Scientific Computing with the NAG Toolbox for *MATLAB*

John Muddle
UK Academic Account Manager
NAG Ltd, Oxford

December 1st 2015

nag®

Experts in numerical algorithms and HPC services

The University Of Sheffield.

NAG at Sheffield University

- Unlimited use under Linux, Windows and Mac (32-bit and 64-bit)
  - As long as for academic or research purposes
  - Installation may be on any university, staff or student machine
- Products
  - All NAG Libraries: Fortran, C, SMP, Python, Java, .NET, Toolbox for MATLAB
- How do you get the software?
  - Help yourself from: http://www.nag.co.uk/downloads/index.asp
  - Request licence keys via support@nag.co.uk using your university e-mail address please e.g. xxxx@sheffield.ac.uk
- Full access to NAG Support
  - Send support requests to support@nag.co.uk

Objectives of today

Aim to:

- Extend your practical MATLAB programming abilities
- Gain an overview of the contents of the NAG mathematical library
- Understand how to find information about relevant NAG routines
- Use NAG routines in helping to solve simple but realistic problems

Slides

These slides are available as PowerPoint or PDF:

http://monet.nag.co.uk/nag_toolbox_training/sheffield/MATLAB

Introduction to NAG

- Numerical Algorithms Group - Founded 1970
  - Co-operative software project: Birmingham, Leeds, Manchester, Nottingham, Oxford, and Atlas Laboratory
- Incorporated as NAG Ltd. in 1976
  - Not-for-profit
  - Based in Oxford, with offices in Manchester, Chicago, Tokyo

What do NAG do?

- Mathematical algorithm development
  - Collaboration
- Software engineering – production of software libraries
- HPC services
- Consultancy
- Implementation / porting

A definition of Numerical Analysis

- "The study of methods for obtaining approximate solutions to mathematical problems" – *T. Hopkins and C. Phillips, 1988*
  - But beware of assuming that "approximation" is a pejorative term

What are numerical analysts concerned with?

- Accuracy of algorithms
  - Problems due to floating-point arithmetic
  - Error analysis - how are errors propagated?
- Stability
  - Sensitivity of a computed solution to changes in input data
- Efficiency
  - Which methods avoid unnecessary computation?
- Speed
  - Desirable – but beware – “how fast do you want the wrong answer?”

Why use numerical methods?
A simple example: roots of polynomials

- Famous formula for roots of quadratic equation $ax^2 + bx + c = 0$:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

- Formulae for roots of cubic $ax^3 + \cdots$ a bit more complicated
- Formulae for roots of quartic much too big to put on one slide
- No formulae for roots of higher order polynomials – numerical methods must be used

How do we decide what goes into libraries?

- Typically we get functionality requests from customers
  - via technical support calls
  - via salespeople
- We maintain a database of requests
- Probability of responding is weighted by number of requests, importance of customers, and difficulty of job

What's so hard about *that* then?

- Computer arithmetic makes things tricky.
- Many people assume that computers are completely accurate when it comes to operations on real numbers.
  - Usually they are not!
  - Numbers have finite precision
  - Also a finite size
  - They are subject to rounding error.

e.g. on a (hypothetical) decimal machine with 3 digits:

a = 5.55 b = 6.56 c = 12.1

a+b-c = 0.00 (due to rounding) instead of 0.01 in exact arithmetic

- We must also worry about overflow and underflow

Example: PDF of Gamma Distribution

Probability density function (PDF)

$$f(x;a,b) = \frac{1}{b^a \Gamma(a)} x^{a-1} e^{-x/b} \qquad x \geq 0, \quad a,b > 0$$

looks easy to evaluate. But, depending on a, b and x, any of the quantities

$$b^a \qquad \Gamma(a) \qquad x^{a-1} \qquad e^{-x/b}$$

might overflow or underflow, even if the end result is in a reasonable range.

Program code to check all contingencies can become monstrous!

We also have to worry about performance

- Tune up code
  - Use benchmarking programs
  - Use high-performance math libraries like ACML or MKL where possible
- Investigate possibilities for parallelism
  - OpenMP (for NAG SMP Library)
  - MPI (for NAG Cluster Parallel Library)
- But correctness is most important
  - How fast do you want the wrong answer?

Why is software testing important?

Six reasons not to test:

- "It takes too long – I've no time to do it"
  - Fine – but be aware it may take longer later
- "It costs too much"
  - It generally costs more to fix things later
- "This code will only be used for a short time"
  - That's what they said about Windows notepad
- "My code always works first time"
  - You've got to be kidding
- "It's someone else's responsibility"
  - Bad attitude!
- "I'm starting a new job next month"
  - Let's hope you don't last long in that one either

A Famous Software Bug

Ariane 5 rocket explodes (June 1996)

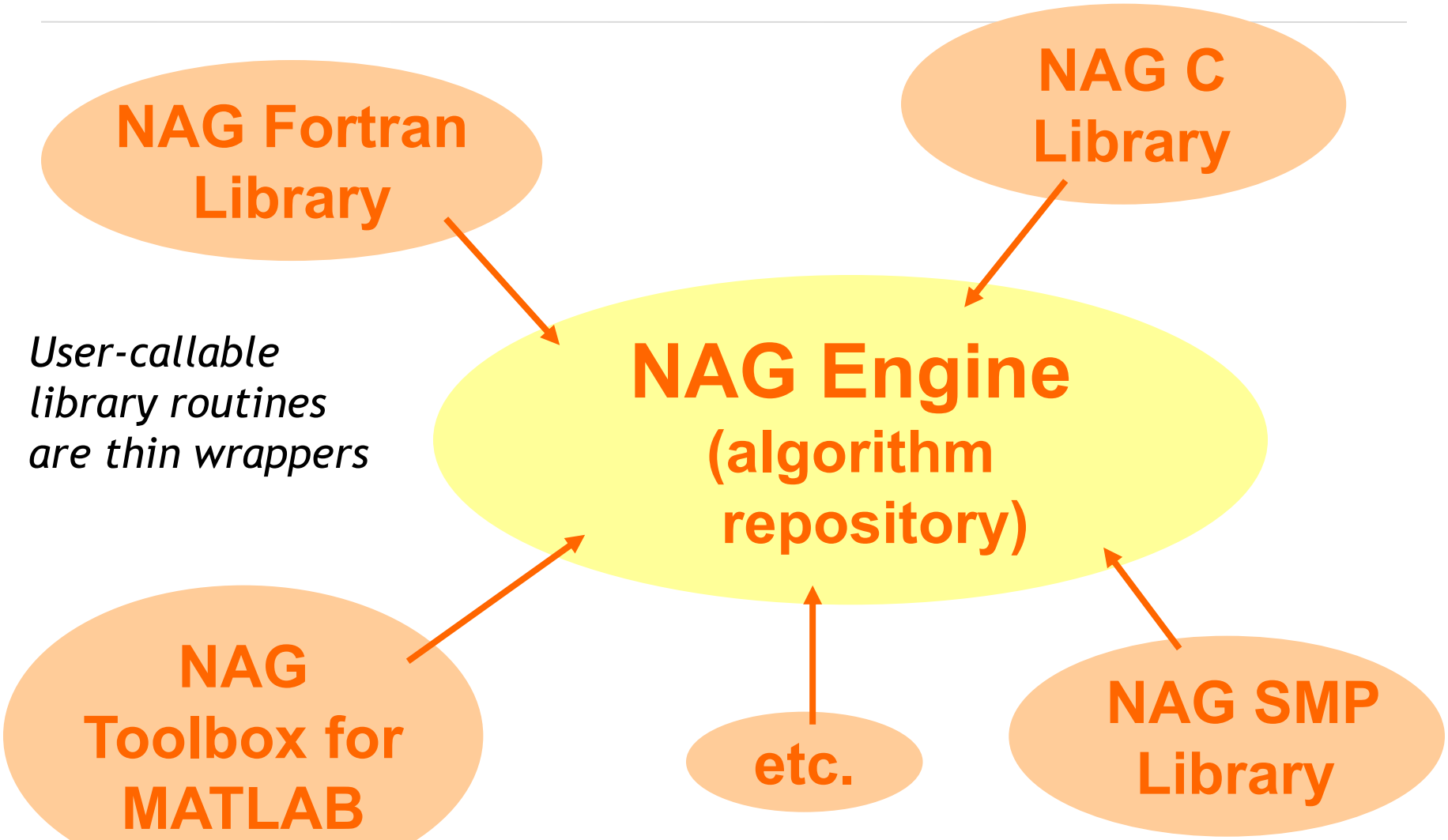
The Ariane 5 rocket exploded on its maiden flight because the navigation package was inherited from the Ariane 4 without proper testing. The new rocket flew faster, resulting in larger values of some variables in the navigation software. Shortly after launch, an attempt to convert a 64-bit floating-point number into a 16-bit integer generated an overflow. The error was caught, but the code that caught it elected to shut down the subsystem. The rocket veered off course and exploded. (Kernighan, 1999)

A successful launch in 2007

www.space.com/imageoftheday/image_of_day_070316.html

Credit: Arianespace. www.arianespace.com

The NAG Engine



NAG Toolbox *for MATLAB* – Mark 24

- Root Finding
- Summation of Series
- Quadrature
- Ordinary Differential Equations
- Partial Differential Equations
- Numerical Differentiation
- Wavelets
- Integral Equations
- Mesh Generation
- Interpolation
- Curve and Surface Fitting
- Optimization (local and global)
- Approximations of Special Functions
- Dense and sparse Linear Algebra
- Option pricing
- Correlation and Regression Analysis
- Nearest correlation matrix
- Multivariate Analysis of Variance
- Random Number Generators
- Univariate Estimation
- Nonparametric Statistics
- Smoothing in Statistics
- Contingency Table Analysis
- Survival Analysis
- Time Series Analysis
- Operations Research

Installation of the NAG Toolbox

Get a version of the NAG Toolbox appropriate to your copy of MATLAB from:

http://www.nag.co.uk/downloads/mbdownloads.asp

You will need a licence key – try the first one below:

MBW6I24DD TRIAL 2016/01/04 "R6kunal0pgtbWdaflgu1iwmkIR"
MBW3223DC TRIAL 2016/01/04 "60gn+ddlimgk40lKGnZSg2+kEz"
MBL6A24DN TRIAL 2016/01/04 "ChghxpANiuHo3xiJ1dRDoEhluk"
MBMI624DD TRIAL 2016/01/04 "9XhGMe8Ba=+ptxgYnlt4eDYeON"

Check correct installation

To check that you have the NAG Toolbox installed, in MATLAB command window type a00aa

To check you have the licence key installed, type a00ac
(return value 1 indicates key found).

NAG Toolbox *for MATLAB*

- The NAG Toolbox (like all NAG libraries) is divided into *chapters*, each devoted to a branch of mathematics or statistics. Each has a 1 or 3-character name and a title
  - e.g. *S – Special Functions* or *F03 – Determinants*
- All routines in the Toolbox have five-character names, beginning with the characters of the chapter name, e.g. d01aj.
- There are also *long names* that are more descriptive – e.g. c02ag() = nag_zeros_poly_real()

NAG Toolbox *for MATLAB*

- Documentation has an informative introduction to each chapter:
  - Technical background to the area.
  - Assistance in choosing the appropriate routine
- And a document for each routine with:
  - Description of method and references
  - Specification of arguments
  - Explanation of error exit
  - Remarks on accuracy
  - An example to illustrate use of routine, some enhanced with graphics

NAG Documentation

- Documentation is available through the MATLAB help system.
- In newer versions of MATLAB the NAG documentation may be a little harder to find. Click the F1 key, then choose “Supplemental Software”, then open the “NAG Toolbox” folder
- You can also find NAG Toolbox documentation online:
  - www.nag.co.uk/numeric/MB/manual64_24_1/html

NAG Toolbox *for MATLAB*

- Let's take a look …

A Simple Example

- Here is an example of how to use the NAG Library to compute the solution of a real system of linear equations, Ax = b, where A is an n by n matrix and x and b are n vectors.

```
a = [ 1.80,  2.88,  2.05, -0.89;
      5.25, -2.95, -0.95, -3.80;
      1.58, -2.69, -2.90, -1.04;
     -1.11, -0.66, -0.59,  0.80];

b = [ 9.52; 24.35; 0.77; -6.22];
```

A Simple Example

- And we call it like this:

```
[lu, ipiv, x, info] = f07aa(a, b);
x
x =
    1.0000
   -1.0000
    3.0000
   -5.0000
```

- Here the NAG routine f07aa takes two arguments, the matrix of coefficients, A, and the vector representing the right-hand side, b.

Try the c02ag example program

- Find the c02ag document
- Go to the example program section
- Click link to open the program in the MATLAB editor
- Run the program as it stands
- Save the program to another location
  - e.g. somewhere in your local file store
- Then modify the program to find all the roots of the equation $x^8 = 1$

What do we expect

- $x^8 = 1$ can be re-written as:
- $(x^4 + 1)(x^2 + 1)(x + 1)(x - 1) = 0$
- Obviously $x = 1, -1$ are roots
- $x^2 = -1 \Rightarrow x = i, -i$
- For $x^4 = -1$ we can use Euler's equation and de Moivre's equation to get:
- $x = \frac{1+i}{\sqrt{2}}, \frac{1-i}{\sqrt{2}}, \frac{-1+i}{\sqrt{2}}, \frac{-1-i}{\sqrt{2}}$

Optional Arguments

- Optional arguments are provided after all compulsory arguments.
- Optional arguments appear in pairs: a string representing the name followed by the value.
- The pairs can be provided in any order.
- There are optional arguments where:
  - A sensible default value exists which applies to many problems.
  - The argument only applies to some cases.
  - The value of the argument can normally be determined from that of other arguments at runtime.

Optional arguments

- For example, in the system of equations given in the previous section, it is obvious that n, the size of the matrix A, is 4.
- However we can tell MATLAB that n is 3, in which case it will solve the system represented by the top-left 3x3 section of A, and the first three elements of b.
- And so we would call like this …

Optional arguments

```
[lu, ipiv, x, info] = f07aa(a, b, 'n', nag_int(3));
x

x =
    4.1631
   -2.1249
    3.9737
   -6.2200
```

- The last element of x can (should) be ignored. Since b was a 4x1 matrix on input, it will be a 4x1 matrix on output, even though the last element is not being used.

Arguments

- A similar outcome can be achieved by:

```
[lu, ipiv, x, info] = f07aa(a(1:3,1:3), b(1:3));
x

x =
    4.1631
   -2.1249
    3.9737
```

- Here x is of appropriate size.

Another Example – Overriding Defaults

- g01hb (nag_stat_prob_multi_normal) computes probabilities associated with a multivariate Normal distribution, to a relative accuracy which defaults to 0.0001:

```
xmu = [0;0;0;0]; a = [-2;-2;-2;-2]; b = [2;2;2;2];
sig = [1,0.9,0.9,0.9;     % Variance-covariance matrix
       0.9,1,0.9,0.9;
       0.9,0.9,1,0.9;
       0.9,0.9,0.9,1];
nag_stat_prob_multi_normal(xmu,sig,'a',a,'b',b)
 ans = 0.9142
```

- We can vary *tol*:

```
nag_stat_prob_multi_normal(xmu,sig,'a',a,'b',b,'tol',0.1)
 ans = 0.9182
```

Errors and Warnings

- NAG toolbox routines can produce a number of errors (names are on the next slide)
- In most cases the error message will give more precise details of how the error was triggered. For example a NAG:arrayBoundError might display the message:

```
??? The dimension of argument 2 (A)
should be at least 4
```

Errors and Warnings

- NAG:arrayBoundError - Array provided is too small.
- NAG:callBackError - An error occurred when executing an M-File passed as a parameter to the routine.
- NAG:missingInputParameters
- NAG:optionalParameterError - Not in name/value pairs, or the name provided is not an optional parameter.
- NAG:tooManyOutputParameters
- NAG:typeError - A parameter is of the wrong type.
- NAG:unsetCellArrayError - A cell array has been passed without all elements being set.
- NAG:valueError - An incorrect value has been provided for a parameter.
- NAG:licenceError - A valid licence couldn't be found.

Errors and Warnings

- The NAG routines can produce two warnings:
  - NAG:truncationWarning - A string was truncated when copying cell array of strings to a Fortran data structure.
  - NAG:warning - The NAG routine returned an error or warning.
- The latter is important, and means that on exit the value of the argument *ifail* (or, in chapters f07 and f08, *info*) was non-zero on exit.
- For details about how to interpret this value the user should consult the Error Indicators and Warnings section of the document for the particular routine.

Errors and Warnings

- If you do not wish to see a warning then you can disable it in the usual MATLAB way, for example:

```
warning('off', 'NAG:warning')
```

- In this case it is vital that you check the value of ifail or info on exit from the routine.

Turning NAG warnings on and off

```
warning('on', 'NAG:warning');          % Turn warnings on
a = [1 2; 2 4];                        % N.B. a is a singular matrix
b = [1; 1];
[lu, ipiv, x, info] = f07aa(a,b);      % Try to solve equations
Warning: nag_lapack_dgesv (f07aa) returned a warning indicator (2)

warning('off', 'NAG:warning');         % Turn warnings off
[lu, ipiv, x, info] = f07aa(a,b);      % Try to solve equations
if (info ~= 0)                         % Must check info
  'a warning occurred'
else
  'everything OK'
end
```

Types

- The interfaces to NAG routines in the Toolbox are quite precise about the types of their arguments.
- Since MATLAB assumes by default that every number is a double users need to convert their input data to the appropriate type if it is an integer, a complex number or a logical.
- Similarly, in M-Files called by a NAG routine, the user must ensure that the results returned are of the appropriate type. This is to ensure the correct alignment between the MATLAB and Fortran types.

MATLAB Data Types - Complex

- The *complex* function constructs a complex result from real and imaginary parts.
- The statement

```
c = complex(x,y)
```

  returns the complex result *x + yi*, where *x* and *y* are identically sized real arrays or scalars of the same data type.
- y is optional, and without this argument a complex variable is retuned with zero imaginary part.
- You can test a variable with the *isreal* function, which returns false if the variable is complex.

MATLAB Data Types - Integers

- There are also 8 integer data types in MATLAB:

```
int8, int16, int32, int64,
uint8, uint16, uint32, uint64
```

- The number refers to the number of bits that are used to store the value. uint* are unsigned integers.
- For example the int32 function: `myint = int32(n)`
- You can find the range of values supported by these data types with intmin and intmax:

```
>> [intmin('int8') intmax('int8')]
ans =
    -128   127
```

Integers

- Integers used by the NAG Toolbox are chosen to be compatible with those used internally by MATLAB, and will be int32 or int64

Integers

- For portability across versions of the Toolbox that use both 32 and 64 bit integers we provide two functions:
  - nag_int(x) - converts x to the integer type compatible with the current version of the NAG toolbox
  - nag_int_name - returns the name of the integer class compatible with your version of the NAG toolbox

```
>> a = nag_int(3)
a = 3
>> nag_int_name()
ans = int64
>> b = zeros(10, nag_int_name());
>> whos
  Name        Size     Bytes  Class     Attributes
  a           1x1          8  int64
  ans         1x5         10  char
  b           10x10       800  int64
```

MATLAB Data Types - Logical

- Logical data types:

```
>> a = true; b = false; a | b
ans =
     1
```

- In many areas of MATLAB we can use integers interchangeably with logical variables - in an `if` test for example.
- However to create a logical from other datatypes, use the logical function `mylog = logical(0)`
  This returns true for all but variables with value zero.

Creating variables of correct type

- nag_int(1) - to create an integer with value 1
- complex(1,1) - to create 1.0000 + 1.0000i
- logical(0) - to create a logical that is .FALSE.
- If an object of the incorrect type is provided then a NAG:typeError will be thrown:

```
s01ea(0)
??? argument number 1 is not a complex scalar
of class double.
s01ea(complex(0))
ans = 1.0000 + 0.0000i
```

Providing m-files as arguments

- Many NAG routines allow the user to provide an m-file to evaluate a function, which might represent an integrand, or the objective function in an optimization problem, etc.
- Here is an example showing how to compute a definite integral; `'d01ah_f'` is the name of an m-file which evaluates the integrand:

```
d01ah(0, 1, 1e-5, 'd01ah_f', nag_int(0))
ans =
    3.1416
```

Providing m-files as arguments

- In this case 'd01ah_f.m' contains:

```
function [result] = d01ah_f(x)
    result = 4.0/(1.0+x^2);
```

- For every instance where a NAG routine expects an M-File to be provided, an example is given.
- Function handles can also be used.

Function handles

- These are useful in three main circumstances:
  - when the argument is an existing MATLAB command
  - when the argument is a simple expression returning one value which can be represented as an anonymous function
  - when the argument is a function that is local to an m-file.
- So we could have, for example, definite integrals:
  - `nag_quad_1d_fin_well(0, pi, 1e-5, @sin, nag_int(0))`
  - ```
    [result, abserr] = ...
       d01aj(@(x) 4.0/(1.0+x^2), 0, 1, 1e-5, 1e-5)
    ```
    ```

User Data Arguments

- Sometimes you may need to pass data to be used by a ‘callback’ routine via a ‘user’ parameter:

```
x = [0.5; 1; 1.5];
mydata = [0.14,0.18,0.22,2.10,4.39];

[xOut, fsumsq] = e04fy(nag_int(15), ...
    'e04fy_lsfun1', x, 'user', mydata);
```

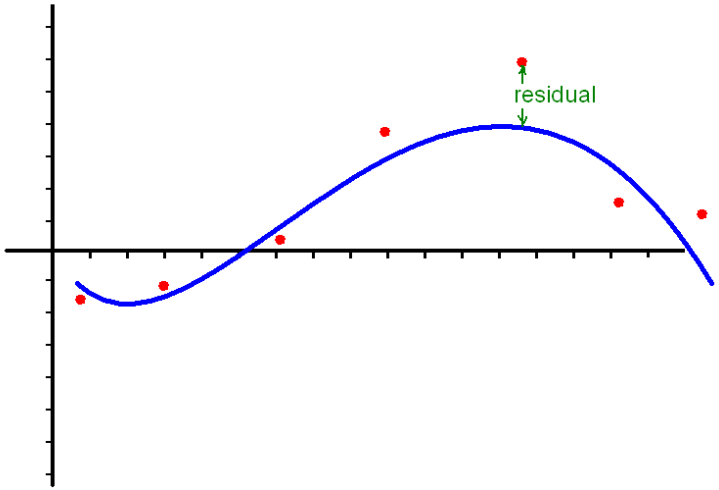
- The data in ‘mydata’ is then passed on to e04fy_lsfun1.m

FUNCTIONALITY AND DEMOS:

Curve and surface fitting

Chapter e02 – Curve and Surface Fitting

Problem to solve: given a set of data points, find the value of a function at points other than the data



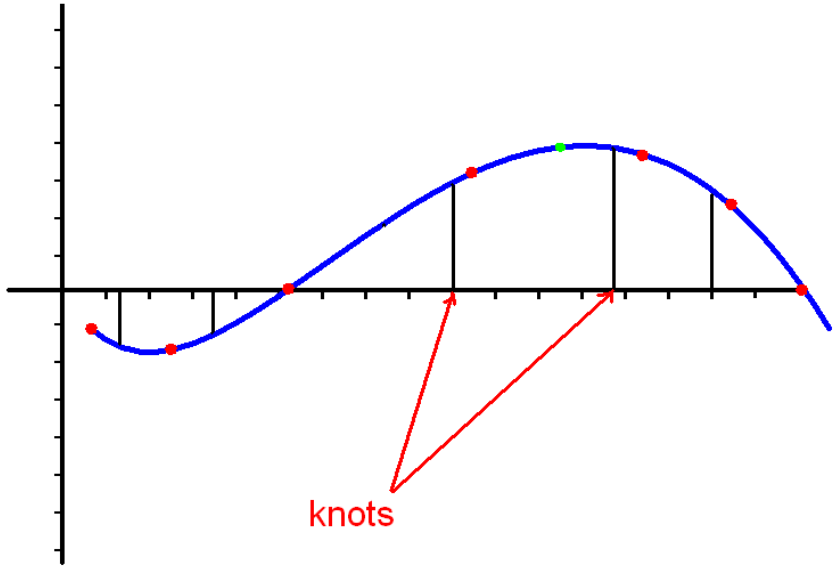
Unlike with interpolation, fitted function need not pass through data points

Typically, data contain random errors (e.g. from experimental measurement) – so interpolation is not appropriate

Smoothness of fitting function is likely to be desirable

e02 – Curve and Surface Fitting

Piecewise polynomial *splines* are useful.



Segments are joined with first and second derivative continuity at the joins (*knots*).

Chapter e02

- Data fitting usually involves minimizing the norm of the residuals
  - e.g. minimize largest residual
  - or minimize sum of squares of residuals
- Data points may be weighted according to their importance – bigger weight = more confidence
- Splines play an important role
  - Choice of knots may be crucial
  - Some routines are automatic – no need to choose knots
- Spline representation:

$$f(x) = c_1N_1(x) + c_2N_2(x) + \ldots + c_pN_p(x)$$

where $N_i(x)$ is a normalized cubic B-spline

Fitting examples – e02be and e02dc

- e02be/e02dc compute spline approximations to sets of data values, given on an interval (1D) or rectangular grid in the x-y plane (2D).
- The knots (where the individual splines meet) of the spline are located automatically.
- A single argument, s, is specified to control the trade-off between closeness of fit and smoothness of fit. “Small” s closer fit.
- In theory, a value of s = 0 will produce an interpolating spline.

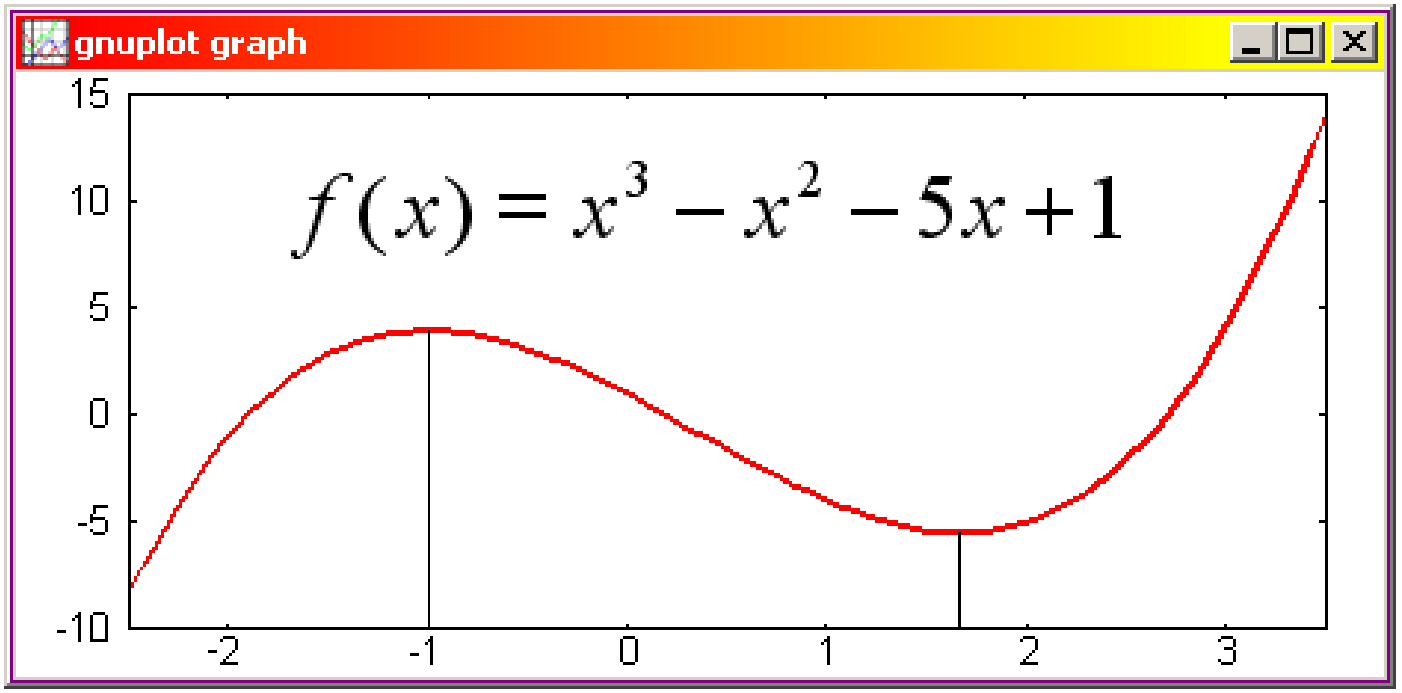
< run e02be and e02dc demos here>

FUNCTIONALITY AND DEMOS:

Optimization

What do we mean by “Optimization”?

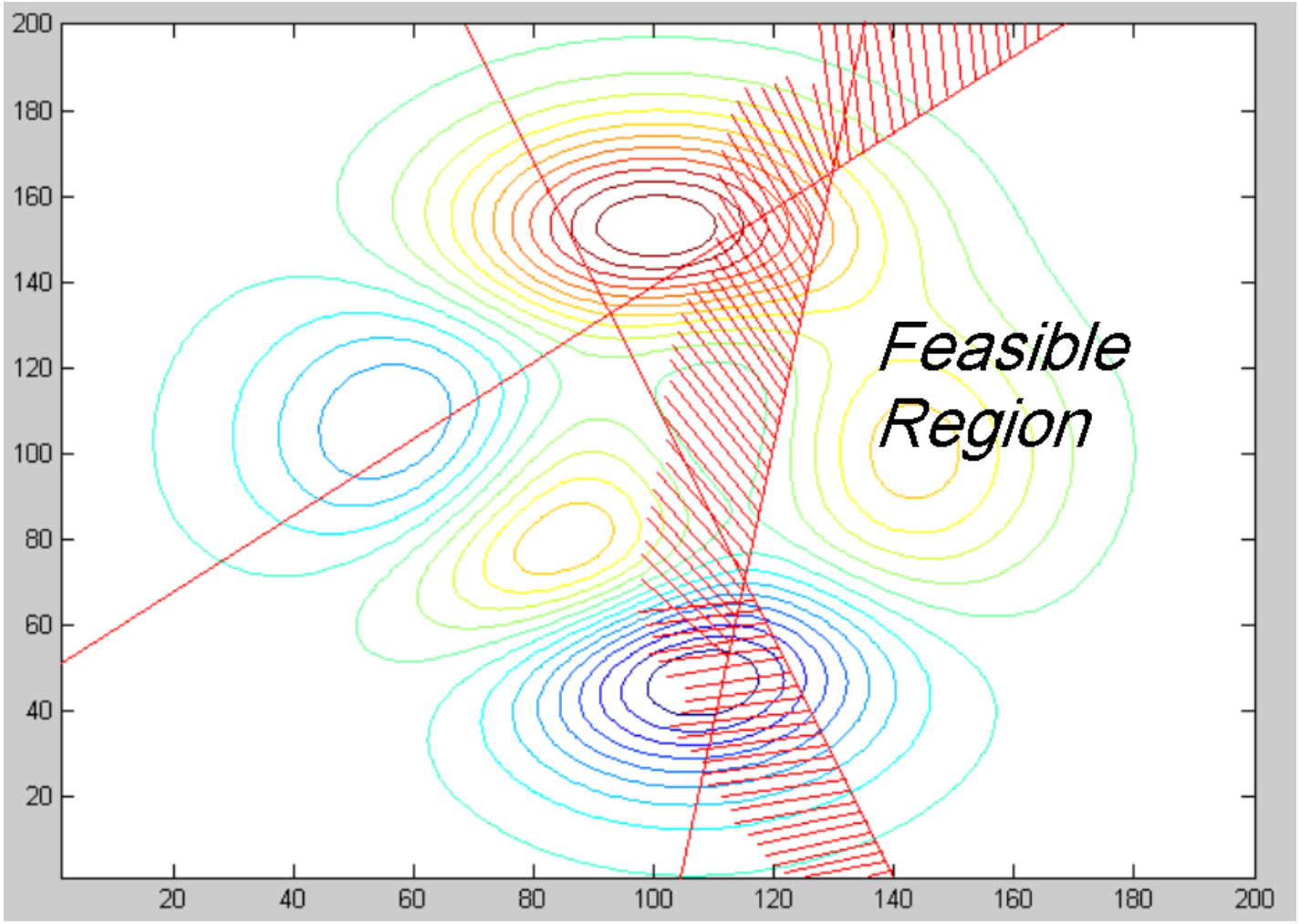
- For our purposes we mean “given a scalar real-valued mathematical function of *n* variables x_i, find values of the variables *x* that make the function as small (or as large) as possible”



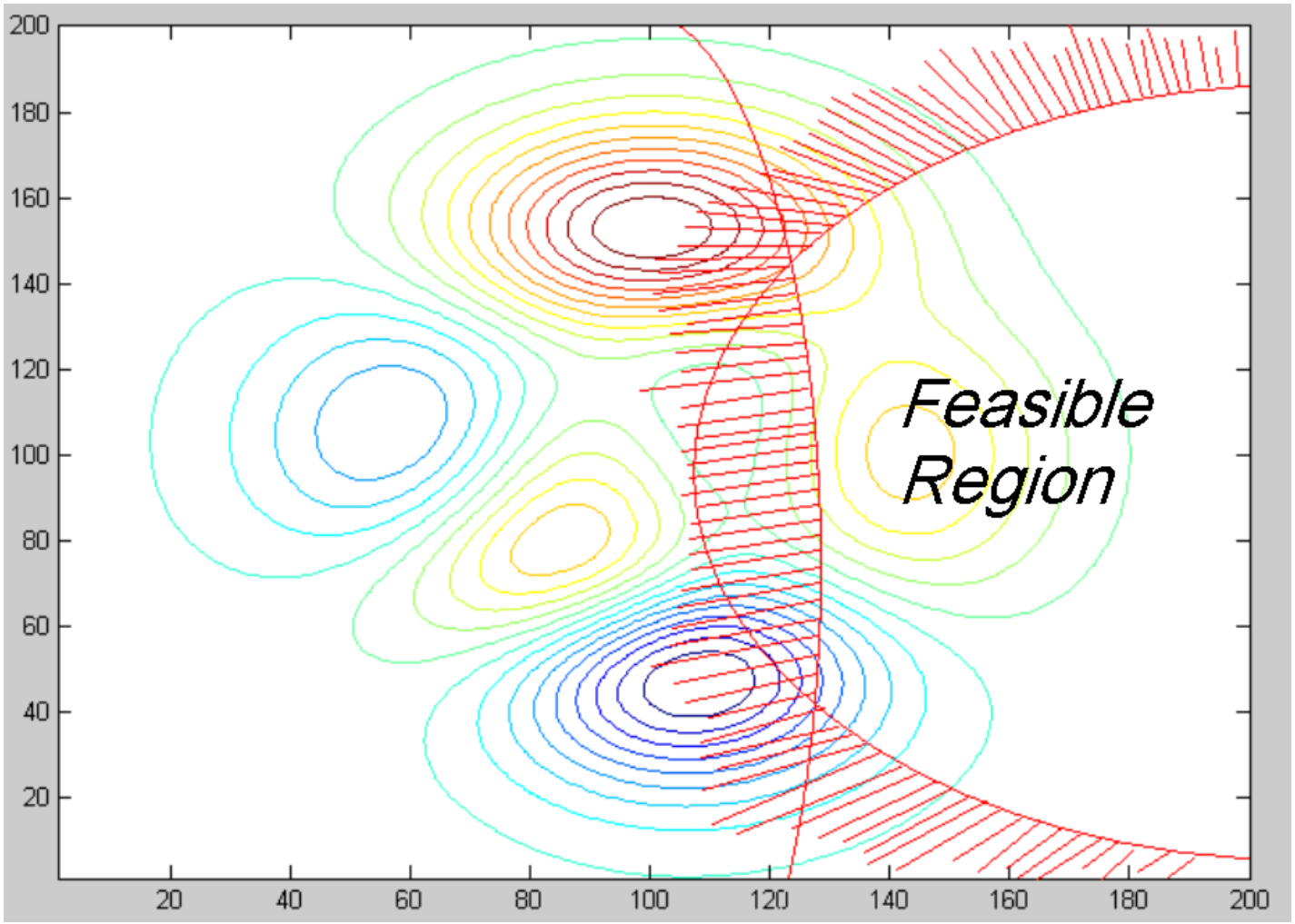
How do we do it?

- We systematically choose values of the variables x_i from within the set of values that are allowed.
- Sometimes *all* values are allowed (unconstrained optimization)
- Sometimes some values are forbidden (constrained optimization)
  - e.g. $-3 <= x_1 <= 3$ or $x_2 > 0$
- Typically the user must supply a *starting point* $\underline{x}$

Linearly constrained optimization



Optimization - Nonlinear Constraints



e04 Routine Classification

- Number of variables:
  - Single variable $f(x)$
  - Multiple variable $f(\underline{x})$
- Type of objective:
  - Linear
  - Quadratic or sum of squares
  - Nonlinear
- Constraints:
  - None
  - Simple bounds
  - Linear
  - Nonlinear

Are derivatives necessary?

- Some algorithms require them
- Some don't
- Some prefer them but can manage without
  - They may use finite difference estimates

<run steepest_descent_demo and e04uc_demo here>

NAG Optimization

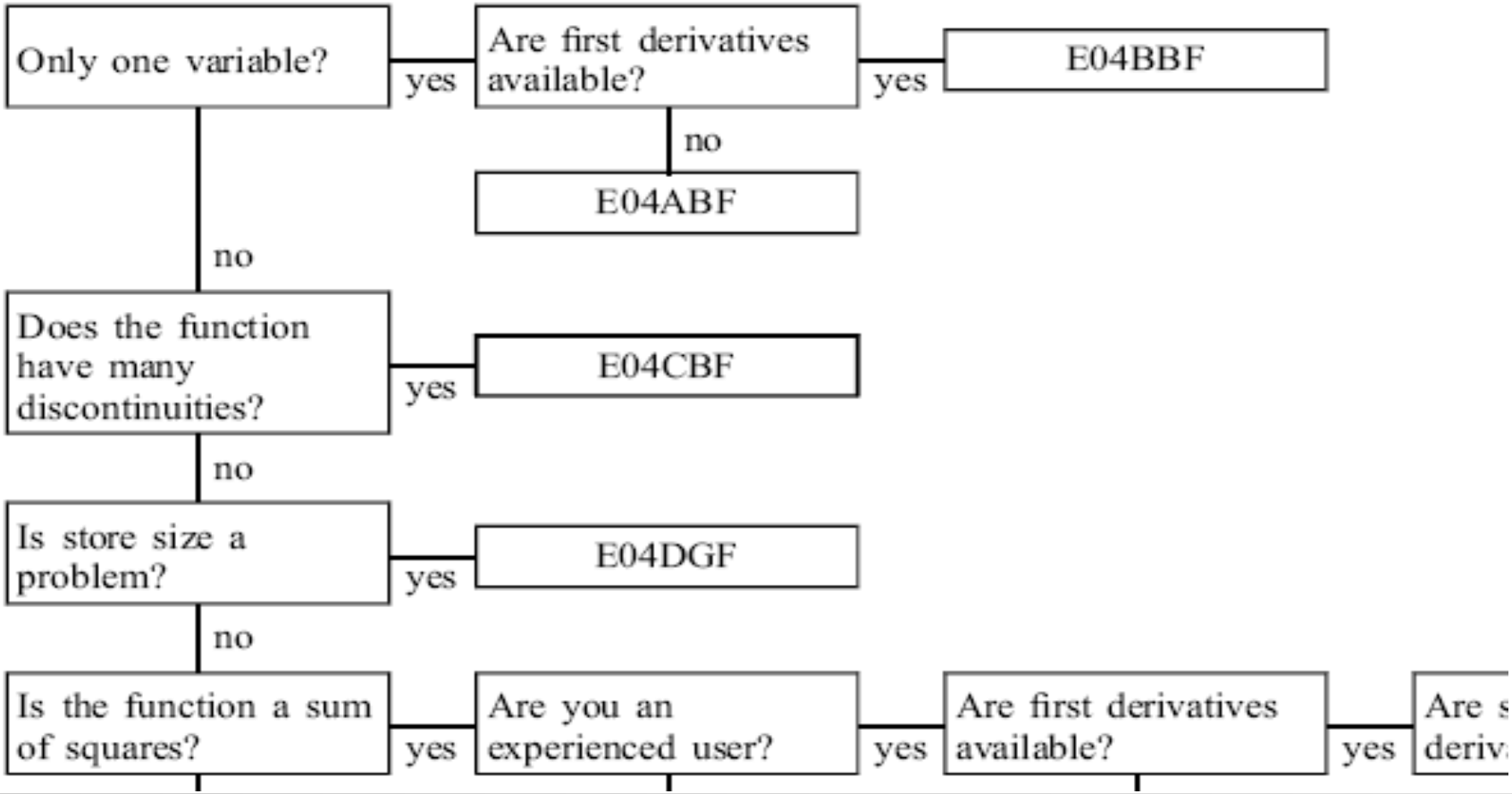
- Problems categorized according to properties of objective function:
  - linear
  - nonlinear
  - sum of squares of nonlinear functions
  - quadratic

- It is important to choose a method appropriate to your problem type, for efficiency and the best chance of success.

Best Advice – Use the Decision Trees

4 Decision Trees

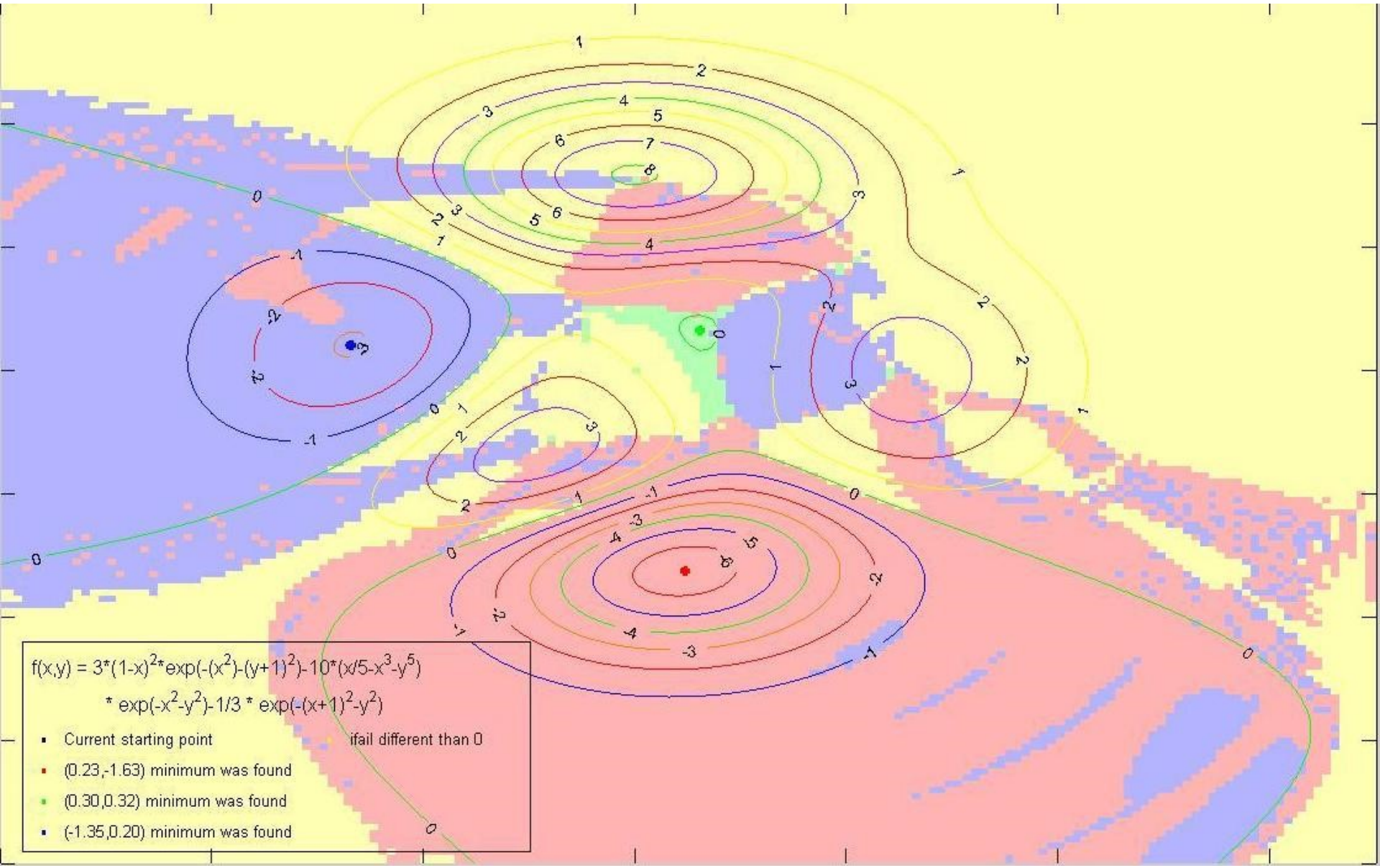
Tree 1: Selection chart for unconstrained problems



Run the e04uc demo yourself

- Go to the NAG Toolbox demos page in MATLAB
- Find **Minimization (e04uc_demo)** and click the 'Run' link
- Try adding linear constraints
  - Click twice on the contour plot to set a line
  - Click once more to determine which side of the line is feasible (the other side will turn yellow)
- Change the function from *Rosenbrock* to *Peaks*
- Try using various different starting points and observe what happens

Problem with local optimization



Hence - Global Optimization

- Multilevel Coordinate Search
  - MCS - a “box splitting” method
  - Arnold Neumaier (Vienna)
- Particle Swarm Optimization
  - No assumptions about differentiability etc.
  - No guarantees about finding optimal solution
  - Can be useful for “noisy” problems
- Multi-Start e05uc
  - Choose a large set of start positions depending on problem size and how many threads available
  - Based on solid foundation of local optimizer e04uc

Maybe try the demo **Global_Minimization (e05jb_demo)**

Continue with exercises

- Try more questions from the exercise sheets

FUNCTIONALITY AND DEMOS:

Random numbers

Chapter g05 – Random Numbers

Random numbers – used to model real-life processes.

Humans are bad at choosing them.

e.g. faking a random sequence of coin tosses is very difficult: HTHHTHTTHTHHTTTHTTHHTTHTHHTHTH …

Most people would choose a sequence easily proved not to be random

Chapter g05 – Random Numbers

Random numbers from non-uniform distributions

- Generated from uniform numbers
  - Transformation methods
  - Rejection methods
  - Table search methods

Normal (Gaussian) distribution

Student's t distribution

Beta and Exponential

Chi-Squared and Binomial

Geometric, Poisson, F, Gamma, …

Pseudo- versus Quasi-Random Numbers

Pseudo-random numbers:

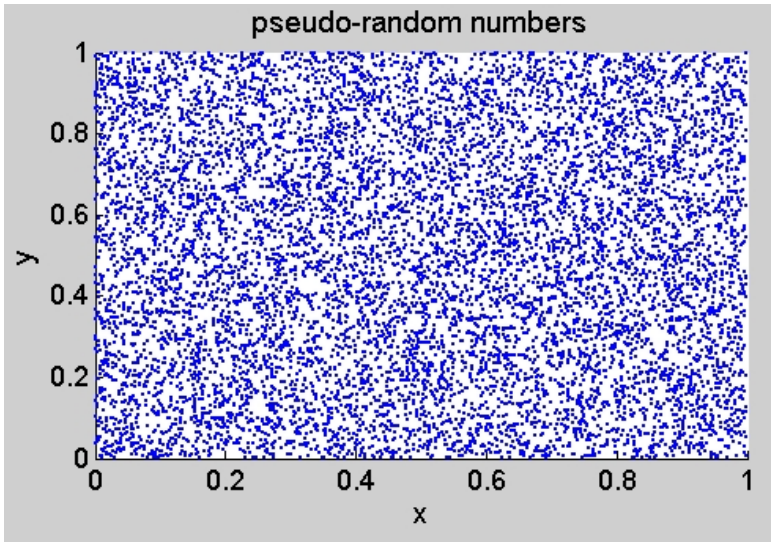
- generated systematically
  - e.g. *multiplicative congruential* $n_i = a\ n_{i-1} \mod m$
- properties close to true random numbers
  - *(assuming that a and m are chosen wisely)*
- negligible correlation between consecutive numbers

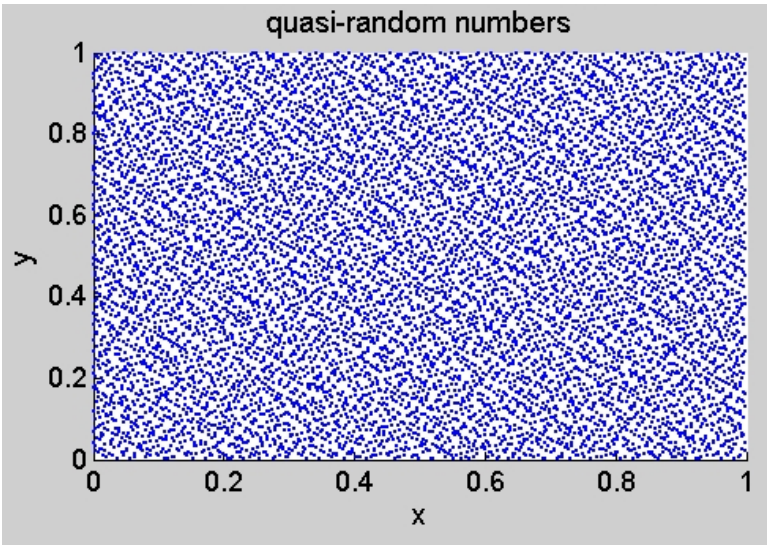
Quasi-random numbers:

- not statistically independent
- give more even distribution in space ("looks more random")
- useful for Monte Carlo integration

Pseudo- versus Quasi-Random Numbers

- Core MATLAB only has pseudo random numbers
- NAG Toolbox has pseudo and quasi
- This shows the difference:





<run quasi_integral here>

Mesh generation / PDE solution app

- In the NAG demo list, find the ‘Mesh Generation’ app (it is at the bottom of the list of demos)
- Run the app (you will probably need to read the documentation too!)

EXAMPLES:

Nearest Correlation Matrix

Nearest Correlation Matrix

- Models of more than one asset (e.g. stocks) all have *correlation*
  - What is correlation?
- Mathematically, a correlation matrix $C \in \mathbb{R}^{n \times n}$ is
  - Square
  - Symmetric with ones on diagonal
  - Positive semi-definite: $x^T C x \geq 0$ for all $x \in \mathbb{R}^n$
- Estimating correlations is difficult!
  - Historical data is typically dirty, has missing values, contains arbitrages, ...

Nearest Correlation Matrix

- Most estimation techniques will give a symmetric, square matrix with ones on the diagonal
  - They WON'T give a positive semi-definite matrix!
  - If you use these estimates, in certain conditions you will get negative variances
- NAG Library can find the "nearest" correlation matrix to a given square matrix A
  - g02aa solves problem $\min_C \|A - C\|_F^2$ in Frobenius norm
  - g02ab incorporates weights $\min_C \left\|W^{1/2}(A - C)W^{1/2}\right\|_F^2$
  - Weights useful when have more confidence in accuracy of observations for certain variables than for others

Show NCM demo

Various other demos in the NAG Toolbox

Including:

- Root finding
- Quadrature
- Global optimization
- Time Series

Continue with exercises

- Try the ODE IVP question if you haven't already

Other versions of NAG Libraries

- NAG C/C++ Library
- NAG Fortran Library
- NAG C# / .NET Library
- Also callable from
  - Microsoft Excel
  - Visual Basic
  - Java
  - Python
  - … and others

Hands-on exercise material

Exercise questions can be found here:

http://monet.nag.co.uk/nag_toolbox_training/MATLAB/toolbox_questions

Start off with the first question sheet – *short questions*

- Try whichever you like

Work alone or in pairs if you prefer