Chapter 3

Designing and Analyzing Linear Systems

Now that we have some methods for solving linear systems of equations, we can use them to solve a variety of problems. In this chapter, we will explore a few such applications and accompanying analytical techniques to characterize the types of solutions we can expect.

3.1 Solution of Square Systems

At the beginning of the previous chapter we made several assumptions on the types of linear systems we were going to solve. While this restriction was a nontrivial one, in fact many if not most applications of linear solvers can be posed in terms of square, invertible linear systems. We explore a few contrasting applications below.

3.1.1 Regression

We will start with a simple application appearing in data analysis known as *regression*. Suppose that we carry out a scientific experiment and wish to understand the structure of our experimental results. One way to model such an relationship might be to write the *independent variables* of the experiment in some vector $\vec{x} \in \mathbb{R}^n$ and to consider the *dependent variable* as a function $f(\vec{x}) : \mathbb{R}^n \to \mathbb{R}$. Our goal is to predict the output of f without carrying out the full experiment.

Example 3.1 (Biological experiment)**.** *Suppose we wish to measure the effect of fertilizer, sunlight, and water on plant growth. We might do a number of experiments applying different amounts of fertilizer (in cm^3), sunlight (in watts), and water (in ml) and measuring the height of the plant after a few days. We might model our observations as a function $f : \mathbb{R}^3 \to \mathbb{R}$ taking in the three parameters we wish to test and outputting the height of the plant.*

In *parametric* regression, we make a simplifying assumption on the structure of f. For example, suppose that f is linear:

$$f(\vec{x}) = a_1 x_1 + a_2 x_2 + \cdots + a_n x_n.$$

Then, our goal becomes more concrete: to estimate the coefficients a_k.

Suppose we do a series of experiments that show $\vec{x}^{(k)} \mapsto y^{(k)} \equiv f(\vec{x}^{(k)})$. By plugging into our form for f, we obtain a series of statements:

$$\begin{aligned} y^{(1)} = f(\vec{x}^{(1)}) &= a_1 x_1^{(1)} + a_2 x_2^{(1)} + \cdots + a_n x_n^{(1)} \\ y^{(2)} = f(\vec{x}^{(2)}) &= a_1 x_1^{(2)} + a_2 x_2^{(2)} + \cdots + a_n x_n^{(2)} \\ &\vdots \end{aligned}$$

Notice that contrary to our notation $A\vec{x} = \vec{b}$, the unknowns here are the a_k's, *not* the $\vec{x}$ variables. If we make n observations, we can write:

$$\begin{pmatrix} - & \vec{x}^{(1)\top} & - \\ - & \vec{x}^{(2)\top} & - \\ & \vdots & \\ - & \vec{x}^{(n)\top} & - \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{pmatrix}$$

In other words, if we carry out n trials of our experiment and write them in the columns of a matrix $X \in \mathbb{R}^{n \times n}$ and write the dependent variables in a vector $\vec{y} \in \mathbb{R}^n$, the coefficients $\vec{a}$ can be recovered by solving $X^\top \vec{a} = \vec{y}$.

In fact, we can generalize our approach to other more interesting nonlinear forms for the function f. What matters here is that f is a *linear combination* of functions. In particular, suppose $f(\vec{x})$ takes the following form:

$$f(\vec{x}) = a_1 f_1(\vec{x}) + a_2 f_2(\vec{x}) + \cdots + a_m f_m(\vec{x}),$$

where $f_k : \mathbb{R}^n \to \mathbb{R}$ and we wish to estimate the parameters a_k. Then, by a parallel derivation given m observations of the form $\vec{x}^{(k)} \mapsto y^{(k)}$ we can find the parameters by solving:

$$\begin{pmatrix} f_1(\vec{x}^{(1)}) & f_2(\vec{x}^{(1)}) & \cdots & f_m(\vec{x}^{(1)}) \\ f_1(\vec{x}^{(2)}) & f_2(\vec{x}^{(2)}) & \cdots & f_m(\vec{x}^{(2)}) \\ \vdots & \vdots & \cdots & \vdots \\ f_1(\vec{x}^{(m)}) & f_2(\vec{x}^{(m)}) & \cdots & f_m(\vec{x}^{(m)}) \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{pmatrix} = \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{pmatrix}$$

That is, even if the f's are nonlinear, we can learn weights a_k using purely linear techniques.

Example 3.2 (Linear regression)**.** *The system $X^\top \vec{a} = \vec{y}$ can be recovered from the general formulation by taking $f_k(\vec{x}) \equiv x_k$.*

Example 3.3 (Polynomial regression)**.** *Suppose that we observe a function of a single variable $f(x)$ and wish to write it as an n-th degree polynomial*

$$f(x) \equiv a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n.$$

Given n pairs $x^{(k)} \mapsto y^{(k)}$, we can solve for the parameters $\vec{a}$ via the system

$$\begin{pmatrix} 1 & x^{(1)} & (x^{(1)})^2 & \cdots & (x^{(1)})^n \\ 1 & x^{(2)} & (x^{(2)})^2 & \cdots & (x^{(2)})^n \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & x^{(n)} & (x^{(n)})^2 & \cdots & (x^{(n)})^n \end{pmatrix} \begin{pmatrix} a_0 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{pmatrix}.$$

In other words, we take $f_k(x) = x^k$ in our general form above. Incidentally, the matrix on the left hand side of this relationship is known as a Vandermonde matrix, which has many special properties specific to its structure.

Example 3.4 (Oscillation)**.** *Suppose we wish to find a and ϕ for a function $f(x) = a\cos(x+\phi)$. Recall from trigonometry that we can write $\cos(x+\phi) = \cos x\cos\phi - \sin x\sin\phi$. Thus, given two sample points we can use the technique above to find $f(x) = a_1\cos x + a_2\sin x$, and applying this identity we can write*

$$a = \sqrt{a_1^2 + a_2^2}$$
$$\phi = -\arctan\frac{a_2}{a_1}$$

This construction can be extended to finding $f(x) = \sum_k a_k\cos(x+\phi_k)$, giving one way to motivate the discrete Fourier transform of f.

3.1.2 Least Squares

The techniques in §3.1.1 provide valuable methods for finding a continuous f matching a set of data pairs $\vec{x}_k \mapsto y_k$ *exactly*. There are two related drawbacks to this approach:

- There might be some error in measuring the values $\vec{x}_k$ and y_k. In this case, an approximate relationship $f(\vec{x}_k) \approx y_k$ may be acceptable or even preferable to an exact $f(\vec{x}_k) = y_k$.
- Notice that if there were m functions f_k total, then we needed exactly m observations $\vec{x}_k \mapsto y_k$. Additional observations would have to be thrown out, or we would have to change the form of f.

Both of these issues related to the larger problem of *over-fitting*: Fitting a function with n degrees of freedom to n data points leaves no room for measurement error.

More generally, suppose we wish to solve the linear system $A\vec{x} = \vec{b}$ for $\vec{x}$. If we denote row k of A as $\vec{r}_k^\top$, then our system looks like

$$\begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} = \begin{pmatrix} - & \vec{r}_1^\top & - \\ - & \vec{r}_2^\top & - \\ \vdots & \cdots & \vdots \\ - & \vec{r}_n^\top & - \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} \vec{r}_1\cdot\vec{x} \\ \vec{r}_2\cdot\vec{x} \\ \vdots \\ \vec{r}_n\cdot\vec{x} \end{pmatrix} \text{ by definition of matrix multiplication.}$$

Thus, each row of the system corresponds to an observation of the form $\vec{r}_k\cdot\vec{x} = b_k$. That is, yet another way to interpret the linear system $A\vec{x} = \vec{b}$ is as n statements of the form, "The dot product of $\vec{x}$ with $\vec{r}_k$ is b_k."

From this viewpoint, a tall system $A\vec{x} = \vec{b}$ with $A \in \mathbb{R}^{m\times n}$ and $m > n$ simply encodes more than n of these dot product observations. When we make more than n observations, however, they may be *incompatible*; as explained §2.1, tall systems likely will not admit a solution. In our "experimental" setup explained above, this situation might correspond to errors in the measurement of the pairs $\vec{x}_k \mapsto y_k$.

When we cannot solve $A\vec{x} = \vec{b}$ exactly, we might relax the problem a bit to approximate $A\vec{x} \approx \vec{b}$. In particular, we can ask that the residual $\vec{b} - A\vec{x}$ be as small as possible by minimizing the

norm $\|\vec{b} - A\vec{x}\|$. Notice that if there is an exact solution to the linear system, then this norm is minimized at zero, since in this case we have $\|\vec{b} - A\vec{x}\| = \|\vec{b} - \vec{b}\| = 0$. Minimizing $\|\vec{b} - A\vec{x}\|$ is the same as minimizing $\|\vec{b} - A\vec{x}\|^2$, which we expanded in Example 0.16 to:

$$\|\vec{b} - A\vec{x}\|^2 = \vec{x}^\top A^\top A\vec{x} - 2\vec{b}^\top A\vec{x} + \|\vec{b}\|^2.^1$$

The gradient of this expression with respect to $\vec{x}$ must be zero at its minimum, yielding the following system:

$$\begin{aligned} \vec{0} &= 2A^\top A\vec{x} - 2A^\top\vec{b} \\ \text{Or equivalently:} \quad A^\top A\vec{x} &= A^\top\vec{b}. \end{aligned}$$

This famous relationship is worthy of a theorem:

Theorem 3.1 (Normal equations)**.** *Minima of the residual $\|\vec{b} - A\vec{x}\|$ for $A \in \mathbb{R}^{m\times n}$ (with no restriction on m or n) satisfy $A^\top A\vec{x} = A^\top\vec{b}$.*

If at least n rows of A are linearly independent, then the matrix $A^\top A \in \mathbb{R}^{n\times n}$ is invertible. In this case, the minimum residual occurs (uniquely) at $(A^\top A)^{-1}A^\top\vec{b}$, or equivalently, solving the least squares problem is as easy as solving the *square* linear system $A^\top A\vec{x} = A^\top\vec{b}$ from Theorem 3.1. Thus, we have expanded our set of solution strategies to $A \in \mathbb{R}^{m\times n}$ with $m \geq n$ by applying only techniques for square matrices.

The underdetermined case $m < n$ is considerably more difficult to deal with. In particular, we lose the possibility of a *unique* solution to $A\vec{x} = \vec{b}$. In this case we must make an additional assumption on $\vec{x}$ to obtain a unique solution, e.g. that it has a small norm or that it contains many zeros. Each such *regularizing* assumption leads to a different solution strategy; we will explore a few in the exercises accompanying this chapter.

3.1.3 Additional Examples

An important skill is to be able to identify linear systems "in the wild." Here we quickly enumerate a few more examples.

Alignment

Suppose we take two photographs of the same scene from different positions. One common task in computer vision and graphics is to stitch them together. To do so, the user (or an automatic system) may mark a number of points $\vec{x}_k, \vec{y}_k \in \mathbb{R}^2$ such that $\vec{x}_k$ in image one corresponds to $\vec{y}_k$ in image two. Of course, likely mistakes were made while matching these points, so we wish to find a stable transformation between the two images by oversampling the number of necessary pairs $(\vec{x}, \vec{y})$.

Assuming our camera has a standard lens, camera projections are linear, so a reasonable assumption is that there exists some $A \in \mathbb{R}^{2\times 2}$ and a translation vector $\vec{b} \in \mathbb{R}^2$ such that

$$\vec{y}_k \approx A\vec{x}_k + \vec{b}.$$

[1] It may be valuable to return to the preliminaries in Chapter 0 at this point for review.

Our unknown variables here are A and $\vec{b}$ rather than $\vec{x}_k$ and $\vec{y}_k$.

In this case, we can find the transformation by solving:

$$\min_{A,\vec{b}} \sum_{k=1}^{p} \|(A\vec{x}_k + \vec{b}) - \vec{y}_k\|^2.$$

This expression is once again a sum of squared linear expressions in our unknowns A and $\vec{b}$, and by a similar derivation to our discussion of the least-squares problem it can be solved linearly.

Deconvolution

Often times we accidentally take photographs that are somewhat out of focus. While a photo that is completely blurred may be a lost cause, if there is localized or small-scale blurring, we may be able to recover a sharper image using computational techniques. One simple strategy is *deconvolution*, explained below.

We can think of a photograph as a point in $\mathbb{R}^p$, where p is the number of pixels; of course, if the photo is in color we may need three values (RGB) per pixel, yielding a similar technique in $\mathbb{R}^{3p}$. Regardless, many simple image blurs are *linear*, e.g. Gaussian convolution or operations averaging pixels with their neighbors on the image. In image processing, these linear operations often have other special properties like shift-invariance, but for our purposes we can think of blurring as some linear operator $\vec{x} \mapsto G * \vec{x}$.

Suppose we take a blurry photo $\vec{x}_0 \in \mathbb{R}^p$. Then, we could try to recover the sharp image $\vec{x} \in \mathbb{R}^p$ by solving the least-squares problem

$$\min_{\vec{x} \in \mathbb{R}^p} \|\vec{x}_0 - G * \vec{x}\|^2.$$

That is, we ask that when you blur $\vec{x}$ with G, you get the observed photo $\vec{x}_0$. Of course, many sharp images might yield the same blurred result under G, so we often add additional terms to the minimization above asking that $\vec{x}_0$ not vary wildly.

3.2 Special Properties of Linear Systems

Our discussion of Gaussian elimination and the LU factorization led to a completely generic method for solving linear systems of equations. While this strategy always works, sometimes we can gain speed or numerical advantages by examining the particular system we are solving. Here we discuss a few common examples where knowing more about the linear system can simplify solution strategies.

3.2.1 Positive Definite Matrices and the Cholesky Factorization

As shown in Theorem 3.1, solving a least-squares problem $A\vec{x} \approx \vec{b}$ yields a solution $\vec{x}$ satisfying the square linear system $(A^\top A)\vec{x} = A^\top \vec{b}$. Regardless of A, the matrix $A^\top A$ has a few special properties that make this system special.

First, it is easy to see that $A^\top A$ is symmetric, since

$$(A^\top A)^\top = A^\top (A^\top)^\top = A^\top A.$$

Here, we simply used the identities $(AB)^\top = B^\top A^\top$ and $(A^\top)^\top = A$. We can express this symmetry index-wise by writing $(A^\top A)_{ij} = (A^\top A)_{ji}$ for all indices i, j. This property implies that it is sufficient to store only the values of $A^\top A$ on or above the diagonal, since the rest of the elements can be obtained by symmetry.

Furthermore, $A^\top A$ is a *positive semidefinite* matrix, as defined below:

Definition 3.1 (Positive (Semi-)Definite)**.** *A matrix $B \in \mathbb{R}^{n\times n}$ is positive semidefinite if for all $\vec{x} \in \mathbb{R}^n$, $\vec{x}^\top B\vec{x} \geq 0$. B is positive definite if $\vec{x}^\top B\vec{x} > 0$ whenever $\vec{x} \neq \vec{0}$.*

It is easy to show that $A^\top A$ is positive semidefinite, since:

$$\vec{x}^\top A^\top A\vec{x} = (A\vec{x})^\top (A\vec{x}) = (A\vec{x}) \cdot (A\vec{x}) = \|A\vec{x}\|_2^2 \geq 0.$$

In fact, if the columns of A are linearly independent, then $A^\top A$ is positive definite.

More generally, suppose we wish to solve a *symmetric positive definite* system $C\vec{x} = \vec{d}$. As we have already explored, we could LU-factorize the matrix C, but in fact we can do somewhat better. We write $C \in \mathbb{R}^{n\times n}$ as a block matrix:

$$C = \begin{pmatrix} c_{11} & \vec{v}^\top \\ \vec{v} & \tilde{C} \end{pmatrix}$$

where $\vec{v} \in \mathbb{R}^{n-1}$ and $\tilde{C} \in \mathbb{R}^{(n-1)\times(n-1)}$. Thanks to the special structure of C, we can make the following observation:

$$\begin{aligned} \vec{e}_1^\top C\vec{e}_1 &= \begin{pmatrix} 1 & 0 & \cdots & 0 \end{pmatrix} \begin{pmatrix} c_{11} & \vec{v}^\top \\ \vec{v} & \tilde{C} \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 & \cdots & 0 \end{pmatrix} \begin{pmatrix} c_{11} \\ \vec{v} \end{pmatrix} \\ &= c_{11} \\ &> 0 \text{ since } C \text{ is positive definite and } \vec{e}_1 \neq \vec{0}. \end{aligned}$$

This shows that–ignoring numerical issues– we do not have to use pivoting to ensure that $c_{11} \neq 0$ for the first step Gaussian elimination.

Continuing with Gaussian elimination, we can apply a forward substitution matrix E, which generically has the form

$$E = \begin{pmatrix} 1/\sqrt{c_{11}} & \vec{0}^\top \\ \vec{r} & I_{(n-1)\times(n-1)} \end{pmatrix}.$$

Here, the vector $\vec{r} \in \mathbb{R}^{n-1}$ contains the multiples of row 1 to cancel the rest of the first column of C. We also scale row 1 by $1/\sqrt{c_{11}}$ for reasons that will become apparent shortly!

By design, after forward substitution we know the product

$$EC = \begin{pmatrix} \sqrt{c_{11}} & \vec{v}^\top/\sqrt{c_{11}} \\ \vec{0} & D \end{pmatrix}$$

for some $D \in \mathbb{R}^{(n-1)\times(n-1)}$.

Here's where we diverge from Gaussian elimination: rather than moving on to the second row, we can post-multiply by $E^\top$ to obtain a product $ECE^\top$:

$$\begin{aligned} ECE^\top &= (EC)E^\top \\ &= \begin{pmatrix} \sqrt{c_{11}} & \vec{v}^\top / \sqrt{c_{11}} \\ \vec{0} & D \end{pmatrix} \begin{pmatrix} 1/\sqrt{c_{11}} & \vec{r}^\top \\ \vec{0} & I_{(n-1)\times(n-1)} \end{pmatrix} \\ &= \begin{pmatrix} 1 & \vec{0}^\top \\ \vec{0} & \tilde{D} \end{pmatrix} \end{aligned}$$

That is, we have eliminated the first row *and* the first column of C! Furthermore, it is easy to check that the matrix $\tilde{D}$ is also positive definite.

We can repeat this process to eliminate all the rows and columns of C symmetrically. Notice that we used both symmetry and positive definiteness to derive the factorization, since

- symmetry allowed us to apply the same E to both sides, and
- positive definiteness guaranteed that $c_{11} > 0$, thus implying that $\sqrt{c_{11}}$ exists.

In the end, similar to LU factorization we now obtain a factorization $C = LL^\top$ for a lower triangular matrix L. This is known as the *Cholesky factorization* of C. If taking the square roots along the diagonal causes numerical issues, a related $LDL^\top$ factorization, where D is a diagonal matrix, avoids this issue and is straightforward to derive from the discussion above.

The Cholesky factorization is important for a number of reasons. Most prominently, it takes half the memory to store L than the LU factorization of C or even C itself, since the elements above the diagonal are zero, and as in LU solving $C\vec{x} = \vec{d}$ is as easy as forward and back substitution. You will explore other properties of the factorization in the exercises.

In the end, code for Cholesky factorization can be very succinct. To derive a particularly compact form, suppose we choose an arbitrary row k and write L in block form isolating that row:

$$L = \begin{pmatrix} L_{11} & \vec{0} & 0 \\ \vec{\ell}_k^\top & \ell_{kk} & \vec{0}^\top \\ L_{31} & \vec{\ell}'_k & L_{33} \end{pmatrix}$$

Here, L_{11} and L_{33} are both lower triangular square matrices. Then, carrying out a product yields:

$$\begin{aligned} LL^\top &= \begin{pmatrix} L_{11} & \vec{0} & 0 \\ \vec{\ell}_k^\top & \ell_{kk} & \vec{0}^\top \\ L_{31} & \vec{\ell}'_k & L_{33} \end{pmatrix} \begin{pmatrix} L_{11}^\top & \vec{\ell}_k & L_{31}^\top \\ \vec{0}^\top & \ell_{kk} & (\vec{\ell}'_k)^\top \\ 0 & \vec{0} & L_{33}^\top \end{pmatrix} \\ &= \begin{pmatrix} \times & \times & \times \\ \vec{\ell}_k^\top L_{11}^\top & \vec{\ell}_k^\top \vec{\ell}_k + \ell_{kk}^2 & \times \\ \times & \times & \times \end{pmatrix} \end{aligned}$$

We leave out values of the product that are not necessary for our derivation.

In the end, we know we can write $C = LL^\top$. The middle element of the product shows:

$$\ell_{kk} = \sqrt{c_{kk} - \|\ell_k\|_2^2}$$

where $\vec{\ell}_k \in \mathbb{R}^{k-1}$ contains the elements of the k-th row of L to the left of the diagonal. Furthermore, the middle left element of the product shows

$$L_{11}\vec{\ell}_k = \vec{c}_k$$

where $\vec{c}_k$ contains the elements of C in the same position as $\vec{\ell}_k$. Since L_{11} is lower triangular, this system can be solved by forward substitution!

Notice that our discussion above yields an algorithm for computing the Cholesky factorization from top to bottom, since L_{11} will already be computed by the time we reach row k. We provide pseudocode below, adapted from CITE:

```
// Takes as input an n-by-n matrix A[i,j]
// Edits A in place to obtain the Cholesky factorization in its lower triangle

for k from 1 to n {
    // Back-substitute to find l_k
    for i from 1 to k-1 { // element i of l_k
        sum = 0;
        for j from 1 to i-1
            sum += A[i,j]*A[k,j];
        A[k,i] = (A[k,i]-sum)/A[i,i];
    }

    // Apply the formula for l_kk
    normSquared = 0
    for j from 1 to i-1
        normSquared += A[k,j]^2;
    A[k,k] = sqrt(A[k,k] - normSquared);
}
```

As with LU factorization, this algorithm clearly runs in $O(n^3)$ time.

3.2.2 Sparsity

Many linear systems of equations naturally enjoy *sparsity* properties, meaning that most of the entries of A in the system $A\vec{x} = \vec{b}$ are exactly zero. Sparsity can reflect particular structure in a given problem, including the following use cases:

- In image processing, many systems for photo editing and understanding express relationships between the values of pixels and those of their neighbors on the image grid. An image may be a point in $\mathbb{R}^p$ for p pixels, but when solving $A\vec{x} = \vec{b}$ for a new size-p image, $A \in \mathbb{R}^{p\times p}$ may have only $O(p)$ rather than $O(p^2)$ nonzeros since each row only involves a single pixel and its up/down/left/right neighbors.
- In machine learning, a *graphical model* uses a graph structure $G \equiv (V, E)$ to express probability distributions over several variables. Each variable is represented using a node $v \in V$ of the graph, and edge $e \in E$ represents a probabilistic dependence. Linear systems arising in this context often have one row per vertex $v \in V$ with nonzeros only in columns involving v and its neighbors.
- In computational geometry, shapes are often expressed using collections of triangles linked together into a mesh. Equations for surface smoothing and other tasks once again link positions and other values at a given vertex with those at their neighbors in the mesh.

Example 3.5 (Harmonic parameterization)**.** *Suppose we wish to use an image to texture a triangle mesh. A mesh can be represented as a collection of vertices $V \subset \mathbb{R}^3$ linked together by edges $E \subseteq V \times V$ to form triangles. Since the vertices of the geometry are in $\mathbb{R}^3$, we must find a way to map them to the image plane to store the texture as an image. Thus, we must assign texture coordinates $t(v) \in \mathbb{R}^2$ on the image plane to each $v \in V$. See Figure NUMBER for an illustration.*

One strategy for making this map involves a single linear solve. Suppose the mesh has disk *topology, that is, it can be mapped to the interior of a circle in the plane. For each vertex v_b on the boundary of the mesh, we can specify the position of v_b by placing it on a circle. In the interior, we can ask that the texture map position be the average of its neighboring positions:*

$$t(v) = \frac{1}{|n(v)|} \sum_{w \in n(v)} t(w)$$

Here, $n(v) \subseteq V$ is the set of neighbors of $v \in V$ on the mesh. Thus, each $v \in V$ is associated with a linear equation, either fixing it on the boundary or asking that its position equal the average of its neighboring positions. This $|V| \times |V|$ system of equations leads to a stable parameterization strategy known as harmonic parameterization*; the matrix of the system only has $O(|V|)$ nonzeros in slots corresponding to vertices and their neighbors.*

Of course, if $A \in \mathbb{R}^{n \times n}$ is sparse to the point that it contains $O(n)$ rather than $O(n^2)$ nonzero values, there is no reason to store A as an $n \times n$ matrix. Instead, *sparse matrix* storage techniques only store the $O(n)$ nonzeros in a more reasonable data structure, e.g. a list of row/column/value triplets. The choice of a matrix data structure involves considering the likely operations that will occur on the matrix, possibly including multiplication, iteration over nonzeros, or iterating over individual rows or columns.

Unfortunately, it is easy to see that the LU factorization of a sparse A may not result in sparse L and U matrices; this loss of structure severely limits the applicability of using these methods to solve $A\vec{x} = \vec{b}$ when A is large but sparse. Thankfully, there are many *direct sparse solvers* adapting LU to sparse matrices that can produce an LU-like factorization without inducing much *fill*, or additional nonzeros; discussion of these techniques is outside the scope of this text. Alternatively, *iterative* techniques have been used to obtain approximate solutions to linear systems; we will defer discussion of these methods to future chapters.

Certain matrices are not only sparse but also *structured*. For instance, a *tridiagonal* system of linear equations has the following pattern of nonzero values:

$$\begin{pmatrix} \times & \times & & & \\ \times & \times & \times & & \\ & \times & \times & \times & \\ & & \times & \times & \times \\ & & & \times & \times \end{pmatrix}$$

In the exercises following this chapter, you will derive a special version of Gaussian elimination for dealing with this this simple *banded* structure.

In other cases, matrices may not be sparse but might admit a sparse representation. For example, consider the *cyclic* matrix:

$$\begin{pmatrix} a & b & c & d \\ d & a & b & c \\ c & d & a & b \\ b & c & d & a \end{pmatrix}$$

Obviously, this matrix can be stored using only the values a, b, c, d. Specialized techniques for this and other classes of matrices are well-studied and often more efficient than generic Gaussian elimination.

3.3 Sensitivity Analysis

As we have seen, it is important to examine the matrix of a linear system to find out if it has special properties that can simplify the solution process. Sparsity, positive definiteness, symmetry, and so on all call can provide clues to the proper solver to use for a particular problem.

Even if a given solution strategy might work in theory, however, it is equally important to understand how well we can trust the answer to a linear system given by a particular solver. For instance, due to rounding and other discrete effects, it might be the case that an implementation of Gaussian elimination for solving $A\vec{x} = \vec{b}$ yields a solution $\vec{x}_0$ such that $0 < \|A\vec{x}_0 - \vec{b}\| \ll 1$; in other words, $\vec{x}_0$ only solves the system approximately.

One way to understand the likelihood of these approximation effects is through *sensitivity analysis*. In this approach, we ask what might happen to $\vec{x}$ if instead of solving $A\vec{x} = \vec{b}$ in reality we solve a *perturbed* system of equations $(A + \delta A)\vec{x} = \vec{b} + \delta\vec{b}$. There are two ways of viewing conclusions made by this type of analysis:

1. Likely we make mistakes representing A and $\vec{b}$ thanks to rounding and other effects. This analysis then shows the best possible accuracy we can expect for $\vec{x}$ given the mistakes made representing the problem.

2. If our solver generates an approximation $\vec{x}_0$ to the solution of $A\vec{x} = \vec{b}$, it is an exact solution to the system $A\vec{x}_0 = \vec{b}_0$ if we *define* $\vec{b}_0 \equiv A\vec{x}_0$ (be sure you understand why this sentence is not a tautology!). Understanding how changes in $\vec{x}_0$ affect changes in $\vec{b}_0$ show how sensitive the system is to slightly incorrect answers.

Notice that our discussion here is similar to and indeed motivated by our definitions of forward and backward error in previous chapters.

3.3.1 Matrix and Vector Norms

Before we can discuss the sensitivity of a linear system, we have to be somewhat careful to define what it means for a change $\delta\vec{x}$ to be "small." Generally, we wish to measure the length, or *norm*, of a vector $\vec{x}$. We have already encountered the two-norm of a vector:

$$\|\vec{x}\|_2 \equiv \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$$

for $\vec{x} \in \mathbb{R}^n$. This norm is popular thanks to its connection to Euclidean geometry, but it is by no means the only norm on $\mathbb{R}^n$. Most generally, we define a *norm* as follows:

Definition 3.2 (Vector norm)**.** *A vector norm is a function* $\|\cdot\| : \mathbb{R}^n \to [0, \infty)$ *satisfying the following conditions:*

- $\|\vec{x}\| = 0$ *if and only if* $\vec{x} = \vec{0}$.

- $\|c\vec{x}\| = |c|\|\vec{x}\|$ *for all scalars* $c \in \mathbb{R}$ *and vectors* $\vec{x} \in \mathbb{R}^n$.
- $\|\vec{x} + \vec{y}\| \leq \|\vec{x}\| + \|\vec{y}\|$ *for all* $\vec{x}, \vec{y} \in \mathbb{R}^n$.

While we use the two subscript $\|\cdot\|_2$ to denote the two-norm of a vector, unless we note otherwise we will use the notation $\|\vec{x}\|$ to denote the two-norm of $\vec{x}$. Other than this norm, there are many other examples:

- The p-norm $\|\vec{x}\|_p$, for $p \geq 1$, given by:

$$\|\vec{x}\|_p \equiv (|x_1|^p + |x_2|^p + \cdots + |x_n|^p)^{1/p}$$

  Of particular importance is the 1-norm or "taxicab" norm, given by

$$\|\vec{x}\|_1 \equiv \sum_{k=1}^{n} |x_k|.$$

  This norm receives its nickname because it represents the distance a taxicab drives between two points in a city where the roads only run north/south and east/west.

- The ∞-norm $\|\vec{x}\|_\infty$ given by:

$$\|\vec{x}\|_\infty \equiv \max(|x_1|, |x_2|, \cdots, |x_n|).$$

In some sense, many norms on $\mathbb{R}^n$ are the same. In particular, suppose we say two norms are *equivalent* when they satisfy the following property:

Definition 3.3 (Equivalent norms)**.** *Two norms* $\|\cdot\|$ *and* $\|\cdot\|'$ *are* equivalent *if there exist constants* c_{low} *and* c_{high} *such that*

$$c_{low}\|\vec{x}\| \leq \|\vec{x}\|' \leq c_{high}\|\vec{x}\|$$

for all $\vec{x} \in \mathbb{R}^n$.

This condition guarantees that up to some constant factors, all norms agree on which vectors are "small" and "large." In fact, we will state without proof a famous theorem from analysis:

Theorem 3.2 (Equivalence of norms on $\mathbb{R}^n$)**.** *All norms on* $\mathbb{R}^n$ *are equivalent.*

This somewhat surprising result implies that all vector norms have the same *rough* behavior, but the choice of a norm for analyzing or stating a particular problem can make a huge difference. For instance, on $\mathbb{R}^3$ the ∞-norm considers the vector $(1000, 1000, 1000)$ to have the same norm as $(1000, 0, 0)$ whereas the 2-norm certainly is affected by the additional nonzero values.

Since we perturb not only vectors but also matrices, we must also be able to take the norm of a matrix. Of course, the basic definition of a norm does not change on $\mathbb{R}^{n\times m}$. For this reason, we can "unroll" any matrix in $\mathbb{R}^{m\times n}$ to a vector in $\mathbb{R}^{nm}$ to adopt any vector norm to matrices. One such norm is the *Frobenius norm*, given by

$$\|A\|_{\text{Fro}} \equiv \sqrt{\sum_{i,j} a_{ij}^2}.$$

Such adaptations of vector norms, however, are not always very meaningful. In particular, the priority for understanding the structure of a matrix A often is its *action* on vectors, that is, the likely results when A is multiplied by an arbitrary $\vec{x}$. With this motivation, we can define the norm *induced* by a vector norm as follows:

Definition 3.4 (Induced norm)**.** *The norm on* $\mathbb{R}^{m\times n}$ induced *by a norm* $\|\cdot\|$ *on* $\mathbb{R}^n$ *is given by*

$$\|A\| \equiv \max\{\|A\vec{x}\| : \|\vec{x}\| = 1\}.$$

That is, the induced norm is the maximum length of the image of a unit vector multiplied by A*.*

Since vector norms satisfy $\|c\vec{x}\| = |c|\|\vec{x}\|$, it is easy to see that this definition is equivalent to requiring

$$\|A\| \equiv \max_{\vec{x}\in\mathbb{R}^n\setminus\{0\}} \frac{\|A\vec{x}\|}{\|\vec{x}\|}.$$

From this standpoint, the norm of A induced by $\|\cdot\|$ is the largest achievable ratio of the norm of $A\vec{x}$ relative to that of the input $\vec{x}$.

This general definition makes it somewhat hard to compute the norm $\|A\|$ given a matrix A and a choice of $\|\cdot\|$. Fortunately, the matrix norms induced by many popular vector norms can be simplified. We state some such expressions without proof:

- The induced one-norm of A is the maximum sum of any one column of A:

$$\|A\|_1 = \max_{1\le j\le n} \sum_{i=1}^{m} |a_{ij}|$$

- The induced ∞-norm of A is the maximum sum of any one row of A:

$$\|A\|_\infty = \max_{1\le i\le m} \sum_{j=1}^{n} |a_{ij}|$$

- The induced two-norm, or *spectral norm*, of $A \in \mathbb{R}^{n\times n}$ is the square root of the largest eigenvalue of $A^\top A$. That is,

$$\|A\|_2^2 = \max\{\lambda : \text{there exists } \vec{x} \in \mathbb{R}^n \text{ with } A^\top A\vec{x} = \lambda\vec{x}\}$$

At least the first two norms are relatively easy to compute; the third we will return to while discussing eigenvalue problems.

3.3.2 Condition Numbers

Now that we have tools for measuring the action of a matrix, we can define the *condition number* of a linear system by adapting our generic definition of condition numbers from Chapter 1. We follow the development presented in CITE.

Suppose we perturbation δA of a matrix A and a corresponding perturbation $\delta\vec{b}$. For each $\varepsilon \ge 0$, ignoring invertibility technicalities we can write a vector $\vec{x}(\varepsilon)$ as the solution to

$$(A + \varepsilon\cdot\delta A)\vec{x}(\varepsilon) = \vec{b} + \varepsilon\cdot\delta\vec{b}.$$

If we differentiate both sides with respect to ε and apply the product rule, we obtain the following result:

$$\delta A\cdot\vec{x} + (A + \varepsilon\cdot\delta A)\frac{d\vec{x}}{d\varepsilon} = \delta\vec{b}$$

In particular, when $\varepsilon = 0$ we find

$$\delta A \cdot \vec{x}(0) + A \frac{d\vec{x}}{d\varepsilon}\Big|_{\varepsilon=0} = \delta\vec{b}$$

or, equivalently,

$$\frac{d\vec{x}}{d\varepsilon}\Big|_{\varepsilon=0} = A^{-1}(\delta\vec{b} - \delta A \cdot \vec{x}(0)).$$

Using the Taylor expansion, we can write

$$\vec{x}(\varepsilon) = \vec{x} + \varepsilon\vec{x}'(0) + O(\varepsilon^2),$$

where we define $\vec{x}'(0) = \frac{d\vec{x}}{d\varepsilon}\big|_{\varepsilon=0}$. Thus, we can expand the relative error made by solving the perturbed system:

$$\begin{aligned}
\frac{\|\vec{x}(\varepsilon) - \vec{x}(0)\|}{\|\vec{x}(0)\|} &= \frac{\|\varepsilon\vec{x}'(0) + O(\varepsilon^2)\|}{\|\vec{x}(0)\|} \text{ by the Taylor expansion} \\
&= \frac{\|\varepsilon A^{-1}(\delta\vec{b} - \delta A \cdot \vec{x}(0)) + O(\varepsilon^2)\|}{\|\vec{x}(0)\|} \text{ by the derivative we computed} \\
&\leq \frac{|\varepsilon|}{\|\vec{x}(0)\|}(\|A^{-1}\delta\vec{b}\| + \|A^{-1}\delta A \cdot \vec{x}(0))\|) + O(\varepsilon^2) \\
&\qquad \text{by the triangle inequality } \|A + B\| \leq \|A\| + \|B\| \\
&\leq |\varepsilon|\|A^{-1}\|\left(\frac{\|\delta\vec{b}\|}{\|\vec{x}(0)\|} + \|\delta A\|\right) + O(\varepsilon^2) \text{ by the identity } \|AB\| \leq \|A\|\|B\| \\
&= |\varepsilon|\|A^{-1}\|\|A\|\left(\frac{\|\delta\vec{b}\|}{\|A\|\|\vec{x}(0)\|} + \frac{\|\delta A\|}{\|A\|}\right) + O(\varepsilon^2) \\
&\leq |\varepsilon|\|A^{-1}\|\|A\|\left(\frac{\|\delta\vec{b}\|}{\|A\vec{x}(0)\|} + \frac{\|\delta A\|}{\|A\|}\right) + O(\varepsilon^2) \text{ since } \|A\vec{x}(0)\| \leq \|A\|\|\vec{x}(0)\| \\
&= |\varepsilon|\|A^{-1}\|\|A\|\left(\frac{\|\delta\vec{b}\|}{\|\vec{b}\|} + \frac{\|\delta A\|}{\|A\|}\right) + O(\varepsilon^2) \text{ since } \|A\vec{x}(0)\| \leq \|A\|\|\vec{x}(0)\| \\
&\qquad \text{since by definition, } A\vec{x}(0) = \vec{b}
\end{aligned}$$

Here we have applied some properties of the matrix norm which follow from corresponding properties for vectors. Notice that the sum $D \equiv {}^{\|\delta\vec{b}\|}/_{\|\vec{b}\|} + {}^{\|\delta A\|}/_{\|A\|}$ encodes the relative perturbations of A and $\vec{b}$. From this standpoint, to first order we have bounded the relative error of perturbing the system by ε using a factor $\kappa \equiv \|A\|\|A^{-1}\|$:

$$\frac{\|\vec{x}(\varepsilon) - \vec{x}(0)\|}{\|\vec{x}(0)\|} \leq \varepsilon \cdot D \cdot \kappa + O(\varepsilon^2)$$

In this way, the quantity κ bounds the conditioning of linear systems involving A. For this reason, we make the following definition:

Definition 3.5 (Matrix condition number)**.** *The condition number of $A \in \mathbb{R}^{n\times n}$ for a given matrix norm $\|\cdot\|$ is*

$$\text{cond } A \equiv \|A\|\|A^{-1}\|.$$

If A is not invertible, we take cond $A \equiv \infty$.

It is easy to see that cond $A \geq 1$ for all A, that scaling A has no effect on its condition number, and that the condition number of the identity matrix is 1. These properties contrast with the determinant, which can scale up and down as you scale A.

If $\|\cdot\|$ is induced by a vector norm and A is invertible, then we have

$$\begin{aligned}
\|A^{-1}\| &= \max_{\vec{x}\neq\vec{0}} \frac{\|A^{-1}\vec{x}\|}{\|\vec{x}\|} \text{ by definition} \\
&= \max_{\vec{y}\neq\vec{0}} \frac{\|\vec{y}\|}{\|A\vec{y}\|} \text{ by substituting } \vec{y} = A^{-1}\vec{x} \\
&= \left(\min_{\vec{y}\neq\vec{0}} \frac{\|A\vec{y}\|}{\|\vec{y}\|}\right)^{-1} \text{ by taking the reciprocal}
\end{aligned}$$

In this case, the condition number of A is given by:

$$\text{cond } A = \left(\max_{\vec{x}\neq\vec{0}} \frac{\|A\vec{x}\|}{\|\vec{x}\|}\right)\left(\min_{\vec{y}\neq\vec{0}} \frac{\|A\vec{y}\|}{\|\vec{y}\|}\right)^{-1}$$

In other words, cond A measures the maximum to minimum possible stretch of a vector $\vec{x}$ under A.

More generally, a desirable stability property of a system $A\vec{x} = \vec{b}$ is that if A or $\vec{b}$ is perturbed, the solution $\vec{x}$ does not change considerably. Our motivation for cond A shows that when the condition number is small, the change in $\vec{x}$ is small relative to the change in A or $\vec{b}$, as illustrated in Figure NUMBER. Otherwise, a small change in the parameters of the linear system can cause large deviations in $\vec{x}$; this instability can cause linear solvers to make large mistakes in $\vec{x}$ due to rounding and other approximations during the solution process.

The norm $\|A^{-1}\|$ can be as difficult as computing the full inverse A^{-1}. One way to lower-bound the condition number is to apply the identity $\|A^{-1}\vec{x}\| \leq \|A^{-1}\|\|\vec{x}\|$. Thus, for any $\vec{x} \neq \vec{0}$ we can write $\|A^{-1}\| \geq {}^{\|A^{-1}\vec{x}\|}/_{\|\vec{x}\|}$. Thus,

$$\text{cond } A = \|A\|\|A^{-1}\| \geq \frac{\|A\|\|A^{-1}\vec{x}\|}{\|\vec{x}\|}.$$

So, we can bound the condition number by solving $A^{-1}\vec{x}$ for some vectors $\vec{x}$; of course, the necessity of a linear solver to find $A^{-1}\vec{x}$ creates a circular dependence on the condition number to evaluate the quality of the estimate! When $\|\cdot\|$ is induced by the two-norm, in future chapters we will provide more reliable estimates.

3.4 Problems

Something like:

- Kernel regression as an example of §3.1.1
- Least-norm solution to $A\vec{x} = \vec{b}$, least squares matrix is invertible otherwise
- Variational versions of Tikhonov regularization/"ridge" regression (not the usual approach to this, but whatever); completing the underdetermined story this way
- L^1 approaches to regularization for contrast – draw a picture of why to expect sparsity, draw unit circles, show that p norm isn't a norm for $p < 1$, take limit as $p \to 0$
- Mini-Riesz – derive matrix for inner product, use to show how to rotate space
- tridiagonal solve
- properties of condition number