Earlier steps in Mathematica, Perl, GnuPlot, Emacs Lisp, PAW, and ROOT

SEAL-MINUIT

```
#include "Minuit/MnMinos.h"
#include "Minuit/FunctionMinimum.h"
#include "Minuit/FCNBase.h"
#include "Minuit/MnFunctionCross.h"
#include "Minuit/MnCross.h"
#include "Minuit/MinosError.h"

std::pair<double,double> MnMinos::operator()(unsigned int par, unsigned
int maxcalls) const {
  MinosError mnerr = minos(par, maxcalls);
  return mnerr();
}

double MnMinos::lower(unsigned int par, unsigned int maxcalls) const {

  MnUserParameterState upar = theMinimum.userState();
  double err = theMinimum.userState().error(par);

  MnCross aopt = loval(par, maxcalls);

  double lower = aopt.isValid() ? -1.*err*(1.+ aopt.value()) :
(aopt.atLimit() ? upar.parameter(par).lowerLimit() : upar.value(par));

  return lower;
}
```

C++

Fit function

```
 DOUBLE PRECISION FUNCTION GBWKF(RM,GAM,WSPREAD,HC)
       IMPLICIT NONE
c  implicit real (A-H,O-Z), integer (I-N)

C   Observed shape of resonance peak, starting with Breit-Wigner
C   Normalized per unit W, MeV^-1
C   F_KF(x,s) convoluted with unit-area Breit-Wigner, and with beam
resolution
C    unit-area Gaussian
C   If GAM<1 keV, the Breit-Wigner is replaced with a delta function
C   BW(w) = (GAM/2*pi)/[(w-M)^2+GAM^2/4] (unit integral over w)
C   Gaussian(w) = exp{-.5*[(w-M)^2/WSPREAD^2]}/(2*pi*WSPREAD) (unit int...)

       INTEGER   MKBIN, MTBIN, JTBIN
       REAL*8    ROOT2, PI, RTWOPI, RTWOBPI, PMAX, TINY

 REAL*8    FKFT(MKBIN)               ! F_KF(x,s) (Kuraev-Fadin eq.28)
 REAL*8    HA(MKBIN)                   ! WA-RM of centroid WA bin
 REAL*8    DWA(MKBIN)                  ! bin width in WA-RM
 REAL*8    HT(0:MTBIN)
 REAL*8    DWT(0:MTBIN)                ! WT-RM bin center
 REAL*8    BWKF(-MTBIN:MTBIN)          ! result of BW-F_KF convolution
...
 DO 11 ITBIN=-MTBIN,MTBIN            ! sum over WT's
   H=HT(IABS(ITBIN))                 ! WT-RM
   IF(ITBIN.LT.0)H=-H
   SIGS=(H-HC)/(ROOT2*WSPREAD)       ! sqrt of exponent
   IF((ITBIN.EQ.MTBIN).AND.(SIGS.LT.2.))GO TO 12 ! running out of bins?
   IF(SIGS.GT.3.)GO TO 13            ! Gaussian getting negligible?
   P=SIGS**2                         ! exponent
   IF(P.LT.PMAX)THEN
     IF(DWT(IABS(ITBIN)).GT..2*WSPREAD)GO TO 12  ! binning too coarse?
           SUM=SUM+EXP(-P)*BWKF(ITBIN)*DWT(IABS(ITBIN))
   ENDIF
 11 CONTINUE
 13 GBWKF=SUM/(RTWOPI*WSPREAD)          ! convolution
C         1/(sqrt(2*pi)*WSPREAD) normalizes Gaussian
 RETURN
 END
```

FORTRAN

PyMinuit

```
// g++ minuit.cpp -I/nfs/cleo3/Offline/rel/current/other_sources/pyth
include/python2.4/ -I/cdat/daf9/mccann/software/src/minuit/Minuit-1_5_2 /
cdat/daf9/mccann/software/src/minuit/Minuit-1_5_2/src/*.o -shared -o
_minuit.so
// g++ minuit.cpp -I/usr/include/python2.3 -I/root/src/Minuit-1_5_2/ /
root/src/Minuit-1_5_2/src/*.o -shared -o _minuit.so

#include <Python.h>
#include "Minuit/MnUserParameters.h"
#include "Minuit/MnMigrad.h"
#include "Minuit/MnMinimize.h"
...
PyObject* dominos(PyObject *self, PyObject *args)
{
   // parameter list:
   PyObject *p_fcn;                      // objective function p_fcn
   int npar;                             // number of parameters in p_fcn
   double up;                            // 1 for chi^2, 0.5 for loglike
   PyObject *p_min;                      // the minimum you previously found

   if (!PyArg_ParseTuple(args, "OidOiiOOiOO", &p_fcn, &npar, &up, &p_min,
&maxcalls, &strategy, &p_dolower, &p_doupper, &parnum, &p_grad,
&p_checkgrad)) {
      PyErr_SetString(PyExc_TypeError, "calling format must be: FCN(f),
npar(i), up(d), minimum(FunctionMinimum), maxcalls(i or 0), strategy(i),
dolower(b), doupper(b), parnum(i), gradient(f or None), checkgrad(b or
None)");
      return NULL;
   }
```

C API

Fitting script

```
# get_runs has been given a thorough look-over: it is correct (7 Oct
# 2005) (get_runs contains all corrections, from numbers of events to
# real, live cross-section.)

from minuit import *
execfile("/home/mccann/antithesis/utilities.py")
import gbwkf
import gbwkftau
...
def dofitgauss(h):
  def gauss(m, s, x): return exp(-(x-m)**2/2./s**2)/sqrt(2.*pi)/s
  def fitgauss(m,s):
    c = 0.
    for x in h.data:
      c += -log(gauss(m, s, x))
    return c
  m = Minuit(fitgauss, start=[0., 1.], up=0.5)
  m.migrad()
  m.minos([0,1])
  err0 = (m.minos_errors[0][1] - m.minos_errors[0][0])/2.
  err1 = (m.minos_errors[1][1] - m.minos_errors[1][0])/2.
  return m.values[0], err0, m.values[1], err1, lambda x:
0.1*extraarea*len(h.data)*gauss(m.values[0], m.values[1], x)
```

python

(pickle)

GNU plotutils

biggles

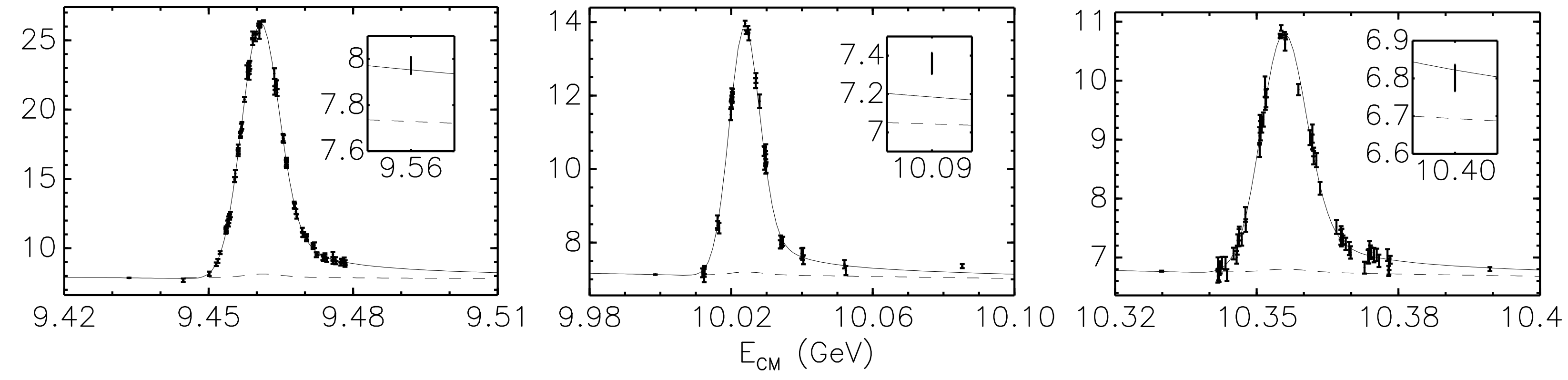
Numeric

Plotting script

```
from math import *
import biggles, Numeric, cPickle as pickle
import gbwkf
import gbwkftau

allthat = pickle.load(file("/home/mccann/antithesis/novemberdata.p"))
u1runs = allthat["u1runs"]
u2runs = allthat["u2runs"]
u3runs = allthat["u3runs"]
...
q = biggles.FramedPlot()
adddata(q, [None], u2data["high"], 0.)
addfunc(q, thefunc, 10080., 10090.)
addfunc(q, thefunc_bkgnd, 10080., 10090., linetype="dashed")
```

python


E_{CM} (GeV)