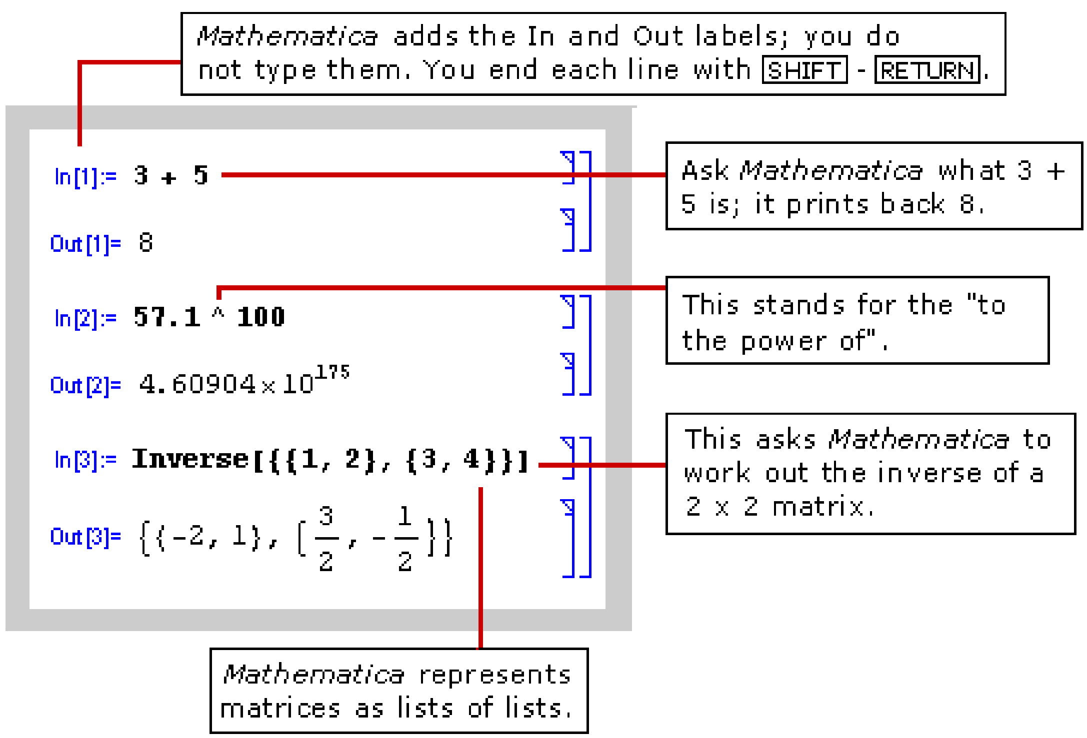
Applications: Analysis Tools

- Analysis Tools
  - Used to extract Physics from reconstructed event data.
- Generics Tools e.g.
  - Mathematica
- HEP Specific Tools e.g
  - Paw
    - Fortran based
  - ROOT
    - C++

Applications: Analysis Tools Mathematica …

The diagrams shown here come from
The Mathematica Book
by Stephen Wolfram.

- Mathematica
  - A set of interactive tools for numeric, symbolic and graphical representation of mathematical concepts
- Numeric Computation



Applications: Analysis Tools … Mathematica

- ## Symbolic Computation

This asks *Mathematica* to integrate a simple function.

```
In[1]:= Integrate[Sqrt[x] Sqrt[1+x], x]
```

Out[1]= $\sqrt{1+x}\left(\frac{\sqrt{x}}{4}+\frac{x^{3/2}}{2}\right)-\frac{\text{ArcSinh}\left[\sqrt{x}\right]}{4}$

```
In[2]:= Solve[x^2 + x == a, x]
```

This stands for mathematical equality.

Out[2]= $\left\{\left\{x\to\frac{1}{2}\left(-1-\sqrt{1+4a}\right)\right\},\left\{x\to\frac{1}{2}\left(-1+\sqrt{1+4a}\right)\right\}\right\}$

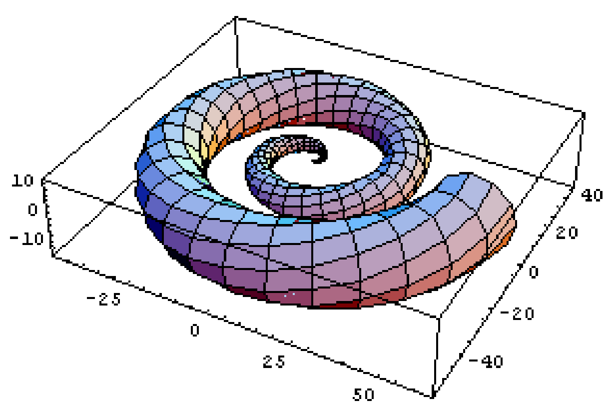
This asks *Mathematica* to solve a quadratic equation.

The result is a list of rules for *x* convenient for use in other calculations.

- ## Graphics

This creates a 3D parametric plot with automatic choices for most options.

```
In[1]:= ParametricPlot3D[{u Cos[u] (4 + Cos[v + u]),
          u Sin[u] (4 + Cos[v + u]), u Sin[v + u]},
          {u, 0, 4 Pi}, {v, 0, 2 Pi}, PlotPoints -> {60, 12}]
```




```
Out[1]= - Graphics3D -
```

Applications: Analysis Tools
Paw…

- What is PAW?
  - A tool to display and manipulate data.
- Using PAW
  - Involves operations on 3 data types:-
  - Vectors: 1, 2 or 3 dimensional arrays
  - Histograms: 1 or 2 dimensional
  - N-tuples: Tables of events
- Vectors
  - Have character identifier e.g. vec
  - 1, 2 or 3 dim. arrays e.g. vec(10,3)
  - Arbitrary size and number (almost!)
  - Create in memory, Read from disk (can filter) and Write to disk
  - Combine e.g.
  - Select subrange e.g. vec(2:5,2)
  - Draw (as histogram bins), Plot (histogram points) and Fit to function

Applications: Analysis Tools
…Paw…

- Histograms
  - Have a numeric identifier e.g. 123
  - 1 or 2 dimensional
  - Can associate errors with bins
  - Read from / Write to disk / Create in memory.
  - Combine e.g. A = B * C
  - Select subrange e.g.123(1:20)
  - Wide range of plotting and fitting facilities
- N-tuples
  - Have numeric identifiers e.g. 123
  - Record a set of n numbered (1..n) events each with m named attributes
  - Create in memory, I/O to disk.
  - Merge two or more.
  - Can plot functions of attributes, e.g. if have attributes x,y plot:-

    ```
    sqrt(x**2+y**2)
    ```
  - Can apply cuts on points to plot e.g.:-

    ```
    sin(x)+log(y)    z>1.0.and.z<10.0
    ```

Applications: Analysis Tools …Paw...

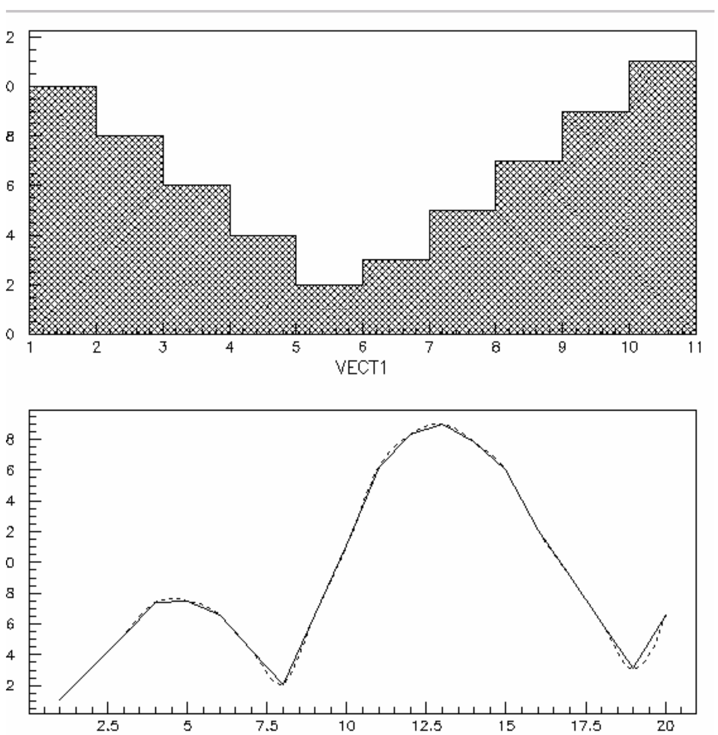
- SIGMA
  - A system for vector operation e.g:-
    ```
    sigma x=array(200,0#2*pi)
    sigma s=sin(x)
    ```
    - Create a 200 point vector x running 0 .. 2□
    - Create a 200 point vector s of sin(x)
- COMIS
  - A FORTRAN interpreter
  - Supports a subset of FORTRAN77
  - Has access to PAW's internal data structures
- Commands
  - Have a tree structure e.g.
    ```
    vector/operations/vscale
    ```
  - Can be stored in files as macros. Files have the extension `.kumac`
  - Complete programming language with:-
    - Local and Global variables
    - Flow control
    - Argument passing - so that macros can be used like subroutines
    - Embedded data files
  - Typically users develop kumac files as part of their analysis tool set

Applications: Analysis Tools
…Paw

- Demo .kumac files
  - To run some standard demos:-

```
cd some-work-dir
cp ~west/lectures/paw/*.*  ./
paw
   (press return for default display)
  exec pawex1.kumac
  exec pawex2.kumac
  ...
  exec  pawex29.kumac
```

  - For example `exec pawex1.kumac` should produce:-



- Paw Tutorial:-
  - See `http://wwwinfo.cern.ch/asd/paw/`

Applications: Analysis Tools
ROOT...

ROOT: Tool and a Framework for OO Data Analysis

- As a Tool
  - Enter commands to display and manipulate data
  - Commands are C++
    - (covers ~85% of full language – including simple templates)
  - Can use to prototype new code
- As a Framework
  - Use ROOT as a library and link in user C++ code
  - Can use result as a tool
- Supports User Classes
  - User can define new classes
  - Can do this either as Tool or Framework
  - These can inherit from ROOT classes.

Applications: Analysis Tools
...ROOT...

- Base Classes
  - Objects
  - Files and Directories
  - I/O
  - System interface
  - Basics maths
- Containers
  - Collections
  - Lists
  - Arrays
  - Maps (hashs)
- Histogram and Minimisation
  - Up to 3D
  - Profile
  - Minuit (general fitting package)
- Trees and N-tuples
  - Generalise n-tuple concept to a binary tree of objects
- Matrices

Applications: Analysis Tools
...ROOT...

- 2D Graphics
  - lines, text, shapes etc.
- 3D Graphics and Geometry
  - 3D shapes e.g. cone, helix
  - Geometry description
    - Tree of nodes
    - Nodes properties: shape, material, rotation
- GUI
  - Toolkit to build GUI
- Meta Data
  - For describing classes
- Network
  - Access to network, including HTTP
- Documentation
  - Build HTML directly from source code
- Interactive Interface
  - For user application to act as a tool
- Parallel Process Support

Applications: Analysis Tools
...ROOT...

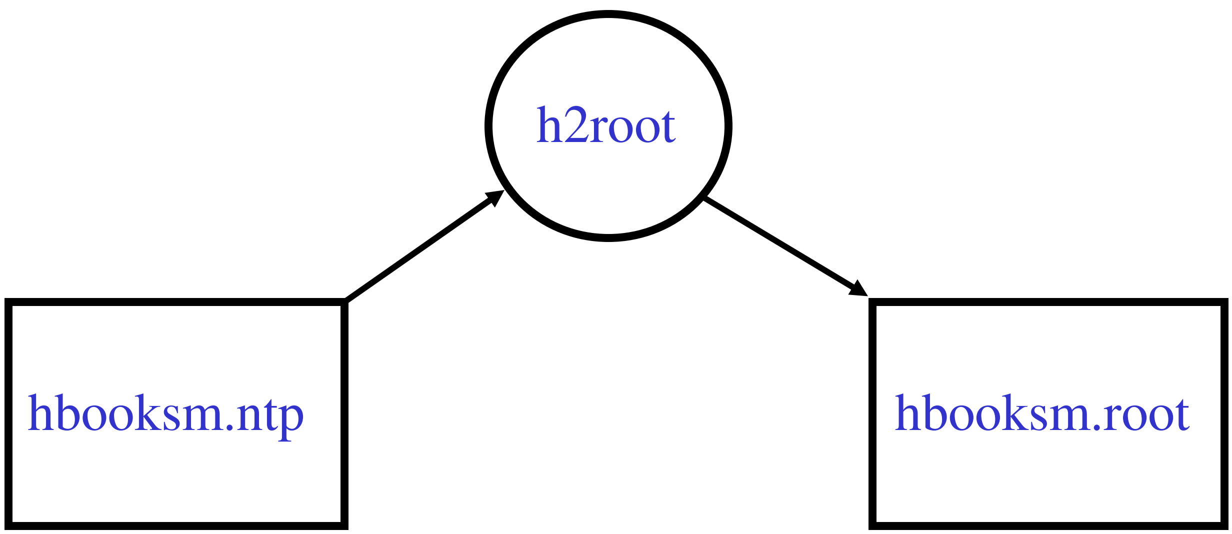
- Naming Convention
  - TName e.g. TList
- TObject
  - Is the primordial object
  - Most other classes inherit from it
  - Provides base for generic operations such as:-
    - I/O, Graphics, Containerisation
- Graphics: TCanvas, TPad
  - TCanvas is rectangular window holding TPads.
  - TPad maps to a rectangular area on a TCanvas
  - TPad holds a list of objects (including TPads) to be displayed

Applications: Analysis Tools ...ROOT...

- File access: TDir and TFile
  - A TDir is a directory.
    - It holds a list of named objects (can include other TDir)
  - A TFile is a file.
    - It consists of a series of TDir objects.
  - Reading from a file
    - Involves passing TFile the name of the object to be retrieved. It returns pointer.
- Event I/O: TTree and TBranch
  - General concept of an event
    - A heterogeneous collection of objects.
  - All have to be output together
  - A TBranch holds a collection of objects
    - It can include TBranch. It has its own buffer.
  - A TTree is a collection of TBranch.
    - It synchronises I/O.
  - But, can just input partial event
    - Select TBranch
    - Input rest of Ttree conditionally

Applications: Analysis Tools
...ROOT...

- Working with a PAW HBOOK N-tuple
  - Converted using h2root:-



- hbooksm.ntp
  - n-tuple 800
  - has variables:-
    - Vx_x
    - Vx_y
    - Vx_z

- hbooksm.root
  - TTree h800
  - TBranch:-
    - Vx_x
    - Vx_y
    - Vx_z

Applications: Analysis Tools
...ROOT...

```
{
// Clear out any object created by user in the current session by
// sending Reset message to the master object gROOT.

  gROOT->Reset();

// Create a canvas (window) and within it define 3 pads (sub-windows
// holding graphical material).

// Create canvas giving name, title,  top left corner, width and height
// in pixels.

  c1 = new TCanvas("c1","ROOT Demo",200,10,700,500);

// Create pads giving name, title, limits (as fraction of canvas) and
// background colour (8 = white)

  pad1 = new TPad("pad1","Pad1: (Top half)",     0.02,0.52,0.98,0.98,8);
  pad2 = new TPad("pad2","Pad2: (Bottom left)", 0.02,0.02,0.48,0.48,8);
  pad3 = new TPad("pad3","Pad3: (Bottom right)",0.52,0.02,0.98,0.48,8);

// Tell the pads to draw themselves.  This actually adds them to the list
// of objects that the canvas holds.  Later, when the canvas is sent the
// Update message, it will send an Update message to all its pads.

  pad1->Draw();
  pad2->Draw();
  pad3->Draw();

// Create a File object as an input from the file hbooksm.root.

  TFile *hfile = new TFile("hbooksm.root", "READ");

// Set up a Tree object pointer by asking hfile to find the object whose
// name is h800 (the name created by h2root for n-tuple 800).  The Get
// message returns a pointer to Object so have to be cast up to a Tree.

  TTree *my_ntuple = (TTree *) hfile->Get("h800");
```

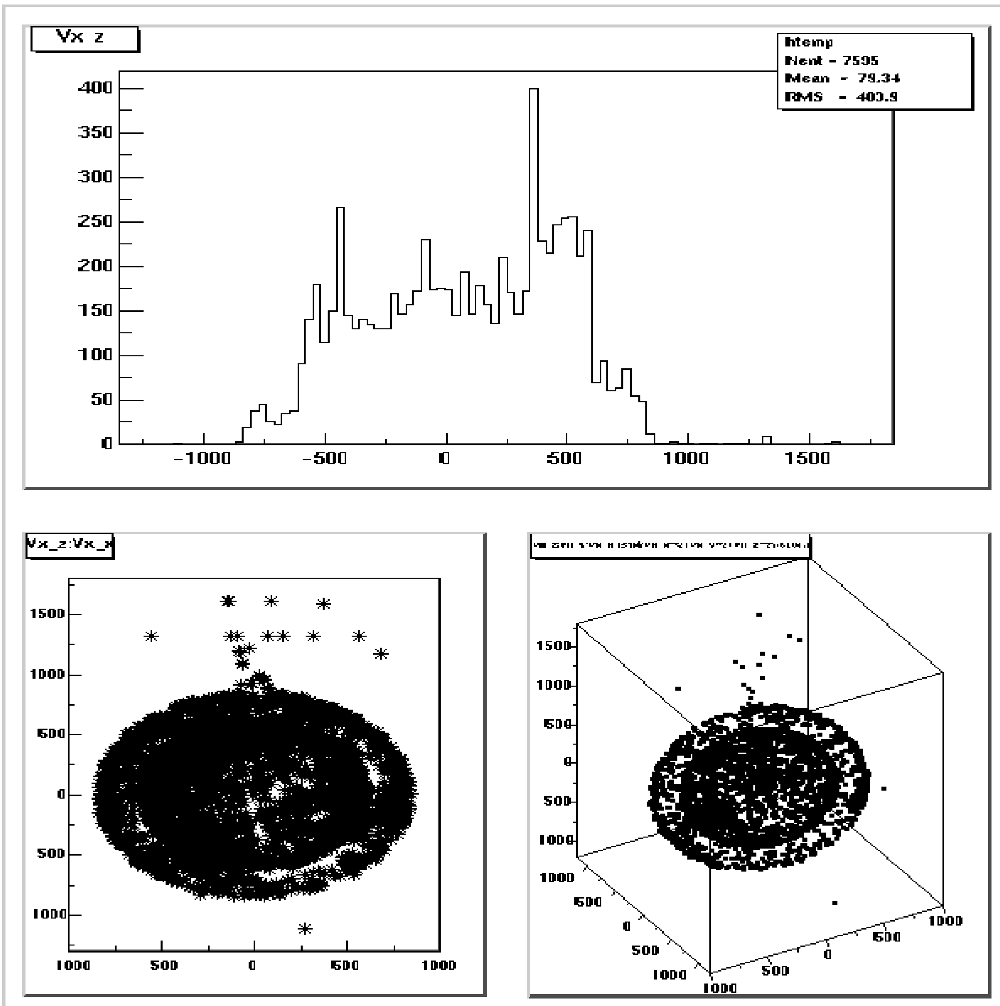
Applications: Analysis Tools
...ROOT...

```
// Make pad1 the current working graphics directory by sending it cd
// (cf. Unix).  From now on, any Draw message will draw in this pad.

   pad1->cd();

// Send the n-tuple a Draw message, supplying the expression to be drawn.
// This automatically creates and fills a histogram object (like PAW).
// pad1 will contain this histogram.

   my_ntuple->Draw("Vx_z");

// In a similar way, plot a 2d histogram in pad2.

   pad2->cd();

// This time we tell the n-tuple to change the defaults to be used when
// creating the 2d histogram.

   my_ntuple->SetFillColor(5);
   my_ntuple->SetMarkerStyle(3);

// Note the syntax is different to PAW (Vx_z%Vx_x).

   my_ntuple->Draw("Vx_z:Vx_x");

// Finally plot a 3d plot in pad3.  This time, we also place a cut on the
// data to be plotted.

   pad3->cd();
   my_ntuple->SetMarkerStyle(7);
   my_ntuple->Draw("Vx_z:Vx_y:Vx_x",
                   ”sqrt(Vx_x**2+Vx_y**2+Vx_z**2)<5000.");

// Now tell the canvas to update itself, causing all its pads to tell all
// the objects they contain to paint themselves.

   c1->Update();

}`
```

Applications: Analysis Tools
...ROOT

- To run this example:-

```
root
.x ~west/lectures/root/demo/root_demo.C
```



- Documentation

```
– Root home page:-
http://root.cern.ch/root/
From there look at the Root Tutorial
```