Comp-Exe 1: χ^2-function and χ^2-probability distributions

Macro Chi2_rnd.C

generates χ^2 distributions according to $\chi^2 = \sum_{i=1}^{n_{dof}} x^2$, where x are simple gaussian random numbers (with mean 0 and width 1.). This corresponds to the χ^2 of n_{dof} measurements of an observable (e.g. measuring the electron mass) to the known true value.

- Steering parameters in the macro:
  - n_{dof} = Number of measurements
  - ntr = Number of repeated experiments
- Output
  - Histo $chi2h$ - the resulting χ^2 distribution
  - Histo $pchi2h$ - the resulting χ^2-probability distribution

Tasks:

a) Have a few minutes look in the code and try to understand what's going on

b) Run the macro as it is by .x Chi2_rnd.C and plot the histograms by chi2h→ Draw(); pchi2h→ Draw();

c) Study the *mean* and *rms* of the χ^2 function and the χ^2-probability distribution versus n_{dof}: run the code repeatedly with n_{dof} set to (*by simple editing*) values $n_{dof} = 1, 4, 10, 100$ and read off from the $chi2h$ and $pchi2h$ histograms the mean values and RMS and fill them in the table below. Try to obtain simple scaling laws for the dependences vs n_{dof}.

	χ^2		χ^2/n_{dof}		$prob(\chi^2, n)$	
N_{dof}	mean	rms	mean	rms	mean	rms
1						
4						
10						
100						

d) Extend the programme (Places are marked in the macro by //*****): Book and fill histogram of the χ^2/n_{dof} distribution and repeat exercise c) with it and fill resulting mean and rms values in the table

e) Judge on the chances of obtaining the following two χ^2 values which have the same χ^2/n_{dof}:

1. $n_{dof} = 5$, $\chi^2 = 7$
2. $n_{dof} = 500$, $\chi^2 = 700$

Hint: the χ^2-probability is a good measure for this.

Comp-Exe 2: Toy-experiments of fits of a constant

Macro p0toyf.C generates toy experiments for the fit of a constant ("p0-fit") from several measurements of the same resolution. A physics example for this would be the determination of the (average) vertical position of a horizontal flying track in track-detectors. Additional Problem: the detectors are noisy, for each detector sometimes instead of a good signal hit a flatly distributed noise hit is observed.

- Steering parameters in the macro:
  - $Ntra$ = Number of repeated toy experiments (Default is 1000)
  - $Ndet$ = Number of measurements (Default is 10)
  - $frac\_noise$ Average fraction of noise hits (Default is 0.)
- Output
  - Histo $p0d$ - Distribution of residual: fitted constant - true value
  - Histo $chi2d$ - χ^2 distribution from the $p0$-fits
  - Histo $pchi2d$ - χ^2-probability distribution from the $p0$-fits
  - Histo a - shows the p0-fit for the last toy experiment

Tasks:

- Take a deep breath and have a few minutes look in the code and try to understand what's going on

a) Run the macro as it is with .x p0toyf.C and plot the histograms by p0d→ Draw(); chi2d→ Draw(); pchi2d→ Draw(); a→ Draw(); Fill the RMS value of the $p0d$ histogram in the table below (first row) and also the mean values of the $chi2d$ and $pchi2d$ distributions.

b) Now edit the macro and set $frac\_noise$=0.1;

- run the macro again and fill the obtained values in the table (second row).
- How much has the RMS of the residuals increased?
- Can you identify the bad track-fits in the $pchi2d$ distribution?

c) Rejection of tracks with bad χ^2-probability: Book another residual histogram $p0d\_rej$ in the macro:

- fill it only for the case that the χ^2-probability is not too small (see also hints in the code)
- fill in the third row of the table how many tracks survive this selection and the resulting RMS of the residuals.

d) Hit-outlier-rejection: Advanced method for super experts!

Book another three histograms $p0d\_iter$, $chi2d\_iter$ and $pchi2d\_iter$ in the macro and

- try hit outlier rejection and repeating the track-fit (see detailed instructions in the code);
- fill the resulting RMS of $p0d\_iter$ into the fourth row of the table.
- How does the RMS results of this method compare to the other cases a)-c)?

- Repeat the above exercises b)-d) for increased $frac\_noise$=0.5 and fill the results in the table. How much are the RMS results worse in this case?
- If time permits repeat the above exercises a)-d) for different number of detectors $ndet = 5$ and study how the results change qualitatively.

Task	*Ndet*	*frac_noise*	Outlier rejection	Resid.-distr.	#tracks	Resid. RMS	chi2d mean	pchi2d mean
a)	10	0.0	No	p0d	1000			
b)	10	0.1	No	p0d	1000			
c)	10	0.1	Reject tracks with bad pchi2	p0d_rej			–	–
d)	10	0.1	Outlier hit-rejection and repeated track-fit	p0d_iter	1000		–	–
b)	10	0.5	No	p0d	1000			
c)	10	0.5	Reject tracks with bad pchi2	p0d_rej			–	–
d)	10	0.5	Outlier hit-rejection and repeated track-fit	p0d_iter	1000		–	–

Comp-Exe 3: World average of m_W

Macro AverageMW.C fits a world average value of m_W using five different results from Tevatron, LEP and NuTeV.

Tasks:

a) Have a brief look in the code and try to understand what's going on

b) Run the macro as it is by .x AverageMW.C

- Is the obtained χ^2 value reasonable?

c) Change in the macro the variable Bool_t inclu NuTeV to kFalse (this takes out the NuTeV result from the fit) and run the macro again:

- How much does the error on m_W change?
- Is the obtained χ^2 value reasonable?
- Do you think this rejecting of the NuTeV result is justified?

Comp-Exe 4: World average of $m_{K^\pm}$

Macro AverageMK.C fits a world average value of $m_{K^\pm}$ using six different measurements.

Tasks:

a) Run the macro as it is by .x AverageMK.C

- Is the obtained χ^2 value reasonable?
- Can you identify good candidates for "outliers"?

b) Increase the errors of all measurements in the macro by a common scaling factor, e.g. MK→SetBinError(i+1, err_MK[i]*1.1) and run the macro again and again until you find a scaling factor for which the resulting $\chi^2/ndof \approx 1 \quad \Rightarrow$ thus obtain a final result a la PDG (particle data group).

Comp-Exe 5: Straight line trajectory fit

Physics example: A muon track is measured in four layers of streamer tube detectors at x positions of 4., 5., 6. and 7. (in cm), with a measurement precision for y of 0.5 cm. The goal is to determine its trajectory, assuming it to be a straight line.

Macro StraightLineFit.C fits a straight line track trajectory through four measured points.

- Steering parameters in the macro:
  - $xmin$, $xmax$ = Interval of the trajectory displayed
- Output:
  - Histogram $data$ (it's of the type TGraphErrors)
  - Plots are drawn of the
    - fitted histogram with error bands
    - error ellipse of the two fitparameters

Tasks:

a) Run the macro as it is by .x StraightLineFit.C and fill the fit results for $p0$, $p1$, their errors and correlation into the table below

b) Precision of trajectory: Evaluate (by eye) from the shown error bands at which point roughly the trajectory is known best and with which precision (fill the results in the table below)

c) Precision of extrapolated trajectory: Evaluate the precision of the extrapolated trajectory at $x = 100$ (Hint: Change $xmax$ to large value and run the macro again)

d) Effect of shift of x coordinate origin: Shift all four $xVal$ points in the macro (simply by overwriting by hand) by a constant value -5.5, set $xmin = -4.$ and $xmax = 4.$ and run the macro again. Fill the fit results in the table. Can you explain why the correlation of $p0$ and $p1$ has changed?

e) Apply a very precise vertex constraint at the origin: Change N to 5 and add a new first point to the measurement points list with $xVal = 0.0$, $xErr = 0.0$, $yVal = 0.0$ and $yErr = 0.0001$ (just by hand). Run the macro again and write down the fitted results in the table. How much are the parameter errors reduced by adding this extra point?

	Straight line fit trough four points
Task a)	$p0 =$ $p1 =$ corr $=$
Task b)	x-best precision $=$ y-error $=$
Task c)	y-error$(x = 100) =$
Task d)	Shifting all x values by -5.5: $p0 =$ $p1 =$ corr $=$
Task e)	Adding vertex constraint at $x = 0$: $p0 =$ $p1 =$ corr $=$