Physics 75.502/487*
Computational Physics
Fall/Winter 1998/99

Dean Karlen
Department of Physics
Carleton University

- Part I: Introduction
- Part II: Numerical Methods
- Part III: Monte Carlo Techniques
- Part IV: Statistics for Physicists
- Part V: Special Topics

Part II: Numerical Methods

Topics:

- Linear Algebra
- Interpolation and Extrapolation
- Integration
- Root Finding
- Minimization or Maximization
- Differential Equations

References:

- *Numerical Recipes (in Fortran or C) The Art of Scientific Computing, Second Edition* W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, Cambridge University Press, 1992.
- *Numerical Methods for Physics*, A.L. Garcia, Prentice Hall, 1994.

Solving Linear Algebraic Equations

General Problem

There are N unknowns, x_j and M equations,

$$\sum_{j=1}^{N} a_{ij} x_j = b_i \quad i = 1, ..., M \quad .$$

If $N = M$ there can be a solution, unless there is row or column degeneracy (ie. singular).

Numerical solutions to this problem can have additional problems:

- equations are so close to being singular, that round off error renders them so and hence the algorithm fails
- equations are close to being singular and N is large that roundoff errors accumulate and swamp the result

Limits on N, if not close to singular:

- 32 bit $\rightarrow$ N up to around 50
- 64 bit $\rightarrow$ N up to few hundred (CPU limited)

If coefficients are sparse, the $N > 1000$ or more can be handled by special methods.

Common Mistake

A common mistake when manipulating matrices, is that incorrect logical and physical dimensions are passed to a function:

In Fortran for example, one might set up a general purpose matrix as follows:

```
PARAMETER (NP=4,MP=6)
REAL A(NP,MP)
```

If a particular problem deals with 3 equations with 4 unknowns, the logical size of the matrix is (3,4) whereas the physical size is (NP, MP). In order for a function to interpret the matrix properly, it needs to know both the logical and physical dimensions. Fortran stores the elements of the matrix as follows:

Physical Memory

1	5	9	13	17	21
2	6	10	14	18	22
3	7	11	15	19	23
4	8	12	16	20	24

$\rightarrow$

Logical Array

a_{11}	a_{12}	a_{13}	a_{14}	–	–
a_{21}	a_{22}	a_{23}	a_{24}	–	–
a_{31}	a_{32}	a_{33}	a_{34}	–	–
–	–	–	–	–	–

Typical Linear Algebra Problems

- $\mathbf{Ax} = \mathbf{b}$ where $\mathbf{A}$ is a known $N \times N$ matrix, and $\mathbf{b}$ is a known vector. The problem is to find the solution vector, $\mathbf{x}$.
- Given $\mathbf{A}$, find $\mathbf{A}^{-1}$ or find $\det(\mathbf{A})$.
- If $\mathbf{A}$ is an $N \times M$ matrix with $M < N$, find the solution space.
- If $M > N$ find the "best" result (least squares).

Basic Methods

1. Gauss-Jordan elimination
2. Gaussian elimination with backsubstitution
3. LU decomposition

Gauss-Jordan Elimination

+ an efficient method for inverting **A**
− 3 times slower than other methods not producing $\mathbf{A}^{-1}$
− not recommended as a general purpose method

Method without pivoting

Perform operations that transform **A** into the identity matrix:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix}$$

$$\begin{pmatrix} 1 & \frac{a_{12}}{a_{11}} & \frac{a_{13}}{a_{11}} & \frac{a_{14}}{a_{11}} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} \frac{b_1}{a_{11}} \\ b_2 \\ b_3 \\ b_4 \end{pmatrix}$$

$$
\begin{pmatrix}
1 & \frac{a_{12}}{a_{11}} & \frac{a_{13}}{a_{11}} & \frac{a_{14}}{a_{11}} \\
0 & a_{22} - a_{12}\frac{a_{21}}{a_{11}} & a_{23} - a_{13}\frac{a_{21}}{a_{11}} & a_{24} - a_{14}\frac{a_{21}}{a_{11}} \\
0 & a_{32} - a_{12}\frac{a_{31}}{a_{11}} & a_{33} - a_{13}\frac{a_{31}}{a_{11}} & a_{34} - a_{14}\frac{a_{31}}{a_{11}} \\
0 & a_{42} - a_{12}\frac{a_{41}}{a_{11}} & a_{43} - a_{13}\frac{a_{41}}{a_{11}} & a_{44} - a_{14}\frac{a_{41}}{a_{11}}
\end{pmatrix}
\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}
=
\begin{pmatrix}
\frac{b_1}{a_{11}} \\
b_2 - b_1\frac{a_{21}}{a_{11}} \\
b_3 - b_1\frac{a_{31}}{a_{11}} \\
b_4 - b_1\frac{a_{41}}{a_{11}}
\end{pmatrix}
$$

$$
\begin{pmatrix}
1 & \frac{a_{12}}{a_{11}} & \frac{a_{13}}{a_{11}} & \frac{a_{14}}{a_{11}} \\
0 & 1 & a'_{23} & a'_{24} \\
0 & a_{32} - a_{12}\frac{a_{31}}{a_{11}} & a_{33} - a_{13}\frac{a_{31}}{a_{11}} & a_{34} - a_{14}\frac{a_{31}}{a_{11}} \\
0 & a_{42} - a_{12}\frac{a_{41}}{a_{11}} & a_{43} - a_{13}\frac{a_{41}}{a_{11}} & a_{44} - a_{14}\frac{a_{41}}{a_{11}}
\end{pmatrix}
\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}
=
\begin{pmatrix}
\frac{b_1}{a_{11}} \\
b'_2 \\
b_3 - b_1\frac{a_{31}}{a_{11}} \\
b_4 - b_1\frac{a_{41}}{a_{11}}
\end{pmatrix}
$$

$$
\begin{pmatrix}
1 & 0 & a'_{13} & a'_{14} \\
0 & 1 & a'_{23} & a'_{24} \\
0 & 0 & a'_{33} & a'_{34} \\
0 & 0 & a'_{43} & a'_{44}
\end{pmatrix}
\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}
=
\begin{pmatrix}
\frac{b_1}{a_{11}} \\
b'_2 \\
b'_3 \\
b'_4
\end{pmatrix}
$$

After continuing this process, one gets the following:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \end{pmatrix}$$

And hence the solutions are, $x_i = b'_i$.

Note that the same method could have produced $\mathbf{A}^{-1}$. That is, replace $\mathbf{x}$ by $\mathbf{Y}$ and $\mathbf{b}$ by the identity matrix:

$$\mathbf{AY} = \mathbf{I}$$

Then after performing the same operations as above that transforms $\mathbf{A}^{-1}$into the identity,

$$\mathbf{IY} = \mathbf{I}' = \mathbf{A}^{-1}$$

What if diagonal element is zero?

If $a_{11} = 0$ or another derived diagonal element (such as $a_{22} - a_{12}\frac{a_{21}}{a_{11}}$ in the example above) is zero, then algorithm fails.

If instead of being exactly 0, one of these terms is very small, then the remaining equations can become identical, in the presence of round off error.

Solution: Pivoting

By interchanging rows (partial pivoting) or both rows and columns (full pivoting), this problem can be avoided.
To maintain the identity matrix being formed, interchange rows below and columns to the right.

If rows are interchanged $\rightarrow$ one must also interchange corresponding rows in **b**.
If columns are interchanged $\rightarrow$ one must also interchange corresponding rows in **x**. These rows will have to be restored to the original order at the end.

How to decide which rows (or columns) to substitute?
Choosing the row with the largest value works quite well.

Implementation

To minimize storage requirements:

- Use `b` to built up solution. There is no need to have a separate array.
- Similarly the inverse can be built up in the input matrix.

The disadvantage with this is that the input matrix and RHS vector are destroyed by the operation.

Numerical Recipes:

```
SUBROUTINE gaussj(a,n,np,b,m,mp)
```

where

`a` is an `n` $\times$ `n` matrix in array of physical dimension `np` $\times$ `np`

`b` is an `n` $\times$ `m` matrix in array of physical dimension `np` $\times$ `mp`

Note that `a` is replaced by its inverse, and `b` by its solutions.

Gaussian Elimination with Backsubstitution

This method reduces the number of operations compared with Gauss-Jordan method (including inverse calculation) by about 3 (if inverse is not required).

Method without pivoting

Perform operations that transform **A** into an upper triangular matrix:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix}$$

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a'_{22} & a'_{23} & a'_{24} \\ 0 & a'_{32} & a'_{33} & a'_{34} \\ 0 & a'_{42} & a'_{43} & a'_{44} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} b_1 \\ b'_2 \\ b'_3 \\ b'_4 \end{pmatrix}$$

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a'_{22} & a'_{23} & a'_{24} \\ 0 & 0 & a'_{33} & a'_{34} \\ 0 & 0 & 0 & a'_{44} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} b_1 \\ b'_2 \\ b'_3 \\ b'_4 \end{pmatrix}$$

Pivoting is important for this method also.

To solve for x_i, backsubstitute:

$$\begin{aligned} x_4 &= \frac{b'_4}{a'_{44}} \\ x_3 &= \frac{1}{a'_{33}}[b'_3 - x_4 a'_{34}] \end{aligned}$$

Note that both this method and Gauss-Jordan method require all RHS to be known in advance.

LU decomposition

Any matrix **A** can be decomposed into into the product of a lower triangular matrix (**L**) and an upper triangular matrix (**U**).

$$\begin{aligned} \mathbf{Ax} &= \mathbf{b} \\ (\mathbf{LU})\mathbf{x} &= \mathbf{b} \\ \mathbf{L}(\mathbf{Ux}) &= \mathbf{b} \end{aligned}$$

So solve, $\mathbf{Ly} = \mathbf{b}$ for $\mathbf{y}$ and then solve, $\mathbf{Ux} = \mathbf{y}$ for $\mathbf{x}$. These are easily solved for. Once the **LU** decomposition is found, one can solve for as many RHS vectors as needed.

How to find L and U?

Crout's algorithm:
Note that

$$\sum_{k=1}^{N} \ell_{ik} u_{kj} = a_{ij}$$

represents N^2 equations where there are $N^2 + N$ unknowns. Arbitrarily set the terms, $\ell_{ii} = 1$, to define a unique solution.

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ \ell_{21} & 1 & 0 & 0 \\ \ell_{31} & \ell_{32} & 1 & 0 \\ \ell_{41} & \ell_{42} & \ell_{43} & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ 0 & u_{22} & u_{23} & u_{24} \\ 0 & 0 & u_{33} & u_{34} \\ 0 & 0 & 0 & u_{44} \end{pmatrix}$$

$$= \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}$$

The terms in **L** and **U** can be determined as follows:

$$\begin{aligned} u_{11} &= a_{11} \\ u_{12} &= a_{12} \\ \ell_{21} &= \frac{a_{21}}{u_{11}}, \quad \ell_{31} = \frac{a_{31}}{u_{11}}, \quad \ell_{41} = \frac{a_{41}}{u_{11}} \\ u_{22} &= a_{22} - \ell_{21} u_{12} \\ \ell_{32} &= \frac{1}{u_{22}}(a_{32} - \ell_{31} u_{12}), \quad \ell_{42} = \frac{1}{u_{22}}(a_{42} - \ell_{41} u_{12}) \\ u_{13} &= a_{13} \\ u_{23} &= a_{23} - \ell_{21} u_{13} \\ u_{33} &= a_{33} - \ell_{31} u_{13} - \ell_{32} u_{23} \end{aligned}$$

etc.

- The order above must be followed so the terms ℓ_{ij} and u_{ij} are available when necessary.
- Each a_{ij} appears once and only once, when the corresponding ℓ_{ij} or u_{ij} terms are calculated. In order to save memory, these terms can be stored in the corresponding a_{ij} locations.
- Pivoting is essential here too, but only the interchange of rows is efficient.

Numerical Recipes:

```
SUBROUTINE ludcmp(a,n,np,indx,d)
```

where

`a`	is an `n` $\times$ `n` matrix in array of physical dimension `np` $\times$ `np`
`indx,d`	keep track of rows permuted by pivoting

Note that `a` is replaced by

$$\begin{pmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ \ell_{21} & u_{22} & u_{23} & u_{24} \\ \ell_{31} & \ell_{32} & u_{33} & u_{34} \\ \ell_{41} & \ell_{42} & \ell_{43} & u_{44} \end{pmatrix}$$

Once the LU decomposition is found, find solutions using backsubstitution:

```
SUBROUTINE lubksb(a,n,np,indx,b)
```

where

`a, indx` are the results from the call to `ludcmp`

`b` is RHS on input, is solution on output

Note that `a` and `indx` are not modified by this routine so `lubksb` can be called repeatedly.

To find inverse, solve

$$\mathbf{Ax} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix},$$

to find the columns of $\mathbf{A}^{-1}$.

The determinant is easily found,

$$det(\mathbf{A}) = \prod_{\mathbf{i=1}}^{\mathbf{N}} \mathbf{u_{ii}}$$

Iterative Improvement of a Solution

The algorithms presented above sometimes yield solutions with precision less than the machine limit (depending on how close equations are to being singular). Improved precision can be made by an iterative approach.

Suppose $\mathbf{x}$ is the exact solution to

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

and the resulting numerical solution is instead $\mathbf{x} + \delta\mathbf{x}$. Then,

$$\mathbf{A}(\mathbf{x} + \delta\mathbf{x}) = \mathbf{b} + \delta\mathbf{b}$$

so,

$$\mathbf{A}(\delta\mathbf{x}) = \mathbf{A}(\mathbf{x} + \delta\mathbf{x}) - \mathbf{b}$$

and so solve for $\delta\mathbf{x}$, subtract it from the previous solution to get an improved solution.

Numerical Recipes:

```
SUBROUTINE mprove
```

can be called repeatedly to improve solution (although once is usually enough)

Singular Value Decomposition

If $\mathbf{A}$ is an $N \times N$ matrix, it can be decomposed

$$\mathbf{A} = \mathbf{U}\mathbf{W}\mathbf{V}^{\mathrm{T}}$$

where $\mathbf{U}$ and $\mathbf{V}$ are orthogonal ($\mathbf{U}^{-1} = \mathbf{U}^{\mathrm{T}}$), and $\mathbf{W}$ is diagonal.

The inverse of A is easily found to be

$$\mathbf{A}^{-1} = \mathbf{V}\,\mathrm{diag}(\frac{1}{\mathrm{w_j}})\,\mathbf{U}^{\mathrm{T}}$$

- If one or more w_j is zero, then $\mathbf{A}$ is singular.
- If the ratio $\min(w_j)/\max(w_j)$ is less than the machine precision then the matrix is ill conditioned. In this case it is often better to set such small w_j to 0.

Note that if $\mathbf{A}$ is singular:

$\mathbf{Ax} = \mathbf{0}$ for some subspace of $\mathbf{x}$. The space is called the nullspace its dimension is called the nullity.

$\mathbf{Ax} = \mathbf{b}$ the space of all possible $\mathbf{b}$ is called the range and its dimension is called the rank.

- nullity + rank = N
- nullity = number of zero w_i's
- The columns of U with non-zero w_i's span the range.
- The columns of V with zero w_i's span the nullspace.

If $\mathbf{A}$ is singular or ill-conditioned, a space of vectors may satisfy $\mathbf{Ax} = \mathbf{b}$. If the solution with the smallest $|\mathbf{x}|$ is desired, this can be found by replacing $\frac{1}{w_j}$ by zero for all $w_j = 0$!

Numerical Recipes:

```
SUBROUTINE svdcmp(a,m,n,mp,np,w,v)
```

Sparse Linear Systems

Systems with many zero matrix elements can be solved with special algorithms that save time and/or space (by not using memory to hold all those zeros).

Tridiagonal systems, for example

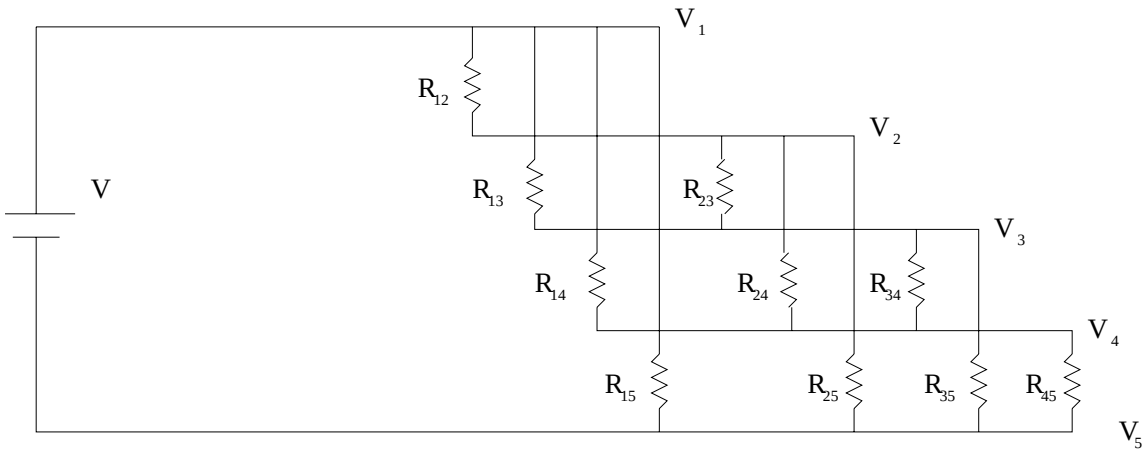
$$\begin{pmatrix} a_{11} & a_{12} & 0 & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 & 0 \\ 0 & a_{32} & a_{33} & a_{34} & 0 \\ 0 & 0 & a_{43} & a_{44} & a_{45} \\ 0 & 0 & 0 & a_{54} & a_{55} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{pmatrix}$$

can be LU decomposed much quicker than Crout's method. See `SUBROUTINE tridiag`.

Other forms of sparse matrices have special methods. See Numerical Recipes for details.

Exercise 1

Any resistor divider network can be put in the form:



This network has 5 voltage points, V_i. To calculate the total current, apply Kirchoff laws:

$$I = \sum_{i=1}^{5}(V_1 - V_i)\frac{1}{R_{1i}} \qquad (1)$$

$$0 = \sum_{i=1}^{5}(V_2 - V_i)\frac{1}{R_{2i}}$$

$$0 = \sum_{i=1}^{5}(V_3 - V_i)\frac{1}{R_{3i}}$$

$$0 = \sum_{i=1}^{5}(V_4 - V_i)\frac{1}{R_{4i}}$$

where $\frac{1}{R_{ii}}=0$.

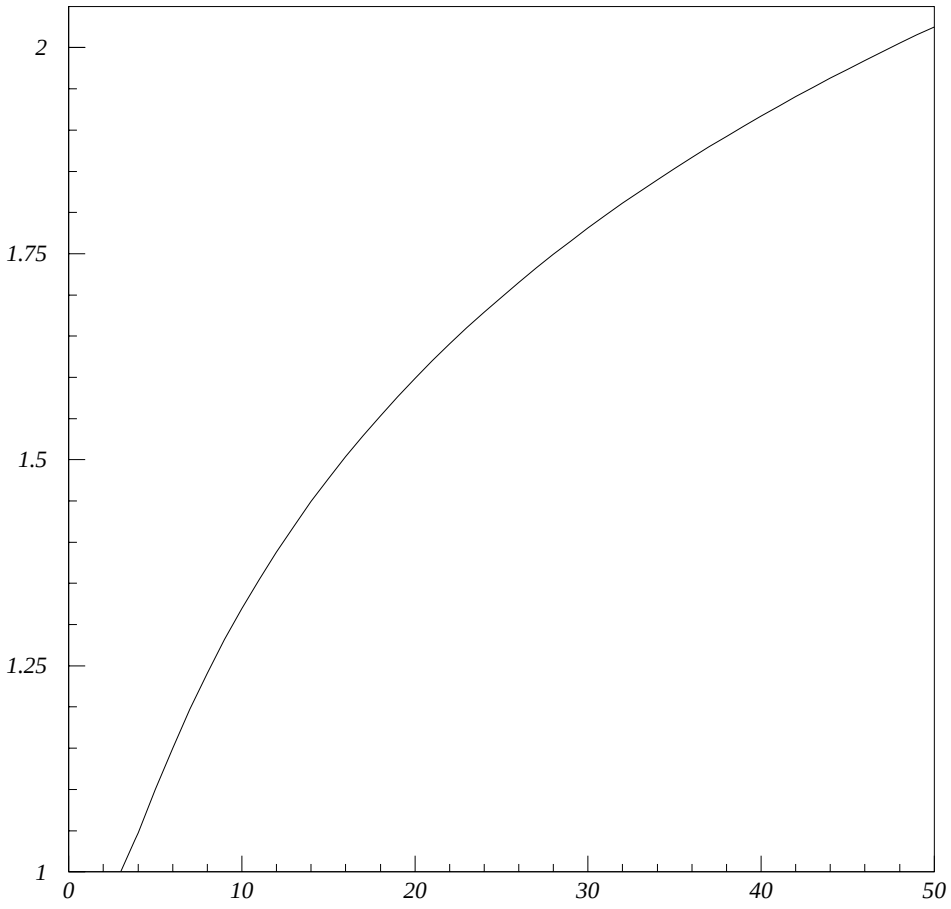
In order to solve for the four unknowns (I, V_2, V_3, and V_4), one can rearrange the last three equations, and identify $V_1 = V$, and $V_5 = 0$.

$$\begin{array}{ccccccc}
-\sum_{i=1}^{5} \frac{1}{R_{2i}} V_2 & + & \frac{1}{R_{23}} V_3 & + & \frac{1}{R_{24}} V_4 & = & -\frac{1}{R_{12}} V \\
\frac{1}{R_{23}} V_2 & - & \sum_{i=1}^{5} \frac{1}{R_{3i}} V_3 & + & \frac{1}{R_{34}} V_4 & = & -\frac{1}{R_{13}} V \\
\frac{1}{R_{24}} V_2 & + & \frac{1}{R_{34}} V_3 & - & \sum_{i=1}^{5} \frac{1}{R_{4i}} V_4 & = & -\frac{1}{R_{14}} V
\end{array}$$

The voltages V_2, V_3, and V_4 can then be found by numerical methods, and substituted into the equation 1, in order to find the total current, I.

Write a program that solves this problem for any number of voltage points between 3 and 50. Consider the special case where V=1 Volt, all resistors are present, and the value of each resistor is given by $R_{ij} = |i - j|\,\Omega$. Plot the current drawn as a function of number of voltage points.

Solution:



Interpolation and Extrapolation

General Problem

Given a table of values, $y(x_i)$, $i = 1, ..., N$, estimate $y(x)$ for arbitrary x.

- graphically: drawing a smooth curve through the points
- different from fitting: tabulated values have no errors. The curve should go through all points.
- most commonly used curves are polynomials

Methods

1) Determine interpolating function using a set of points $x_i, y(x_i)$, then evaluate the function at the point x.
$\rightarrow$ not recommended...

- inefficient
- roundoff error
- no error estimate

2) Start from $y(x_i)$ for x_i close to x, and add corrections from x_j further away. Successive corrections should decrease and the size of the last correction can be used as an estimate of the error.

- If interpolation method only uses a set of points x_i near x, the coefficients of the interpolating function change from one range to another. As a result the interpolating function can be continuous but will not have continuous first derivatives.
- If continuous derivatives are important, spline functions (such as the cubic spline) can be used. These tend to be more stable than polynomial functions (less prone to wild oscillations).
- The number of tabulated points used (minus one) is the order of the interpolation. Increasing the order does **not** lead to increased precision. Recommended to not use order > 5.
- Extrapolation is prone to error. Definitely not to be trusted beyond typical spacing of x_i from the last x_i.

Polynomial Interpolation

Through any set of N points there is a unique polynomial of order $N-1$ through those points. It is defined by the Lagrange formula:

$$P_{N-1}(x) = \sum_{i=1}^{N} \left(\prod_{j=1, j\neq i}^{N} \frac{x - x_j}{x_i - x_j} \right) y_i$$

A better method to specify the polynomial is to start with the order 0 polynomial $P_i = y(x_i)$. Add corrections from additional points x_j one at a time, each time increasing the order of the polynomial. Each term can be determined by a recurrence relation (*Neville's algorithm*: see text).

Numerical Recipes:

```
SUBROUTINE polint(xa,ya,n,x,y,dy)
```

returns:

- `y` is the estimate of $y(x)$ given `n` tabulated entries in the arrays `xa(n)`, `ya(n)`
- `dy` is the last correction applied, and can be used as an error estimate

Rational Function Interpolation

Some functions are better approximated by ratios of polynomials:

$$R(x) = \frac{P_\mu(x)}{Q_\nu(x)} = \frac{p_0 + p_1 x + ... + p_\mu x^\mu}{q_0 + q_1 x + ... + q_\nu x^\nu}$$

- this form can model poles (zeros of denominators)
- $R(x)$ goes through N points, where $N = \mu + \nu + 1$
- Similar recurrence relation has been developed to determine p_μ and q_ν for the case where:

  $\mu = \nu = (N-1)/2$ for N odd; or

  $\mu + 1 = \nu = N/2$ for N even

Numerical Recipes:

```
SUBROUTINE ratint(xa,ya,n,x,y,dy)
```

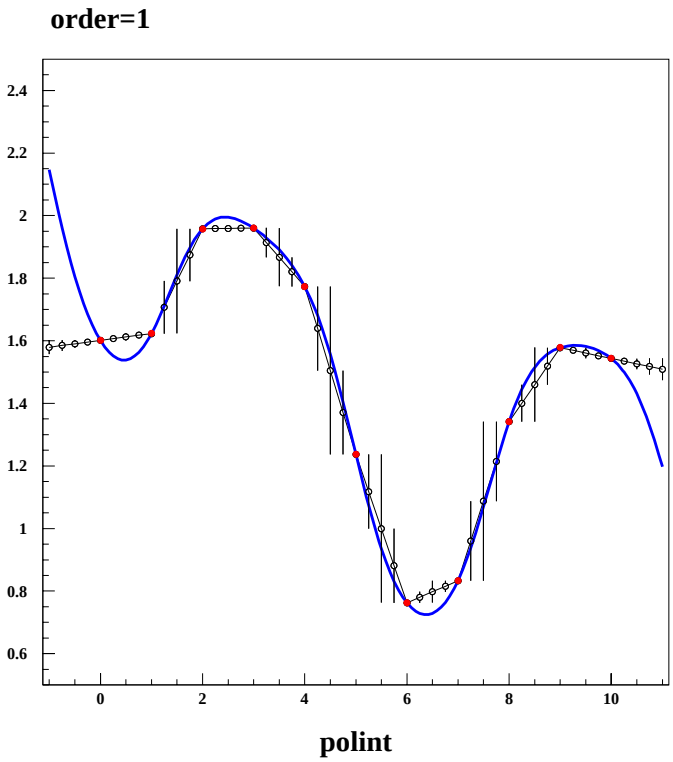
Example of Polynomial Interpolation

Thick blue line is given by

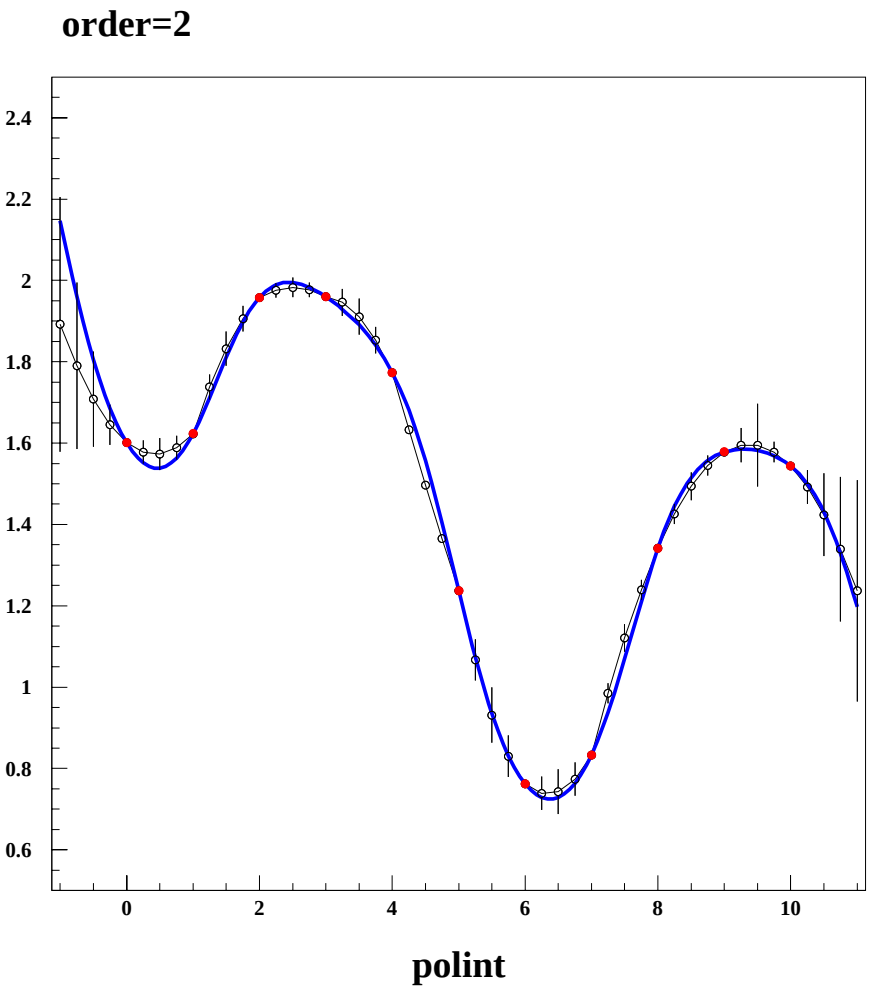
$$y(x) = \mathrm{erfc}(\cos(\pi + \log(x+4))) + \sqrt{\mathrm{erfc}(\cos x)}$$

Solid red points are tabulated values

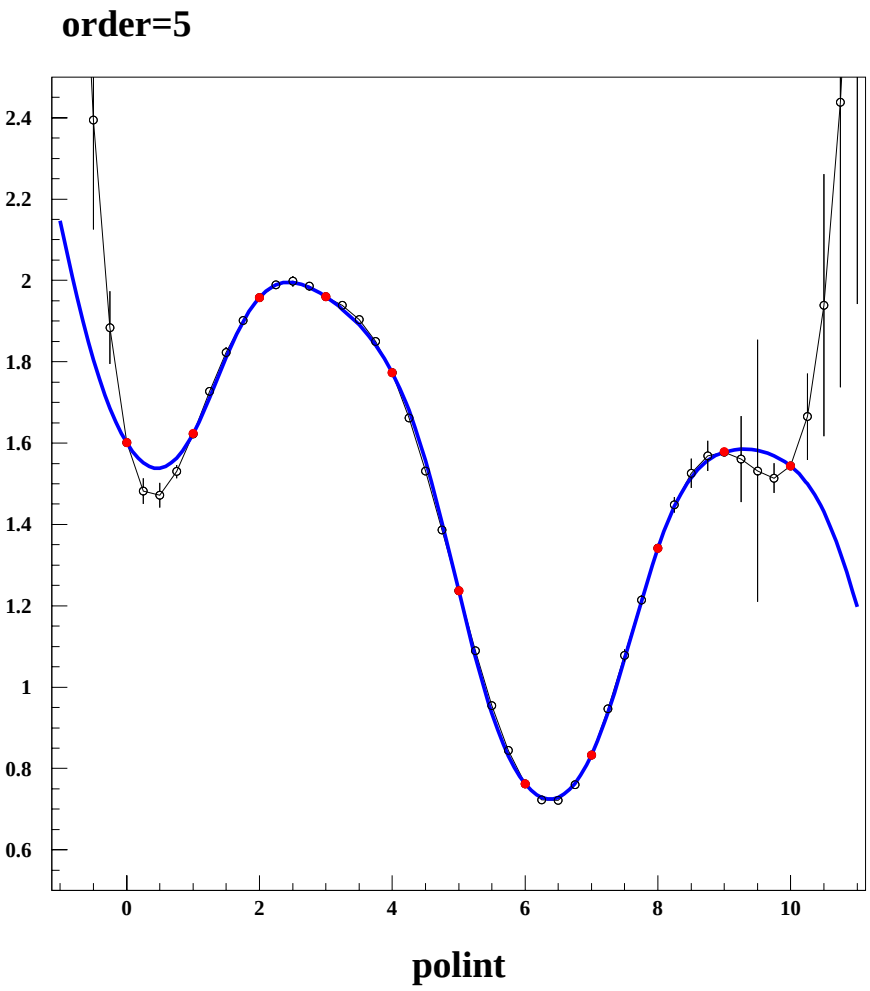
Black circles and errors show the interpolation:



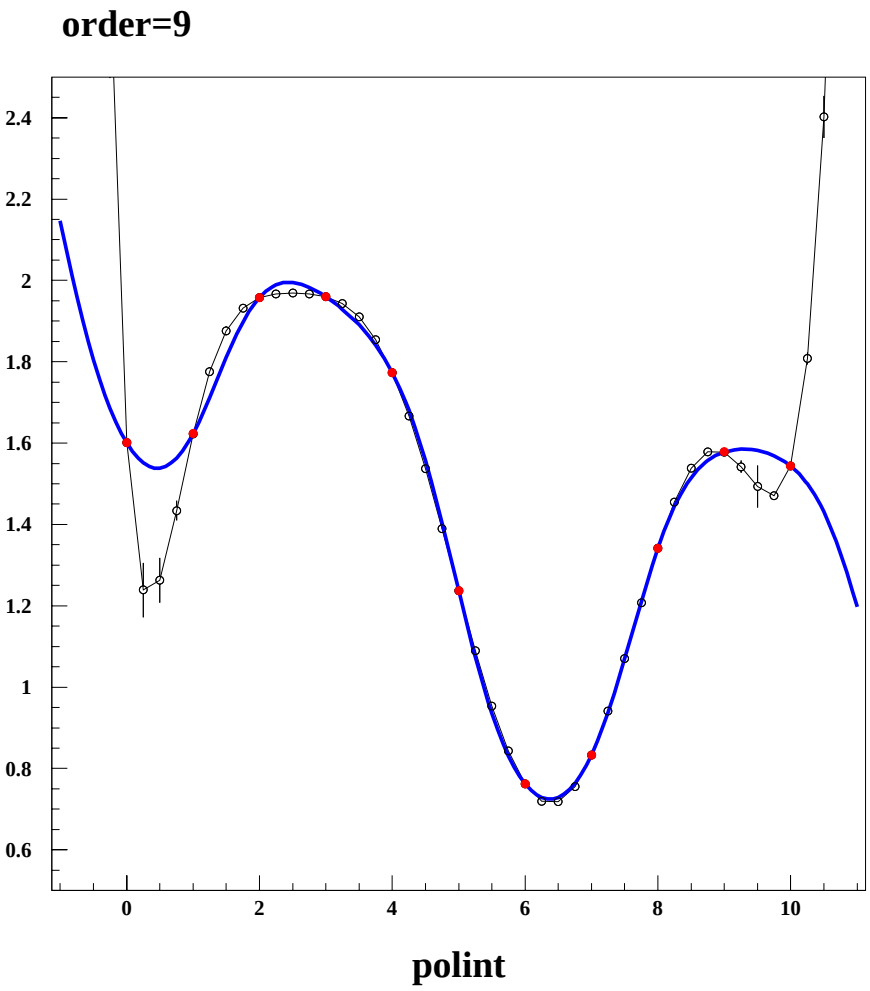
Second order improves approximation:



Fifth order gives bad extrapolations:

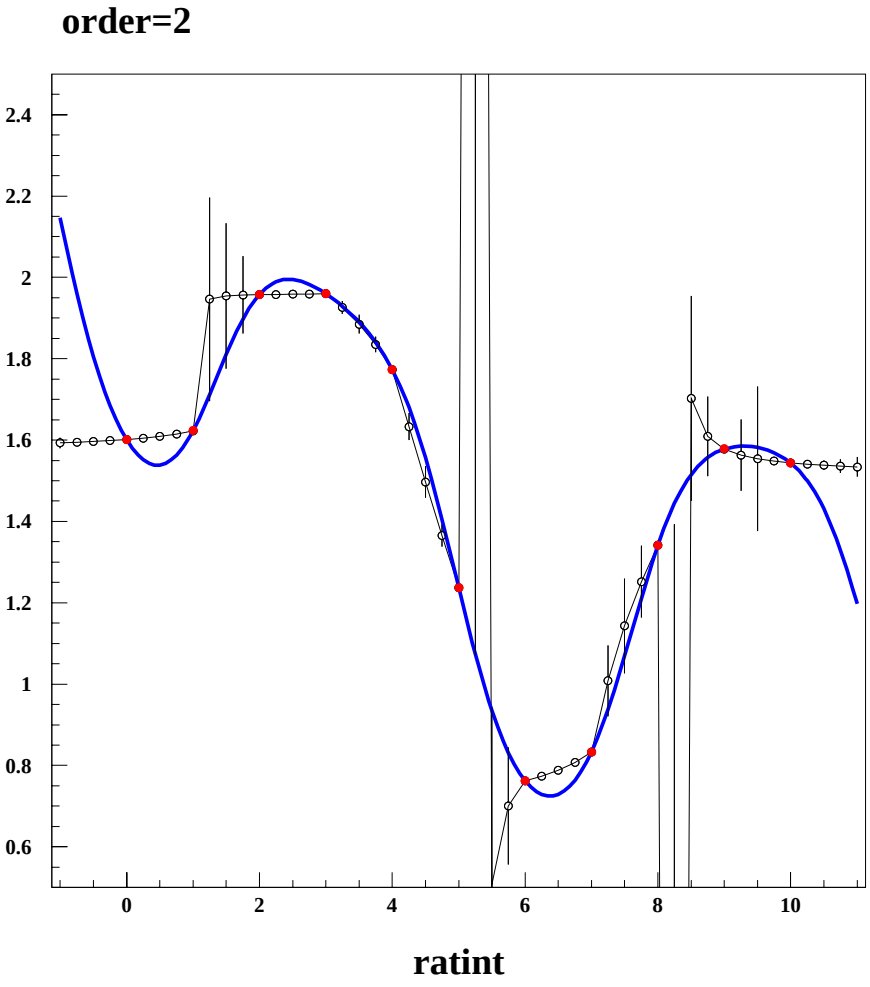


Ninth order has still worse accuracy



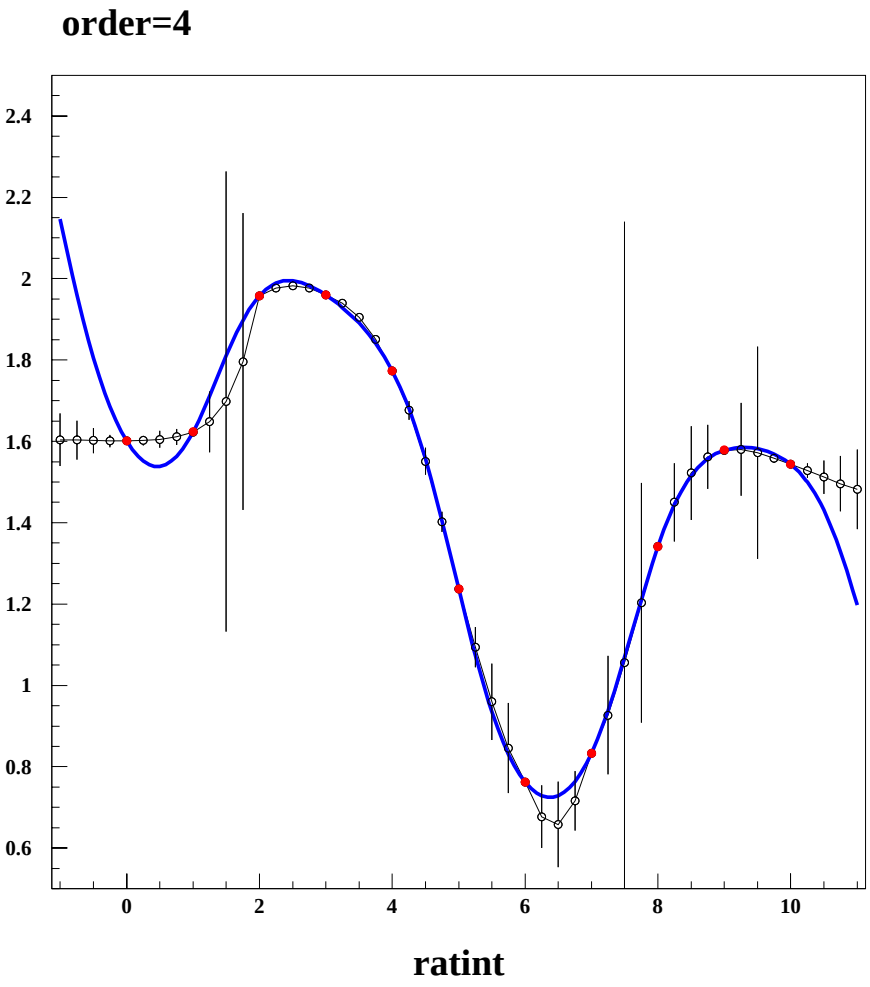
Example of Rational Interpolation

Note unusual functional behaviour. The parent function is not well described by rational functions.

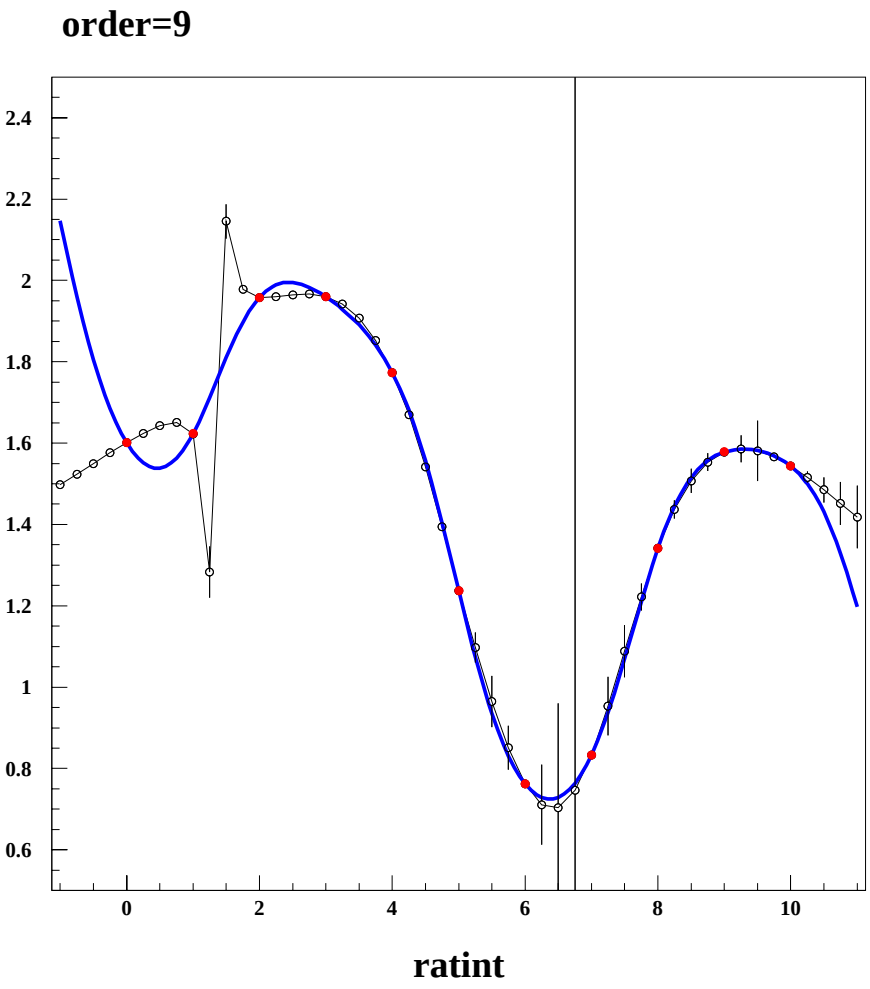


Central values of approximation are better at fourth order.

Large error estimates indicate that last correction was large.



Still a poor approximation in some regions for ninth order:

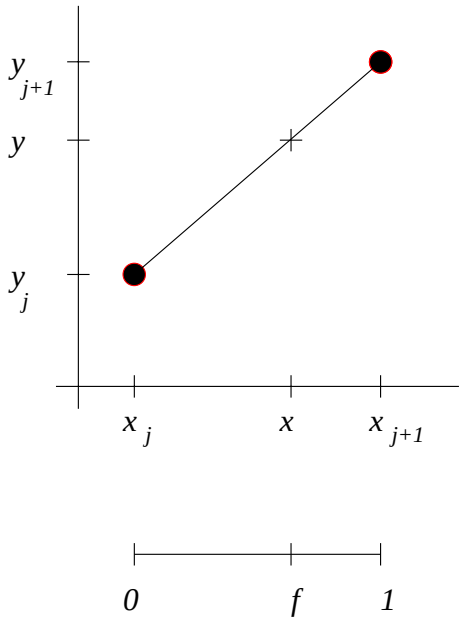


Cubic Spline Interpolation

Designed so that the 1st and 2nd order derivatives are continuous.

Method does not give an error estimate, and cannot be used for extrapolation.

Algorithm: Begin with a linear interpolation:



$$x = f\,x_{j+1} + (1\text{-}f)\,x_j$$
$$y = f\,y_{j+1} + (1\text{-}f)\,y_j$$

This linear interpolation function has $y'' = 0$ in the interval and typically undefined y'' at the end points.

If y_i'' were known at each of the tabulated points, then a cubic polynomial could be added that allows the interpolating function to have y'' vary linearly from one tabulated point to the next. This cubic function would have to be zero at the tabulated points. There is a unique solution:

$$y(x) = f y_{j+1} + (1-f) y_j + g y''_{j+l} + h y''_j$$

where,

$$\begin{aligned} g &= \frac{1}{6} f(f-1)(f+1)(x_{j+1}-x_j)^2 \\ h &= \frac{1}{6} f(f-1)(f-2)(x_{j+1}-x_j)^2 \end{aligned}$$

The additional terms are clearly zero at the endpoints (f=0, f=1), and it is easily shown that:

$$y'' = f y''_{j+1} + (1-f) y''_j \ .$$

One problem: y_i'' are typically not known...

By requiring the first derivatives be continuous across each tabulated point x_j, $j = 2...N-1$, the following relations are found:

$$\frac{x_j - x_{j-1}}{6} y''_{j-1} + \frac{x_{j+1} - x_{j-1}}{3} y''_j + \frac{x_{j+1} - x_j}{6} y''_{j+1} = \frac{y_{j+1} - y_j}{x_{j+1} - x_j} - \frac{y_j - y_{j-1}}{x_j - x_{j-1}}$$

- This gives $N-2$ linear equations for N unknowns
  $\rightarrow$ 2 undetermined parameters
- Two ways to specify a unique solution:
  1) set $y''_1 = y''_N = 0$ (natural spline)
  2) specify y'_1 and y'_N

Numerical Recipes:

Call the following routine once in order to calculate the second derivatives at the tabulated points:

```
SUBROUTINE spline(xa,ya,n,yp1,ypn,y2a)
```

where

`yp1` and `ypn` are to contain the first derivatives at the endpoints. If they are larger than 10^{30}, zero second derivatives on the boundary are assumed instead.

`y2a` is the (returned) array of second derivatives

The following routine may then be called as many times as desired to calculate the interpolated function for any value of x.

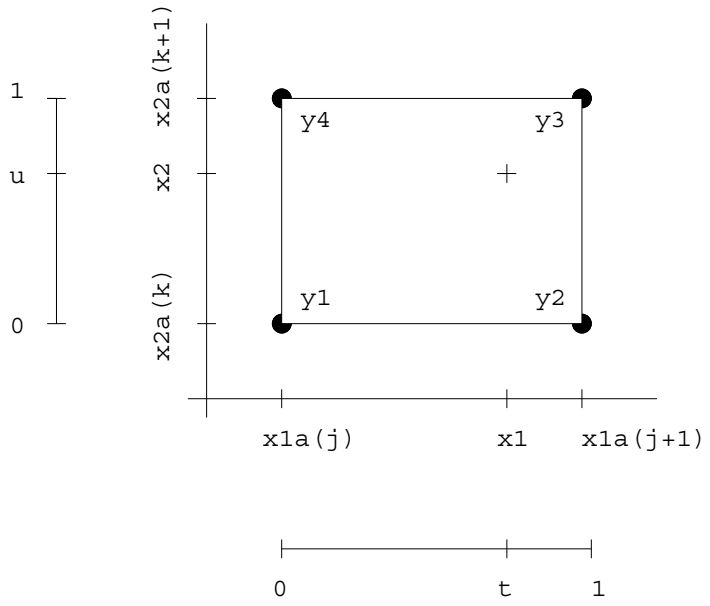
```
SUBROUTINE splint(xa,ya,y2a,n,x,y)
```

Exercise 2:
Use a natural cubic spline to interpolate between tabulated points for $x = 0, 1, ..., 10$ from the function shown on page 26. Show the results in a table, plot the interpolation function and compare to the original function.

Interpolation in 2 or more dimensions

To be specific, consider 2D. Higher dimensions are treated in a similar fashion:

Bilinear interpolation:



$$y(x_1, x_2) = (1-t)(1-u)y_1 + t(1-u)y_2 + tuy_3 + (1-t)uy_4$$

This results in a continuous interpolation function but the gradient is discontinuous at boundaries of each square.

Two possible methods to improve on the Bilinear interpolation:

1) Go to higher order, to improve the accuracy, without fixing the gradient problem. For example, to include m points along the x_1 direction and n points along the x_2 direction, perform m 1D interpolations of order $n-1$. Then use the values of these interpolations at x_2 to do a 1D interpolation of order $m-1$.

Numerical Recipes:

```
SUBROUTINE polin2(x1a,x2a,ya,m,n,x1,x2,y,dy)
```

2) Go to higher order to impose continuity of the gradient or higher derivatives...

Bicubic Interpolation

This method requires additional information for all the tabulated points:

$$\frac{\partial y}{\partial x_1}, \quad \frac{\partial y}{\partial x_2}, \quad \frac{\partial^2 y}{\partial x_1 \partial x_2},$$

Numerical Recipes:

```
SUBROUTINE bcucof(y,y1,y2,y12,d1,d2,c)
```

Bicubic Spline

Use the 1D natural cubic spline interpolation function to determine the derivatives needed for bicubic interpolation.

Numerical Recipes:

```
SUBROUTINE splie2(x1a,x2a,ya,m,n,y2a)
```

```
SUBROUTINE splin2(x1a,x2a,ya,y2a,m,n,x1,x2,y)
```

Integration of Functions

Concentrate on 1D integrals: $I = \int_a^b f(x)\,dx$

Classical methods

Not recommended, but have been around a long time.

Divide x into equal intervals:

$$x_i = x_0 + i\,h \quad i = 0, 1, ..., N+1 \qquad f_i = f(x_i)$$

To evaluate $I = \int_{x_0}^{x_{N+1}} f(x)\,dx$, can use

- closed formula: $I = F(f_0, f_1, ..., f_{N+1})$
- open formula: $I = F(f_1, f_2, ..., f_N)$

Open formulas are especially useful if the function is poorly behaved at one or both endpoints of the integral.

Closed Formulas

Trapezoidal rule:

$$\int_{x_1}^{x_2} f(x)\,dx = h[\frac{1}{2}f_1 + \frac{1}{2}f_2] + \mathcal{O}(h^3\ f'')$$

is exact for linear functions.

Next higher order formula: Simpson's Rule

$$\int_{x_1}^{x_3} f(x)\,dx = h[\frac{1}{3}f_1 + \frac{4}{3}f_2 + \frac{1}{3}f_3] + \mathcal{O}(h^5\, f^{(4)})$$

is exact for polynomials up to third order.

Extended Closed Formulas

Extended Trapezoidal rule:

$$\begin{aligned}
\int_{x_1}^{x_3} f(x)\,dx &= \int_{x_1}^{x_2} f(x)\,dx + \int_{x_2}^{x_3} f(x)dx \\
&= h[\frac{1}{2}f_1 + \frac{1}{2}f_2] + h[\frac{1}{2}f_2 + \frac{1}{2}f_3] + 2\mathcal{O}(h^3\, f'') \\
\int_{x_1}^{x_N} f(x)\,dx &= h[\frac{1}{2}f_1 + f_2 + f_3 + ... + f_{N-1} + \frac{1}{2}f_N] \\
&\quad + \mathcal{O}(\frac{1}{N^2})
\end{aligned}$$

Note: $N\mathcal{O}(h^3 f'') = N\mathcal{O}(\frac{(b-a)^3}{N^3} f'') = \mathcal{O}(\frac{1}{N^2})$

Extended Simpson rule:

$$\begin{aligned}
\int_{x_1}^{x_N} f(x)\,dx &= h[\frac{1}{3}f_1 + \frac{4}{3}f_2 + \frac{2}{3}f_3 + \frac{4}{3}f_4 ... + \frac{2}{3}f_{N-2} \\
&\quad + \frac{4}{3}f_{N-1} + \frac{1}{3}f_N] + \mathcal{O}(\frac{1}{N^4})
\end{aligned}$$

Extrapolation Formulas

$$\int_{x_0}^{x_1} f(x)\,dx = h\,f_1 + \mathcal{O}(h^2 f')$$
$$\int_{x_0}^{x_1} f(x)\,dx = h[\frac{3}{2}f_1 - \frac{1}{2}f_2] + \mathcal{O}(h^3 f'')$$
$$\int_{x_0}^{x_1} f(x)\,dx = h[\frac{23}{12}f_1 - \frac{16}{12}f_2 + \frac{5}{12}f_3] + \mathcal{O}(h^4 f''')$$

Extended Open Formulas

Just add the extrapolation formulas to the closed formulas:
semi-open:

$$\int_{x_0}^{x_N} f(x)\,dx = h[\frac{3}{2}f_1 + f_2 + ... + f_{N-1} + \frac{1}{2}f_N] + \mathcal{O}(\frac{1}{N^2})$$

open:

$$\int_{x_0}^{x_{N+1}} f(x)\,dx = h[\frac{3}{2}f_1 + f_2 + ... + f_{N-1} + \frac{3}{2}f_N] + \mathcal{O}(\frac{1}{N^2})$$

Higher order formulas exist which converge as $(\frac{1}{N^3})$, $(\frac{1}{N^4})$.
See text.

Extended midpoint rule:

$$\int_{x_0}^{x_{N+1}} f(x)\,dx = h[f_{\frac{1}{2}} + f_{\frac{3}{2}} + ... + f_{N+\frac{1}{2}}] + \mathcal{O}(\frac{1}{N^2})$$

Elementary Algorithms

One approach to use is to start with a small value of N and re-evaluate integral for increasing N. The extended trapezoidal rule is the easiest to use for this. It is not the fastest to converge (in terms of N), but has the advantage that as N is increased, previous results can be used directly (thus reducing the number of calls to determine $f(x)$).

ie:

$$\begin{aligned} I_1 &= (b-a)[\frac{1}{2}f_a + \frac{1}{2}f_b] \\ I_2 &= \frac{(b-a)}{2}[\frac{1}{2}f_a + f(\frac{x_a+x_b}{2}) + \frac{1}{2}f_b] \\ I_3 &= \frac{(b-a)}{3}[\frac{1}{2}f_a + f(\frac{3x_a+x_b}{4}) + f(\frac{x_a+x_b}{2}) \\ &\quad + f(\frac{x_a+3x_b}{4}) + \frac{1}{2}f_b] \end{aligned}$$

The method is then to evaluate I_1, I_2, I_3, ... and stop when $|(I_{j+1} - I_j)/I_j| <$ tolerance.

Numerical Recipes:

```
SUBROUTINE qtrap(func,a,b,s)
```

where `s` is the result. The tolerance is set to be 10^{-6} but should be careful that machine precision doesn't prevent the result from converging.

To be even more efficient, use the fact that the error in the trapezoidal method is even in $1/N$:

$$I = \int_a^b f(x)\,dx = S_N + \frac{\alpha}{N^2} + \frac{\beta}{N^4} + ...$$

$$I = S_{2N} + \frac{\alpha}{4N^2} + \frac{\beta}{16N^4} + ...$$

You can cancel out the $1/N^2$ error:

$$I = \frac{4}{3}S_{2N} - \frac{1}{3}S_N - \frac{\beta}{4N^4} + ...$$

and so this formula is accurate to order $1/N^4$. In fact this is just the Simpson rule!

Numerical Recipes:

```
SUBROUTINE qsimp(func,a,b,s)
```

where `s` is the result.

Romberg Integration

This is just the extension of the technique of cancelling successive terms of the error. It is equivalent to an extrapolation of S_N as $h \to 0$.

Numerical Recipes:

```
SUBROUTINE qromb(func,a,b,s)
```

The subroutine uses the trapezoidal rule for $N = 1, 2, 4, 8, ...$ and uses `polint` to extrapolate to $h \to 0$. This subroutine has much faster convergence than `qtrap` or `qsimp`.

Improper Integrals

If the integrand is poorly behaved at the endpoints, the extended midpoint rule can be used instead of the trapezoidal rule, and Romberg integration can again be performed:

Numerical Recipes:

```
SUBROUTINE qromo(func,a,b,s,choose)
```

where `choose` is a NR subroutine name. `midpnt` would be used for an integral poorly behaved at the endpoints.

If the integral has limits $a = -\infty$ or $b = \infty$, make a change of variables.

$$\int_a^b f(x)\,dx = \int_{\frac{1}{a}}^{\frac{1}{b}} \frac{1}{t^2} f\left(\frac{1}{t}\right) dt$$

This is only valid if the range of the integral does not contain $x = 0$. Otherwise it is necessary to break the integral into two. The change of variables can be done analytically, or it could be handled automatically:

```
call qromo(func,a,b,s,midinf)
```

Special cases

Integrands with power law singularities at upper or lower limits:

$$f(x) \approx (x-a)^{-\gamma} \qquad 0 < \gamma < 1$$

then let $t = (x-a)^{1-\gamma}$. For $\gamma = \frac{1}{2}$, then

$$\int_a^b f(x)\,dx = \int_0^{\sqrt{b-a}} 2t\, f(a+t^2)\,dt$$

Numerical Recipes:

```
call qromo(func,a,b,s,midsql)
```

to deal with lower limit inverse square root divergences. Use `qromo(func,a,b,s,midsqu)` to deal with upper limit inverse square root divergences.

For a integrand that falls off exponentially, the change of variables: $x = -\log t$ gives,

$$\int_a^\infty f(x)\,dx = \int_0^{e^{-a}} f(-\log t)\,\frac{dt}{t}$$

```
call qromo(func,a,b,s,midexp)
```

Gaussian Quadrature

Methods presented so far involve breaking the range into N equal intervals, evaluating the integrand at the interval boundaries, and forming the sum,

$$I = \sum_{i=1}^{N} \alpha_i \, f_i$$

where the weights α_i depend on the order of the calculation. Polynomials of that order or less are handled exactly by these methods.

Gaussian Quadrature estimates an integral using unequal intervals. This allows an extended class of integrands to be treated exactly. For example, a known function $W(x)$ times a polynomial $f(x)$ is integrated using,

$$\int_a^b W(x)\, f(x)\, dx = \sum_{i=1}^{N} w_i \, f(x_i)$$

which will be exact for a polynomial with order $< 2N$.

How are w_i and x_i determined? Not easy to do.

- look up in tables
- use specific routines

General Idea:

Consider the set of orthogonal polynomials over a function $W(x)$:

$$\langle p_i | p_j \rangle = \int_a^b W(x)\, p_i(x)\, p_j(x)\, dx = \alpha_i \delta_{ij}$$

For an N-point Gaussian quadrature,

- x_i are the roots of $p_N(x)$ (all between a and b)

$$w_j = \frac{\langle p_{N-1} | p_{N-1} \rangle}{p_{N-1}(x_j) p'_N(x_j)}$$

Recurrance relations can be used to form the orthogonal polynomials and their roots can be found numerically.

Numerical Recipes provide several routines for special choices of $W(x)$. For example, $W(x) = 1$ correspons to Gauss-Legendre quadrature. Use the following routine:

```
SUBROUTINE gauleg(x1,x2,x,w,n)
```

where `x1` and `x2` are the limits and n the number of points. The routine returns, `x(n)` and `w(n)`, so that the integral can be evaluated:

$$\int_{x_1}^{x_2} W(x), f(x)\, dx = \sum_{j=1}^{N} w_j f(x_j)$$

name	W(x)		subroutine
Gauss-Legendre	1		`gauleg(x1,x2,x,w,n)`
Gauss-Laguerre	$x^\alpha\, e^{-x}$	$0 < x < \infty$	`gaulag(x,w,n,alf)`
Gauss-Hermite	e^{-x^2}	$-\infty < x < \infty$	`gauher(x,w,n)`
Gauss-Jacobi	$(1-x)^\alpha(1+x)^\beta$	$-1 < x < 1$	`gaujac(x,w,n,alf,bet)`

Multidimensional Integrals

Difficult for two reasons:

- number of function evaluations for an integral in N dimensions scales as α^N
- the boundary, (an $N-1$ dimensional surface) may be complicated.

As long as high precision is not required, Monte Carlo integration is usually the easiest to impliment, especially if the boundary is complicated.

For smooth functions to be integrated over a region with a simple boundary, repeated one dimensional integration can be performed:

$$\begin{aligned} I &= \int\int\int f(x,y,z)\,dx\,dy\,dz \\ &= \int_{x_1}^{x_2} dx \int_{y_1(x)}^{y_2(x)} dy \int_{z_1(x,y)}^{z_2(x,y)} dz\, f(x,y,z) \\ &= \int_{x_1}^{x_2} H(x)dx \end{aligned}$$

$H(x)$ is given by

$$\begin{aligned} H(x) &= \int_{y_1(x)}^{y_2(x)} dy \int_{z_1(x,y)}^{z_2(x,y)} dz\, f(x,y,z) \\ &= \int_{y_1(x)}^{y_2(x)} G(x,y)\, dy \end{aligned}$$

where,

$$G(x,y) = \int_{z_1(x,y)}^{z_2(x,y)} f(x,y,z)\, dz$$

The implementation depends on whether the system allows recursion (subroutine calling itself). The evaluation of I involves calling a integration routine, say `qgaus` with H as the integrand. The evaluation of H and G also involves calling `qgaus`. So `qgaus` calls H which calls `gaus` which calls G which calls `qgaus`.

If recursion is not allowed, then three copies of the `qgaus` routine need to be created, each with a unique name so that each subprogram calls a different version.

If recursion is allowed:

```
call qgaus(H,x1,x2,s)

SUBROUTINE H(xx)
COMMON /xyz/x,y,z
x=xx
call qgaus(G,y1(x),y2(x),s)
H=s
return
end

SUBROUTINE G(yy)
COMMON /xyz/x,y,z
y=yy
call qgaus(F,z1(x,y),z2(x,y),s)
G=s
return
end

SUBROUTINE F(zz)
COMMON /xyz/x,y,z
z=zz
F=func(x,y,z)
return
end
```

Exercise #3

The convolution of an exponential decay and a Gaussian resolution function is given by:

$$f(t) = \int_0^\infty \frac{e^{-\frac{(t-t')^2}{2\sigma_t^2}}}{\sqrt{2\pi}\sigma_t} \frac{e^{-\frac{t'}{\tau}}}{\tau}\, dt'$$

Evalulate this integral using the Gauss-Laguerre quadrature, with $\alpha = 0$, for $\tau = 1$, $\sigma_t = 0.5$, and $t = -2, -1.5, ..., 5.5, 6$. Use $N = 5, 10, 15, 20$ and compare to the analytic solution:

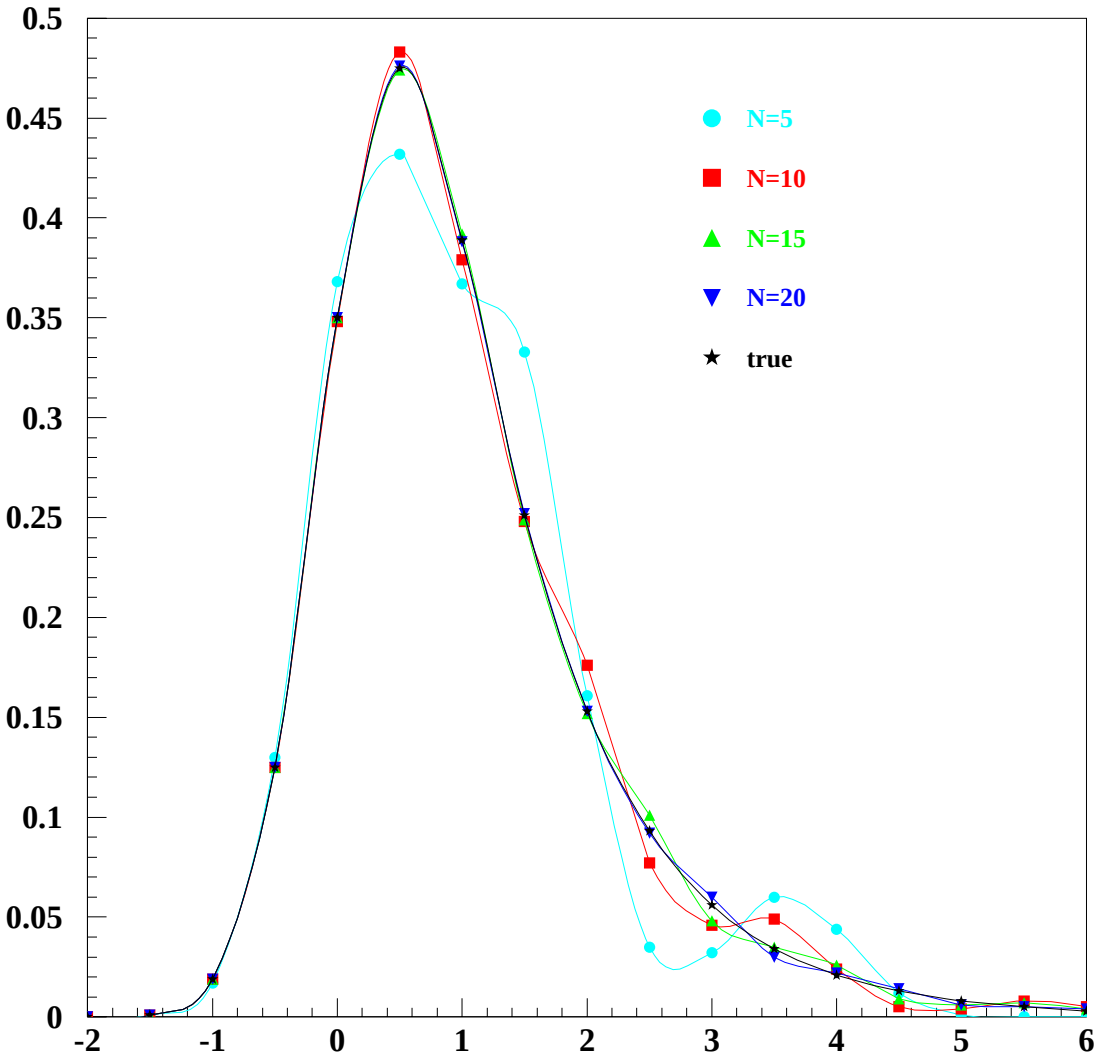
$$f(t) = \frac{1}{2\tau}\exp\left(\frac{\sigma_t^2}{2\tau^2} - \frac{t}{\tau}\right)\operatorname{erfc}\left(\frac{\sigma_t}{\sqrt{2}\tau} - \frac{t}{\sqrt{2}\sigma_t}\right)$$

Also evalute the double integral,

$$I = \int_{-2}^{10} f(t)\, dt$$

for the same choices for N. For this exercise, do not substitute the analytical solution for $f(t)$, but instead perform the double integral using `qromb` and `gaulag`.

Solution


0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
-2
-1
0
1
2
3
4
5
6
N=5
N=10
N=15
N=20
true

Root Finding

General problem of solving a nonlinear equation:

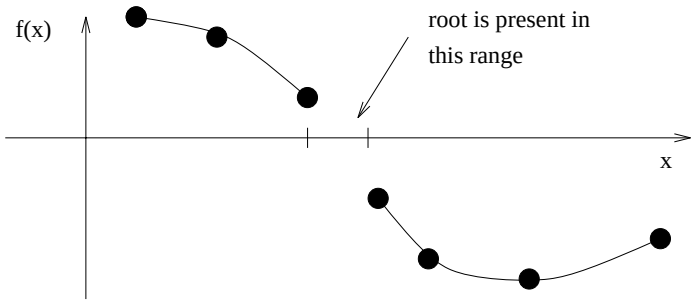
$$\begin{aligned} g(x) &= h(x) \\ g(x) - h(x) &= 0 \\ f(x) &= 0 \end{aligned}$$

or in N dimensions,

$$\mathbf{f}(\mathbf{x}) = 0$$

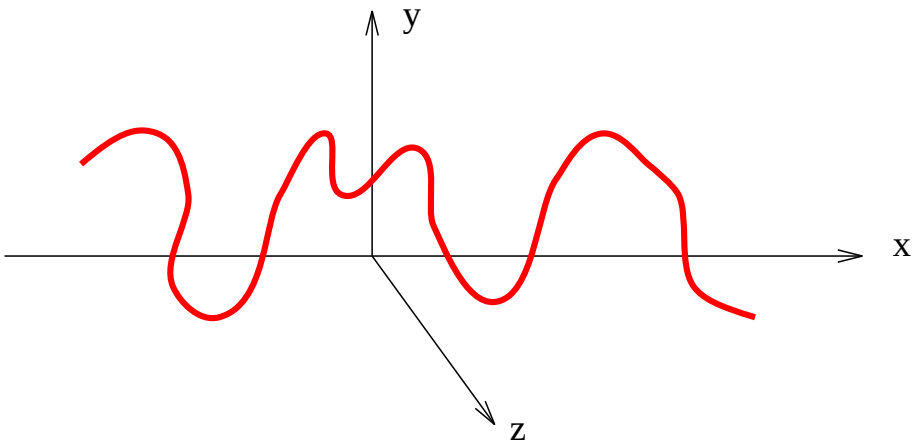
or in other words, N simultaneous equations.

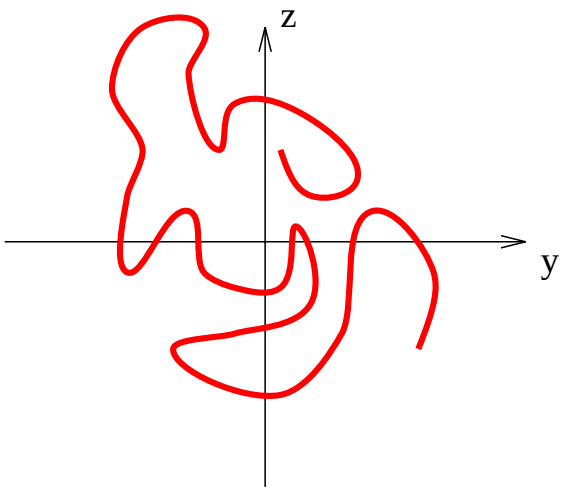
The problem is much simpler in 1 dimension, because it is possible to define a range where a root must exist:



It can be difficult to find a bracketed region if two roots are near each other.

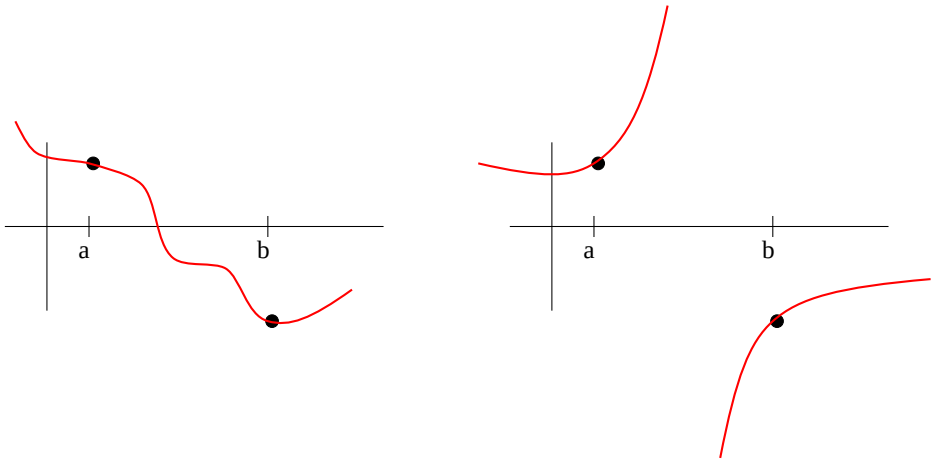
In two dimensions root bracketing is not possible. Consider the system, $y(x) = 0$, and $z(x) = 0$. This defines a curve, as shown below. It is not possible to bracket a region $[x_1, x_2]$ in which it is known that a root exists.





Bracketing

A root is bracketed in (a, b) if $f(a)$ and $f(b)$ have opposite signs. It must contain at least one root, unless a singularity is present:



Numerical Recipes provide two simple bracketing utilities:

```
SUBROUTINE zbrac(func,x1,x2,succes)
```

This routine begins with the range (x_1, x_2) and expands it until the range brackets a root. If successful, it sets `success=true`, and the new range is returned in `x1` and `x2`.

```
SUBROUTINE zbrak(func,x1,x2,n,xb1,xb2,nb)
```

This routine breaks the range (x_1, x_2) into n intervals and returns the number `nb` and the ranges `xb1(1:nb),xb2(1:nb)` of those intervals that bracket roots. On input `nb` specifies the maximum number sought.

Bisection

Starting from a bracketed range, evaluate the function at the midpoint of the range. Thus a new bracketed range, half the size is found. The size of the interval after $n+1$ iterations is

$$\epsilon_{n+1} = \frac{1}{2}\epsilon_n$$

and the iterations stop when $\epsilon_n < \epsilon$, the desired tolerance. Care must be taken when defining ϵ:

$\epsilon = 10^{-6}$ not possible for $x_{\text{root}} = 10^{20}$, and
$\epsilon/x_{\text{root}} = 10^{-6}$ not good for x_{root} near 0.

Properties of bisection method:

- not the most efficient
- guaranteed to work
- does not distinguish singularities from roots
- will find only one root

This method is said to converge linearly, since $\epsilon_{n+1} = \alpha\epsilon_n$, other methods converge superlinearly:
$\epsilon_{n+1} = \alpha(\epsilon_n)^m, \quad m > 1.$

Numerical Recipes:

```
FUNCTION rtbis(func,x1,x2,xacc)
```

returns the root as `rtbis` once it has been determined to be within an interval of $\pm$`xacc`.

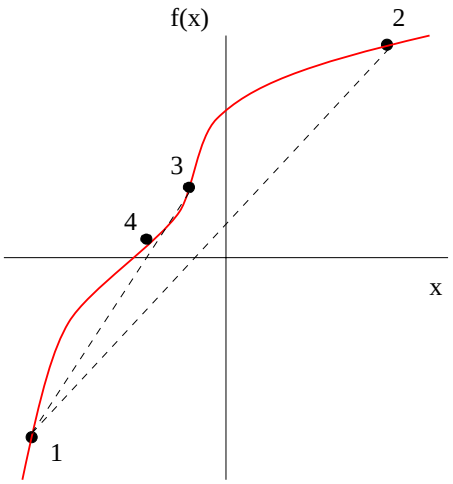
False Position and Secant methods

Instead of choosing the middle of the interval, these methods assume the function is linear in the region of interest, to decide next point to evaluate.

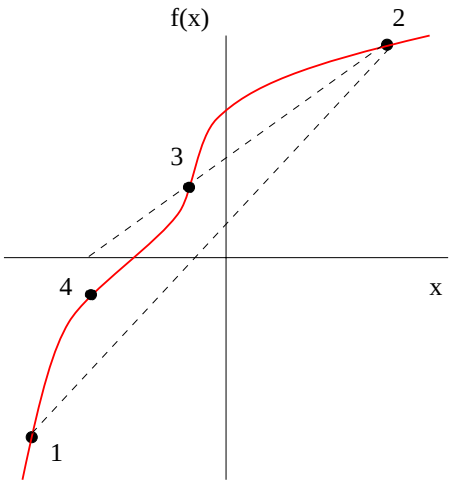
False position: maintain the bracket
Secant method: use the two most recent points

Numerical Recipes:

```
FUNCTION rtflsp(func,x1,x2,xacc)
FUNCTION rtsec(func,x1,x2,xacc)
```



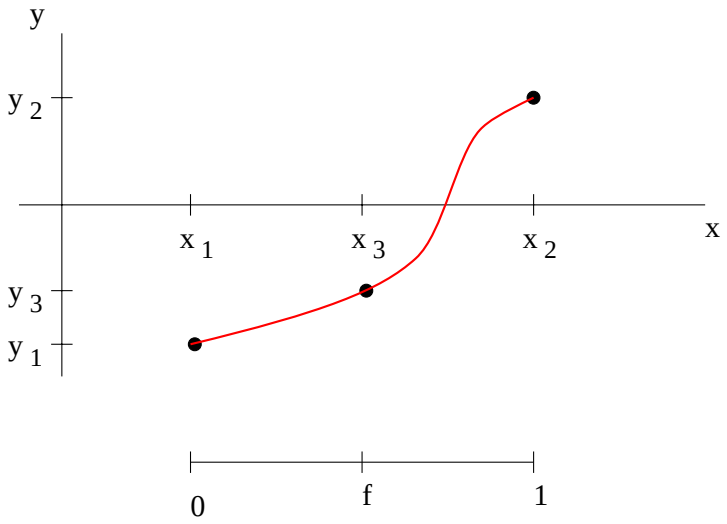
False position method



Secant Method

Neither are usually the best choice. Use Ridders' or Brent's method instead.

Ridders Method



A linear interpolation of the function in the bracketed range is given by:

$$y = (1 - f)\, y_1 + f\, y_2 \qquad f = \frac{x - x_1}{x_2 - x_1}$$

Instead, Ridders method uses an exponential interpolation:

$$y = (1 - f)\, y_1\, Q^{-f} + f\, y_2\, Q^{1-f} \qquad Q > 0$$

In order to determine Q, use the midpoint, $f = \frac{1}{2}$:

$$\sqrt{Q} = \frac{y_3 + \text{sign}[y_2](y_3^2 - y_1 y_2)^{\frac{1}{2}}}{y_2}$$

The next point x_4 is selected to be the root of the exponential interpolation:

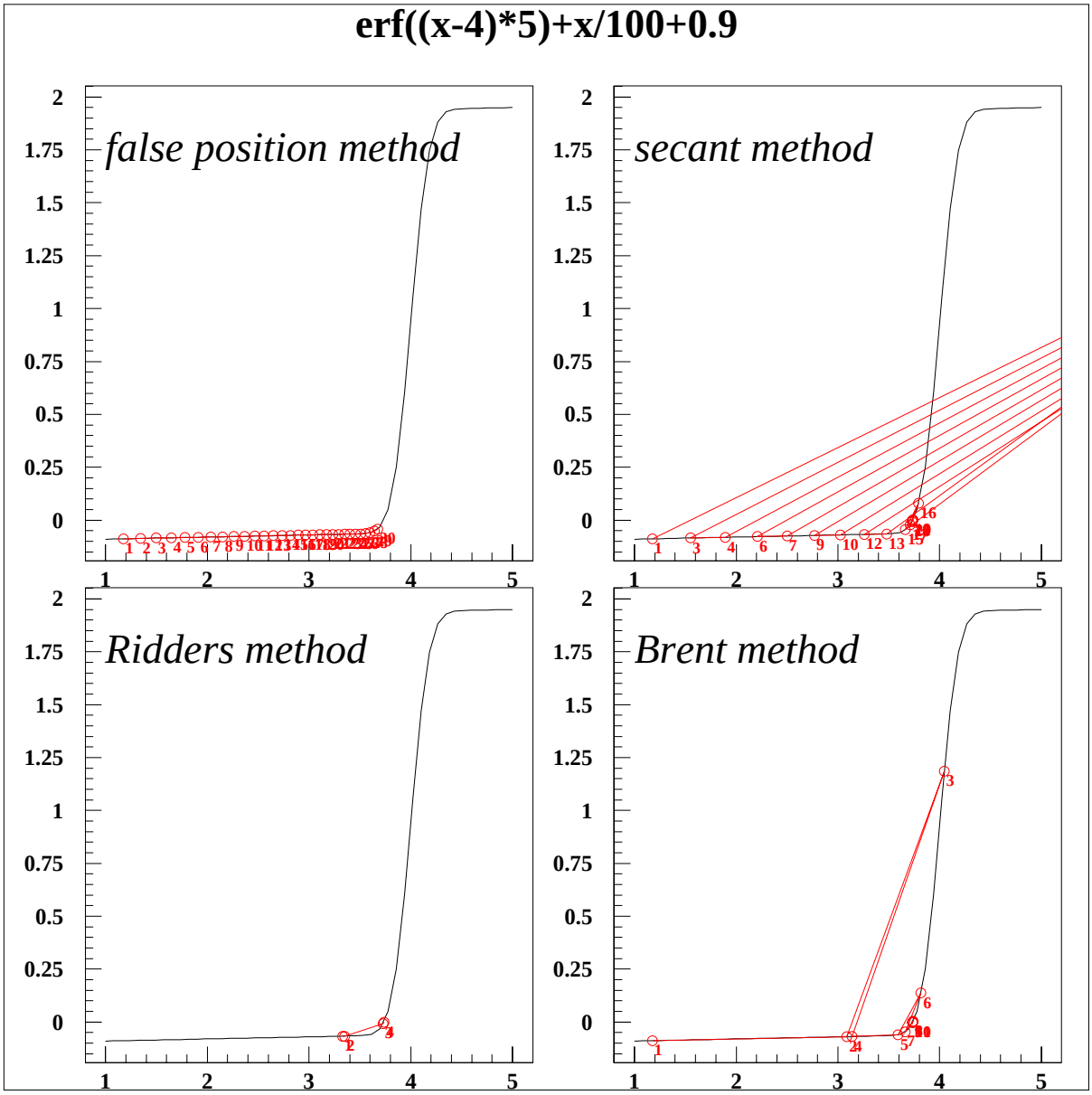
$$x_4 = x_3 + (x_3 - x_1)\frac{\text{sign}[y_1 - y_2]y_3}{\sqrt{y_3^2 - y_1 y_2}}$$

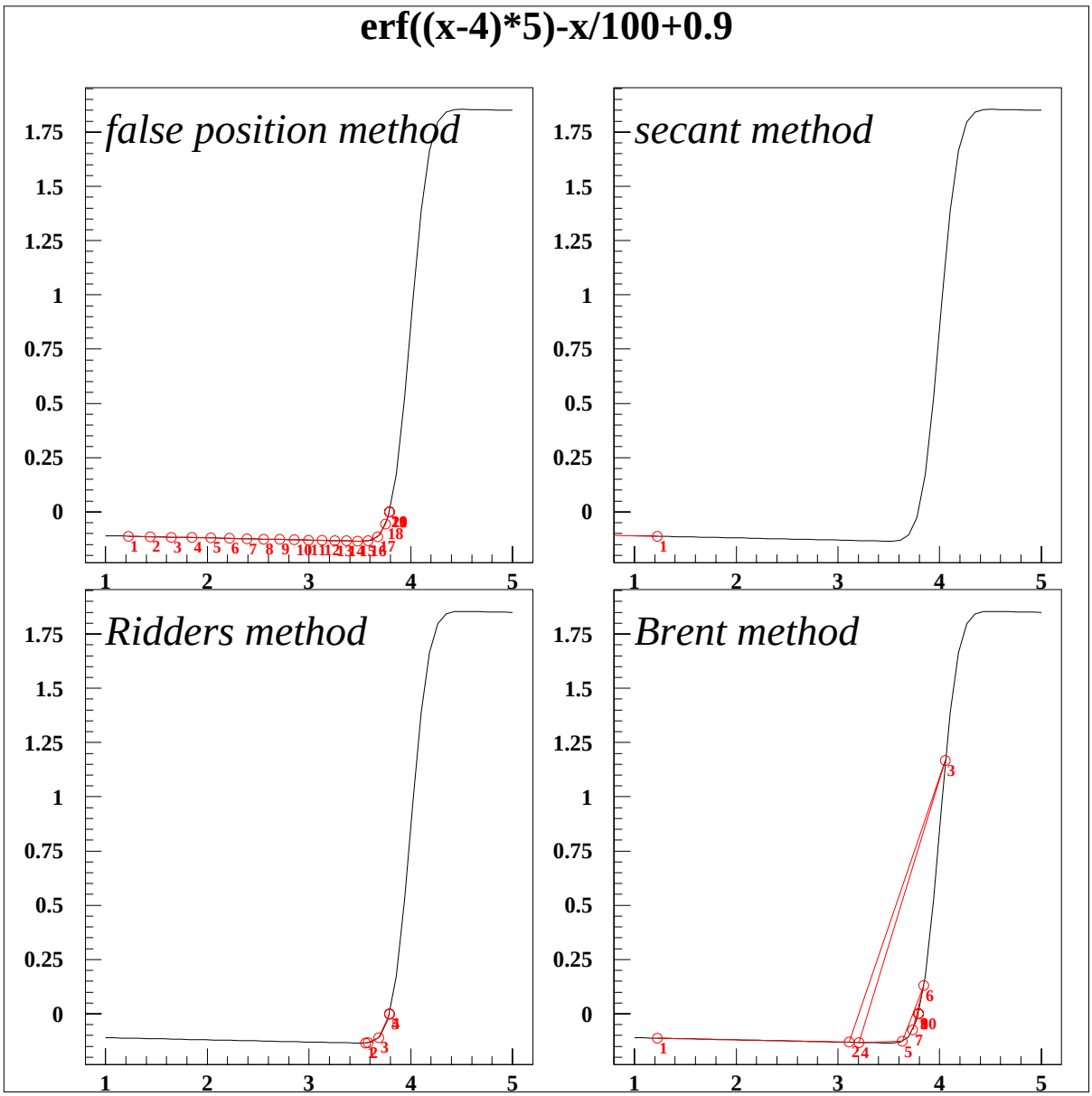
Since the bracket is maintained, it is a robust method, and the convergence is superlinear, $m = \sqrt{2}$.

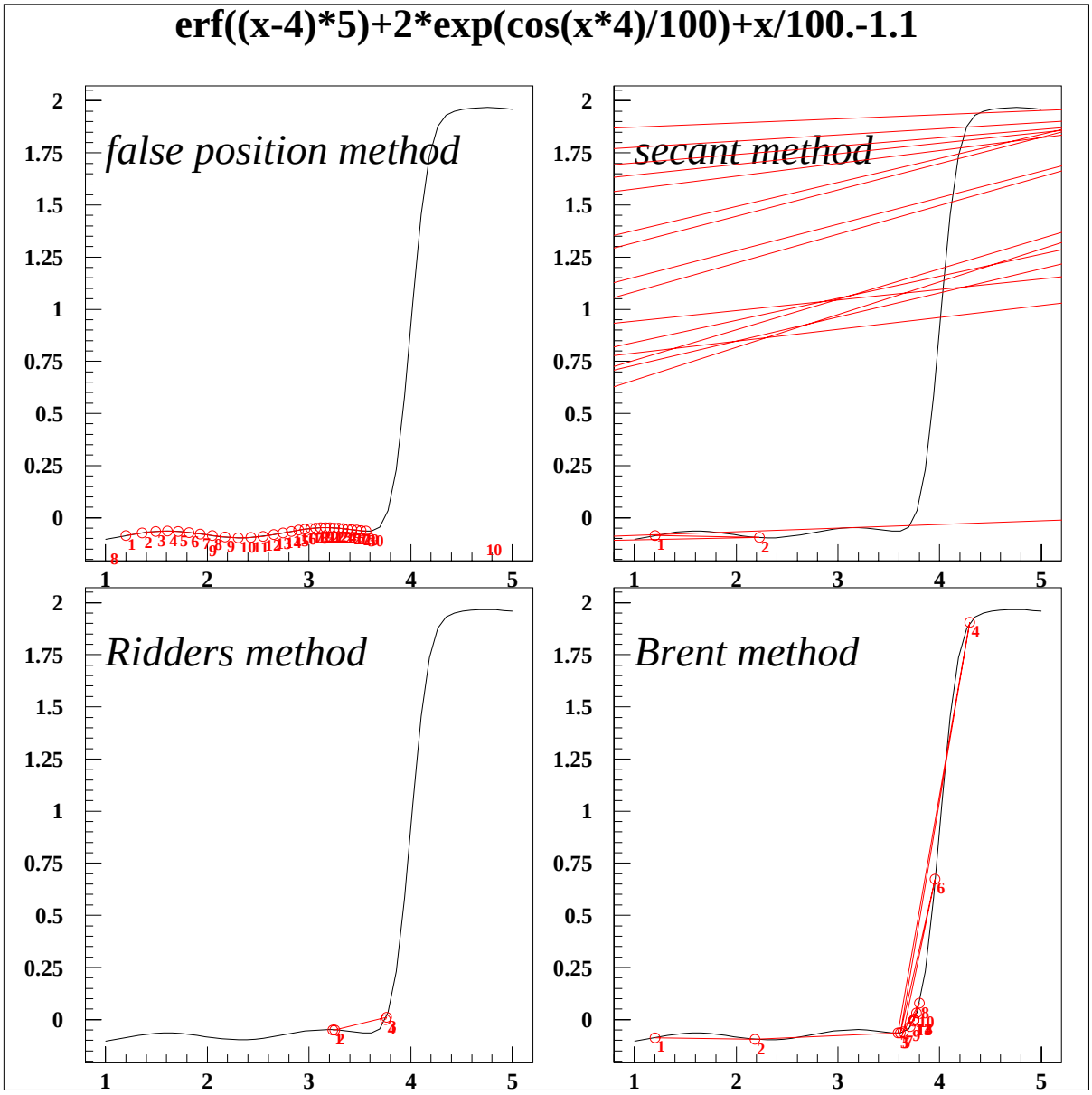
Brent Method

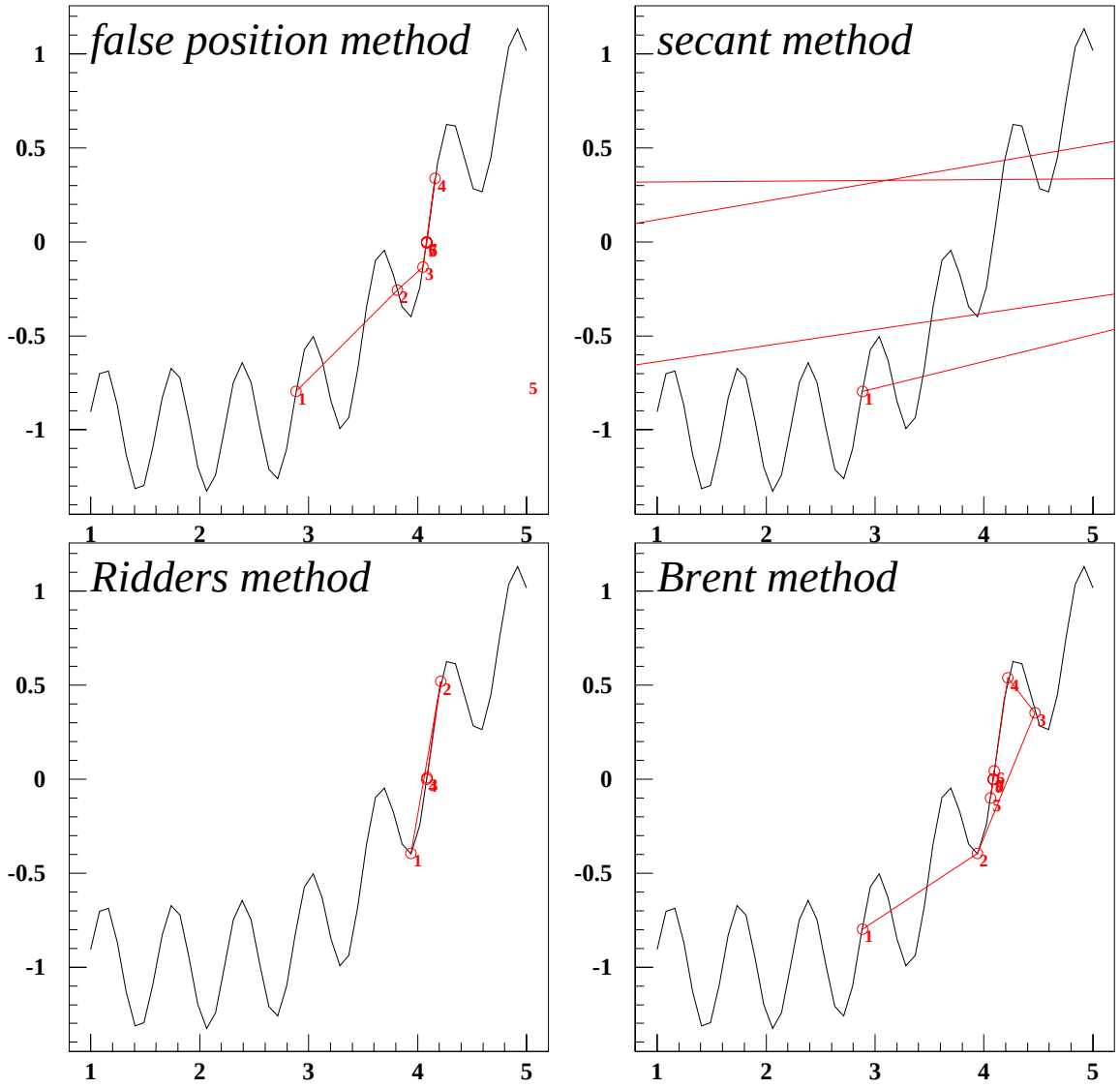
Rather than using a linear interpolation, as in the secant method, a quadratic interpolation is made. Checks are made to ensure the method is converging rapidly, and if not, a bisection step is made. It is thus both robust and fast.

The following four figures compare the convergence of various one dimensional root finding algorithms. For these examples, it is seen that the false position method can sometimes be slow to converge, and the secant method sometimes fails.


erf((x-4)*5)+x/100+0.9
false position method
secant method
Ridders method
Brent method


erf((x-4)*5)-x/100+0.9
false position method
secant method
Ridders method
Brent method


erf((x-4)*5)+2*exp(cos(x*4)/100)+x/100.-1.1
false position method
secant method
Ridders method
Brent method


cos(x*10-5)/3+erf(x-4)
false position method
secant method
Ridders method
Brent method

Newton-Raphson Method

This method requires the calculation of both $f(x)$ and $f'(x)$. From an initial starting value x it uses the linear approximation,

$$f(x+\delta) \approx f(x) + \delta\, f'(x)$$

to determine the next point to try, $x+\delta$,

$$f(x+\delta) = 0 \quad \rightarrow \quad \delta = -f(x)/f'(x) \quad .$$

Should the procedure bring you close to a local maximum or minimum, δ can become quite large, causing the algorithm to fail. It is also possible to get into an infinite loop. Otherwise the convergence is very fast, as long as there is no penalty for calculating $f'(x)$.

Numerical Recipes:

```
FUNCTION rtnewt(funcd,x1,x2,xacc)
```

where `funcd(x,fn,df)` returns the function and its derivative.

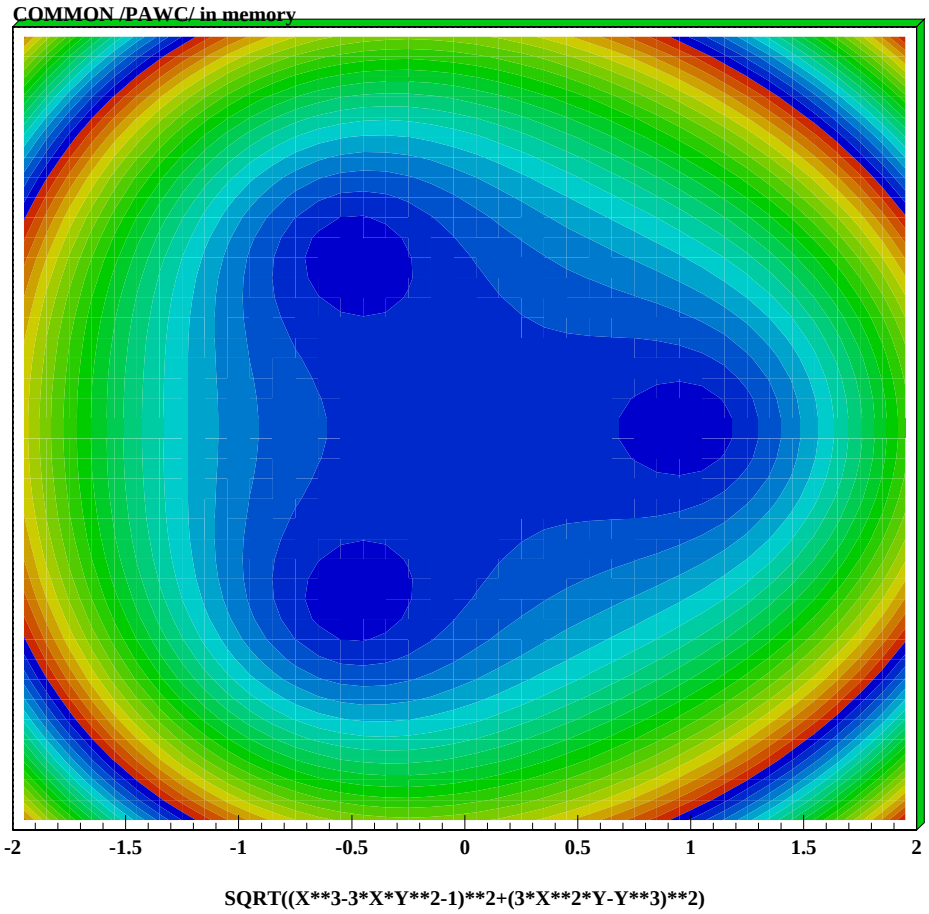
A fail-safe routine, that protects against leaving the bracketed region and against infinite loops, uses the bisection method in addition:

```
FUNCTION rtsafe(funcd,x1,x2,xacc)
```

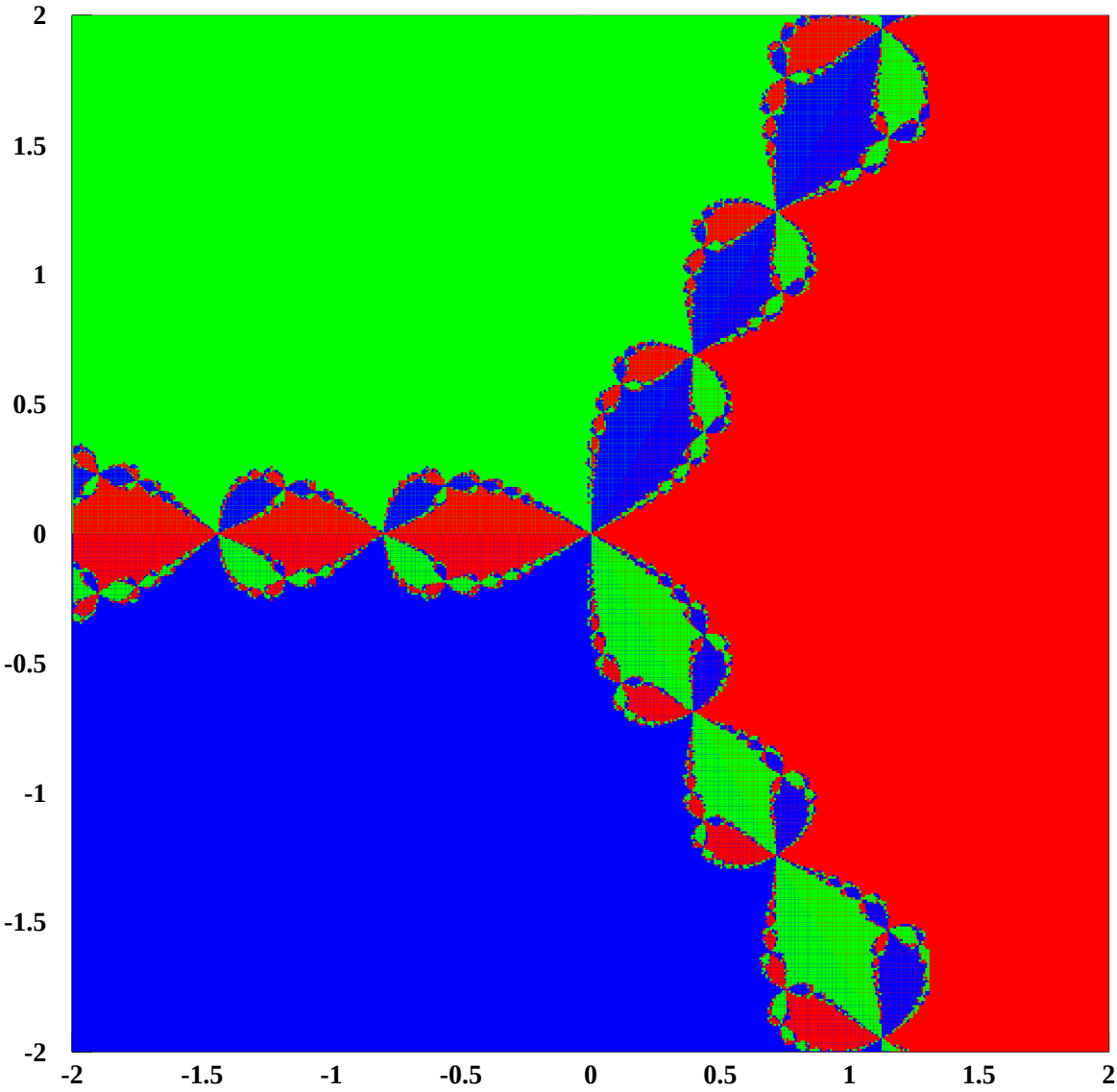
Newton-Raphson and Fractals

The Newton-Raphson method can have poor convergence, depending on the problem and the initial conditions. It is interesting to determine the set of starting values that will lead to a particular root.

For example, $f(z) = z^3 - 1 = 0$ will converge for all positive starting values, but not certain negative values. If one considers the complex roots as well, there are three roots. The following contour plot shows $|f(z)|$.



The following plot shows the starting points that lead to each of the roots, three fractals.



Roots of Polynomials

- a polynomial of order n has n roots, some may be complex
- can be a difficult problem for high order polynomials, especially when two roots are nearby
- when each root is found the order of the polynomial can be reduced by one order:

$$Q(x) = P(x)/(x - r)$$

You can use `poldiv(u,n,v,nv,q,r)` to do this division, but the successive roots can be susceptible to rounding errors. It is recommended to always polish them up, by using them as initial guesses with the original function $P(x)$.

Note: you should never evaluate the polynomial,

$$P(x) = c_1 + c_2 x + c_3 x^2 + c_4 x^3 + c_5 x^4$$

as

```
p = c(1)+c(2)*x+c(3)*x**2+c(4)*x**3+c(5)*x**4
```

but instead as

```
p = c(1) + x*(c(2) + x*(c(3) + x*(c(4) + x*c(5))))
```

which reduces steps and improves precision.

Laguerre's Method

For polynomials with all real roots this method is guaranteed to converge to a root for any starting point. It works well with complex roots, but not guaranteed.

Method:

Assume one root is a distance a from the current guess and *all* the other roots are a distance b away. Use $P(x)$, $P'(x)$, and $P''(x)$ to solve for a, then take $(x - a)$ as the next guess. Continue process until a becomes small.

Numerical Recipes:

```
SUBROUTINE laguer(a,m,x,its)
```

where `a` and `x` are complex and

`a(1:m+1)`	the coefficients
`m`	the order of the polynomial
`x`	input: starting point, output: solution
`its`	the number of iterations taken

To find all the roots use the driver routine:

```
SUBROUTINE zroots(a,m,roots,polish)
```

where `polish` can be set to `.true.` if polishing of the roots is desired.

Systems of Nonlinear Equations

No general methods exist, even for the two dimensional problem,

$$f(x,y) = 0 \qquad g(x,y) = 0 \quad .$$

Each equation defines a set of *a priori* unknown number of separate curves. Where these two sets of curves intersect is the solution to the problem.

If you have a good enough initial guess, then you can use the Newton-Raphson method.

$$F_i(\mathbf{x} + \delta\mathbf{x}) = F_i(\mathbf{x}) + \sum_{j=1}^{N} \frac{\partial F_i}{\partial x_j}\,\delta x_j + \mathcal{O}(\delta\mathbf{x}^2)$$

Neglect the $\mathcal{O}(\delta\mathbf{x}^2)$ term, set the LHS to zero and solve for $\delta\mathbf{x}$, using matrix methods. Use $\mathbf{x}_{\text{new}} = \mathbf{x}_{\text{old}} + \delta\mathbf{x}$ as the next point in the iteration.

Numerical Recipes:

```
SUBROUTINE mnewt(ntrial,x,n,tolx,tolf)
```

a maximum of `ntrial` iterations are made to improve on the initial estimate of `x`. Iteration stops if either $\sum |\delta x_i| <$ `tolx` or $\sum |F_i| <$ `tolf`.

A more globally convergent technique checks that

$$f = \sum_i F_i^2$$

reduces each time a new $\delta\mathbf{x}$ is calculated, Otherwise a smaller step is taken:

$$\mathbf{x}_{\text{new}} = \mathbf{x}_{\text{old}} + \lambda\,\delta\mathbf{x} \qquad \mathbf{0} < \lambda < \mathbf{1}$$

Numerical Recipes:

```
SUBROUTINE lnsrch(n,xold,fold,g,p,x,f,stpmax
                                    ,check,func)
```

If the derivatives are not known, the following driver routine can be used instead:

```
SUBROUTINE newt(x,n,check)
```

which computes partial derivatives numerically.

Exercise 4:

For blackbody radiation, the radiant energy per unit volume in the wavelength range λ to $\lambda + d\lambda$ is,

$$u(\lambda)\,d\lambda = \frac{8\pi}{\lambda^5}\,\frac{hc}{\exp(hc/\lambda kT) - 1}\,d\lambda$$

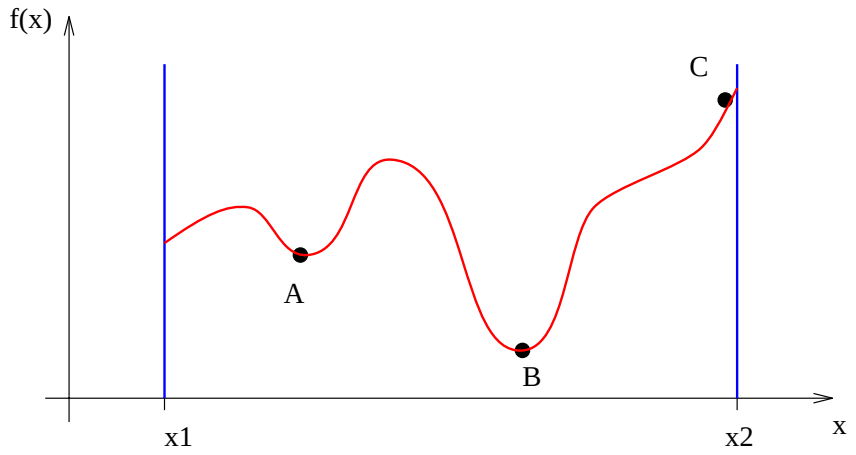
where T is the temperature of the body, c is the speed of light, h is Planck's constant, and k is Boltzmann's constant. Show that the wavelength at which $u(\lambda)$ is maximum may be written as $\lambda_{\max} = \alpha hc/kT$, where α is a constant. Determine the value of α numerically from the resulting transcendental equation.

Minimization (Maximization) of Functions

General problem is to find $\mathbf{x}$ that minimizes $f(\mathbf{x})$ with as few function calls as possible. It is called for in doing likelihood fits, and optimization problems.

Most programs are designed to find a minimum. To find the maximum of $f(\mathbf{x})$, simply find the minimum of $-f(\mathbf{x})$.

Types of minima in 1D:



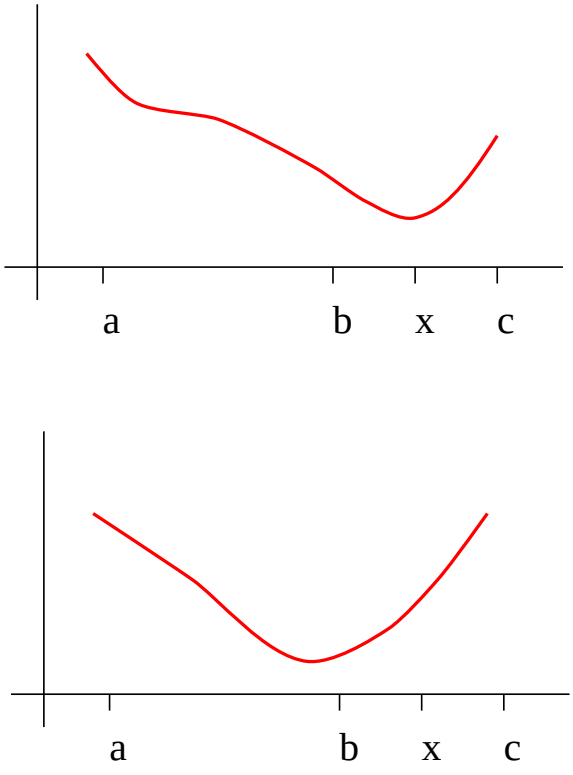
A is a local minimum. B is the global minimum. C is the global maximum at the boundary, and so $f'(x) \neq 0$ at that location.

Golden Section Search in 1D

In a similar way as bracketing a root, one can bracket a minimum with three points if:

$$\begin{aligned} a < \quad & b \quad < c \\ f(a) > \quad & f(b) \quad < f(c) \end{aligned}$$

then there is a minimum in the range (a, c). The bracketed range can be reduced by considering a new point x between b and c:



Depending on the function, either (b, c) or (a, x) will be the new bracketed region. This can be continued until region is smaller than a given tolerance.

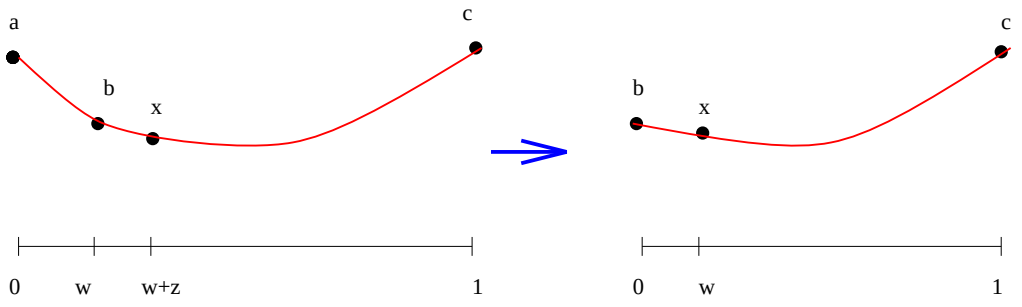
Note: the precision of determining location of $x_{\min}$ is

$$\frac{\delta x_{\min}}{x_{\min}} = \mathcal{O}(\sqrt{\epsilon})$$

where ϵ is the machine precision. This follows from the fact that near the minimum:

$$f(x) \approx f(x_{\min}) + \frac{1}{2} f''(x_{\min}) (x - x_{\min})^2$$

There is an optimal choice to split the bracketed region: golden section



After choosing the next point, the size of the next bracketed region is either $w + z$ or $1 - w$. The optimal strategy would make these equal:

$$w + z = 1 - w \quad \rightarrow \quad z = 1 - 2w$$

But the original value of w should have been chosen in the same way, so

$$w = \frac{z}{1-w} = \frac{1-2w}{1-w} \rightarrow w = \frac{3-\sqrt{5}}{2} = 0.38197...$$

This is called the "golden mean" or "golden section".

Numerical Recipes:

`SUBROUTINE mnbrak(ax,bx,cx,fa,fb,fc,func)`

This routine begins with the initial points `ax` and `bx` and returns a bracketing set of points, `ax,bx,cx` (found by taking successively larger steps downhill). The golden section search can then be performed:

`FUNCTION golden(ax,bx,cx,func,tol,xmin)`

The result is returned in `xmin`.

Parabolic Interpolation and Brent's Method

For smooth functions, the behaviour near the minimum is given by

$$f(x) \approx f(x_{\min}) + \frac{1}{2} f''(x_{\min}) (x - x_{\min})^2$$

If this information is used, convergence will usually be faster than the golden section (but not as robust).

Method:

1. Begin with three points to define a parabola.
2. Next point to evaluate is at the minimum of the parabola.
3. Chose as the next set of three points, the minimum, and the two points on either side.
4. Repeat.

Since the method is less robust, it needs to be combined with a more robust one:

Numerical Recipes:

`FUNCTION brent(ax,bx,cx,func,tol,xmin)`

combines parabolic interpolation and golden section methods.

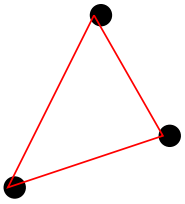
`FUNCTION dbrent(ax,bx,cx,f,df,tol,xmin)`

also uses first derivative as supplied by the user function, `df`. Note that $f'(x)$ is only used to decide which interval (a, b) or (b, c) is used next, on the basis of $f'(b)$.

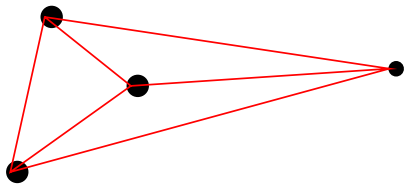
Downhill Simplex Method (multidimensions)

"Not the most efficient but simple and robust."

- "simplex": an object in N dimensions consisting of the lines that connect $N+1$ points
- non-degenerate simplex: none of the lines are collinear, so the simplex encloses a finite N dimensional volume
- examples:



2D - triangle



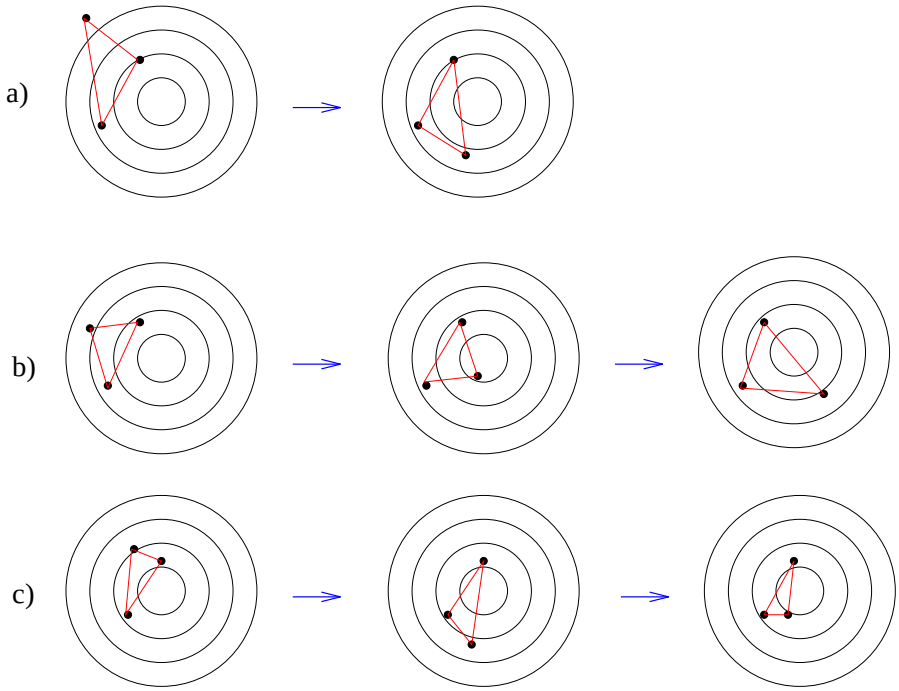
3D - tetrahedron

- If one point is taken as the origin, the N lines from that point define vectors that span the N dimensional space.

Method:

- Start with an initial guess, $\mathbf{P}_0$, and step sizes in each dimension, $\mathbf{e}_i$. This defines a simplex, with the vertices given by $\mathbf{P}_i = \mathbf{P}_0 + \mathbf{e}_i$.
- Perform a series of steps that expand and contract the simplex in the N dimensions.

Simplex transformations:



- a) Reflection: The largest function value is moved through the opposite face of the simplex. The new point is kept if the function value is reduced.
- b) Reflection and expansion: If the function at the new point is smallest of all points, expand.
- c) Reflection and contraction: If the function value has increased, try a smaller step in that direction.
- The simplex eventually encloses a minimum, and the contracts around it, until the function value within the simplex is within some tolerance.

Numerical Recipes:

```
SUBROUTINE amoeba(p,y,mp,np,nd,ftol,funk,iter)
```

input:

`p(1:nd+1,1:nd)`	`nd+1` vertices of the initial simplex
`y(1:nd+1)`	values of funk evaluated initial simplex vertices
`ftol`	fractional function tolerance

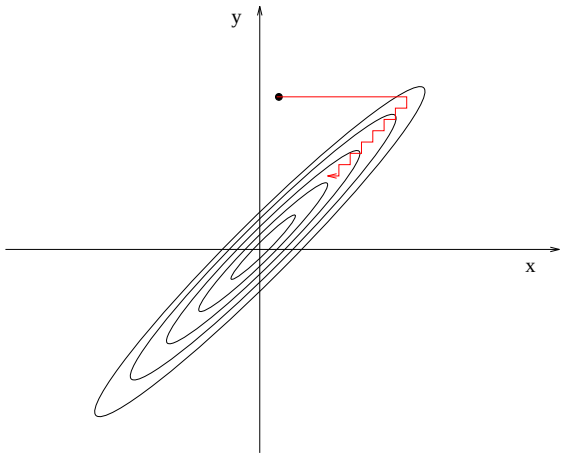
Location of minimum is returned in `p` (a contracted simplex).

Powell's Method

Method:

1. choose a direction
2. find minimum along that direction (using 1D minimization)
3. repeat

It is important to choose the directions carefully. Unit vectors in each dimension can be very inefficient in some cases:



A more efficient approach would be to choose directions such that minimization along one direction does not affect the minimization along the other direction. These are known as "conjugate directions".

Conjugate Directions

Conjugate directions can be found as long as the function is quadratic about the minimum. Otherwise the directions will be only approximately conjugate, but the method improves the rate of convergence in any case.

If the function is nearly quadratic, then it is a good approximation to write

$$\begin{aligned} f(\mathbf{x}) &= f(\mathbf{P}) + \sum_i \frac{\partial f}{\partial x_i}|_{\mathbf{P}}\, x_i + \frac{1}{2}\sum_{i,j} \frac{\partial^2 f}{\partial x_i\, \partial x_j}|_{\mathbf{P}}\, x_i x_j \\ &= c - \mathbf{b}\cdot\mathbf{x} + \frac{1}{2}\mathbf{x}\cdot\mathbf{A}\cdot\mathbf{x} \end{aligned}$$

Hence the gradient of f is approximately,

$$\nabla f = \mathbf{A}\cdot\mathbf{x} - \mathbf{b}$$

The change in the gradient by moving along direction the direction $\delta\mathbf{x}$ is given by,

$$\delta(\nabla f) = \mathbf{A}\cdot\delta\mathbf{x}$$

Suppose that the function is minimized along the direction $\mathbf{u}$. The condition that $\mathbf{u}$ and $\mathbf{v}$ be conjugate directions is that the component of the gradient along the $\mathbf{u}$ direction remain zero when moving along direction $\mathbf{v}$. In other words:

$$0 = \mathbf{u} \cdot \delta(\nabla f) = \mathbf{u} \cdot \mathbf{A} \cdot \mathbf{v}$$

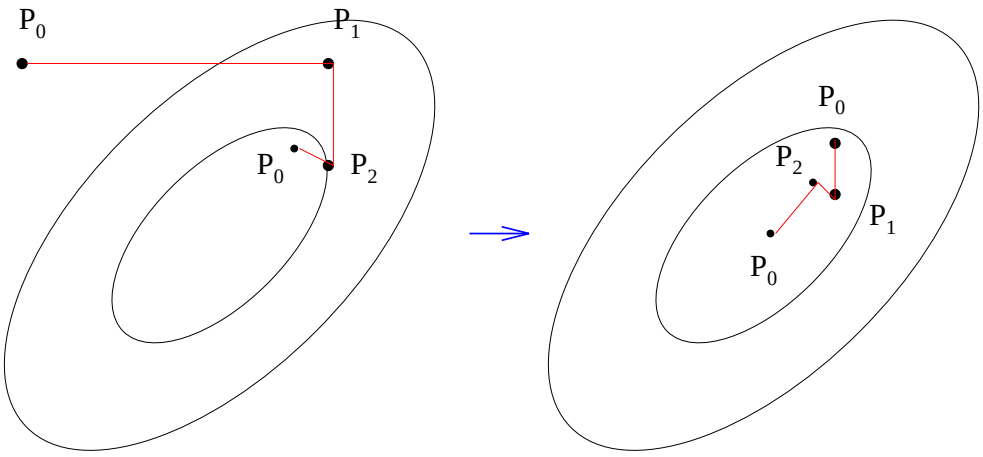
Note in 2D, if $\mathbf{A}$ is diagonal, the contour ellipses are aligned with the x and y directions, and the unit vectors along x and y directions are conjugate.

The challenge is to determine the N conjugate directions. Then, for quadratic functions, the minimum will be found exactly after N 1D minimizations. For most functions the convergence will still be rapid.

Powell's Method

Set the initial set of directions $\mathbf{u}_i$ to be the basis vectors, and pick a starting point $\mathbf{P}_0$. Repeat the following until the minimum is attained:

- Minimize sequentially along each direction $\mathbf{u}_i$.
- Define a new direction; the vector from $\mathbf{P}_0$ to the last point.
- Minimize along that direction, take that point as the new starting point $\mathbf{P}_0$, and replace one of the original directions by this new direction.



For a quadratic function, after N iterations, all the directions will be conjugate, and thus the minimum will be found exactly after $N(N+1)$ 1D minimizations.

There is a problem with this procedure, in that replacing the original directions by $\mathbf{P}_N - \mathbf{P}_0$ can lead to a set of directions that are linearly dependent. As a result, only a subspace of the entire N dimensional space is explored for a minimum.

Powell's Heuristic Method

Improves on previous method by avoiding the problem where directions can become linearly dependent, but gives up property of exact conjugate directions for quadratic problems. The previous method can always be used to polish the result from this method.

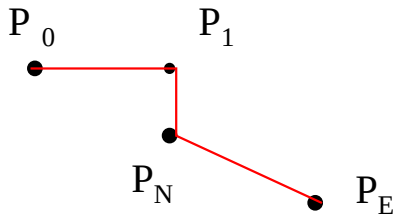
Method:
Follow same procedure, except instead of always replacing an original direction, replace the direction that resulted in the largest decrease in the function. This reduces the chance of it and $\mathbf{P}_N - \mathbf{P}_0$ becoming almost linearly dependent.

Exceptions: do not replace any directions if either

- $f_E \equiv f(2\mathbf{P}_N - \mathbf{P}_0) \geq f(\mathbf{P}_0)$ then since $\mathbf{P}_N - \mathbf{P}_0$ seems to be "played out"; or
- the reduction is not due to a large part on one direction or f'' is large along the direction $\mathbf{P}_N - \mathbf{P}_0$. These conditions can be checked simultaneously by,

$$2(f_0 - 2f_N + f_E)[(f_0 - f_N) - \Delta f]^2 \geq (f_0 - f_E)^2 \Delta f$$

where Δf is the magnitude of the largest decrease along any of the directions.



Numerical Recipes:

```
SUBROUTINE powell(p,xi,n,np,ftol,iter,fret)
```

input:

`p(1:n)`	initial starting point
`xi(1:n,1:n)`	initial directions (columns)
`ftol`	fractional function tolerance

The routine finds the minimum of a user supplier function, `func` and it is returned in `p`.

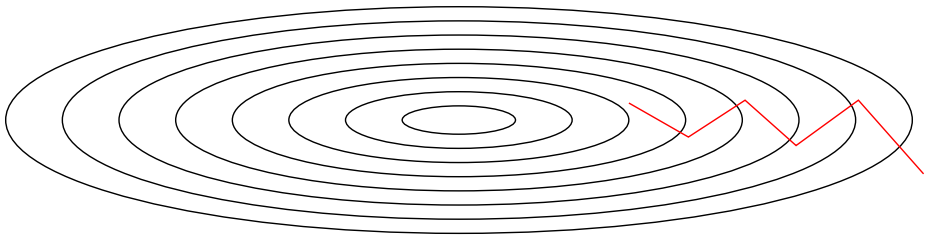
Gradient Methods

If $\nabla f(\mathbf{x})$ is easy to calculate, the speed of convergence can be improved by using both f and ∇f.

Steepest descent usage of ∇f is not a very good algorithm:

- minimize along the direction given by $\nabla f(\mathbf{P}_0)$
- move to this new minimum
- repeat

Even for a quadratic function this can lead to many small steps being taken, because each direction must be orthogonal to the previous one:



A more efficient method would have the directions be conjugate to one another:

$\rightarrow$ **Conjugate Gradient Methods**

By using the gradients, conjugate directions can be found much more elegantly than Powell's method.

- Start at point $\mathbf{x}_0$.
- Minimize along steepest descent at $\mathbf{x}_0$, giving a new point $\mathbf{x}_1$.
- The next direction $\mathbf{d}$ needs to be conjugate to the previous direction of movement, $\mathbf{x}_1 - \mathbf{x}_0$.

$$\mathbf{d} \cdot \mathbf{A} \cdot (\mathbf{x}_1 - \mathbf{x}_0) = 0$$

- Fortunately $\mathbf{A}$ does not need to be calculated:

$$\begin{aligned} \nabla f(\mathbf{x}) &= \mathbf{A} \cdot \mathbf{x} - \mathbf{b} \qquad \text{and so,} \\ \mathbf{d} \cdot \mathbf{A} \cdot (\mathbf{x}_1 - \mathbf{x}_0) &= \mathbf{d} \cdot (\nabla f(\mathbf{x}_1) - \nabla f(\mathbf{x}_0)) = 0 \end{aligned}$$

- The next direction $\mathbf{d}$ is some combination of the two gradient vectors:

$$\mathbf{d} = \nabla f(\mathbf{x}_1) + \alpha \nabla f(\mathbf{x}_0)$$

- solve for α, using $\nabla f(\mathbf{x}_1) \cdot \nabla f(\mathbf{x}_0) = 0$:

$$\alpha = \frac{(\nabla f(\mathbf{x}_1))^2}{(\nabla f(\mathbf{x}_0))^2}$$

- Continue the process.

Numerical Recipes:

```
SUBROUTINE frprmn(p,n,ftol,iter,fret)
```

where `p(1:n)` is the starting point, and the user supplies the functions `func` and `dfunc`.

Variable Metric Methods

Competitive with conjugate gradient method.

Basic idea is that of Newton's method for finding roots:

$$\nabla f(\mathbf{x}) = \nabla f(\mathbf{x}_0) + \mathbf{A} \cdot (\mathbf{x} - \mathbf{x}_0)$$

At the minimum, $\nabla f(\mathbf{x}_{\min}) = 0$, so

$$\mathbf{x}_{\min} = \mathbf{x}_0 - \mathbf{A}^{-1} \cdot \nabla f(\mathbf{x}_0)$$

The complication, arises in that $\mathbf{A}$ is not known, and instead an evolving approximation for $\mathbf{A}$ is used instead. The method of successive improvements to $\mathbf{A}$ is not straight forward (see text).

Numerical Recipes:

```
SUBROUTINE dfpmin(p,n,gtol,iter,fret,func,dfunc)
```

`p(1:n)` is the starting position. The programs returns once the magnitude of the gradient is reduced to `gtol`.

Linear Programming (Optimization)

General problem is to maximize:

$$z = a_{01}x_1 + a_{02}x_2 + ... + a_{0N}x_N$$

subject to the N primary constraints,

$$x_1 \geq 0, \quad x_2 \geq 0, \quad ... \quad x_N \geq 0$$

and M additional constraints,

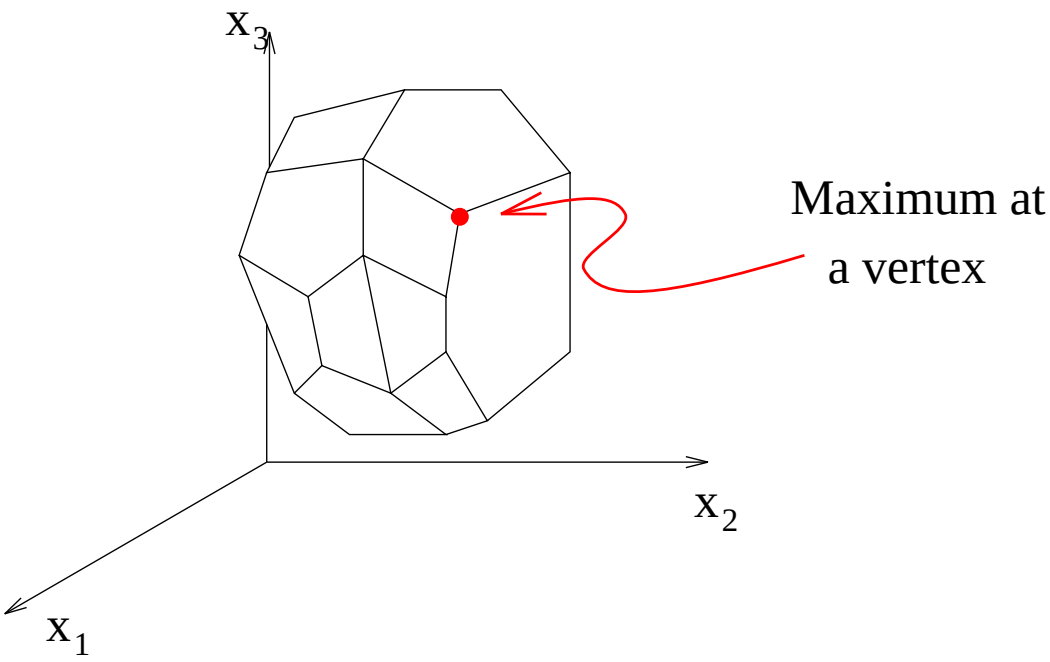
$$\sum_{\ell=1}^{N} a_{i\ell}x_\ell \quad \leq \quad b_i \geq 0 \quad i = 1, ..., m_1$$

$$\sum_{\ell=1}^{N} a_{j\ell}x_\ell \quad \geq \quad b_j \geq 0 \quad j = m_1 + 1, ..., m_1 + m_2$$

$$\sum_{\ell=1}^{N} a_{k\ell}x_\ell \quad = \quad b_k \geq 0 \quad k = m_1 + m_2 + 1, ..., M$$

Problems of this sort are common in accounting where the concepts of negative dollars, negative widgets, etc. are meaningless.

Starting from N dimensional space, the inequality constraints define boundary planes, such that the range of allowed space is constrained within a convex polyhedron. Each equality constraint reduces the dimensionality of the polyhedron by one. Since z is a linear function, the maximum of z must occur at a vertex.



There are a total of $N + M$ constraints. The problem of finding the optimal position is equivalent to finding which N of the $N + M$ constraints, all treated as *equality* constraints, define the position of the vertex.

The brute force method is to try each of the $\binom{N+M}{N}$ possibilities, each time solving the set of N linear equations. This could take forever for sufficiently complicated problems.

A more optimal method is to reformulate the problem in "restricted normal form", and then apply the simplex method (not related to the multidimensional minimization method).

- normal form: only equality constraints appear
- restricted form: each equality constraint has a variable unique to that constraint and it has a positive coefficient

Simplex method

An example:

$$z = 2x_2 - 4x_3$$

subject to the constraints

$$\begin{aligned} x_1 + 6x_2 - x_3 &= 2 \\ -3x_2 + 4x_3 + x_4 &= 8 \end{aligned}$$

Since there are 4 variables, and only 2 additional constraints, the solution must have at least two of the variables being zero.

- First step is to rewrite the constraints so that the unique variables are on the LHS:

$$\begin{aligned} x_1 &= 2 - 6x_2 + x_3 \\ x_4 &= 8 + 3x_2 - 4x_3 \end{aligned}$$

- One can easily find a vertex in the 4D space (not necessarily the best one) by setting $x_2 = x_3 = 0$:

$$\rightarrow x_1 = 2, \quad x_4 = 8, \quad z = 0$$

- To increase z, it is clear that x_2 should be increased. How far can x_2 increase while keeping the LHS variables ≥ 0?

- There is no problem for x_4 since its coefficient is positive. (If all coefficients were positive, there would be no upper limit to z).
- If there are several constraint equations with negative coefficients, the critical one is the one with the smallest value:

  (constant coefficient)/(coefficient of x_2)
- Rewrite the critical constraint equation so that x_2 is on LHS.

$$\rightarrow x_2 = \frac{1}{3} - \frac{1}{6}x_1 + \frac{1}{6}x_3$$

- Rewrite z in terms of the RHS variables only.
- Repeat until all the coefficients of expression for z are ≤ 0. Solution has RHS variables $= 0$.

To put a general problem into normal form, replace inequality constraints by adding extra (non-negative) variables:

$$x_1 + 2x_2 \geq 3 \quad \rightarrow \quad x_1 + 2x_2 - y_1 = 3$$
$$x_2 + 3x_3 \leq 4 \quad \rightarrow \quad x_2 + 3x_3 + y_2 = 4$$

At the end, the solutions for y_1 and y_2 are ignored.

To put into restricted normal form, introduce more variables:

$$\begin{aligned} z_1 &= 3 - x_1 - 2x_2 + y_1 \\ z_2 &= 4 - x_2 - 3x_3 - y_2 \end{aligned}$$

And solve the new problem, maximizing

$$z' = -z_1 - z_2 = -7 + x_1 + 3x_2 + 3x_3 - y_1 + y_2$$

with all z_1 and z_2 constrained to be ≥ 0 as usual. Since the solution to this problem has $z_1 = z_2 = 0$, the simplex procedure will result in z_1 and z_2 becoming RHS variables, which can be set to zero. This leaves the original problem, but set up in restricted normal form.

Numerical Recipes subroutine:

```
simplx(a,m,n,mp,np,m1,m2,m3,icase,izrov,iposv)
```

The input variables follow the naming convention introduced above. Note that for internal calculations the physical dimension of `a` must be `a(mp,np)` with `mp`$\geq$ `m+2` and `np`$\geq$ `n+1`.
`icase` specifies if a solution is found
`iposv(1:M)` and `izrov(1:N)` are pointers to the solution stored in `a` (see text).

Simulated Annealing Methods

An analogy is made with freezing:

- slowly cooled systems find the global minimum energy state (a crystal state for example)
- quickly cooled systems do not, instead they find a local minimum (an amorphous state)

Algorithms presented so far are of the "quickly cooling" type: converge to the nearby solution as fast as possible.

Nature has a different approach:

- The probability that a system at temperature T is in a state of energy E is given by,

$$p(E) \sim e^{-E/kT}$$

- Even at low temperatures there is some chance to be in a high energy state.
- This allows the system to get out of local energy minimums (as long as enough time is allowed).

Metropolis Algorithm

To simulate a thermodynamic system, consider various configurations. Define the probability to change from $1 \to 2$ to be

$$p = \min\left(1, e^{-(E_2 - E_1)/kT}\right)$$

In other words, always take a downhill step, and sometimes take an uphill step.

Can be applied to non-thermodynamic systems as well. One needs to define

1. a set of possible configurations
2. a method to randomly modify the configurations
3. a function (E) to minimize as the goal of the problem
4. a control parameter (T) and an annealing schedule (how to lower T).

Example: Traveling Salesman (minimize total trip distance)

1. Number cities, $i = 1, ..., N$, each with coordinate (x_i, y_i). A configuration consists of a permutation of the numbers $1, ..., N$ which specifies the order that the cities are visited.
2. Modify the permutation as follows
   a) reverse order of 2 adjacent numbers
   b) move 2 adjacent numbers to random location

3. $E = \sum L_i$, or some other penalty function could be included.

4. Set $k = 1$ so that

$$p = \min\left(1, e^{-(E_2 - E_1)/T}\right)$$

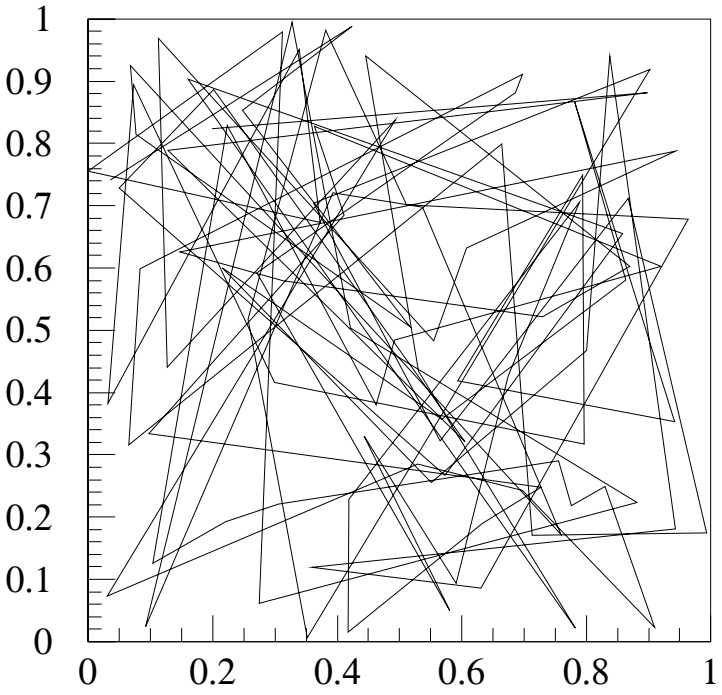
and experiment with a few trial values to get the scale of ΔE. Choose $T \gg \Delta E$, so initially all configurations are sampled with little penalty. Do $100N$ configurations or $10N$ successful transitions then reduce T by 10%. Repeat until E no longer decreases substantially.

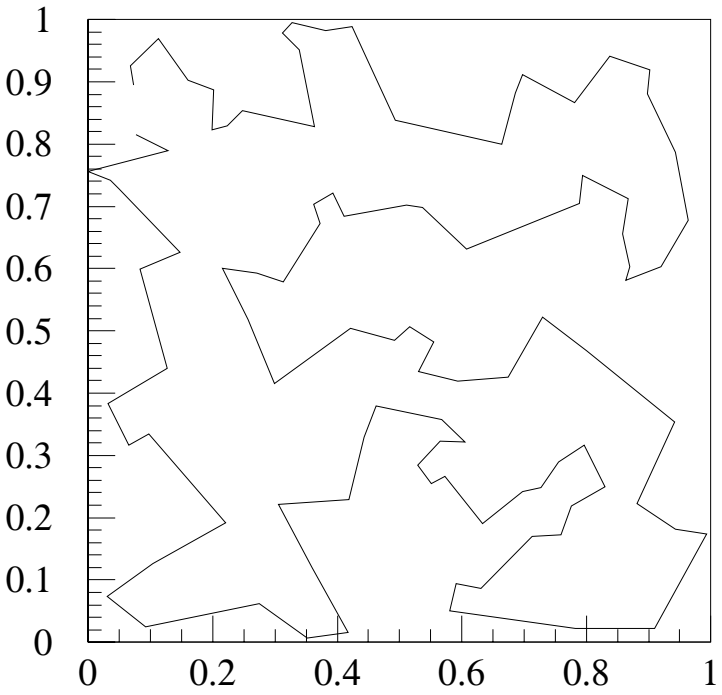
Numerical Recipes:

```
SUBROUTINE anneal(x,y,iorder,ncity)
```

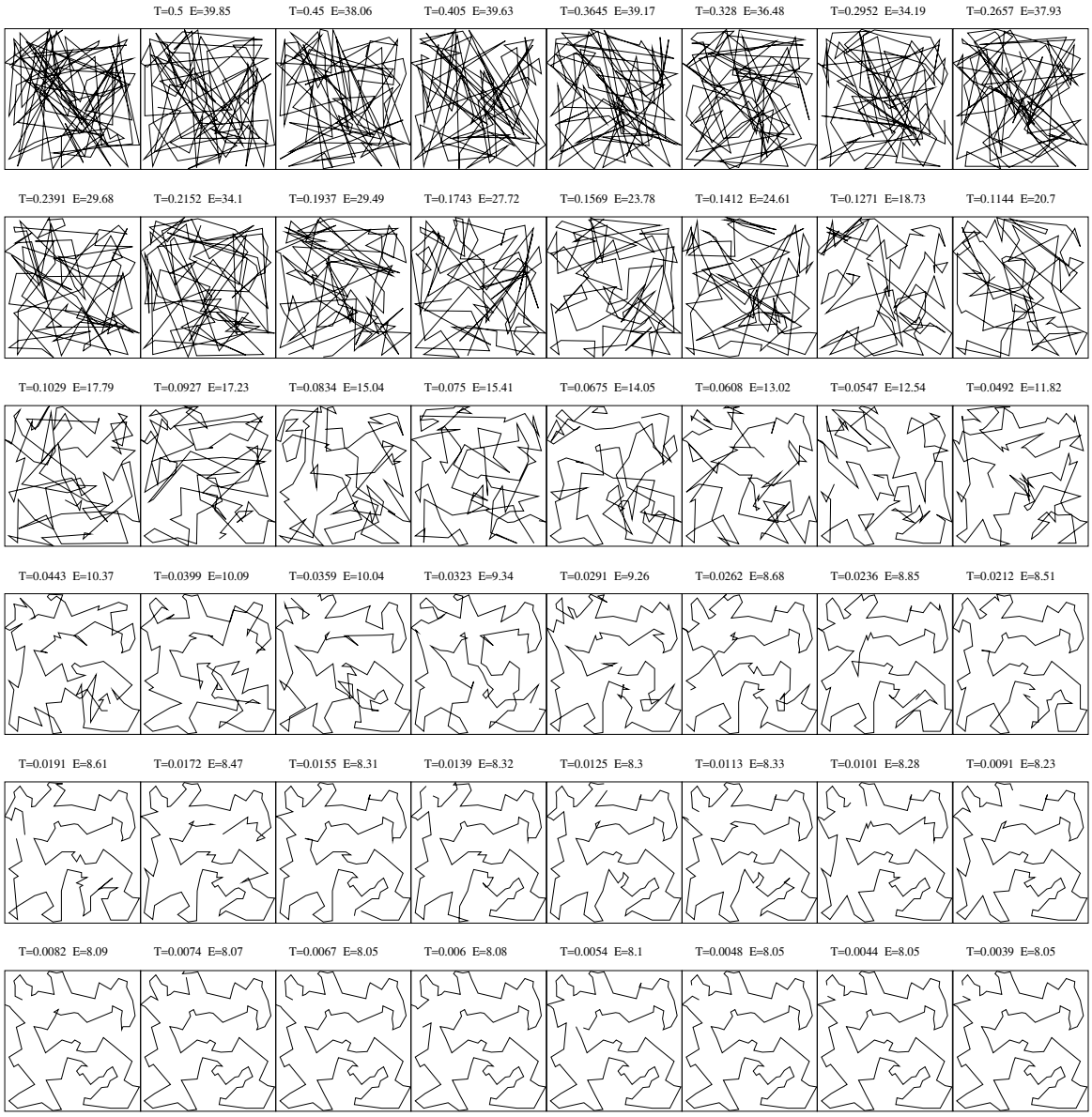
The best route is specified by the array `iorder(1:ncity)`.

Example of 100 randomly placed cities:





How the solution was found:



Ordinary Differential Equations

Any ODE can be rewritten in terms of a set of first-order ODE's. For example,

$$\frac{d^2y}{dx^2} + q(x)\frac{dy}{dx} = r(x)$$

can be written as two equations,

$$\begin{aligned} \frac{dy}{dx} &= z(x) \\ \frac{dz}{dx} + q(x)\,z(x) &= r(x) \quad . \end{aligned}$$

The general problem therefore can be written in terms of N first order equations of the form,

$$\frac{dy_i}{dx} = f_i(x, y_1, ..., y_N) \quad i = 1, ..., N \quad .$$

In order to solve a specific problem, boundary conditions need to be specified, usually in the form of initial conditions, $y_i(x_0)$.

To deal with problems with boundary conditions given at more than one value of x, see text (two-point boundary value problems).

Eulers method

Inaccurate and can be unstable: should not be used!

Simplest method of all, just rewrite the differential equation in terms of finite differences:

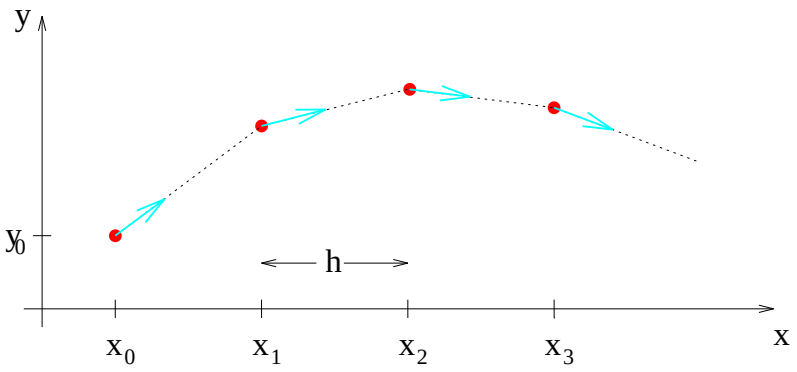
$$\begin{aligned} \frac{dy}{dx} &= f(x,y) \\ \frac{\Delta y}{\Delta x} &= f(x,y) \\ \Delta y &= \Delta x\, f(x,y) \end{aligned}$$

This leads to the recursion relation,

$$y_{n+1} = y_n + h\, f(x_n, y_n) + \mathcal{O}(h^2)$$

where, $y_n = y(x_n)$ and $x_n = x_{n-1} + h$.

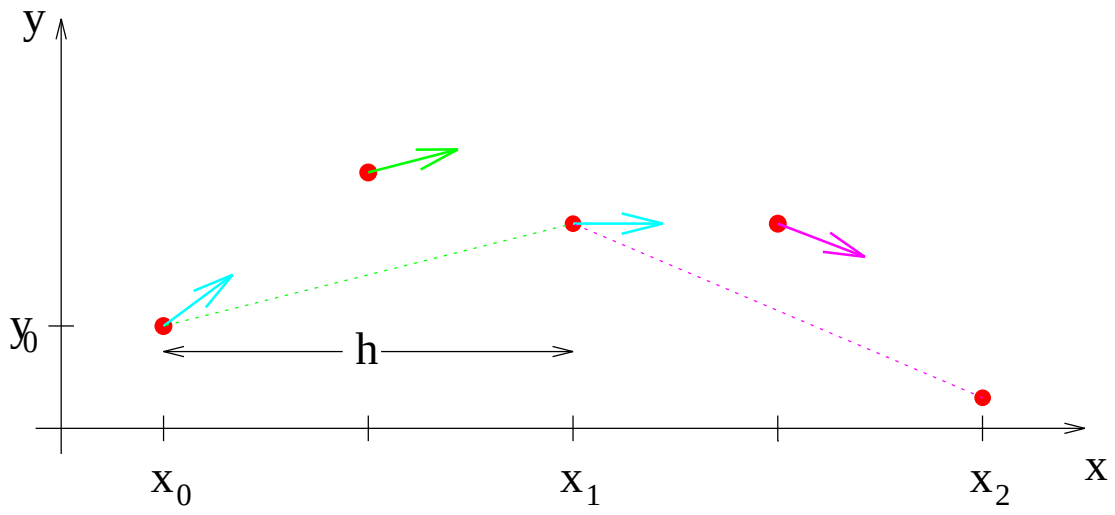
By specifying the initial conditions, x_0, y_0, the solution is found as shown below.



Runge-Kutta Method

"Robust, but inefficient and only moderately accurate"

Instead of using the derivative at the start of the interval, the Runge-Kutta method uses the derivative evaluated at the midpoint of the interval. This reduces the error in the method.



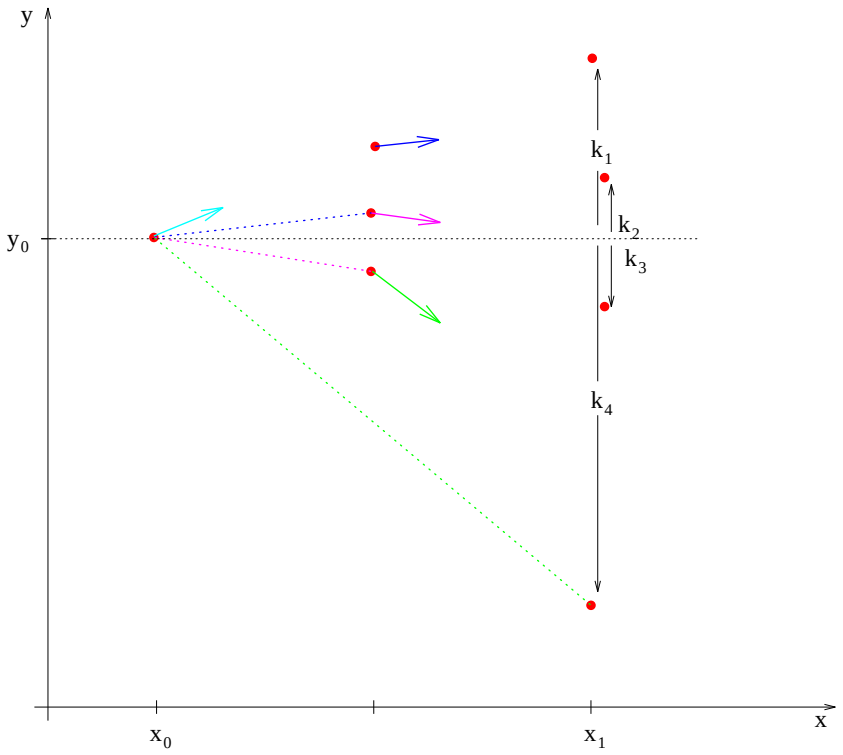
The algorithm for this method is:

$$\begin{aligned} k_1 &= h\, f(x_n, y_n) \\ k_2 &= h\, f(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1) \\ y_{n+1} &= y_n + k_2 + \mathcal{O}(h^3) \end{aligned}$$

and is called the second order Runge-Kutta formula.

Most often used is the 4^{th} order Runge-Kutta formula:

$$
\begin{aligned}
k_1 &= h\, f(x_n, y_n) \\
k_2 &= h\, f(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1) \\
k_3 &= h\, f(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_2) \\
k_4 &= h\, f(x_n + h, y_n + k_3) \\
y_{n+1} &= y_n + \frac{1}{6}k_1 + \frac{1}{3}k_2 + \frac{1}{3}k_3 + \frac{1}{6}k_4 + \mathcal{O}(h^5)
\end{aligned}
$$



Adaptive Stepsize Control

To improve accuracy and efficiency, h should not be kept constant, but rather vary, according to the nature of the solution.

- when solution is smoothly changing, h should be large to improve efficiency
- when solution is rapidly varying, h needs to be small to ensure reasonable accuracy

Step Doubling

Compare the result of a step of size $2h$ with that of size h. The difference in the two results can be used to estimate the error in the approach. Adjust h to keep the error in a reasonable range (not too large and not too small).

Embedded Runge-Kutta formulas

This is another technique to estimate the error, but requires fewer function calls. The 5^{th} order Runge-Kutta formula requires 6 function calls, but another combination of the 6 function values gives a 4^{th} order Runge-Kutta formula.

The error estimate $\Delta \sim h^5$. If the desired accuracy for one step is Δ_0 then the appropriate size to use for the next step is

$$h_0 = h \left| \frac{\Delta_0}{\Delta} \right|^{0.2}$$

If the problem involves a set of ODE's, then the largest value of Δ should be used. Since the errors can accumulate (all with the same sign), the tolerable error should scale with the step size, ie $\Delta_0 = \epsilon\, h \frac{dy}{dx}$.

Numerical Recipes supplies the general ODE integrator:

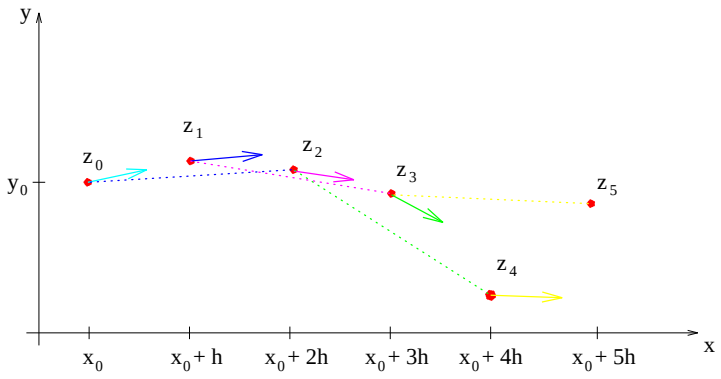
```
FUNCTION odeint(ystart,nvar,x1,x2,eps,
h1,hmin,nok,nbad,derivs,choose)
```

User supplies routine `derivs(x,y,dydx)`, which returns `dydx(1:nvar)`. The starting values, $y(x_1)$, are given by `ystart(1:nvar)`, and x_2 is the final point. The intermediate results are stored in `common /path/`. The final argument specifies the stepping routine. Use `rkqs` for the fifth order embedded Runge-Kutta formula.

Modified Midpoint Method

A large step of size H can be broken into n equal substeps each of size h:

$$\begin{aligned} z_0 &= y_0 \\ z_1 &= z_0 + h\, f(x, z_0) \\ z_2 &= z_0 + 2h\, f(x+h, z_1) \\ z_3 &= z_1 + 2h\, f(x+2h, z_2) \\ z_n &= z_{n-2} + 2h\, f(x+(n-1)h, z_{n-1}) \end{aligned}$$



The estimate of the solution at $(x_0 + H)$ is given by,

$$y_n = \frac{1}{2}\left[z_n + (z_{n-1} + h\, f(x+H, z_n))\right]$$

and the error in this estimate is even in powers of h:

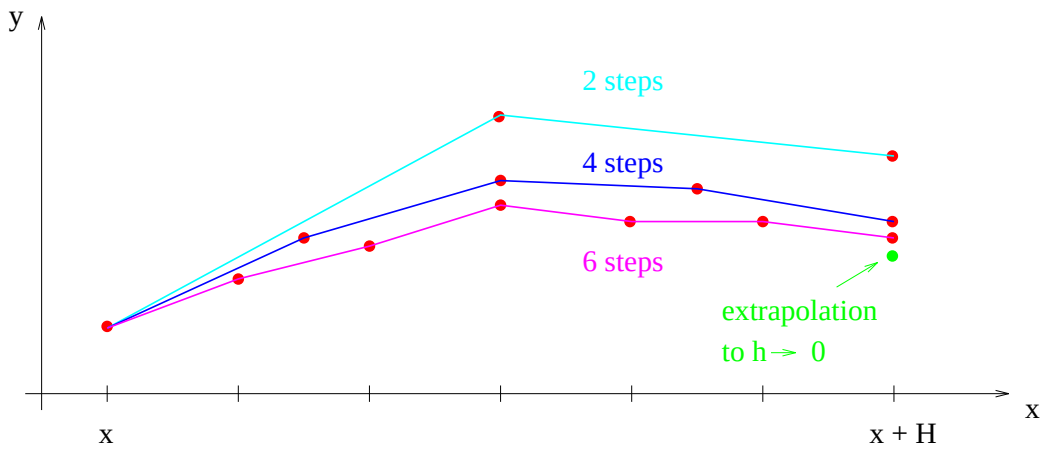
$$y(x+H) = y_n + \alpha_1 h^2 + \alpha_2 h^4$$

Bulirsch-Stoer Method

Best method for smooth functions, otherwise use Runge-Kutta with adaptive step size.

Method:

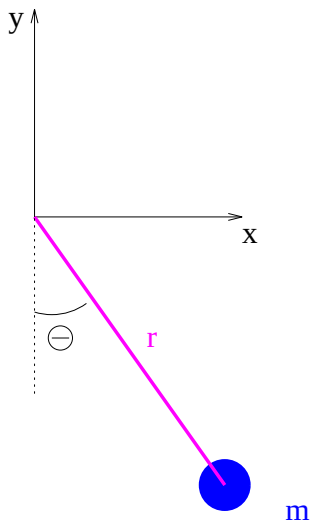
- Use midpoint method with $n = 2, 4, 6, 8, ...$
- Extrapolate result y_n for $h \to 0$ using polynomial. The error estimate from the polynomial extrapolation is used to decide when n is large enough.
- Reduce H if adequate precision is not attained after $n_{\max}$ iterations
- Increase H if precision is better than that requested.



Numerical Recipes routine, `odeint`, can be used to drive the Bulirsch-Stoer algorithm by using the routine name, `bsstep` as the last argument.

Exercise 5

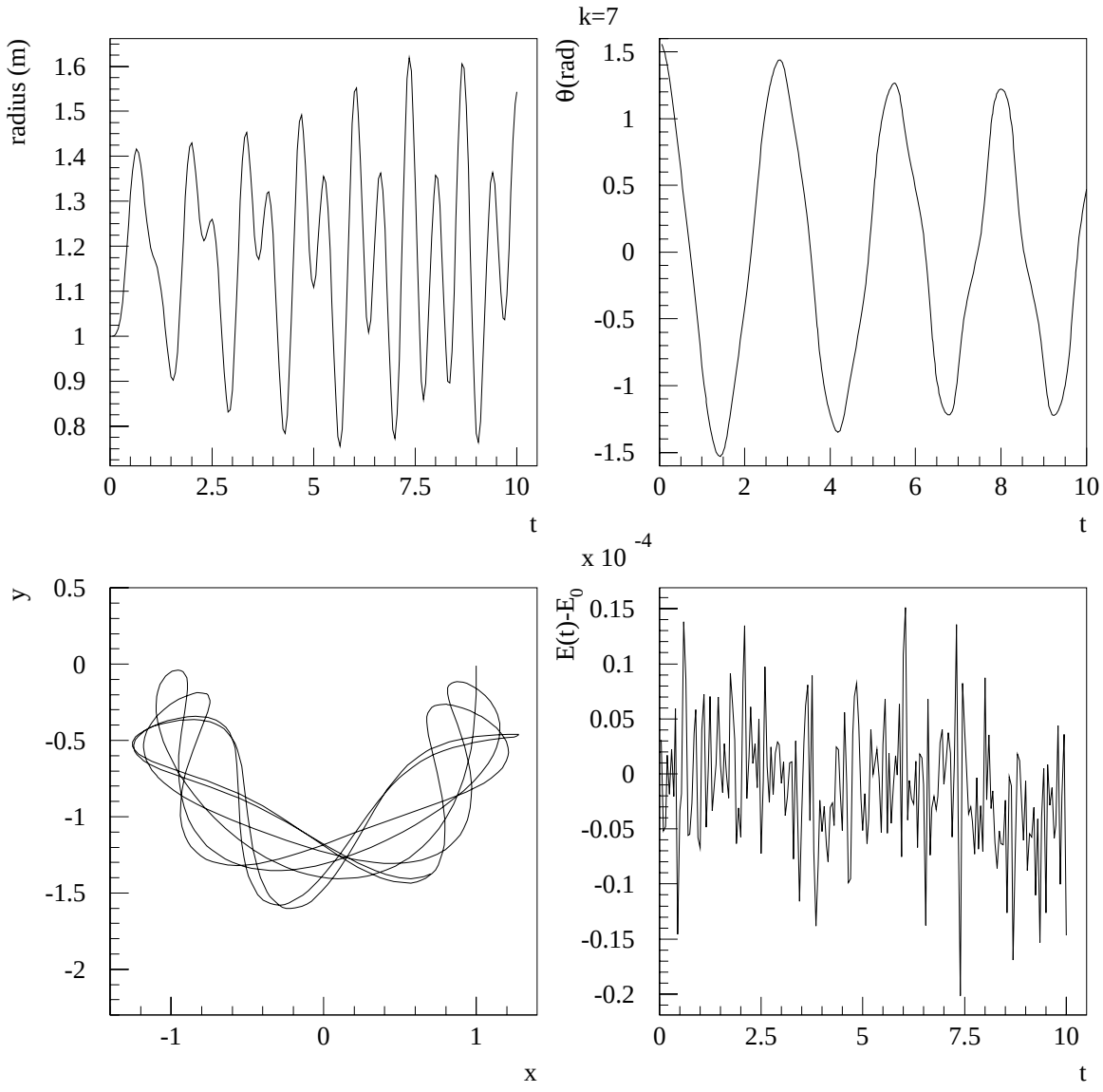
A pendulum consists of a bob of mass m connected to a rod of natural length L, which acts like a spring with spring constant k. The pendulum is started from rest in a horizontal position and let go. Use the Burlisch-Stoer method with the following set of parameters: $m = 0.1$ kg, $L = 1$ m, $k = 6$ N/m. Repeat for $k = 1000$ N/m. In each case, plot the radius r, θ and total energy as a function of time, and plot the path of the bob.



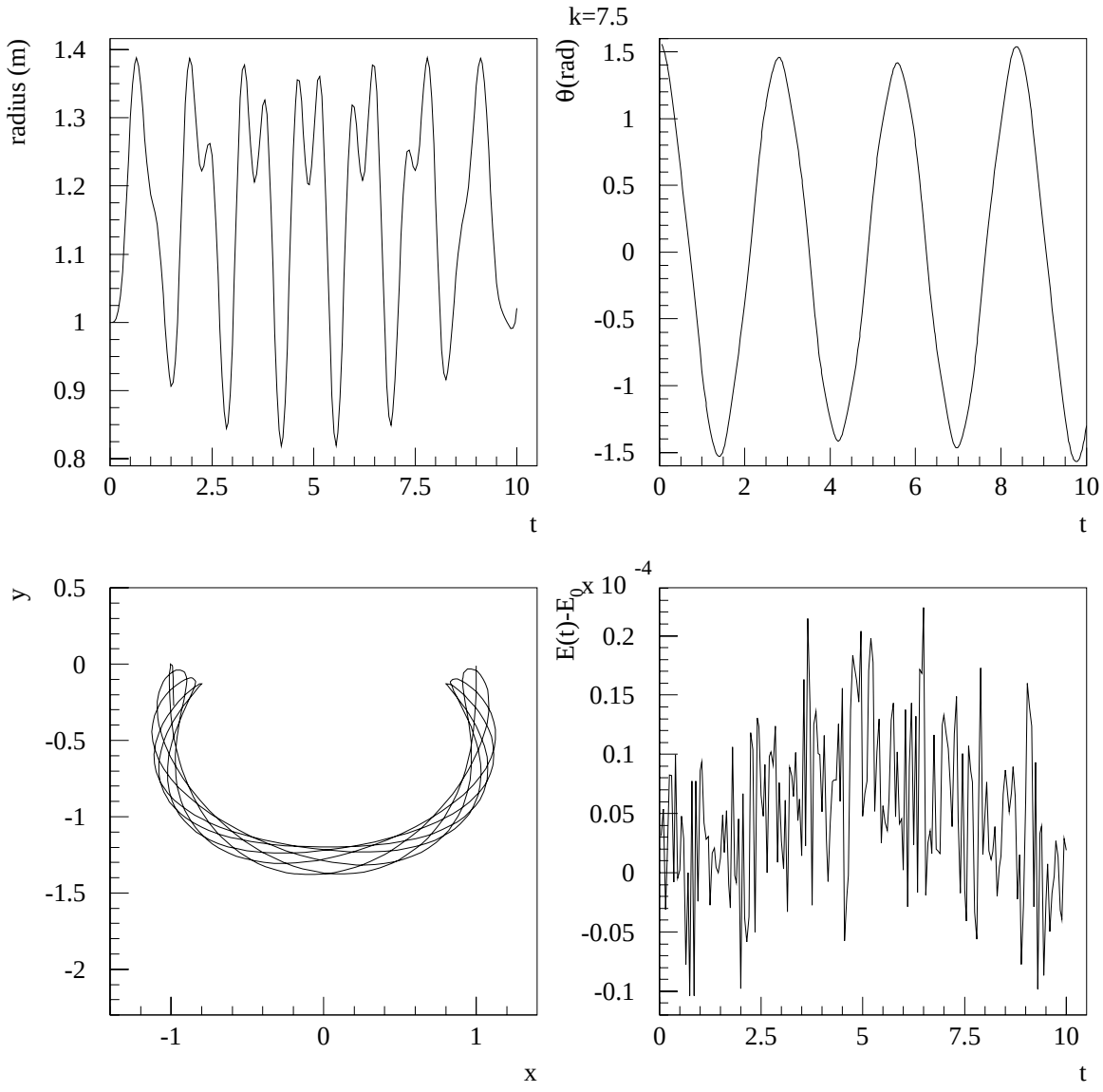
The differential equations of motion for this system are:

$$\begin{aligned} \ddot{r} - r\dot{\theta}^2 &= -\frac{k}{m}(r - L) + g\cos\theta \\ r\ddot{\theta} + 2\dot{r}\dot{\theta} &= -g\sin\theta \end{aligned}$$

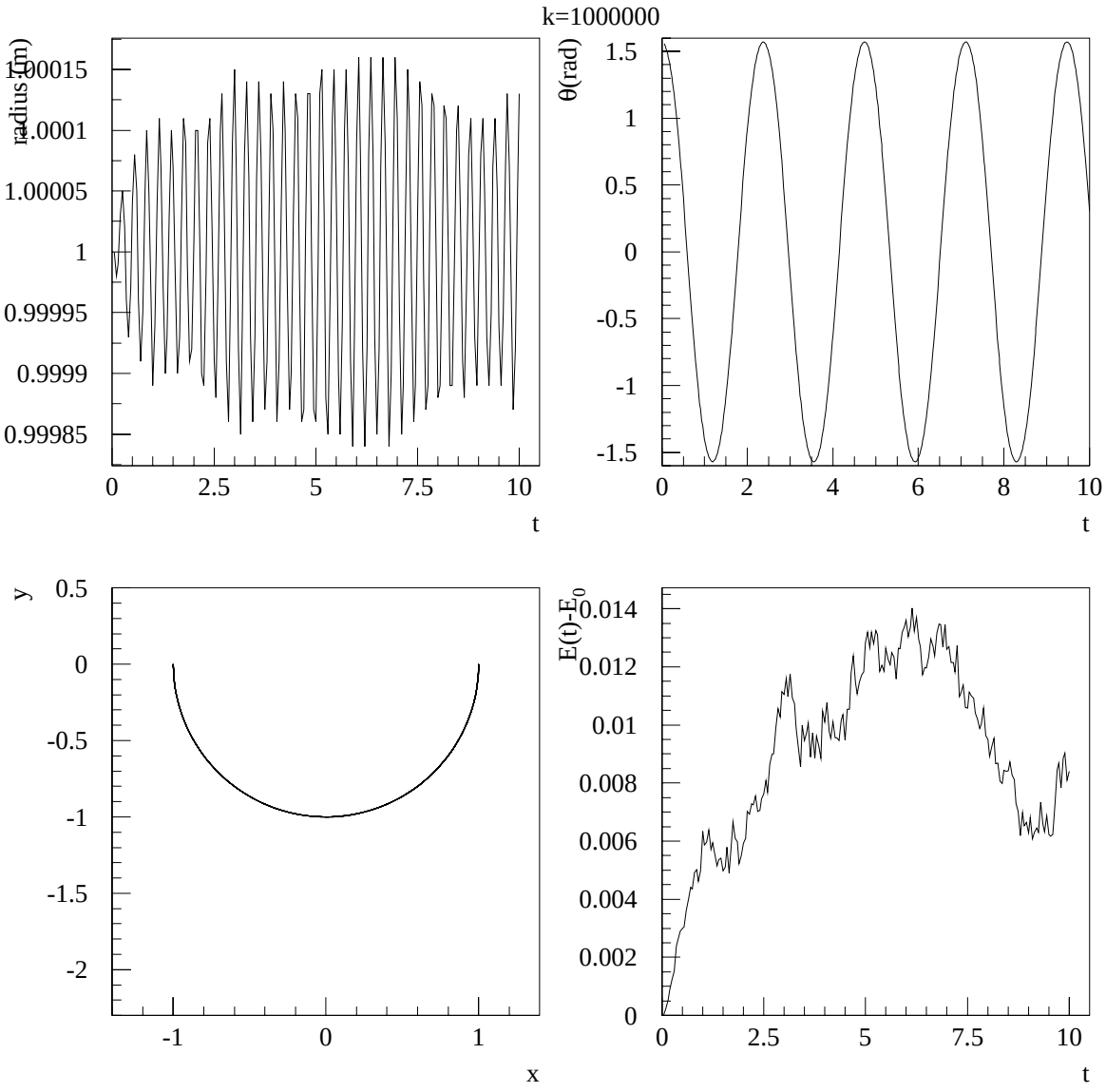
Result for k=7:



Result for k=7.5:



Result for k=1,000,000:



Partial Differential Equations

Numerical methods for solving PDE is a vast and complex subject area. This review only scratches the surface!

PDE's pose two classes of problems:

Initial Value Problems:

- could be a hyperbolic equation, such as the wave equation:

$$\frac{\partial^2 u}{\partial t^2} = v^2 \frac{\partial^2 u}{\partial x^2}$$

- or a parabolic equation, such as the diffusion equation:

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left(D\frac{\partial u}{\partial x}\right)$$

- given $u(x, t = 0)$, the problem is to find $u(x, t)$.

Boundary Value Problems:

- elliptic equations, such as Laplace's equation:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$$

- Given $u(x, y)$ on the boundary, the problem is to find $u(x, y)$ elsewhere.

Flux-conservative IVP

A flux conservative initial value problem in 1D:

$$\frac{\partial u}{\partial t} = -\frac{\partial F(u)}{\partial x}$$

The 1D wave equation with constant velocity:

$$\frac{\partial^2 u}{\partial t^2} = v^2 \frac{\partial^2 u}{\partial x^2}$$

can be rewritten as two first order PDEs of this form:

$$\frac{\partial s}{\partial t} = v\frac{\partial r}{\partial x} \quad \frac{\partial r}{\partial t} = v\frac{\partial s}{\partial x}$$

where $s \equiv \partial u/\partial t$ and $r \equiv v\partial u/\partial x$. Letting,

$$\mathbf{a} = \begin{pmatrix} s \\ r \end{pmatrix} \quad \text{and} \quad \mathbf{B} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

allows the equation to be written in the form above with u replaced by the vector $\mathbf{a}$ and $F(u)$ by $-v\mathbf{Ba}$.

Instead, consider the scalar form of this equation

$$\frac{\partial u}{\partial t} = -v\frac{\partial u}{\partial x}$$

The analytical solution to this problem is a wave propagating in the positive x direction:

$$u = f(x - vt)$$

The numerical solution to this problem is not as simple!

FTCS Method

The Forward Time Centered Space method...

Put time and space onto a grid:

$$\begin{aligned} x_j &= x_0 + j\,\Delta x \quad j = 0, 1, ..., J \\ t_n &= t_0 + n\,\Delta t \quad n = 0, 1, ..., N \end{aligned}$$

and denote $u(t_n, x_j)$ as u_j^n. To solve the advection equation,

$$\frac{\partial u}{\partial t} = -v\frac{\partial u}{\partial x}$$

write the derivatives as finite differences, the time using forward Euler differencing, and the space derivative centred:

$$\begin{aligned} \frac{\partial u_j^n}{\partial t} &= \frac{u_j^{n+1} - u_j^n}{\Delta t} + \mathcal{O}(\Delta t) \\ \frac{\partial u_j^n}{\partial x} &= \frac{u_{j+1}^n - u_{j-1}^n}{2\Delta x} + \mathcal{O}(\Delta x^2) \end{aligned}$$

Then the advection equation becomes,

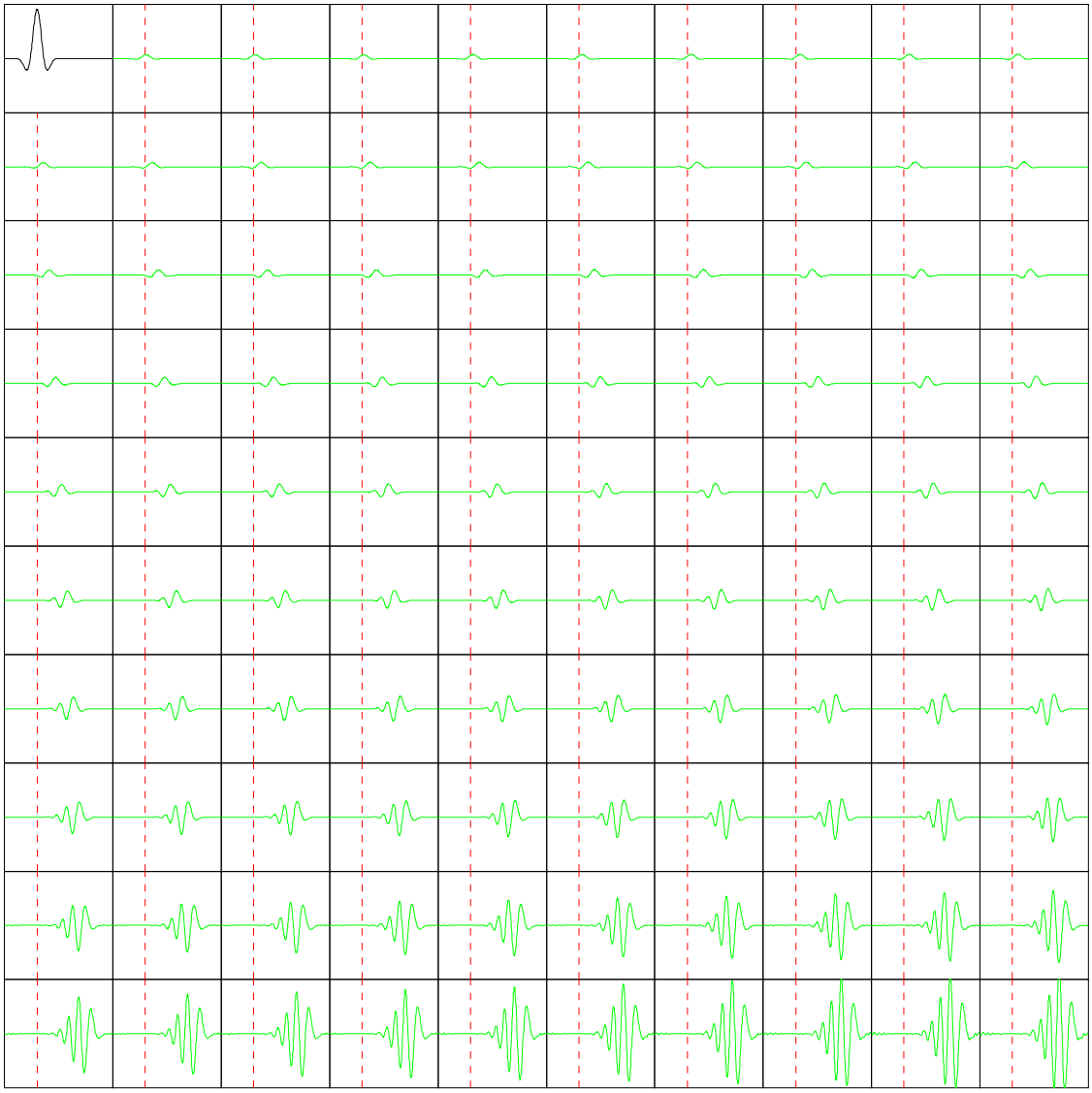
$$u_j^{n+1} = u_j^n - \frac{v\,\Delta t}{2\Delta x}(u_{j+1}^n - u_{j-1}^n) \quad .$$

Given the initial values, u_j^0 for all j, subsequent values, u_j^n can be determined by this equation.

In PAW, the formula is easily handled,

$$\texttt{sigma} \quad \texttt{u} = \texttt{u} + [\texttt{c}] * (\texttt{ls(u,1)} - \texttt{ls(u,-1)})$$

Unfortunately the FTCS method is unstable for the advection equation. The following is an example with $\frac{v\Delta t}{\Delta x} = 0.6$. Each box represents a new time bin.



Lax Method

A simple modification to the FTCS method, improves the stability of the method. Replace u_j^n by its average, $\frac{1}{2}(u_{j+1}^n + u_{j-1}^n)$, so the recurrence relation is now

$$u_j^{n+1} = \frac{1}{2}(u_{j+1}^n + u_{j-1}^n) - \frac{v\,\Delta t}{2\Delta x}(u_{j+1}^n - u_{j-1}^n) \quad .$$

This method is stable for $\frac{v\,\Delta t}{\Delta x} \leq 1$, but for $\frac{v\,\Delta t}{\Delta x} < 1$ the amplitude diminishes. For $\frac{v\,\Delta t}{\Delta x} = 1$ the solution is exact,

$$u_j^{n+1} = u_{j-1}^n$$

Note that the Lax equation can be rewritten as,

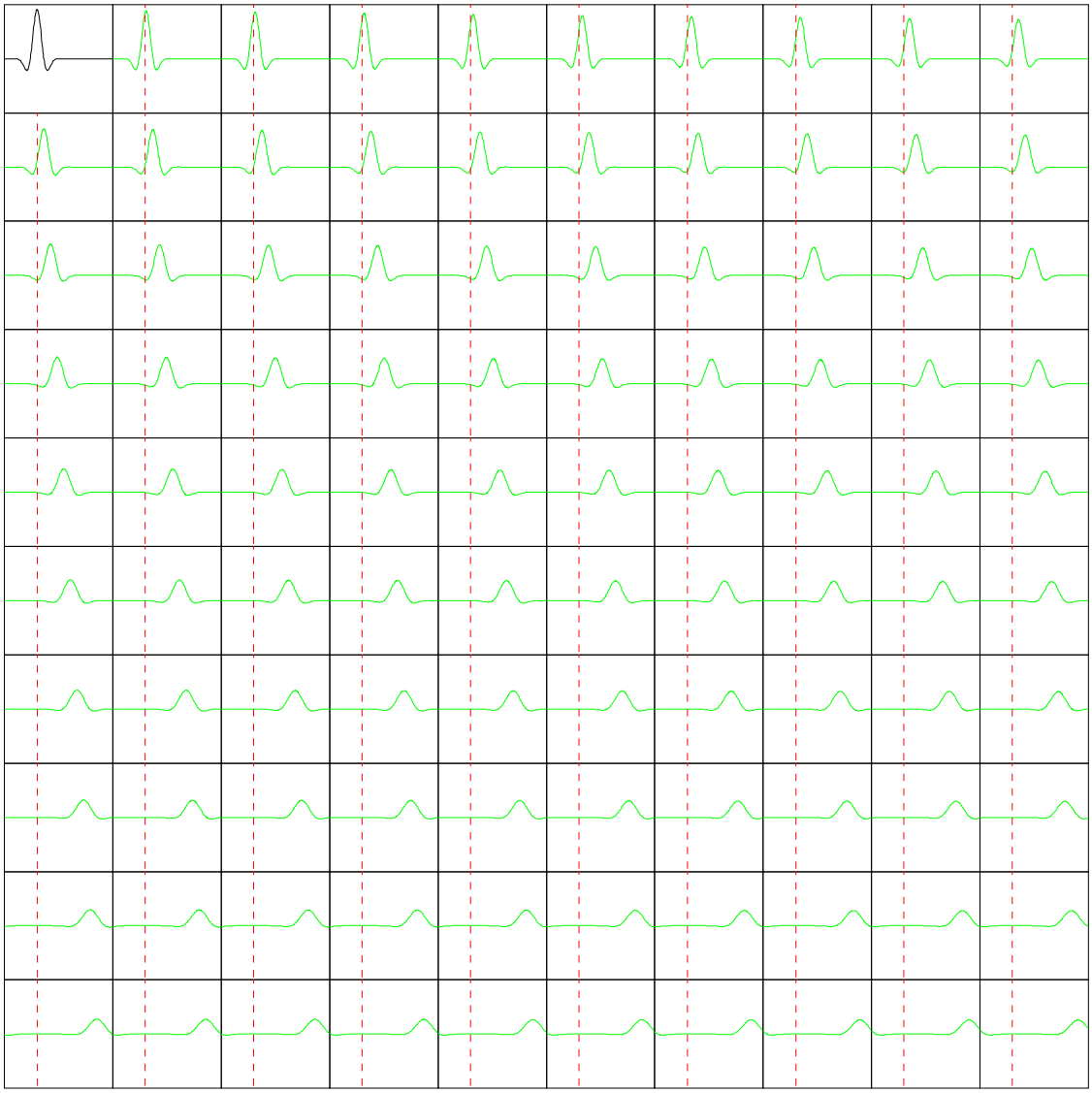
$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = -v\left(\frac{u_{j+1}^n - u_{j-1}^n}{2\Delta x}\right) + \frac{1}{2}\left(\frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{\Delta t}\right)$$

which is the FTCS representation of,

$$\frac{\partial u}{\partial t} = -v\frac{\partial u}{\partial x} + \frac{(\Delta x)^2}{2\Delta t}\frac{\partial^2 u}{\partial x^2} \quad .$$

The new term is a dissipative term, said to add "numerical viscosity" to the equation.

Example of a solution to the advection equation using Lax method, with $\frac{v\Delta t}{\Delta x} = 0.6$. Each box represents a new time bin.



Lax-Wendroff Scheme

To improve the difference equation to second order in time, consider the Taylor expansion of the solution,

$$u(x,t+\Delta t) = u(x,t) + \Delta t \left(\frac{\partial u}{\partial t}\right) + \frac{1}{2}\Delta t^2 \left(\frac{\partial^2 u}{\partial t^2}\right) + \mathcal{O}(\Delta t^3)$$

The second term is easily represented using the original PDE, which can be written in a more general form,

$$\frac{\partial u}{\partial t} = -\frac{\partial F(u)}{\partial x}$$

where $F(u) = vu$ for the advection equation. The second partial derivative can be written,

$$\begin{aligned} \frac{\partial^2 u}{\partial t^2} &= -\frac{\partial}{\partial t}\left(\frac{\partial F}{\partial x}\right) = -\frac{\partial}{\partial x}\left(\frac{\partial F}{\partial t}\right) = -\frac{\partial}{\partial x}\left(\frac{\partial F}{\partial u}\frac{\partial u}{\partial t}\right) \\ &= \frac{\partial}{\partial x}\left(F'(u)\frac{\partial F}{\partial x}\right) \quad . \end{aligned}$$

The Taylor's expansion then leads to,

$$\begin{aligned} u_j^{n+1} &= u_j^n - \frac{\Delta t}{2\Delta x}(F(u_{j+1}^n) - F(u_{j-1}^n)) \\ &+ \frac{\Delta t^2}{2\Delta x^2}\left(F'(u_{j+\frac{1}{2}}^n)(F(u_{j+1}^n) - F(u_j^n)) - F'(u_{j-\frac{1}{2}}^n)(F(u_j^n) - F(u_{j-1}^n))\right) \end{aligned}$$

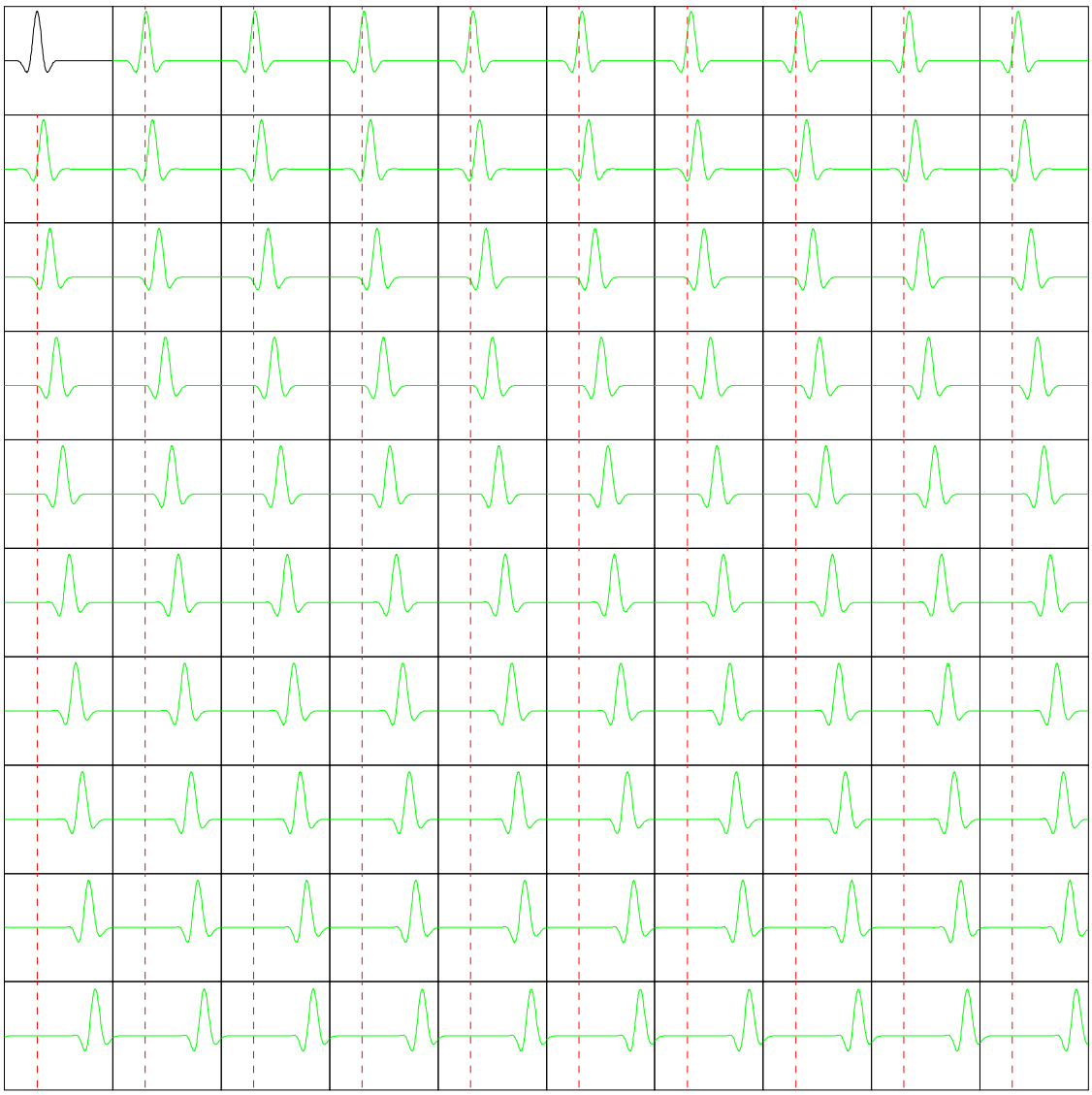
where $u_{j\pm\frac{1}{2}}^n = (u_{j\pm1}^n + u_j^n)/2$.

For the advection equation this simplifies to

$$u_j^{n+1} = u_j^n - \frac{v\,\Delta t}{2\Delta x}(u_{j+1}^n - u_{j-1}^n) + \frac{v^2\Delta t^2}{2\Delta x^2}(u_{j+1}^n + u_{j-1}^n - 2u_j^n) \quad .$$

The method is stable, because of "numerical viscosity" but the solution does not dissapate as rapidly as the Lax method.

Example of a solution to the advection equation using Lax-Wendroff scheme, with $\frac{v\Delta t}{\Delta x} = 0.6$. Each box represents a new time bin.



Application: Fluid Mechanics in 1D

A 1D fluid in motion satisfies the continuity equation,

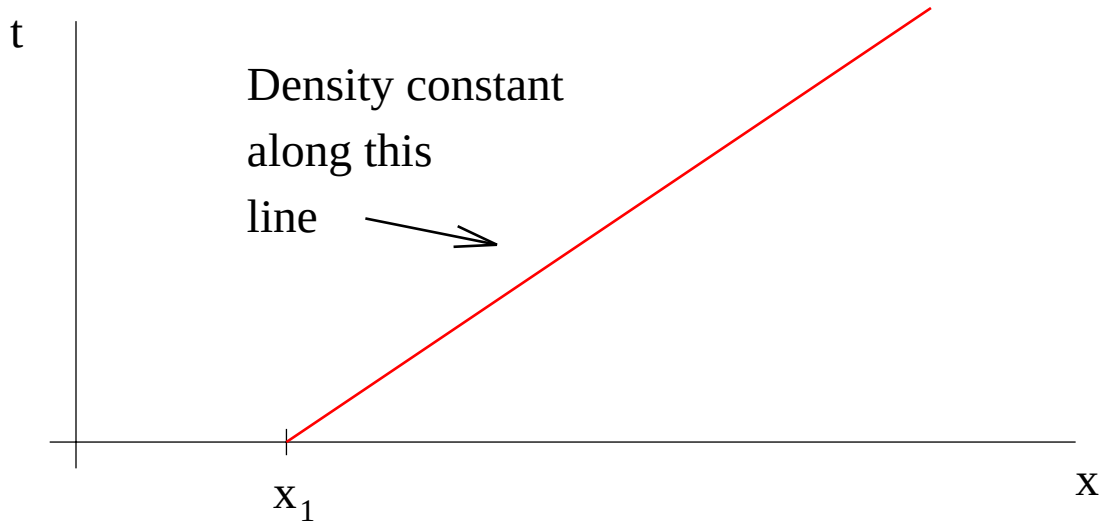
$$\frac{\partial \rho(x,t)}{\partial t} = -\frac{\partial}{\partial x}\left\{\rho(x,t)\, v(x,t)\right\}$$

where ρ is the mass density and v the velocity. There are also equations for the other conserved quantities; the momentum density and the energy density. An exact treatment requires that all three be solved simultaneously.

Consider the simplifying assumption that the velocity depends only on the density. Then,

$$\begin{aligned}
\frac{\partial \rho}{\partial t} &= -\frac{\partial}{\partial x}(\rho\, v(\rho)) \\
&= -\left(\frac{\partial \rho}{\partial x} v(\rho) + \rho \frac{dv}{d\rho}\frac{\partial \rho}{\partial x}\right) \\
&= -\left(\frac{d}{d\rho}\rho\, v(\rho)\right)\frac{\partial \rho}{\partial x} \\
&= -c(\rho)\frac{\partial \rho}{\partial x} \quad .
\end{aligned}$$

The term $c(\rho)$ is the speed at which density waves travel, or in other words the density is constant along the line with slope $1/c(\rho_0)$:



Proof: The time derivative along this line is,

$$\frac{d\rho}{dt} = \frac{\partial\rho}{\partial t} + \frac{\partial\rho}{\partial x}\frac{dx}{dt} = 0$$

$$\rightarrow \frac{dx}{dt} = -\frac{\partial\rho}{\partial t}\Big/\frac{\partial\rho}{\partial x} = c(\rho_0)$$

Traffic Simulation

The velocity of automobile traffic is limited to a maximum and decreases roughly linearly with increasing density,

$$v(\rho) = v_m(1 - \rho/\rho_m) \quad .$$

In this case,

$$\begin{aligned} c(\rho) &= v_m(1 - 2\rho/\rho_m) \\ \rightarrow c(0) &= v_m \\ \rightarrow c(\rho_m) &= -v_m \end{aligned}$$

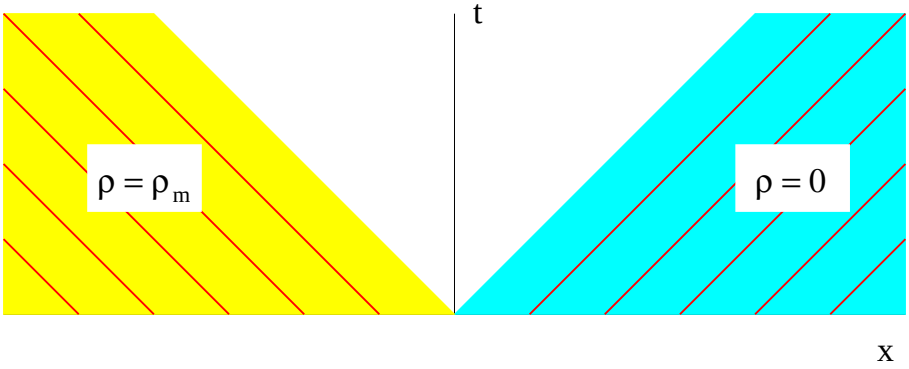
So the density waves can travel in either direction.

Traffic at a stoplight

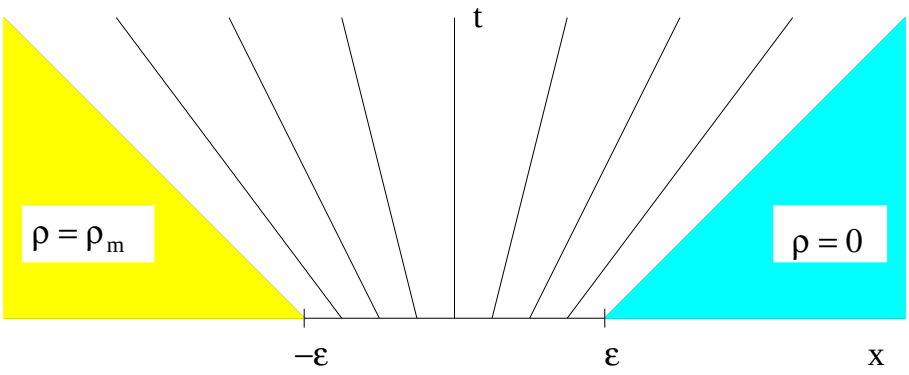
The analytical solution to the problem with initial density,

$$\rho(x, t = 0) = \begin{cases} \rho_m & x \leq 0 \\ 0 & x > 0 \end{cases}$$

contains the following regions:



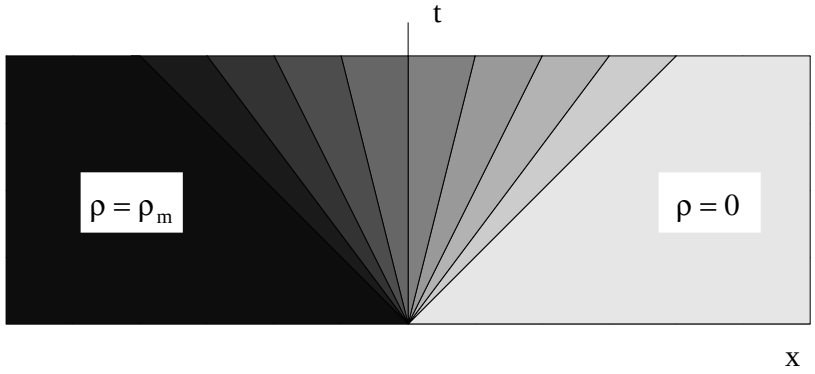
To determine ρ in the central region, the discontinuous initial condition at $x = 0$ must be considered. If in the region $(-\epsilon, \epsilon)$ the density varied linearly from ρ_m to 0, the solution would be:



Taking the limit $\epsilon \to 0$, the solution is given by

$$\rho(x,t) = \begin{cases} \rho_m & \text{for} \quad x \le -v_m t \\ c^{-1}(x/t) & \text{for} \quad -v_m t < x < v_m t \\ 0 & \text{for} \quad x \ge -v_m t \end{cases}$$

where $c^{-1}(x/t) = \rho_m(1 - x/(v_m t))/2$. Graphically the solution is given by,



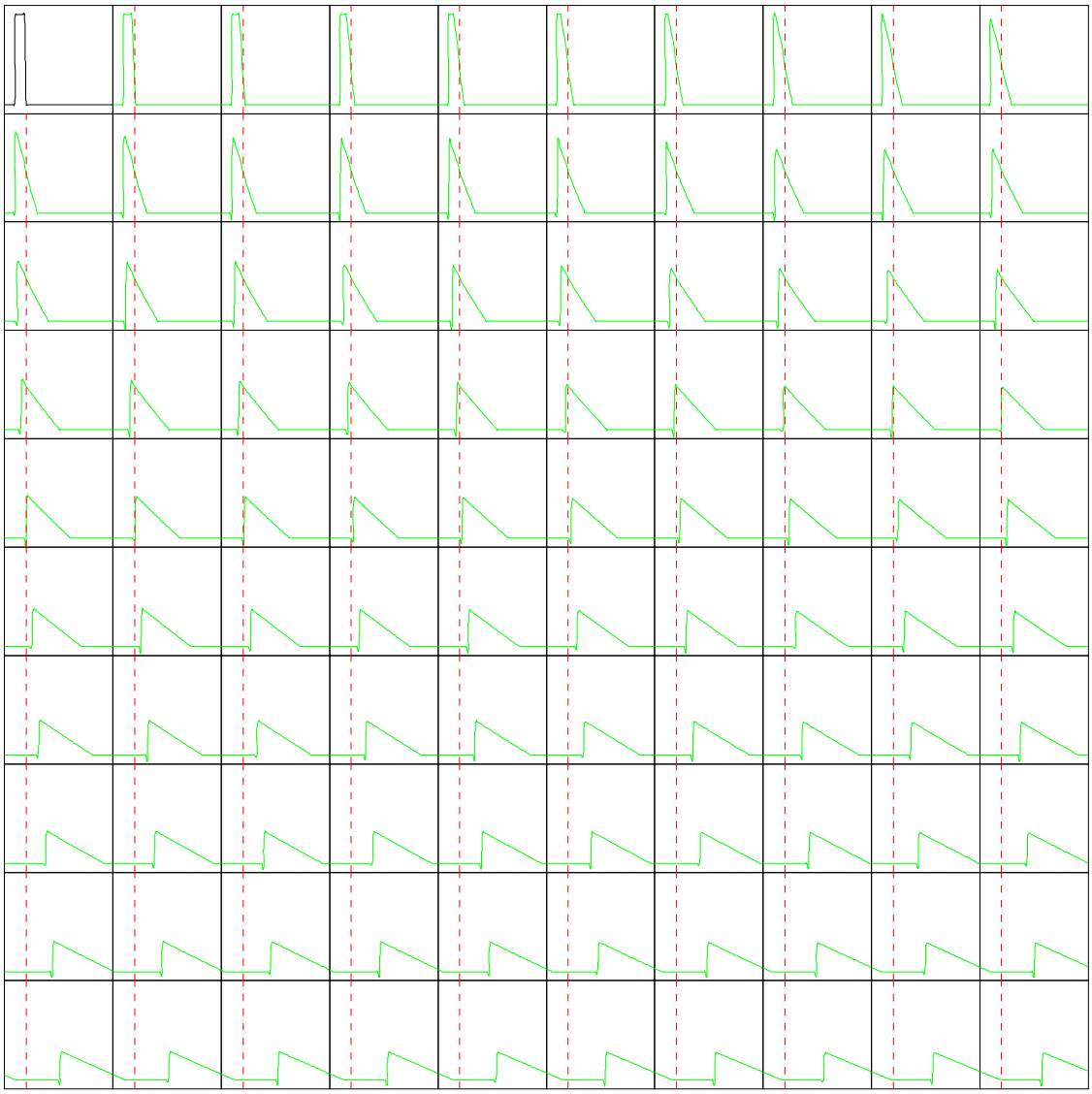
Now that the analytic solution is understood, check to see if the numerical methods reproduce these results:

Use the Lax-Wendroff Scheme, where

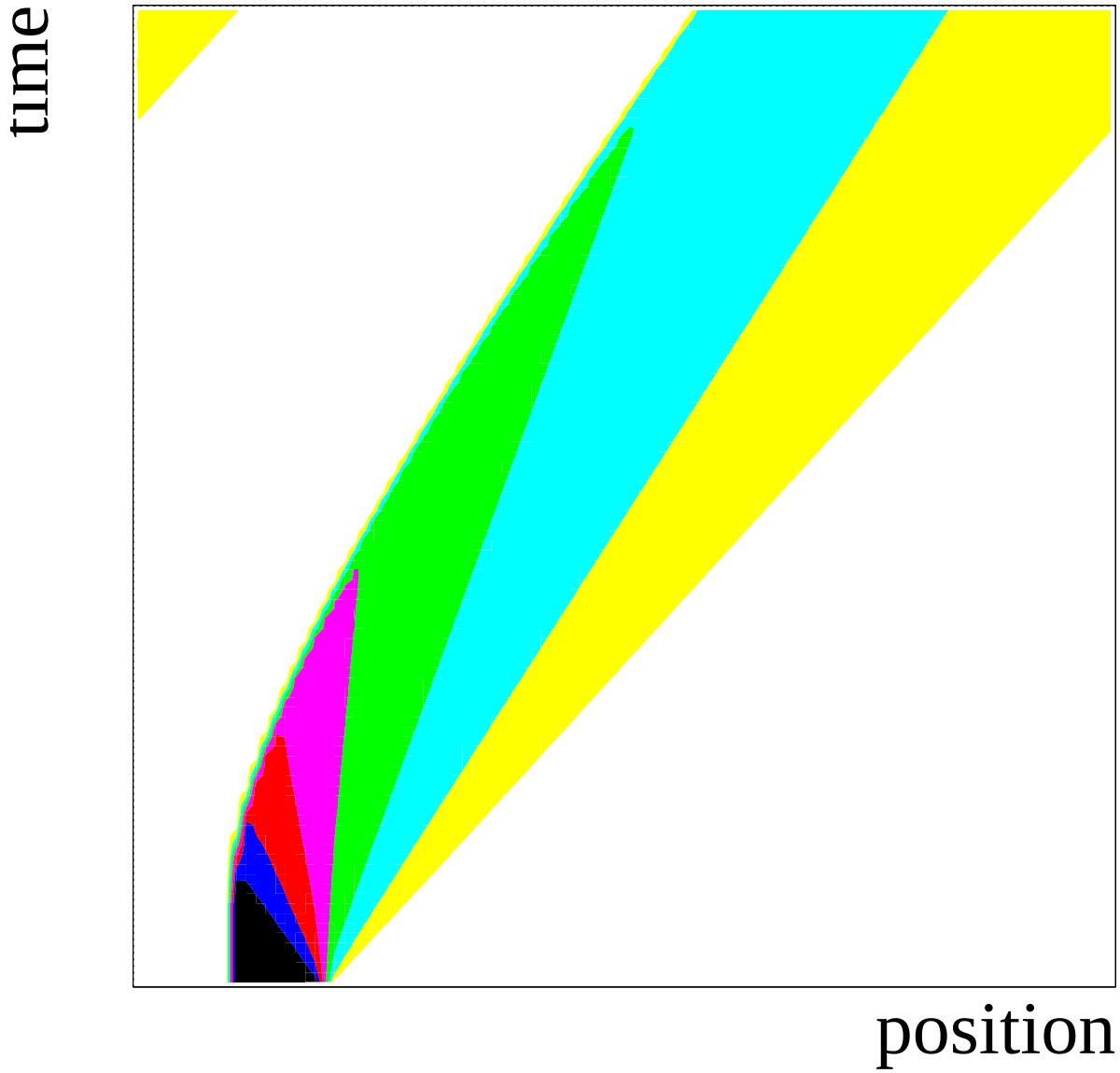
$$\begin{aligned} F(\rho) &= \rho\, v(\rho) = \rho\, v_m(1-\rho/\rho_m) \\ F'(\rho) &= c(\rho) = v_m(1-2\rho/\rho_m) \end{aligned}$$

Use 100 bins in x, with periodic boundary conditions. Consider the initial configuration to be a square pulse over 10 bins in x. The back edge of the pulse forms a traveling discontinuity, known as a shock front. Even if the initial configuration is smooth, the shock front will still appear.

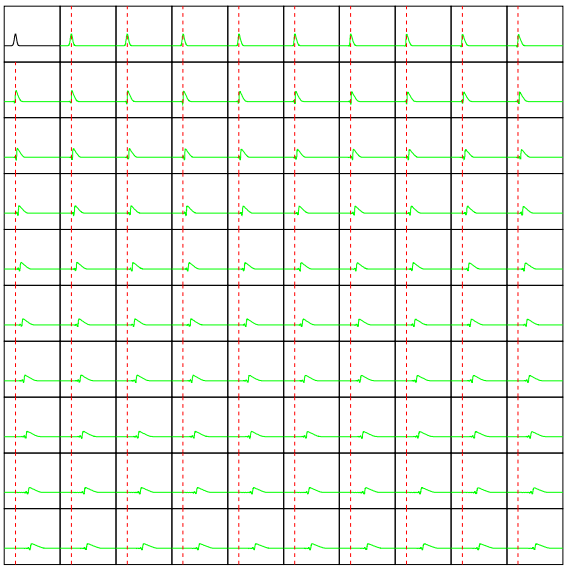
The result for $v_m = 1$, $\rho_m = 1$, $\Delta t/\Delta x = 1$:

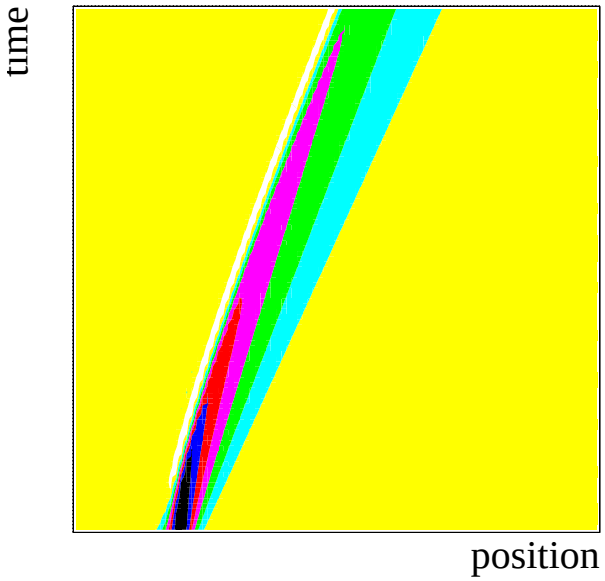


Shown as a density contour plot:

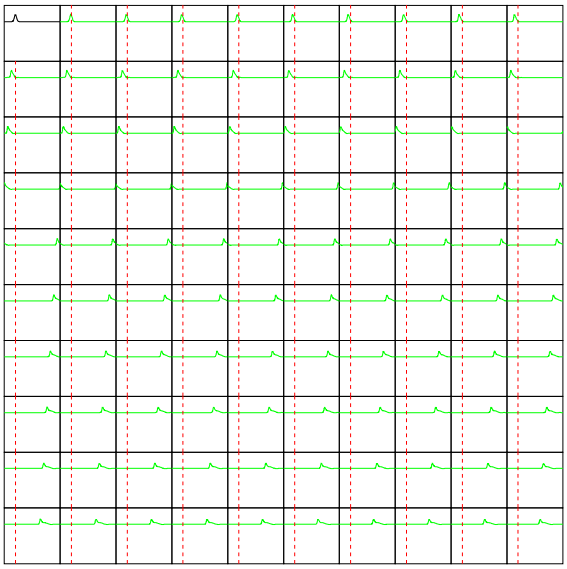


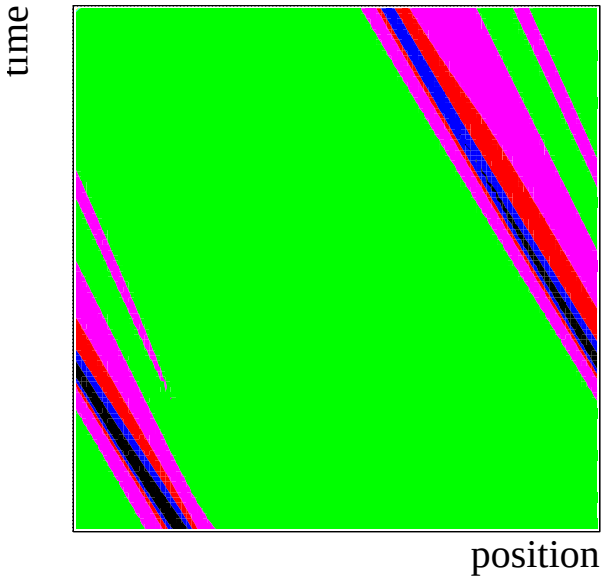
A disturbance in light traffic moves forward:





A disturbance in heavy traffic moves backward:





Diffusive Initial Value Problem

General form in 1D:

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left(D\frac{\partial u}{\partial x}\right)$$

where D is a diffusion coefficient, and $D \geq 0$. This equation is of the flux-conservative form with $F(u) = -D\partial u/\partial x$.

If D is a constant,

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2}$$

can be evaluated with FTCS as,

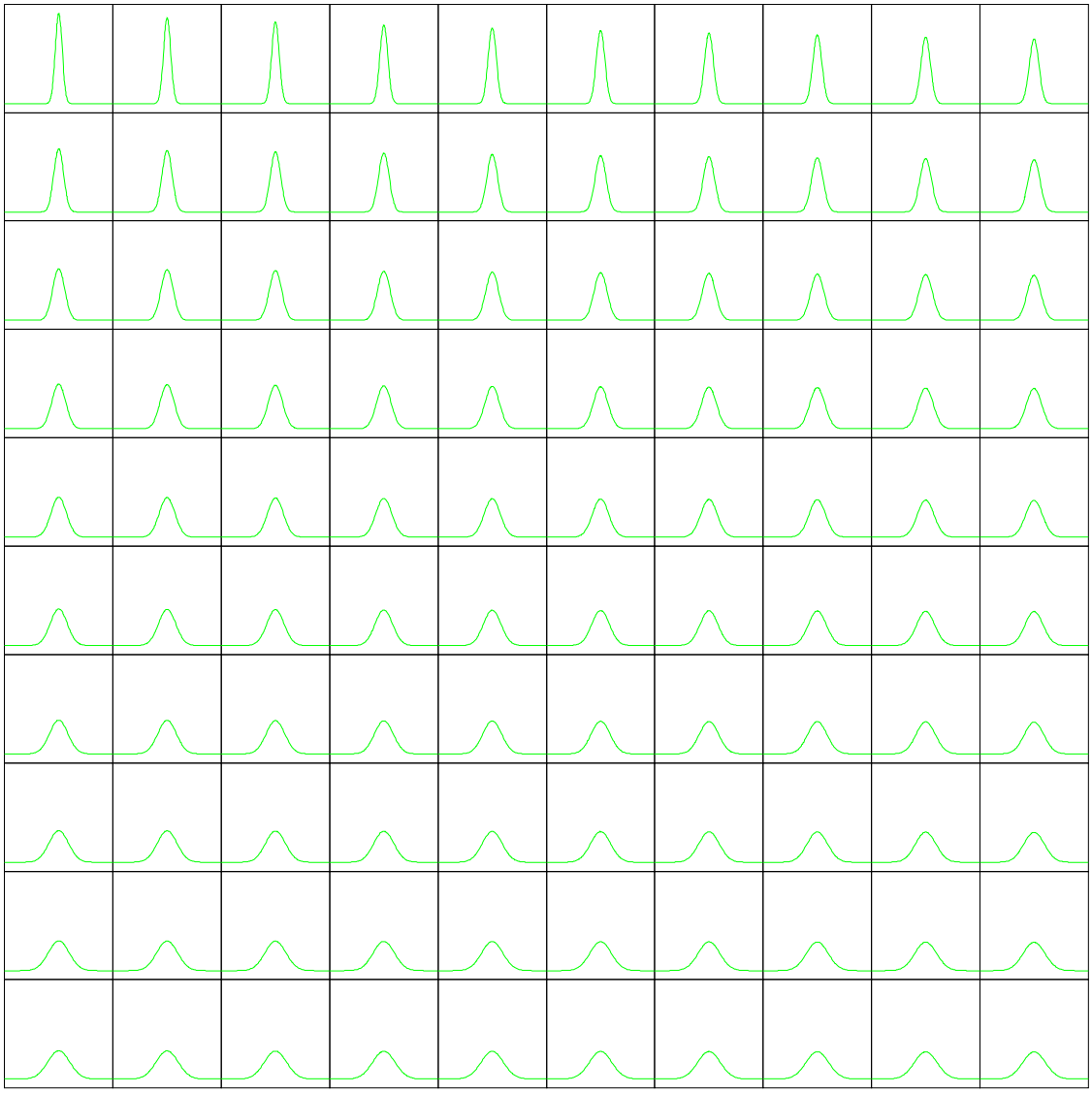
$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = D\left(\frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{\Delta x^2}\right) \quad .$$

This time FTCS is stable as long as,

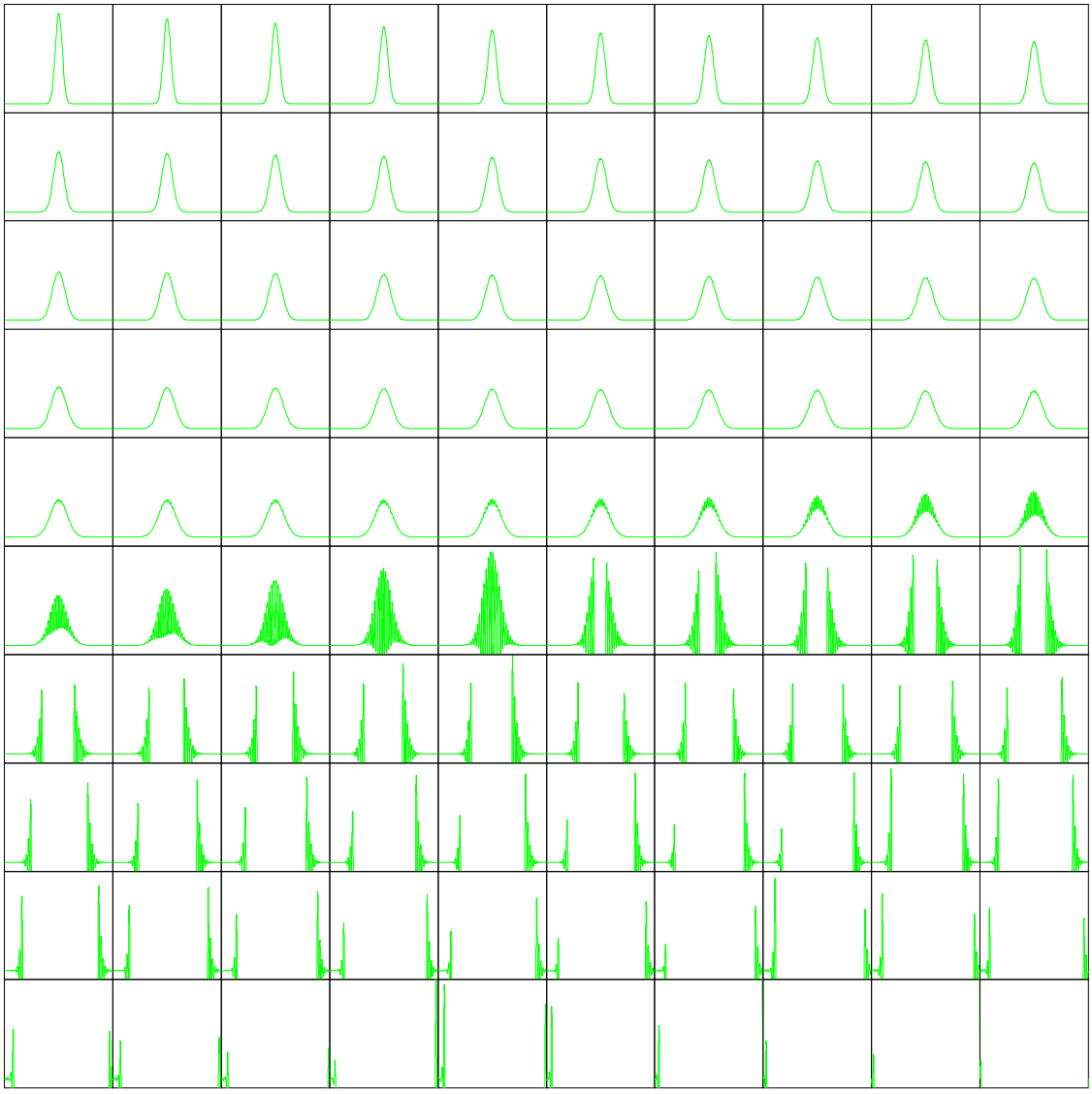
$$\frac{2D\Delta t}{(\Delta x)^2} \leq 1$$

But this can sometimes put a too small upper limit on the time steps Δt for some problems.

The result for $\frac{2D\Delta t}{(\Delta x)^2} = 1$



The result for $\frac{2D\Delta t}{(\Delta x)^2} = 1.2$



To improve the stability, one can use the following differencing scheme

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = D\left(\frac{u_{j+1}^{n+1} - 2u_j^n + u_{j-1}^{n+1}}{\Delta x^2}\right)$$

where the space derivatives are evaluated at time t_{n+1}, and the method is named *backward time*. To solve for u_j^{n+1} requires a solution of a set of linear equations. The method is stable for all choices of Δt.

Crank-Nicholson scheme

Even better, is to simply average the result from forward and backward time methods. This gives a a method that is second order in other time and space and stable for all Δt.

Boundary Value Problems

An example is a problem involving Laplace's equation,

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$$

Relaxation Methods: Jacobi's method

Rewrite the problem as a diffusion equation:

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$$

and an initial distribution for u will relax to an equilibrium solution as $t \to \infty$, where $\frac{\partial u}{\partial t} = 0$.

Define $u^n_{j,\ell} = u(x_j, y_\ell, t_n)$ with $\Delta x = \Delta y = \Delta$. Use FTCS differencing to get

$$u^{n+1}_{j,\ell} = u^n_{j,\ell} + \frac{\Delta t}{\Delta^2}\left(u^n_{j+1,\ell} + u^n_{j-1,\ell} + u^n_{j,\ell+1} + u^n_{j,\ell-1} - 4u^n_{j,\ell}\right)$$

which is stable if $\Delta t/\Delta^2 \leq 1/4$. At the maximum stable time step this gives,

$$u^{n+1}_{j,\ell} = \frac{1}{4}\left(u^n_{j+1,\ell} + u^n_{j-1,\ell} + u^n_{j,\ell+1} + u^n_{j,\ell-1}\right) \quad .$$

This is just a simple average of the 4 neighbouring points in space. The method is to continue iterations until solution converges. However, this is usually too slow for most problems.

Gauss-Seidel Method

There is a slight improvement in converge if updated values of $u^n_{j,\ell}$ are used as they become available,

$$u^{n+1}_{j,\ell} = \frac{1}{4}\left(u^n_{j+1,\ell} + u^{n+1}_{j-1,\ell} + u^n_{j,\ell+1} + u^{n+1}_{j,\ell-1}\right) \quad .$$

Successive Overrelaxation

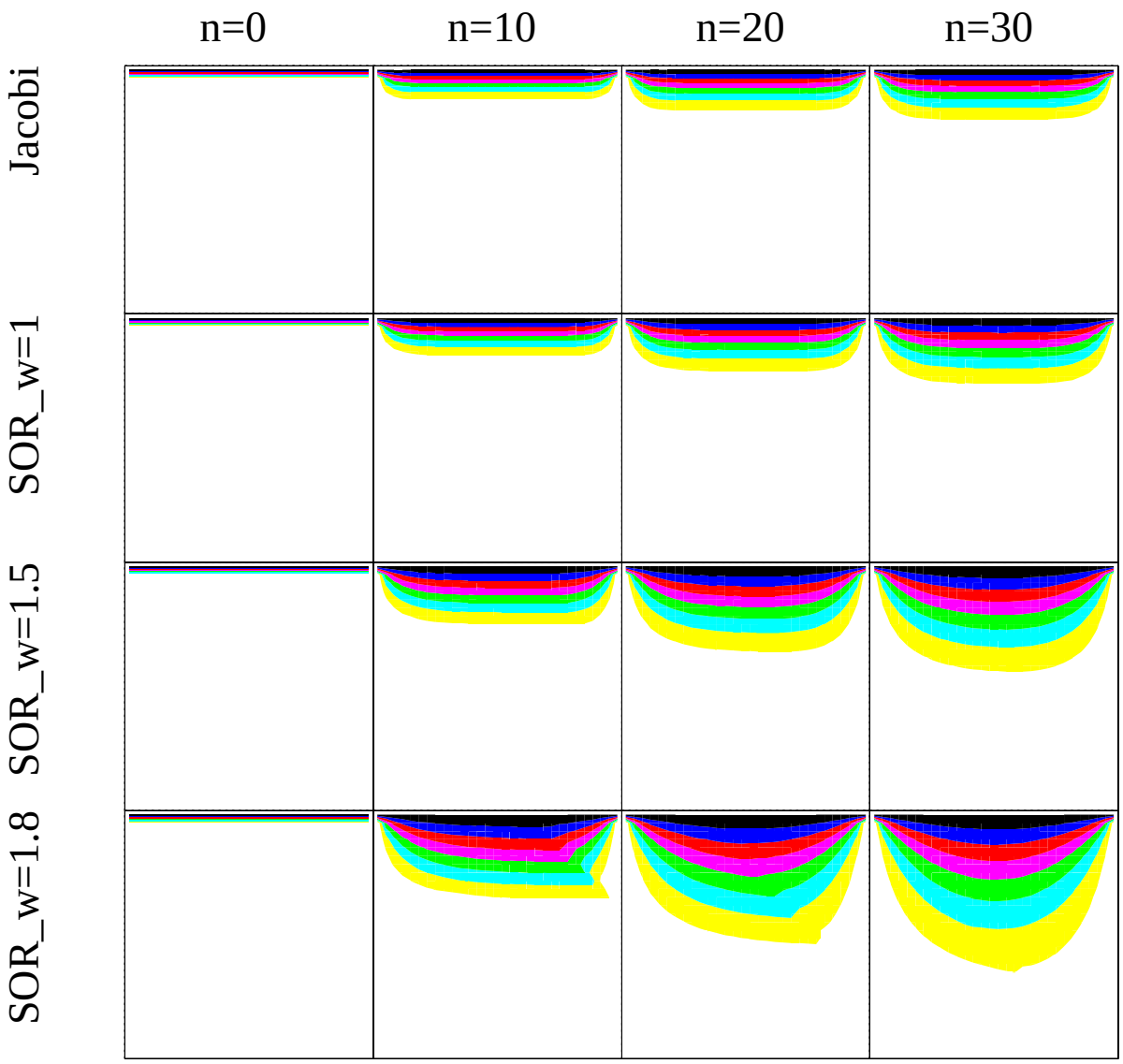
This algorithm converges much more quickly by overcorrecting the values for u at each iteration,

$$u^{n+1}_{j,\ell} = (1-\omega)u^n_{j,\ell} + \frac{\omega}{4}\left(u^n_{j+1,\ell} + u^{n+1}_{j-1,\ell} + u^n_{j,\ell+1} + u^{n+1}_{j,\ell-1}\right) \quad .$$

- $\omega = 1$ is the Gauss-Seidel method
- $0 < \omega < 1$ underrelaxation
- $1 < \omega < 2$ overrelaxation

The optimal choice of ω depends on the problem, and usually found by trial/error.

As an example, the potential within a square cavity where one side is held at a constant potential, and the others held at 0, is shown below. The starting point for each method is potential=0 for all interior points:



Part III: Monte Carlo Methods

Topics:

- Introduction
- Random Number generators
- Special distributions
- General Techniques
- Multidimensional simulation

References:

- *The Art of Computer Programming*, D.E. Knuth, Addison-Wesley, vol 2, 1969.
- *Monte Carlo Theory and Practice*, F. James, Rep. Prog. Phys., Vol. 43, 1980, 1145.
- *Portable Random Number Generators*, W.H. Press, S.A. Teukolsky, Computers in Physics, Vol. 6, No. 5, 1992, 522.

Monte Carlo Techniques

Monte Carlo refers to any procedure that makes use of random numbers.

Monte Carlo methods are used in:

Simulation of natural phenomena
Simulation of experimental appartus
Numerical analysis

Random Numbers

What is a random number? Is 3?

☞ No such thing as a single random number.

A sequence of random numbers is a set of numbers that have nothing to do with the other numbers in the sequence.

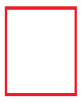
In a uniform distribution of random numbers in the range [0,1] , every number has the same chance of turning up.

Note that 0.00001 is just as likely as 0.50000

How to generate a sequence of random numbers.

Use some chaotic system. (like balls in a barrel - Lotto 6-49).

Use a process that is inherently random:
- radioactive decay
- thermal noise
- cosmic ray arrival



Tables of a few million truely random numbers do exist, but this isn't enough for most applications.

Hooking up a random machine to a computer is not a good idea. This would lead to irreproducable results, making debugging difficult.

Random Number Generation

Pseudo-Random Numbers

These are sequences of numbers generated by computer algorithms, usally in a uniform distribution in the range [0,1].

To be precise, the alogrithms generate integers between 0 and M, and return a real value:

$$x_n = I_n / M$$

An early example :

Middle Square (John Von Neumann, 1946)

To generate a sequence of 10 digit integers, start with one, and square it amd then take the middle 10 digits from the answer as the next number in the sequence.

eg. $5772156649^2 = 33317792380594909291$

so the next number is given by 

The sequence is not random, since each number is completely determined from the previous. But it appears to be random.

This algorothm has problems in that the sequence will have small numbers lumped together, 0 repeats itself, and it can get itself into short loops, for example:

$6100^2 = 37210000$
$2100^2 = 4410000$
$4100^2 = 16810000$
$8100^2 = 65610000$

With more bits, long sequences are possible.
38 bits ➔ 750,000 numbers

A more complex algorithm does not necessarily lead to a better random sequence. It is better to use an algorithm that is well understood.

Linear Conguential Method (Lehmer, 1948)

$$I_{n+1} = (a\, I_n + c) \bmod m$$

Starting value (seed) = I_0

a, c, and m are specially chosen

$a, c \geq 0$ and $m > I_0, a, c$

A poor choice for the constants can lead to very poor sequences.

example: $a=c=I_o=7$, $m=10$

results in the sequence:
7, 6, 9, 0, 7, 6, 9, 0,...

The choice c=0 leads to a somewhat faster algorithm, and can also result in long sequences. The method with c=0 is called: Multiplicative congruential.

Choice of modulus, m

m should be as large as possible since the period can never be longer than m.

One usually chooses m to be near the largest integer that can be represented. On a 32 bit machine, that is $2^{31} \approx 2 \times 10^9$.

Choice of multiplier, a

It was proven by M. Greenberger in 1961 that the sequence will have period m, if and only if:

i) c is relatively prime to m;
ii) a-1 is a multiple of p, for every prime p dividing m;
iii) a-1 is a multiple of 4, if m is a multiple of 4

With c=0, one cannot get the full period, but in order to get the maximum possible, the following should be satisfied:

i) I_0 is relatively prime to m
ii) a is a primative element modulo m

It is possible to obtain a period of length m-1, but usually the period is around m/4.

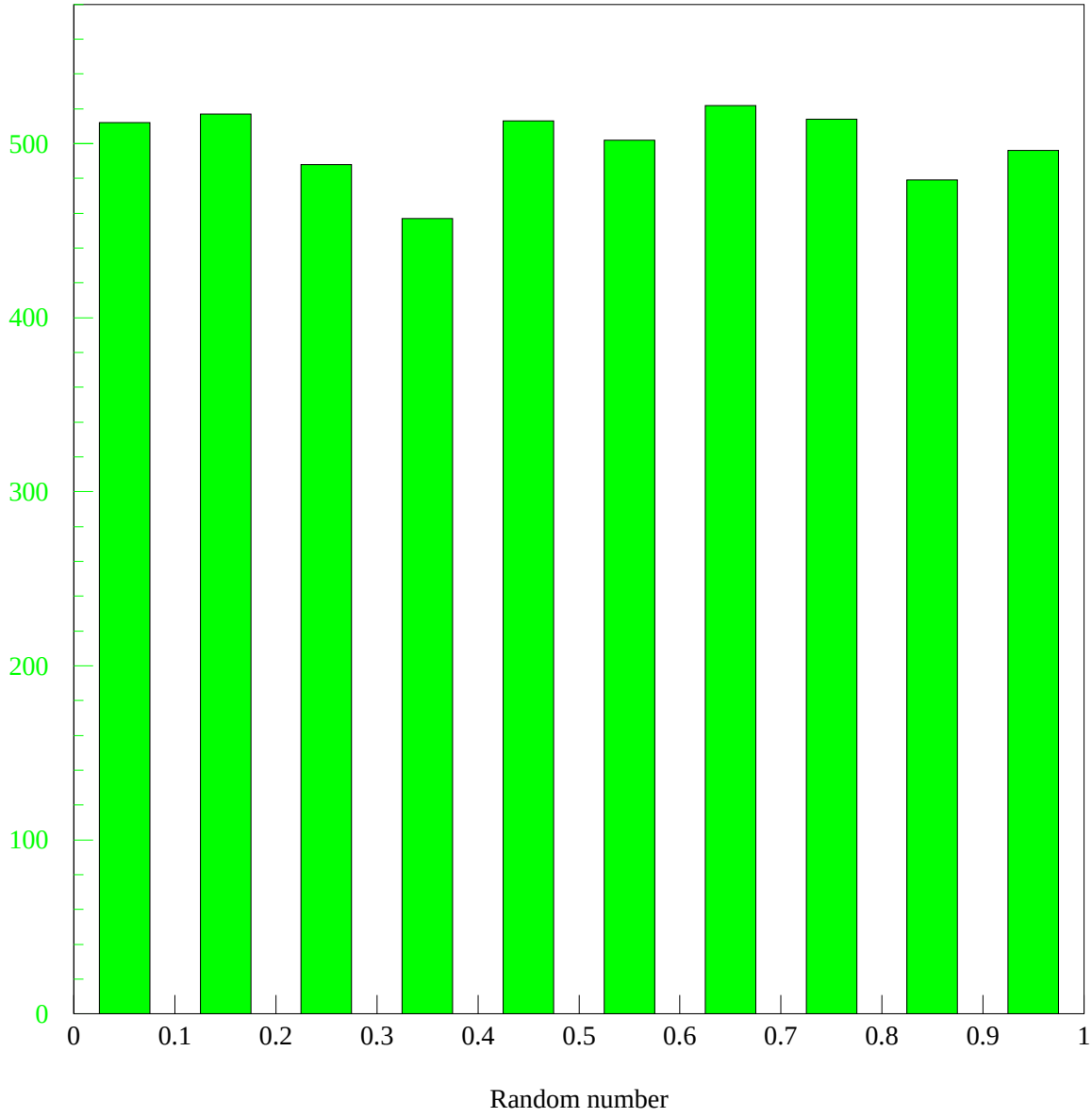
RANDU generator

A popular random number generator was distributed by IBM in the 1960's with the algorithm:

$$I_{n+1} = (65539 \times I_n) \bmod 2^{31}$$

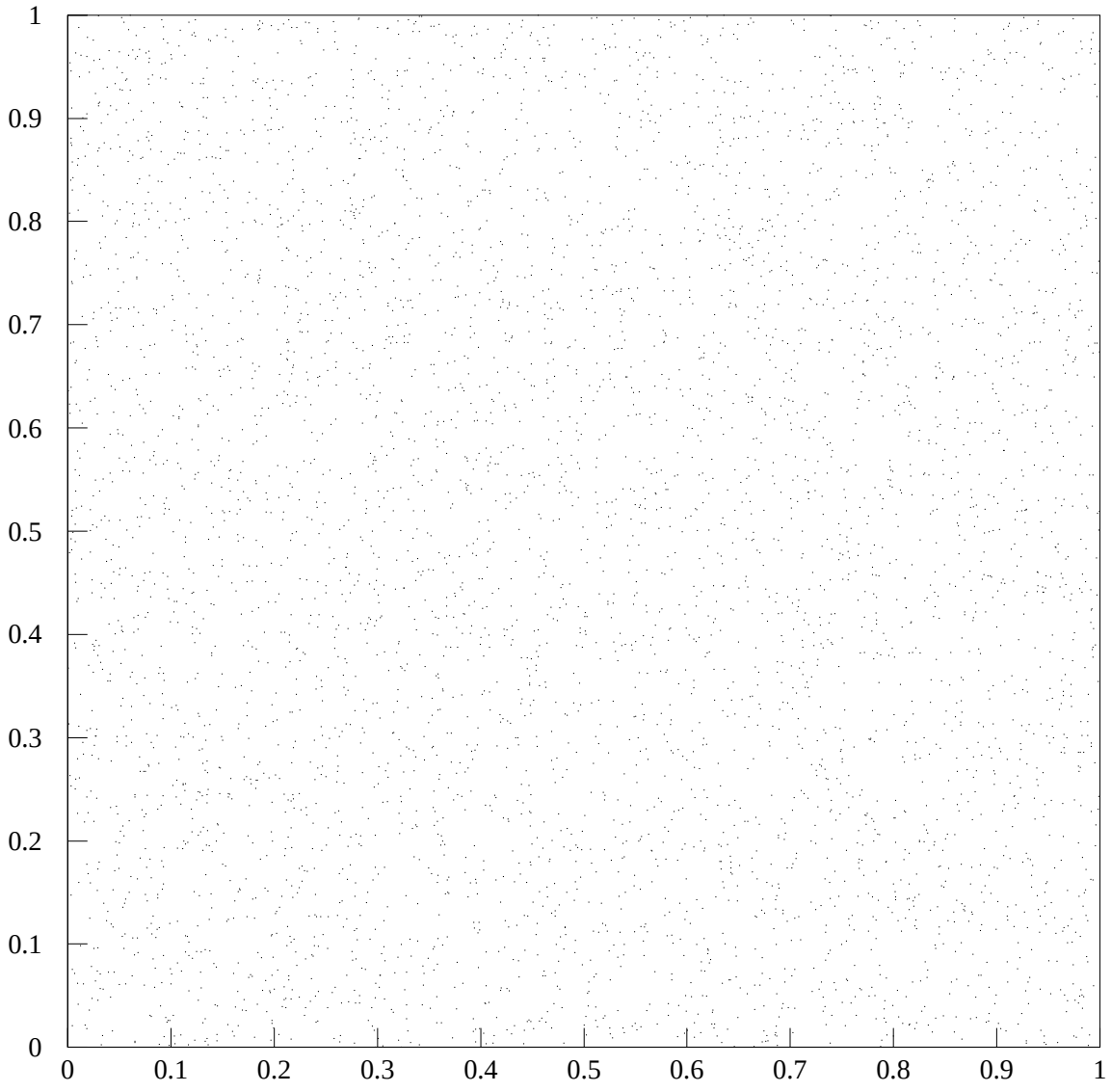
This generator was later found to have a serious problem...

Results from Randu: 1D distribution



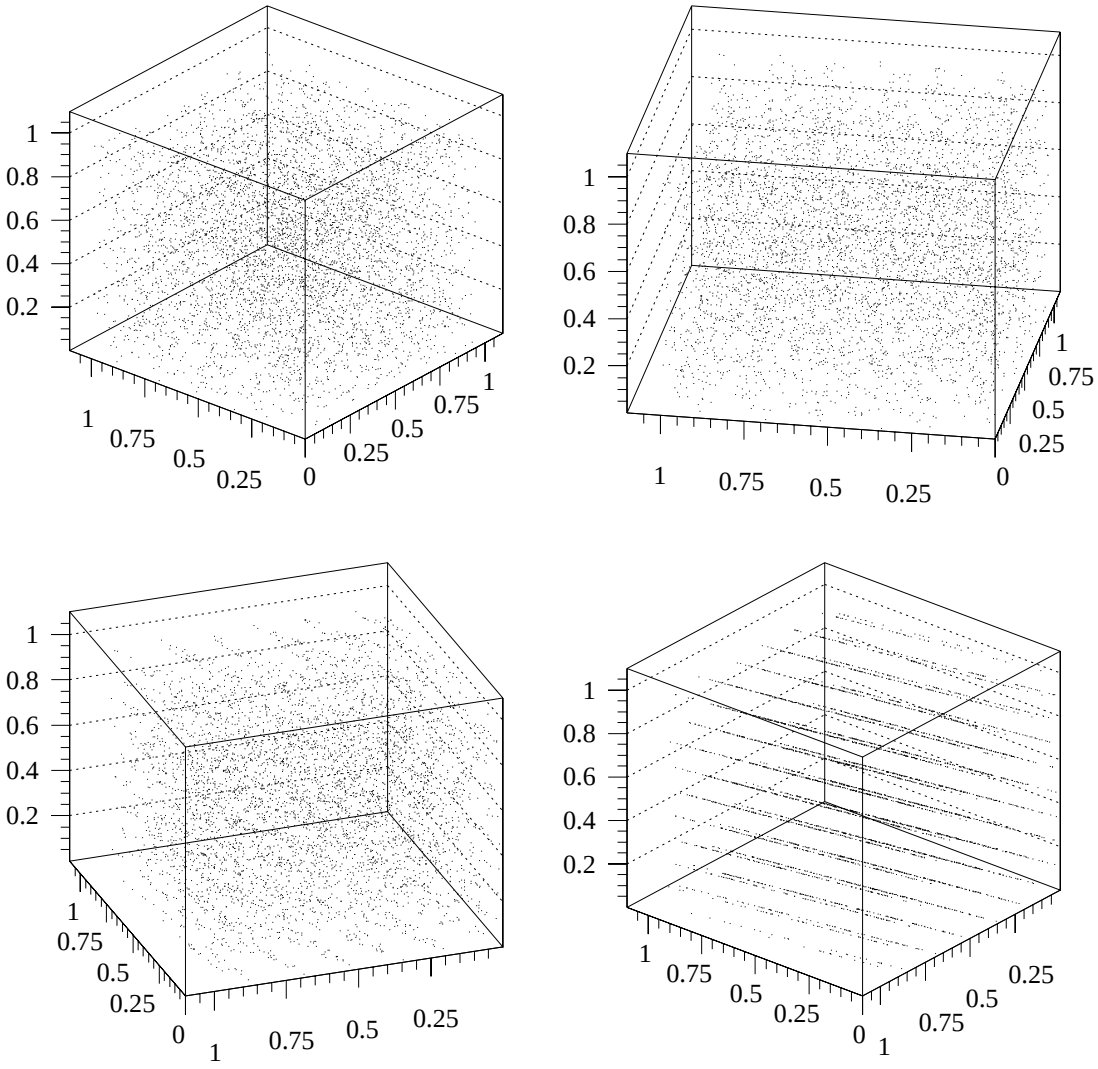
Looks okay

Results from Randu: 2D distribution



Still looks okay

Results from Randu: 3D distribution



Problem seen when observed at the right angle!

The Marsaglia effect

In 1968, Marsaglia published the paper,

Random numbers fall mainly in the planes

(Proc. Acad. Sci. 61, 25) which showed that this behaviour is present for any multiplicative congruential generator.

For a 32 bit machine, the maximum number of hyperplanes in the space of d-dimensions is:

d= 3	2953
d= 4	566
d= 6	120
d=10	41

The RANDU generator had much less than the maximum.
The replacement of the multiplier from 65539 to 69069 improves the performance signifigantly.

Warning

The authors of *Numerical Recipies* have admitted that the random number generators, RAN1 and RAN2 given in the first edition, are “at best mediocre”.

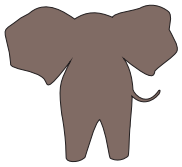
In their second edition, these are replaced by ran0, ran1, and ran2, which have much better properties.

The new routines can also be found in the recent edition of *Computers in Physics*, (Sept/Oct 1992 edition, page 522).

One way to improve the behaviour of random number generators and to increase their period is to modify the algorithm:

$$I_n = (a \times I_{n-1} + b \times I_{n-2}) \bmod m$$

Which in this case has two initial seeds and can have a period greater than m.



RANMAR generator

This generator (available in the CERN library, KERNLIB, requires 103 initial seeds. These seeds can be set by a single integer from 1 to 900,000,000.

Each choice will generate an independat series each of period, $\approx 10^{43}$.

This seems to be the ultimate in random number generators!

Warning on the use of Random Number generators.

In FORTRAN, random number generators areusually used called as functions,

```
x=RAND(IDUM)
```

Where the arguement, `IDUM`, is not used. In fortran, a function is supposed to be a function of only the arguments, and so some compilers will try to optimise the code by removing multiple calls to random number generators.

For example

```
x=RAND(IDUM)+RAND(IDUM)
```

may be changed to

```
x=2.*RAND(IDUM)
```

This can also be a problem when the random number generator is called inside DO loops.

Solution:

Fool the optimiser by always changing the dummy argument:

```
      DO 1 I=1,100
        IDUM=IDUM+1
        x=RAND(IDUM)
        ...
    1 CONTINUE
```

But don't try this if the random number generator uses the argument to save the seed for the next random number. (Numerical Recipies generators, for example)!

Simulating Radioactive Decay

This is a truly random process: The probability of decay is constant (independent of the age of the nuclei).

The probability that a nucleus undergoes radioactive decay in time Δt is p:

$$p = \alpha\,\Delta t \quad (\text{for} \quad \alpha\,\Delta \text{t} \ll 1)$$

Problem:

Consider a system initially having N_0 unstable nuclei. How does the number of parent nuclei, N, change with time?

Algorithm:

```
LOOP from t=0 to t, step Δt
    LOOP over each remaining parent nucleus
        Decide if the nucleus decays:
        IF(random # < α Δt) then
            reduce the number of parents by 1
        ENDIF
    END LOOP over nuclei
    PLOT or record N vs.  t
END LOOP over time
END
```

Exercise 6

Write a program to implement the preceding algorithm. Graph the number of remaining nuclei as a function of time for the following cases:

$$N_0 = 100, \quad \alpha = 0.01\ s^{-1}, \quad \Delta t = 1\ s \quad ;$$

$$N_0 = 5000, \quad \alpha = 0.03\ s^{-1}, \quad \Delta t = 1\ s \quad .$$

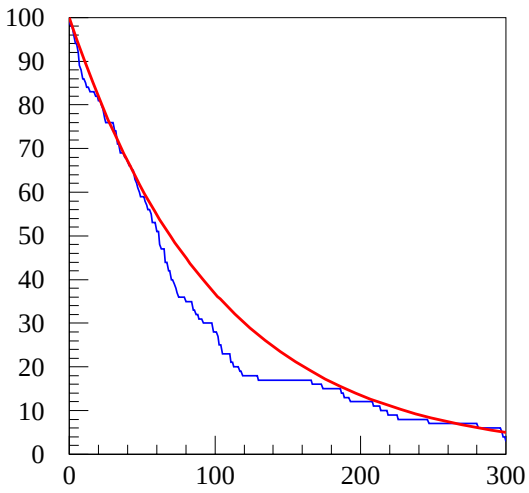
Show the results on both linear and logarithmic scales for times between 0 and 300 seconds. In addition, plot on the same graphs the expected curve, given:

$$dN = -N\,\alpha\,dt$$

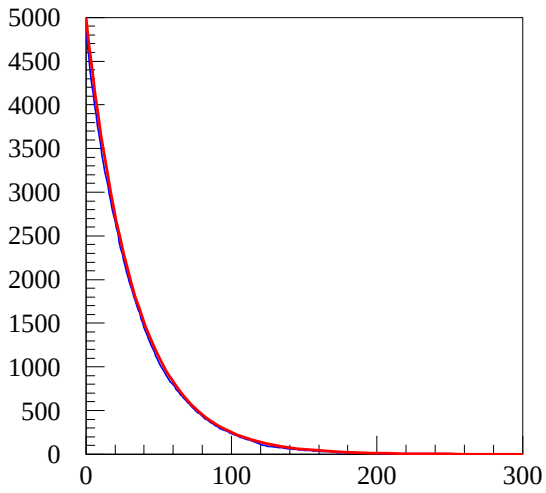
$$ie. \quad N = N_0\,e^{-\alpha t}$$

Solution to exercise 6:

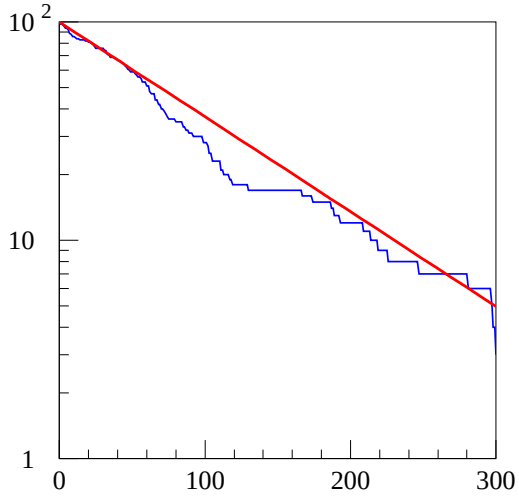
The 'experimental' results do not perfectly follow the expected curve; there are statistical fluctuations.



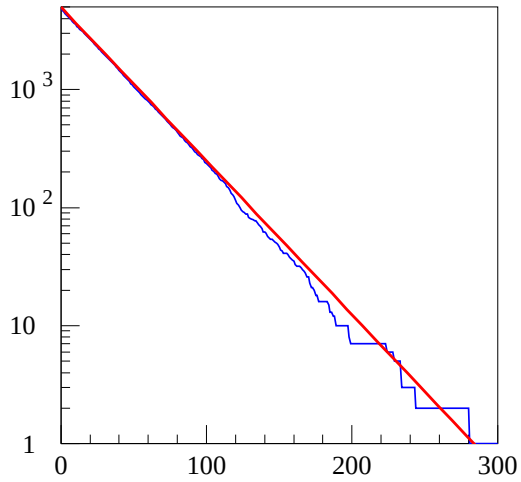
N_0=100, α=0.01



N_0=5000, α=0.03



N_0=100, α=0.01



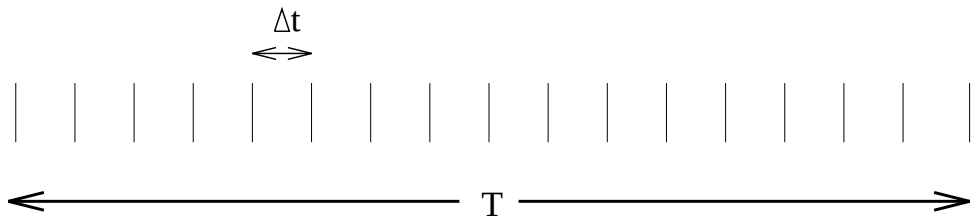
N_0=5000, α=0.03

Poisson Distribution

The probability of observing a total of n decays in a time interval T can be worked out as follows:

Assume the number of decays in time T is much less than the number of parent nuclei. (ie. assume constant probability to observe a decay):

Break up T into m shorter intervals, duration Δt:



The probability to observe 1 decay in time Δt is:

$$p = \beta\,\Delta t$$

where $\beta = \alpha N$ as Δt must be small enough so that $\beta\,\Delta t \ll 1$. The probability of observing n decays in time T is therefore:

$$P = p^n (1-p)^{m-n} \binom{m}{n} .$$

$$\begin{aligned} P &= p^n (1-p)^{m-n} \frac{m!}{(m-n)!\,n!} \\ &= \left(\frac{\beta T}{m}\right)^n \left(1 - \frac{\beta T}{m}\right)^{m-n} \frac{m!}{(m-n)!\,n!} \end{aligned}$$

In the limit of $\Delta t \to 0$ (ie. $m \to \infty$),

$$\begin{aligned} \left(1 - \frac{\beta T}{m}\right)^m &\to e^{-\beta T} \\ \left(1 - \frac{\beta T}{m}\right)^{-n} &\to 1 \\ \frac{m!}{(m-n)!} &\to m^n \end{aligned}$$

The result is,

$$P = \mu^n e^{-\mu}/n!$$

where $\mu = \beta T$. This is known as the Poisson distribution.

Exercise 7

Modify the program written for exercise 6 to simulate an experiment that counts the number of decays observed in a time interval, T.

Allow the experiment to be repeated and histogram the distribution of the number of decays for the following two cases:

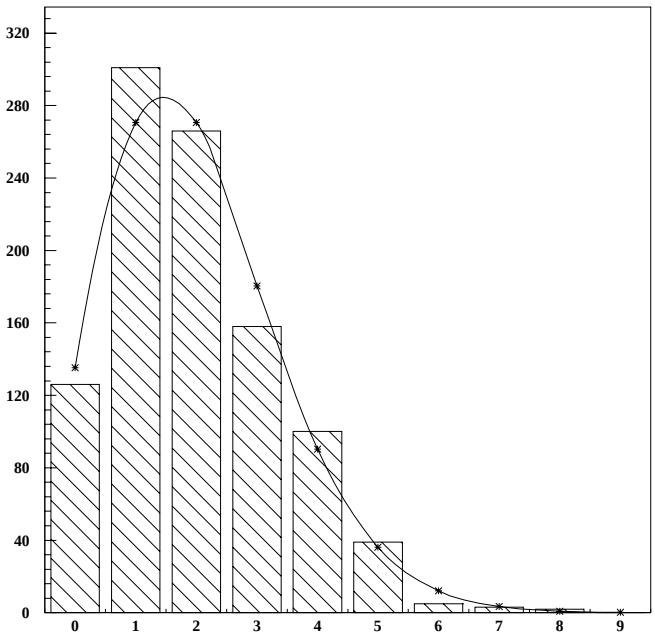
$$N_0 = 500,\ \alpha = 4 \times 10^{-5}\ s^{-1},\ \Delta t = 10\ s,\ T = 100\ s$$

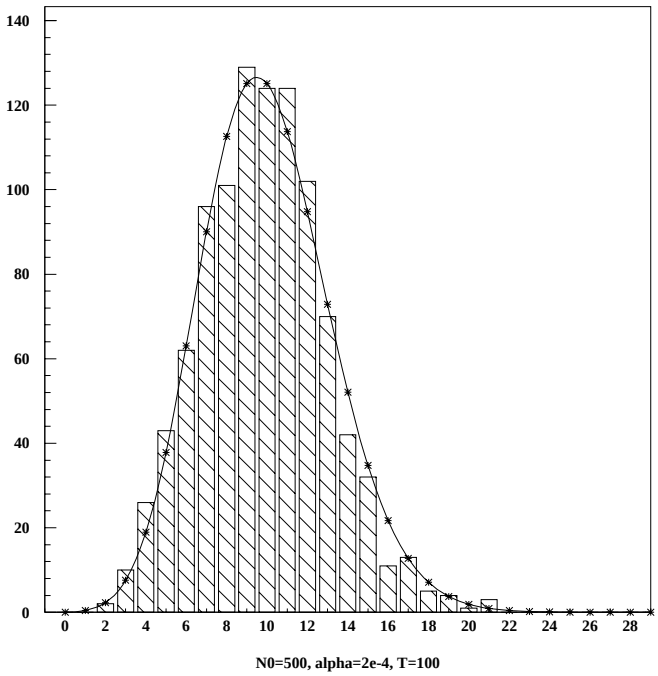
$$N_0 = 500,\ \alpha = 2 \times 10^{-4}\ s^{-1},\ \Delta t = 10\ s,\ T = 100\ s$$

In each case show the distribution using 1000 experiments. Also, overlay the expected Poisson distribution.

Question: Are there limits on the value of Δt so that your program will give reliable results? Explain.

Solution to Exercise 7:





Properties of the Poisson distribution

$$P_n = \frac{\mu^n}{n!}\, e^{-\mu} \qquad (\mu = \alpha N T)$$

Mean value:

$$\begin{aligned}
\langle n \rangle &= \sum_{n=0}^{\infty} \left(n\, \frac{\mu^n}{n!}\, e^{-\mu} \right) \\
&= \mu\, e^{-\mu} \sum_{n=1}^{\infty} \frac{\mu^{n-1}}{(n-1)!} \\
&= \mu\, e^{-\mu} \sum_{m=0}^{\infty} \frac{\mu^m}{m!} = \mu
\end{aligned}$$

Variance:

$$\begin{aligned}
\sigma^2 &= \sum_{n=0}^{\infty} \left((n-\mu)^2\, \frac{\mu^n}{n!}\, e^{-\mu} \right) \\
&= \sum_{n=0}^{\infty} \left((n^2 - 2n\mu + \mu^2)\, \frac{\mu^n}{n!}\, e^{-\mu} \right)
\end{aligned}$$

Do each term individually,

$$\begin{aligned}\sum_{n=0}^{\infty}\left(n^2\,\frac{\mu^n}{n!}\,e^{-\mu}\right) &= \sum_{n=1}^{\infty}\left(n\,\frac{\mu^{n-1}}{(n-1)!}\,e^{-\mu}\right)\mu \\ &= \sum_{n=0}^{\infty}\left((n+1)\,\frac{\mu^n}{n!}\,e^{-\mu}\right)\mu \\ &= (\mu+1)\mu\end{aligned}$$

$$\sum_{n=0}^{\infty}\left(-2n\mu\,\frac{\mu^n}{n!}\,e^{-\mu}\right) = -2\mu^2$$

$$\sum_{n=0}^{\infty}\left(\mu^2\,\frac{\mu^n}{n!}\,e^{-\mu}\right) = \mu^2$$

So, $\sigma^2 = \mu^2 + \mu - 2\mu^2 + \mu^2 = \mu$.

Hence if n decays are observed, the 1 standard deviation uncertainty is $\sqrt{n}$. (This is also true for any other variable that follows the Poisson distribution.)

Many observables follow the Poisson distribution: Anything whose probability of occurring is constant in time.

For example:

- number of observed events when efficiency is constant
- number of entries in a histogram bin

Some measurements lead to non-Poisson distributions:

For example:

- number of radioactive decays observed in a fixed time interval, when there is a significant reduction of parent nuclei
- number of radioactive decays observed in a fixed time interval, when there is significant deadtime. (ie. the detector is not active for some period after an event is recorded)

Gaussian (or Normal) Distribution

This is the most important distribution in statistical analysis.

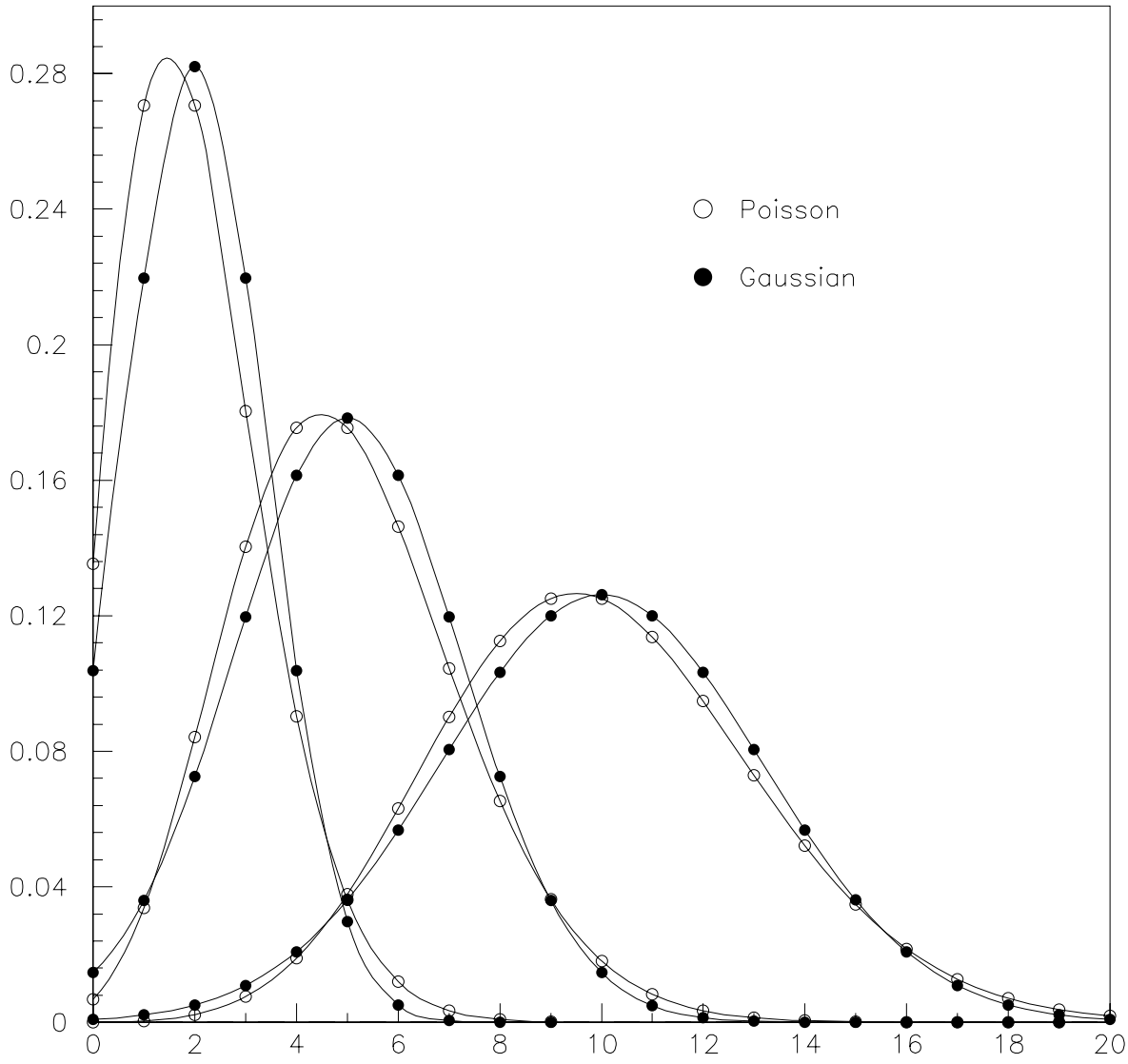
$$G(x|\,\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

The mean of the distribution is μ and the variance is σ^2.

For large μ, the Poisson distribution approaches the Gaussian distribution (with $\sigma^2 = \mu$).

The Gaussian distribution is a reasonable approximation of the Poisson distribution even for μ as small as 5.

Comparison of Poisson and Gaussian distributions:



Binomial Distribution

The binomial distribution describes the results of repeated experiments which has only two possible outcomes.

Suppose a radioactive source is monitored for a time interval T. There is a probability p that one or more disintegrations would be detected in that time interval. If a total of m intervals were recorded, the probability that n of them had at least one decay is

$$P = p^n (1-p)^{m-n} \binom{m}{n} .$$

The mean of this distribution is: np

The variance of this distribution is: $np(1-p)$

Simulating General Distributions

The simple simulations considered so far, only required a random number sequence that is uniformly distributed between 0 and 1. More complicated problems generally require random numbers generated according to specific distributions.

For example, the radioactive decay of a large number of nuclei (say 10^{23}), each with a tiny decay probability, cannot be simulated using the methods developed so far. It would be far too inefficient and require very high numerical precision.

Instead, a random number generated according to a Poisson distribution could be used to specify the number of nuclei that disintigrate in some time T.

Random numbers following some special distributions, like the Poisson distribution, can be generated using special purpose algorithms, and efficient routines can be found in various numerical libraries.

If a special purpose generator routine is not available, then use a general purpose method for generating random numbers according to an arbitrary distribution.

Rejection Technique

Problem: Generate a series of random numbers, x_i, which follow a distribution function $f(x)$.

In the rejection technique, a trial value, x_{trial} is chosen at random. It is accepted with a probability proportional to $f(x_{\text{trial}})$.

Algorithm:

Choose trial x, given a uniform random number λ_1:

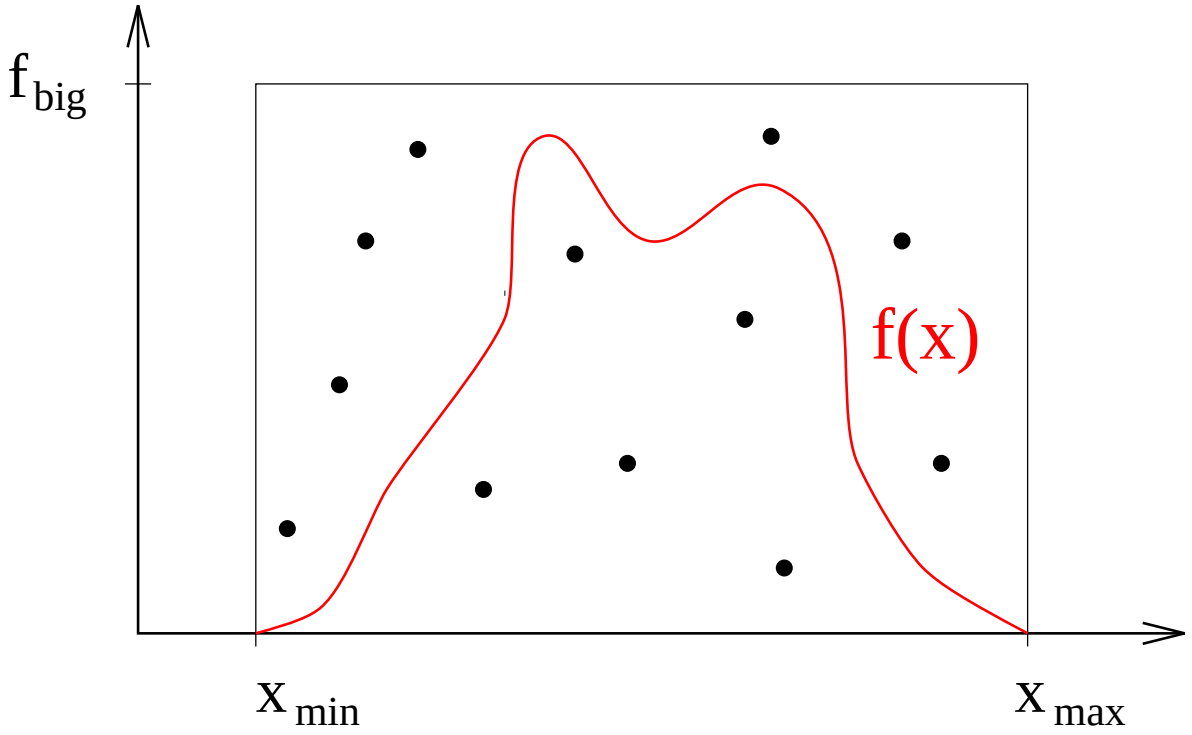
$$x_{\text{trial}} = x_{\min} + (x_{\max} - x_{\min})\lambda_1$$

Decide whether to accept the trial value:

$$\text{if } f(x_{\text{trial}}) > \lambda_2 f_{\text{big}} \text{ then accept}$$

where $f_{\text{big}} \geq f(x)$ for all x, $x_{\min} \leq x \leq x_{\max}$. Repeat the algorithm until a trial value is accepted.

This algorithm can be visualized as throwing darts:



This procedure also gives an estimate of the integral of $f(x)$:

$$I = \int_{x_{\min}}^{x_{\max}} f(x)\,dx \approx \frac{n_{\text{accept}}}{n_{\text{trial}}} f_{\text{big}}(x_{\max} - x_{\min})$$

The 1 standard deviation uncertainty can be derived using the variance of the binomial distribution:

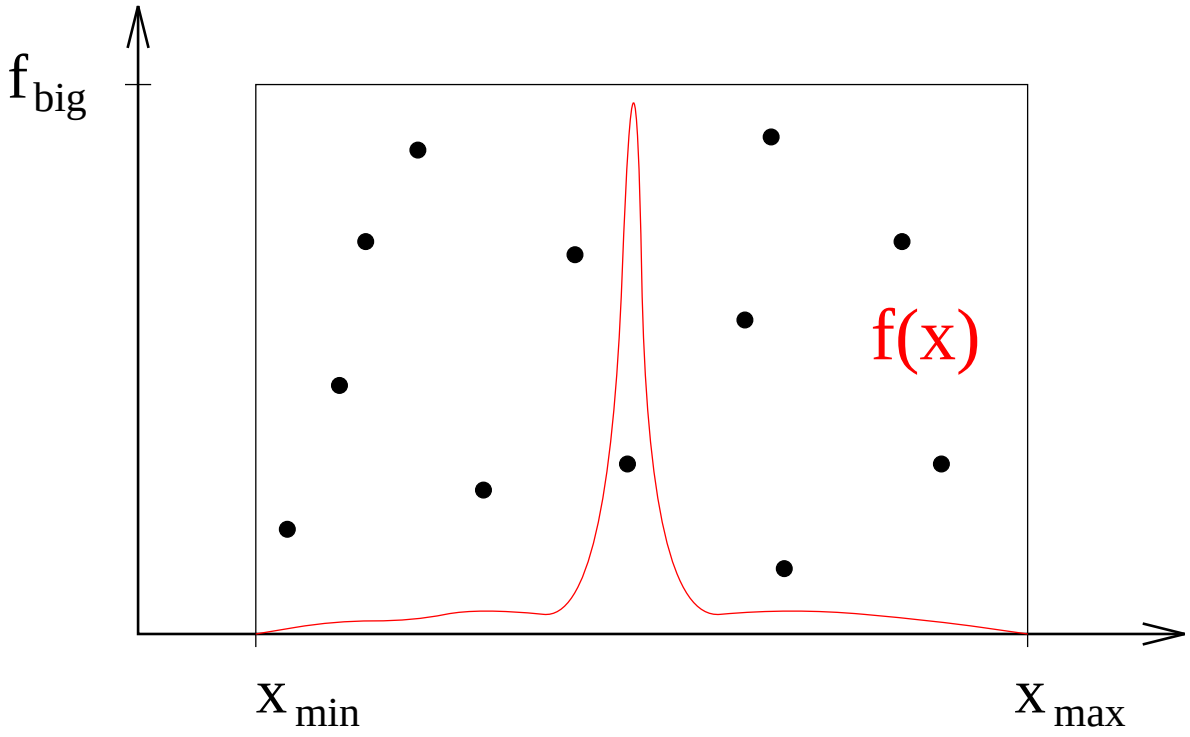
$$\delta N_{\text{accept}} = \sqrt{p(1-p)N_{\text{trial}}} \qquad p = \frac{N_{\text{accept}}}{N_{\text{trial}}}$$

$$\begin{aligned}
\left(\frac{\delta I}{I}\right)^2 &= \left(\frac{\delta N_{\text{accept}}}{N_{\text{accept}}}\right)^2 \\
&= \frac{N_{\text{accept}}}{N_{\text{trial}}}\left(1 - \frac{N_{\text{accept}}}{N_{\text{trial}}}\right) N_{\text{trial}} \frac{1}{N_{\text{accept}}^2} \\
&= \frac{1}{N_{\text{accept}}} - \frac{1}{N_{\text{trial}}} \\
&= \frac{1}{N_{\text{trial}}}\left(\frac{1-p}{p}\right)
\end{aligned}$$

So the relative accuracy only improves with $N_{\text{trial}}^{-\frac{1}{2}}$

The rejection algorithm is not efficient if the distribution has one or more large peaks (or poles).

In this case trial events are seldomly accepted:



In extreme cases, where there is a pole, f_{big} cannot be specified. This algorithm doesn't work when the range of x is $(-\infty, +\infty)$. A better algorithm is needed...

Inversion Technique

This method is only applicable for relatively simple distribution functions:

- First normalize the distribution function, so that it becomes a probability distribution function (PDF).
- Integrate the PDF analytically from the minimum x to an aritrary x. This represents the probability of chosing a value less than x.
- Equate this to a uniform random number, and solve for x. The resulting x will be distributed according to the PDF.

In other words, solve the following equation for x, given a uniform random number, λ:

$$\frac{\int_{x_{\min}}^{x} f(x)\,dx}{\int_{x_{\min}}^{x_{\max}} f(x)\,dx} = \lambda$$

This method is fully efficient, since each random number λ gives an x value.

Examples of the inversion technique

1) generate x between 0 and 4 according to $f(x) = x^{-\frac{1}{2}}$:

$$\frac{\int_0^x x^{-\frac{1}{2}}\,dx}{\int_0^4 x^{-\frac{1}{2}}}\,dx = \lambda$$

$$\tfrac{1}{2}x^{\frac{1}{2}} = \lambda$$

$\Rightarrow$ generate x according to $x = 4\lambda^2$

2) generate x between 0 and ∞ according to $f(x) = e^{-x}$:

$$\frac{\int_0^x e^{-x}\,dx}{\int_0^\infty e^{-x}\,dx} = \lambda$$

$$1 - e^{-x} = \lambda$$

$\Rightarrow$ generate x according to $x = -\ln(1-\lambda)$

Note that the simple rejection technique would not work for either of these examples.

Exercise 8

Write a program that generates the value θ according to the distribution function:

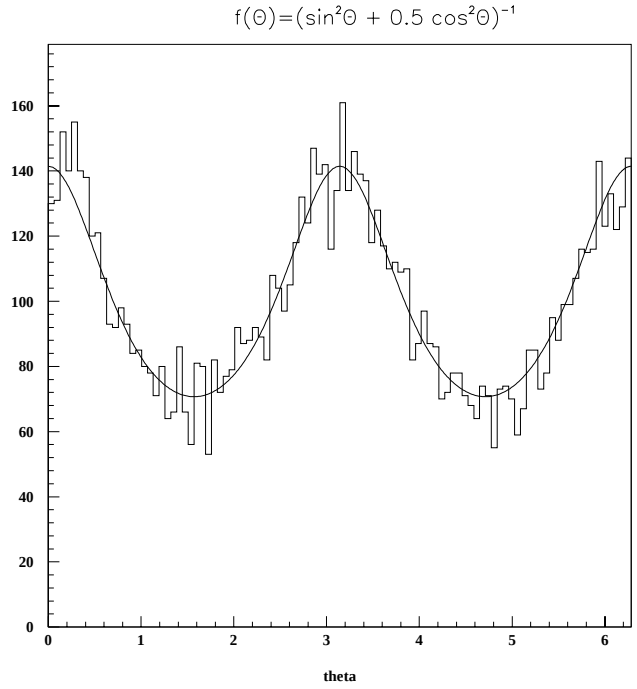
$$f(\theta) = (\sin^2\theta + a\,\cos^2\theta)^{-1}$$

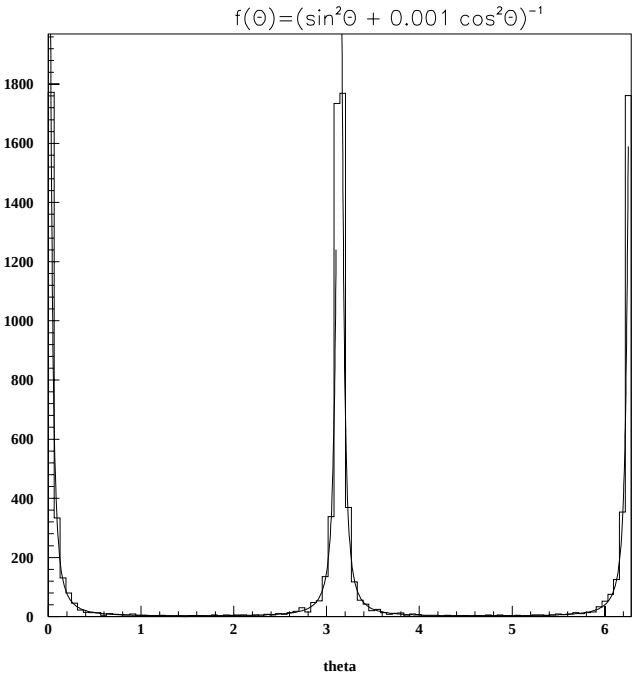
in the range $0 \leq \theta \leq 2\pi$.

Compare the rejection technique and the inversion technique:

- Generate 10000 values for each method using $a = 0.5$ and also for $a = 0.001$.
- Plot the results for each (4 plots) and overlay the distribution curve, $f(\theta)$, properly normalized.
- Compare the CPU time required for the 4 runs.

Solution to Exercise 8:





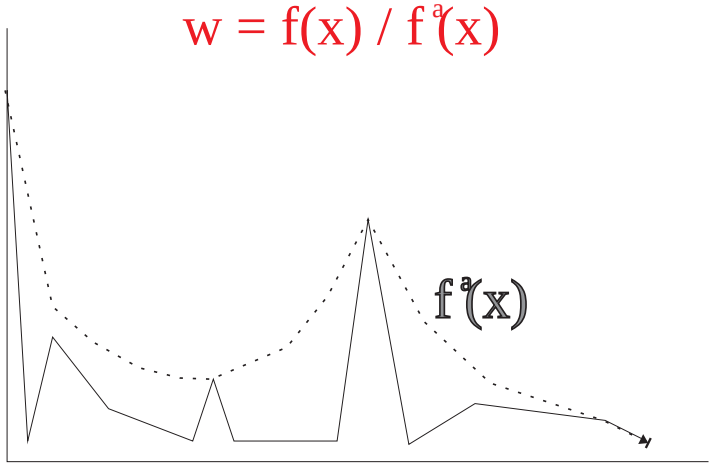
What if the rejection technique is impractical and you can't invert the integral of the distribution function?

Importance Sampling

Replace the distribution function, f(x), by an approximate form, $f^a(x)$, for which the inversion technique can be applied.

Generate trial values for x with the inversion technique according to $f^a(x)$, and accept the trial value with the probability proportional to the weight:

$$w = f(x) / f^a(x)$$



The rejection technique is just the special case where $f^a(x)$ is chosen to be a constant.

Example:

Generate x according to $f(x) = (1+x)x^{-1/2}$
for the range $0 < x < 1$.

There is clearly a problem at x near 0.

$f_a(x)$ needs to be chosen so the weights for the trial values of x are well behaved,

$$w = f(x)/f_a(x)$$

Try $f_a(x) = x^{-1/2}$, then $w=1+x$

Procedure:

Generate trial x: $x = \lambda_1^2$

Decide to accept: if $(1+x) > \lambda_2 w_{max}$ accept

In this case, $w_{max}=2$, but in more complicated cases, you may need to run the program to find the maximum weight generared, and then pick a value a little larger, and rerun.

Note: the integral can be evaluated as before

$$I = \int_{x_{\min}}^{x_{\max}} f(x)\,dx = \frac{n_{\text{accept}}}{n_{\text{trial}}}\, w_{\max}\, I_a$$

where $I_a = \int_{x_{\min}}^{x_{\max}} f^a(x)\,dx$.

But the integral can be found more efficiently (ie. more accuratly for the same amount of CPU), by using the weights of all trial values:

$$\begin{aligned} I = \int_{x_{\min}}^{x_{\max}} f(x)\,dx &= \int_{x_{\min}}^{x_{\max}} \frac{f(x)}{f^a(x)} f_a(x)\,dx \\ &= \int_{x_{\min}}^{x_{\max}} w(x) f^a(x)\,dx \end{aligned}$$

But, $\int_{x_{\min}}^{x} f^a(x)\,dx / I_a = \lambda$, so $f^a(x)dx = I_a\,d\lambda$

$$I = \int_0^1 w(\lambda) I_a\,d\lambda = I_a \frac{1}{n_{\text{trial}}} \sum_i w = I_a \langle w \rangle$$

And the one standard deviation uncertainty is,

$$\left(\frac{\delta I}{I}\right)^2 = \frac{1}{n_{\text{trial}}} \frac{\langle w^2 \rangle - \langle w \rangle^2}{\langle w \rangle^2}$$

Generating random numbers according to the Gaussian distribution.

There are 2 ways to handle this special case.

1) Central limit theorem

"The sum of a large number of random numbers will approach a Gaussian distribution"

For a uniform istribution from 0 to 1,
the mean value is 1/2

and the variance is

$$\sigma^2 = \int (x-1/2)^2 \, dx = 1/12$$

So just add 12 random numbers and subtract 6.
The mean will be 0 and the variance will be 1.

This algorithm is coded in RG32 in the CERN library.

2) 2 D gaussian

Consider the 2 dimensional Gaussian dist:

$$f(x,y)\,dx\,dy = e^{-x^2/2}\,dx\,e^{-y^2/2}\,dy$$
$$= e^{-(x^2+y^2)/2}\,dx\,dy$$

Let $r^2 = x^2 + y^2$ and $\theta = \tan^{-1} y/x$

then, $dx\,dy = r\,dr\,d\theta$

$$f(r,\theta)\,dr\,d\theta = e^{-r^2/2}\,r\,dr\,d\theta$$

Let $u = r^2/2$ then $du = r\,dr$

$$f(u,\theta)\,du\,d\theta = e^{-u}\,du\,d\theta$$

So generate u between 0 and ∞ according to e^{-u} and θ between 0 and 2π (uniformly):

$$u = -\log(1-\lambda_1)$$
$$r = (2u)^{1/2}$$
$$\theta = 2\pi\lambda_2$$
$$x = r\cos(\theta)$$
$$y = r\sin(\theta)$$

This is coded in the CERN library function, RANNOR, but a method 5 times faster is available in NORRAN.

Multidimensional Simulation

Simulating a distribution in more than 1 dimension:

If the distribution is separable, the variables are uncorrelated, and hence each can be generated as before:

For example, if $f(x,y) = g(x)\,h(y)$

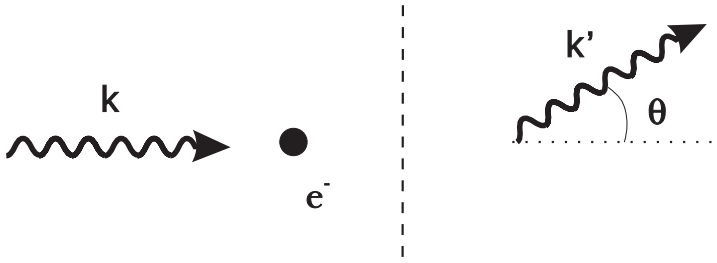
then generate x according to g(x) and y according to h(y).

Otherwise, the distributions along each dimension needs to be calculated:

$$D_x(x) = \int_{y_{min}}^{y_{max}} f(x,y)\,dy$$

Typically, you will need to choose an approximation of the distribution, $f^a(x,y)$ so the integrals, $\int f^a(x,y)dx$ and $\int f^a(x,y)dy$ are invertable. The weights for trial events are given by, $w = f(x,y) / f^a(x,y)$ and the integral can be evaluated as before, using the weights of all trial events. (Event = x and y pair)

Simulation of Compton Scattering



The energy of the final state photon is given by

$$k' = \frac{k}{1 + (k/m)(1 - \cos\theta)}$$

The differential cross section is:

$$\frac{d\sigma}{d\Omega} = \frac{\alpha^2}{2m^2}\left(\frac{k'}{k}\right)^2\left(\frac{k'}{k} + \frac{k}{k'} - \sin^2\theta\right)$$

O. Klein, Y. Nishina, Z. Physik, 52, 853 (1929)

The angular distribution of the photon is:

$$\sigma(\theta,\phi)\,d\theta\,d\phi = \frac{\alpha^2}{2m^2} \times \left(\left(\frac{k'}{k}\right)^3 + \left(\frac{k'}{k}\right) - \left(\frac{k'}{k}\right)^2 \sin^2\theta\right)\sin\theta\,d\theta\,d\phi$$

The azimuthal angle, ϕ, can be generated independantly from θ, by simply: $\phi = 2\pi\lambda_1$.

To generate the polar angle, θ, an approximation is needed. Note that for $k \gg m$, the cross section is sharply peaked at small angles. Also, note that $k' < k$, so the second term is the dominant term in the cross section formula.

A good approximation to the cross section is,

$$\begin{aligned} \sigma^a(\theta,\phi)\,d\theta\,d\phi &= \frac{\alpha^2}{2m^2}\left(\frac{k'}{k}\right)\sin\theta\,d\theta\,d\phi \\ &= \frac{\alpha^2}{2m^2}\left(1+\frac{k}{m}\,u\right)^{-1} du\,d\phi \end{aligned}$$

where $u = (1-\cos\theta)$.

u is generated according to:

$$u = \frac{m}{k}\left[\left(1+2\frac{k}{m}\right)^{\lambda_2} - 1\right]$$

Be careful when $k \ll m$; this procedure would not generate u properly, due to roundoff errors. Similarly, it is much better to generate $u = (1-\cos\theta)$ than $\cos\theta$, when there is a pole at $\theta = 0$.

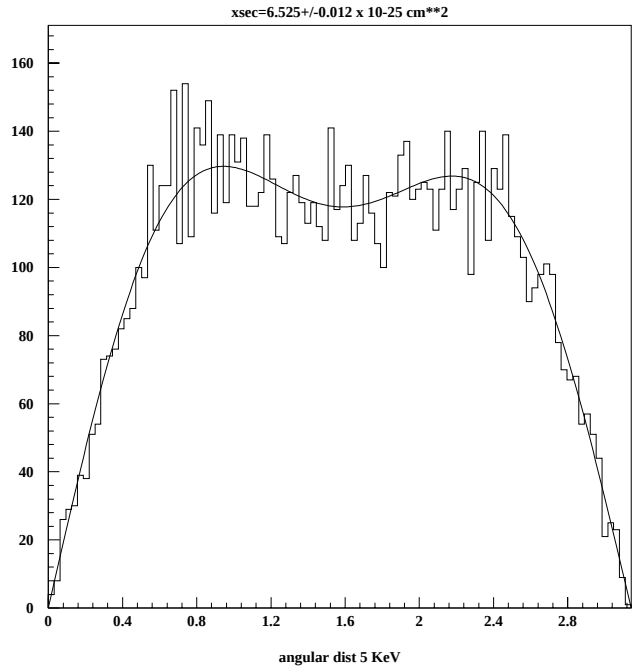
Exercise 9

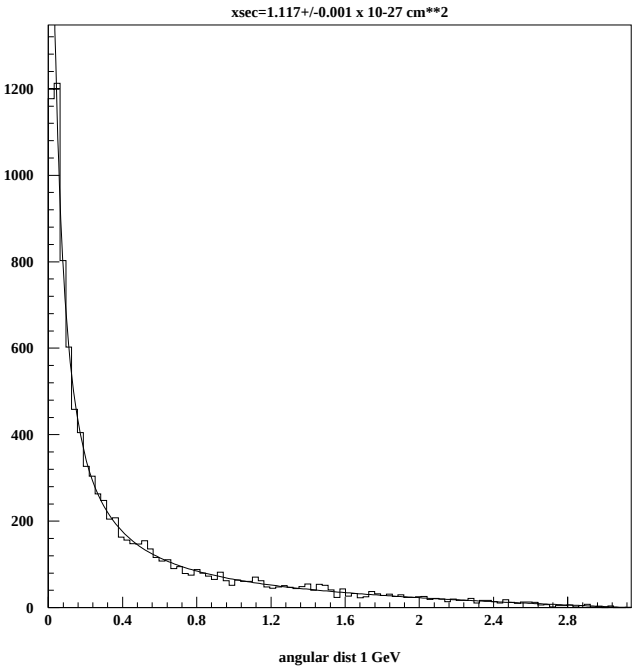
Write a Monte Carlo program that generates Compton scattering events.
The program should ask for the number of events to generate and the photon energy. Show the distribution of the scattering angle of the photon (compared to the Klein Nishina formula) and give the total cross section (ie. use the same program to evaluate the integral and its uncertainty) for the following four cases:

k=5 keV, k=2 MeV, k=1 GeV, k=1 TeV

in each case generate 10000 events.

Solution to Exercise 9:





Photon transport in matter

With this program, and others that simulate the photoelectric effect, pair production, etc., you could produce a program that simulates the interaction of photons with matter:

Algorithm:

Break path into small steps:

For each step decide if an interaction takes place (given the total cross section for each possible interaction).

Simulate the interaction, ie. give photon new momentum vector or possibly produce an e^+e^- pair, which then would be followed, etc.

Such programs already exist. For example:

EGS (SLAC)
GEANT (CERN)

You may use these to simulate photon transport in a particular sample you are testing or to simulate the response of your detector.

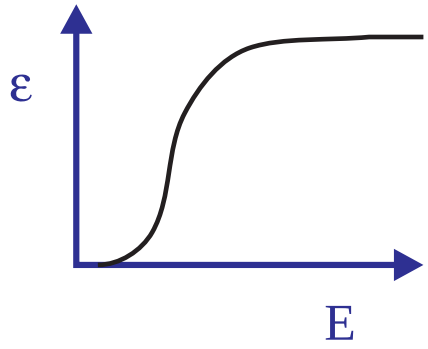
Detector response

It is often sufficient to simulate the general properties of your detector: efficiency, resolution, bias, offset.

Efficiency

From measurements from well understood sources, the effiency as a function of energy (and maybe position) can be found.

For example:

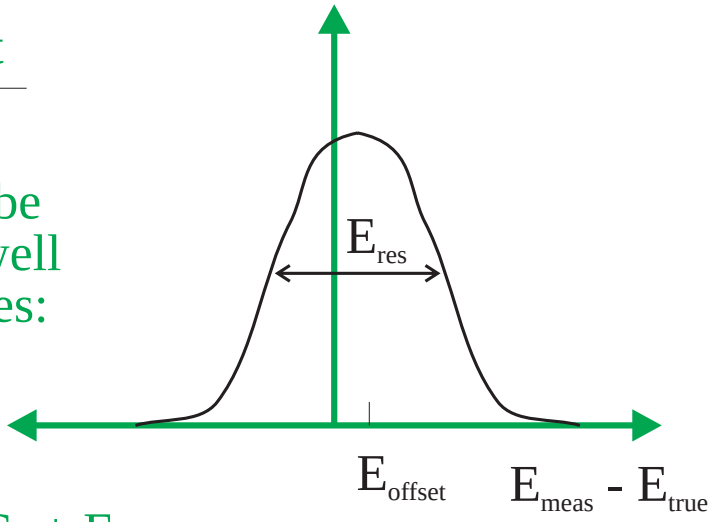


Once ε is known, select events that are missed with a probabilty of ε:

If $\lambda > \varepsilon$ then event is not observed.

Resolution, offset

Again, these can be measured using well understood sources:



$$E_{meas} = E_{true} + E_{res}\, G_{\lambda} + E_{offset}$$

Gaussian random number

Background, noise

Simulate the observed energy distribution when no source is present.

Part IV: Statistics for Physicists

Experimental Measurements and Uncertainties.

The result of an experiment that measures a parameter, x, is usually given by:

$$x = a \pm \sigma$$

a is the most probable value
σ specifies the uncertainty in the measurement (sometimes called the error)

The probability distribution of the measurement is usually assumed to be a Gaussian distribution. Hence the total probability that the true values lies within the range ($a-\sigma$, $a+\sigma$) is 68%.

This is called inverse probability by mathematicians. Physicists use the term probability for both direct and inverse probability.

Random and systematic uncertainties

Random (statistical) uncertainty:

due to the inherent randomness of the process being measured.

Systematic uncertainty:

due to the uncertainty in the behavior of the experimental apparatus

Example:

A measurement of the activity of a radioactive source: Count the number, N, of signals in a detector covering the solid angle Ω, with efficiency ε, over a period of time T.

Statistical uncertainty: uncertainty in the N. For large N, the probability distribution follows a Gaussian with $\sigma = N^{1/2}$

Systematic uncertainty: T, Ω, and ε are not known with perfect precision.

General comments:

- Two measurements often suffer common systematics, whereas never share statistical error, hence one often treats statistical and systematic errors separately.
- It is usually difficult to characterise the one standard deviation systematic uncertainty.
- Most experiments are designed so that the systematic uncertainty is smaller than the statistical uncertainty.

Determining systematic uncertainties

If an "off-the-shelf" instrument is used, the manufacture may quote an uncertainty based on the precision observed for many copies of their instrument.

Otherwise, some calibration of the instrument can be done to a precision limited by a statistical process.

Central limit theorem

If x_i are a set of n independent variables of mean μ and variance σ^2, then for large n:

$$y = \Sigma x_i / n$$

will tend to a Gaussian with mean = μ and variance = σ^2/n

This is true even if x_i come from distributions with different means μ_i and variance σ_i^2:

In this case mean = $\Sigma\mu_i/n$ and variance = $\Sigma\sigma_i^2/n$

Combining errors: general case

x and y are measurements of parameters with true values μ_x, μ_y then,

$$f(x,y) = f(\mu_x,\mu_y) + (x-\mu_x)\frac{\partial f}{\partial x} + (y-\mu_y)\frac{\partial f}{\partial y}$$

If the measurement of x and y is based on n measurements, the variance of f over those measurements is,

$$\begin{aligned}
\sigma_f^2 &= \frac{1}{n}\sum_i \left(f(x_i,y_i) - f(\mu_x,\mu_y)\right)^2 \\
&= \frac{1}{n}\sum_i (x_i-\mu_x)^2\left(\frac{\partial f}{\partial x}\right)^2 + \\
&\quad \frac{1}{n}\sum_i (y_i-\mu_y)^2\left(\frac{\partial f}{\partial y}\right)^2 + \\
&\quad \frac{2}{n}\sum_i (x_i-\mu_x)(y_i-\mu_y)\frac{\partial f}{\partial x}\frac{\partial f}{\partial y} \\
&= \sigma_x^2\left(\frac{\partial f}{\partial x}\right)^2 + \sigma_y^2\left(\frac{\partial f}{\partial y}\right)^2 + 2\mathrm{cov}(x,y)\frac{\partial f}{\partial x}\frac{\partial f}{\partial y}
\end{aligned}$$

The generalisation to m dimensions, where $f = f(x_1, x_2, ..., x_m)$ is:

$$\sigma_f^2 = \sum_{i=1}^{m} \sum_{j=1}^{m} \frac{\partial f}{\partial x_i} \frac{\partial f}{\partial x_j} V_{ij}$$

where, $V_{ij} =< (x_i - \bar{x}_i)(x_j - \bar{x}_j) >$ is the error matrix.

To change from variables $x_1, x_2, ..., x_m$ to $y_1, y_2, ..., y_n$ the error matrix for the y variables is

$$V_{ij}^y = \sum_{a=1}^{m} \sum_{b=1}^{m} \frac{\partial y_i}{\partial x_a} \frac{\partial y_j}{\partial x_b} V_{ab}^x$$

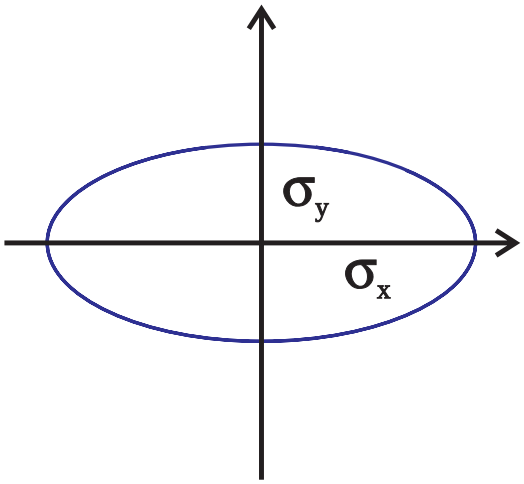
Example: Gaussian in 2 D

If x and y are two uncorrelated variables, then the probability distribution P(x,y) is just the product P(x)P(y). In the case of Gaussian distributions centred on the origin:

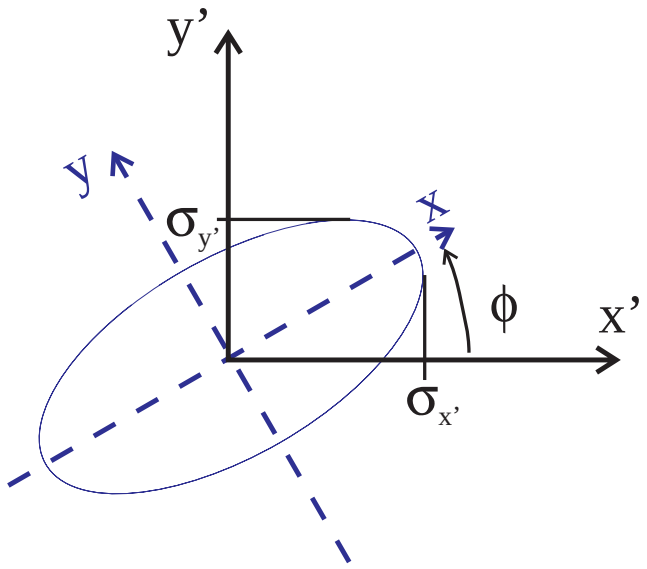
$$P(x,y)=(2\pi\sigma_x\sigma_y)^{-1}\exp(-(x^2/\sigma_x^{\,2}+y^2/\sigma_y^{\,2})/2))$$

The contour of constant probability in the x y plane is an ellipse whose axes are aligned with the x and y axis:

Example, contour at probability reduced by $e^{-1/2}$



To introduce a correlation, rotate the variables:



Then

$$P(x',y') = (2\pi\sigma_{x'}\sigma_{y'})^{-1}(1-\rho^2)^{-1/2} \times$$
$$exp(-(x'^2/\sigma_{x'}^2 + y'^2/\sigma_{y'}^2 - 2\rho x'y'/(\sigma_{x'}\sigma_{y'}))/(2(1-\rho^2)))$$

where $\rho = V_{x'y'}/(\sigma_{x'}\sigma_{y'})$ is the correlation coefficient

$|\rho| \leq 1$; and $\rho = 0$ corresponds to no correlation

$$\tan 2\phi = 2\rho\sigma_{x'}\sigma_{y'}/(\sigma_{x'}^2 - \sigma_{y'}^2)$$

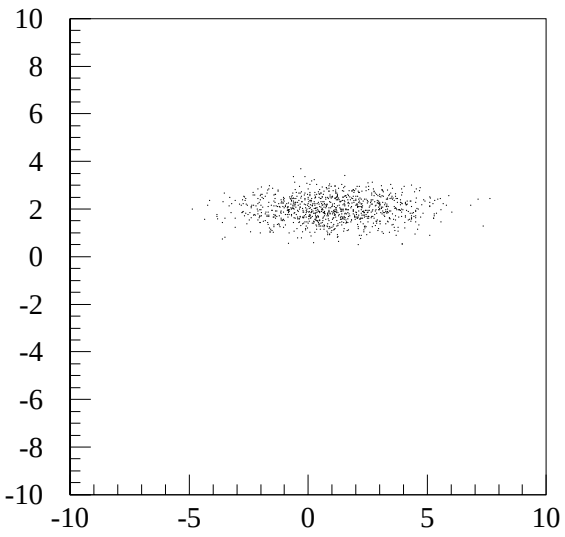
Exercise 10

Generate 1000 events of uncorrelated (x,y) values, each given by a Gaussian distribution with

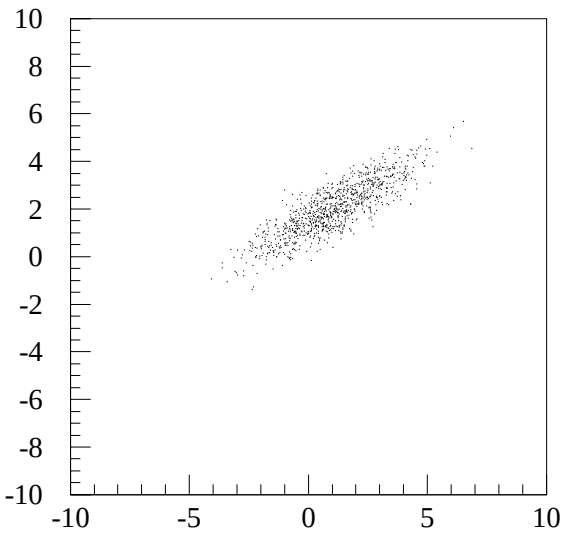
$$\mu_x = 1, \quad \sigma_x = 2, \quad \mu_y = 2, \quad \sigma_y = 0.5$$

Show a scatter plot of the data. Calculate the error matrix, V, and ρ and ϕ for this data set. Consider the function $f(x, y) = 5x + 8y$. Evaluate the variance of this function directly using the data set, and also by using the equation using the error matrix.

Now rotate the same events by $\phi = 30°$ (about the center of the distribution, not the origin), and repeat the above exercises.



Uncorrelated

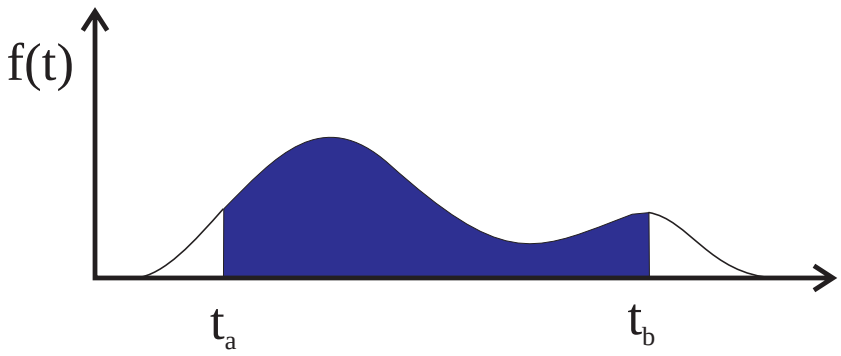


Correlated

Confidence intervals

If θ is a parameter we wish to determine from a sample of n measurements, $x_1, x_2, \ldots x_n$, we form an estimator, $t = t(x_1,x_2,\ldots,x_n)$.

t is a random variable. That is, if the experiment was repeated several times, we would find the the value of t would follow some distribution function, f(t):



If $\int_{t_a}^{t_b} f(t)dt=\gamma$, then $P(t_a \le \theta \le t_b)=\gamma$

Some say "The probability that the true value, θ, is within the range $[t_a,t_b]$ is γ."

Not really!

The distribution of the true value, θ, is a delta function at $t=\theta$. (ie. θ is not a random variable). Hence the probability that the true value is within the range $[t_a,t_b]$ is 1 if $t_a \le \theta \le t_b$ and is 0 otherwise.

Proper interpretation of $P(t_a \le \theta \le t_b)=\gamma$ is that if you have a large number of samples of size n (ie. the experiment is repeated many times) then $t_a \le \theta \le t_b$ for 100γ% of the experiments.

Example: Gaussian distribution

μ is an unknown quantity, x is a measurement of μ: it is a random variable that has a normal distribution about a mean value μ, with variance σ^2

Then, $z=(x-\mu)/\sigma$ is a random variable distributed according to the unit Gaussian, $G_{(0,1)}(z)$

Then, for example,

$$P(-2 \le (x-\mu)/\sigma \le 2) = \int_{-2}^{2} G_{(0,1)}(z)\, dz = 0.954$$

which states the probability of $|(x-\mu)/\sigma|<2$ is 95.4%

The expression can be rewritten as:

$$P(x-2\sigma \le \mu \le x+2\sigma) = 0.954$$

which apparently treats μ as a random variable and $[x-2\sigma, x+2\sigma]$ as a fixed interval.

Instead:

$[x-2\sigma, x+2\sigma]$ is a random interval, and the statement says that the probability the interval contains μ is 0.954.

Gaussian confidence intervals (1D)

The integral $I = \int_{-c}^{c} G_{(0,1)}(z)\, dz$ is given below:

c	I
1	0.683
1.5	0.866
1.64	0.900
1.96	0.950
2.0	0.955
2.58	0.990
3.0	0.997

What if some measurements are in a non physical region? (eg. $m_\nu^2<0$).

Classical approach

To determine a confidence interval, proceed as before:

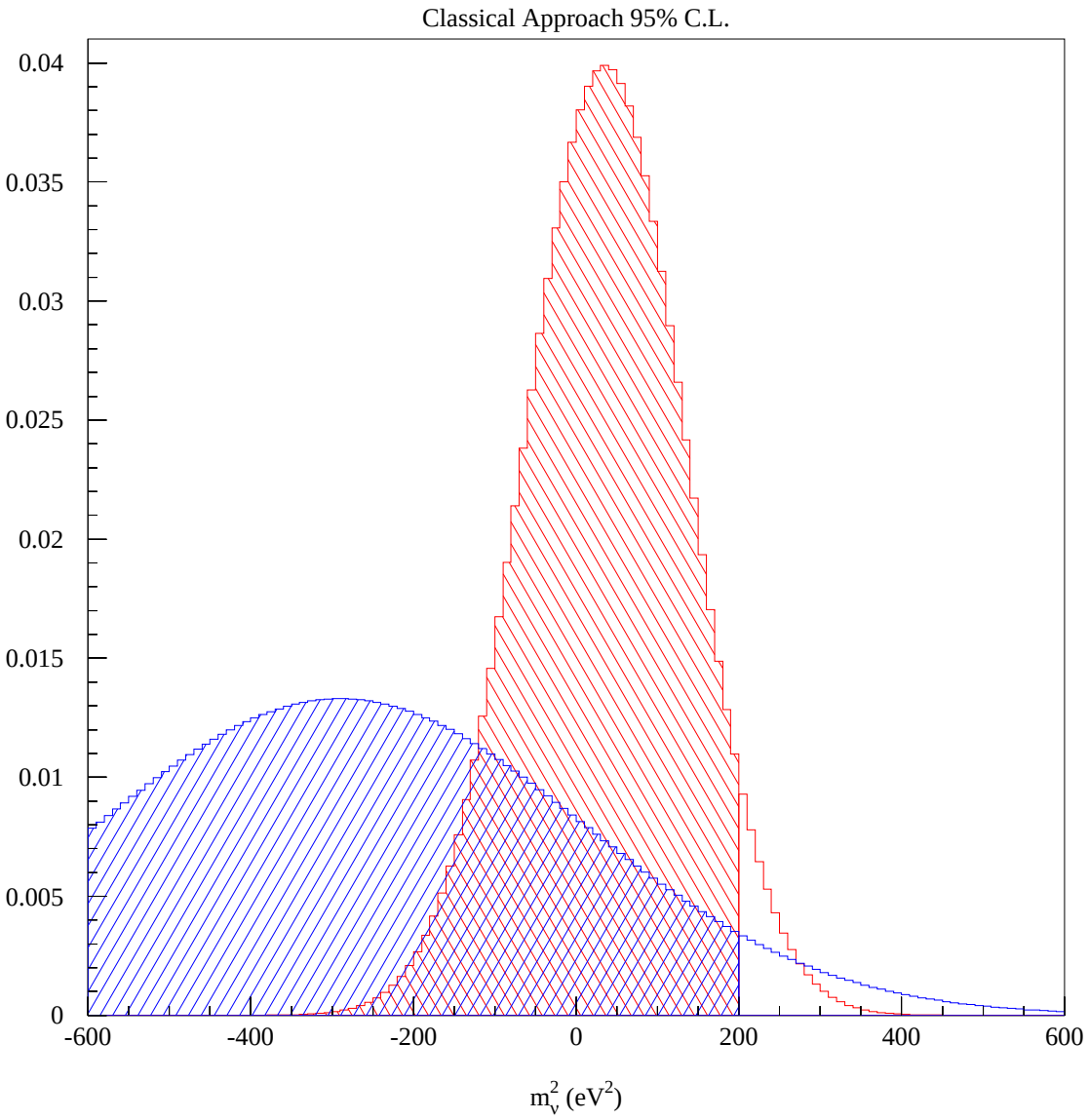
$$\int_{-\infty}^{m_\gamma^2} G(m^2)dm^2 = \gamma$$

The probability of the interval $[-\infty,m_\gamma^2]$ to contain the true m_ν^2 is γ.

Quote result as $m_\nu<(m_\gamma^2)^{-1/2}$ at 100γ % confidence level.
One usually chooses γ to be large enough so that $m_\gamma^2>0$!.

Note that a precise experiment and an imprecise one with a statistical fluctuation can give the same limit!

Classical Approach



Bayesian approach

Multiply the result of the experiment, $L(x|\theta)$, by the prior belief function, $Q(\theta)$,
where $Q(\theta)=0$ in the unphysical region,
in order to obtain the posterior density function,

$$R(x|\theta)=L(x|\theta)Q(\theta)$$

The particle data group suggests this method with $Q(\theta)$ taken as a constant in the physical region. and $R(x|\theta)$ is normalised so that

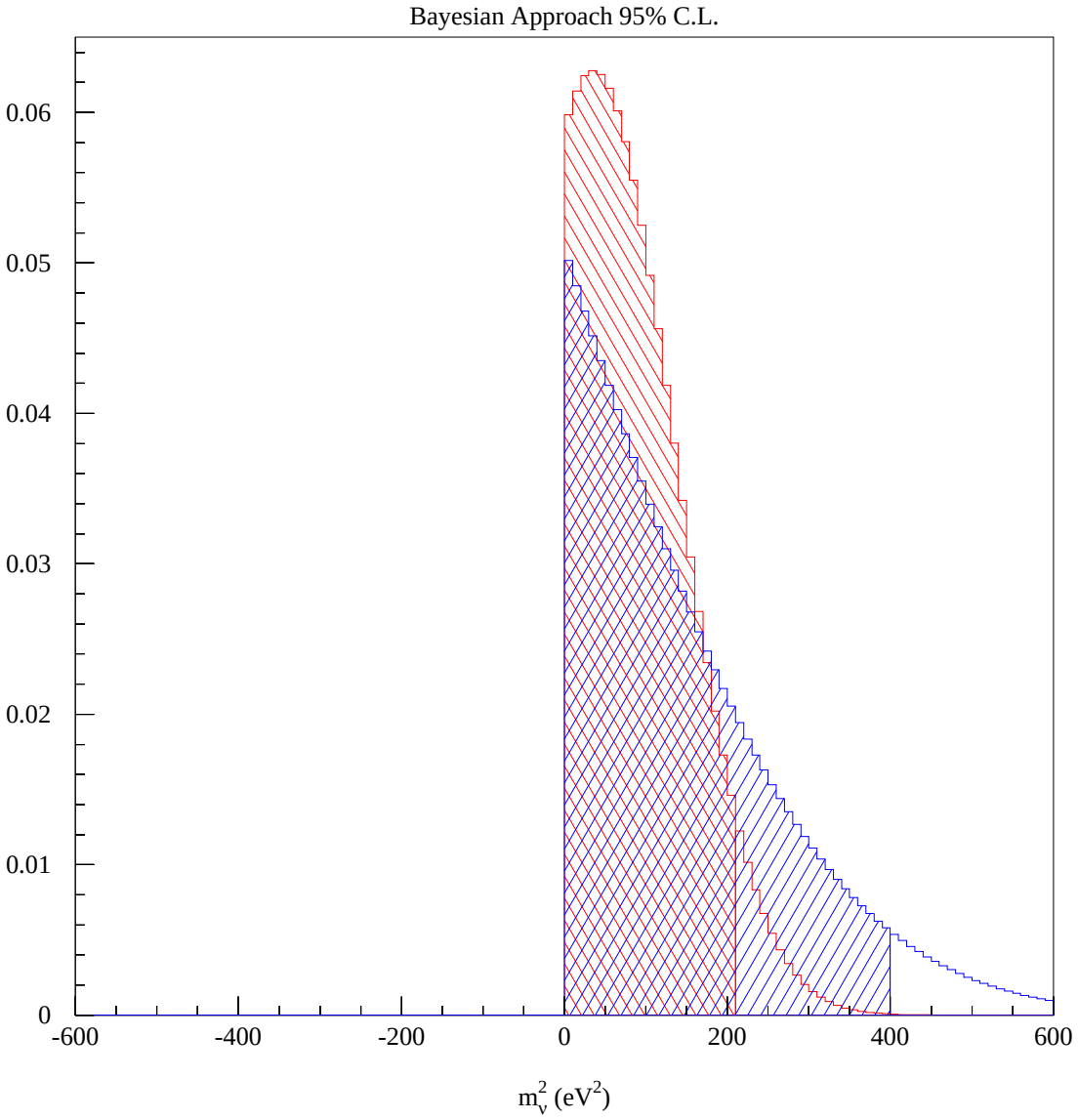
$$\int R(x|\theta)\, dx = 1$$

This is a conservative approach, in that the probability that the range $[0,m_\gamma^2]$ contains the true m_ν^2 is $> \gamma$.

But it is not possible to combine the results of experiments that just quote a mass interval and confidence level. It is better to quote m^2 and σ_{m^2}.

See F.James, M.Roos, Phys.Rev.D44, 299 (1991)

Bayesian Approach



Estimation of parameters

General Problem:

Given n observations, x_i, one wants to describe the underlying (or parent) distribution. The form of the parent distribution may be known but may have a number of unknown parameters, θ_j. The n observations should be used to determine the parameters, θ_j, as accurately as possible.

Definitions:

estimator: a function, t, of the observations used to determine the unknown parameter θ.

estimate: the resulting value of the estimator, $\hat{\theta}$

Properties of good estimators.

A good estimator must have the following properties:

- should not deviate from true parameter value in the limit of large n
- accuracy should improve with larger n

In addition:

- should be centered around true parameter value for all n (For example, $x' = \Sigma x_i/(n-30)$ does not satisfy this criterion.)
- should exhaust all the information in the data x_i
- should have the minimum possible variance (For example, the mean has a smaller variance than the median.)
- should be robust so as not to be sensitive to background or outliers

Maximum Likelihood Method

1. It is very powerful and general method of parameter estimation when the functional form of the parent distribution is known.

2. For large samples the maximum likelihood (ML) estimators are normally distributed, hence the variances of the estimates are easy to determine.

3. Even for small samples, the ML estimators possess most of the "good" properties.

Likelihood function

Given n measurements, x_i, of a quantity with probability density function $f(x|\theta)$

(ie. $\int_{x\ min}^{x\ max} f(x|\theta)\, dx = 1$ for all θ)

Then, $L(x_1,x_2,...,x_n|\theta) \equiv \Pi\ f(x_i,\theta)$

Each x_i could also denote a set of measurements, and θ could be a set of parameters.

The estimate $\hat{\theta}$ is that value which maximises L

Since L and $\ln L$ attain their maximum values at the same point, one usually uses $\ln L$, since sums are easier to work with than products:

$$\ln L = \sum_{i=1}^{n} \ln(f(x_i|\theta))$$

Normally, the point of maximum likelihood is found numerically.

Simple example: Gaussian parent distribution

If the parent distribution of x_i is $G(\mu,\sigma_i^2)$ then,

$$L(x_i|\mu,\sigma_i) = \Pi\, (2\pi)^{-1/2}\sigma_i^{-1}\exp(-(x_i-\mu)^2/2\sigma_i^2)$$

To estimate μ,

$$\partial \ln L/\partial\mu\,|_{\hat{\mu}} = \partial/\partial\mu\, \Sigma(-\ln(2\pi\sigma_i^2)/2 - (x_i-\mu)^2/2\sigma_i^2)\,|_{\hat{\mu}} = 0$$

so, $\Sigma(x_i-\hat{\mu})/\sigma_i^2 = 0 \quad \Rightarrow \hat{\mu} = \Sigma\,(x_i/\sigma_i^2) / \Sigma\,(1/\sigma_i^2)$

so the ML estimator of the population mean, is the weighted mean.

Properties of Maximium Likelihood estimators

Invariant under parameter transformation

The choice of parameterisation is arbitrary:

if θ is the parameter, $\partial L/\partial\theta\,|_{\hat{\theta}} = 0$

If instead some function of θ is used, $t(\theta)$

$$\partial L/\partial\theta|_{\hat{\theta}} = (\partial L/\partial t\ \partial t/\partial\theta)|_{\hat{\theta}} = 0 \Rightarrow \partial L/\partial t\,|_{\hat{\theta}} = 0$$

Consistent

estimators converge on true parameter

Unbiased

sometimes biased for finite samples. Note: $\hat{\theta}$ may be unbiased but $\hat{t(\theta)}$ may be biased.

Efficient

if a sufficient estimator exists, the ML method produces it, and this will give the minimum attainable variance.
ie. You can't do better than this.

Variance of ML estimates:

$$\mathcal{L}(\vec{x},\vec{\theta}) = L(x_1,...,x_n|\theta_1,...\theta_k) = \prod_i^n f(x_i,\vec{\theta})$$

If the estimates can be written as functions of x_i, then the error matrix for $\hat{\theta}$ is

$$V_{ij}(\hat{\vec{\theta}}) = \int_\Omega (\hat{\theta}_i - \theta_i)(\hat{\theta}_j - \theta_j)\mathcal{L}(\vec{x},\vec{\theta})d\vec{x}$$

which could be found without using any data.

If only a single parameter (and sufficient)

$$V(\hat{\theta}) = \left(\frac{-\partial^2 \ln \mathcal{L}}{\partial\theta^2}\right)^{-1}_{\theta=\hat{\theta}}$$

Note: this is easily shown for normally distributed estimates:

$$\mathcal{L} \approx \exp\left(-\frac{(\theta-\hat{\theta})^2}{2V(\hat{\theta})}\right)$$

$$\frac{\partial^2 \ln \mathcal{L}}{\partial\theta^2} = -\frac{1}{V(\hat{\theta})}$$

Example: variance of the weighted mean

$$\hat{\mu} = \frac{\sum_{i=1}^{n} \frac{x_i}{\sigma_i^2}}{\sum_{i=1}^{n} \frac{1}{\sigma_i^2}}$$

Recall the log likelihood function is

$$\ln \mathcal{L} = \sum_{i=1}^{n} \left[-\frac{1}{2} \ln(2\pi\sigma_i^2) - \frac{1}{2} \left(\frac{x_i - \mu}{\sigma_i} \right)^2 \right]$$

Then, the variance is,

$$V(\hat{\mu}) = \left(-\frac{\partial^2 \ln \mathcal{L}}{\partial \mu^2} \right)^{-1}_{\mu=\hat{\mu}} = \left(\sum_{i=1}^{n} \frac{1}{\sigma_i^2} \right)^{-1}$$

In the case where $\sigma_i = \sigma$, $\Delta\mu = \sigma/\sqrt{n}$

For multiparameter large sample estimates,

$$\begin{aligned} V_{ij}^{-1}(\hat{\vec{\theta}}) &= \left(-\frac{\partial^2 \ln \mathcal{L}}{\partial\theta_i \partial\theta_j} \right)_{\vec{\theta}=\hat{\vec{\theta}}} \\ &= n \int_\Omega \frac{1}{f} \left(\frac{\partial f}{\partial \theta_i} \right) \left(\frac{\partial f}{\partial \theta_j} \right) d\vec{x} \end{aligned}$$

Graphical determination of the ML estimate and error.

This method can be used for 1 or 2 parameters when the ML estimate and variance cannot be found analytically.

One parameter

- Plot ln$L$ as a function of θ and read off the value of $\hat{\theta}$ at the position where L is the largest.
- Sometimes there is more than one peak. Take the highest one.
- Uncertainty is deduced from the positions where ln L is reduced by an amount 1/2

Note that for a Gaussian LF,

$$\ln L = \ln L_{max} - (\theta - \hat{\theta})^2/2V(\hat{\theta})$$

so,

$$\ln L(\hat{\theta} + V(\hat{\theta})^{1/2}) = \ln L_{max} - 1/2$$

The formula,

$$\ln L(\hat{\theta} + V(\hat{\theta})^{1/2}) = \ln L_{max} - 1/2$$

even applies for a non-Gaussian likelihood function.

Proof:

Change variables to $g(\theta)$, which produces a Gaussian distribution. L is invariant under parameter transformations.

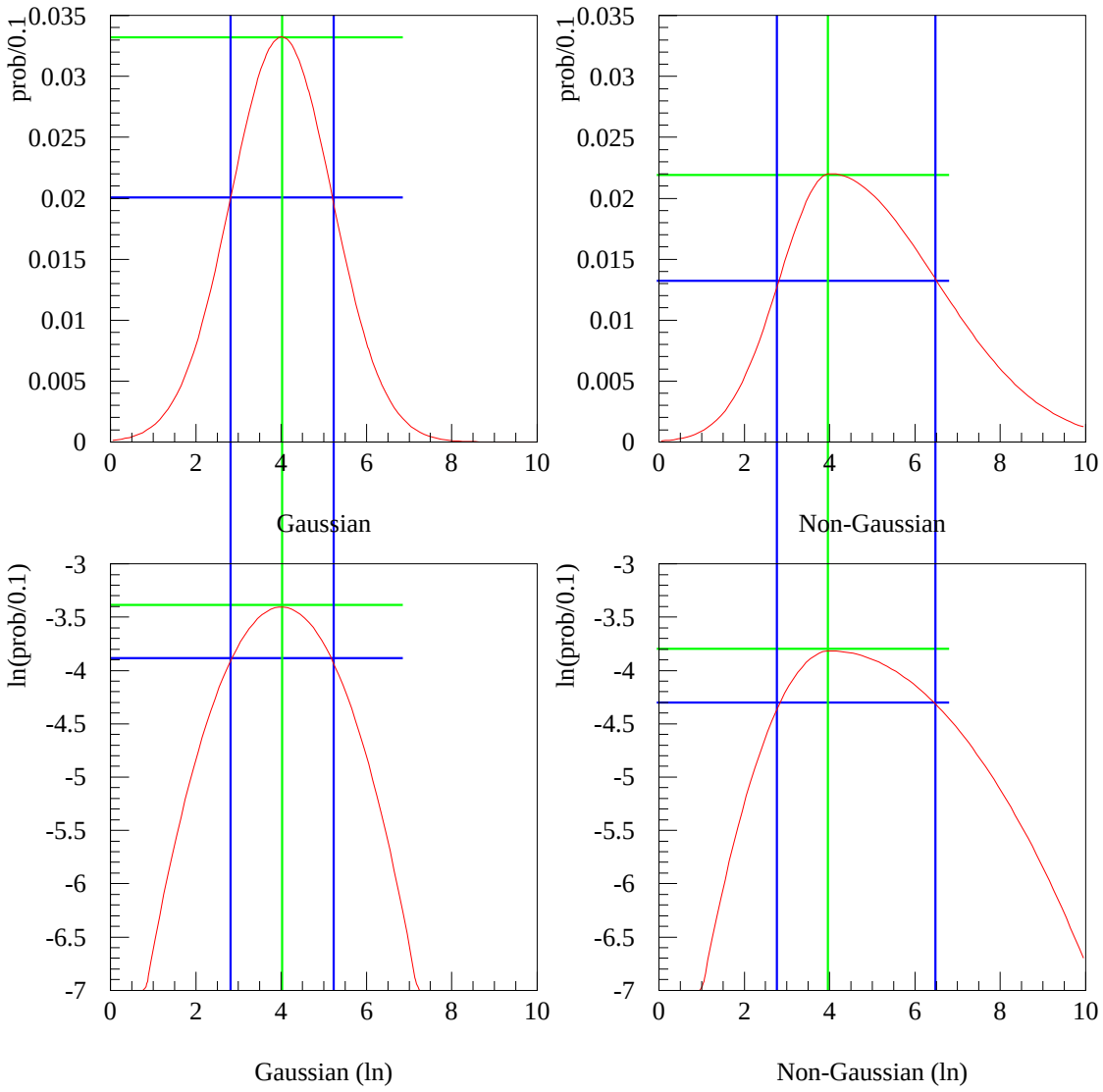
If the likelihood function is asymmetric (typically the case for small sample size) then an asymmetric interval about the most likely value may result. In this case the measured result usually quoted as:

$$1.23\ ^{+\ 0.12}_{-\ 0.09}$$

for example.

Examples of Likelihood distributions

Central Values and 1 σ intervals are shown:



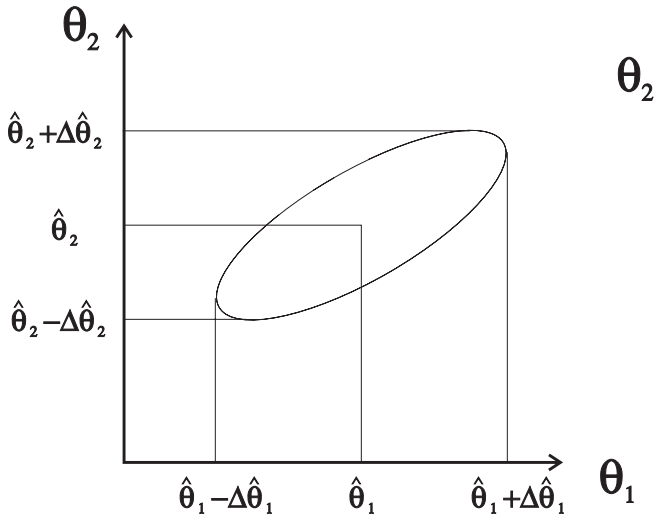
Two parameters

Given $L(x|\theta_1,\theta_2)$, plot contours of constant likelihood in the θ_1,θ_2 plane.

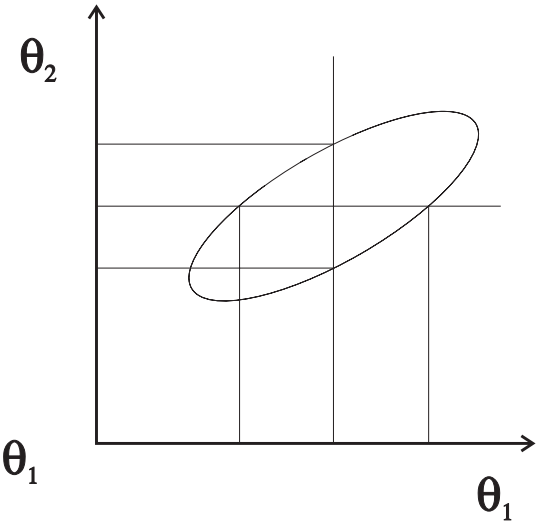
Often there may be more than one maximum, if one isn't much larger than all the rest, then an additional (different) experiment may be needed to decide which of the peaks to take.

To find the uncertainty, plot the contour with $\ln L = \ln L_{max} - 1/2$ and look at the projection of the contour on the 2 axes.

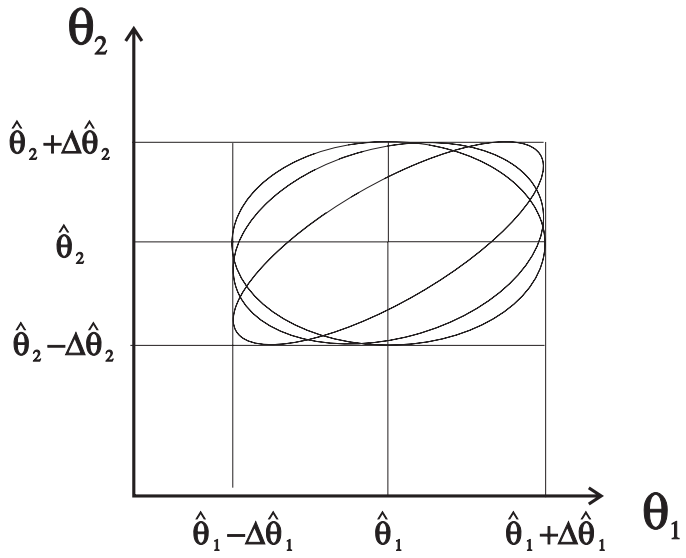
correct method



incorrect method



Using the correct method, the uncertainties do not depend on the correlation of the variables:



For a two dimensional Gaussian LF, the probability that the range $(\hat{\theta}_1-\Delta\hat{\theta}_1, \hat{\theta}_1+\Delta\hat{\theta}_1)$ contains θ_1 is still 0.683.

The probability that the ellipses of constant $\ln L = \ln L_{max}$ - a contains the true point θ_1 and θ_2, is given in the following table:

a	σ	γ
0.5	1	0.393
2.0	2	0.865
4.5	3	0.989

If the LF contours are very irregular so that a transformation to a 2D Gaussian is not possible, or if the contour consists of more than one closed curve, it is probably better to show the LF contour directly, instead of quoting any intervals.

If there are 3 or more parameters, larger samples are necessary to have the LF to be Gaussian.

A general maximisation (minimisation) program will be necessary to find the estimate and the uncertainties.

A good program widely used in HEP is MINUIT, in the CERN library.

The routines, BRENT and POWELL, from *Numerical Recipies* can be used for simple problems.

Generalized Likelihood function

If the total number of events expected is a function of θ, $\nu=\nu(\theta)$, and n events are observed, then

$$L(n,x|\theta) = P(n,\nu)L(x|\theta)$$

where $P(n,\nu)=\nu^n e^{-\nu}/n!$ is the Poisson distriubtion.

In problems where the shape of $f(x|\theta)$ is of primary interest, this modification will gain little in the precision of $\hat{\theta}$.

Using likelihood on binned data

If the sample is very large, and $f(x|\theta)$ is complex, computation can be reduced by grouping the sample into bins, and write L as the product of the probability of finding n entries in each bin i (multinomial distribution)

$$L(n_1,n_2,...,n_m|\theta)=n!\Pi(n_i!)^{-1}p_i^{n_i}$$

p_i is the probability for bin i: $p_i = \int_{\Delta x_i} f(x|\theta)dx$
Since L depends on θ only through p_i, find maximum of L through $\ln L = \Sigma n_i \ln p_i(\theta)$.
Rather obvious when you look at it!

There will be some loss of information by binning the data, but as long as the variation in f across each bin is small, there should be no great loss in precision of $\hat{\theta}$.

Using weighted events

Recall that if the efficiency, $\varepsilon_i<1$, then you need to correct each event by the weight, $w_i=1/\varepsilon_i$. Then

$$\ln L(x|\theta) = \Sigma\, w_i \ln f(x_i|\theta)$$

Combining results from two experiments

Suppose two independent experiments designed to measure the same parameter θ, result in two measurements x and y. If $L(x|\theta)$ and $L(y|\theta)$ are approximately Gaussian, then just use the weighted average.

Otherwise, use the product of the likelihood functions:

$$L(x,y|\theta) = \prod_i f_1(x_i|\theta)\prod_j f_2(x_j|\theta)=L(x|\theta)L(y|\theta)$$

Exercise 11

Consider an experiment that is set up to measure the lifetime of an unstable nucleus, N, using the chain reaction,

$$A \to N e \bar{\nu} \quad , \quad N \to X p$$

The creation and decay of N is signaled by the electron and proton.

The lifetime of each N, which follows the p.d.f, $f = \frac{1}{\tau} e^{-t/\tau}$, is measured from the time between observing the electron and proton with a resolution of σ_t.

The expected probability density function is the convolution of the exponential decay and the Gaussian resolution:

$$f(t|\tau, \sigma_t) = \int_0^\infty \frac{e^{-\frac{(t-t')^2}{2\sigma_t^2}}}{\sqrt{2\pi}\sigma_t} \frac{e^{-\frac{t'}{\tau}}}{\tau} dt'$$

$$= \frac{1}{2\tau} \exp\left(\frac{\sigma_t^2}{2\tau^2} - \frac{t}{\tau}\right) \operatorname{erfc}\left(\frac{\sigma_t}{\sqrt{2}\tau} - \frac{t}{\sqrt{2}\sigma_t}\right)$$

Exercise 11 (continued)

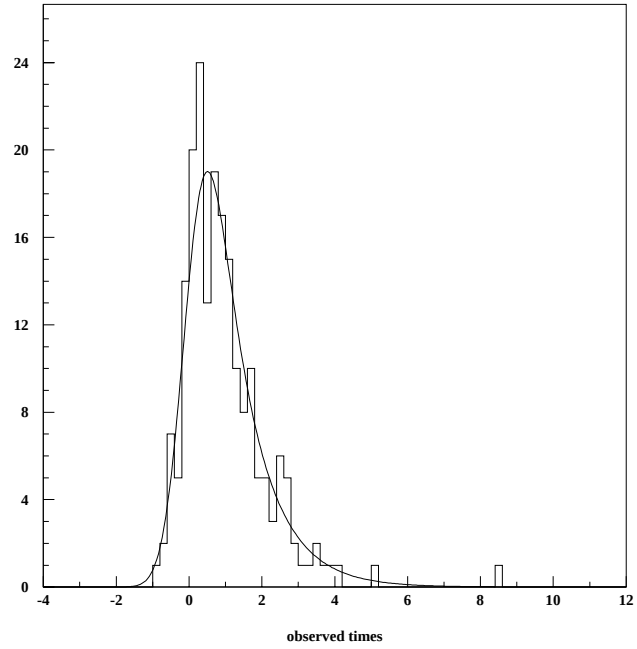
Generate 200 events with $\tau = 1$ s and σ_t=0.5 s. (Use the inversion technique followed by a Gaussian smearing.) Use the maximum likelihood method to find $\hat{\tau}$ and the uncertainty, $\sigma_{\hat{\tau}}$. Plot the likelihood function, and the resulting p.d.f. for the measured times compared to a histogram containing the data.

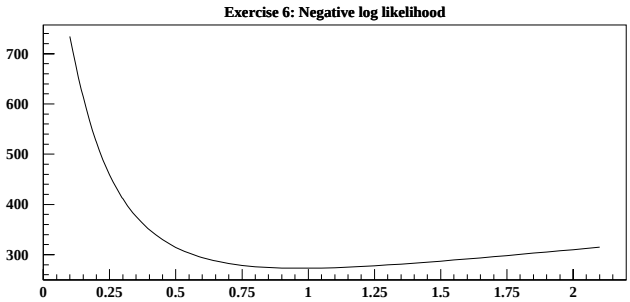
Automate the ML procedure so as to be able to repeat this exercise 100 times, and plot the distribution of $(\hat{\tau} - \tau)/\sigma_{\hat{\tau}}$ for your 100 experiments and show that it follows a unit Gaussian.

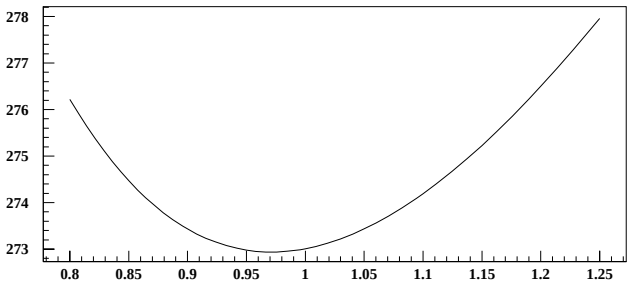
For 1 data sample, assume that σ_t is unknown, and show a contour plot in the τ, σ_t plane with constant likelihood,

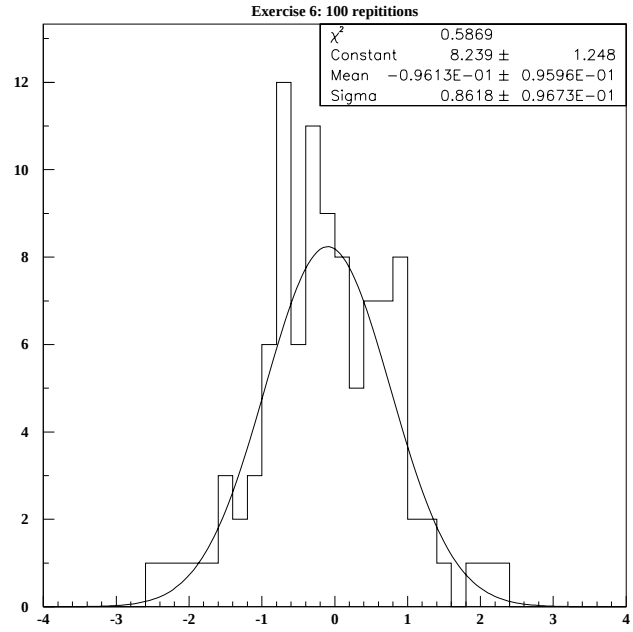
$$\ln \mathcal{L} = \ln \mathcal{L}_{\max} - \frac{1}{2}$$

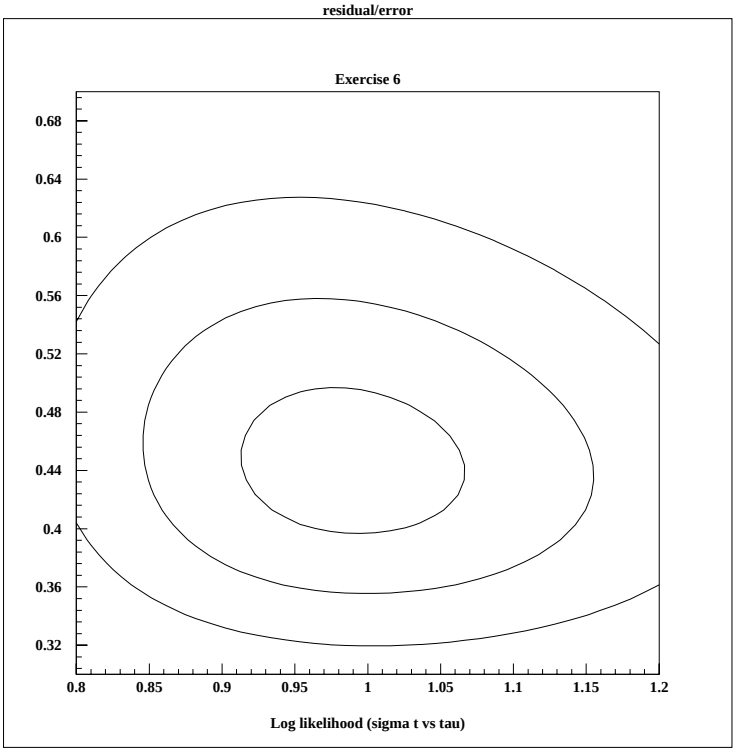
Solution to Exercise 11:








Exercise 6: 100 repititions
χ² 0.5869
Constant 8.239 ± 1.248
Mean −0.9613E−01 ± 0.9596E−01
Sigma 0.8618 ± 0.9673E−01
residual/error


Exercise 6
Log likelihood (sigma t vs tau)

The Least Squares Method

- The most frequently used method, but has no general optimal properties to recommend it.
- For problems where the parameter dependence is linear, the Least Squares (LS) method produces unbiased estimators of minimum variance.

Method

At observational points $x_1,...,x_N$ we measure experimental values of $y_1,...,y_N$. The true functional form is defined by L parameters, $f_i = f_i(\theta_1,...,\theta_L)$

To find the parameter estimates, $\theta_1,...,\theta_L$, minimise $X^2=\Sigma w_i(y_i-f_i)^2$, where w_i is the weight that expresses the accuracy of y_i.

If constant accuracy, $w_i=1$;
if accuracy for y_i given by σ_i, $w_i=1/\sigma_i^2$;
if y_i represents a Poisson distributed random number, $w_i=1/f_i$ (or sometimes $w_i=1/y_i$).

If the observations are correlated then,

$$X^2 = \sum_{i=1}^{N} \sum_{j=1}^{N} (y_i - f_i) V_{ij}^{-1} (y_j - f_j)$$

The x_i values are assumed to have no uncertainty associated with them.

If y_i are Gaussian distributed then LS is equivalent to the ML method.

If in addition, the observables are linear functions of the parameters, then X^2_{min} will follow the χ^2 distribution.

χ^2 distribution

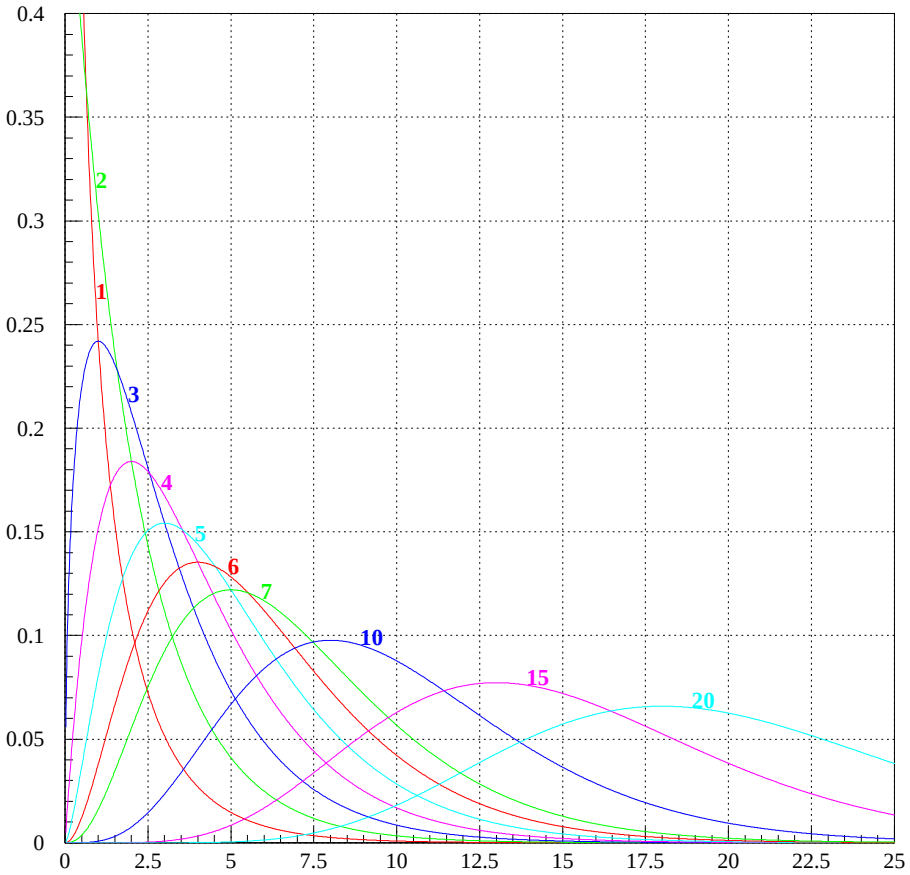
If x_i (i=1,...,N) are distributed according to the Gaussian with mean μ_i and variance σ_i^2, the quantity, $\chi^2 \equiv \Sigma (x_i - \mu_i)^2 / \sigma_i^2$ has the p.d.f. given by,

$$f(\chi^2|N) = 2^{-N/2} \Gamma^{-1}(N/2) \chi^{2(N/2-1)} e^{-\chi^2/2} \qquad 0 \le \chi^2 \le \infty$$

where N is called the number of degrees of freedom.

Properties of the χ^2 distribution

- If r_N has the distribution, $f(\chi^2|N)$, then $r_{N_1} + r_{N_2}$ will have the distribution, $f(\chi^2|N_1 + N_2)$.
- The maximum of $f(\chi^2|N)$ occurs at $N - 2$ (and at 0 for $N = 1$).
- The mean is N and the variance is $2N$
- For large N, it approaches the Gaussian distribution.

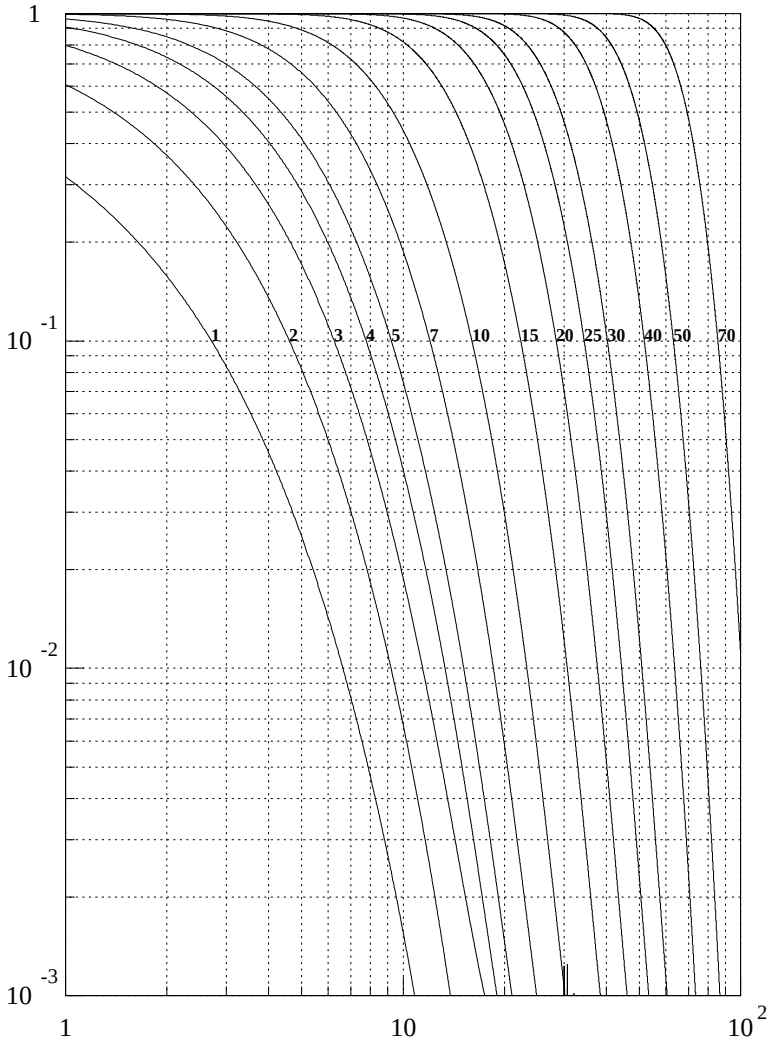


Cumulative χ^2 distribution

$$F(\chi^2_\alpha|N) = \int_0^{\chi^2_\alpha} f(\chi^2|N)\, d\chi^2 = 1 - \alpha$$

The p.d.f. of F is uniform over $[0, 1]$ (of course!).

The following graph shows $\alpha = 1 - F(\chi^2_\alpha)$, for various N



Linear Least Squares Model

If the observables are linear functions of the unknown parameters and the weights are independent of the parameters, then the LS method has an exact solution that can be written in closed form. The estimates are unique, unbiased and have the minimum variance.

Example: Unweighted straight line fit

Data points: $(x_1,y_1), (x_2,y_2),...,(x_N,y_N)$

model: $f_i = \theta_1 + x_i\theta_2$

minimise X^2 to find the estimates:

$$\hat{\theta}_1 = \frac{\Sigma x_i^2 \, \Sigma y_i - \Sigma x_i y_i \, \Sigma x_i}{N \, \Sigma x_i^2 - (\Sigma x_i)^2}$$

$$\hat{\theta}_2 = \frac{N \, \Sigma x_i y_i - \Sigma x_i \, \Sigma y_i}{N \, \Sigma x_i^2 - (\Sigma x_i)^2}$$

General Weighted Linear Case

The N measured quantities are given by $\vec{y}$, the expectations by $\vec{f}$ which depend on the L parameters, $\vec{\theta}$; $f_i = A_{i\ell}\theta_\ell$.

For example, $f_i = \theta_1 + x_i\theta_2 + x_i^2\theta_3$

If the error matrix for $\vec{y}$ is given by V, minimise

$$X^2 = (\vec{y} - A\vec{\theta})^T V^{-1} (\vec{y} - A\vec{\theta})$$

which has the solution for the estimates and error matrix,

$$\hat{\vec{\theta}} = (A^T V^{-1} A)^{-1} A^T V^{-1} \vec{y} \qquad V(\hat{\vec{\theta}}) = (A^T V^{-1} A)^{-1}$$

Polynomial fitting

For high order polynomials (≥ 6), roundoff errors may cause serious numerical inaccuracies. It is better to use orthogonal polynomials, since the error matrix is diagonal and easy to invert.

Take as a model,

$$f_i = \sum_{\ell=1}^{L} \xi_\ell(x_i)\omega_\ell$$

where ξ_ℓ are orthogonal over the observables,

$$\sum_{i=1}^{N} \xi_k(x_i)\xi_\ell(x_i) = \delta_{k\ell}$$

Degrees of Freedom

If y_i are Gaussian distributed with true mean η_i and variance σ_i^2, then

$$X^2 = \sum_{i=1}^{N} \left(\frac{y_i - \eta_i}{\sigma_i} \right)^2$$

follows a χ^2 distribution with N degrees of freedom.

But η_i are unknown. If we instead use $\hat{\eta}_i$ (the result from the LS minimisation to a linear model with L independant parameters), then

$$X^2_{\text{min}} = \sum_{i=1}^{N} \left(\frac{y_i - \hat{\eta}_i}{\sigma_i} \right)^2$$

is distributed according to the χ^2 distribution with $N - L$ degrees of freedom.

This can been proven by showing that for a linear model, X^2_{min} can be expressed as a sum of $(N - L)$ independant terms each being the square of a Gaussian distributed variable.

Non-linear Least Squares Model

- one cannot write down a closed form solution; must find minimum by numerical methods
- usually produces biassed estimates and $X^2_{\rm min}$ follows an unknown distribution, but for large N approachs the χ^2 distribution.

Estimate of σ^2 in the linear model

Recall solution of linear model:

$$\hat{\vec{\theta}} = (A^T V^{-1} A)^{-1} A^T V^{-1} \vec{y}$$

so to determine the estimates, V needs only be known to a multiplicative factor. That is, writing $V(\vec{y}) = r^2 V_r(\vec{y})$, only V_r needs to be known, and r is some unknown constant. But in order to determine the variance of the estimates, $V(\hat{\vec{\theta}}) = (A^T V^{-1} A)^{-1}$, V has to be known absolutely.

Since $X^2_{\rm min}$ follows $f(\chi^2|N-L)$, one can estimate the value of r^2 from the data using,

$$r^2 = Q^2_{\rm min}/(N-L)$$

where, $Q^2_{\rm min}$ is $X^2_{\rm min}$ with V replaced by V_r, and where L is the number of parameters in the linear model.

Goodness of Fit

Since $X^2_{\min}$ follows a known (χ^2) distribution (for a linear model with Gaussian distributed observables), the value of $X^2_{\min}$ obtained in a particular case is a measure of the agreement between the fitted quantities $\hat{\eta}$ and the measurements y.

A larger $X^2_{\min}$ corresponds to a poorer agreement. The probability of obtaining a value of $X^2_{\min}$ or larger is

$$P_{X^2_{\min}} = \int_{X^2_{\min}}^{\infty} f(\chi^2|N)\, d\chi^2 = 1 - F(X^2_{\min}|N) = \alpha$$

where F is the cumulative distribution.

$P_{X^2_{\min}}$ has a uniform distribution over [0,1].

- If in a series of similar minimisations, $P_{X^2_{\min}}$ is non-uniform, then the model or the data (or both) may be flawed.
- If $P_{X^2_{\min}}$ peaks at low (high) probability, the measurement uncertainties may have been over- (under-) estimated
- If large value of $P_{X^2_{\min}}$ is due to one of the measurements, should examine that measurement in detail.

Application of LS method to binned data

If the data is split into N bins, with n_i entries in bin i, and $p_i(\vec{\theta})$ is the probability of an event to populate bin i,
then the expected number of events in each bin is given by,

$$f_i = n\, p_i \quad \text{where } n = \sum_{i=1}^{N} n_i$$

If the number of bins is large enough, the error matrix is diagonal and the LS method reduces to minimising

$$X^2 = \sum_{i=1}^{N} (n_i - f_i)^2/\sigma_i^2 \approx \sum_{i=1}^{N} (n_i - f_i)^2/f_i$$

which can be done numerically.

Sometimes σ_i^2 is approximated by n_i, but the estimates $\hat{\theta}$ found this way are more sensitive to statistical fluctuations. (For large sample sizes the two choices give the same result.)

Since 1 degree of freedom has been lost due to the normalisation condition, $\Sigma n_i = n$, X^2_{min} would follow $f(\chi^2|N-1-L)$ if the model consisted of L independent parameters.

Choice of binning

Two common choices:

- equal width
- equal probability

Must not choose the binning in order to try to make X^2_{min} as small as possible! In this case X^2_{min} would no longer follow the χ^2 distribution.

It is necessary to have several entries in each bin, so that $(n_i\text{-}f_i)/f_i^{1/2}$ approximates a unit Gaussian. It is customary to require a minimum expectation of 5 entries per bin. The bins that contain less than this number can be ignored or combined to make larger bins in the less probable regions.

Using LS method with biased data samples

Some data samples may not reflect the true underlying distribution because of unequal detection efficiency for each event.

The best method to deal with this is to modify the theoretical model to account for the detection efficiency. Then no modification of the least squares minimisation is necessary. If this is not possible, then you can do either:

1) Modify n_i: If the detection efficiency for event j in bin i is ε_{ij}, then

$$n_i' = \sum_{j=1}^{N} 1/\varepsilon_{ij}$$

and minimise, $X^2 = \Sigma_{i=1}^{N} (n_i'-f_i)^2/f_i$

2) Modify f_i: $f_i' = f_i D_i$ where $D_i = n_i^{-1} \Sigma_{j=1}^{N} \varepsilon_{ij}$

and minimise $X^2 = \Sigma_{i=1}^{N} (n_i-f_i')^2/f_i'$

These alternatives work reasonably well when the variation of the weights is small. Otherwise the uncertainty of the estimates will not be well defined. (For example, by including large weight events, the estimated variances can actually increase.)

Linear LS estimation with linear constraints

Often it turns out that the true values, $\vec{\eta}$, are related through algebraic constraint equations. The observations, $\vec{y}$, do not strickly satisfy these contraints, but one wishes to form estimates, $\hat{\vec{\eta}}$, that do. The variance of these estimates should be smaller than if the constraints were not taken into account.

Two methods exist: elimination and Lagrange

multipliers.

Example: 3 angles of a triangle

Elimination:

model has 2 parameters, η_1, η_2 and minimise:

$$X^2(\eta_1,\eta_2) = (y_1-\eta_1)^2/\sigma_1^2 + (y_2-\eta_2)^2/\sigma_2^2 + (y_3-(\pi-\eta_1-\eta_2))^2/\sigma_3^2$$

Lagrange multipliers

To solve this problem with Lagrange multipliers, minimise the distribution:

$$X^2(\eta_1,\eta_2,\eta_3,\lambda) = \sum_{i=1}^{3} (y_i-\eta_i)^2/\sigma_i^2 + 2\lambda(\sum_{i=1}^{3}\eta_i-\pi)$$

In general, if $B\vec{\theta} - \vec{b} = 0$ represents K constraint equations (B is a K×L matrix) then minimise,

$$X^2(\vec{\theta},\vec{\lambda})=(\vec{y}-A\vec{\theta})^T V^{-1}(\vec{y}-A\vec{\theta})+2\vec{\lambda}^T(B\vec{\theta}-\vec{b})$$

The solution to this is given by,

$$\hat{\vec{\theta}} = C^{-1}\vec{c} - C^{-1} B^T V_B^{-1} (BC^{-1}\vec{c} - \vec{b})$$

and

$$V(\hat{\vec{\theta}})=C^{-1} - (BC^{-1})^T V_B^{-1} (BC^{-1})$$

where $C \equiv A^T V^{-1} A$

$\vec{c} \equiv A^T V^{-1} \vec{y}$

$V_B \equiv B C^{-1} B^T$

Confidence intervals from the LS method.

When the theoretical model is linear in the parameters, we are able to write down the solutions for $\vec{\hat{\theta}}$ and $V(\vec{\hat{\theta}})$. The expression for X^2 can be rewritten as:

$$X^2(\vec{\theta}) = X^2_{min} + (\vec{\theta} - \vec{\hat{\theta}})^T V^{-1}(\vec{\hat{\theta}})(\vec{\theta} - \vec{\hat{\theta}})$$

One can also write a Taylor expansion about the minimum, by comparison with the above, yeilds the estimate

$$V_{ij}(\vec{\hat{\theta}}) = 2\,(\,\partial^2 X^2/\partial\theta_i\partial\theta_j)^{-1}$$

The confidence intervals are then given by the region within the "ellipse"

$$X^2(\vec{\theta}) = X^2_{min} + a$$

1 and 2 parameter case:

a	γ (1 par)	γ (2 par)
1^2	0.683	0.393
2^2	0.954	0.865
3^2	0.997	0.989

Hypothesis Testing

Instead of estimating an unknown parameter, the results of an experiment may be used to decide whether the theoretical model (with no unknown parameters) is acceptable, given the observations.

Example: Suppose a model estimates the lifetime of the nucleus in Exercise 11 to be τ_0. Is the data compatable with the model?

Notation: $H_0 : \tau=\tau_0$ (*null hypothesis)*
$H_1 : \tau\neq\tau_0$

This is an example of a *parametric test* which follows the idea of confidence intervals. Examples of *non-parametric tests*: is the underlying distribution consistent with the model ? (this is answered by *goodness-of-fit tests*); are the two experimental distributions of the same form ? (can be studied with *distribution-free tests.*)

Typically, the hypothesis cannot be proven true or false, but one can determine the probability of obtaining the observed result, assuming the hypothesis was true.

Hypothesis testing may also be part of the data analysis, for example to decide if each event is due to signal or background process.

General concepts and terms

Suppose two hypotheses imply two different choices of the parameter θ:

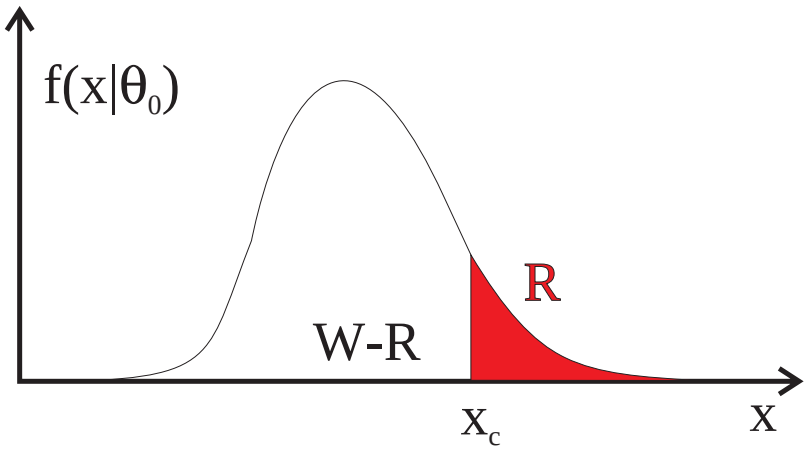
$$H_0 : \theta = \theta_0$$
$$H_1 : \theta = \theta_1 \quad (\textit{simple hypothesis})$$

or

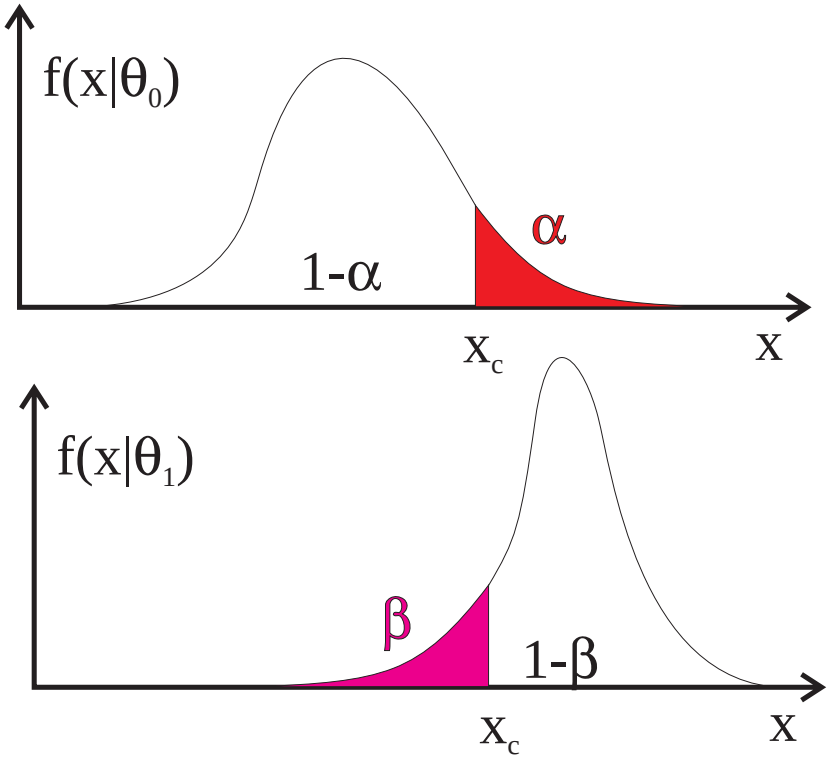
$$H_1 : \theta > \theta_1 \quad (\textit{composite hypothesis})$$

Assuming H_0 is true, we can define a region, R, from the complete sample space W, such that the probability that $x \in R$ is α, a preassigned number (usually $\alpha << 1$)

R is the *rejection* (or *critical*) region for H_0
W-R is the *acceptance* region for H_0
α is the *signifigance* or *size of test*
x_c is the *critial value* that separates R:



So, if $x_{obs} > x_c$, we reject hypothesis H_0, and otherwise accept it. It is clear that in 100α % of all decisions, H_0 will be rejected when in fact it should have been accepted. This mistake is called a *Type I error* (or error of the first kind). A *Type II error* occurs when H_0 is accepted, when in fact it was false.:



1-β is the *power of the test*, the probability of rejecting H_0 when it is false.
We wish to choose x_c so that the number of Type I and Type II errors are as small as possible.

Neyman-Pearson test

This is a method to choose x_c, when both H_0 and H_1 are simple hypotheses. (ie. θ can take only two possible values, θ_0 or θ_1).

$$\alpha = \int_R f(x|\theta_0)\, dx$$

$$1-\beta = \int_R f(x|\theta_1)\, dx$$

$$= \int_R \frac{f(x|\theta_1)}{f(x|\theta_0)} f(x|\theta_0)\, dx$$

Given α, we want to find the region R which maximises $1-\beta$. To do this take the region in which

$$\frac{f(x|\theta_1)}{f(x|\theta_0)}$$

is the largest. That is define R as the set of points satisfying

$$\frac{f(x|\theta_1)}{f(x|\theta_0)} > k$$

where k is determined from α.

If the experiment consists of a series of measurements $\vec{x}$, replace f by $L(\vec{x}\,|\theta) = \prod_i f(x_i|\theta)$

Likelihood ratio test for composite hypotheses

Denote the total parameter space: Ω
H_0 places some contraints on some of the parameters, ie. $\vec{\theta} \in \omega$ (ω is a subspace of Ω).
Given the observations $\vec{x}_n$, form the likelihood function, $L = \prod_{i=1}^{n} f(x_i|\vec{\theta})$

If the maximum of L in overall space is $L(\hat{\Omega})$ and in the subspace ω is $L(\hat{\omega})$ then the likelihood ratio is,

$$\lambda \equiv \frac{L(\hat{\omega})}{L(\hat{\Omega})} \qquad 0 \le \lambda \le 1$$

If $\lambda \approx 1$ then it is likely that H_0 is true and if $\lambda \approx 0$ then it is unlikely that H_0 is true. So define a critical region for λ: $0 < \lambda < \lambda_a$:

$$\text{where } \alpha = \int_0^{\lambda_a} g(\lambda|H_0)\, d\lambda$$

If g is not known but the distribution of some function of λ is known, then take

$$\alpha = \int_{y(0)}^{y(\lambda_a)} h(y|H_0)\, dy$$

If the sample is large, we can use the asymptotic behavoir for likelihood ratios: If H_0 imposes r constraints then $-2 \ln \lambda$ is distributed as a χ^2 distribution with r degrees of freedom.

Exercise 12

Apply the likelihood ratio test to the hypothetical experiment defined in exercise 11. Suppose σ_t is unknown and we want to test the hypothesis that $\tau=\tau_0=1$s.

$$H_0 : \tau = \tau_0 \qquad H_1 : \tau \neq \tau_0$$

Ω is given by $0 < \tau < \infty$, $0 < \sigma_t < \infty$
ω is given by $\tau = \tau_0$, $0 < \sigma_t < \infty$

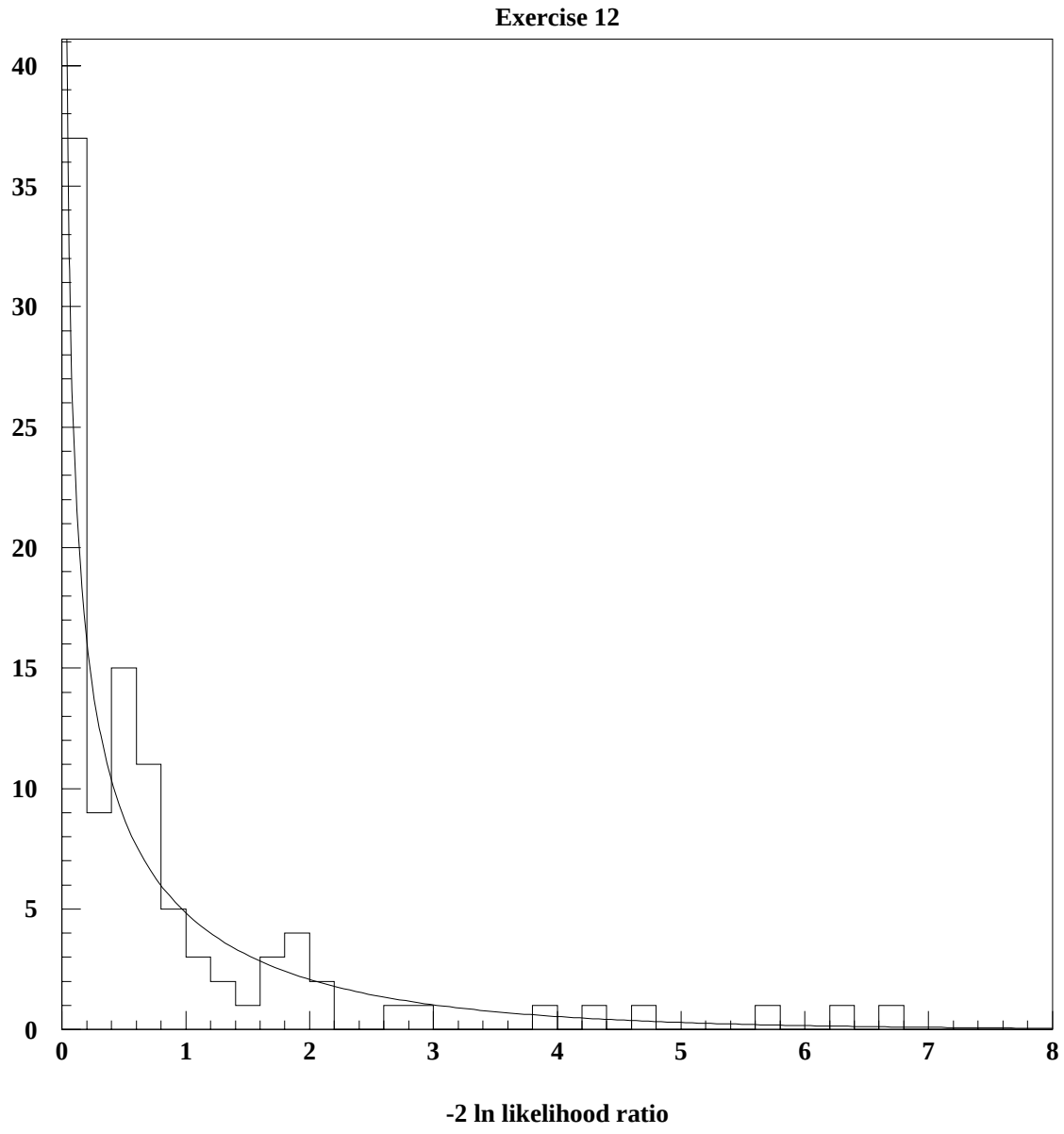
Define $\lambda = L(\hat{\omega})/L(\hat{\Omega})$

Show the distribution of $-2 \ln \lambda$ (for the 100 repititions of the experiment) and compare this to the χ^2 distribution with 1 degree of freedom.

Note: $\ln \lambda = \ln L(\hat{\omega}) - \ln L(\hat{\Omega})$ is easier to compute than λ.

What is the rejection region if the size of the test (α) is to be 10%? How many trials of your 100 experiments fail this test?

Solution to Exercise 12



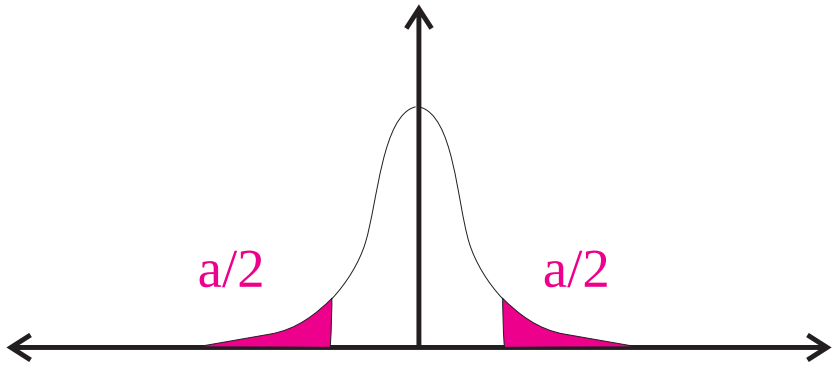
Parametric tests for Gaussian variables

If x_i are measurements from a Gaussian distribution, $G_{(\mu,\sigma^2)}$ we may want to test the hypothesis:

$$H_0 : \mu = \mu_0 \quad (\mu_0 \text{ is some number})$$

the alternative is, $H_1 : \mu \neq \mu_0$

If σ is known, then form the variable, $\langle x \rangle = \Sigma x_i/n$ and if H_0 is true, $d=(\langle x \rangle - \mu_0)/(\sigma/\sqrt{n})$ would follow the standard Gaussian $G_{(0,1)}$. Would likely reject H_0 if $d \gg 1$ or $d \ll -1$, so define the rejection region:



If σ is also unknown, then $(\langle x \rangle - \mu_0)/(s/\sqrt{n})$ follows the student-t distribution.

Comparison of means: 2 Gaussian distributions

Suppose $\vec{x}$ and $\vec{y}$ represent n and m measurements distributed according to $G_{(\mu_x,\sigma_x^2)}$ and $G_{(\mu_y,\sigma_y^2)}$ respectively.

If σ_x, σ_y are known

$\bar{x}$ and $\bar{y}$ have distributions: $G_{(\mu_x,\sigma_x^2/n)}$ $G_{(\mu_y,\sigma_y^2/m)}$, so the variable

$$\frac{(\bar{x}-\bar{y})-(\mu_x-\mu_y)}{\sqrt{\sigma_x^2/n+\sigma_y^2/m}}$$

will be distributed according to the standard Gaussian, $G_{(0,1)}$. To test if $\mu_x = \mu_y$ use

$$(\bar{x}-\bar{y})/\sqrt{\sigma_x^2/n+\sigma_y^2/m}$$

and proceed as before.

If σ_x, σ_y are unknown but equal

Use $d = (\bar{x}-\bar{y})/\sqrt{s^2/n+s^2/m}$ where

$$s^2 = \frac{1}{n+m-2}\left(\sum_i (x_i-\bar{x})^2 + \sum_i (y_i-\bar{y})^2\right)^2$$

and d follows the student t-distribution with n+m-2 degrees of freedom.

If σ_x, σ_y are unknown

If the sample size is large enough, the variable

$$d = (\bar{x} - \bar{y}) / \sqrt{s_x^2/n + s_y^2/m}$$

will follow the standard Gaussian, $G_{(0,1)}$. For example if two experiments quote the results $x \pm \Delta x$ and and $y \pm \Delta y$, then use

$$\frac{x - y}{\sqrt{(\Delta x)^2 + (\Delta y)^2}}$$

to test if they are compatable.

To compare several experimental results: $x_i,\ \Delta x_i$

If the hypothesis:

$$H_0 \quad : \quad \mu_1 = \mu_2 = \mu_3 ...$$

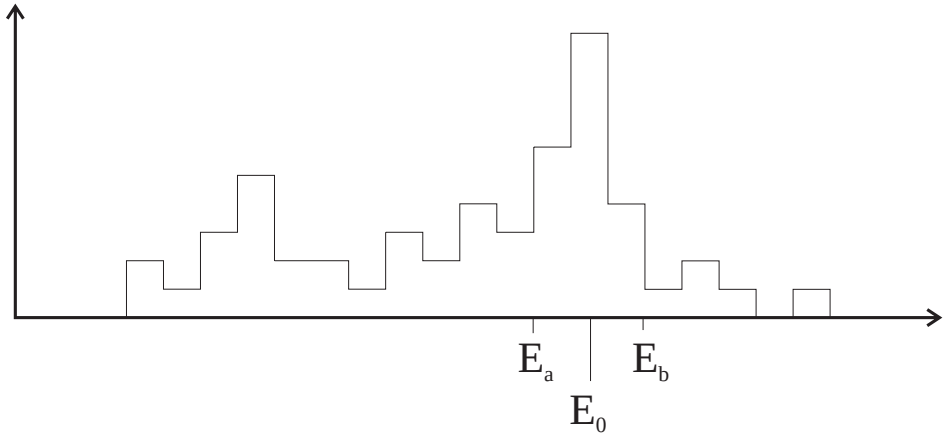
is true, then

$$X^2 = \sum_{i=1}^{N} \frac{(x_i - \bar{x})^2}{\Delta x_i^2}$$

(where $\bar{x}$ is the weighted average) should follow the χ^2 distribution for $N - 1$ degrees of freedom. The cumulative χ^2 distribution can be used to calculate the rejection region.

Significance of signal above background

For example, suppose a spectrum from an x-ray source was seen to be:



Is the effect at E_0 real or just a statistical fluctuation?

We can ask:

1. What is the probability that a statistical fluctuation of the background could produce an effect as large (or larger) than the one observed at the value E_0?

2. What is the probability to observe such a fluctuation at any position?

Answers:

1 N = total number of counts in $[E_a,E_b]$
B = total amount of background in $[E_a,E_b]$

Hypothesis: $H_0 : N = B$

Assume B and V(B) are known (theory or sidebands). N is distributed according to Poisson distribution, so under assumption at N=B,
$V(N) = N = B$
then, $V(N-B) = V(N) + V(B) = B + V(B)$
If N is large, approximate Poisson by Gaussian, then use

$$d = (N-B)/(V(N-B))^{1/2} \approx (N-\hat{B})/(\hat{B}+V(\hat{B}))^{1/2}$$

which follows $G_{(0,1)}$

So, $$P(d;E=E_0) = \int_d^\infty G_{(0,1)}(x)\,dx$$

is the probability that an statistical fluctuation is produced at least as large as the one observed. It is common to quote d as the number of standard deviations of the effect.

② If we consider bumps extending over k bins, and the total number of bins is n, then the central value of a bump could be located in (n-k+1) different bins in the plot.

The probability to observe a fluctuation of at least d standard deviations anywhere in the histogram is:

$$P(d) = 1 - (1 - P(d; E=E_0))^{n-k+1}$$

For large d,

$$P(d) \approx (n-k+1)\, P(d; E=E_0)$$

(In HEP, typically 5σ signifigance is necessary to claim the observation of a new resonance.)

Goodness of Fit Tests

Given measurements, $x_1,...,x_n$, following an unknown distribution f(x) and if $f_0(x)$ is a specified distribution, we may want to test the hypothesis,

$$H_0 : \quad f(x) = f_0(x)$$

As usual we form a test statistic of known distribution and define rejection and acceptance regions with probabilities α and $1-\alpha$, assuming H_0 is true.

Pearson's χ^2 test

- exact for large samples only
- data are binned into N exclusive bins
- the hypothesis under test:

$$H_0 : \quad p_1=p^0_1, \quad p_2=p^0_2, \quad .... \quad p_N=p^0_N$$

where $\sum_{i=1}^{N} p^0_i = 1$

To test whether the observed number of entries in each bin is compatible with the predicted number, form the variable,

$$X^2 = \sum_{i=1}^{N} (n_i - np^0_i)^2/np^0_i = (\sum_{i=1}^{N} n_i^2/np^0_i) - n$$

when H_0 is true, X^2 approximately follows the χ^2 distribution with N-1 degrees of freedom. (As long as np^0_i is large enough so Poisson is approximately Gaussian.)

- If H_0 is false, then X^2_{obs} will take on larger values, so define the rejection region to be at the largest values of X^2.

- Remarks about the choice of binning in section on Least squares fitting apply here.

- If the data were used to determine L linear parameters of the model, the X^2 would follow χ^2 distribution with N-1-L degrees of freedom (if the determination was done with the same binning and found using LS or ML).

- If unbinned ML used to determine parameters, X^2 no longer strictly χ^2 (N-1-L), but it is bounded by χ^2(N-1) and χ^2(N-1-L). If N>>L, there is little difference.

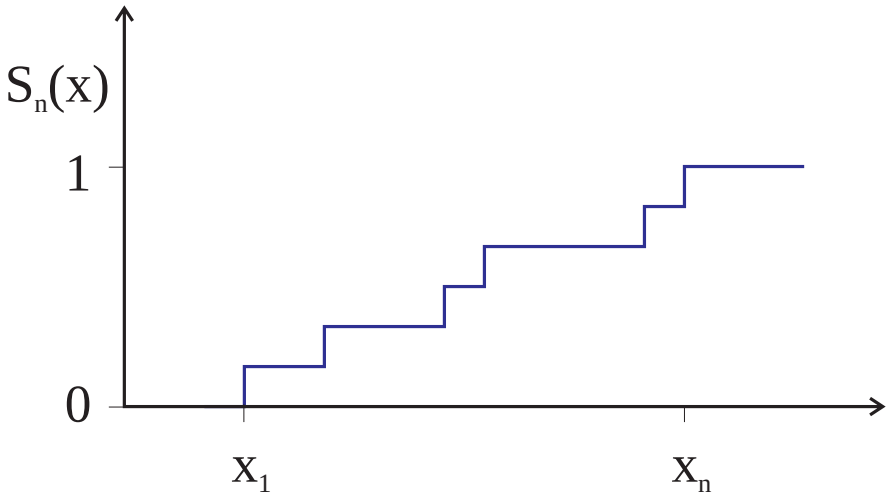
Kolmogorov - Smirnov test

- avoids binning of data
- superior to Pearson's χ^2 for small samples

Given n observations of x, form an ordered sample, ascending in magnitude: $x_1, x_2, \ldots, x_n$

The cumulative distribution is defined by

$$S_n(x) = \begin{cases} 0 & x < x_1 \\ i/n & x_i \le x < x_i+1 \\ 1 & x \ge x_n \end{cases}$$



Compare this to the expected cumulative distribution, $F_0(x)$:

Form the quantity, $D_n = \max |S_n(x) - F_0(x)|$

If $F_0(x)$ is completely specified (ie. no parameters deduced from the data), then D_n is independent of $F_0(x) \Rightarrow D_n$ is distribution free.

Large n limit:

$$P(D_n \leq z/\sqrt{n}) = 1 - 2 \sum_{r=1}^{\infty} (-1)^{r-1} e^{-2r^2z^2}$$

which is valid for $n \geq 80$.

For $n \geq 100$, the following table can be used to define rejection region:

$P(D_n \leq d_\alpha) = 1-\alpha$:

α	0.20	0.10	0.05	0.01
d_α	$1.07/\sqrt{n}$	$1.22/\sqrt{n}$	$1.36/\sqrt{n}$	$1.63/\sqrt{n}$

Exercise 13

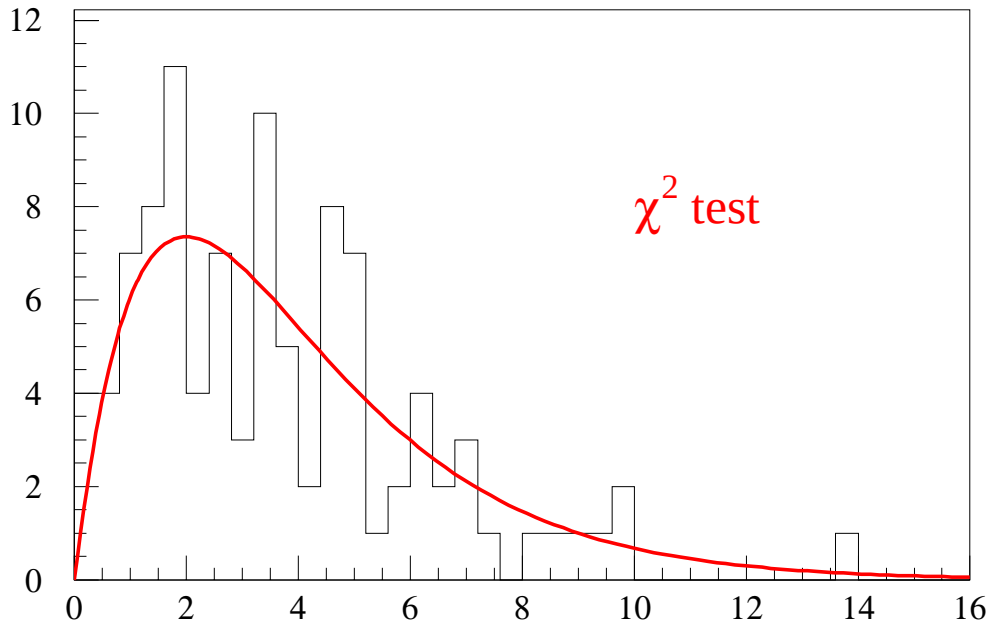
Apply (a) Pearson's χ^2 and (b) Kolmogorov - Smirnov tests to the experimental samples produced in exercise 11.

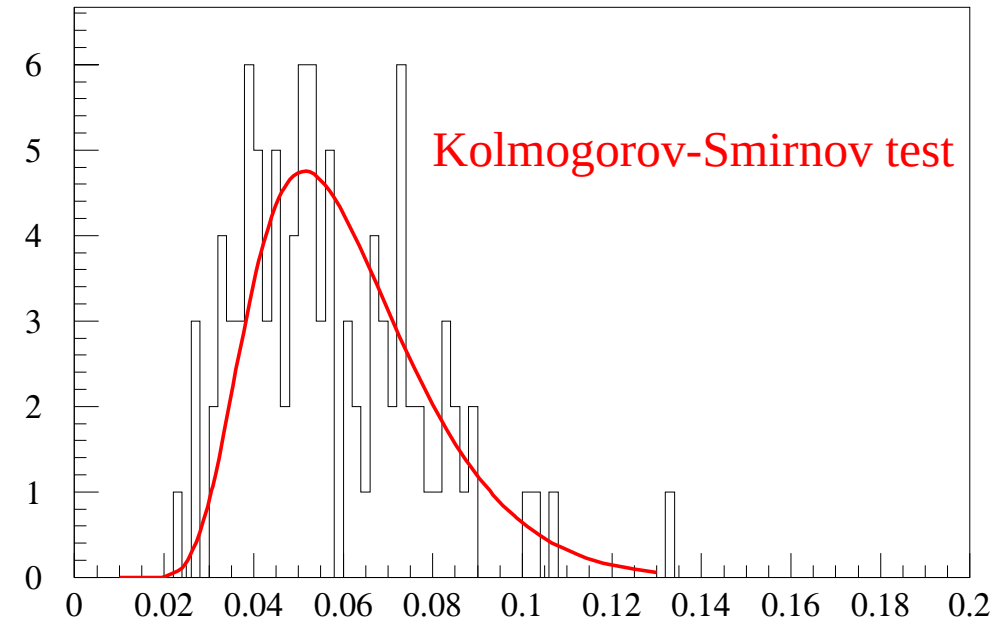
(a) Compare the observed time distribution to the model with τ = 1 s, σ_t = 0.5 s. To do this, make 5 one-second bins starting at t=-1 s. Show the X^2 distribution for the 100 repititions and compare to the appropriate χ^2 distribution. What is the rejection region for a test of size (α) of 10% ? How many of your 100 experiments fail this?

(b) Compare the cumulative distribution with the model for τ = 1 s, σ_t = 0.5 s. Show the D_n distribution for your data. How many experiments fail a test of size α=10%?
Hint: The cumulative distribution is given by:

$$F_0(t) = \int_{-\infty}^{t} f(t|\tau,\sigma_t)$$
$$= \frac{1}{2}\left(\mathrm{erfc}\left(\frac{-t}{\sqrt{2}\,\sigma_t}\right) - \exp\left(\frac{\sigma_t^2}{2\,\tau^2} - \frac{t}{\tau}\right)\mathrm{erfc}\left(\frac{\sigma_t}{\sqrt{2}\,\tau} - \frac{t}{\sqrt{2}\,\sigma_t}\right)\right)$$

Solution to Exercise 13





Test of Independence

Are two different properties measured in an experiment correlated? The hypothesis under test is then:

$$H_0 : \quad f(x,y) = f_1(x)\, f_2(y)$$

χ^2 can again be used. This time bin in 2 D:
$n_{ij} \equiv$ the # in bin i of x, bin j of y , $n_{i.} \equiv \sum_j n_{ij}$,
$n_{.j} \equiv \sum_i n_{ij}$

If true probability is given by p_{ij}, then

$$H_0 : \quad p_{ij} = p_{i.}\, p_{.j} \quad \text{for all i,j}$$

So form the variable,

$$X^2 = \sum_i \sum_j (n_{ij} - n_{i.} n_{.j}/n)^2/(n_{i.} n_{.j}/n)$$

$$= n\{\sum_i \sum_j n_{ij}^2/(n_{i.} n_{.j}) - 1\}$$

Then X^2 follows χ^2 distribution with # of d.o.f: (I-1)(J-1)

Run test for comparing 2 samples

Given two sets of measurements, $(x_1, ..., x_n)$ and $(y_1, ..., y_m)$, where $n \leq m$, we wish to test the hypothesis,

$$H_0 \quad : \quad f_x(x) = f_y(y)$$

To do this, make an ordered list of the combined sample, for example

$$x_1 \; x_2 \; y_1 \; x_3 \; y_2 \; y_3 \; x_4 \; x_5$$

and count the number of *runs* (groups of elements from the same set of measurements). If H_0 is true, there should be a large number of runs. To find the probability to find r runs for two random samples from the same distribution is a problem in combinatorics,

$$p(r = 2k) = 2\frac{\binom{n-1}{k-1}\binom{m-1}{k-1}}{\binom{n+m}{n}}, \quad r \text{ even}$$

$$p(r = 2k-1) = \frac{\binom{n-1}{k-2}\binom{m-1}{k-1} + \binom{n-1}{k-1}\binom{m-1}{k-2}}{\binom{n+m}{n}}, \quad r \text{ odd}$$

The distribution has the mean and variance,

$$\mu_r = \frac{2nm}{n+m} + 1 \qquad V(r) = \frac{2nm(2nm-n-m)}{(n+m)^2(n+m-1)}$$

and for large n, m $(n, m > 10)$, $d = (r - \mu_r)/\sqrt{V(r)}$ approximately follows the standard Gaussian, $G_{(0,1)}$.

Run test to supplement Pearson's χ^2 test

Recall that X^2 is insensitive to the sign of $(n_i - np_i^0)$. An additional test comes from considering the sign of this quantity in subsequent bins and counting the number of runs. In the case where no parameters of the model have been determined from the data, the run test and the χ^2 tests are independent, and so they can be combined into a simple test, and the quantity $u = -2(\ln P_{\chi^2} + \ln p(r))$ will follow χ^2 with 4 d.o.f.

Proof: Suppose x is uniformly distributed in $[0, 1]$, consider $u = -2 \ln x$. To work out its distribution function:

$$\begin{aligned} g(u)\,du &= f(x)\,dx \qquad (f(x) = 1) \\ g(u) &= \left|\frac{dx}{du}\right| = \frac{1}{2} e^{-\frac{1}{2}u} \end{aligned}$$

This is the χ^2 distribution function for 2 degrees of freedom.

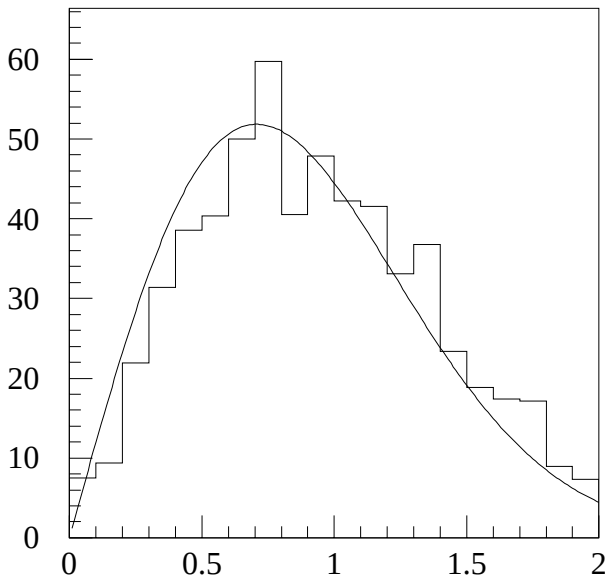
Given x_1 and x_2 which are both uniformly distributed in $[0, 1]$, the variable u,

$$u = -2\left(\ln x_1 + \ln x_2\right)$$

will follow a χ^2 distribution with 4 degrees of freedom.

Example of Run test with χ^2 test

A simulated data sample is shown below along with a distribution function that was not used to generate the data sample. There are 20 bins shown, and the distribution function was normalized to match the number of events in the data sample.



The value of X^2 for this distribution is 25.2, for 19 d.o.f., resulting in $P_{\chi^2} = 0.16$, which alone gives little reason to suspect the model. There are 7 bins with negative $(n_i - np_i^0)$ and 13 positive bins, with only 5 runs in the signs, so that $p(r = 5) = 0.0074$ is reason to reject the hypothesis.
The combination $u = -2(\ln P_{\chi^2} + \ln p(r)) = 13.5$ corresponds to probability ≈ 0.009.

References for Statistics

1. *Probability and Statistics in Particle Physics*, A. G. Frodesen, O. Skjeggestad, H. Tøfte, Columbia University Press, 1978.

2. *Statistical Methods in Experimental Physics*, W. T. Eadie, D. Drijard, F. E. James, M. Roos, B. Sadoulet, North Holland, 1971.

3. *Statistics for Nuclear and Particle Physicists*, L. Lyons, Cambridge University Press, 1986. (Very elementary.)

4. *Probability, Statistics, and Monte Carlo*, in Review of Particle Properties, *Phys. Rev.* D50 Part I (1994) 1271–1284.

Part V: Chaotic Dynamics

Do not confuse chaotic with random :

Random: unreproducible, unpredictable

Chaotic: deterministic - same initial conditions
lead to same final state BUT
the final state is very different for
small changes to initial conditions
➠ difficult or impossible to make predictions.

The study of chaotic systems is now a popular branch of physics. Little was known in this field before computers were applied to the problem.

Chaos is seen in many physical systems, for example:

- fluid dynamics (weather forecasting)
- some chemical reactions
- lasers
- particle accelerators

Conditions necessary for chaos:

1. system has 3 independant dynamical variables
2. the equations of motion are non-linear

A damped driven pendulum will be used to demonstrate features of chaotic motion:

$$ml\, d^2\theta/dt^2 + c\, d\theta/dt + mg \sin\theta = A \cos(\omega_D t + \phi)$$

Instead we will use a dimensionless form:

$$d\omega/dt + q\, d\theta/dt + \sin\theta = f_0 \cos(\omega_D t)$$

The three dynamic variables are: ω, θ, t

The non-linear term is $\sin\theta$

We have already seen that the pendulum is chaotic only for centain values of q, f_0, and ω_D.

In the examples below, we use:
ω_D=2/3, q=1/2, and f_0 near 1.

To watch the onset of chaos (as f_0 is increased) we will look at the motion of the system in phase space, once transients die away.

Pay close attention to the period doubling that preceeds the onset of chaos.

Poincaré Section

The phase space curves we have seen are 2D projections of the full 3D phase space that completely describes the pendulum. These projections hide the detail of the intricate surface that the chaotic pendulum

The Poincaré section is a slice of the 3D phase space at a fixed value of: $\omega_D t \bmod 2\pi$

This is analogus to viewing the phase space deveopment with a strobe light in phase with the driving force. Periodic motion results in a single point, period doubling results in two points.

Attractors

•The surfaces in phase space along which the pendulum follows (after transient motion decays) are called attractors.

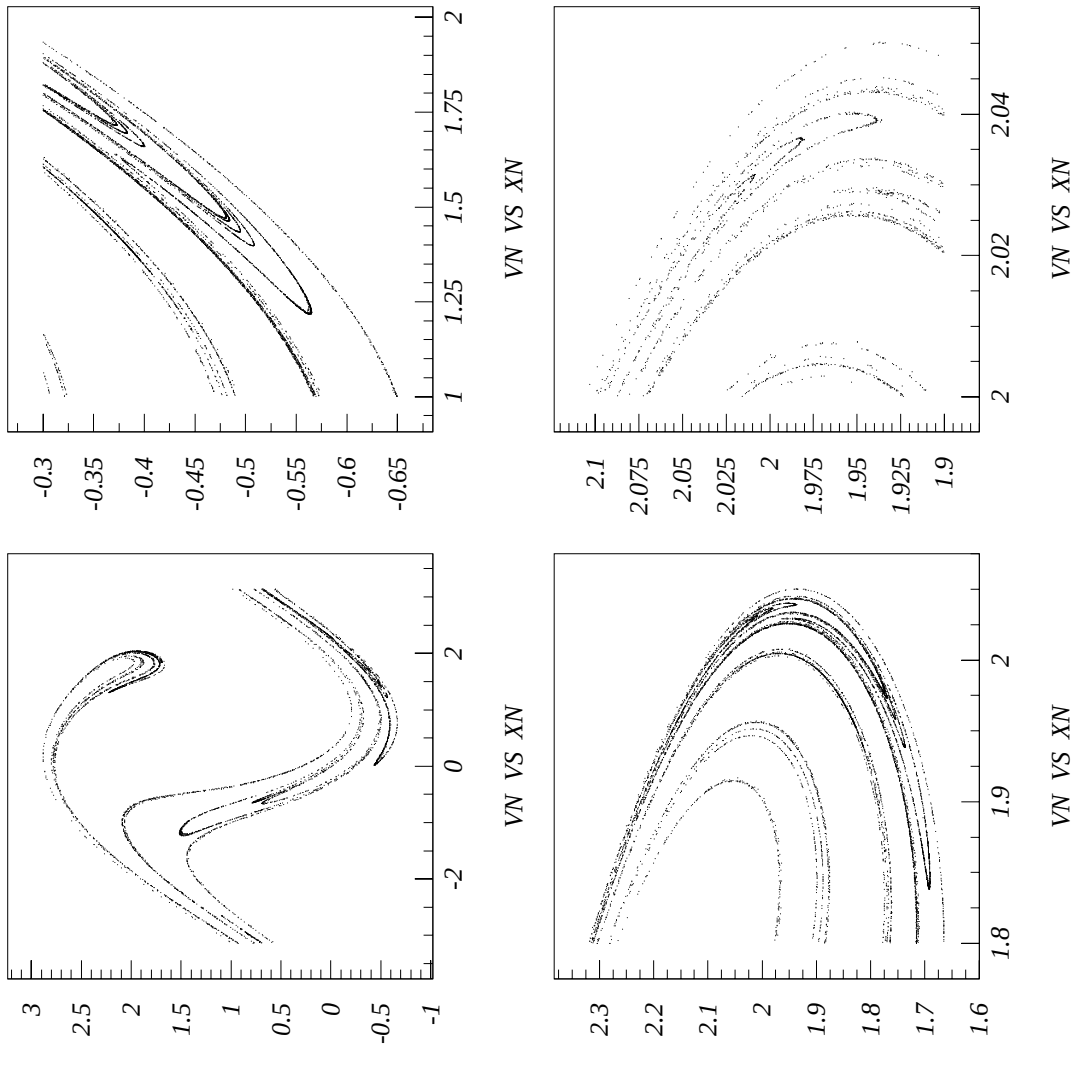
•An attractor in a damped undriven pendulum is just a point at $\theta=\omega=0$. (0D in 2D phase space).

•An attractor of a periodic pendulum is a curve (1D in 3D phase space).

•Chaotic attractors (sometimes called strange attractors) are fractals. They have a non-integer dimension. (In this case 2<dim<3.)

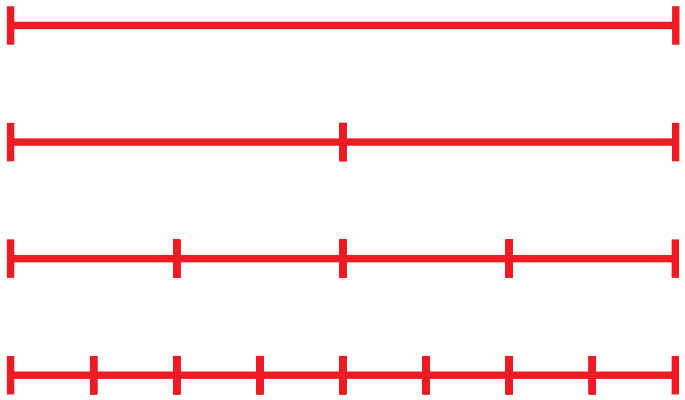
•The fine structure is seen to be quite complex and similar to the gross structure: self-similarity.

Exploring an attractor:



Fractional dimension

Capacity dimension of a line and square:



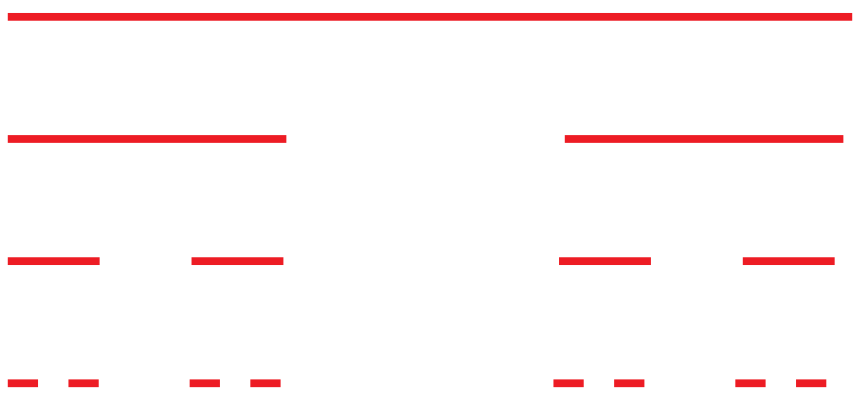
N	ε	N	ε
1	L	1	L
2	L/2	4	L/2
4	L/4	16	L/4
8	L/8		
2^n	$L/2^n$	2^{2n}	$L/2^n$

$$N(\varepsilon) = L^d (1/\varepsilon)^d$$

$$d_c = \lim_{\varepsilon \to 0} \log N(\varepsilon) / \log (1/\varepsilon)$$

Example: dimension of the Cantor set

The Cantor set is produced by the following iterative process:



N and ε to cover the Cantor set:

N	ε
1	1
2	1/3
4	1/9
8	1/27
2^n	$1/3^n$

$$d_c = \lim_{n\to\infty} \log 2^n / \log 3^n$$

$$d_c = \log 2 / \log 3 < 1$$

The fractional dimension of a chaotic attractor is a result of the extreme sensitivity to initial conditions:

Lyapunov exponents are a measure of the average rate of divergence of neighbouring trajectories on an attractor.

Consider a small sphere in phase space. After a short time the sphere will evolve into an ellipsoid:

The average rate of expansion along the principle axes are the Lyapunov exponents.

Chaos implies at least one exponent is > 0.

For the pendulum, it can be shown that:

$$\Sigma \lambda_i = -q$$ (the damping coefficient)

There is no contraction or expansion along the $\omega_D t$ direction so one of the exponents is zero.

Furthermore it can be shown that the dimension of the attractor is: $d = 2 + \lambda_1 / (-\lambda_2)$

Bifurcation Diagrams

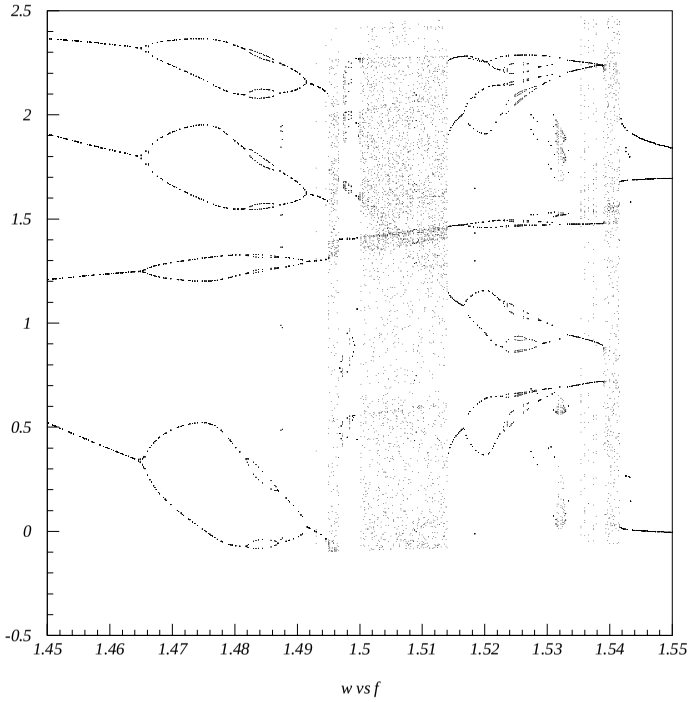
Bifurcation: a change in the number of solutions to a differential equation when a parameter is varied.

To observe bifurcations, plot long term values of ω (at a fixed value of $\omega_D t \bmod 2\pi$) as a function of the force term f_0.

Periodic $\rightarrow$ single value
Two solutions (left and right moving) $\rightarrow$ 2 solutions
Period doubling $\rightarrow$ double the number of solutions

The onset of chaos is often seen as a result of successive period doublings.

Bifurcation diagram for the damped driven pendulum. The horizontal axis is the driving force coefficient, f_0, and the vertical axis shows the possible long term velocities for a fixed phase of the driven force. The initial conditions are chosen at random for each point, and the first 100 cycles are not shown, so that transients will have decayed.



Comparison of the pendulum to simpler chaotic systems

Difference equations also exhibit chaotic behavior, for example the logistic map:

$$x_n = \mu\, x_{n-1}\,(1 - x_{n-1})$$

For some values of μ, x tends to a fixed point, for other values, x oscillates between two points (period doubling), and for other values x becomes chaotic.

Feigenbaum number

The ratio of spacings between consecutive values of μ at the bifurcations approaches a universal constant (the Feigenbaum number)

$$\lim_{k\to\infty} \frac{\mu_k - \mu_{k-1}}{\mu_{k+1} - \mu_k} = \delta = 4.669201...$$

This is universal to all differential equations (within certain limits) and applies to the pendulum. By using the first few bifurcation points, one can predict the onset of chaos.

References for Chaotic dynamics

- *Chaotic Dynamics, an Introduction*, G. L. Baker and J. P. Gollub, Cambridge University Press, 1990.
- *Introduction to Computational Physics*, M. L. De Jong, Addison-Wesley Publishing Company, 1991.
- *Nonlinear Dynamics and Chaos*, J. M. T. Thompson and H. B. Stewart, John Wiley and Sons, 1986.
- *Dynamical Systems and Fractals*, K.-H. Becker and M. Dörfler, Cambridge University Press, 1989.
- *Chaos*, J. P. Crutchfield, J. D. Farmer, N. H. Packard, and R. S. Shaw, *Scientific American*, December 1986.
- *Chaos in Dynamical Systems*, E. Ott, Cambridge University Press, 1993.

Index