Adaptive Quadrature

Basic idea

Split in subintervals: Do each with two schemes estimate difference/error if ok, done. Else split again.

Singular integrals

Either infinite interval or singularities of function.

Previous methods need functions to be polynomials like.

Note: $\int_a^b f(x) \ln(x) dx$ No problem if

infinite, or singularly in $W(x)$

Need to be 'polynomial-like.'

A few ideas:

1) Change variables: (Ex: Atkinson p305)

$$\int_0^{\infty} \frac{\ln(x)}{x^2} dx = 2 \int_0^{\infty} \frac{\ln(u^2)}{u^2} du$$

2) Integrate by parts
→ Subtract function with same type of singularity but with known integral

→ Taylor expand locally around singularity

\[ \lim_{t \to 0^+} \frac{1}{\sqrt{t}} \frac{1}{\sqrt{t}} = 5 \]

\[ \sin t = t - \frac{t^3}{3!} + \frac{t^5}{5!} - \ldots \]

→ IMT A method (Special case of change of variable) (Atkinson p306-307)

\[ I = \int_{a}^{b} f(x) \, dx \]

Tri: Teruguchi
Takasawa

→ Choose some \( C > 0 \)
→ Define \( y(t) = \exp \left( \frac{-c}{1-t^2} \right) \)

→ Define \( y(t) = a + (b-a) \cdot \frac{\int_{c}^{t} y(u) \, du}{\int_{c}^{1} y(u) \, du} \)

Again, all derivatives zero at \( t = \pm 1 \) by choice of \( C \).
Use \( y(t) \) for change of variable in original integral:

\[
I = \int_{a}^{b} f(x) dx = \int_{\alpha}^{\beta} f(y(t)) \frac{dy}{dt} dt
\]

Change
\( x = \psi(t) \)

New integrand has all derivatives zero at ends
\( \Rightarrow \) Trap. rule now excellent
(even if \( f(x) \) has algebraic singularity at ends).
Summation of infinite series

Slowly convergent: Two main cases:

Positive

Alternating

Euler–Maclaurin

\[
\sum_{n=N}^{\infty} f(n) = \int_{N}^{\infty} f(x) \, dx + \frac{1}{2} f(N) + \frac{1}{12} f'(N) + \frac{1}{720} f''(N) - \frac{1}{30240} f'''(N) + \cdots
\]

Expansion usually 'asymptotic'

Partial sums in Taylor expansion

Very often true value bound between successive partial sums. So one can reach very high accuracy before divergence.

Another 'perspective': Typical error as function of \( N \) and \# terms.
Ex: Evaluate numerically
\[ \gamma = \lim_{N \to \infty} \left( 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{N} - \ln(N) \right) \]

Very slow convergence as it stands:
\[ \text{Error} \approx \frac{1}{N} \implies \text{Error} \approx 10^{-10} \text{ requires } N \approx 10^{10} \]

Step 1: Rewrite as summing an infinite series:
Note: \( \ln(k-1) - \ln(k) = \ln \frac{k-1}{k} = \ln(1 - \frac{1}{k}) \)
\[ \gamma = 1 + \sum_{k=2}^{\infty} \left( \frac{1}{k} + \ln(1 - \frac{1}{k}) \right) \]
\[ = \frac{1}{1!} - \frac{1}{2!} - \frac{1}{3!} - \ldots \]
So sum converges

Step 2: Apply Euler-Maclaurin:
Choose \( n \) to start summing nine terms explicitly; i.e. \( N = 10 \).
\[ 1 + \sum_{k=2}^{5} \frac{1}{k} \approx 0.6317436767 \]
\[ \frac{\xi}{12} = -0.0617553591 \]
\[ \frac{\eta f^{(10)}}{12} = -0.0026802578 \]
\[ \frac{\zeta f^{(10)}}{12} = -0.0000925926 \]
\[ \frac{\gamma_{20} f^{(10)}}{12} = 0.000001993 \]
\[ \frac{\gamma_{240} f^{(5)}}{12} = -0.000000005 \]
\[ \sum = 0.5772156650 \]
I should be 0.58

5 terms gives 10^{-10} accuracy,
(rather than 10^{10} terms!)}
Subtract Known Series

Ex: \( S = \sum_{n=1}^{\infty} \frac{1}{1+n^2} \) (Exact: \( = \frac{\pi \coth \pi - 1}{2} \),

Say we happen to know
\( S_1 = \sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6} \).

Then consider
\( S - S_1 = \sum_{n=1}^{\infty} \left( \frac{1}{1+n^2} - \frac{1}{n^2} \right) = \sum_{n=1}^{\infty} \frac{1}{n^2(1+n^2)} \)

Converges so fast
that it can be
summed over

If one does not happen to know a similar series with known sum, one can always create one

Ex: \( S = \sum_{R=N}^{\infty} f(k) \) Can start at \( N = 1 \), or somewhere higher

Suppose \( F'(x) = f(x) \).

Then \( F(k+\frac{1}{2}) - F(k-\frac{1}{2}) \) is close to \( f(k) \).

Also:
\( S_2 = \sum_{R=N}^{\infty} \left\{ F(k+\frac{1}{2}) - F(k-\frac{1}{2}) \right\} = -F(k-N-\frac{1}{2}) \)

Consider
\( S - S_1 = \sum_{k=N}^{\infty} \left[ f(k) - \frac{1}{2} F(k+\frac{1}{2}) - F(k-\frac{1}{2}) \right] \).

Converges faster. Can here use higher order FD approx of \( F'(x) \).
Alternating Series

\[ \sum_{n=N}^{\infty} (-1)^n f(n) = \]

\[ = \sum_{n=N}^{\infty} \frac{1}{2^n} \left( f(n) - \frac{1}{2} f'(n) + \frac{1}{2^2} f''(n) - \frac{1}{2^4} f'''(n) + \cdots \right) \]

(based on expansion \( \frac{1}{1+e^x} = \frac{1}{2} - \frac{x}{4} + \frac{x^2}{98} - \frac{x^5}{480} + \cdots \))

Very much like Euler-Maclaurin, and integral absent.

Euler's Transform:

Let \( S = \sum_{n=N}^{\infty} \frac{z^n}{n!} f(n) = f(n) + zf(n) + z^2 f''(n) + \cdots \)

\[ = (1 + z + z^2 + \cdots) f(n) \]

\[ = \frac{f(n)}{1 - e^{-z}} = \frac{f(n)}{1 - x - x\Delta} \]

\[ = \frac{1}{1 - z} \cdot \frac{f(n)}{1 - \frac{x}{1 - z}} \]

If \( f \) changes slowly, disappears after few terms.
What about \((\frac{x}{1-x})^n\)?

Decays in shaded region.

Special case \(z = -1\),

\[
\sum_{n=N}^{\infty} (-1)^n f(n) = (-1)^N \sum_{k=0}^{\infty} \frac{(\frac{-1}{2})^k}{k!} \Delta^k f(n)
\]

Both factors decay fast and the sum is smooth.

Very powerful, and needs no derivatives.

---

**Numerical Differentiation**

At the end of Section 5.3.

Solution of linear systems of equations

Non-iterative (Direct) methods.
Consider for now only square systems:

\[ A x = b \]

1. **Diagonal:**

\[
\begin{pmatrix}
  a_{11} & 0 & \cdots & 0 \\
  0 & a_{22} & \cdots & 0 \\
  \vdots & \vdots & \ddots & \vdots \\
  0 & 0 & \cdots & a_{nn}
\end{pmatrix}
\begin{pmatrix}
  x_1 \\
  x_2 \\
  \vdots \\
  x_n
\end{pmatrix} =
\begin{pmatrix}
  b_1 \\
  b_2 \\
  \vdots \\
  b_n
\end{pmatrix}
\]

\[ x_i = b_i / a_{ii}, \text{ Op count: } n \]

2. **Lower Triangular:**

\[
\begin{pmatrix}
  a_{11} & a_{12} & \cdots & a_{1n} \\
  0 & a_{22} & \cdots & a_{2n} \\
  \vdots & \vdots & \ddots & \vdots \\
  0 & 0 & \cdots & a_{nn}
\end{pmatrix}
\begin{pmatrix}
  x_1 \\
  x_2 \\
  \vdots \\
  x_n
\end{pmatrix} =
\begin{pmatrix}
  b_1 \\
  b_2 \\
  \vdots \\
  b_n
\end{pmatrix}
\]

Back-substitute from top \( x_1, x_2, \ldots \to x_n \)

**Upper Triangular:**

\[
\begin{pmatrix}
  a_{11} & -a_{12} & \cdots & -a_{1n} \\
  0 & a_{22} & \cdots & -a_{2n} \\
  \vdots & \vdots & \ddots & \vdots \\
  0 & 0 & \cdots & a_{nn}
\end{pmatrix}
\begin{pmatrix}
  x_1 \\
  x_2 \\
  \vdots \\
  x_n
\end{pmatrix} =
\begin{pmatrix}
  b_1 \\
  b_2 \\
  \vdots \\
  b_n
\end{pmatrix}
\]

Equivalent: Back-substitute from bottom \( x_n, x_{n-1}, \ldots \to x_1 \)

**Op count:**

\[
\begin{array}{c|c|c}
\text{step} & + & \times/ \\
\hline
1 & 0 & 1 \\
2 & 1 & 2 \\
\vdots & \vdots & \vdots \\
 n & n-1 & n \end{array}
\]

\[
\sum: \frac{n(n-1)}{2} \quad \frac{n(n+1)}{2}
\]

Arithmetic progression; average of \( \sum \) of terms \( x \) is 1st and last term

Total: \( n^2 \) (Exactly)
3. **Full system**

Many variations available:

Regular Gaussian elimination:

Concept:

\[
\begin{bmatrix}
    a_{11} & a_{12} & \cdots & a_{1n} \\
    a_{21} & a_{22} & \cdots & a_{2n} \\
    \vdots & \vdots & \ddots & \vdots \\
    a_{n1} & a_{n2} & \cdots & a_{nn}
\end{bmatrix}
\begin{bmatrix}
    x_1 \\
    x_2 \\
    \vdots \\
    x_n
\end{bmatrix}
= 
\begin{bmatrix}
    b_1 \\
    b_2 \\
    \vdots \\
    b_n
\end{bmatrix}
\]

Assume at first no pivoting needed

Subtract mulit of top row from rows below

\[
\begin{bmatrix}
    1 & 1 & \cdots & 1 \\
    0 & 1 & \cdots & 1 \\
    \vdots & \vdots & \ddots & \vdots \\
    0 & 0 & \cdots & 1
\end{bmatrix}
\begin{bmatrix}
    x_1 \\
    x_2 \\
    \vdots \\
    x_n
\end{bmatrix}
= 
\begin{bmatrix}
    b_1 \\
    b_2 \\
    \vdots \\
    b_n
\end{bmatrix}
\]

Then mulit of row 2 from rows below

\[
\begin{bmatrix}
    1 & 1 & \cdots & 1 \\
    0 & 1 & \cdots & 1 \\
    \vdots & \vdots & \ddots & \vdots \\
    0 & 0 & \cdots & 1
\end{bmatrix}
\begin{bmatrix}
    x_1 \\
    x_2 \\
    \vdots \\
    x_n
\end{bmatrix}
= 
\begin{bmatrix}
    b_1 \\
    b_2 \\
    \vdots \\
    b_n
\end{bmatrix}
\]

Arrive at upper triangular system:

**Op count**

\[
\begin{array}{ccc}
    \text{Step} & \text{Operation} & \text{Count} \\
    1 & n(n-1) & \frac{n(n+1)(n-1)}{6} \\
    2 & \frac{n(n-1)(n-2)}{n(n-2)} & \frac{n(n+1)(n-2)}{6} \\
    \vdots & \vdots & \vdots \\
    n-1 & 21 & 31 \\
    \hline
    \sum & \frac{1}{3}n(n-1)(n+1) & \frac{1}{6}n(n-1)(n+1) \\
    \text{Total} & \frac{1}{6}n^2(4n^2+3n-7) & \frac{2n^3}{3} \\
    & & \text{(round off order)}
\end{array}
\]
Elementary row operations:

First step of elimination can be interpreted as multiplication from left with second row:

\[ L_1 = \begin{pmatrix} 1 & a_{21} & \cdots & a_{m1} \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix} \quad L_2 = \begin{pmatrix} 1 & a_{22} & \cdots & a_{m2} \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix} \]

\[ L_{n-1} L_{n-2} \cdots L_2 L_1 A = U \]

\[ A = L_1^{-1} L_2^{-1} \cdots L_{n-1}^{-1} U \]

Two key facts about this product:

1. \[ \begin{pmatrix} 1 & a_{12} & \cdots & a_{1m} \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}^{-1} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ -a_{12} & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ -a_{1m} & 0 & \cdots & 1 \end{pmatrix} \]
   
   First swap signs

2. A product

\[ \begin{pmatrix} 1 & a_{12} & \cdots & a_{1m} \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ -a_{12} & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ -a_{1m} & 0 & \cdots & 1 \end{pmatrix} \]

Just put the columns below diagonal side-by-side.
Gaussian Elimination has thus in fact produced a LU-factorization of $A$.

L-part is made up by all the different multiples of rows we used in the elimination.
VARIANTS OF GAUSSIAN ELIMINATION

1. Complete (Total) pivoting
2. Gauss-Jordan
3. Crout
4. Cholesky
5. Iterative improvement

(Iterative methods best covered in context of PDES; over/under-determined systems in context of QR & SVD decomposition, e.g. col. problem)

1. Complete (Total) Pivoting:

Dangers (can be seen from 'backward error analysis' - if we have time, will cover end of this semester)

Elements grow large during elimination.

\[ A_0 = \begin{bmatrix}
 1 & 0 & 0 & 0 \\
 -1 & 1 & 0 & 0 \\
 -1 & -1 & 1 & 0 \\
 -1 & -1 & -1 & 1 \\
\end{bmatrix} \Rightarrow A_1 = \begin{bmatrix}
 1 & 0 & 0 & 0 \\
 0 & 1 & -1 & 0 \\
 0 & 1 & -1 & 1 \\
 0 & 1 & -1 & -1 \\
\end{bmatrix} \]

Note: Regular pivoting never called for:

\[ A_4 \begin{bmatrix} 2 & 4 & 8 \end{bmatrix} \]

\[ A_4 \begin{bmatrix} 2 & 4 & 8 \end{bmatrix} \]

Same structure as \( A_0 \) but last column doubled.

With partial pivoting:

Growth to \( \tilde{f}(n) = 2^{n/2} \) nn matrix.
Total pivoting:

\[
\begin{bmatrix}
1 & 2 \\
3 & 4 \\
5 & 6
\end{bmatrix}
\]

1. Find largest entry in this column.
2. Move to pivot position.

Total pivoting:

Note: Exchange rows and columns. Re-order unknowns.

Best known with \( f(n) < n^{0.5} [2^{0.5}, 3^{0.3}, 4^{0.3}, \ldots, n^{0.5}]^{0.5} \)

No case known with \( f(n) > n \).

See completely safe.

Looks tempting to use, but:

1. Already partial pivoting growth extremely rare.
2. All testing needed in total pivoting brings cost up to \( A = QR \) solution, which is totally safe.
3. Iterative improvement corrects (and gives error estimate) at only \( O(n^2) \) additional cost.
4. Partial pivoting preserves zero-pattern much better.
Gauss-Jordan:

Eliminate both above & below main diagonal, to reach

\[
\begin{bmatrix}
1 & 0 & | & 1 \\
0 & 1 & | & 1
\end{bmatrix}
\]

Then becomes solution

+ No back substitution step needed
+ Cost increased (G.EL. \( \frac{2}{3} n^3 \) vs \( \frac{1}{3} n^3 \) op total)
+ Allows a pretty (but slower) way to calculate matrix inverse

\[
\begin{bmatrix}
A \\
\end{bmatrix}^{-1} \Rightarrow \begin{bmatrix}
E \\
\end{bmatrix} \Rightarrow \begin{bmatrix}
I \\
\end{bmatrix} \Rightarrow \begin{bmatrix}
A^{-1}
\end{bmatrix}
\]

Crout:

First w/o pivoting

\[
\begin{bmatrix}
A \\
\end{bmatrix} = \begin{bmatrix}
W & B \\
\end{bmatrix} \begin{bmatrix}
I & \tau \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
W & B \\
\end{bmatrix} \Rightarrow \begin{bmatrix}
1 & 0 \\
0 & 1
\end{bmatrix}
\]

→ Can fill in explicitly \( \begin{bmatrix} 1, 2 \Rightarrow, 3, 4 \Rightarrow \ldots \end{bmatrix} \)

→ Can overwrite \( L \& U \) on top of \( A \)
(convenient when solving by hand!)

→ Can pivot right across composite of \( L \& U \)
→ Operation count identical to G.EL.
Cholesky Decomposition

Requires a symmetric, positive definite matrix. Can then solve $A = LL^T$.

**Version 1:**

$$\begin{bmatrix} A \\ \end{bmatrix} = \begin{bmatrix} \text{block} \\ \end{bmatrix} \begin{bmatrix} \text{block} \\ \end{bmatrix}^T$$

- Requires $n$ square roots.
- Can show, pivoting never needed.
  
  Suppose $A$ scaled so $|\text{diagonal}| \leq 1$.
  
  Then $a_{11}^2 + a_{22}^2 + \ldots + a_{ii}^2 = a_{ii} \leq 1$

  $\Rightarrow$ No element in $L$ can ever become big.


**Version 2:**

Aim instead for $A = LDL^T$, with ones in diagonal of $L$.

No square roots needed, else same.

If $A$ is symmetric but not positive definite, we may get complex numbers in $L$.

However:
- Columns of $L$ can be purely real or purely imaginary.

**Bash:**
- May need pivoting, destroying symmetry.
- If not symmetric, can start by Cholesky, nothing lost.
ITERATIVE IMPROVEMENT (RESIDUAL CORRECTION)

Assume we have solved $Ax = b$ by LU factorization, obtained approx solution $\hat{x}$.

Form residual $A\hat{x} - b = \gamma$

\[ A \hat{x} - b = \gamma \]

\[ A \hat{x} - b = 0 \]

Key idea:

\[ A(\hat{x} - x) = \gamma \]

\[ x' \text{; correction needed to bring } \hat{x} \text{ to } x, \]

Another linear system to solve, but re-use LU-factorization $\implies O(n^2)$ op. only.

All elements of $\gamma$ and $x'$ small: floating point takes care of scaling.

Use $x'$ corrects for all errors that might have accumulated during LU back sub.\[ \Rightarrow \text{Fast solution in high precision arithmetic} \]

Web notes: Iterative improvement