Quals Revision Notes Numerics Sequence

Tyler Chen

1 Introduction

This document contains personal revision notes for the 2018 Numerics Qualification Exam in the Applied Mathematics department at the University of Washington. These notes are heavily based on the textbooks [?, ?, ?] used in the AMATH 584/585/586 courses.

For notational convenience any result which was determined to be worth memorizing was marked as a “Theorem” even if the result is not usually classified as a Theorem.

Contents

2 Fundamentals

2.1 Matrix Vector Multiplication

Definition. (Vector Vector)

Definition. (Matrix Vector)

Definition. (Matrix Matrix)

Note: Add different ways to think of matrix multiplication in terms of rows and columns. I.e. projectors as sum of rank one outer products?? are there analogous general forms.

Definition. (Range)

Definition. (Nullspace)

Theorem.

The range of A is the space spanned by the columns of A.

Definition. (Rank)

Theorem.

A matrix $A \in \mathbb{C}^{m\times n}$ with $m \geq n$ has full rank if and only if it maps no two distinct vectors to the same vector.

Definition. (Inverse)

Theorem.

For $A \in \mathbb{C}^{m\times m}$ the following are equivalent:

- A has an inverse
- $\text{rank}(A) = m$
- $\text{range}(A) = \mathbb{C}^m$
- $\text{null}(A) = \{0\}$
- 0 is not an eigenvalue of A
- 0 is not a singular value of A
- $\det(A) \neq 0$

2.2 Orthogonal Vectors and Matrices

Definition. (Hermetian Conjugate)

Definition. (Hermetian Matrix)

Definition. (Inner Product)

Definition. (bilinear)

Definition. (orthogonal)

Theorem.

The vectors in an orthogonal set S are linearly independent.

2.2.1 Components of a Vector

Definition. (Unitary Matrix)

2.2.2 Multiplication by a Unitary Matrix

2.3 Norms

2.3.1 Vector Norms

Definition. (vector norm)

A vector norm is a function $\|\cdot\| : \mathbb{C}^m \to \mathbb{R}$ satisfying:

(i) $\|x\| \geq 0$, and $\|x\| = 0$ only if $x = 0$.
(ii) $\|x + y\| \leq \|x\| + \|y\|$
(iii) $\|\alpha x\| = |\alpha| \, \|x\|$

Important norms are p-norms.

Also have weighted p-norms.

For any non-singular W, $\|x\|_W = \|Wx\|$ defines a matrix norm.

2.3.2 Matrix Norms

Definition. (induced matrix norm)

Given norms $\|\cdot\|_{(n)}$ and $\|\cdot\|_{(m)}$ respectively defined on the domain and range of $A \in \mathbb{C}^{m \times n}$, the induced matrix norm $\|A\|_{(m,n)}$ is defined as,

$$\|A\|_{(m,n)} = \sup_{x \neq 0} \frac{\|Ax\|_{(m)}}{\|x\|_{(n)}} = \sup_{\|x\|_{(n)}=1} \|Ax\|_{(m)}$$

one norms

infinity norms

Theorem. (Hölder Inequality)

Let $1/p + 1/q = 1$ with $p, q \geq 1$. Then,

$$|x^* y| \leq \|x\|_p \, \|y\|_q$$

In the case $p = q = 2$ this is the Cauchy-Schwarz Inequality.

Theorem.

Let $\|\cdot\|_{(l)}$, $\|\cdot\|_{(m)}$, and $\|\cdot\|_{(n)}$ be norms on $\mathbb{C}^l$, $\mathbb{C}^m$, and $\mathbb{C}^n$ respectively, and let $A \in \mathbb{C}^{l\times m}$ and $B \in \mathbb{C}^{m\times n}$. Then,

$$\|AB\|_{(l,n)} \leq \|A\|_{(l,m)} \|B\|_{(m,n)}$$

Definition. (matrix norm)

A matrix norm is a function $\|\cdot\| : \mathbb{C}^{m\times n} \to \mathbb{R}$ satisfying:

(i) $\|A\| \geq 0$, and $\|A\| = 0$ only if $A = 0$.
(ii) $\|A + B\| \leq \|A\| + \|B\|$
(iii) $\|\alpha A\| = |\alpha| \|A\|$

Definition. (Frobenius Norm)

Theorem.

$\|A\|_F = \sqrt{\operatorname{tr}(AA^*)} = \sqrt{\operatorname{tr}(A^*A)}$

Theorem. (Invariance under Unitary Multiplication)

For any $A \in \mathbb{C}^{m\times n}$ and unitary $Q \in \mathbb{C}^{m\times m}$ we have,

$$\|QA\|_2 = \|A\|_2, \qquad \|QA\|_F = \|A\|_F$$

Note: Check that this is not true for general norms

I guess it should be fore the norm induced by the inner product on the inner product space.

2.4 The Singular Value Decomposition

The SVD is motivated by the fact that the image of the unit ball under matrix multiplication is a hyperellipse.

Definition. (Singular Value Decomposition)

Given $A \in \mathbb{C}^{m\times n}$ a singular value decomposition of A is a factorization $A = U\Sigma V^*$, where $U \in \mathbb{C}^{m\times m}$ and $V \in \mathbb{C}^{n\times n}$ are unitary and $\Sigma \in \mathbb{C}^{m\times n}$ is diagonal.

In addition, we assume that the diagonal entries of Σ are in non-increasing order.

A reduced SVD drops the zero entries of Σ and the corresponding columns and rows of U and V^*.

Theorem. (SVD Existence and Uniqueness)

Every matrix $A \in \mathbb{C}^{m\times n}$ has a singular value decomposition. Furthermore, the signular values $\{\sigma_j\}$ are uniqely determined, and if A is square and the σ_j are distinct, then left and right singular vectors, $\{u_j\}$ and $\{v_j\}$ are uniquely determined up to complex signs.

Theorem.

The rank of A is the number of nonzer singular values

Theorem.

$\text{range}(A) = \text{span}\{u_1, \dots, u_r\}$ and $\text{null}(A) = \text{span}\{v_1, \dots, v_r\}$.

Theorem.

$\|A\|_2 = \sigma_1$ and $\|A\|_F = \sqrt{\sigma_1^2 + \cdots + \sigma_r^2}$

Theorem.

The nonzero singular values of A are the square roots of the eigenvalues of A^*A or AA^*. (These matrices have the same nonzero eigenvalues.)

Theorem.

If $A = A^*$, then the singular values of A are the absolute values of the eigenvalues of A.

Theorem.

$|\det(A)| = \Pi_{j=1}^{m}\sigma_j$

Theorem.

A is the sum of r rank-one matrices:

$$A = \sum_{j=1}^{r} \sigma_j u_j v_j^*$$

Theorem.

For any $0 \leq k \leq r$, define,

$$A_k = \sum_{j=1}^{k} \sigma_j u_j v_j^*$$

if $k = \min\{m, n\}$ define $\sigma_{\min\{n,m\}} = 0$. Then,

$$\|A - A_k\|_2 = \inf_{\text{rank}(B)=k} \|A - B\|_2 = \sigma_{k+1}$$

This basically gives rise to principal component analysis (PCA). I.e.

Theorem.

$\|A - A_k\|_F = \inf_{\text{rank}(B)=k} \|A - B\|_F = \sqrt{\sigma_{k+1} + \cdots + \sigma_r^2}$

3 QR Factorization and Least Squares

3.1 Projectors

Definition. (Projector)

A projector is a square matrix P satisfying,

$$P^2 = P$$

Definition. (Complimentary Projector)

Given a projector P, the matrix $I - P$ is also a projector and is called the complimentary projector to P.

Theorem.

range$(I - P) =$ null(P) and range$(P) \cap$ null$(P) = \{0\}$

Theorem.

Let S_1 and S_2 be two subspaces of $\mathbb{C}^m$ satisfying $S_1 \cap S_2 = \{0\}$ and $S_1 + S_2 = \mathbb{C}^m$, where $S_1 + S_2$ denotes the span of S_1 and S_2. Then there exists a projector P with range$(P) = S_1$ and null$(P) = S_2$. We say P is a projector onto S_1 along S_2.

3.1.1 Orthogonal Projectors

Definition. (Orthogonal Projector)

A projector P is orthogonal if range$(P) \perp$ null(P).

Theorem.

A projector P is orthogonal if and only if $P = P^*$.

Proof.

Forward: trivial. Backward: Pick orthonormal bases for range and kernel. Then this basically is basically the SVD and is clearly Hermetian.

3.1.2 Projection with an Orthonormal Basis

The SVD of an orthogonal projector has some zero entries. If we drop these we obtain the reduced SVD,

$$P = \hat{Q}\hat{Q}^*$$

It is also clear that given any $\hat{Q}$ orthonormal that $\hat{Q}\hat{Q}^*$ is an orthogonal projector. A special case is rank one projectors where $\hat{Q} = q$.

Theorem.

Given any linear independent set of vectors, $A = \{a_1, a_2, \ldots, a_r\}$, $P = A(A^*A)^{-1}A^*$ is an orthogonal projector onto range(A).

3.2 QR Factorization

Definition. (QR Factorization)

Given $A \in \mathbb{C}^{m\times n}$ ($m \geq n$) of full rank, a QR factorization of A is a factorization $A = QR$, where $Q \in \mathbb{C}^{m\times m}$ is unitary and $R \in \mathbb{C}^{m\times n}$ is upper triangular.

A reduced QR factorization drops the silent columns of Q and rows of R.

A natural way of constructing a QR decomposition is Gram–Schmidt orthogonalization.

Algorithm. (Gram–Schmidt ($\mathcal{O}(2mn^2)$))

```
for $j = 1$ to $n$:
    $v_j = a_j$
    for $i = 1$ to $j - 1$:
        $r_{ij} = q_i^* a_j$
        $v_j = v_j - r_{ij} q_i$
    end for
    $r_{jj} = \|v_j\|_2$
    $q_j = v_j / r_{jj}$
end for
```

Theorem. (QR Existence and Uniqueness)

Every $A \in \mathbb{C}^{m\times n}$, $(m \geq n)$ of full rank has a full QR factorization, hence a reduced QR factorization.

The reduced QR factorization is unique when $r_{jj} > 0$.

Method. (Solve Linear System using QR)

1. Compute QR factorization $A = QR$
2. Compute $y = Q^*b$
3. Solve triangular system $Rx = Q^*b$

Note: Add running time

It turns out that Gram–Schmidt in the above form is unstable.

Algorithm. (Modified Gram–Schmidt ($\mathcal{O}(2mn^2)$))

```
for $j = 1$ to $n$:
    $v_j = a_j$
    for $i = 1$ to $j - 1$:
        $r_{ij} = q_i^* v_j$
        $v_j = v_j - r_{ij} q_i$
    end for
    $r_{jj} = \|v_j\|_2$
    $q_j = v_j / r_{jj}$
end for
```

Note: why were the indexing orders also switched??. Rewrite this in a more logical way..

3.3 Householder Triangularization

Householder QR decomposition triangularizes a matrix using unitary operations. This is done by choosing Q_k so that $Q_n \cdots Q_2 Q_1 A$ is triangular. For instance,

$$\underset{A}{\begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix}} \xrightarrow{Q_1} \underset{Q_1 A}{\begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \end{bmatrix}} \xrightarrow{Q_2} \underset{Q_2 Q_1 A}{\begin{bmatrix} \times & \times & \times \\ & \times & \times \\ & 0 & \times \\ & 0 & \times \\ & 0 & \times \end{bmatrix}} \xrightarrow{Q_3} \underset{Q_3 Q_2 Q_1 A}{\begin{bmatrix} \times & \times & \times \\ & \times & \times \\ & & \times \\ & & 0 \\ & & 0 \end{bmatrix}}$$

3.3.1 Householder Reflectors

We would like a map like,

$$x = \begin{bmatrix} \times \\ \times \\ \times \\ \vdots \\ \times \end{bmatrix} \qquad \longrightarrow \qquad Fx = \begin{bmatrix} \|x\| \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \|x\| \, e_1$$

To do this we will defined F as the reflection across the hyperplane orthogonal to $v = \|x\| e_1 - x$.

First note that vv^*/v^*v gives the vector from x to the hyperplane. Therefore,

$$F = I - 2\frac{vv^*}{v^*v}$$

will is a reflection across this hyerplane.

Note that we can reflect to any points $z \|x\| e_1$, where $|z| = 1$. For numerical stability we want to move as far as possible.

In the real case this is done by picking $z = -\operatorname{sign}(x_1)$. Then $v = \operatorname{sign}(x_1) \|x\| e_1 + x$.

Algorithm. (Householder QR Factorization ($\mathcal{O}(2mn^2 - \frac{2}{3}n^3)$))

```
for $k = 1$ to $n$:
    $x = A_{k:m,k}$
    $v_k = \operatorname{sign}(x_1) \|x\|_2 e_1 + x$
    $v_k = v_k / \|v_k\|_2$
    $A_{k:m,k:n} = A_{k:m,k:n} - 2v_k(v_k^* A_{k:m,k:n})$
end for
```

Note that this reduces A to an upper triangular matrix R, however Q is not explicitly constructed.

We do not always need to construct Q explicitly.

Algorithm. (Implicit Calculation of Q^*b)

```
for $k = 1$ to $n$:
    $b_{k:m} = b_{k:m} - 2v_k(v_k^* b_{k:m})$
end for
```

Algorithm. (Implicit Calculation of Qx)

```
for $k = n$ downto $1$:
    $x_{k:m} = x_{k:m} - 2v_k(v_k^* x_{k:m})$
end for
```

3.4 Least Squares

The general least squares problem is to solve,

$$\min_x \|b - Ax\|$$

Theorem.

Let $A \in \mathbb{C}^{m \times n}$ $(m \geq n)$ and $b \in \mathbb{C}^m$. A vector x minimizes the residual norm $\|r\|_2 = \|b - Ax\|_2$ if and only if $r \perp \text{range}(A)$, that is,

$$A^* r = 0$$

or equivalently,

$$A^* A x = A^* b$$

or again equivalently,

$$Pb = Ax$$

where $P \in \mathbb{C}^{m \times m}$ is the orthogonal projector onto range(A).

The solution x is unique if and only if A has full rank.

Proof.

The three if statements are equivalent by the definition of r and by properties of orthogonal projectors.

Then show that the point $Pb \in \text{range}(A)$ is the point minimizing the residual norm. This obvious geometrically.

Finally, note that A^*A is full rank if and only if A is full rank. Could use SVD.

Definition. (Pseudoinverse)

The Pseudoinverse of A is defined as $A^\dagger = (A^*A)^{-1}A^*$

To solve the least squares problem we can compute $x = A^\dagger b$ or $y = Ax = Pb$, where P is the projection onto range(A).

3.4.1 Methods for Solving the Least Squares Problem

Method. (Least squares via Normal Equations ($\mathcal{O}(mn^2 + \frac{1}{3}n^3)$))

1. Form the matrix A^*A and the vector A^*b.
2. Compute Cholesky factorization $A^*A = R^*R$.
3. Solve lower-triangular system $R^*w = A^*b$ for w.

4. Solve upper-triangular system $Rx = w$ for x.

Method. (Least squares via QR factorization ($\mathcal{O}(2mn^2 - \frac{2}{3}n^3)$))

1. Compute the reduced QR factorization $A = \hat{Q}\hat{R}$.
2. Compute the vector $\hat{Q}^*b$.
3. Solve upper-triangular system $\hat{R}x = \hat{Q}^*b$ for x.

Method. (Least squares via SVD ($\mathcal{O}(2mn^2 + 11n^3)$))

1. Compute the reduced SVD $A = \hat{U}\hat{\Sigma}\hat{V}^*$.
2. Compute the vector $\hat{U}^*b$.
3. Solve the diagonal system $\hat{\Sigma}w = \hat{U}^*b$ for w.
4. Set $x = Vw$.

3.4.2 Comparison of Methods

Each of these methods may be advantageous in certain situations. When speed is the only consideration solving the normal equations may be best. However this is not always stable. QR is a good middle ground and is the generally suggested algorithm. However, if A is close to rank-deficient then it may also be unstable and so the SVD may be better.

4 Conditioning and Stability

4.1 Conditioning

We can view a problem as a function $f : X \to Y$ between normed vector spaces.

Definition. (Well-Conditioned)

A well-conditioned problem is one with the property that all small perturbations of x lead to only small changes in $f(x)$.

Definition. (Absolute Condition Number)

The absolute condition number $\hat{\kappa} = \hat{\kappa}(x)$ of the problem f at x is defined as,

$$\hat{\kappa} = \lim_{\delta \to 0} \sup_{\|\delta x\| \leq \delta} \frac{\|f(x+\delta x) - f(x)\|}{\|\delta x\|}$$

If f is differentiable then $f(x+\delta x) - f(x) \approx J(x)\delta x$ to first order. Therefore,

$$\hat{\kappa} = \|J(x)\|$$

Definition. (Relative Condition Number)

The relative condition number $\kappa = \kappa(x)$ of the problem f at x is defined as,

$$\kappa = \lim_{\delta \to 0} \sup_{\|\delta x\| \leq \delta} \left(\frac{\|f(x+\delta x) - f(x)\|}{\|\delta x\|} \middle/ \frac{\|\delta x\|}{\|x\|} \right)$$

If f is differentiable,

$$\kappa = \frac{\|J(x)\|}{\|f(x)\| \,/\, \|x\|}$$

Theorem.

Let $A \in \mathbb{C}^{m \times m}$ be non-singular and consider the equation $Ax = b$. The problem of computing b, given x, has condition number,

$$\kappa = \|A\| \frac{\|x\|}{\|b\|} \leq \|A\| \left\|A^{-1}\right\|$$

with respect to perturbations of x. The problem of computing x, given b, has condition number

$$\kappa = \left\|A^{-1}\right\| \frac{\|b\|}{\|x\|} \leq \|A\| \left\|A^{-1}\right\|$$

with respect to perturbations of b.

If $\|\cdot\| = \|\cdot\|_2$, the top bound is tight when x is a multiple of the right singular vector corresponding to the smallest singular value, and the bottom bound is tight when b is a multiple of the left singular vector corresponding to the largest singular value.

Definition. (Condition Numer of a Matrix)

The condition number of a matrix is defined as $\kappa(A) = \|A\| \, \|A^{-1}\|$.

Theorem.

If $\|\cdot\| = \|\cdot\|_2$, then $\kappa(A) = \sigma_{\max}/\sigma_{\min}$.

Theorem.

Let b be fixed and consider the problem of computing $x = A^{-1}b$, where A is square and non-singular. The condition number of this problem with respect to perturbations of A is $\kappa = \kappa(A)$

4.2 Floating Point Arithmetic

Definition. (Floating Point Numers)

A floating point system $\mathbb{F}$ for some fixed base β, fixed exponent length k, and fixed precision t, are numbers of the form,

$$x = \pm 1.m_1 m_2 m_3 \cdots m_t \times \beta^{e_1 e_2 \cdots e_k}$$

Definition. (Machine Precision)

Machine precision is defined to be half the distance between one and the next largest number.

Theorem. (Fundamental Axiom of Floating Point Arithmetic)

For all $x, y \in \mathbb{F}$, there exists ϵ with $|\epsilon| \leq \epsilon_{\text{machine}}$ such that,

$$x \circledast y = (x * y)(1 + \epsilon)$$

4.3 Stability

Since computers are discrete and finite, it isn't possible to obtain exact solutions to many problems.

> **Definition.** (Algorithm)
>
> An algorithm is a map $\tilde{f} : X \to Y$ which is computed by, taking the input x, rounding it to the floating point system and using the computer to solve the problem.

Note: IF WE ROUND HOW DO WE KNOW WE WILL STILL BE IN Y IF Y ISNT R OR C??

At the very least $\tilde{f}(x)$ will be affected by rounding errors. It could also be affected by other things depending on the machine it is run on.

> **Definition.** (Absolute and Relative Errors)
>
> The absolute error of a computation is, $\left\|\tilde{f}(x) - f(x)\right\|$ and the relative error of a computation is $\left\|\tilde{f}(x) - f(x)\right\| / \|f(x)\|$.

If a problem f is ill-conditioned, having a relative accuracy on the order of machine precision is "unreasonably ambitious". Rounding error is unavoidable, and so even if all subsequent computations could be done exactly, the initial rounding error might a large enough perturbation to lead to a computed solution very different from the actual solution.

> **Definition.** (Stable Algorithm)
>
> We say an algorithm $\tilde{f}$ for a problem f is stable if for each $x \in X$,
>
> $$\frac{\left\|\tilde{f}(x) - f(\tilde{x})\right\|}{\|f(\tilde{x})\|} = \mathcal{O}(\epsilon_{\text{machine}})$$
>
> for some $\tilde{x}$ with,
>
> $$\frac{\|x - \tilde{x}\|}{\|x\|} = \mathcal{O}(\epsilon_{\text{machine}})$$

In words: A stable algorithm gives nearly the right answer to nearly the right problem

Definition. (Backward Stable Algorithm)

We say an algorithm $\tilde{f}$ for a problem f is backward stable if for each $x \in X$,

$$\tilde{f}(x) = f(\tilde{x})$$

for some $\tilde{x}$ with,

$$\frac{\|x - \tilde{x}\|}{\|x\|} = \mathcal{O}(\epsilon_{\text{machine}})$$

In words: A backward stable algorithm gives exactly the right answer to nearly the right question.

Definition.

We say $\varphi(t) = \mathcal{O}(\psi(t))$ if there is some $C > 0$ such that $|\varphi(t)| \leq C\psi(t)$ as t approaches some understood limit.

General the constant C may depend on the size of the vector spaces X and Y. This doens't necessarily contradict our definitions, since changing the dimension of these spaces means we need a new norm.

Theorem.

For problems f and algorithms $\tilde{f}$ defined on finite-dimensional spaces X and Y, the properties of accuracy, stability and backward stability all hold or fail to hold independently of the choices of norms for X and Y.

Theorem.

Suppose a backward stable algorithm is applied to solve a problem $f : X \to Y$ with condition number κ. Then the relative errors satisfy,

$$\frac{\left\|\tilde{f}(x) - f(x)\right\|}{\|f(x)\|} = \mathcal{O}(\kappa(x)\epsilon_{\text{machine}})$$

4.3.1 Stability of Householder Triangularization

We can construct an experiment to test Householder triangularization. Let Q and R be unitary and triangular and construct $A = QR$. Now use Householder to compute

$\tilde{Q}$ and $\tilde{R}$. It turns out that $\left\|Q - \tilde{Q}\right\|$ and $\left\|R - \tilde{R}\right\|$ are large. However, $\left\|A - \tilde{Q}\tilde{R}\right\|$ is not.

Theorem.

Let $A = QR$ be the QR factorization of a matrix $A \in \mathbb{C}^{m\times n}$ be computed using Householder triangularization, and let $\tilde{Q}$ and $\tilde{R}$ be the outputs. Then there exists δA such that,

$$\tilde{Q}\tilde{R} = A + \delta A, \qquad \frac{\|\delta A\|}{\|A\|} = \mathcal{O}(\epsilon_{\text{machine}})$$

Theorem.

Solving $Ax = b$ using QR and back substitution is backward stable, satisfying,

$$(A + \Delta A)\tilde{x} = b, \qquad \frac{\|\Delta A\|}{\|A\|} = \mathcal{O}(\epsilon_{\text{machine}})$$

Moreover,

$$\frac{\|\tilde{x} - x\|}{\|x\|} = \mathcal{O}(\kappa(A)\epsilon_{\text{machine}})$$

4.4 Stability of Back Substitution

Backward/Forward substitution are the standard methods of solving (upper/lower) triangular systems.

Algorithm. (Backward Substitution ($\mathcal{O}(m^2)$))

```
$x_m = b_m / r_{mm}$
for $j = m - 1$ downto $1$:
    $x_j = \left(b_j - \sum_{k=j+1}^{m} x_k r_{jk}\right) \Big/ r_{jj}$
end for
```

Theorem.

Backward Substitution is stable in the sense that the compute solution $\tilde{x}$ satisfies,

$$(R + \delta R)\tilde{x} = b$$

for some upper-triangular matrix δR with,

$$\frac{\|\delta R\|}{\|R\|} = \mathcal{O}(\epsilon_{\text{machine}})$$

4.5 Conditioning of Least Squares Problems

Note: IDK if this is actually important

4.6 Stability of Least Squares Algorithms

Theorem.

Let the full-rank least squares problem be solved by Householder triangularization. This algorithm is backwards stable in the sense t hat the compute solution $\tilde{x}$has the property,

$$\|(A+\delta A)\tilde{x} - b\| = \min, \qquad \frac{\|\delta A\|}{\|A\|} = \mathcal{O}(\epsilon_{\text{machine}})$$

for some δA. This is true whether Q^*b is compute via explicit formation of $\hat{Q}$ or implicitly. It also holds for Householder triangularization with arbitrary column pivoting.

Theorem.

The solution of the full-rank least squares problem via Gram–Schmidt orthogonalization is also backward stable, satisfying,

$$\|(A+\delta A)\tilde{x} - b\| = \min, \qquad \frac{\|\delta A\|}{\|A\|} = \mathcal{O}(\epsilon_{\text{machine}})$$

for some δA provided Q^*b is formed implicitly as indicated in the code segment above.

Note: So forming Q^*b implicitly is a better way to compute Q^*b than just projecting the normal way. But what about the order you do it implicitly. Like does it really matter that you start from the first coordinate?

> **Theorem.**
>
> The solution of the full-rank least squares problem via the normal equations is unstable. Stability can be achieved for problems where $\kappa(A)$ is uniformly bounded above, or $(\tan\theta)/\eta$ is uniformly bounded below.

> **Theorem.**
>
> The solution of the full-rank least squares problem via the SVD is backward stable, satisfying,
>
> $$\|(A+\delta A)\tilde{x}-b\| = \min, \qquad \frac{\|\delta A\|}{\|A\|} = \mathcal{O}(\epsilon_{\text{machine}})$$
>
> for some δA.

4.6.1 Rank-Deficient Least Squares Problems

When A has rank $< n$ column pivoting and SVD become important. The only fully stable algorithms for rank0-deficient problems are based on the SVD. Householder triangularization with column pivoting is stable for almost all problems.

5 Systems of Equations

5.1 Gaussian Elimination

Gaussian elimination transforms a full linear system into an upper-triangular one by applying simple linear transformations on the left. In this respect it is analogous to Householder triangularization for computing QR factorization. The difference is that the transformations applied in Gaussian elimination are not unitary.

5.1.1 LU factorization

> **Definition.** (LU factorization)
>
> The LU factorization of a square matrix A is: $A = LU$, L lower triangular, U upper triangular.

Such an factorization can sometimes be computing by turning A inter an upper triangular matrix by introducing zeros below the diagonal, first the the first column, then the second, and so on. This can be done by subtracting multiples of each row from subsequent rows. This can be thought of as multiplying A by a sequence of lower triangular matrices L_k on the left,

$$\underbrace{L_{m-1}\cdots L_2 L_1}_{L^{-1}} A = U$$

Setting $L = L_1^{-1}L_2^{-1}\cdots L_{m-1}^{-1}$ gives $A = LU$.

Suppose x_k denotes the k-th column of the matrix at the beginning of step k. Then the transformation,

$$x_k = \begin{bmatrix} x_{1,k} \\ \vdots \\ x_{k,k} \\ x_{k+1,k} \\ \vdots \\ x_{m,k} \end{bmatrix} \xrightarrow{L_k} L_k x_k = \begin{bmatrix} x_{1,k} \\ \vdots \\ x_{k,k} \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

To do this we wish to subtract $\ell_{j,k}$ times row k from row j, where $\ell_{j,k}$ is the multiplier

$$\ell_{j,k} = \frac{x_{j,k}}{x_{kk}} \qquad\qquad (k < j \le m)$$

In matrix form,

$$L_k = \begin{bmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & -\ell_{k+1,k} & 1 & & \\ & & \vdots & & \ddots & \\ & & -\ell_{m,k} & & & 1 \end{bmatrix}$$

We can write $L_k = I - \ell_k e_k^*$, where e_k is the column vector with 1 in position k and zeros elsewhere.

We can then observe,

$$(I - \ell_k e_k^*)(I + \ell_k e_k^*) = I - \ell_k e_k^* \ell_k e_k^* = I$$

Thus the inverse of L_k is $I + \ell_k e_k^*$.

We also observe that $L_k^{-1}L_{k+1}^{-1}$ is the unit lower-triangular matrix with entries of both L_k^{-1} and L_{k+1}^{-1} inserted in their usual places below the diagonal. In fact,

$$L = L_1^{-1}L_2^{-1}\cdots L_{m-1}^{-1} = L_k = \begin{bmatrix} 1 & & & & \\ \ell_{2,1} & 1 & & & \\ \ell_{3,1} & \ell_{3,2} & 1 & & \\ \vdots & \vdots & \ddots & \ddots & \\ \ell_{m,1} & \ell_{m,2} & \cdots & \ell_{m,m-1} & 1 \end{bmatrix}$$

In practise we do not ever form or multiply the L_k.

Algorithm. (Gaussian Elimination without Pivoting $\mathcal{O}\left(\frac{2}{3}m^3\right)$)

```
$U = A, L = I$
for $k = 1$ to $m - 1$:
    for $j = k + 1$ to $m$:
        $\ell_{j,k} = u_{j,k}/u_{kk}$
        $u_{j,k:m} = u_{j,k:m} - \ell_{j,k}u_{k,k:m}$
    end for
end for
```

To save memory note that we can write L and U to the same array as A.

However this algorithm can be really unstable sometimes.

5.1.2 Pivoting

Without pivoting we always subtracted some multiple of $x_{k,k}$ from the other rows. However, we could use a different entry as the “pivot” and then interchange rows and columns.

However, this would require $\mathcal{O}(m^2)$ entries to be examined each of the m iterations. In total this is $\mathcal{O}(m^3)$ operations.

Instead we could only interchange rows. This keeps the total operations to $\mathcal{O}(m^2)$.

After $m-1$ steps A becomes the upper-triangular and U,

$$L_{m-1}P_{m-1}\cdots L_2P_2L_1P_1A = U$$

Define,

$$L_k' = P_{m-1}\cdots P_{k+1}L_kP_{k+1}^{-1}\cdots P_{m-1}^{-1}$$

Then,

$$(L'_{m-1} \cdots L'_2 L'_1)(P_{m-1} \cdots P_2 P_1 (A = U$$

Algorithm. (Gaussian Elimination without Partial Pivoting $\mathcal{O}\left(\frac{2}{3}m^3\right)$)

```
$U = A, L = I, P = I$
for $k = 1$ to $m - 1$:
    select $i \geq k$ to maximize $|u_{i,k}|$
    $u_{k,k:m} \leftrightarrow u_{i,k:m}$ (interchange two rows)
    $\ell_{k,1:k-1} \leftrightarrow \ell_{i,1:k-1}$
    $p_{k,:} \leftrightarrow p_{i,:}$
    for $j = k + 1$ to $m$:
        $\ell_{j,k} = u_{j,k}/u_{kk}$
        $u_{j,k:m} = u_{j,k:m} - \ell_{j,k} u_{k,k:m}$
    end for
end for
```

Complete pivoting gives a factorization $PAQ = LU$.

5.2 Stability of Gaussian Elimination

Theorem.

Let the LU factorization a non-singular matrix A be computed by Gaussian elimination without pivoting. If A has an LU factorization, then for all sufficiently small $\epsilon_{\text{machine}}$, the factorization completes successfully in floating point arithmetic, and for some δA, the computed matrices $\tilde{L}$ and $\tilde{U}$ satisfy,

$$\tilde{L}\tilde{U} = A + \delta a, \qquad \frac{\|A\|}{\|L\| \, \|U\|} = \mathcal{O}(\epsilon_{\text{machine}})$$

For Gaussian elimination without pivoting, both L and U can be unboundedly large.

5.2.1 Growth Factors

Definition. (Growth Factor)

The growth factor for a factorization $A = LU$ computed using Gaussian elimination without pivoting is defined as the ratio,

$$\rho = \frac{\max_{i,j} |u_{i,j}|}{\max_{i,j} |a_{i,j}|}$$

Theorem.

Let the PLU factorization a non-singular matrix A be computed by Gaussian elimination without pivoting. Then the compute matrices $\tilde{P}$, $\tilde{L}$ and $\tilde{U}$ satisfy,

$$\tilde{L}\tilde{U} = \tilde{P}A + \delta A, \qquad \frac{\|\delta A\|}{\|A\|} = \mathcal{O}(\rho \epsilon_{\text{machine}})$$

for some δA, where ρ is the growth factor for A. If $|\ell_{i,j}| < 1$ for each $i > j$, impliying that there are no ties in teh selection of pivots on exact arithmetic, then $\tilde{P} = P$ for all sufficiently small $\epsilon_{\text{machine}}$.

Theorem.

Gaussian elimination with partial pivoting is backward stable since $\rho \leq 2^{m-1} = \mathcal{O}(1)$ uniformly for all matrices of a given dimension m.

However, the growth factor can go like 2^m so this is mostly a failing of the definition of backward stability to require that the bound do not depend on m.

However in practise Gaussian elimination is stable. (Claimed by [**?**] however there are papers providing examples of real world matrices which are unstable).

6 Cholesky Factorization

Hermetian positive definite matrices can be decomposed into triangular factors twice as quickly as general matrices.

Definition. (Hermetian Matrix)

A matrix A is Hermetian if $A = A^*$

Definition. (Positive Definite)

A positive definite matrices A satisfies $x^*Ax > 0$ for all $x \neq 0$.

Theorem.

Eigenvectors taht correspond to distinct eigenvalues of a Hermetian matrix are orthogonal.

7 Eigenvalues

Definition. (Eigenvalues and Eigenvectors)

Let A be a square matrix. A nonzero vector x is an eigenvector of A, and λ is its corresponding eigenvalue if,

$$Ax = \lambda x$$

Definition. (Spectrum)

The spectrum of a square matrix A, denoted $\Lambda(A)$, is the set of all eigenvalues of A.

Definition. (Eigenvalue Decomposition)

A factorization $A = X\Lambda X^{-1}$, where Λ is diagonal, is called an eigenvalue decomposition.

Definition. (Eigenspace)

Let A be a square matrix. The eigenspace corresponding to an eigenvalue λ, denoted E_λ is the subspace formed by the span of all eigenvectors corresponding to λ. That is, $E_\lambda = \text{null}(A - \lambda I)$.

Definition. (Geometric Multiplicity)

The geometric multiplicity of an eigenvalue λ is the dimension of E_λ.

Definition. (Characteristic Polynomial)

The characteristic polynomial of a square matrix A is defined to be,

$$p_A(z) = \det(zI - A)$$

Theorem.

λ is an eigenvalue of A if and only if $p_A(\lambda) = 0$.

Definition. (Algebraic Multiplicity)

The algebraic multiplicity of an eigenvalue λ is the multiplicity of λ as a root of p_A.

Theorem.

If $A \in \mathbb{C}^{m\times m}$, then A has m distinct eigenvalues (counted with algebraic multiplicity) In particular, if the roots of p_A are simple, then A has m distinct eigenvalues.

7.0.1 Similarity Transforms

Definition.

If $X \in \mathbb{C}^{m\times m}$ is non-singular, then the map $A \mapsto X^{-1}AX$ is called a similarity transformation of A.

Definition. (Similar Matrices)

We say two matrices A and B are similar if there exists a similarity transform between them.

Theorem.

If X is non-singular, then A and $X^{-1}AX$ have the same characteristic polynomial, eigenvalues, and algebraic and geometric multiplicites.

Theorem.

The algebraic multiplicity of an eigenvalue λ is at least as great as its geometric multiplicity.

Definition. (Defective Eigenvalue)

An eigenvalues whose algebraic multiplicity exceeds its geometric multiplicity is called defective.

Definition. (Defective Matrix)

A matrix who has at least one defective eigenvalues is called defective.

Theorem.

A matrix is non-defective if and only if it has an eigenvalue decomposition $X \Lambda X^{-1}$.

Theorem.

Let $A \in \mathbb{C}^{m \times m}$. Then,

$$\det(A) = \prod_{j=1}^{m} \lambda_j, \qquad \operatorname{tr}(A) = \sum_{j=1}^{m} \lambda_j$$

Definition. (Unitarily Diagonalizable)

A matrix is called unitarily diagonalizable if there exists a unitary matrix Q such that $A = Q \Lambda Q^*$

Note that this is both an eigen-decomposition and an SVD (up to sign).

Theorem.

A Hermetian matrix is unitarily diagonalizable, and its eigenvalues are real.

Definition. (Normal Matrix)

A matrix A is normal if $A^*A = AA^*$.

Theorem.

A matrix is unitarily diagonalizable if and only if it is normal.

Definition. (Schur Factorization)

A Schur Factorization of a matrix A is a factorization $A = QTQ^*$ where Q is unitary and T is upper triangular.

Theorem.

Every square matrix has a Schur factorization.

7.1 Overview of Eigenvalue Algorithms

Given a polynomial $p(z) = z^m + a_{m-1}z^{m-1} + \cdots + a_1 z + a_0$ we can construct the matrix,

$$\begin{bmatrix} -z & & & & & -a_0 \\ 1 & -z & & & & -a_1 \\ & 1 & -z & & & -a_2 \\ & & 1 & \ddots & & \vdots \\ & & & \ddots & -z & -a_{m-2} \\ & & & & 1 & (-z - a_{m-1}) \end{bmatrix}$$

The determinant of this matrix is $(-1)^m p(z)$. Thus the matrix,

$$A = \begin{bmatrix} 0 & & & & & -a_0 \\ 1 & 0 & & & & -a_1 \\ & 1 & 0 & & & -a_2 \\ & & 1 & \ddots & & \vdots \\ & & & \ddots & 0 & -a_{m-2} \\ & & & & 1 & -a_{m-1} \end{bmatrix}$$

has eigenvalues equal to the roots of $p(z)$.

In particular this implies that any eigenvalue solver must be iterative as the roots of a polynomial cannot be solved for in general using just addition, subtraction, multiplication, division, and k-th roots.

7.1.1 Schur Factorization and Diagonalization

> **Definition.** (Upper Hessenberg Matrix)
>
> An upper Hessenberg matrix is one for which all entries below the first sub diagonal are zero.

Most eigenvalue algorithms compute the Schur factorization $A = QTQ^*$. This is done by converting A to an upper Hessenberg matrix H, and then to an upper triangular matrix.

If we tried using the same Householder reflectors as for computing the QR factorization then when we comput $Q_1^* A Q_1$ the sparsity may not change. Instead we use Householder reflectors to introduce zeros below the first sub-diagonal. Then we constructing $Q_1^* A Q_1$ the sparsity does decrease.

$$\underset{A}{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix}} \xrightarrow{Q_1^* \cdot} \underset{Q_1^* A}{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \end{bmatrix}} \xrightarrow{\cdot Q_1} \underset{Q_1^* A Q_1}{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \end{bmatrix}}$$

Repeating this process will take A to an upper Hessenberg matrix.

Algorithm. (Householder Reduction to Hessenberg Form ($\mathcal{O}(\frac{10}{3}m^3)$))

```
for $k = 1$ to $m - 2$:
    $x = A_{k+1:m,k}$
    $v_k = \text{sign}(x_1) \left\| x \right\|_2 e_1 + x$
    $v_k = v_k / \left\| v_k \right\|_2$
    $A_{k+1:m,k:m} = A_{k+1:m,k:m} - 2v_k(v_k^* A_{k+1:m,k:m})$
    $A_{1:m,k+1:m} = A_{1:m,k+1:m} - 2(A_{1:m,k+1:m} v_k) v_k^*$
end for
```

If A is Hermetian then the above algorithm will produce a tridiagonal matrix (up to numerical precision) we can then avoid computing these entries and reduce the cost to $\mathcal{O}(\frac{8}{3}m^3)$. This saving is based only on sparsity. By using symmetry we can reduce further to $\mathcal{O}(\frac{4}{3}m^3)$, but no algorithm is given.

7.1.2 Stability

Theorem.

Let the Hessenberg reduction $A = QHQ^*$ of a matrix A be computed as above with the computed factors $\tilde{Q}$ be the exact unitary matrix corresponding to the computed reflection vectors and $\tilde{H}$ the compute Hessenberg reduction. Then for some δA,

$$\tilde{Q}\tilde{H}\tilde{Q}^* = A + \delta A, \qquad\qquad \frac{\|\delta A\|}{\|A\|} = \mathcal{O}(\epsilon_{\text{machine}})$$

7.2 Rayleigh Quotient

Suppose $A = A^T \in \mathbb{R}^{m\times m}$. Then A has real eigenvalues and a complete set of orthogonal eigenvectors.

Definition. (Rayleigh Quotient)

For a real symmetric matrix A, the Rayleigh quotient of a vector x is defined as,

$$r(x) = \frac{x^TAx}{x^Tx}$$

Note that this is the solution to $\min_\alpha \|Ax - \alpha x\|$. That is, the scalar which “acts most like an eigenvalue” for a given x.

We can compute the gradient of $r(x)$ as,

$$\nabla r(x) = \frac{2}{x^Tx}(Ax - r(x)x)$$

From this formula it is clear that at an eigenvector x of A the gradient is zero. Moreover, if $\nabla r(x) = 0$ then $x = 0$ or x is an eigenvector with eigenvalue $r(x)$.

Geometrically the eigenvectors of A are stationary points of the function $r(x)$.

The Rayleigh quotient is a quadratically accurate estimate of an eigenvalue. That is,

$$r(x) - r(v) = \mathcal{O}(\|x - v\|^2) \qquad\qquad \text{as } x \to v$$

where v is an eigenvector of A.

7.3 Power Iteration

Algorithm. (Power Iteration)

```
$v^{(0)} =$ some vector with $\|v\| = 1$
for $k = 1, 2, \dots$:
    $w = Av^{(k-1)}$
    $v^{(k)} = w / \|w\|$
    $\lambda^{(k)} = (v^{(k)})^T A v^{(k)}$
end for
```

Let v be any vector and write it as a linear combination of the orthonormal q_i as,

$$v^{(0)} = a_1 q_1 + a_2 q_2 + \cdots + a_m q_m$$

Then,

$$\begin{aligned} v^{(k)} &= c_k A^k v^{(0)} \\ &= c_k (a_1 \lambda_1^k q_1 + a_2 \lambda_2^k q_2 + \cdots + a_m \lambda_m^k q_m) \\ &= c_k \lambda_1^k \left(a_1 q_1 + a_2 (\lambda_2/\lambda_1)^k + \cdots + a_m (\lambda_m/\lambda_1)^k q_m \right) \end{aligned}$$

Theorem.

Suppose $|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_m| \geq 0$ and $q_1^T v^{(0)} \neq 0$. Then, as $k \to \infty$,

$$\left\| v^{(k)} - (\pm q_1) \right\| = \mathcal{O}\left(\left| \frac{\lambda_2}{\lambda_1} \right|^k \right), \qquad |\lambda^{(k)} - \lambda_1| = \mathcal{O}\left(\left| \frac{\lambda_2}{\lambda_1} \right|^{2k} \right)$$

7.4 Inverse Iteration

For any $\mu \in \mathbb{R}$ not equal to an eigenvalue of A, the eigenvectors of $(A - \mu I)^{-1}$ are the same as those of A, and the eigenvalues are $\{(\lambda_j - \mu)^{-1}\}$ where $\{\lambda_j\}$ are the eigenvalues of A.

If we can guess μ near λ_J then $(\lambda_J - \mu)^{-1}$ may be much larger than $(\lambda_j - \mu)^{-1}$ for all $j \neq J$. Thus power iteration applied to $(A - \mu)^{-1}$ will converge rapidly.

Algorithm. (Inverse Iteration)

```
$v^{(0)} =$ some vector with $\|v\| = 1$
for $k = 1, 2, \dots$:
    Solve $(A - \mu I)w = v^{(k-1)}$ for $w$
    $v^{(k)} = w / \|w\|$
    $\lambda^{(k)} = (v^{(k)})^T A v^{(k)}$
end for
```

Note: Something about mu near lambda being ok. Exercise

Like power iteration, inverse iteration only convergest linearly. However, the rate of this convergence depends on the ratio of the eigenvalues of $(A - \mu I)^{-1}$ which we have some control over by our choice of μ.

Theorem.

Suppose λ_J is the closes eigenvalue to μ and λ_K is the second closes. Suppose further that $q_J^T v^{(0)} \neq 0$. Then, as $k \to \infty$,

$$\left\| v^{(k)} - (\pm q_1) \right\| = \mathcal{O}\left(\left| \frac{\mu - \lambda_J}{\mu - \lambda_K} \right|^k \right), \qquad |\lambda^{(k)} - \lambda_1| = \mathcal{O}\left(\left| \frac{\mu - \lambda_J}{\mu - \lambda_K} \right|^{2k} \right)$$

7.5 Rayleigh Quotient Iteration

Since the Rayleigh quotient gives an approximation we can use this to update our guess for μ.

Algorithm. (Rayleigh Quotient Iteration)

```
$v^{(0)}$ = some vector with $\|v\| = 1$
$\lambda^{(0)} = (v^{(0)})^T A v^{(0)}$
for $k = 1, 2, \ldots$:
    Solve $(A - \lambda^{(k-1)} I) w = v^{(k-1)}$ for $w$
    $v^{(k)} = w / \|w\|$
    $\lambda^{(k)} = (v^{(k)})^T A v^{(k)}$
end for
```

Theorem.

Rayleigh quotient iteration converges to an eigenvalue/eigenvector pair for all except a set of measure zero of starting vector $v^{(0)}$. When it converges, the convergence is ultimately cubig in the sense that if λ_J is an eigenvalue of A and $v^{(0)}$ is sufficiently close to the eigenvector q_J, then as $k \to \infty$,

$$\left\| v^{(k+1)} - (\pm q_J) \right\| = \mathcal{O}(\left\| v^{(k)} - (\pm q_J) \right\|^3)$$

and

$$|\lambda^{(k+1)} - \lambda_J| = \mathcal{O}(|\lambda^{(k)} - \lambda_J|^3)$$

7.6 Opeartion Counts

Power iteration requires multiplication by A and costs $\mathcal{O}(m^2)$ flops. Inverse iteration require solving a linear system and costs $\mathcal{O}(m^3)$ flops. However, A can be factored ahead of time reducing each iteration to $\mathcal{O}(m^2)$. For Rayleigh quotient iteration is is not straightforward to keep it below $\mathcal{O}(m^3)$ each iteration.

7.7 QR Algorithm without Shifts

Algorithm. (“Pure” QR Algorithm)

```
$A^{(0)} = A$
$\lambda^{(0)} = (v^{(0)})^T A v^{(0)}$
for $k = 1, 2, \ldots$:
    $Q^{(k)} R^{(k)} = A^{(k-1)}$
    $A^{(k)} = R^{(k)} Q^{(k)}$
end for
```

Matrices will converge to Schur form, or diagonal form if Hermetian.

At each step we triangularize $A^{(k)}$ by forming $R^{(k)} = (Q^{(k)})^T A^{(k-1)}$ and then multiply on the right by $Q^{(k)}$. This is the same transform we called a “bad idea” earlier as the sparsity pattern did not change. It turns out that while this will not reduce A to triangular form in a finite number of steps, it can be a powerful method to use as a basis for iterative methods.

We can obtain cubic convergence. To do this first reduce A to tridiagonal form

Algorithm. (“Practical” QR Algorithm)

```
$(Q^{(0)})^T A^{(0)} Q^{(0)} = A$ puts $A^{(0)}$ in tridiagonal form
$\lambda^{(0)} = (v^{(0)})^T A v^{(0)}$
for $k = 1, 2, \ldots$:
    Pick a shift $\mu^{(k)}$
    $Q^{(k)} R^{(k)} = A^{(k-1)} - \mu^{(k)} I$
    $A^{(k)} = R^{(k)} Q^{(k)} + \mu^{(k)} I$
    if If any off-diagonal element $A^{(k)}_{j,j+1}$ is sufficiently close to zero, then
```

```
            set $A_{j,j+1} = A_{j+1,j} = 0$ to obtain
            $\begin{bmatrix} A_1 & 0 \\ 0 & A_2 \end{bmatrix} = A^{(k)}$
            and now apply the algorithm to each $A_1$ and $A_2$
        end if
    end for
```

7.7.1 Unnormalized Simultaneous Iteration

If we do power iteration like $V^{(k)} = A^k V^{(0)}$, where $V^{(0)} = [v_1^{(0)}, \ldots, v_n^{(0)}]$, then we may expect the columns of $V^{(k)}$ to converge to the span of the first n eigenvectors.

However, the columns of $V^{(k)}$ will all be mostly in the direction of the first eigenvector. Therefore we need to compute a QR factorization of $V^{(k)}$ to obtain a well conditioned basis.

Theorem.

Suppose this iteration is carried out and that,

$$|\lambda_1| > |\lambda_2| > \cdots > |\lambda_n| \geq |\lambda_{n+1}| \geq |\lambda_{n+2}| \geq \cdots \geq |\lambda_m|$$

and that all leading principle minors of $\hat{Q}^T V^{(0)}$ are nonsingular, where $\hat{Q}$ denotes the first m eigenvectors (By leading principal minors, we mean all upper-left square submatrices of $\hat{Q}^T V^{(0)}$. This is equivalent to $\hat{Q}^T V^{(0)}$ having an LU decomposition).

Then as $k \to \infty$ the columns of the matrix $\hat{Q}^{(k)}$ converge linearly to the eigenvectors of A:

$$\left\| q_j^{(k)} - (\pm q_j) \right\| = \mathcal{O}\left(\left(\max_{1 \leq k \leq n} \left| \frac{\lambda_{k+1}}{\lambda_k} \right| \right)^k \right)$$

for each j with $1 \leq j \leq n$.

7.7.2 Simultaneous Iteration

As $k \to \infty$, the vectors $v_1^{(k)}, \ldots, v_n^{(k)}$ all converge to multiples of the same dominant eigenvector q_1. The span does converge to something useful, but if compute numerically the rounding errors will be too large as the vectors are a highly ill-conditioned basis for the desired space.

To fix this we must orthonormalize at every step.

Algorithm. (Simultaneous Iteration)

```
Pick $(Q^{(0})$ with orthonormal columns
for $k = 1, 2, \ldots$:
    $Z = AQ^{(k-1)}$
    $Q^{(k)}R^{(k)} = Z$ reduced QR factorization
end for
```

Theorem.

The matrices $\hat{Q}^{(k)}$ are the same as those without orthonormalizing each iteration, and converge in the same way.

7.7.3 Simultaneous Iteration equivalent to QR Algorithm

Note: How important is this?

7.8 QR Algorithm with Shifts

7.9 Other Eigenvalue Algorithms

7.10 Computing the SVD

One way to compute the SVD may be to solve the eigenproblem of A^*A. However, this is unstable since this eigenvalue problems could be much more sensitive to perturbations.

We could instead solve the eigenvalue problem for,

$$H = \begin{bmatrix} 0 & A^* \\ A & 0 \end{bmatrix}$$

which has eigen-decomposition,

$$\begin{bmatrix} 0 & A^* \\ A & 0 \end{bmatrix} \begin{bmatrix} V & V \\ U & -U \end{bmatrix} = \begin{bmatrix} V & V \\ U & -U \end{bmatrix} \begin{bmatrix} \Sigma & 0 \\ 0 & -\Sigma \end{bmatrix}$$

The singular values and vectors can be extracted from the eigenvalues and vectors of H. This methods is stable, and most SVD methods are based on this.

8 Finite Difference Approximations

Let $u(x)$ be some sufficiently smooth function. We would like to approximate $u'(x)$ based only on values of u. Some possible choices are,

$$\begin{aligned} D_+u(x) &:= \frac{u(x+h)-u(x)}{h} && \text{right approximation} \\ D_-u(x) &:= \frac{u(x)-u(x+h)}{h} && \text{left approximation} \\ D_0u(x) &:= \frac{u(x+h)-u(x-h)}{2h} && \text{centered approximation} \end{aligned}$$

> **Definition.** (One Sided Approximation)
>
> A one sided approximation to $u'(x)$ requires that u be evaluated only one one side of x.

If $E(h) \approx Ch^p$ then $\log|E(h)| \approx \log|C| + p\log h$, so plotting the error on a log-log scale can be useful for finding the order of the error.

8.1 Truncation Errors

> **Definition.** (Truncation Error of Finite Difference Approximation)
>
> Amount by which the finite difference approximation fails to satisfy the derivative it approximates.

8.2 Deriving Finite Difference Approximations

We can derive finite difference approximations by method of undetermined coefficients an Taylor expansions.

8.3 Second/Higher Order Derivatives

We have standard second order approximation to u'',

$$D^2u(x) := \frac{u(x+h)+u(x-h)-2u(x)}{h^2} = u''(x) + \frac{1}{12}h^2u''''(x) + \mathcal{O}(h^4)$$

Note: RICHARDSON EXTRAPOLATION

8.4 General Approach to Deriving Coefficients

9 Steady State and Boundary Value Problems

9.1 Boundary Conditions

> **Definition.** (Dirichlet Boundary Condition)
>
> A Dirichlet boundary condition means the value of the function is given at a point.

> **Definition.** (Neumann Boundary Condition)
>
> A Neumann boundary condition means the value of the derivative of the function is given at a point.

9.2 Error for Finite Difference Method

We approximate the solution evaluated on the mesh U by solving $A\hat{U} = F$.

> **Definition.** (Global Error)
>
> The global error for a finite difference method is the difference between the computed solution and the actual solution at the given mesh points.
>
> $$E = U - \hat{U}$$

> **Definition.** (Local Truncation Error)
>
> The local truncation error is the amount by which the finite difference method fails to satisfy the true solution at each mesh point.
>
> $$\tau = A\hat{U} - F$$

We then have,

$$AE = -\tau$$

with boundary conditions $E_0 = E_{m+1} = 0$. Therefore the global error satisfies the same finite difference equations as the original equations.

Definition. (Stable)

Suppose a finite difference method for a linear BVP gives a sequence of matrix equations of the form $A^{(h)}E^{(h)} = F^{(h)}$, where h is the mesh width. We say that the method is stable if $(A^{(h)})^{-1}$ exists for all h sufficiently small and if there is a constant C, independent of h, such that,

$$\left\|(A^{(h)})^{-1}\right\| \leq C, \qquad \forall h \text{ sufficiently small}$$

Definition. (Consistency)

We say a method is consistent with the differential equation and boundary conditions if,

$$\left\|\tau^{(h)}\right\| \to 0 \qquad \text{as} \qquad h \to 0$$

Definition. (Convergence)

We say a method is convergent if,

$$\left\|E^{(h)}\right\| \to 0 \qquad \text{as} \qquad h \to 0$$

Theorem.

$$\text{consistency} + \text{stability} \implies \text{convergence}$$

$$\mathcal{O}(h^p) \text{ local truncation error} + \text{stability} \implies \mathcal{O}(h^p) \text{ global error}$$

9.3 Deferred Corrections

If we had an exact expression for the LTE, τ, then we could solve $AE = -\tau$ for E, and in turn use this to compute the exact solution. However in general this is not possible.

Sometimes we could eliminate some of the error. For instance, if we are solving $u''(x) = f(x)$, the dominant term will be $h^2u''''(x)/12$, and we know $u'''' = f''$.

Alternatively we could use the compute solution U to estimate τ_j, and then solve another problem to eliminate E.

9.4 Spectral Methods

Note: IS THIS USEFUL?

10 Finite Element Method

Suppose we have differential operator $\mathcal{L} = -D^2 + r(x)$ and wish to solve,

$$\mathcal{L}u = f, \qquad 0 < x < 1, \qquad u(0) = u(1) = 0$$

Define,

$$\langle f, g\rangle = \int_0^1 f(x)g(x)\mathrm{d}x$$

Observe that $\mathcal{L}u = f$ if and only if for any function φ,

$$\langle \mathcal{L}u, \varphi\rangle = \langle f, \varphi\rangle$$

We can write this explicitly in integral as,

$$\int_0^1 \left(-u''(x) + r(x)u(x)\right)\varphi(x)\mathrm{d}x = \int_0^1 f(x)\varphi(x)\mathrm{d}x$$

This is called the weak form the the differential equation

Integrating the left hand side by parts we obtain,

$$\left[-u'(x)\varphi(x)\right]_0^1 + \int_0^1 u'(x)\varphi'(x)\mathrm{d}x + \int_0^1 r(x)u(x)\varphi(x)\mathrm{d}x = \int_0^1 f(x)\varphi(x)\mathrm{d}x$$

Note that since we have integrated we now only require that our solution u be first differentiable.

We would like to find $u \in S$, where S is some “trial space” of functions such that the weak form of the differential equation is satisfied for all $\varphi \in \mathcal{L}^2[0,1]$. However, this is still too hard of a problem, so we will restrict to only satisfying the equation for some $\varphi \in T$, where T is some “test space” (often chosen to be S).

If we pick $S = T$ the be the set of all continuous piecewise (on some fixed partition) polynomial functions satisfying the boundary conditions, this gives the Galerkin finite element method.

In particular, suppose $S = \{\varphi_i\}_{i=1}^n$. Then for every $u \in S$, there are coefficients c_i such that,

$$u = \sum_{i=1}^{n} c_i \varphi_i$$

We solve for the coefficients c_i so that,

$$\sum_{j=1}^{n} c_j \langle \mathcal{L}\varphi_j, \varphi_i \rangle = \langle \mathcal{L}u, \varphi_i \rangle = \langle f, \varphi_i \rangle, \qquad i = 1, \ldots, n$$

In matrix form we have,

$$\begin{bmatrix} \langle \mathcal{L}\varphi_1, \varphi_1 \rangle & \langle \mathcal{L}\varphi_2, \varphi_1 \rangle & \cdots & \langle \mathcal{L}\varphi_n, \varphi_1 \rangle \\ \langle \mathcal{L}\varphi_1, \varphi_2 \rangle & \langle \mathcal{L}\varphi_2, \varphi_2 \rangle & \cdots & \langle \mathcal{L}\varphi_n, \varphi_2 \rangle \\ \vdots & \vdots & & \vdots \\ \langle \mathcal{L}\varphi_1, \varphi_n \rangle & \langle \mathcal{L}\varphi_2, \varphi_n \rangle & \cdots & \langle \mathcal{L}\varphi_n, \varphi_n \rangle \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} = \begin{bmatrix} \langle f, \varphi_1 \rangle \\ \langle f, \varphi_2 \rangle \\ \vdots \\ \langle f, \varphi_n \rangle \end{bmatrix}$$

Note that we use $\langle \mathcal{L}f, g \rangle$ to denote the inner product after integrating by parts.

11 Elliptic Equations

A common form of equations is,

$$a_1 u_{xx} + a_2 u_{xy} + a_3 u_{yy} + a_4 u_x + a_5 u_y + a_6 u = f$$

where $a_2^2 - 4a_1a_3 < 0$.

Such equations often arise as steady-state equations for some time-dependent physical problem.

11.1 Laplacian Stencils

The 2-dimensional Laplace operator ∇^2 is defined as $\nabla^2 u = u_{xx} + u_{yy}$.

11.1.1 5-point Laplacian

On a rectangular grid a simple way to approximate this operator would be to use centered, second order accurate, finite difference approximations. This gives the

5-point operator,

$$\nabla_5^2 u_{i,j} := \frac{1}{h^2}(u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} - 4u_{ij})$$

If we order the grid from bottom left to top right we will obtain the system $AU = F$ where,

$$A = \frac{1}{h^2}\begin{bmatrix} T & I & & & \\ I & T & I & & \\ & I & T & I & \\ & & \ddots & \ddots & \ddots \\ & & & I & T \end{bmatrix}, \qquad T = \begin{bmatrix} -4 & 1 & & & \\ 1 & -4 & 1 & & \\ & 1 & -4 & 1 & \\ & & \ddots & \ddots & \ddots \\ & & & 1 & -4 \end{bmatrix}$$

and

$$U = \begin{bmatrix} u^{[1]} \\ u^{[2]} \\ \vdots \\ u^{[m]} \end{bmatrix}, \qquad u^{[j]} = \begin{bmatrix} u_{1j} \\ u_{2j} \\ \vdots \\ u_{mj} \end{bmatrix}$$

11.1.2 Ordering of Equations

Other orderings such as “red-black” are possible. These will give a system with different structure, and are (obviously) equivalent to a permutation of the variables and equation orders. However the system may have nicer properties for certain solvers.

11.1.3 9-point Laplacian

We could define the 9-point operator,

$$\begin{aligned}\nabla_9^2 u_{i,j} := \frac{1}{h^2}(&4u_{i-1,j} + 4u_{i+1,j} + 4u_{i,j-1} + 4u_{i,j+1} \\ &+ u_{i-1,j-1} + u_{i-1,j+1} + u_{i+1,j-1} + u_{i+1,j+1} - 20u_{ij})\end{aligned}$$

This satisfies,

$$\nabla_9^2 u(x_i, y_j) = \nabla^2 u + \frac{1}{12}h^2(u_{xxxx} + 2u_{xxyy} + u_{yyyy}) + \mathcal{O}(h^4)$$

Now observe,

$$u_{xxxx} + 2u_{xxyy} + u_{yyyy} = \nabla^2(\nabla^2(u) = \nabla^2 f$$

Thus, we can correct for the leading order error term in the 9-point operator if we are able to compute $\nabla^2 f$. In fact, even if we are unable to compute this, if f is sufficiently smooth, $\nabla_5^2 f$ can be used instead.

12 Iterative Methods

12.1 Jacobi, Gauss–Seidel, Successive Overrelaxation

Note: Include motivation

Algorithm. (Simple Iteration)

```
Pick one of $M = \begin{cases} \operatorname{diag}(A) & \text{Jacobi Iteration} \\ \operatorname{tril}(A) & \text{GS Iteration} \\ \omega^{-1}\operatorname{diag}(A) - \operatorname{tril}(A, k=-1) & \text{SOR} \end{cases}$
Given initial guess $x_0$ compute, $r_0 = b - Ax_0$ and solve $Mz_0 = r_0$
for $k = 1, 2, \dots$:
    $x_k = x_{k-1} + z_{k-1}$
    $r_k = b - Ax_k$
    Solve $Mz_k = r_k$
end for
```

12.1.1 Convergence

Theorem.

Simple iteration converges if and only if $\rho(I - M^{-1}A) < 1$.

Proof.

We have,

$$x_k = x_{k-1} + M^{-1}r_{k-1} = x_{k-1} + M^{-1}Ae_{k-1}$$

Thus,

$$e_k = A^{-1}b - x_k = A^{-1}b - x_{k-1} - M^{-1}Ae_{k-1} = (I - M^{-1}A)e_{k-1} = (I - M^{-1}A)^k e_0$$

Suppose $\rho(I - M^{-1}A) < 1$. By Theorem,

$$\rho(I - M^{-1}A) = \lim_{k\to\infty} \left\|(I - M^{-1}A)^k\right\|^{1/k}$$

Since $x \mapsto x^k$ is continuous,

$$\lim_{k\to\infty} \left\|(I - M^{-1}A)^k\right\| = \lim_{k\to\infty} \rho(I - M^{-1}A)^k = 0$$

Therefore,

$$\lim_{k\to\infty} \|e_k\| \leq \lim_{k\to\infty} \left\|(I - M^{-1}A)^k\right\| \|e_0\| = 0$$

Suppose $\rho(I - M^{-1}A) \geq 1$. Then There is some (nonzero) eigenvector v of $I - M^{-1}A$ with corresponding eigenvalue $\lambda \geq 1$. Thus, if we pick some x_0 such that $e_0 = v$ we will have,

$$\|e_k\| = \left\|(I - M^{-1}A)^k v\right\| = |\lambda^k| \|v\| > 0 \qquad \square$$

12.2 Conjugate Gradient

Conjugate gradient is a method for solving $Ax = b$ when A is symmetric positive definite. In exact arithmetic it will find the exact solution in at most n steps.

12.2.1 Motivation for Algorithm

Suppose that at each step we will minimize the A-norm of the error over $\mathcal{K}_k = \text{span}\{r_0, Ar_0, \ldots, A^{k-1}r_0\}$.

Let $\{p_0, p_1, \ldots, p_{k-1}\}$ be a basis for $\mathcal{K}_k$. We will pick x_k of the form,

$$x_k = x_0 + \sum_{j=0}^{k-1} c_j p_j$$

This gives the relation for the error at step k,

$$e_k = e_0 - \sum_{j=0}^{k-1} c_j p_j$$

If we minimize $\|e_k\|_A$ we must pick,

$$c_j = \frac{\langle e_0, Ap_j \rangle}{\langle p_j, Ap_j \rangle}$$

While we do not know e_0, since A is SPD we have,

$$\langle e_0, Ap_j \rangle = \langle Ae_0, p_j \rangle = \langle r_0, p_j \rangle$$

In the present form this algorithm minimizes e_k over the entire space $\mathcal{K}_k$ at each step. It also requires that the basis $\{p_0, p_1, \ldots, p_{k-1}\}$ be constructed ahead of time.

12.2.2 Improvement

Note that at each step of the above algorithm we have,

$$x_k = x_{k-1} + c_{k-1} p_{k-1}, \qquad\qquad e_k = e_{k-1} - c_{k-1} p_{k-1}$$

Moreover, $e_{k-1} \perp_A \operatorname{span}\{p_0, p_1, \ldots, p_{k-2}\}$. Thus,

$$c_{k-1} = \frac{\langle e_{k-1}, Ap_{k-1} \rangle}{p_{k-1}, Ap_{k-1}} = \frac{\langle r_{k-1}, p_{k-1} \rangle}{p_{k-1}, Ap_{k-1}}$$

With this update we no longer need to minimize over the entire space.

First observe that,

$$r_k = b - Ax_k = b - A(x_{k-1} + c_{k-1} p_{k-1}) = r_{k-1} - c_{k-1} A p_{k-1}$$

From this expression it is clear that $r_k \in \mathcal{K}_{k+1}$.

We now note that,

$$r_k \perp \operatorname{span}\{p_0, p_1, \ldots, p_{k-1}\} = \operatorname{span}\{r_0, Ar_0, \ldots A^{k-1} r_0\}$$

Thus,

$$r_k \perp_A \operatorname{span}\{r_0, Ar_0, \ldots, A^{k-2} r_0\} = \operatorname{span}\{p_0, p_1, \ldots, p_{k-2}\}$$

Since r_k is A-orthogonal to all of these p_j, we can obtain p_k by A-orthogonalizing r against p_{k-1}. That is,

$$p_k = r_k - b_k p_k, \qquad\qquad b_k = \frac{\langle r_k, Ap_{k-1}\rangle}{p_{k-1}, Ap_{k-1}}$$

This finally gives the standard presentation of the Conjugate Gradient method.

Algorithm. (HS Conjugate Gradient)

```
Given initial guess $x_0$ compute, $r_0 = b - Ax_0$ and set $p_0 = r_0$
for $k = 1, 2, \ldots$:
    $a_{k-1} = \langle r_{k-1}, p_{k-1}\rangle \,/\, \langle p_{k-1}, Ap_{k-1}\rangle$
    $x_k = x_{k-1} + a_{k-1}p_{k-1}$
    $r_k = r_{k-1} - a_{k-1}Ap_{k-1}$
    $b_{k-1} = \langle r_k, Ap_{k-1}\rangle \,/\, \langle p_{k-1}, Ap_{k-1}\rangle$
    $p_k = r_k - b_k p_{k-1}$
end for
```

12.3 GMRES

Note: Should we include other methods such as BiCGSTAB

12.4 Multigrid Methods

Simple iteration is slow to eliminate low frequency error on a fine mesh. We can remove this low frequency component by estimating it on a much coarser grid, and then subtracting away an interpolation of this estimate on the fine grid.

This idea can be improved further by cycling over a range of mesh widths.

Note: How much should I know the exact details. This doesn't seem too important or difficult.

Note: When would I use this vs. conjugate gradient?

13 Other Direct Methods

Note: DID WE DO THIS?

I remember fast Poisson solvers and stuff.

14 The Initial Value Problem for ODEs

We would like to solve,

$$u'(t) = f(u(t), t), \qquad t > t_0, \qquad u(t_0) = \eta$$

14.1 Linear ODEs

> **Definition.** (Linear ODE)
>
> A system of ODEs $u'(t) = f(u(t), t)$ is called linear if,
>
> $$f(u, t) = A(t)u + g(t)$$
>
> where $A(t) \in \mathbb{R}^{s \times s}$ and $g(t) \in \mathbb{R}^s$.

14.2 Some stuff on Existence and Uniqueness. Is this important?

14.3 Basic Numerical Methods

The simplest method is forward Euler,

$$\frac{U^{n+1} - U^n}{k} = f(U^n), \qquad n = 0, 1, \ldots$$

It is possible to solve this equation explicitly for U^{n+1} in terms of U^n. Therefore the method is called explicit.

Another method is backward Euler,

$$\frac{U^{n+1} - U^n}{k} = f(U^{n+1})$$

However, this is now an implicit method, and U^{n+1} must be solve for. One way to do this would be to find a root to,

$$g(u) = u - kf(u) - U^n$$

Another implicit method is the trapezoidal method, obtained by averaging the two Euler method,

$$\frac{U^{n+1} - U^n}{k} = \frac{f(U^{n+1}) + f(U^n)}{2}$$

14.3.1 Truncation and One-step Errors

The truncation error is defined as before. The one step error is the error in U^{n+1} if all the previous points used were computed exactly. Thus the one step error is always has an extra factor of k and some constant compared with the local truncation error.

14.4 Taylor Series Methods

Forward Euler can be viewed as a Taylor series expansion in the sense that it arises by dropping all terms of order k^2 or higher.

Other methods can be constructed this way. However, these methods require being able to compute derivatives of f.

14.5 Runge–Kutta Methods

Rather than explicitly computing higher derivatives finite difference approximations can be designed to model these terms automatically.

> **Definition.** (Runge–Kutta Method)
>
> A genera r-stage Runge–Kutta method has the form,
>
> $$\begin{aligned} Y_i &= U^n + k\sum_{j=1}^{r} a_{ij} f(Y_j, t_n + c_j k), && i = 1, 2, \ldots, r \\ U^{n+1} &= U^n + k\sum_{j=1}^{r} b_j f(Y_j, t_n + c_j k) \end{aligned}$$

Note: should i think of this like a matrix?

Consistency requires,

$$\begin{aligned} \sum_{j=1}^{r} a_{ij} &= c_i, && i = 1, 2, \ldots, r \\ \sum_{j=1}^{r} b_j &= 1 \end{aligned}$$

We can represent this using a "Butcher tableau" as,

$$\begin{array}{c|ccc} c_1 & a_{11} & \cdots & a_{1r} \\ \vdots & \vdots & & \vdots \\ c_r & a_{r1} & \cdots & a_{rr} \\ \hline & b_1 & \cdots & b_r \end{array}$$

If $a_{ij} = 0$ for all $j \geq i$ the method is explicit. If $a_{ij} = 0$ for all $j > i$ the method is diagonally implicit. This is means only r equations each of size s must be solved, rather than a coupled set of sr equations.

14.5.1 Embedded Methods and Error Estimation

Most practical ODE solvers do not use a fixed time step, but rather adjust it to try to achieve some given error bound. A simple way to do this is to take a step with two methods, one of order p and one of order $p+1$. Assuming the step is small enough that the higher order method generates a better approximation, then the difference of the two results is an estimate of the one step error in the lower order method.

In general this could double the cost of computing the solution since two methods would have to be run. However, there are many methods where the same coefficients can be used for both the methods of order p and $p+1$. This saves computational costs.

14.6 One-step vs Multistep methods

One step methods are self starting, meaning they only require a single initial value. The timestep can be changed at any step, and if the solution is not smooth at a single point, as long as this point is part of the mesh the solution usually is fully accurate.

On the other hand, Taylor series methods require knowing derivatives, and RK methods require many function evaluations. This can be expensive, especially for implicit methods.

14.7 Linear Multistep Methods

> **Definition.** (Linear Multistep Method)
>
> A linear multistep method has the form,
>
> $$\sum_{j=0}^{r} \alpha_j U^{n+1} = k \sum_{j=0}^{r} f(U^{n+j}, t_{n+1})$$

For LMMs it is easy to compute the LTE,

$$\begin{aligned}\tau(t_{n+r}) &= \frac{1}{k}\left(\sum_{j=0}^{r} \alpha_j U^{n+1} - k\sum_{j=0}^{r} f(U^{n+j}, t_{n+1})\right)\\ &= \frac{1}{k}\left(\sum_{j=0}^{r} \alpha_j\right) u(t_n) + \sum_{q=1}^{\infty} k^{q-1}\left(\sum_{j=1}^{r}\left(\frac{1}{q!} j^q \alpha_j - \frac{1}{(q-1)!} j^{q-1}\beta_j\right)\right) u^{(q)}(t_n)\end{aligned}$$

For consistency we require,

$$\sum_{j=0}^{r} \alpha_j = 0 \qquad\qquad \sum_{j=0}^{r} j\alpha_j = \sum_{j=1}^{r} \beta_j$$

> **Definition.** (Characteristic Polynomials)
>
> The characteristic polynomials $\rho(\zeta)$ and $\sigma(\zeta)$ of a LMM are,
>
> $$\rho(\xi) = \sum_{j=0}^{r} \alpha_j \zeta^j, \qquad\qquad \sigma(\zeta) = \sum_{j=0}^{r} \beta_j \zeta^j$$

14.7.1 Starting Values

One difficulty with LMMs with $r > 0$ is that we need the values of $U^0, U^1, \ldots, U^{r-1}$ before we can apply the LMM. We know $U^0 = \eta$, however the other values are not known.

In general, if we are using a r-step method of order p, we need to generate $U^0, U^1, \ldots U^{r-1}$ using a method that has a one-step error of order p (LTE of order $p-1$).

15 Zero-Stability and Convergence for IVPs

15.1 Convergence

Definition. (Convergence Method)

An r-step method is said to be convergent if applying the method to any ODE $u' = f(u(t), t)$ with $f(u, t)$ Lipschitz continuous in u, with any set of starting values converging to η as $k \to 0$, and for every fixed time $T > 0$ for which the ODE has a unique solution we have,

$$\lim_{\substack{k \to 0 \\ Nk = T}} U^N = u(T)$$

15.2 Solving Linear Difference Equations

Consider the general homogeneous linear difference equation,

$$\sum_{j=0}^{r} \alpha_j U^{n+j} = 0$$

Let $\rho(\zeta) = \sum_{j=0}^{r} \alpha_j \zeta^j$, and let $\lambda_1, \ldots, \lambda_k$ be the roots of $\rho(\zeta)$ with multiplicities $\beta_1, \ldots, \beta_k$. Then the solution to the above difference equation is,

$$U^n = \sum_{i=1}^{k} \left(c_{i,1}\lambda_i^n + c_{i,2} n \lambda_i^n + \cdots + c_{i,\beta_i} n^{\beta_i - 1} \lambda_i^n \right)$$

Definition. (Zero-Stable)

An r-step LMM is said to be zero-stable if the roots $\zeta_1, \ldots, \zeta_r$ of the characteristic polynomial $\rho(\zeta)$ satisfy,

$$\begin{aligned} &|\zeta_j| \leq 1 \text{ for } j = 1, 2, \ldots, r \\ &\text{if } |\zeta_j| = 0 \text{ then } \zeta_j \text{ is a simple root} \end{aligned}$$

Theorem.

For LMMs applied to the IVP $u'(t) = f(u(t), t)$,

$$\text{consistency} + \text{zero-stability} \implies \text{convergence}$$

We can think of zero stable as meaning “stable in the limit as $k \to 0$”.

However, zero-stability says nothing about how small we need k to be.

16 Absolute Stability for ODEs

Consider forward Euler applied to the test problem $u'(t) = \lambda u(t)$. We have,

$$U^{n+1} = (1 + k\lambda)U^n = (1 + k\lambda)^{n+1}U^0$$

If the actual solution is decaying we want the numerical solution to also decay. For this we need $|1 + k\lambda| \leq 1$.

Note: But why is this useful? What if $\lambda > 0$?

> **Definition.** (Region of Absolute Stability)
>
> The region of absolute stability for a LMM is the set of points $z \in \mathbb{C}$ for which $\pi(\zeta; z) = \rho(\zeta) - z\sigma(\zeta)$ satisfies the root condition.

Note: Why do we never have a neighborhood of the origin?

16.1 Practical Choice of Step Size

We need to choose k small enough so that the local truncation error is acceptably small. At this time, we also need that $k\lambda$ lies within the region of absolute stability for the given problem.

16.2 Systems of ODEs

How do we relate a given problem to the test equation?

Suppose we have the linear system $u'(t) = Au(t)$.

In the linear case, if A is normal (unitarily diagonalizable) then all the eigenvalues being in the region of stability is enough

Note: ENOUGH FOR WHAT. WHY ARE THERE NO PROPER DEFINITIONS IN APPLIED MATH

If the systems are nonlinear the Jacobian may give a good linear approximation if the system is not varying too quickly in time.

16.3 Plotting the Stability Regions

16.3.1 Boundary Locus Method for LMMs

If a point z is on the boundary of the stability region, then $\rho(\zeta) - z\sigma(\zeta)$ has at least one root with modulus one. Thus, $\pi(e^{i\theta}; z) = 0$ for some θ. Therefore, to find the boundary we could plot the points,

$$z(\theta) = \rho(e^{i\theta})/\sigma(e^{i\theta})$$

This will give all the boundary points, however not all points of this form need to be on the boundary.

16.4 Plotting Stability Regions of One-step Methods

When we apply a one-step method to the test equation $u' = \lambda u$ we typically obtain an expression of the form,

$$U^{n+1} = R(z)U^n, \qquad z = k\lambda$$

The absolute stability region is,

$$\{z \in \mathbb{C} : |R(z)| \leq 1\}$$

In this special case we can use contour plots to find the boundary of the stability region.

16.5 Relative Stability Regions and Order Stars

Note: IS THIS RELEVANT??

17 Stiff ODEs

Roughly speaking a problem is stiff if we are computing a solution which is smooth and slowly varying (relative to the time interval of the computation), but in a context where nearby solution curves are much more rapidly varying. That is, if we perturb our solution slightly the resulting solution curve has rapid variation.

> **Definition.** (A-stability)
>
> An ODE method is A-stable if its region of absolute stability contains the entire left half plane, $\{z \in \mathbb{C} : \text{Re}(z) \leq 0\}$.

It turns out that any A-stable LMM is at most second order accurate.

> **Definition.** (L-stablity)
>
> A one-step method is L-stable if it is A-stable and $R(z) \to 0$ as $z \to 0$.

18 Diffusion Equations and Parabolic Problems

We will look at the heat equation,

$$u_t = \kappa u_{xx}$$

This is the classic example of a parabolic equation.

We need initial conditions at time zero, as well as boundary conditions.

We generally discritize in time and space on a discrete rectangular grid.

18.1 Local Truncation Error and Order of Accuracy

We define LTE as usual, by plugging the exact solution into the discrete approximation and finding how much it fails to satisfy the equation. We now have some error in time and some error in space.

18.2 Method of Lines Discritizations

We would like to develop stability theory for PDEs. One way to do this is to view our PDE methods in terms of IVP solvers. That is, discritizing in space and then considering how IVP methods work on the resulting (often linear) system.

In this case we obtain a method like,

$$U'(t) = AU(t) + g(t)$$

18.3 Stability Theory

We can now apply stability results from before to the MOL discritization. However, as we refine the space mesh A and its eigenvalues will change.

For example, suppose the discritization $U' = AU + g$ is obtained for $u' = u_{xx}$ using a standard second order differencing scheme. Then the eigenvalues of A are $\lambda_p = 2(\cos(p\pi h) - 1)/h^2$ for $p = 1, 2, \ldots, m$.

These are all on the negative real axis, with the one farthest from the origin being bounded by $-4/h^2$ (where we can get arbitrarily close to this bound by making the mesh fine enough).

If we use forward Euler to solve $U' = AU + g$ then we need $|1 + k\lambda| \leq 1$ for all λ. Thus we require, $-2 \leq -4k/h^2 \leq 0$. This means we need,

$$\frac{k}{h^2} \leq \frac{1}{2}$$

However, if we use Crank-Nicholson, which is obtained by applying the trapezoid method, the method is stable for any step size $k > 0$ as the region of absolute stability of the trapezoid method is the entire left half plane.

18.4 Stiffness of the Heat Equation

The heat equation is really stiff. In the continuous case it is infinitely stiff. In the discrete case it is not infinitely stiff, but as we take $h \to 0$ it is unboundedly stiff.

18.5 Convergence

In general we cannot just take $h \to 0$ and $k \to 0$ independently and except convergence.

> **Definition.** (Lax–Richtmyer Stable)
>
> A linear method of the form $U^{n+1} = B(k)U^n + b^n(k)$ is Lax–Richtmyer stable if, for each time T, there is a constant $C_T > 0$ such that,
>
> $$\|B(k)^n\| \leq C_T$$
>
> for all $k > 0$ and integers n for which $kn \leq T$.

> **Theorem.** (Lax Equivalence Theorem)
>
> A consistent linear method of the form $U^{n+1} = B(k)U^n + b^n(k)$ is stable if and only if it is Lax-Richtmyer stable.

Note: What about multistep method? And non-linear methods?

18.6 Von Neumann Analysis

To show a finite difference method is stable in 2-norm we have to show that $\|B\|_2 \leq 1 + \alpha k$, where $B = B(k)$ as before. This ammounts to showing there is a constant α such that,

$$\left\|U^{n+1}\right\|_2 \leq (1 + \alpha k) \left\|U^n\right\|_2$$

This is difficult to show since everything in coupled together. That is, U_j^{n+1} depends on values of U^n at many points.

However, using Parseval's relation it is sufficient to show,

$$\left\|\hat{U}^{n+1}\right\| \leq (1 + \alpha k) \left\|\hat{U}^n\right\|_2$$

Where we have used the notation $\hat{X}$ to denote the Fourier transform of X, and by Parseval, $\left\|\hat{X}\right\|_2 = \|X\|_2$.

In this case it turns out that $\hat{U}^n(\xi)$ decouples from all other wave numbers. For a two level method this has the form,

$$\hat{U}^{n+1}(\xi) = g(\xi)\hat{U}^n(\xi)$$

We call the factor $g(\xi)$ the amplification factor for the wave number ξ.

If we can show $g(\xi) \leq 1 + \alpha k$ for each ξ, then,

$$|\hat{U}^{n+1}| \leq (1 + \alpha k)|\hat{U}^n$$

and so,

$$\left\|\hat{U}^{n+1}\right\| \leq (1 + \alpha k) \left\|\hat{U}^n\right\|_2$$

To use this analysis we replace,

$$U_j^n \leftarrow e^{ijh\xi}, \qquad\qquad U_j^{n+1} \leftarrow g(\xi)e^{ijh\xi}$$

Note: I think it is misleading to set U_j^n as its fourier transform. Double check this though.

Note: See p 213 for three level method

18.7 Multidimensional Problems

In two space dimensions the heat equation takes the form,

$$u_t = u_{xx} + u_{yy}$$

If we discretize in space using the 5-point Laplacian, and in time using trapezoid method we obtain the two-dimensional version of Crank–Nicolson,

$$U_{ij}^{n+1} = U_{ij}^n + \frac{k}{2}\left[\nabla_h^2 U_{ij}^n + \nabla_h^2 U_{ij}^{n+1}\right]$$

Equivalently,

$$\left(I - \frac{k}{2}\nabla_h^2\right)U_{ij}^{n+1} = \left(I + \frac{k}{2}\nabla_h^2\right)U_{ij}^n$$

This is a linear system whose left hand side matrix has eigenvalues,

$$\lambda_{p,q} = 1 - \frac{k}{h^2}\left[(\cos(p\pi h) - 1) + (\cos(q\pi h) - 1)\right]$$

The largest eigenvalue of this matrix is $\mathcal{O}(k/h^2)$ while the smallest is $1+\mathcal{O}(k)$. Thus the condition number of A is $\mathcal{O}(k/h^2)$. This is in contrast to the condition number of $\mathcal{O}(1/h^2)$ of the 5-point Laplacian alone.

We therefore expect iterative methods to converge quicker on this system. Moreover, we can use the solution at a previous step as a good starting guess for the solution at the next step. We could also use an explicit method (such as forward Euler) to guess at the solution, and use this guess as the starting point for an iterative method.

In practice it is often the case that only one or two iterations are needed in each time step to obtain global second order accuracy.

18.8 The Locally One-Dimensional Method

Rather than solving the coupled equation above, we could solve $u_t = u_{xx}$ in one time step, and then $u_t = u_{yy}$ in the next time step. This would give two tridiagonal systems.

Some care must be taken with the boundary conditions, but if they are treated properly the LOD method is second order accurate and unconditionally stable for any time-step.

18.9 The Alternating Direction Implicit Method

We can modify the LOD method by discritizing in both spatial directions at once. That is,

$$U_{ij}^{*} = U_{ij}^{n} + \frac{k}{2}\left(D_y^2 U_{ij}^{n} + D_x^2 U_{ij}^{*}\right), \qquad U_{ij}^{n+1} = U_{ij}^{*} + \frac{k}{2}\left(D_x^2 U_{ij}^{*} + D_y^2 U_{ij}^{n+1}\right),$$

This again gives two tridiagonal systems. Each step is first oder accurate, however the errors almost cancel exactly and the two steps together are second order accurate. Since U^* is the approximate solution at time $t_{n+1/2}$ it is possible to evaluate the given boundary data to generate the boundary values for U^*. This maintains second order accuracy.

19 Advection Equations and Hyperbolic Systems

Hyperbolic PDEs arise in many physical problems, especially when wave motion is observed.

We consider the simplest case of a hyperbolic equation, the advection equation,

$$u_t + au_x = 0, \qquad\qquad u(x,0) = \eta(x)$$

This has exact solution $u(x,t) = \eta(x-at)$. Trying to solve this equation numerically will give some insight into the type of issues which arise from hyperbolic equations.

19.1 Method of Lines Discretization

Discretizing in space with periodic boundary conditions gives $U' = AU$ where,

$$A = -\frac{a}{2h}\begin{bmatrix} 0 & 1 & & & & -1 \\ -1 & 0 & 1 & & & \\ & -1 & 0 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -1 & 0 & 1 \\ 1 & & & & -1 & 0 \end{bmatrix}$$

This matrix is skew-Hermetian and therefore has pure imaginary eignevalues. Specifically,

$$\lambda_p = -\frac{ia}{hj}\sin(2\pi ph), \qquad\qquad p = 1, 2, \ldots, m+1$$

19.1.1 Forward Euler Time Discretization

Since λ_p are always imaginary, forward Euler will not be stable no matter how small we make k/h. However, the method is convergent. To see this note we have $B = I + kA$ and take $k = h^2$. Then,

$$|1 + k\lambda_p^2| \leq 1 + (ka/h)^2 \leq 1 + a^2k$$

Therefore $\|B^n\| = 1 + \mathcal{O}(k)$ so the method is Lax–Richtmyer stable (and equivalently convergent).

19.1.2 Leapfrog

A better time discretization is the midpoint method. This gives a 3-level explicit method which is second order in time and space. Recall the region of absolute stability of the midpoint method is the imaginary axis between $-i$ and i. Therefore the method is stable on the advection equation if $|k\lambda_p| \leq 1$ since λ_p is pure imaginary. In particular this require $|ak/h| < 1$.

However, $k\lambda_p$ will always be on the boundary of the stability region so the method is only marginally stable (the eignemodes do not grow or decay). We call such a difference method non-dissipative.

19.1.3 Lax–Friedrichs

Consider the Lax–Friedrichs method,

$$U_j^{n+1} = \frac{1}{2}\left(U_{j-1}^n + U_{j+1}^n\right) - \frac{ak}{2h}(U_{j+1}^n - U_{j-1}^n)$$

We can rewrite this as,

$$U_j^{n+1} = U_j^n - \frac{ak}{2h}(U_{j+1}^n - U_{j-1}^n) + \frac{1}{2}(U_{j-1}^n - 2U_j^n + U_{j+1}^n)$$

Rearranging terms we find,

$$\frac{U_j^{n+1}}{k} + a\left(\frac{U_{j+1}^n - U_{j-1}^n}{2h}\right) = \frac{h^2}{2k}\left(\frac{U_{j-1}^n - 2U_j^n + U_{j+1}^n}{h^2}\right)$$

This is consistent with the advection equation $u_t = au_x = 0$ as the right hand side vanishes as $k, h \to 0$ (assuming h/k is fixed).

However, it looks more like a discretization of the advection-diffusion equation $u_t + au_x = \epsilon u_{xx}$ where $\epsilon = h^2/2k$.

We can then view Lax–Friedrichs as the result of applying forward Euler to the system $U' = A_\epsilon U$ where,

$$A_\epsilon = -\frac{a}{2h}\begin{bmatrix} 0 & 1 & & & & -1 \\ -1 & 0 & 1 & & & \\ & -1 & 0 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -1 & 0 & 1 \\ 1 & & & & -1 & 0 \end{bmatrix} + \frac{\epsilon}{h^2}\begin{bmatrix} -2 & 1 & & & & -1 \\ 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2 & 1 \\ 1 & & & & 1 & -2 \end{bmatrix}$$

The eigenvalues of A_ϵ are,

$$\mu_p = -\frac{ia}{h}\sin(2\pi ph) - \frac{2\epsilon}{h^2}(1 - \cos(2\pi ph))$$

For the special case $\epsilon = h^2/2k$ used in Lax–Friedrichs we have all the eignevalues in the unit circle centered at -1 if $|ak/h| \leq 1$. In this case the forward Euler method is stable as a time-discretization, and hence the Lax–Friedrichs method is Lax–Richtmyer stable.

Note: I don't see this implication. What does it mean to be "stable".

19.2 The Lax–Wendroff Method

One way to achieve second order accuracy is to use a second order temporal discretization. This was done in the leapfrog scheme, however now this is a three-level method.

Instead we could use a Taylor series method and make the substitute $U'' = AU' = A^2U$ to obtain,

$$U^{n+1} = U^n + kAU^n + \frac{1}{2}k^2A^2U^n$$

This gives the method,

$$U_j^{n+1} = U_j^n - \frac{ak}{2h}(U_{j+1}^n - U_{j-1}^n) + \frac{a^2k^2}{8h^2}(U_{j-2}^n - 2U_j^n + U_{j+2}^n)$$

Now, noting that the last term is an approximation to $a^2k^2u_{xx}/2$ with step size $2h$, we can replace it with an approximation using step size h. This reduces the 5-point method to the 3-point Lax–Wendroff method,

$$U_j^{n+1} = U_j^n - \frac{ak}{2h}(U_{j+1}^n - U_{j-1}^n) + \frac{a^2k^2}{2h^2}(U_{j-1}^n - 2U_j^n + U_{j+1}^n)$$

This can be viewed as forward Euler applied to $U' = A_\epsilon U$ with $\epsilon = a^2k/2$. All the eigenvalues of this matrix lie within the stability region of forward Euler provided $|ak/h| < 1$.

19.3 Upwind Methods

Rather than using centered difference approximations for the u_x term, we could use one sided approximations. This may be useful in capturing some of the asymmetry many hyperbolic equations have.

Note: IDK if these are worth reading a lot about right now

19.4 Von Neumann Analysis

We can use Von Neumann analysis, replacing U_j^n by $g(\xi)^n e^{i\xi jh}$ and finding an expression for $g(\xi)$.

Note: UNDERSTAND WHY THIS CAN BE USED FOR MULTI-LEVEL METHOD (p. 213)

19.5 Characteristic Tracing and Interpolation

The characteristics for the advection equation are lines and $u(x_j, t_{n+1}) = u(x_j - ak, t_n)$. If we pick k, h such that $ak/h = 1$ then we can compute the exact solution numerically as $U_j^{n+1} = U_{j-1}^n$.

However, if $ak/h < 1$, then the point $x_j - ak$ is not a grid point. To find the value at U_j^{n+1} we could estimate the value of U on the characteristic by interpolating U_{j-1}^n and U_j^n.

Fitting a linear function gives,

$$p(x) = U_j^n + (x - x_j)\left(\frac{U_j^n - U_{j-1}^n}{h}\right)$$

When we evaluate this at $x_j - ak$ and use this value as U_j^{n+1} we obtain,

$$U_j^{n+1} = p(x_j - ak) = U_j^n - \frac{ak}{h}(U_j - U_{j-1}^n) = \left(1 - \frac{ak}{h}\right)U_j^n + \frac{ak}{h}U_{j-1}^n$$

For better accuracy we might try a higher order polynomial depending on more points. Using a quadratic polynomial interpolating U_{j-1}, U_j, and U_{j+1} gives the Lax–Wendroff method. If instead we interpolate between U_{j-2}, U_{j-1}, and U_j we obtain the Beam–Warming method.

19.6 The Courant–Friedrichs–Lewy Condition

> **Theorem.** (CFL condition)
>
> A numerical method can be convergent only if its numerical domain of dependence contains the true domain dependence of the PDE in the limit as $k, h \to 0$.

In the case of the advection equation, the domain of dependence is the characteristic $x_j - ak$.

19.7 Modified Equations

The standard tool for estimating the accuracy of finite difference methods has been the LTE. We can extend this approach slightly, and find a PDE which is better satisfied by our difference method. By doing this we may gain insight into how the difference method behaves.

Note: But do I need to know stuff about PDEs. Because without knowing what “disspersive” terms are and stuff this doesn’t seem like I can really do anything

19.8 Hyperbolic Systems

Given some diagonalizable $A = R\Lambda R^{-1}$, the system $u_t + Au_x = 0$, $u(x, 0) = \eta(x)$ is called hyperbolic.

Note: Some stuff about solving these??

19.9 Numerical Methods for Hyperbolic Systems

Most methods for the advection equation can be extended directly to a general hyperbolic system.