a_{i,n}x_n$$

or equivalently using a matrix notation:

$$E = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}$$

with $\boldsymbol{A} = (a_{i,j})$.

3.1 The Gauss-Markov theorem

The main properties of the least-squares estimate, $\hat{X}_{LSE}$, are described in the Gauss-Markov theorem. We first state the theorem and then discuss its assumptions and conclusion.

If $\{b_i\}_m$ and $\{x_j\}_n$ are two sets of random variables such that

$$e_i = b_i - a_{i,1}x_1 - \ldots - a_{i,n}x_n$$

and

$$\begin{array}{ll} \text{A1:} & \{a_{i,j}\} \text{ are not random variables,} \\ \text{A2:} & E(e_i) = 0, \text{ for all } i, \\ \text{A3:} & var(e_i) = \sigma^2, \text{ for all } i, \text{ and} \\ \text{A4:} & cov(e_i, e_j) = 0, \text{ for all } i \text{ and } j, \end{array}$$

then the least-squares estimator,

$$\hat{X}_{LSE}(b_1, \ldots, b_m) = \arg\min_x \sum_i e_i^2,$$

is the **best unbiased linear estimator**.

3.1.1 Assumptions

Assumption A1 states that the independent variables are deterministic. In particular no error is made while measuring them.

Assumption A2, A3, and A4 describe the statistical properties of the errors. Each error has a zero mean (A2) and all the errors have the same variance (A3). Moreover, the errors are assumed uncorrelated (A4) (i.e. there is no linear dependency between any two errors). Hypothesis A2 implies that the linear model is correct up to some unknown parameters since $E(E) = E(\boldsymbol{b} - \boldsymbol{Ax}) = 0$

3.1.2 Conclusion

Let us examine what it means to be the best unbiased linear estimator. Form a trivial point of view it means that the estimator is the best in the category of unbiased linear estimators. Let us examine first what an unbiased estimator is and then how goodness is measured.

Unbiased estimators. $\hat{X}_{LSE}$ is linear in the sense that is is a linear function of B. $\hat{X}_{LSE}$ is also unbiased which means that its expected value is equal to the true value of $E(\hat{X}_{LSE}(B)) = X$. In other words, if we could do multiple experiments, we would notice that the distribution of $E(\hat{X}_{LSE}(B))$ is centered at the true value of X. This is a good property for an estimator since a biased estimator would seem to make systematic error, but it is not the only criteria. We should not be only concerned about the center of the distribution of the estimator but also about its spread or variance. If the estimator exhibit a lot of variance, it might give the correct answer on average (or in the limit) but provide an arbitrary bad estimate for a given set of observation. Unbiasedness does not guarantee that the estimate must be close to the true value of the parameter. In fact, an unbiased estimator can provide arbitrarily bad estimates. [23] presents some sobering examples of unbiased estimators.

Comparing estimators. According to the Gauss-Markov estimate $\hat{X}_{LSE}$ is the best unbiased linear estimator. A good estimator is one that is efficient. In term of statistic, for a general (not necessarily unbiased) estimator the comparison is based on the mean squares criteria, $E((\hat{X}-X)(\hat{X}-X)^T)$. For unbiased estimator, it can be shown that it is equivalent to minimum variance. So to recap, among all the linear estimator of X, $\hat{X}_{LSE}(B)$, is the estimator that has the lowest variance. This variance is equal to:

$$var(\hat{X}_{LSE}) = \sigma^2(A^TA)^{-1}.$$

In other words it means that as a random variable, the estimate is on average equal to its true value and the estimator is one that has the least spread/variance.

3.2 What can go wrong?

Each of the assumption in the Gauss-Markov theorem carries the seed . In what follows we examine each assumption in turn.

- A1 is in general unrealistic: if we treat dependent variables as random, it does not make much sense to treat independent variables differently. In the case of a linear regression, it means that errors are only taken into account along one of the dimension.
- A2 requires that the linear assumption be correct up to some unknown parameters. It is quite possible that the dependency between b and x is actually non-linear. In this case fitting a linear model could give unexpected results.
- A3 requires that the error or the importance of each data sample be the same. This is often not the case for practical applications where the accuracy of the sensor might vary as a function or space and or time, or where
- A4 requires that the errors on the data be uncorrelated. This is not always the case. Need an example.

To summarize, the three main breaking points of Gauss-Markov theorem and the ordinary least squares technique are the presence of errors in the dependent variables, non-identically distributed samples, and non-linearity. In the rest of this note, we

Errors in the dependent variables. The total least-squares technique (section 4) is meant to handle errors in all (dependent and independent) variables.

Non-identically distributed errors. The case of non-identically distributed errors can be handled in different ways. For small errors, weighted least-squares , total-least squares, and generalized total-squares allow the specification of increasing general data errors (section 4). For large errors, it might be best to leave out some of the data out of the estimation. These data samples are called outliers and the technique are qualified as robust (section 5. Another alternative is to make the model itself robust to errors using regularization (section 7). Regularization penalizes models to prevent overfitting.

Non-linearity. If the samples data does not fit a linear model (section 8), using a non-linear model might be the only sensible approach. A least-squares criteria can always be used to fit the model but the techniques are widely different from the linear case.

Although, there are many techniques that can address the shortcomings of ordinary least-squares, they often involve higher computational cost and sometime elusive solutions (e.g. non-linear least squares).

3.3 Other estimators

To shine further light on the theoretical roots of the least-squares estimate, we delve deeper in estimation theory to explore two general classes of estimator: the maximum likelihood and the maximum a posteriori.

3.3.1 Maximum likelihood estimate

If we assume a particular value for our parameter X, we can measure the probability of observing some data, B, using the probability distribution $f_{B|X}$[2]. However, the data vector B is the measured value not the parameters X. What we need is to find the probability density for X that is most likely to have produced the data. This inverse problem can be solved by reversing the role of the parameter and the data using the likelihood function, L, defined such that:

$$L_{X|B} = f_{B|X}.$$

It is important to keep in mind that these two functions are defined on different domains. $f_{B|X}$ is defined on the domain of the data (with the parameters fixed) whereas $L_{X|B}$ is defined in the domain of the parameters (with the data fixed). The *maximum likelihood estimator* is the value of the parameter that maximizes the likelihood function:

$$\hat{X}_{MLE}(b) = \max_{x} L_{X|B}(x)$$

Note also that the identification of the likelihood of the parameters with the probability of the data given the parameters is purely based on intuition and has no formal mathematical basis. In particular the likelihood is a function of the parameter not a probability

[2]The sign "|" denotes a conditional probability. $f_{B|X}$ is the probability density of B given X. Here we are interested in which way the knowing X informs us on B.

distribution over it. It remains that maximum likelihood estimators are more general than least-squares estimators and have good mathematical properties.

Going back to our least-squares topic, we can show that if the errors on the data follow independent and identically distributed gaussian distributions then the least-squares estimate is the maximum likelihood estimate. To briefly derive this result, assume that the error E_i follows:

$$f_{E_i}(e_i) = \frac{1}{\sigma\sqrt{(}2\pi)} \exp\left(\frac{-e_i^2}{2\sigma^2}\right)$$

$$L_{X|B}(x) = f_{B|X}(b_1,\dots,b_n) = \prod_i \frac{1}{\sqrt{2\pi}\sigma^2} \exp\left(\frac{-(b_i - xa_i)^2}{2\sigma^2}\right)$$

Since log is a non-decreasing function, minimizing $\log(L_{X|B})$ is the same as minimizing $L_{X|B}$.

$$\log(L_{X|B}(x)) = \sum_i \frac{-(b_i - xa_i)^2}{2\sigma^2} + \text{const.}$$

This last equations shows that maximizing the likelihood is equivalent to minimizing the sum of squares $\sum_i (b_i - xa_i)^2$. Note that requiring that the errors follow the same gaussian distribution, $N(0,\sigma^2)$, is stronger requirement than assumptions A2-4. We can extend this result to the case where the errors follow different gaussian distributions. With, E_i, following $N(0,\sigma_i)$, the MLE estimate is the minimum of :

$$\sum_i \frac{(b_i - xa_i)^2}{\sigma_i^2}$$

This is a weighted least-squares problem where the weights are the inverse of the variance of the errors.

A similar approach can be taken for arbitrary error distributions. However, depending on the distribution, the maximum likelihood estimate might not exist nor be unique.

3.3.2 Maximum a posteriori

Using the maximum likelihood estimate, we do not make any assumption on the parameter X. If we do know something about the distribution of X, we cannot take advantage of it. The *maximum a posteriori* (or MAP) criteria addresses this shortcoming by treating X as a random variable. In this case, we have two sources of information for our parameter, a prior distribution, g_B, that tells us what are the likely values of the parameters before considering any data, and, $f_{B|X}$, that tells us the probabilty if the data given the parameters. Following Bayes rule, we can derive a posterior density for the parameter:

$$\frac{f_{B|X}(x)g_X(x)}{Pr(b)}$$

The maximum a posterior estimate is then defined as the value of the parameter that maximizes the posterior distribution:

$$\hat{X}_{MAP}(b) = \max_x \frac{f_{B|X}(x)g_X(x)}{Pr(b)} = \max_x f_{B|X}(x)g_X(x)$$

The denominator of the posterior distribution does not depend on X and therefore plays no role in the optimization. Observe that the MAP estimate of X coincides with the ML estimate when the prior g_X is uniform (that is, a constant function).

3.4 Measuring goodness of fit

After fitting a linear model to some sample data, one might wonder how well one did. Goodness of fit can be measured by the sum of squared residuals. Unfortunately this metric is not invariant with respect to the measurement units used to describe the data and as such is not good for model comparison. What we really need is a relative measure for evaluating the fit. To begin recall that the geometry of least-squares guarantee that the residual and the solution are orthogonal:

$$\hat{\boldsymbol{x}}^T \boldsymbol{r} = 0,$$

from which we derive

$$\boldsymbol{b}^T\boldsymbol{b} = (\boldsymbol{A}\hat{\boldsymbol{x}})^T\boldsymbol{A}\hat{\boldsymbol{x}} + \boldsymbol{r}^T\boldsymbol{r} + 2\hat{\boldsymbol{x}}^T\boldsymbol{r} = (\boldsymbol{A}\hat{\boldsymbol{x}})^T\boldsymbol{A}\hat{\boldsymbol{x}} + \boldsymbol{r}^T\boldsymbol{r}$$

This equation can be used to define the *coefficient of determination*

$$R^2 = \frac{(\boldsymbol{A}\hat{\boldsymbol{x}})^T\boldsymbol{A}\hat{\boldsymbol{x}}}{\boldsymbol{b}^T\boldsymbol{b}} = 1 - \frac{\boldsymbol{r}^T\boldsymbol{r}}{\boldsymbol{b}^T\boldsymbol{b}}$$

In other words, R^2 is the proportion of the squares variations in the data that can be explained by the regression hyperplane. Clearly, $R^2 \in [0,1]$ and the large R^2 is the better the fit. Geometrically, R^2 is the square of the cosine between the two vectors $\boldsymbol{b}$ and $\boldsymbol{A}\hat{\boldsymbol{x}}$.

One of the issue with the R^2 is that it is sensitive to the addition of a constant. The Centered R^2 removes this sensitivity:

$$\text{Centered } R^2 = 1 - \frac{\boldsymbol{r}^T\boldsymbol{r}}{\boldsymbol{b}^T\boldsymbol{b} - n\bar{\boldsymbol{b}}^2}$$

where $\bar{VB} = \sum_{i=1}^n b_i$.

The two criteria presented so far suffer from yet another problem, they are both non-decreasing function of the number of parameters. This means that using these criteria a more complex model would be prefered. The criteria $\bar{R}^2$ corrects this problem:

$$\bar{R}^2 = 1 - \frac{\boldsymbol{r}^r\boldsymbol{r}/(n-m)}{(\boldsymbol{b}^T\boldsymbol{b} - m\bar{\boldsymbol{b}}^2)/(n-1)} = R^2 - \frac{m-1}{n-m}(1-R^2)$$

$\bar{R}^2$ is the Centered R^2 with a penalty term that depends on model complexity. As m, the number of parameters (i.e. complexity of the model) increases, the scalar $\frac{m-1}{n-m}$ increases but $1-R^2$ decreases. Thus, $\bar{R}^2$ need not be increasing with the number of parameters.

4 Least-squares with generalized errors

As we have reviewed in the previous section, the Gauss-Markov theorem under rather strict hypothesis: the error on the data samples have to be centered, have the same variance, and be uncorrelated. This section presents how to adapt the ordinary least squares algorithm when the errors follow more complex distributions.

Although these methods are more general than ordinary least-squares, keep in mind that they require that the error distribution is known. If we have no knowledge of these distributions, the best guess is the least-squares estimate. Much of the remainder of these course notes covers methods for dealing with alternate error distributions.

4.1 Weighted least-squares

The weighted least-squares method extend the least-squares procedure to the case where the data samples have different variance. In other words some samples have more error or less influence than others. Assuming that $var(e_i) = \sigma_i^2$, if this is assumption is the only departure from the Gauss-Markov theorem assumption the best linear unbiased estimate for the parameters is:

$$\arg\min_{\boldsymbol{x}} \sum_{i=1}^{n} \frac{(b_i - \sum_{j=1}^{m} x_j a_{i,j})^2}{\sigma_i^2}.$$

If we call $\boldsymbol{W} = Var(\boldsymbol{e})^{-1} = diag(\frac{1}{\sigma_1^2}, \ldots, \frac{1}{\sigma_n^2})$, we can rewrite the previous formula in matrix form as

$$\arg\min_{\boldsymbol{x}} (\boldsymbol{W}(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}))^T (\boldsymbol{W}(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x})),$$

from which we can derive the modified normal equation:

$$\boldsymbol{A}^T \boldsymbol{W}^T \boldsymbol{W} \boldsymbol{A} \boldsymbol{x} = \boldsymbol{A}^T \boldsymbol{W}^T \boldsymbol{W} \boldsymbol{b},$$

and the weighted least-squares estimate:

$$\hat{\boldsymbol{x}}_{WLSE} = (\boldsymbol{A}^T \boldsymbol{W}^T \boldsymbol{W} \boldsymbol{A})^{-1} \boldsymbol{A}^T \boldsymbol{W}^T \boldsymbol{W} \boldsymbol{b}.$$

One of the difficulties with the previous method is to estimate the variance in the errors. We can use the sample variance using repeated measurements. However if the data is deterministic, this is irrelevant. Still even in this case, we might want to weight the different samples. The weights can be chosen arbitrarily to reflect the importance of each sample. For instance, we could weight the sample according to a sampling density to counter-act the effect of sample clusters. Another technique is to use the square residuals as weights.

4.2 Total least-squares

Line fitting

For 1D lines one typically uses the one-dimensional y=f(x) parameterization. Fitting a line using this representation is easy, but it only works well when the line slope is shallow (Figure 4). When the slope is steep the error used in the fit is not between the data point and the closest point on the line (as would be desired), rather it is between the data point and and the point on the line that is vertically above or below it.

In *total least squares* the fit is measured as the sum squared distance between the data and their closest points on the line. This approach generalizes to fitting hyperplanes.

Total least squares is one example where an initial least squares formulation of the problem results in an eigenvalue problem rather than the usual linear system.

The standard line representation is $y = ax + c$. Rewrite this as $1 \cdot y - a \cdot x = c$, or $(1, -a)^T(y, x) = c$; call this

$$\boldsymbol{a}^T\boldsymbol{x} = c$$

(a hyperplane).

Now minimize the squared distances from the points x_k to the line

$$\min \sum (c - \boldsymbol{a}^T\boldsymbol{x}_k)^2$$

subject to $\|\boldsymbol{a}\| = 1$.

$\boldsymbol{a}^T\boldsymbol{x} - c$ is the distance to the line ($\boldsymbol{a}^T\boldsymbol{x} + c$ might look more familiar, it would be the distance if we had used the hyperplane equation $\boldsymbol{a}^T\boldsymbol{x} + c = 0$.) Note that $\boldsymbol{a}$ and c can be scaled without changing the plane, so scale so that the normal vector $\boldsymbol{a}$ has length 1 to eliminate this freedom.

$$\begin{array}{rl}
\text{argmin}_{\boldsymbol{a},c} & \sum (c - \boldsymbol{a}^T\boldsymbol{x}_k)^2 + \lambda(\boldsymbol{a}^T\boldsymbol{a} - 1) \\
 & = \sum \boldsymbol{a}^T\boldsymbol{x}_k\boldsymbol{x}_k^T\boldsymbol{a} - 2\sum c\boldsymbol{a}^T\boldsymbol{x}_k + \sum c^2 + \lambda(\boldsymbol{a}^T\boldsymbol{a} - 1) \\
 & = \boldsymbol{a}^T(\sum \boldsymbol{x}_k\boldsymbol{x}_k^T)\boldsymbol{a} - 2c\boldsymbol{a}^T\sum \boldsymbol{x}_k + \sum c^2 + \lambda(\boldsymbol{a}^T\boldsymbol{a} - 1) \\
\text{calling} & \boldsymbol{X} \equiv \sum \boldsymbol{x}_k\boldsymbol{x}_k^T \text{ and } \hat{\boldsymbol{x}} \equiv \sum \boldsymbol{x}_k \text{ then} \\
 & = \boldsymbol{a}^T\boldsymbol{X}\boldsymbol{a} - 2c\boldsymbol{a}^T\hat{\boldsymbol{x}} + Nc^2 + \lambda(\boldsymbol{a}^T\boldsymbol{a} - 1) \\
\dfrac{d}{d\boldsymbol{a}} = 0 & = 2\boldsymbol{X}\boldsymbol{a} - 2c\hat{\boldsymbol{x}} + 2\lambda\boldsymbol{a} \\
\dfrac{d}{dc} = 0 & = 2\boldsymbol{a}^T\hat{\boldsymbol{x}} + 2Nc = 0 \Rightarrow c = \boldsymbol{a}^T\hat{\boldsymbol{x}}/N \\
\text{substitute c} & 2\boldsymbol{X}\boldsymbol{a} - 2(\boldsymbol{a}^T\hat{\boldsymbol{x}})\hat{\boldsymbol{x}}/N + 2\lambda\boldsymbol{a} = 0 \\
 & = \boldsymbol{X}\boldsymbol{a} - \hat{\boldsymbol{x}}(\boldsymbol{a}^T\hat{\boldsymbol{x}})/N + \lambda\boldsymbol{a} = 0 \\
 & = \boldsymbol{X}\boldsymbol{a} - \hat{\boldsymbol{x}}(\hat{\boldsymbol{x}}^T\boldsymbol{a})/N + \lambda\boldsymbol{a} = 0
\end{array}$$

$$= (\boldsymbol{X} - \frac{1}{N}\hat{\boldsymbol{x}}\hat{\boldsymbol{x}}^T)\boldsymbol{a} + \lambda \boldsymbol{a} = 0$$

which is an eigenvalue problem, and the minimum eigenvalue minimizes the original problem, with the corresponding eigenvector being the desired coefficient (normal) vector $\boldsymbol{a}$.

Total least squares are more commonly used in computer vision than computer graphics, but they are mentioned for example in [1] (which in turn references M. Pauly's 2003 thesis).

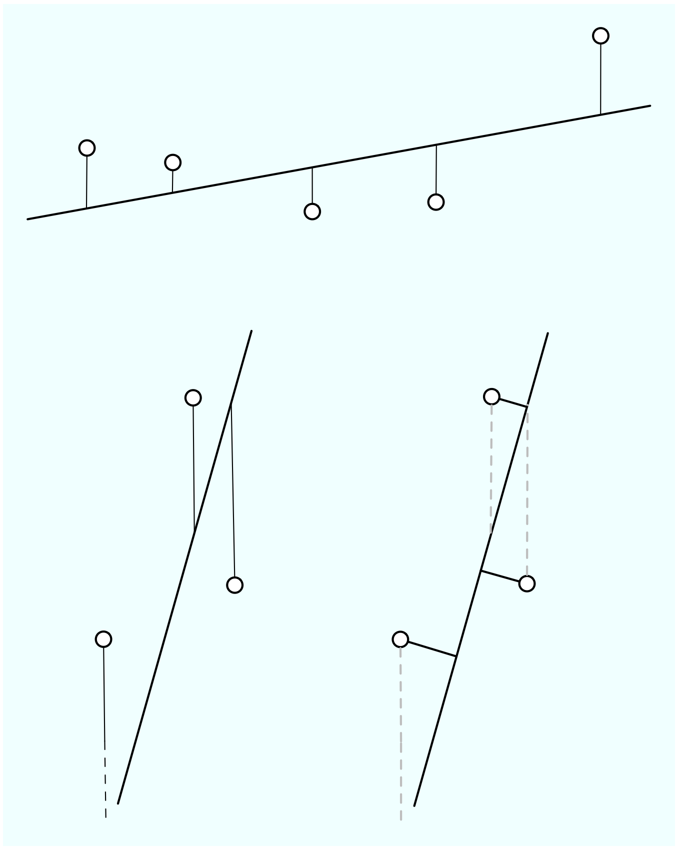

Figure 4: Fitting a $y = ax$ line by minimizing the squared errors between y and ax makes sense when the line slope is shallow (top), but not when the slope is large (bottom left). In both cases it is preferable to minimize the actual distance between the points and the line (bottom right) rather than measuring only the y component of the error. This is accomplished by *total least squares*.

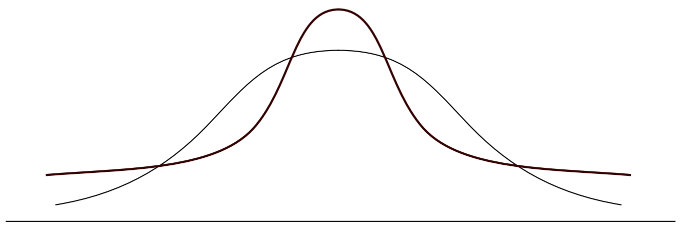

Figure 5: A high kurtosis density (heavy line) has both more data close to the mean, and more outliers, than a Gaussian distribution (light line).

5 Robust least squares

As described earlier in this course, least squares assumes that the errors in the data have a Gaussian distribution. This is the case, sometimes. For example, the sum of a number of separate error terms, regardless of their individual distributions, rapidly approaches a Gaussian.

The Gaussian assumption is incorrect in many other cases. These distributions can be divided according to their *kurtosis*. Figure 5 illustrates this concept. A higher kurtosis density (heavy line) has both more data close to the mean, and more outliers, than a Gaussian distribution of identical variance. The distribution of differences between adjacent image pixels is one example of a high kurtosis distribution: most differences are small, but there are a significant number of large differences at occlusions.

The least squares estimate can be degraded arbitrarily by even a single outlying point (Figure. 6). Intuitively, note that because least squares is minimizing the sum of the *squared* error, the largest errors dominate the result. Noise from a heavy tailed distribution can thus be expected to cause degraded fits if Gaussian noise is assumed, because there are more outliers than would be expected under the Gaussian distribution.

Robust methods address this problem, in essence, by attempting to identify and ignore or de-weight these outlying points.

5.1 Redescending estimators

In the standard least squares approach, the square of the error is (obviously) what is being minimized. One class of robust estimators replace the error function with something other than the square.

First, consider estimating the mean of some data using least squares: The mean estimate is

$$\arg\min_{\bar{x}} \frac{1}{N}(\sum x_k - \bar{x})^2$$

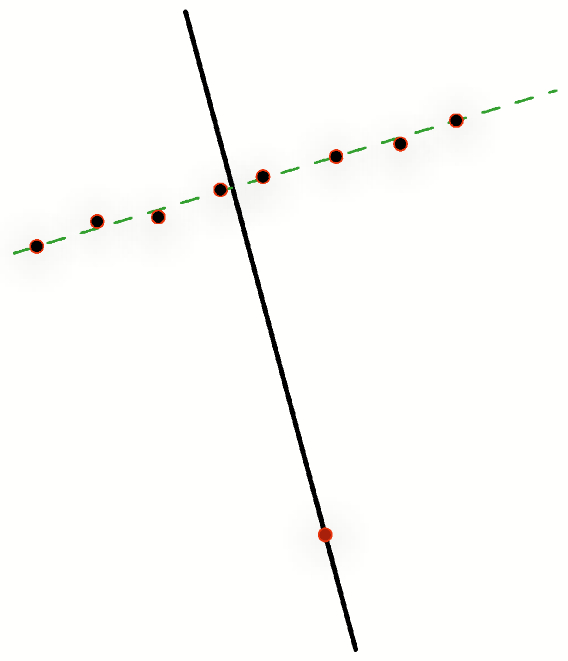

Figure 6: Even a single outlying point (red point) can destroy a least squares estimate.

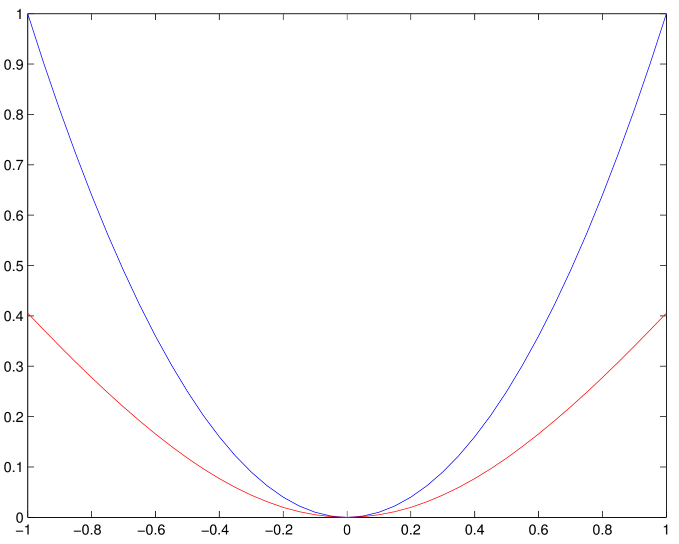

Figure 7: The redescending estimator function $\log\left(1+\frac{1}{2}\left(\frac{x}{\sigma}\right)^2\right)$ (red) versus the standard quadratic error $y = x^2$ (blue).

The $\bar{x}$ that minimizes this is found by setting the derivative to zero,

$$\frac{2}{N}(\sum x_k - \bar{x}) = 0$$

From this, it can be seen that any single x_k *linearly* effects the estimate.

Next, consider using the first power (the absolute value) rather than the square: The mean estimate is

$$\arg\min_{\bar{x}} \frac{1}{N} |\sum x_k - \bar{x}|^1$$

Here, after taking the derivative, any particular x_k only affects the estimate by a *constant* amount.

Redescending estimators take this one step further. In these functions, the derivative goes to zero with increasing error (although the error itself continues to rise). Thus, large errors are completely ignored.

An example of such an estimator, discussed in [3] (see Figure 7), is

$$\log\left(1+\frac{1}{2}\left(\frac{x}{\sigma}\right)^2\right)$$

with derivative

$$\frac{2x}{2\sigma^2 + x^2}$$

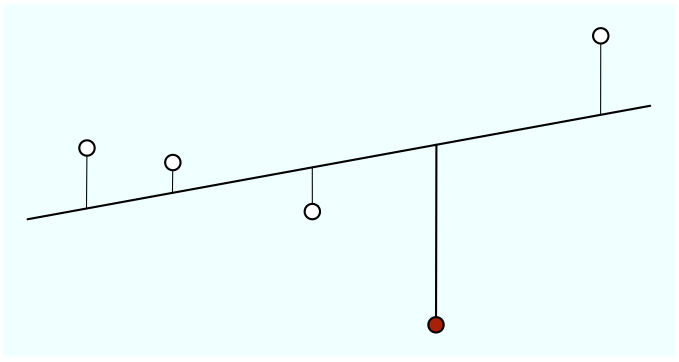

Figure 8: A simple approach to robust least squares fitting is to first do an ordinary least squares fit, then identify the k data points with the largest residuals, omit these, perform the fit on the remaining data.

5.2 Iteratively reweighted least-squares

In concept the goal of robust least squares fitting is to identify outlying points and discard them. One simple way to approach this is to first fit the data using ordinary least squares, then identify the k points with the largest residuals, discard these, and fit the remaining data (Figure 8. This approach has been called trimmed least squares.

A drawback of this approach is that it requires manual tuning of the number of expected outliers, k. It also assumes that the data can be neatly divided into good data and outliers – which may or may not be the case.

Iteratively Reweighted Least Squares (IRLS) addresses this by minimizing

$$\|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}\|_p$$

where $\|\|_p$ is the "l_p" norm, i.e.,

$$\|\boldsymbol{x}\|_p = \left(\sum \boldsymbol{x}_k^p)\right)^{1/p}$$

The usual least squares minimizes $\|\|_p$ for $p = 2$. As noted above, minimizing the squared errors causes the largest errors to be fit at the expense of disregarding small errors. This is the wrong approach. if the largest errors are due to outliers. Reducing p reduces this emphasis of the largest errors; with $p = 1$ all errors are treated equally (so the sum of the absolute values of the errors is what is minimized).

The key to doing this is to note that an error $|e|^p$ can be restated as

$$|e|^p = |e|^{p-2}e^2$$

Then, interpret the $|e|^{p-2}$ factor as a weight, and minimize e^2 using weighted least squares.

The residuals e_k are not known of course. IRLS approaches this by an iterative scheme in which the residuals are found from a fit using the existing weights, and these residuals then form the weights for the next iteration:

```
$\boldsymbol{W} = \boldsymbol{I}$
iterate for $i = 1\ldots$
$\boldsymbol{e}_i = \boldsymbol{W}(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_i)$
$\boldsymbol{W} = \mathrm{diag}(|\boldsymbol{e}_i|^{p-2}/2)$
solve $\boldsymbol{x}_{k+1} = \mathrm{argmin}\|\boldsymbol{W}(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_k)\|_2$
```

Note that the division by two in $\boldsymbol{e}_i^{p-2}/2$ is because in solving $\|\boldsymbol{W}(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_k)\|_2$ the weights $\boldsymbol{W}$ are squared.

This algorithm converges for $1 < p < 3$, although it requires many iterations as p approaches 1.

IRLS is used in [25] in a texture synthesis context order to robustly compare pixel values between corresponding regions of the reference texture and the texture being synthesized.

5.3 RANSAC

RANSAC [12] is a simple and general procedure for fitting models to data that has clearly separated outliers. The acronym stands for RANdom SAmple Consensus. Although RANSAC itself is not directly connected to least squares, it forms the bases for procedures such as Least Median of Squares (described below).

The starting point for RANSAC is the idea that it is possible to select a smaller subset of k samples from the N data points that contains no outliers. Of course there is no automatic way to know which data points are outliers – if this were not the case, this information could be used to just remove the outliers from the data.

Instead RANSAC iterates the following steps:

1. randomly select k points
2. fit the model to these points
3. evaluate the quality of this fit on the remaining points

The subset that has the best fit the remaining data is retained – it is assumed that this subset contains no outliers. Finally, the model is re-fit to all the data that are “sufficiently close” to the model initially fit from the k points.

Instead RANSAC repeatedly makes a random selection of the k points, fits the model to these data, and then evaluates the fit on the remaining data. The subset that has the best fit the remaining data is retained – it is assumed that this subset contains no outliers. Finally, the model is re-fit to all the data that are "sufficiently close" to the model initially fit from the k points.

The previous paragraph describes the general RANSAC strategy. To give a more specific example, consider the case of fitting a line to data with 50% outliers. In this case, in choosing a single point, there is a 50% chance of choosing a "good" point rather than an outlier. A single point is not sufficient to fit a line, so instead pick $k = 2$. There is a 25% chance of selecting a subset of 2 points that contains no outliers. A line is fit to each of a number of randomly chosen subsets of two points. If the subset contains two good points, then it will be the case that the resulting line *lies close to all the remaining good data*, whereas if the subset contains one or two outliers, then very few other points will like close to the line. Thus, the RANSAC fitting procedure can easily identify a set of good data. If a sufficient number of subsets are evaluated, one of them will contain good data with high probability – see a description of RANSAC for more details [14] (or you can easily work out the probabilities).

5.4 Least Median of Squares, Forward Search

Least Median of Squares (sometimes abbreviated LMedS) is a RANSAC-like scheme in which the median error is used to evaluate RANSAC step 3, the fit of the remaining points to the model.

The median is a well known example of a robust estimator: it can produce a reasonable estimate in data corrupted by nearly 50% outliers. For example, consider the following data,

```
0.3 0.2 99 99 0.5 0.8 99
```

Here the valid values are 0.3, 0.2, etc., and 99 is an outlier. The sorted set of values is

```
0.2 0.3 0.5 0.8 99 99 99
```

and the median value is 0.8 – which is clearly not representative of the mean value of the inlying data, but at least it is an inlying data sample. In the robust statistics terminology, the median is said to have a breakdown point of 50%. The breakdown point is the maximum percentage of data points to which an arbitrarily large error can be added without causing the estimator to change arbitrarily.

Figure 9 shows an example of data fit with LMedS and ordinary least squares.

Forward Search is another variation on these ideas. In this procedure, a subset of inlying points is first identified (e.g., using RANSAC, or perhaps manually), and then this set is grown by iterating the following steps:

1. add the data point with the lowest residual to the currently fit model

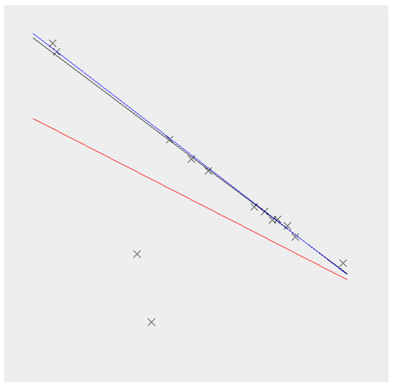

Figure 9: Successful application of Least Median of Squares fitting: The LMedS line (blue) lies close to the model line (black) from which the inliers were generated. The ordinary least squares fit (red line) is influenced by the two outliers.

2. re-fit the model to the new set of points.

The iteration is terminated when the lowest residual is larger than some threshold reflecting an outlier.

Forward search is used to achieve surface fitting to noisy data, while avoiding smoothing of edges, in a Siggraph 2005 paper [13]. In this paper, Fleishman et al. realized that points on the other side of a crease can be regarded as outliers. They are thus able to apply forward search to grow the set of points up to but not across creases, thus producing smoothed but crease-preserving models from noisy scan data.

5.4.1 Example: detecting kinematic constraints in motion data

Robustness issues often arise when processing recorded data like motion capture data. In [6] full-body motion capture data is analyzed to detect kinematic constraints such as foot plant contacts. For a given joint, the technique assumes that between frames i and $i+1$ a rigid motion $\boldsymbol{D}_i$ occurred. Instantaneous constraints are defined as the set of points remaining stationary under a transformation $\boldsymbol{D}_i$. With $\boldsymbol{D}_i = (\boldsymbol{R}_i|\boldsymbol{t}_i)$, a point p is stationary if and only if:

$$(\boldsymbol{R}-\boldsymbol{I})\boldsymbol{p} = -\boldsymbol{t}.$$

Note that the previous equation has a solution only if the translation $\boldsymbol{t}$ does not have a component along the axis of $\boldsymbol{r}$. The existence of stationary points can be hidden by noise in the data. A test can be designed using the residual $\|(\boldsymbol{R}-\boldsymbol{I})\boldsymbol{p}-\boldsymbol{t}\|$ (to determine the existence of a solution) and the singular values of $\boldsymbol{R}-\boldsymbol{I}$ (to determine the number of dimensions of the constraint). Both tests must be robust to noisy data to avoid selecting incorrect frames. “Inlier” frames are detected using a least-median of squares approach.

6 Constrained Least Squares

6.1 Lagrange Multipliers

Lagrange multipliers are a general approach to adding equality constraints for many types of equations, not just linear systems formulated using least squares. Lagrange multipliers have been used to add constraints to linear systems to solve problems such as inverse kinematics, camera control, and constraint problems in physical simulation. We will describe an inverse kinematics solution below.

A example of an *equality constraint* is: find the minimum value of x^2+y^2 subject to x being constrained to the equality $x=1$. In this simple case Lagrange multipliers are not needed – the solution can be found by substituting the value of x for the variable in the equation, resulting in an equation in the one variable y whose solution is easy. This approach might be termed "substitute and simplify".

Lagrange multipliers are helpful in more complicated situations where the "substitute and simplify" approach cannot be easily applied. In the case of linear systems, the Lagrange multiplier approach often leads to a block matrix system, which we will see below.

First, let's work through a Lagrange multiplier example for a non-matrix case: find the maximum value of $f(x,y)=x$ subject to x,y lying on the unit circle, i.e., $x^2+y^2=1$. This function is a linear "ramp" increasing in the x direction, and the point of maximum x on the unit circle will clearly turn out to be $x=1, y=0$.

The steps involved in using Lagrange multipliers are:

1. Identify the constraint, $x^2+y^2=1$ in this case.
2. Reformulate the constraint as "something = 0". In the example, $x^2+y^2=1$ becomes $x^2+y^2-1=0$.
3. Multiply the "something" by λ. So $\lambda(x^2+y^2-1)$.
4. Add this to the original equation. In the example,
$$x+\lambda(x^2+y^2-1)$$
5. Lastly, try to find the maximum or minimum of this new equation by the standard calculus approach of setting the derivative with respect to the variables to zero, and solving. *The λ is one of these variables.*

$$\frac{d}{dx}\left[x+\lambda(x^2+y^2-1)\right] \quad = \quad 1+2\lambda x=0$$

$$\begin{aligned}\frac{d}{dy}\left[x+\lambda(x^2+y^2-1)\right] &= 2\lambda y = 0\\ \frac{d}{d\lambda}\left[x+\lambda(x^2+y^2-1)\right] &= (x^2+y^2-1) = 0\end{aligned}$$

If possible, solve for λ and substitute into the other equation(s). In this case, from the first equation, solve for $\lambda = -1/2x$. Then substitute this into the second equation, obtaining $y/x = 0$. Considering this and the third equation then, the solution is evidently $x = 1, y = 0$.

This technique might seem a bit weird at first – what is this new variable "λ"? Because of this it may be easiest to first practice using it and see that it works, and then work on the intuition behind it.

The intuition behind it is not that hard, either. Consider minimizing $f(x,y)$ constrained by $g(x,y) = 0$. The proposed solution can only move perpendicular to the gradient of g, otherwise it is violating the constraint. A minimum along the path $g(x,y) = 0$ is found at a place where moving along g locally will not change f. This means that the gradient of f is locally parallel to the gradient of g, and so is a linear multiple of it, which is expressed by the Lagrange multiplier setup

$$\nabla f(x,y) + \lambda \nabla g(x,y) = 0$$

6.1.1 Example: closest point on a sphere.

As a slightly more relevant example, let's take the problem of finding the point on a sphere that is closest to a given point. We could imagine this sort of problem coming up in a graphics context (though of course the solution to this particular problem can be seen immediately without going through the math). The sphere will be at the origin and unit size for simplicity, though the technique works the same for an arbitrary placement and size.

The constraint that a point p is on a unit-sized sphere can be written $\boldsymbol{p}^T\boldsymbol{p} = 1$. The point $\boldsymbol{p}$ on the unit sphere that is closest to a given point $\boldsymbol{q}$ can then be written using a Lagrange multiplier as

$$(\boldsymbol{p}-\boldsymbol{q})^T(\boldsymbol{p}-\boldsymbol{q}) + \lambda(\boldsymbol{p}^T\boldsymbol{p}-1)$$

or

$$\boldsymbol{p}^T\boldsymbol{p} - 2\boldsymbol{p}^T\boldsymbol{q} + \boldsymbol{q}^T\boldsymbol{q} + \lambda(\boldsymbol{p}^T\boldsymbol{p}-1)$$

Now, first take the derivative with respect to $\boldsymbol{p}$ and set to zero, and then solve for $\boldsymbol{p}$:

$$\begin{aligned}&\frac{d}{d\boldsymbol{p}} = 2\boldsymbol{p} - 2\boldsymbol{q} + 2\lambda\boldsymbol{p} = 0\\ &\boldsymbol{p}(1+\lambda) = \boldsymbol{q}\\ &\boldsymbol{p} = \frac{1}{1+\lambda}\boldsymbol{q}\end{aligned}$$

Next take the derivative with respect to λ, recovering the constraint equation $\boldsymbol{p}^T\boldsymbol{p} = 1$, and substitute this expression for $\boldsymbol{p}$ into this:

$$\frac{\boldsymbol{q}^T\boldsymbol{q}}{(1+\lambda)^2} = 1$$
$$\sqrt{\boldsymbol{q}^T\boldsymbol{q}} = 1+\lambda$$
$$\lambda = \sqrt{\boldsymbol{q}^T\boldsymbol{q}} - 1$$

Lastly, substitute this expression for λ back into the expression for $\boldsymbol{p}$:

$$\boldsymbol{p} = \frac{1}{\sqrt{\boldsymbol{q}^T\boldsymbol{q}}}\boldsymbol{q} = \frac{\boldsymbol{q}}{\|\boldsymbol{q}\|}$$

In other words, the solution is to just take the vector to $\boldsymbol{q}$ and normalize its length (as we already knew in this case).

6.1.2 Example: inverse kinematics

We will use inverse kinematics as an example of using the Lagrange multiplier technique in a least squares and graphics context.

Define $\boldsymbol{p}$ as the 2d position of an end-effector, controlled by the mouse. Define $\boldsymbol{q}$ as a "state" vector, such as the vector of n joint angles of a limb. Then $\boldsymbol{p}$ is a nonlinear function of $\boldsymbol{q}$,

$$\boldsymbol{p} = f(\boldsymbol{q})$$

In inverse kinematics, we want to find $\boldsymbol{q}$ given a new position of $\boldsymbol{p}$. This is usually underconstrained since the number n of joint angles (usually 3 for each joint) is greater than the number of known values – the two (x,y) components of the mouse location. The problem is also complicated because $f()$ is nonlinear.

In several papers Gleicher and Witkin used the idea of solving this by first locally linearizing by taking the derivative with respect to time [15, 16]:

$$\frac{d\boldsymbol{p}}{dt} = \frac{df}{d\boldsymbol{q}}\frac{d\boldsymbol{q}}{dt}$$

Let's denote $\boldsymbol{J} \equiv \frac{df}{d\boldsymbol{q}}$, so

$$\dot{\boldsymbol{p}} = \boldsymbol{J}\dot{\boldsymbol{q}}$$

where $\dot{\boldsymbol{p}}$ denotes the derivative of the mouse position (or end effector) with respect to time, and similarly for $\dot{\boldsymbol{q}}$.

Now the system is linear, but still underconstrained – the mouse has two degrees of freedom, but the skeleton typically has more. This can be fixed by adding an additional goal. Gleicher suggested that the change in state be as small as possible, thus the squared difference of the change in each joint angle from zero

$$\|\dot{\boldsymbol{q}}\|^2 = \sum_i^n (\dot{\boldsymbol{q}}_i - 0)^2$$

is minimized. Minimizing this alone would result in no movement to the limb, so instead constrain it so that the result of the joint angle change, $\boldsymbol{J}\dot{\boldsymbol{q}}$, matches the change in the mouse position / end effector, $\dot{\boldsymbol{p}}$. Combining these gives the goal

$$\arg\min_{\boldsymbol{q}} \frac{1}{2}\dot{\boldsymbol{q}}^T\dot{\boldsymbol{q}} + \lambda(\dot{\boldsymbol{p}} - \boldsymbol{J}\dot{\boldsymbol{q}})$$

(The $\frac{1}{2}$ is included to cancel the factor 2 that always appears when taking the derivative.)

Taking the derivatives with respect to all the variables $\dot{\boldsymbol{q}}, \lambda$,

$$\begin{aligned}\tfrac{d}{d\dot{\boldsymbol{q}}} &= 0 = \dot{\boldsymbol{q}} - \boldsymbol{J}^T\lambda \\ \tfrac{d}{d\lambda} &= 0 = \dot{\boldsymbol{p}} - \boldsymbol{J}^T\dot{\boldsymbol{q}}\end{aligned}$$

As mentioned earlier, simultaneous linear equations such as these can sometimes be solved by first solving for one unknown such as the λ and substituting the resulting expression into the other equation(s), as was done in the previous example in this section. However, a numerical solution is also possible and does not require any further algebra. The numerical solution uses a block matrix form,

$$\begin{bmatrix} \boldsymbol{J} & 0 \\ \boldsymbol{I} & -\boldsymbol{J}^T \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{q}} \\ \lambda \end{bmatrix} = \begin{bmatrix} \dot{\boldsymbol{p}} \\ 0 \end{bmatrix}$$

where (because this is a block matrix system) the variables themselves represent vector/matrix quantities rather than individual numbers ("scalars"). In our example, $\dot{\boldsymbol{p}}$ is a 2×1 column vector, $\dot{\boldsymbol{q}}$ is a $n \times 1$ vector, J is the $2 \times n$ matrix of derivatives, I is an $n \times n$ identity matrix, and 0 denotes a zero vector of length n. Schematically, the dimensions are

$$\begin{bmatrix} 2\times n & 2\times 1 \\ n\times n & n\times 2 \end{bmatrix} \begin{bmatrix} n\times 1 \\ 2\times 1 \end{bmatrix} = \begin{bmatrix} 2\times 1 \\ n\times 1 \end{bmatrix}$$

A correct block matrix system can be symbolically manipulated as if the individual components were ordinary scalars. You should try multiplying out the left hand side of the block matrix equation above to see that it does represent the original pair of equations.

6.1.3 Other Applications

6.2 Convex Weights

In applications such as skinning and blendshapes it is desirable that the weights (for example the skinning weights on the various joints) sum to one. With this constraint, the weights are no longer independent degrees of freedom. One approach to this problem would be to remove one degree of freedom.

In a skinning paper in Siggraph 2005, James and Twigg [24] took the alternate approach of explicitly adding the sum-to-one constraint to the system:

$$\begin{bmatrix} \boldsymbol{A} \\ 1 \ldots 1 \end{bmatrix} [\boldsymbol{x}] = \begin{bmatrix} \boldsymbol{b} \\ 1 \end{bmatrix}$$

(Some other techniques in this paper are discussed in Section 7).

6.3 Inequality constraints

It is also often desirable that weights (such as blendshape weights) be non-negative, in addition to summing to one. Although the set of blendshape bases will be designed with this in mind, negative weights can arise, for example, when attempting to fit marker motion that lies outside the range of the blendshapes (this may result from either noise in the marker motion, or from positions that are simply not part of the range of motion achievable using the blendshapes). Chuang and Bregler discussed this problem in [8].

Unfortunately achieving inequality constraints such as non-negative weights cannot be achieved with standard matrix tools such as the SVD. We will mention the type of routine needed in this section.

The constraint of non-negative weights can be accomplished using non-negative least squares (nnls). A routine for this purpose was described in [26], and C code is available on the web (search for nnls.c). This routine was used in [24] to find non-negative vertex-joint weights for skinning.

Quadratic programming solves least squares problems subject to both general inequality constraints

$$Cx \leq d$$

and equality constraints

$$Ex = f$$

There are quite a few C/C++ quadratic programming libraries available on the web, and Matlab's optimization toolbox has includes QP.

In Bregler et al. 's cartoon capture paper [4] quadratic programming was used to simultaneously cause weights to sum to one and to lie in the range $[-0.5, 1.5]$.

6.3.1 Example: illuminant estimation

The problem of *illuminant estimation* can be described as estimating the light from an image of a scene. The *gamut mapping* solution of this problem consists in determining a set of mappings that take the image data onto a gamut of reference colors under a known light. *Gamut Constrained illuminant estimation* [11] is a variant that allows the use of a set of plausible lights.

This is solved as a classification problem where each plausible light is represented by its gamut and is tested again the image data. For a given light, the fit is evaluated for each color sample by measuring how far it is from the illuminant's gamut. Let's call $\boldsymbol{b}_k$ a color sample, $\boldsymbol{a}_{i,j}$ a color of gamut G_i, and $CH(G_i)$ the convex hull of G_i. The error for sample $\boldsymbol{b}_k$ is

$$e_i(\boldsymbol{b}_k) = \|\boldsymbol{b}_k - \sum_{\boldsymbol{a}_{i,j} \in CH(G_i)} x_j \boldsymbol{a}_{i,j}\|^2.$$

Solving for

$$\min_{\{x_j\}} \sum_k e_k^i, \text{ with } x_j \geq 0 \text{ for all } j$$

gives a measure of fit between the data and the ith illuminant. Since, this measures how well the image samples fit within the gamut, the unknown weights, $\{x_j\}$, are constrained to be non-negative.

7 Regularized least-squares

Numerical techniques such as least-squares can be adversely affected by ill-conditioning. Ill-conditioning manifests itself by an acute sensitivity to the data – a slight perturbation in the data can produce a widely different solution to the problem. Ill-conditioning is usually bad news since it is not possible to completely avoid noise or numerical imprecision. *Regularization* refers to a set of techniques whose goal is to improve conditioning.

For a linear-least squares problem, $\boldsymbol{Ax} = \boldsymbol{b}$, ill-conditioning (and inversely conditioning) can be measured using the *condition number*, $\kappa(\boldsymbol{A})$, of the matrix $\boldsymbol{A}$. $\kappa(\boldsymbol{A})$ is computed using:

$$\kappa(\boldsymbol{A}) = \frac{\sigma_{max}}{\sigma_{min}},$$

where σ_{max} and σ_{min} are respectively the largest and smallest singular value of $\boldsymbol{A}$. If $\kappa(\boldsymbol{A})$ is large (e.g. $\kappa(\boldsymbol{A}) \gg 1$) then the problem is ill-conditioned.

Considering the SVD a linear system $\boldsymbol{Ax} = \boldsymbol{b}$ can show how poor conditioning can result in large changes to the solution when the problem changes only slightly. The SVD representation is $\boldsymbol{UDV}^T\boldsymbol{x} = \boldsymbol{b}$ (see appendix A). First pre-multiply the system by $\boldsymbol{U}^T$, obtaining $\boldsymbol{DV}^T\boldsymbol{x} = \boldsymbol{U}^T\boldsymbol{b}$, and then define some new variables $\boldsymbol{y} \equiv \boldsymbol{V}^T\boldsymbol{x}$ and $\boldsymbol{c} \equiv \boldsymbol{U}^T\boldsymbol{b}$. This gives the related system $\boldsymbol{Dy} = \boldsymbol{c}$. Here $\boldsymbol{y}$ is not the original solution, but it has the same norm as the original solution because multiplication by the orthogonal matrix $\boldsymbol{V}^T$ does not change the norm. If the last singular value is a small value like 0.0001, the last row of this related system will look like

$$0.0001 d_1 = c_1$$

so a change $c_1 \to c_1 + \varepsilon$ in the last element of $\boldsymbol{c}$ will result in the last element of the solution, d_1, changing by an amount 10000ε. The actual solution $\boldsymbol{x}$ is obtained from $\boldsymbol{c}$ by application of the orthogonal matrix $\boldsymbol{V}$, so this large change is distributed unpredictably among all the elements of the solution.

Example of ill-conditioned problem. Scattered data interpolation is a popular technique in computer graphics. Its goal is to approximate a smooth function given a limited set of (scattered) samples. For instance, in [7] and [36], it is used to model implicit surfaces from sample points. In [30] and [32], it is used for interpolating a sparse displacement field to animate or deform a facial model.

In all these examples, we are given a set of 3-dimensional point or data sites $\{\boldsymbol{p}_i\}$ and at each of these points, we know the values, $\{\boldsymbol{f}(\boldsymbol{p}_i)\}$, of a function $\boldsymbol{f}$ we are trying to approximate. Using radial basis functions we can compute an approximation, $\tilde{\boldsymbol{f}}$, such that:

$$\tilde{\boldsymbol{f}}(\boldsymbol{p}) = \sum_{i=1}^{n} \lambda_i \phi(\|\boldsymbol{p} - \boldsymbol{p}_i\|),$$

where ϕ is a kernel function (for instance the thin-plate or biharmonic kernel $\phi(\boldsymbol{p}) = \|\boldsymbol{p}\|^2 log(\|\boldsymbol{p}\|)$) (in the 2D case) and the λ_i are a set of weight vectors (they have as many dimensions as the function $\boldsymbol{f}$). The weights can be estimated by solving the least-squares problem:

$$\min_{\{\lambda_i\}} \sum_{j=1}^{n} \|\boldsymbol{f}(\boldsymbol{p}_j) - \tilde{\boldsymbol{f}}(\boldsymbol{p}_j)\|^2 = \min_{\{\lambda_i\}} \sum_{j=1}^{n} \|\boldsymbol{f}(\boldsymbol{p}_j) - \sum_{i=1}^{n} \lambda_i \phi(\|\boldsymbol{p}_j - \boldsymbol{p}_i\|)\|^2.$$

This yields the square linear system of equations:

$$\begin{bmatrix} \phi(\|\boldsymbol{p}_1 - \boldsymbol{p}_1\|) & \dots & \phi(\|\boldsymbol{p}_1 - \boldsymbol{p}_n\|) \\ \vdots & & \vdots \\ \phi(\|\boldsymbol{p}_n - \boldsymbol{p}_1\|) & \dots & \phi(\|\boldsymbol{p}_n - \boldsymbol{p}_n\|) \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_n \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}(\boldsymbol{p}_1) \\ \vdots \\ \boldsymbol{f}(\boldsymbol{p}_n) \end{bmatrix}$$

If we examine the matrix in the system above, we quickly realize that things can turn sour if two data sites p_{i_0} and p_{i_1} are very close spatially since two rows of the matrix are nearly the same and the matrix is then ill-conditioned or rank deficient. In the limit adding an extra row in the matrix should not be problematic per say but if the values $\boldsymbol{f}(p_{i_0})$ and $\boldsymbol{f}(p_{i_1})$ differ because of noise then finding a solution to the system becomes difficult. On one hand we are looking for an approximation in a space of smooth functions, on the other hand the data indicates that the function is not smooth. This contradiction needs to be resolved. In particular, since the data is corrupted with error, it is not necessary for the approximation to interpolate the data.

Since errors in the data can flood their solutions, ill-conditioned problems are bad news. In what follows, we examine different regularization for improving least-squares problem conditioning.

The Gauss-Markov theorem (Section 3) tells us that the least-squares estimator is the unbiased linear estimator with minimum variance. When $\boldsymbol{A}$ is ill-conditioned, this variance is still large. The variance can be reduced if the estimator is allowed to be biased. This is the approach taken by regularization techniques: accept some bias to reduce the variance.

Note that all these techniques requires choosing regularization or damping parameters. There are published techniques that can guide us in this choice. Many of them require some knowledge or assumptions of the noise distribution. Often it may be best to experiment with several approaches. In what follows we briefly mention some of these techniques.

7.1 Truncated SVD

One simple way to improve conditioning is to lower the condition number of the matrix by manipulating its Singular Value Decomposition. The SVD of $\boldsymbol{A}$ (see Appendix A) is:

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T,$$

where $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthogonal matrices and $\boldsymbol{D}$ is the diagonal matrix of singular values ($\boldsymbol{D} = diag(\sigma_1 \dots, \sigma_2)$). We can rewrite this equation using the columns vectors of $\boldsymbol{U}$

and $\boldsymbol{V}$ ($\boldsymbol{u}_i$ and $\boldsymbol{v}_i$ respectively):

$$\boldsymbol{UDV}^T = \left[\begin{array}{c|c|c} \boldsymbol{u}_1 & \dots & \boldsymbol{u}_n \end{array}\right] \begin{bmatrix} \sigma_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_n \end{bmatrix} \left[\begin{array}{c|c|c} \boldsymbol{v}_1 & \dots & \boldsymbol{v}_n \end{array}\right]^T$$

From which we derive:

$$\boldsymbol{A} = \sum_{i=1}^{n} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T.$$

In other words, the SVD allows the reconstruction of the matrix $\boldsymbol{A}$ as a weighted sum of rank 1 matrices ($\boldsymbol{u}_i\boldsymbol{v}_i^T$). If we assume that the singular values are sorted in non-increasing order (i.e. $\sigma_1 \geq \ldots \geq \sigma_n$), we can truncate the reconstruction at index k and approximate $\boldsymbol{A}$ by the matrix $\boldsymbol{A}_k$:

$$\boldsymbol{A} \simeq \sum_{i=1}^{k} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T.$$

$\boldsymbol{A}_k$ is the best rank k approximation of the matrix $\boldsymbol{A}$. In other words, $\boldsymbol{A}_k$ is the minimum

$$\min_{\boldsymbol{M}} \|\boldsymbol{A} - \boldsymbol{M}\|_F, \text{ with } rank(\boldsymbol{M}) = k,$$

where $\|.\|_F$ denotes the Frobenius norm ($\|\boldsymbol{M}\|_F = \sqrt{\sum_{i,j} m_{i,j}^2}$). The condition number of $\boldsymbol{A}_k$ is $\kappa(\boldsymbol{A}_k) = \frac{\sigma_{max}(\boldsymbol{A})}{\sigma_k(\boldsymbol{A})}$ and since $\sigma_k(\boldsymbol{A}) \geq \sigma_{max}(\boldsymbol{A})$ then $\kappa(\boldsymbol{A}_k) \leq \kappa(\boldsymbol{A})$, so using $\boldsymbol{A}_k$ yields a better conditioned system.

Choosing the truncation value. Choosing the optimal value for k can be challenging. For instance, using the concept of the numerical rank of a matrix and techniques from perturbation analysis [18] define a very complex criteria for k. A more simple criteria [20] is to pick k such that it is the smallest integer such that:

$$\|\boldsymbol{b} - \boldsymbol{A}\hat{\boldsymbol{x}}\| = \sum_{i=k+1}^{n} (U^T b)_i^2 < n.$$

The number n in the above formula arise from the expectation of the Chi-squared distribution with n degrees of freedom. This distribution governs the sum of squared residuals under a normal (i.e. Gaussian) error distribution assumption.

The truncated SVD technique is useful for addressing ill-conditioning but provides little choice for controlling how the solution is determined. The techniques presented next offer much more opportunities for tweaking.

7.1.1 Example: skin weights computation from examples

The goal of *skinning* is to deform a surface (e.g. a human body) according to the motion of its skeleton. Traditionally, the skeleton is a collection of bones represented

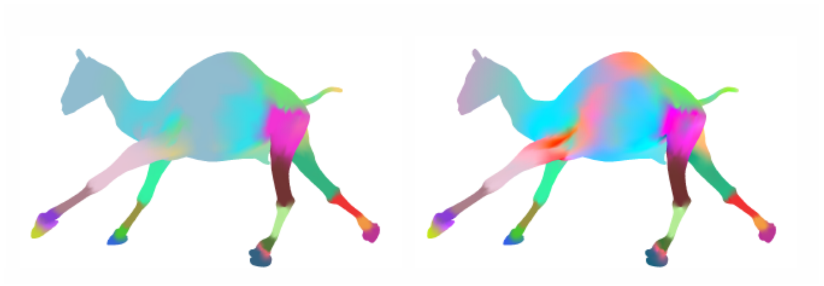

Figure 10: Example of regularization from James and Twigg [24]. Visualization of skin weights on a camel model. On the right the weights were computed using truncated SVD, on the left they were computed by solving a non-negative least-squares problem.

by (usually rigid) transformations, $\{\boldsymbol{T}_b\}$ and each vertex $\boldsymbol{v}_i$ is associated with a set of bones indexes B_i. The animated position of a vertex, $\boldsymbol{v}_i(t)$, is computed by linearly blending the influence of each bone:

$$\boldsymbol{v}_i(t) = \sum_{b \in B} w_{i,b} \boldsymbol{T}_b(t) \boldsymbol{v}_i,$$

where $w_{i,b}$ (called skin weight) is a scalar that tells how much influence bone b as on vertex i. The skin weights are traditionnaly specified using a paint program. An alternative is to derive or learn the skin weights from a set of deformed meshes associated to specific skeleton configuration. This is the approach taken by [27, 24]. To see how this problem is set up, let us call $\boldsymbol{v}_i^s$ the position of vertex i in example mesh s and similarly, $\boldsymbol{T}_b^s$, the transform associated with bone b in the configuration of mesh s. The problem is to find weights, $\{w_{i,b}\}$ that are solutions of:

$$\min_{\{w_{i,b}\}} \sum_s \| \sum_{b \in B} w_{i,b} \boldsymbol{T}_b^s \boldsymbol{v}_i - \boldsymbol{v}_i^s \|^2$$

For each vertex i, its skin weights, $\boldsymbol{w}^i$, are thus determined as the solution of a least-squares problem, $A^i \boldsymbol{w}^i = \boldsymbol{v}^i$. Ill-conditioning can easily creep in this problem. In particular, it will be the case if we introduce , in the example set, meshes that are quite different but are associated with similar skeleton poses. To remedy this problem, [24] uses two strategies. The first strategy is to use a truncated SVD (with a singular value threshold of $10^{-5}\|\boldsymbol{A}^i\|_2$ [3]). The second strategy is to impose a non-negative constraint on each weight. Figure 10 provides a visualization of the weights estimated with these two techniques.

[3]Note that for a matrix $\boldsymbol{M}$, $\|\boldsymbol{M}\|_2 = max_{\boldsymbol{x} \neq \boldsymbol{0}} \frac{\|\boldsymbol{Mx}\|_2}{\|\boldsymbol{x}\|_2} = \sigma_{min}$

7.2 Damped least-squares and Tikhonov regularization

The basic idea behind damping is to add terms to the penalty function to avoid certain solutions. For instance, because ill conditioning causes large values in the solution, we might want to add a preference that the solution be small using a modified least-squares criterion such as:

$$\min_{\boldsymbol{x}}(\|\boldsymbol{b}-\boldsymbol{A}\boldsymbol{x}\|^2+\lambda\|\boldsymbol{x}\|^2),$$

where λ is a scalar.

This type of technique is called *damped least-squares* or *Tikhonov regularization*. It use a penalty function of the general form:

$$\min_{\boldsymbol{x}}(\|\boldsymbol{b}-\boldsymbol{A}\boldsymbol{x}\|^2+\lambda\|\boldsymbol{L}\boldsymbol{x}\|^2),$$

where $\boldsymbol{L}$ is a matrix that allows control over the damping. The minimization problem yields the system of equations:

$$(\boldsymbol{A}^T\boldsymbol{A}+\lambda\boldsymbol{L}^T\boldsymbol{L})\boldsymbol{x}=\boldsymbol{A}^T\boldsymbol{b}.$$

For instance, if we choose $\lambda = 1$ and $\boldsymbol{L} = diag(l_1,\ldots,l_n)$ to selectively damp each parameter in the solution, the solution can computed using the SVD decomposition:

$$\hat{\boldsymbol{x}}=\sum_{i=1}^{n}\frac{\sigma_i}{\sigma_i^2+l_i^2}\boldsymbol{u}_i^T\boldsymbol{b}\boldsymbol{v}_i,$$

Another choice is to use a bi-diagonal matrix L that approximates the first derivative operator:

$$\boldsymbol{L}=\begin{bmatrix} -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{bmatrix}$$

This formulation is sometime used in image processing for enforcing smoothness in the solution by limiting the magnitude of the spatial derivatives (a Laplacian term can also be used).

Choosing the damping parameters. Several techniques have been proposed for selecting a damping parameter. Some of these assume that the variance of the noise is known, such as the discrepancy principle [22]. Other techniques such as generalized cross-validation [17] and L-curve [21] are applicable with less knowledge of the noise. Generalized cross-validation defines the optimal damping parameter as the minimum of a the function $\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|^2/dof^2$, where dof is the effective number of degrees of freedom. The L-curve technique is based on plotting the norm of the solution, $\|\hat{\boldsymbol{x}}\|$, against the norm of the residual, $\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|$. The parameter λ is then selected at the point of maximum curvature. That corner corresponds to a change in behavior of the system. Finally and more recently, [31] proposes a selection criteria based on minimizing an approximation of the error between the regularized solution and the (unknown) true solution.

7.2.1 Example: inverse kinematics.

Ill-conditioning arises naturally in problem like inverse kinematics where we are try to specify the joint angles along a kinematic chain (e.g. an arm) using the target position of an end effector (e.g. a hand). Following the notations of [5], if $\boldsymbol{p}$ is the position of the target, $\boldsymbol{q}$ is the vector of joint angles, and $\boldsymbol{J}$ is the matrix of partial derivatives ($\frac{\partial \boldsymbol{p}_i}{\partial \boldsymbol{q}_j}$, then applying the chain rule yields:

$$\dot{\boldsymbol{p}} = \boldsymbol{J}\dot{\boldsymbol{q}},$$

with $\dot{\boldsymbol{p}}$ and $\dot{\boldsymbol{q}}$ being respectively the derivatives of $\boldsymbol{p}$ and $\boldsymbol{q}$ with respect to time. The previous equation is a linearized approximation of the true inverse problem, $\boldsymbol{p} = f(\boldsymbol{q})$. It can be solved once at every frame, in such case the end effector only approximately follows the target, or iteratively until the end effector is close enough to the target. The least squares solution is:

$$\hat{\dot{\boldsymbol{q}}}_{LSE} = (\boldsymbol{J}^T\boldsymbol{J})^{-1}\boldsymbol{J}^T\dot{\boldsymbol{q}}.$$

Unfortunately this problem is in general under underconstrained (e.g. specifying the position of the hand does not constrain the elbow) and small changed in the target position can result in large changes along the cinematic chain. To address this problem [37, 28] damp the system to penalize large angle variations in the error function and minimize:

$$\min_{\dot{\boldsymbol{q}}} \|\boldsymbol{J}\dot{\boldsymbol{q}} - \dot{\boldsymbol{p}}\|^2 + \lambda\|\dot{\boldsymbol{q}}\|^2.$$

7.2.2 Example: surface correspondences.

We saw that ill-conditioning arises in sparse data interpolation problems. [30] uses sparse interpolation to derive a dense mapping between two meshes from a sparse set of correspondences. First, the corrrespondances are interpolated using Radial Basis Functions. The interpolated function is used to warp the source mesh toward the target mesh. Finally the warped source mesh is cylindrically projected onto the target mesh. Given a set of samples, $\{\boldsymbol{p}_i, \boldsymbol{d}_i\}$, the interpolating warping function $\tilde{\boldsymbol{d}}$ is expressed as:

$$\tilde{\boldsymbol{d}}(\boldsymbol{p}) = \sum_{i=1}^{n} \boldsymbol{w}_i \phi_i(\|\boldsymbol{p} - \boldsymbol{p}_i\|),$$

where ϕ_i is an adaptive kernel of the form:

$$\phi_i(h) = \sqrt{h^2 + s_i^2}.$$

The parameter s_i is used to tune the influence of the kernel to the density of the sample data. It is chosen as the distance to the closest neighboring data site (i.e. $s_i = \min_{i \neq j} \|\boldsymbol{p}_i - \boldsymbol{p}_j\|$).

Fitting the RBF model is then done by solving the minimization problem:

$$\begin{aligned} & \min_{\{\boldsymbol{w}_i\}} \sum_{j=1}^{n} (\|\tilde{\boldsymbol{d}}(\boldsymbol{p}_j) - \boldsymbol{d}_j\|^2 + \lambda \|\boldsymbol{w}_j\|^2) \\ = \quad & \min_{\{\boldsymbol{w}_i\}} \sum_{j=1}^{n} (\| \sum_{i=1}^{n} \boldsymbol{w}_i \phi_i(\|\boldsymbol{p}_j - \boldsymbol{p}_i\|) - \boldsymbol{d}_j\|^2 + \lambda \|\boldsymbol{w}_j\|^2) \end{aligned}$$

The regularization parameter λ is estimated by minimizing the Generalized Cross-Validation function:

$$GCV(\lambda) = \frac{\sum_{j=1}^{n} \|\tilde{\boldsymbol{d}}(\boldsymbol{p}_j) - \boldsymbol{d}_j\|^2}{(n-\gamma)^2},$$

where n is the number of correspondences and γ is the effective number of parameters ($\gamma = n - \lambda tr(\boldsymbol{A}^{-1})$). Since $\{\boldsymbol{w}_i\}$ and λ depends on each other, the estimation proceeds iteratively by alternatively estimating the two quantities.

7.2.3 Example: image registration.

Many image processing problems are ill-posed and need to be regularized. This can be done by imposing a roughness penalty on the solution. For instance, in super-resolution reconstruction [10] the goal is to produce high-resolution images from a set of lower resolution images. The difficulty is to register the lower resolution images. This is usually done assuming a linear motion. Let us assume, n low resolution images $\{\boldsymbol{y}_i\}$ that we would like to fuse into a high-res image $\boldsymbol{x}$. If we represent the images as column vectors sorted in lexicographic order then we can represent linear operators on images as matrices operating on these vectors. In particular, we will assume that the relationship between $\boldsymbol{y}_i$ and $\boldsymbol{x}$ is linear:

$$\boldsymbol{y}_i = \boldsymbol{D}_i \boldsymbol{H}_i \boldsymbol{F}_i \boldsymbol{x} + \boldsymbol{v}_i$$

where $\boldsymbol{F}_i$ is the geometric motion operator from $\boldsymbol{x}$ to $\boldsymbol{y}_i$, $\boldsymbol{H}_i$ is a blur matrix, $\boldsymbol{D}_i$ is a downsampling matrix (or decimator operator), and $\boldsymbol{v}_i$ is a noise vector. The unknown high-res image can be estimated as the solution of the least squares problem:

$$\min_{\boldsymbol{x}} \sum_{i=1}^{n} \|\boldsymbol{y}_i - \boldsymbol{D}_i \boldsymbol{H}_i \boldsymbol{F}_i \boldsymbol{x}\|^2 + \lambda \|\boldsymbol{L}\boldsymbol{x}\|^2$$

where the regularization matrix $\boldsymbol{L}$ is built by replication of the Laplacian kernel:

$$\boldsymbol{L}_{kernel} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

The regularization term enforces spatial smoothness since it penalizes solutions that have large Laplacians.

7.3 Quadratic constraints

The regularization methods introduced in the previous subsection are formulated by adding a penalty term of the form $\lambda\|\boldsymbol{Lx}\|^2$. Another approach is to use a similar expression to define a bounded problem:

$$\min_{\boldsymbol{x}} \|\boldsymbol{b}-\boldsymbol{Ax}\|^2 \text{ subject to } \|\boldsymbol{Lx}\| \leq \gamma,$$

where γ is a scalar defining a region of acceptable solutions.

The constraint $\|\boldsymbol{Lx}\| \leq \gamma$ defines a feasible set among which we must choose the solution. This problem is quite different from the penalty approach described in the previous subsection. The quadratic expression defines a hard constraint whereas the penalty does not.

From a mathematical point of view, this is an instance of a *least-squares with quadratic inequality constraint* (LSQI) problem whose general form is:

$$\min_{\boldsymbol{x}} \|\boldsymbol{b}-\boldsymbol{Ax}\|^2 \text{ subject to } \|\boldsymbol{Lx}-\boldsymbol{d}\| \leq \gamma.$$

Clearly the problem only has a solution if $min_x\|\boldsymbol{Lx}-\boldsymbol{d}\| \leq \gamma$. If the previous condition holds, then there are two cases: either the solution $\boldsymbol{x}_{int} = min_{\boldsymbol{x}}\|\boldsymbol{Ax}-\boldsymbol{b}\|$ is within the feasible set (i.e. $\|\boldsymbol{Lx}_{int}-\boldsymbol{d}\| \leq \gamma$ and $\boldsymbol{x}_{int}$ is the solution, or it is not and we have to look for a solution at the boundary of the set:

$$\min_{\boldsymbol{x}} \|\boldsymbol{b}-\boldsymbol{Ax}\|^2 \text{ subject to } \|\boldsymbol{Lx}-\boldsymbol{d}\| = \gamma.$$

This type of problem can be analyzed using Lagrange multipliers (see Section 6). The resulting normal equation is:

$$(\boldsymbol{A}^T\boldsymbol{A}+\lambda\boldsymbol{L}^T\boldsymbol{L})\boldsymbol{x} = \boldsymbol{A}^T\boldsymbol{b}+\lambda\boldsymbol{L}^t\boldsymbol{d},$$

where λ is the Lagrange multiplier determined by the constraint:

$$\|\boldsymbol{Lx}-\boldsymbol{d}\| = \gamma$$

8 Non-linear least squares

So far, we have focused on problems of the form:

$$\min_{\boldsymbol{x}} \|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}\|^2 = \min_{\boldsymbol{x}} \sum_k (b_k - \boldsymbol{a}_k^T \boldsymbol{x})^2,$$

where $\boldsymbol{A}$ is a matrix with row vectors $\{\boldsymbol{a}_k\}$. From a model fitting perspective the linear least squares formulation assumes a linear model, $\boldsymbol{x}$, and fits it to the data points $\{(\boldsymbol{a}_k, b_k)\}$. The linear assumption, albeit convenient, is often not practical and we often found ourselves considering non-linear models. Let f be such non-linear model. Keeping consistent with our notation, f is a non-linear function from $\Re^n$ to $\Re$. Also call $\boldsymbol{x}$ a vector of parameters and $\{(\boldsymbol{a}_k, b_k)\}$ a set of inputs and outputs such that $f(\boldsymbol{a}_k, \boldsymbol{x}) = b_k$ for all k. Fitting this non-linear model to the data can be formulated as:

$$\min_{\boldsymbol{x}} \sum_k (b_k - f(\boldsymbol{a}_k, \boldsymbol{x}))^2.$$

We can thus write the non-linear error function as $E(\boldsymbol{x}) = \sum_k (b_k - f(\boldsymbol{a}_k, \boldsymbol{x}))^2$ and the solution as $\hat{\boldsymbol{x}}_{LSE} = min_{\boldsymbol{x}} E(\boldsymbol{x})$. We will call the jth residual $r_j = b_j - f(\boldsymbol{a}_j, \boldsymbol{x})$ and the vector of residuals $\boldsymbol{r} = (r_1, \ldots, r_m)$.

Since we will be using derivatives, we can express the first and second order derivatives of E as a function of the derivatives of $\boldsymbol{r}$ and in particular its Jacobian matrix, $\boldsymbol{J} = \left[\frac{dr_j}{dx_i}\right]$. Using the chain rule, derive:

$$\begin{aligned} \nabla E &= \sum_{j=1}^{m} r_j \nabla r_j, \\ &= \boldsymbol{J}^T \nabla r \\ \boldsymbol{H}_E &= \sum_{j=1}^{m} \nabla r_j \nabla r_j^T + \sum_{j=1}^{m} r_j \boldsymbol{H}_{r_j}, \\ &= \boldsymbol{J}^T \boldsymbol{J} + \sum_{j=1}^{m} r_j \boldsymbol{H}_{r_j} \end{aligned}$$

8.1 Characterization and uniqueness of a solution

In general a local minima, $\boldsymbol{x}_{min}$ of E is characterized by the two conditions:

$$\begin{aligned} \nabla E(\boldsymbol{x}_{min}) &= \boldsymbol{0} \\ \boldsymbol{h}^T \boldsymbol{H}_E(\boldsymbol{x}_{min}) \boldsymbol{h} &\geq \boldsymbol{0}, \text{ for all } \boldsymbol{h} \in N(\boldsymbol{x}_{min}) \end{aligned}$$

where $\boldsymbol{H}_E$ is the Hessian matrix of E (e.g. $\boldsymbol{H}_E = (\frac{\partial E}{\partial x_i \partial x_j})$) and $N(\boldsymbol{x}_{min})$ an open neighborhood of $\boldsymbol{x}_{min}$. Looking at the second order Taylor series expansion of E at $\boldsymbol{x}$ gives us some insights on these conditions:

$$\begin{aligned} E(\boldsymbol{x}_{min}+\boldsymbol{h}) &\approx E(\boldsymbol{x}_{min})+\nabla E(\boldsymbol{x}_{min})\boldsymbol{h}+\frac{1}{2}\boldsymbol{h}^T\boldsymbol{H}_E(\boldsymbol{x}_{min})\boldsymbol{h} \\ &\approx E(\boldsymbol{x}_{min})+\frac{1}{2}\boldsymbol{h}^T\boldsymbol{H}_E(\boldsymbol{x}_{min})\boldsymbol{h} \end{aligned}$$

In other words, locally around $\boldsymbol{x}_{min}$ the function E looks like an inverted bowl centered at $\boldsymbol{x}_{min}$.

For smooth functions, these conditions define local minima. They also define a global minimum if the function, E, is convex.

The first two algorithms we cover, steepest descent and Newton's method, are not specific to least-squares problems but rather they can be applied to general minimization problems. They are not often employed in practice, however, they are often used to define or measure other algorithms.

8.2 Iterative descent algorithms

When the function f is non linear, in most cases we cannot solve directly for a solution. Instead, most non-linear minimization technique are iterative and proceed by computing a sequence $\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_l$ such that $f(\boldsymbol{x}_0) \geq f(\boldsymbol{x}_1) \geq \ldots \geq f(\boldsymbol{x}_l)$. The idea is that the algorithms progress toward the solution by doing a local search that bring them closer at each iteration. At the end of each iteration, the conditions for a local minimum are used as stopping criteria.

The local search is often done by selecting a descent direction $\boldsymbol{h}$ and choosing $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha\boldsymbol{h}$. A vector $\boldsymbol{h}$ is a descent direction if there is a scalar α such that $f(\boldsymbol{x}_k + \alpha\boldsymbol{h}) < f(\boldsymbol{x}_k)$. The descent directions can also be characterized by the condition: $\boldsymbol{h}^T\nabla f(\boldsymbol{x}_k) < 0$.

Non-linear least-squares algorithm are usually characterized by their convergence rate which is a measure of how fast it converges toward the solution as a function of the number of iterations. Also keep in mind that choosing a "good" starting point, $\boldsymbol{x}_0$, that is to say one that is close to the solution can significantly speed up finding a solution.

8.3 Steepest descent

The iteration of the steepest descent algorithm is based on following the direction of steepest descent, $-\frac{\nabla E(\boldsymbol{x}_k)}{\|\nabla E(\boldsymbol{x}_k)\|}$, such that the iteration is:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha\nabla E(\boldsymbol{x}_k),$$

where α is a scalar chosen so that $E(\boldsymbol{x}_{k+1})$ is as small as possible (finding such value is in general far from trivial and is the object of *line search algorithms*. The convergence rate of the steepest descent algorithm is linear and often very slow.

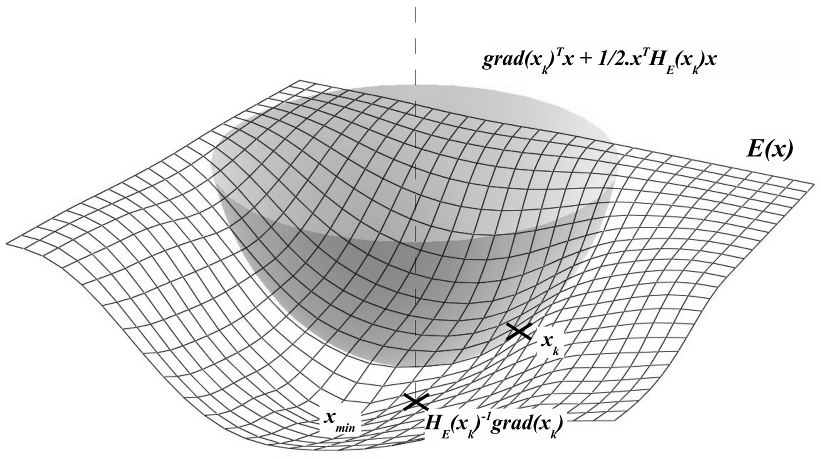

Figure 11: Geometry of Newton's method. The next step $\boldsymbol{x}_{k+1}$ is obtained by minimizing a quadratic approximation of the error function at $\boldsymbol{x}_k$.

8.4 Newton's method

Newton's method iterates by using a second order approximation of the error function, $\tilde{E}$:

$$\tilde{E}(\boldsymbol{x}_k+\boldsymbol{h}) = E(\boldsymbol{x}_k) + \nabla E(\boldsymbol{x}_k)\boldsymbol{h} + \frac{1}{2}\boldsymbol{h}^T\boldsymbol{H}_E(\boldsymbol{x}_k)\boldsymbol{h}$$

Minimizing this expression with respect to $\boldsymbol{h}$, yields:

$$\frac{d\tilde{E}(\boldsymbol{x}_k+\boldsymbol{h})}{d\boldsymbol{h}} = \nabla E(\boldsymbol{x}_k) + \boldsymbol{H}_E(\boldsymbol{x}_k)\boldsymbol{h} = \boldsymbol{0},$$

from which we derive $\boldsymbol{h} = \boldsymbol{H}_E(\boldsymbol{x}_k)^{-1}\nabla E(\boldsymbol{x}_k)$. Now, we would like to choose $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{h}$ since it seems that in this case $\boldsymbol{x}_{k+1}$ would minimize a local second order approximation of $E(\boldsymbol{x})$. Unfortunately, $\boldsymbol{x}_k + \boldsymbol{h}$ is a minima of the approximation only if the curvature at this point is positive or in other words if $\boldsymbol{H}_E(\boldsymbol{x}_k)$ is positive semi-definite. If the function is convex over the domain we are minimizing over, then it is always true; if not, we cannot guarantee that $\boldsymbol{h}$ is a descent direction.

For this reason and also because each step requires solving a system of equations, *quasi-Newton's methods*, that replace the Hessian matrix by a positive definite approximation, are often preferred. Newton's method has a quadratic convergence rate.

8.5 Gauss-Newton method

The Gauss-Newton method is a simplification of Newton's algorithm with line search. Recall that in Newton's method iterations are computed using a step vector $\boldsymbol{h}$ that is solution of the system of equations:

$$\boldsymbol{H}_E(\boldsymbol{x}_k)\boldsymbol{h} = \nabla E(\boldsymbol{x}_k),$$

using the derivatives of $\boldsymbol{r}$ we obtain:

$$(\boldsymbol{J}^T(\boldsymbol{x}_k)\boldsymbol{J}(\boldsymbol{x}_k) + \sum_{j=1}^{m} r_j(\boldsymbol{x}_k)\boldsymbol{H}_{r_j}(\boldsymbol{x}_k))\boldsymbol{h} = \boldsymbol{J}^T(\boldsymbol{x}_k)\nabla r(\boldsymbol{x}_k),$$

finally if we neglect the terms with second order derivatives, $\sum_{j=1}^{m} r_j(\boldsymbol{x}_k)\boldsymbol{H}_{r_j}(\boldsymbol{x}_k)$, the previous expression becomes:

$$\boldsymbol{J}^T(\boldsymbol{x}_k)\boldsymbol{J}(\boldsymbol{x}_k)\boldsymbol{h} = \boldsymbol{J}^T(\boldsymbol{x}_k)\nabla r(\boldsymbol{x}_k)$$

In practice, in many cases the term $\boldsymbol{J}^T\boldsymbol{J}$ dominates the Hessian, so that the Gauss-Newton method exhibits performances similar to Newton's method. Notice also that the approximation is justified when the residuals are small which is the case near the minimum for many problems (interestingly adding a constant to the residual would alter this property but not change the position of the minimum $\hat{\boldsymbol{x}}$).

The advantages of this simplification are two-folds. First, second-order derivatives are not required. Second, if $\boldsymbol{J}$ has full-rank and ∇E is non-zero, then $\boldsymbol{h}$ is a descent direction and as such is a suitable direction for a line search.

8.6 Levenberg-Marquardt

Levenberg-Marquardt generalizes least squares fitting to non-linear models. As with ordinary least squares, a sum of squares is to be minimized, and the errors are assumed to be Gaussian:

$$\arg\min_{\boldsymbol{p}} \sum \left(\frac{b_k - f(\boldsymbol{a}_k, \boldsymbol{x})}{\sigma_k} \right)^2$$

where $f(\boldsymbol{a}_k, \boldsymbol{x})$ is a non-linear function of parameters $\boldsymbol{x}$ evaluated at location $\boldsymbol{a}_k$, σ_k is the assumed noise variance at location $\boldsymbol{a}_k$, and b_k is the measured data value at this location.

The popular *Numerical Recipes* book [33] has a good description of this method, but we will give an intuitive description of the basic idea here.

The Levenberg-Marquardt procedure is based on the idea a quadratic is a good approximation sufficiently close to the minimum, but may be a bad approximation far from the minimum. Thus, it "blends" between gradient descent and Newton approaches based on how well the Newton approach is performing. More specifically, it uses a step size defined by

$$(H + \lambda I)\delta = \nabla f$$

where I is the identity matrix and λ is a scalar. When λ is small, obviously the behavior is guided by the Newton approach. When λ is large, the step size becomes more proportional to the gradient, i.e., gradient descent is used. Levenberg-Marquardt dynamically varies λ, increasing it when previous steps fail to reduce the error. In addition, Levenberg-Marquardt uses an approximation to the Hessian matrix that reduces the second derivative sensitivity to noise.

Although the descriptions on *Numerical Recipes* are clear, the coding style there is not ideal, in part because of the Fortran-style array indexing and short names. One of the authors has produced a Java language Levenberg-Marquardt implementation with a more readable style (`http://www.idiom.com/~zilla/Computer/Javanumeric/index.html`); it should be easy to translate this into C/C++.

9 Conclusion

In these notes, we tried to offer an overview of the different least-squares techniques available. We also tried to illustrate these techniques through examples taken from the computer graphics community. Our hope is to expose some of the less known techniques.

It often seems that every problem can be formulated as a least-squares minimization on which we can unleash an array of numerical techniques. Our experience has taught us that it is valuable to spend some time studying the data and carefully formulating the problem to derive a robust and efficient solution. Also when working with artists who sometime lack the necessary mathematical background, it is often better to prefer techniques and solutions that are predictable and appeal to common sense. Or you might have to explain why non-convexity prevented your non-linear solver from finding the "correct" solution.

References

[1] N. Amenta and Y. J. Kil. Defining point-set surfaces. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 264–270, New York, NY, USA, 2004. ACM Press.

[2] A. Bjork. *Numerical methods for least-squares problems*. SIAM, Philiadephia, 1934.

[3] M. Black. *Robust Incremental Optical Flow*. PhD thesis, Yale, 1992.

[4] C. Bregler, L. Loeb, E. Chuang, and H. Deshpande. Turning to the masters: motion capturing cartoons. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 399–407, New York, NY, USA, 2002. ACM Press.

[5] S. R. Buss and J.-S. Kim. Selectively damped least squares for inverse kinematics. *journal of graphics tools*, 10(3):37–49, 2005.

[6] B. Le Callennec and R. Boulic. Robust kinematic constraint detection for motion data. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 281–290, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[7] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3d objects with radial basis functions. In *Proceedings of ACM SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 67–76, August 2001.

[8] E. Chuang and C. Bregler. Performance driven facial animation using blendshape interpolation. *CS-TR-2002-02, Department of Computer Science, Stanford University*, 2002.

[9] M. Eck, T. D. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. In *Proceedings of SIGGRAPH 95*, Computer Graphics Proceedings, Annual Conference Series, pages 173–182, August 1995.

[10] S. Farsiu, M. D. Robinson, M. Elad, and P. Milanfar. Fast and robust multiframe super resolution. *IEEE Transactions on image processing*, 13(10):1327– 1344, 2004.

[11] G. D. Finlayson, S. D. Hordley, and I. Tastl. Gamut constrained illuminant estimation. *Int. J. Comput. Vision*, 67(1):93–109, 2006.

[12] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[13] S. Fleishman, D. Cohen-Or, and C. T. Silva. Robust moving least-squares fitting with sharp features. *ACM Trans. Graph.*, 24(3):544–552, 2005.

[14] D. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. Prentice Hall, 2003.

[15] M. Gleicher. A graphics toolkit based on differential constraints. In *UIST '93: Proceedings of the 6th annual ACM symposium on User interface software and technology*, pages 109–120, New York, NY, USA, 1993. ACM Press.

[16] M. Gleicher and A. Witkin. Through-the-lens camera control. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 331–340, New York, NY, USA, 1992. ACM Press.

[17] G. Golub, M. Heath, and G. Wahba. Generalized cross-validation as a method for choosing good ridge parameters. *Techntronics*, 21:215–223, 1979.

[18] G. H. Golub, V. C. Klema, and G. W. Stewart. Rank degeneracy and least squares problems. Technical Report CS-TR-76-559, 1976.

[19] Y. Guo, J. Wang, H. Sun, X. Cui, and Q. Peng. A novel constrained texture mapping method based on harmonic map. *Computers & Graphics*, 29(6):972–979, December 2005.

[20] P. C. Hansen. *Rank-deficient and discrete ill-posed problems: numerical aspects of linear inversion*. SIAM, Philiadephia, 1987.

[21] P. C. Hansen. Analysis of discrete ill-posed problems by means of the l-curve. *SIAM Rev.*, 34:561–580, 1994.

[22] P. C. Hansen. *Rank-deficient and discrete ill-Posed problems: numerical aspects of linear inversion*. SIAM, Philiadephia, 1997.

[23] M. Hardy. An Illuminating Counterexample. *ArXiv Mathematics e-prints*, June 2002.

[24] D. L. James and C. D. Twigg. Skinning mesh animations. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, pages 399–407, New York, NY, USA, 2005. ACM Press.

[25] V. Kwatra, I. Essa, A. Bobick, and N. Kwatra. Texture optimization for example-based synthesis. *ACM Trans. Graph.*, 24(3):795–802, 2005.

[26] C. L. Lawson and R. J. Hanson. *Solving Least Squares Problems*. Prentice–Hall, Englewood Cliffs, NJ, 1974.

[27] A. Mohr and M. Gleicher. Building efficient, accurate character skins from examples. *ACM Transactions on Graphics*, 22(3):562–568, July 2003.

[28] Y. Nakamura and H. Hanafusa. Inverse kinematics solutions with singularity robustness for robot manipulator control. *Journal of dynamic systems, measurements, and control*, 108:163–171, 1986.

[29] A. Nealen, T. Igarashi, O. Sorkine, and M. Alexa. Laplacian mesh optimization. In *GRAPHITE '06: Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia*, pages 381–389, New York, NY, USA, 2006. ACM Press.

[30] J. Noh and U. Neumann. Expression cloning. In *Proceedings of ACM SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 277–288, August 2001.

[31] D. P. O'Leary. Near-optimal parameters for tikhonov and other regularization methods. *SIAM Journal on Scientific Computing*, 23(4):1161–1171, 2001.

[32] F. Pighin, J. Hecker, D. Lischinski, R. Szeliski, and D. H. Salesin. Synthesizing realistic facial expressions from photographs. In *Proceedings of SIGGRAPH 98*, Computer Graphics Proceedings, Annual Conference Series, pages 75–84, July 1998.

[33] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C : The Art of Scientific Computing*. Cambridge University Press, 1992.

[34] O. Sorkine and D. Cohen-Or. Least-squares meshes. In *Proceedings of Shape Modeling International*, pages 191–199. IEEE Computer Society Press, 2004.

[35] L. N. Trefethen and III D. Bau. *Numerical linear algebra*. SIAM, Philiadephia, 1997.

[36] G. Turk and J. F. O'Brien. Modelling with implicit surfaces that interpolate. *ACM Transactions on Graphics*, 21(4):855–873, October 2002.

[37] C. W. Wampler. Manipulator inverse kinematic solutions based on vector fomulations and damped least-squares methods. *IEEE Transactions on Systems, Man, and Cybernetics*, 16:93–101, 1986.

[38] D. Zhang and M. Hebert. Harmonic maps and their applications in surface matching. *cvpr*, 02:2524, 1999.

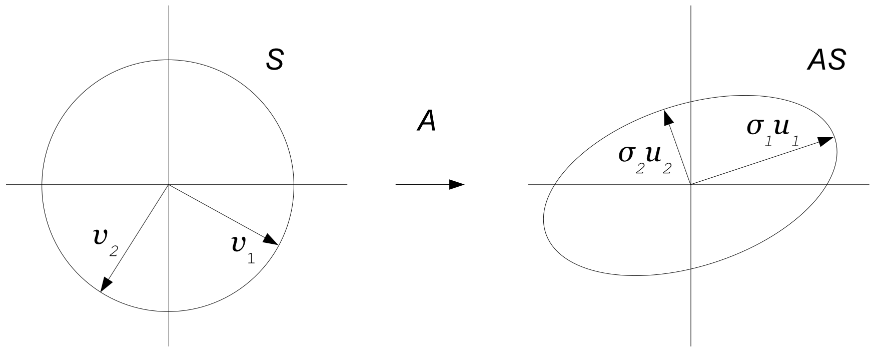

Figure 12: Geometric interpretation of the SVD for a 2×2 matrix, A.

A The Singular Value Decomposition

Matrix decompositions are tools both for solving and for gaining insight into linear algebra problems. One of the most useful decomposition is the *Singular Value Decomposition* (SVD). Consider the case $m \geq n$ which corresponds to an over-determined system of equations (the case $m \leq n$ being quite similar).

Given an $m \times n$ real matrix $\boldsymbol{A}$, the SVD of $\boldsymbol{A}$ is written:

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T,$$

where $\boldsymbol{U}$ is an $m \times m$ orthogonal [4] matrix, $\boldsymbol{V}$ is an $n \times n$ orthogonal matrix, and $\boldsymbol{D}$ is an $m \times n$ matrix of the form:

$$\boldsymbol{D} = \begin{bmatrix} \sigma_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_n \\ 0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 0 \end{bmatrix}$$

In other words the top n rows form a diagonal matrix and the bottom $m-n$ rows are filled with zeroes. The $\{\sigma_i\}$ are non-negative scalars called the *singular values* of $\boldsymbol{A}$. For ease of analysis it is conventional to assume that the singular values are sorted in non-increasing order (i.e. $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_n \geq 0$). Any matrix can be decomposed using the SVD.

In what follows, we call $\{u_i\}$ and $\{v_i\}$ the columns vectors of matrix $\boldsymbol{U}$ and $\boldsymbol{V}$ respectively.

[4]Recall that a square matrix $\boldsymbol{M}$ is orthogonal if $\boldsymbol{M}\boldsymbol{M}^T = \boldsymbol{I}$. Orthogonal matrices preserve distances (i.e. $\|\boldsymbol{Mp}\| = \|\boldsymbol{p}\|$). In 3-space they represent rotations ($det(\boldsymbol{M}) = +1$) and symmetries ($det(\boldsymbol{M}) = -1$).

Geometric interpretation. The SVD has a geometric interpretation: it describes the transformation of a unit sphere (in an n dimensional space) into a hyperellipse [5] (in an m-dimensional space). This relies on the fact that linear transformations (i.e. matrices) transform spheres into hyperellipses. With the unit sphere as input, the resulting hyperellipse has for principal semiaxes the vectors $\{\sigma_i \boldsymbol{u}_i\}$. The vectors $\{\boldsymbol{v}_i\}$ give the directions in which the sphere is stretched in the original space. Figure 12 illustrates this geometric interpretation.

Properties. We can learn a lot about a matrix once it has been decomposed. Here are some of the most useful properties:

$$\begin{aligned} |det(\boldsymbol{A})| &= \prod_{i=1}^{n} \sigma_i, \\ \|\boldsymbol{A}\|_2 &= \sigma_1, \\ \|\boldsymbol{A}\|_F &= \sqrt{\sum_{i=1}^{n} \sigma_i^2}, \end{aligned}$$

and with $r = rank(\boldsymbol{A})$:

$$\begin{aligned} range(\boldsymbol{A}) &= \langle \boldsymbol{u}_1, \ldots, \boldsymbol{u}_r \rangle, \\ null(\boldsymbol{A}) &= \langle \boldsymbol{v}_{r+1}, \ldots, \boldsymbol{v}_n \rangle \end{aligned}$$

[5]An hyperellipse is to an m dimensional space what an ellipse is to a 2-dimensional space.

B Errata

In the Constrained Least Squares subsection on Lagrange Multipliers, the final equation should be

$$\nabla f(x,y) + \lambda \nabla g(x,y) = 0$$