Numerik Summary HS17

Tablet notes

Week 1

1. Computing with matrizes and vectors (1)
Fixed point / floating point representation (2)
Machine numbers, absolute & relative error (3/4)
Axiom of roundoff analysis (5)
Notations, Libraries, Dense matrix storage formats (7-9)
Computational effort, asymptotic complexity (10/11)
Cost of basic operations (12)
some tricks to reduce complexity (13)

Week 2

Hidden summation, Kronecker produkt (1-3)
Cancellation, roots, diff. quotient (4)
Numerical stability, backward/mixed stable (9)

2. Direct methods for solving LSE (12)
Gauss elimination (13)
LU-decomposition (14)
Block elimination (16)

Week 3

Low rank modifications (1)
sherman-Woodbury formula (2)
Sparse linear Systems (3)
Storage, COO/triplet (4)
Direct solution of sparse LSE (9)

3 Direct methods for linear least squares (11)
examples (11)
least squares solution, existence (14)
kernel, range, normal equation, uniquness (15-16)
Generalized solution, uniquness (16)
Moore-Penrose pseudoinverse (19)

Week 4

Stability, extended normal equat. (1-2)
Orthogonal transformations (3)
QR-decomposition, Gram-Schmidt (6)
QR-decomp. theorem, uniquness (10)
Householder reflections (12)
SVD, singular values, rank/range/kernel (18)
SVD in Eigen (21)
Generalized solutions by SVD (22)
SVD Moore-Sherman pseudoinverse (23)

Week 5

SVD optimization: norm-constrained extrema (1)
best-low rank approx (2)
PCA (4)
Total least squares (8)
Constrained least squares, Lagrange (10)
via SVD (11)

4. Filtering (13)
properties of filter (14)
impulse response (15)
discrete convolution (18)

Week 6

discrete periodic convolution (1)
circulant matrix, disc conv → disc period conv (2)
DFT, EV/EW circulant matrix (4)
Inverse of Fourier matrix (6)
Diagonalization FM, DFT (7)
note on total least squares (8)
example on convolution (9)
disc conv via DFT, conv theorem, FFT (13-15)

Week 7

FFT (1)
Frequency filtering via DFT (3)
2D DFT, Filtering with 2D DFT (7)
2D convolution theorem (9)

Week 7 (continued)

- 5 Data interpolation in 1D (11)
- piecewise linear interpolation (13)
- hat functions, interpolant for p.l. interp (14)
- matrix representation (15)
- Global polynomial interpolation, horner scheme (17)
- Lagrange interpolation, existence & uniquness (18-19)

Week 8

- Lagrange, Complexity, matrix representation (1-2)
- Barycentric interpolation (2)
- Newton basis (3)
- Runge's phenomenon (5/6)
- Aitken-Neville scheme (6)
- Divided difference scheme + Horner scheme (8)

Week 9

- Splines, spline space (1)
- Cubic spline interpolation (2)

- 6. Approximation of functions in 1D (6)
- Global polynomials, Taylor approach (7)
- Bernstein approximation (8)

Week 10

- Jackson's theorem (2)
- Jackson's theorem on arbitrary intervals (3-4)
- Affine pullback, transformation under affine pullb. (4-5)
- Lagrangian approx scheme (7)
- algebraic, exponential convergence (8)
- representation of interpolation error (9)
- global L^∞ estimate (10)
- chebychev interpolation (11)
- chebychev polynomials (13)
- chebychev theorem, minimal polynomials, cheby. nodes (14-15)
- chebychev estimate (16)
- Lebesgue constant (18)
- implementation of chebychev interpolation (22)

Week 11

- implementation of Chebychev with Fourier (1)
- piece wise polynomial Lagrange interpolation (4)
- Continuous local Lagrange interpolants (5)
- error estimate (6)

- 7 Numerical quadrature (9)
- affine pullback (10)
- QF with Lagrange interpolation, Newton-Cotes (12)
- Gauss Quadrature (14)
- Sufficient order condition, maximal order (15-16)
- Legendre polynomials, Gauss-Legendre QF (20)

Week 12

- quadrature error for C^r-functions (2-3)
- increase of nodes (4)
- Composite Quadrature (6)

Week 13

- 8 Iterative methods (1)
- Bisection (2/3)
- Fixed point iterations (5)
- Order of convergence, linear convergence (6/7)
- quadratic conv: Newton iteration (8)
- Secant method for derivatives (11)
- Nonlinear systems of equations, equiv norms (13)
- local & global convergence (14)
- Fixed point iterations in $\mathbb{R}^n$, Banach, conditions (14/15)
- Newton's method in $\mathbb{R}^n$ (17)
- Failure examples (20)
- Damped Newton (21)

Week 14

- Unconstrained Optimization (1)
- Optimization with differentiable functions (3)
- Convex optimization (4)
- methods in 1D: Newton, Golden section search (5/6)
- Methods in ND: Gradient descent (8)
- Newton's method (11)
- BFGS method (quasi Newton) (12)

Code snippets

Lambda functions

```
auto func = [&globalVar] (int x) {return globalVar + x; };
```

Eval lamda functions

```
x.unaryExpr(func);
```

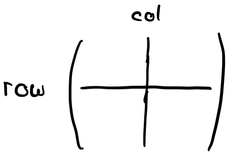
Iterate $2^0, 2^1, 2^2, \ldots$

```
for (int i=0; i≤N; ++i)  int n = 1 << i;
```

Matrix dimensions

A is $m \times n$

```
m = A.rows()
n = A.cols()
```



Solve $Ax = b$

we

```
FullPivLU<MatrixXd> B = A.fullPivLu();
PartialPivLU<MatrixXd> B = A.partialPivLu();      if A is invertible
ColPivHouseholder<MatrixXd> B = A.colPivHouseholderQr();
Full ..                                A.fullPiv...
TriangularView<MatrixXd, Upper> B = A.triangularView<Upper>();
                         Lower                           Lower

SparseLU<SparseMatrix<double>> B;                sparse
B.analyzePattern(A);   B.factorize(A);           matrices

VectorXd b = B.solve(b);
```

C++ vectors

```
std::vector<int> v;
v.push_back(42);
int m = v[0];
v.size();
std::sort(v.begin(), v.end());
v.pop_back();
(needs #include <algorithms>)
Map<VectorXd> myNewEigenVector(v.data(), v.size());
vector<double> v2(mat.data(), mat.data() + mat.rows() * mat.cols());
```

```
for (auto it = v.begin();
  it != v.end(); ++it) {
  cout << *it << endl;
}
for (auto t : v) {
  cout << t << endl;
}
```

Linspaced Vector

For $(0, 1, 2, \ldots, N)$: $(N+1,\ 0,\ N)$

```
VectorXd x = VectorXd::LinSpaced(len, from, to)
ArrayXd x = ArrayXd::LinSpaced(len, from, to)
```

Eps

(not :: denorm_min())

```
double eps = std::numeric_limits<double>::epsilon();
```

Pretty print

```
std::cout << std::scientific << std::setprecision(3) << std::setw(10)
  << out << std::endl;       #include <iomanip>
```

Triplets

```
         t.row()  t.col()  t.value()
Eigen::Triplet<double> t(3, 4, 42);
```

FFT

```
#include <unsupported/Eigen/FFT>
Eigen::FFT<double> fft;
VectorXcd v_fft = fft.fwd(v);
VectorXcd v_ffti = fft.inv(v_fft);
```

C++ constants and functions

```
pi: M_PI   (#include <cmath>)
sin(), exp(), cos(), abs()
```

Max coefficient + index

```
int index;      double max = v.maxCoeff(&index);
```

Sparse matrices

```
SparseMatrix<double> A(n,n);
std::vector<Triplet<double>> triplets;
triplets.reserve(n);
triplets.push_back(Triplet<double>(1, 2, 42))
A.setFromTriplets(triplets.begin(), triplets.end());
A.makeCompressed();
```

Code segments

```
backward_substitution (MatrixXd &A, VectorXd &b, VectorXd &x) {
  int n = A.cols();        x = VectorXd::zero(n);
  for (int i = n-1; i >= 0; --i) {
    x(i) = (b(i) - A.row(i) * x) / A(i,i);
  }
}        (for upper triangular matrices)
```

Householder QR-decomposition (economical)

```
  int m = A.rows();   int n = A.cols();
  HouseholderQR<MatrixXd> QR = A.householderQr();
  MatrixXd Q = QR.householderQ() * MatrixXd::Identity(m, n);
  MatrixXd R = MatrixXd::Identity(n, m) *
        QR.matrixQR().triangularView<Upper>();
```

CholeskyQR

```
  MatrixXd AtA = A.transpose() * A;
  LLT<MatrixXd> LLT = AtA.llt();
  MatrixXd R = LLT.matrixL().transpose();
  MatrixXd Q = R.transpose().triangularView<Lower>().
      solve(A.transpose()).transpose();
```

SVD / k-rank approximation

```
  JacobiSVD<MatrixXd> svd(A, ComputeThinU | ComputeThinV);
  U = svd.matrixU().leftCols(k);
  S = svd.singularValues().head(k).asDiagonal();
  V = svd.matrixV().leftCols(k);
  MatrixXd A_approx = U * S * V.transpose();
```

COO to CRS

```
  A.sort_byRow();          <------  std::sort(list.begin(), list.end(),
  int prevNotEmptyRow = -1;             [] (auto t1, auto t2) {
  auto &l = A.list;                       return t1.row() < t2.row();
                                        });
  for (int i=0; i<l.size(); i++) {
    while (prevNotEmptyRow < l[i].row()) {
      prevNotEmptyRow ++; row_ptr.push_back(i);
    }
    double currVal = l[i].value();
    while (i < l.size()-1 && l[i].col() = l[i+1].col() &&
           l[i].row() = l[i+1].row()) {
      currVal += l[i].value(); i++;
    }
    val.push_back(currVal); col_ind.push_back(l[i].col());
  }
  row_ptr.push_back(val.size());
```

Dense to CRS

```
  int nnz = 0; int prevNotEmptyRow = -1;
  for (int i = 0; i < A.rows(); i++) {
    for (int j = 0; j < A.cols(); j++) {
      if (A(i,j) =! 0) {
        while (prevNotEmptyRow < i) {
          row_ptr.push_back(nnz);
          prevNotEmptyRow ++;
        }
        nnz++;
        val.push_back(A(i,j));
        col_ind.push_back(j);
      }
    }
  }
  row_ptr.push_back(nnz);
```

COO efficient matrix multiplication

```
  if (&A1 == &A2) {
    MatrixCOO copyA2(A2.list); return mult_effic(A1, copyA2);
  }
  MatrixCOO result;
  vector<Triplet<double>> &l1 = A1.list;
  vector<Triplet<double>> &l2 = A2.list;
  vector<Triplet<double>> &lr = result.list;
  A1.sort_byCol();    A2.sort_byRow();
  vector<int> b1;   vector<int> b2;
  b1.push_back(0);  b2.push_back(0);
  for (int i = 0; i < l1.size()-1; i++)
    if (l1[i].col() != l1[i+1].col()) b1.push_back(i+1);
  b1.push_back(l1.size());
  for (int j = 0; j < l2.size()-1; j++)
    if (l2[j].row() != l2[j+1].row()) b2.push_back(j+1);
  b2.push_back(l2.size());
  int i = 0; int j = 0;
  while (i < b1.size()-1 && j < b2.size()-1) {
    if (l1[b1[i]].col() == l2[b2[j]].row()) {
      int c1, c2;
      for (c1 = b1[i]; c1 < b1[i+1]; c1++) {
        for (c2 = b2[j]; c2 < b2[j+1]; c2++) {
          Triplet<double> t(l1[c1].row(), l2[c2].col(),
             l1[c1].value() * l2[c2].value());
          lr.push_back(t);
        }
      } i++; j++;
    } else {
      if (l1[b1[i]].col() < l2[b2[j]].row()) i++;
      if (l1[b1[i]].col() > l2[b2[j]].row()) j++;
    }
  }
} return result;
```

Complexeties

Solve LSE

General $O(n^3)$

triangular $O(n^2)$

sparse system $O(nnz^{3/2})$ - $O(nnz^{5/2})$

low-rank modification $O(n^2)$

Sherman-Morrison factorization $O(nk^2)$

dot product $O(n)$

tensor product $O(mn)$

matrix product $O(mnk)$

Decompositions

SVD $O(n^3)$

thin SVD $O(mn^2)$

Householder QR $O(mn^2)$ for $A \in \mathbb{K}^{m,n}$, $m > n$

FFT $O(n \log n)$

Sparse shit

Insert element into CRS/CCS $O(nnz)$

Efficient initialization $O(n)$ if $nnz = O(n)$

Solve sparse LSE $O(nnz^{3/2})$ - $O(nnz^{5/2})$

Least squares

Normal equations $O(\underbrace{mn^2}_{C=A^TA} + \underbrace{nm}_{c=A^Tb} + \underbrace{n^3}_{Cx=c}) = O(n^3 + mn^2)$

Interpolation

Horner scheme $O(n)$

Monomial basis, evaluating N datasets $O(n^3 N)$

Lagrange, evaluating

L_i is $O(n)$, interpolant p is $O(n^2)$, N datasets $O(Nn^2)$

Barycentric interpolation,

computing $\lambda_0, \dots, \lambda_n$ $O(n^2)$, evaluating p $O(n)$

N datasets $O(n^2 + nN)$

Newton

building Vandermonde matrix $O(n^2)$, solving αs $O(n^2)$

N evaluations $O(n^2 N)$

Aitken-Neville, evaluating interpolant $O(n^2)$

Divided differences, evaluating interpolant $O(n)$,

adding new point $O(n)$

Other

Filter: causality is granted if $y_j = 0 \ \forall j < 0$

$\int_a^b f(t)\,dt \approx \sum_{j=1}^{n} \hat{\omega}_j f(\hat{c}_j)$ under affine pullback

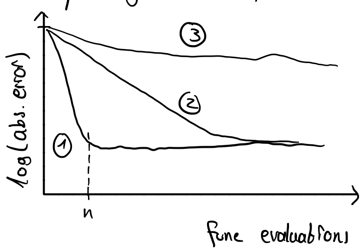
$\hat{c}_j = \frac{1}{2}(1 - c_j)a + \frac{1}{2}(1 + c_j)b$, $\quad \hat{\omega}_j = \frac{1}{2}(b-a)\omega_j$

Quadrature errorplot: linear trend in

lin-log plot $\rightarrow$ exponential convergence

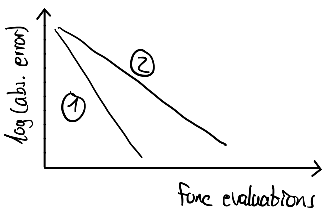
log-log plot $\rightarrow$ algebraic convergence

example global Gauss quadrature



1. $f \in \mathcal{P}_n$, QR exact up to $\sim n$ evaluations
2. $f \in C^\infty$, $f \notin \mathcal{P}_k \ \forall k \in \mathbb{N}$ exponential convergence
3. $f \in C^0$, $f \notin C^1$ algebraic convergence

other quadrature formulas



1. composite 2-point Gauss quadrat., because order ~ 4 (2x order of 2)
2. composite trapazoidal rule, order ~ 2

Estimates

Jackson's theorem for $f \in C^r([-1,1])$, $r \in \mathbb{N}$, for any polynomial degree $n \geq r$:

$$\inf_{p \in \mathcal{P}_n} \|f - p\|_{L^\infty([-1,1])} \leq \left(1 + \frac{\pi^2}{2}\right)^r \frac{(n-r)!}{n!} \|f^{(r)}\|_{L^\infty([-1,1])} = O(n^{-r})$$

$$\inf_{p \in \mathcal{P}_n} \|f - p\|_{L^\infty([a,b])} \leq \left(1 + \frac{\pi^2}{2}\right)^r \frac{(n-r)!}{n!} \left(\frac{b-a}{2}\right)^r \|f^{(r)}\|_{L^\infty([a,b])}$$

Lagrange for $f \in C^{n+1}(I)$ and equidistant nodeset $\mathcal{T} = \{t_0, \dots, t_n\} \subset I$

$$\|f - L_\mathcal{T} f\|_{L^\infty(I)} \leq \frac{\|f^{(n+1)}\|_{L^\infty(I)}}{(n+1)!} \max_{t \in I} |(t - t_0) \cdots (t - t_n)|$$

$$\|f - L_\mathcal{T} f\|_{L^2(I)} \leq \frac{2^{(n-1)/4} |I|^{n+1}}{\sqrt{n!(n+1)!}} \|f^{(n+1)}\|_{L^2(I)}$$

Chebychev for $f \in C^{n+1}(I)$, $I = [-1,1]$, $[a,b]$ and node set $\mathcal{T} = \{t_0, \dots, t_n\} \subset I$

$$\|f - I_\mathcal{T}(f)\|_{L^\infty([-1,1])} \leq \frac{2^{-n}}{(n+1)!} \|f^{(n+1)}\|_{L^\infty([-1,1])}$$

$$\|f - I_\mathcal{T}(f)\|_{L^\infty([a,b])} \leq \frac{2^{-2n-1}}{(n+1)!} |I|^{n+1} \|f^{(n+1)}\|_{L^\infty([a,b])}$$

Lebesgue constant

$$\|f - L_\mathcal{T} f\|_{L^\infty(I)} \leq (1 + \lambda_\mathcal{T}) \inf_{p \in \mathcal{P}_n} \|f - p\|_{L^\infty(I)} \quad \forall f \in C^0(I)$$

Chebychev + Lebesgue constant

$$\|f - L_\mathcal{T} f\|_{L^\infty([-1,1])} \leq \left(\frac{2}{\pi} \log(1+n) + 2\right)\left(1 + \frac{\pi^2}{2}\right)^r \frac{(n-r)!}{n!} \|f^{(r)}\|_{L^\infty([-1,1])}$$

Piecewise Lagrange

$$\|f - s\|_{L^\infty([x_0, x_m])} \leq h_\mathcal{M}^{n+1} \frac{1}{(n+1)!} \|f^{(n+1)}\|_{L^\infty([x_0, x_m])}$$

$$\|f - s\|_{L^2(I)} \leq h_\mathcal{M}^{n+1} \frac{2^{(n-1)/4}}{\sqrt{n!(n+1)!}} \|f^{(n+1)}\|_{L^2([x_0, x_m])}$$