Reactor neutrino experiments and systematical errors in GLoBES

Joachim Kopp

Max-Planck-Institut für Kernphysik, Heidelberg

24 January 2007

Outline

1. Reactor neutrino experiments
   - Important systematical error contributions
   - χ^2 analysis inlcuding systematical errors

2. Systematical errors in GLoBES
   - Internal χ^2 functions
   - User-defined χ^2 functions
   - Example: Implementation of Double Chooz
   - Performance considerations

3. Some physics results

Outline

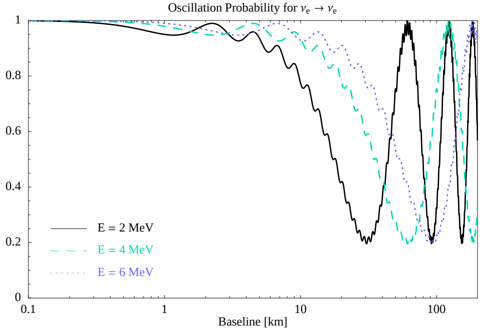
1. Reactor neutrino experiments
   - Important systematical error contributions
   - χ^2 analysis inlcuding systematical errors
2. Systematical errors in GLoBES
   - Internal χ^2 functions
   - User-defined χ^2 functions
   - Example: Implementation of Double Chooz
   - Performance considerations
3. Some physics results

Oscillation probability

$\bar{\nu}_e$ disappearance probability ($\alpha = \Delta m^2_{21}/\Delta m^2_{31}$, $\Delta = \Delta m^2_{31} L/4E$):

$$P_{\bar{e}\bar{e}} = 1 - c_{13}^4 \sin^2 2\theta_{12} \sin^2 \alpha\Delta - \sin^2 2\theta_{13} \sin^2 (\alpha - 1)\Delta$$
$$+ \frac{1}{2} c_{12}^2 \sin^2 2\theta_{13} [\cos 2\Delta - \cos 2(\alpha - 1)\Delta].$$

- Independent of δ_{CP}
  $\Rightarrow$ (Almost) no correlations
- But: Sensitive to tiny systematical errors



Reactor-side errors

- Total neutrino flux
  (from thermal measurements)

Reactor-side errors

- Total neutrino flux
  (from thermal measurements)
- Neutrino spectrum
  (from experiments and theoretical models
  on nuclear fission)

Reactor-side errors

- Total neutrino flux (from thermal measurements) — 2%
- Neutrino spectrum (from experiments and theoretical models on nuclear fission) — 2%

Reactor-side errors

• Total neutrino flux (from thermal measurements)	2%	corr.
• Neutrino spectrum (from experiments and theoretical models on nuclear fission)	2%	corr.

In a near-far setup, the correlated errors will cancel.

Detector-side errors

- Cross sections
  Scintillator properties
  Spill-in/spill-out

Detector-side errors

- Cross sections
  Scintillator properties
  Spill-in/spill-out
- Fiducial mass
  Detector normalization
  Analysis cuts

Detector-side errors

- Cross sections
  Scintillator properties
  Spill-in/spill-out
- Fiducial mass
  Detector normalization
  Analysis cuts
- Energy calibration

Detector-side errors

- Cross sections
  Scintillator properties
  Spill-in/spill-out
- Fiducial mass
  Detector normalization
  Analysis cuts
- Energy calibration
- Backgrounds

Detector-side errors

• Cross sections Scintillator properties Spill-in/spill-out	2.0%
• Fiducial mass Detector normalization Analysis cuts	0.6%
• Energy calibration	0.5%
• Backgrounds	1.0%

Detector-side errors

• Cross sections Scintillator properties Spill-in/spill-out	2.0%	corr.
• Fiducial mass Detector normalization Analysis cuts	0.6%	uncorr.
• Energy calibration	0.5%	uncorr.
• Backgrounds	1.0%	partly

Again, correlated errors will cancel in a near-far setup.

χ^2 analysis for a 2-detector reactor experiment

- Compare event rates for “true” oscillation parameters $\vec{\Theta}$ and no systematical errors vs. event rates for fit parameters $\vec{\Theta}'$

χ^2 analysis for a 2-detector reactor experiment

- Compare event rates for "true" oscillation parameters $\vec{\Theta}$ and no systematical errors vs. event rates for fit parameters $\vec{\Theta}'$
- Contributions from near detector (N) and far detector (F)

χ^2 analysis for a 2-detector reactor experiment

- Compare event rates for "true" oscillation parameters $\vec{\Theta}$ and no systematical errors vs. event rates for fit parameters $\vec{\Theta}'$
- Contributions from near detector (N) and far detector (F)
- Contributions from all bins i

χ^2 analysis for a 2-detector reactor experiment

- Compare event rates for "true" oscillation parameters $\vec{\Theta}$ and no systematical errors vs. event rates for fit parameters $\vec{\Theta}'$
- Contributions from near detector (N) and far detector (F)
- Contributions from all bins i
- Completely uncorrelated "bin-to-bin" errors σ_{bin} to parameterize background subtraction errors

χ^2 analysis for a 2-detector reactor experiment

- Compare event rates for "true" oscillation parameters $\vec{\Theta}$ and no systematical errors vs. event rates for fit parameters $\vec{\Theta}'$
- Contributions from near detector (N) and far detector (F)
- Contributions from all bins i
- Completely uncorrelated "bin-to-bin" errors σ_{bin} to parameterize background subtraction errors

$$\chi^2 = \sum_{A=N,F} \sum_i \frac{\left[(1 + a_{\text{corr}} + a^A_{\text{uncorr}} + a^i_{\text{spect}} + a^{A,i}_{\text{bin}}) N^{A,i}(\vec{\Theta}') - N^{A,i}(\vec{\Theta})\right]^2}{N^{A,i}} + \frac{a^2_{\text{corr}}}{\sigma^2_{\text{corr}}} + \sum_{A=N,F} \frac{(a^A_{\text{uncorr}})^2}{\sigma^2_{\text{uncorr}}} + \sum_i \frac{(a^i_{\text{spect}})^2}{\sigma^2_{\text{spect}}} + \sum_{A=N,F} \sum_i \frac{(a^{A,i}_{\text{bin}})^2}{\sigma^2_{\text{bin}}}$$

χ^2 analysis for a 2-detector reactor experiment

- Compare event rates for "true" oscillation parameters $\vec{\Theta}$ and no systematical errors vs. event rates for fit parameters $\vec{\Theta}'$
- Contributions from near detector (N) and far detector (F)
- Contributions from all bins i
- Completely uncorrelated "bin-to-bin" errors σ_{bin} to parameterize background subtraction errors

$$\chi^2 = \sum_{A=N,F} \sum_i \frac{[(1 + a_{\text{corr}} + a^A_{\text{uncorr}} + a^i_{\text{spect}})N^{A,i}(\vec{\Theta}') - N^{A,i}(\vec{\Theta})]^2}{N^{A,i} + (N^{A,i}\sigma_{\text{bin}})^2} + \frac{a^2_{\text{corr}}}{\sigma^2_{\text{corr}}} + \sum_{A=N,F} \frac{(a^A_{\text{uncorr}})^2}{\sigma^2_{\text{uncorr}}} + \sum_i \frac{(a^i_{\text{spect}})^2}{\sigma^2_{\text{spect}}}$$

Outline

1. Reactor neutrino experiments
   - Important systematical error contributions
   - χ^2 analysis inlcuding systematical errors
2. Systematical errors in GLoBES
   - Internal χ^2 functions
   - User-defined χ^2 functions
   - Example: Implementation of Double Chooz
   - Performance considerations
3. Some physics results

GLoBES analysis procedure

- Minimize $\chi^2 \equiv \chi^2(\theta_{12}, \theta_{13}, \theta_{23}, \delta_{\mathrm{CP}}, \Delta m^2_{21}, \Delta m^2_{31}, \vec{a}_{\mathrm{syst}})$

GLoBES analysis procedure

- Minimize $\chi^2 \equiv \chi^2(\theta_{12}, \theta_{13}, \theta_{23}, \delta_{\mathrm{CP}}, \Delta m^2_{21}, \Delta m^2_{31}, \vec{a}_{\mathrm{syst}})$
- χ^2 contains pull terms of the form a_j^2/σ_j^2 to forbid a_j values too far from zero.

GLoBES analysis procedure

- Minimize $\chi^2 \equiv \chi^2(\theta_{12}, \theta_{13}, \theta_{23}, \delta_{\mathrm{CP}}, \Delta m^2_{21}, \Delta m^2_{31}, \vec{a}_{\mathrm{syst}})$
- χ^2 contains pull terms of the form a_j^2/σ_j^2 to forbid a_j values too far from zero.
- Two χ^2 functions for each rule: `@sys_on_function` and `@sys_off_function`

GLoBES analysis procedure

- Minimize $\chi^2 \equiv \chi^2(\theta_{12}, \theta_{13}, \theta_{23}, \delta_{\mathrm{CP}}, \Delta m_{21}^2, \Delta m_{31}^2, \vec{a}_{\mathrm{syst}})$
- χ^2 contains pull terms of the form a_j^2/σ_j^2 to forbid a_j values too far from zero.
- Two χ^2 functions for each rule: `@sys_on_function` and `@sys_off_function`
- Switch between them with `glbSwitchSystematics(...)`

GLoBES analysis procedure

- Minimize $\chi^2 \equiv \chi^2(\theta_{12}, \theta_{13}, \theta_{23}, \delta_{\text{CP}}, \Delta m^2_{21}, \Delta m^2_{31}, \vec{a}_{\text{syst}})$
- χ^2 contains pull terms of the form a_j^2/σ_j^2 to forbid a_j values too far from zero.
- Two χ^2 functions for each rule: `@sys_on_function` and `@sys_off_function`
- Switch between them with `glbSwitchSystematics(...)`
- Define the σ_j with `@sys_on_errors` and `@sys_off_errors`

GLoBES analysis procedure

- Minimize $\chi^2 \equiv \chi^2(\theta_{12}, \theta_{13}, \theta_{23}, \delta_{\mathrm{CP}}, \Delta m^2_{21}, \Delta m^2_{31}, \vec{a}_{\mathrm{syst}})$
- χ^2 contains pull terms of the form a_j^2/σ_j^2 to forbid a_j values too far from zero.
- Two χ^2 functions for each rule: `@sys_on_function` and `@sys_off_function`
- Switch between them with `glbSwitchSystematics(...)`
- Define the σ_j with `@sys_on_errors` and `@sys_off_errors`
- Old directives `@signalerror` and `@backgrounderror` are still valid for the built-in χ^2 functions.

Built-in χ^2 functions

χ^2 function	Analysis	Systematical errors	`@errordim` (deprecated)
`chiSpectrumOnly`	Spectral	signal normalization	7
`chiNoSysSpectrum`	Spectral	—	2
`chiSpectrumTilt`	Spectral	signal/bckgnd. normalization and spectral “tilt”	0
`chiSpectrumCalib`	Spectral	signal/bckgnd. normalization and energy calibration error	9
`chiTotalRatesTilt`	Total rates	signal/bckgnd. normalization and spectral “tilt”	4
`chiNoSysTotalRates`	Total rates	—	8
`chiZero`	$\equiv 0$	—	—

User-defined χ^2 functions

- New in GLoBES 3.0: User-defined treatment of systematical errors:

```
int glbDefineChiFunction(glb_chi_function
      chi_func, int dim, const char *name,
      void *user_data)
```

Registers an arbitrary function `chi_func` *under a given* `name`*. The number of systematics parameters a_j for* `chi_func` *is* `dim`*.*

User-defined χ^2 functions

- New in GLoBES 3.0: User-defined treatment of systematical errors:

```
int glbDefineChiFunction(glb_chi_function
      chi_func, int dim, const char *name,
      void *user_data)
```

Registers an arbitrary function `chi_func` *under a given* `name`*. The number of systematics parameters* a_j *for* `chi_func` *is* `dim`*.*

- `chi_func` can be used in subsequently loaded AEDL files.

User-defined χ^2 functions

- New in GLoBES 3.0: User-defined treatment of systematical errors:

```
int glbDefineChiFunction(glb_chi_function
        chi_func, int dim, const char *name,
        void *user_data)
```

Registers an arbitrary function `chi_func` *under a given* `name`*. The number of systematics parameters* a_j *for* `chi_func` *is* `dim`*.*

- `chi_func` can be used in subsequently loaded AEDL files.
- Format for user-defined χ^2 functions:

```
double my_chi_function(int exp, int rule,
   int n_params, double *params, double *errors,
   void *user_data)
```

Implementation of Double Chooz: χ^2 function

```
#define EXP_FAR  0
#define EXP_NEAR 1
double chiDC(int exp, int rule, int n_params, double *x, double *errors
             void *user_data)
{
  int n_bins = glbGetNumberOfBins(EXP_FAR);
  double *true_rates_N = glbGetRuleRatePtr(EXP_NEAR, 0);
  double *true_rates_F = glbGetRuleRatePtr(EXP_FAR, 0);
  double signal_fit_rates_N[n_bins], signal_fit_rates_F[n_bins];
  int emin, emax, ew_low, ew_high;
  double fit_rate;
  double chi2 = 0.0;
  glbGetEminEmax(exp,&emin,&emax); // Simulated energy interval
  glbGetEnergyWindowBins(exp,rule,&ew_low,&ew_high); // Analysis cuts

  // Apply energy calibration errors for FD (x[3]) and ND (x[4])
  glbShiftEnergyScale(x[3], glbGetSignalFitRatePtr(EXP_FAR,0),
                              signal_fit_rates_F,n_bins,emin,emax);
  glbShiftEnergyScale(x[4], glbGetSignalFitRatePtr(EXP_NEAR, 0),
                              signal_fit_rates_N,n_bins,emin,emax);
  ⋮
```

Implementation of Double Chooz: χ^2 function (cntd.)

```
  ⋮
  // Systematical normalization bias:
  //   x[0]:        Reactor flux error (correlated)
  //   x[1], x[2]:  Detector normalization errors (uncorrelated)
  for (int i=ew_low; i <= ew_high; i++)
  {
    fit_rate = (1.0 + x[0] + x[1]) * signal_fit_rates_F[i];
    chi2    += SQR(true_rates_F[i] - fit_rate) / true_rates_F[i];

    fit_rate = (1.0 + x[0] + x[2]) * signal_fit_rates_N[i];
    chi2    += SQR(true_rates_N[i] - fit_rate) / true_rates_N[i];
  }

  for (int i=0; i < n_params; i++)
    chi2 += SQR(x[i] / errors[i]);  // Add pull terms
  return chi2;
}
```

Implementation of Double Chooz: AEDL files

D-Chooz_far.glb

```
rule(#rule0)<
  @signal = 1.0@#nu_e_disappearance_CC
  @background = 0.0@#nu_e_disappearance_CC // No background
  @energy_window = 0.0015 :  0.01
  @sys_off_function = "chiNoSysSpectrum"
  @sys_off_errors   = {}
  @sys_on_function  = "chiDC" // Handles chi^2 for both detectors
  @sys_on_errors    = { 0.02,  0.006,   0.006,    0.005,     0.005   }
>                   /*{ Flux, Norm FD, Norm ND, Energy FD, Energy ND }*/
```

Implementation of Double Chooz: AEDL files

D-Chooz_far.glb

```
rule(#rule0)<
  @signal = 1.0@#nu_e_disappearance_CC
  @background = 0.0@#nu_e_disappearance_CC // No background
  @energy_window = 0.0015 :  0.01
  @sys_off_function = "chiNoSysSpectrum"
  @sys_off_errors   = {}
  @sys_on_function  = "chiDC" // Handles chi^2 for both detectors
  @sys_on_errors    = { 0.02,  0.006,   0.006,    0.005,     0.005   }
>                   /*{ Flux, Norm FD, Norm ND, Energy FD, Energy ND }*/
```

D-Chooz_near.glb

```
rule(#rule0)<
  @signal = 1.0@#nu_e_disappearance_CC
  @background = 0.0@#nu_e_disappearance_CC // No background
  @energy_window = 0.0015 :  0.01
  @sys_off_function = "chiNoSysSpectrum"
  @sys_off_errors   = {}
  @sys_on_function  = "chiZero" // Dummy function
  @sys_on_errors    = {}
>
```

Implementation of Double Chooz: Main program

```
int main(int argc, char *argv[])
{
  ⋮
  glbInit(argv[0]);
  glbDefineChiFunction(&chiDC, 5, "chiDC", NULL);
  glbInitExperiment("dchooz-far.glb",
         &glb_experiment_list[0], &glb_num_of_exps);
  glbInitExperiment("dchooz-near.glb",
         &glb_experiment_list[0], &glb_num_of_exps);
  ⋮
  chi2 = glbChiSys(test_values, GLB_ALL, GLB_ALL);
  ⋮
}
```

Inclusion of spectral and bin-to-bin errors

Function `chiDCSpectral`

```
⋮
for (int i=ew_low; i <= ew_high; i++) {
  fit_rate  = (1.0 + x[0] + x[1] + x[5+i]) * signal_fit_rates_F[i];
  sqr_sigma = true_rates_F[i] * (1.0 + true_rates_F[i]*SQR(sigma_bin))
  chi2     += SQR(true_rates_F[i] - fit_rate) / sqr_sigma;

  fit_rate  = (1.0 + x[0] + x[2] + x[5+i]) * signal_fit_rates_N[i];
  sqr_sigma = true_rates_N[i] * (1.0 + true_rates_N[i]*SQR(sigma_bin))
  chi2     += SQR(true_rates_N[i] - fit_rate) / sqr_sigma;
}
⋮
```

Inclusion of spectral and bin-to-bin errors (cntd.)

Main program

```
⋮
glbDefineChiFunction(&chiDCSpectral, 5+n_bins, "chiDCSpectral", NULL);
⋮
sys_errors[0] = 0.02;                // Flux normalization
sys_errors[1] = 0.006;               // Far detector normalization
sys_errors[2] = 0.006;               // Near detector normalization
sys_errors[3] = 0.005;               // Far detector energy calibration
sys_errors[4] = 0.005;               // Near detector energy calibration
for (i=5; i < 5+n_bins; i++)         // Spectral error
  sys_errors[i] = 0.02;
sigma_bin = 0.005;                   // Bin-to-bin error
glbSetChiFunction(EXP_FAR, 0, GLB_ON, "chiDCSpectral", sys_errors);
⋮
chi2 = glbChiSys(test_values, GLB_ALL, GLB_ALL);
⋮
```

Performance considerations

Important performance bottlenecks in GLoBES

- Calculation of three-flavour oscillation probabilities

Performance considerations

Important performance bottlenecks in GLoBES

- Calculation of three-flavour oscillation probabilities
- Evaluation of χ^2

Performance considerations

Important performance bottlenecks in GLoBES

- Calculation of three-flavour oscillation probabilities
- Evaluation of χ^2

$\Rightarrow$ Use minimization algorithm which keeps the number of required probability and χ^2 re-computations as low as possible (alternative, experimental minimizer available in GLoBES 3.0)

Performance considerations

Important performance bottlenecks in GLoBES

- Calculation of three-flavour oscillation probabilities
- Evaluation of χ^2

$\Rightarrow$ Use minimization algorithm which keeps the number of required probability and χ^2 re-computations as low as possible (alternative, experimental minimizer available in GLoBES 3.0)

$\Rightarrow$ Use specialized algorithm for the calculation of oscillation probabilities (new in GLoBES 3.0) JK, physics/0610206

Performance considerations

Important performance bottlenecks in GLoBES

- Calculation of three-flavour oscillation probabilities
- Evaluation of χ^2

⇒ Use minimization algorithm which keeps the number of required probability and χ^2 re-computations as low as possible (alternative, experimental minimizer available in GLoBES 3.0)

⇒ Use specialized algorithm for the calculation of oscillation probabilities (new in GLoBES 3.0)   JK, physics/0610206

⇒ Write efficient χ^2 functions

Performance considerations

Important performance bottlenecks in GLoBES

- Calculation of three-flavour oscillation probabilities
- Evaluation of χ^2

$\Rightarrow$ Use minimization algorithm which keeps the number of required probability and χ^2 re-computations as low as possible (alternative, experimental minimizer available in GLoBES 3.0)

$\Rightarrow$ Use specialized algorithm for the calculation of oscillation probabilities (new in GLoBES 3.0) JK, physics/0610206

$\Rightarrow$ Write efficient χ^2 functions

- Use analytical simplifications as far as possible

Performance considerations

Important performance bottlenecks in GLoBES

- Calculation of three-flavour oscillation probabilities
- Evaluation of χ^2

⇒ Use minimization algorithm which keeps the number of required probability and χ^2 re-computations as low as possible (alternative, experimental minimizer available in GLoBES 3.0)

⇒ Use specialized algorithm for the calculation of oscillation probabilities (new in GLoBES 3.0) JK, physics/0610206

⇒ Write efficient χ^2 functions

- Use analytical simplifications as far as possible
- Avoid unnecessary code repetition

Performance considerations

Important performance bottlenecks in GLoBES

- Calculation of three-flavour oscillation probabilities
- Evaluation of χ^2

⇒ Use minimization algorithm which keeps the number of required probability and χ^2 re-computations as low as possible (alternative, experimental minimizer available in GLoBES 3.0)

⇒ Use specialized algorithm for the calculation of oscillation probabilities (new in GLoBES 3.0) JK, physics/0610206

⇒ Write efficient χ^2 functions

- Use analytical simplifications as far as possible
- Avoid unnecessary code repetition
- Avoid numerically expensive expressions ($\sin$, $\cos$, $\exp$, $\ldots$)

Performance considerations

Important performance bottlenecks in GLoBES

- Calculation of three-flavour oscillation probabilities
- Evaluation of χ^2

$\Rightarrow$ Use minimization algorithm which keeps the number of required probability and χ^2 re-computations as low as possible (alternative, experimental minimizer available in GLoBES 3.0)

$\Rightarrow$ Use specialized algorithm for the calculation of oscillation probabilities (new in GLoBES 3.0) JK, physics/0610206

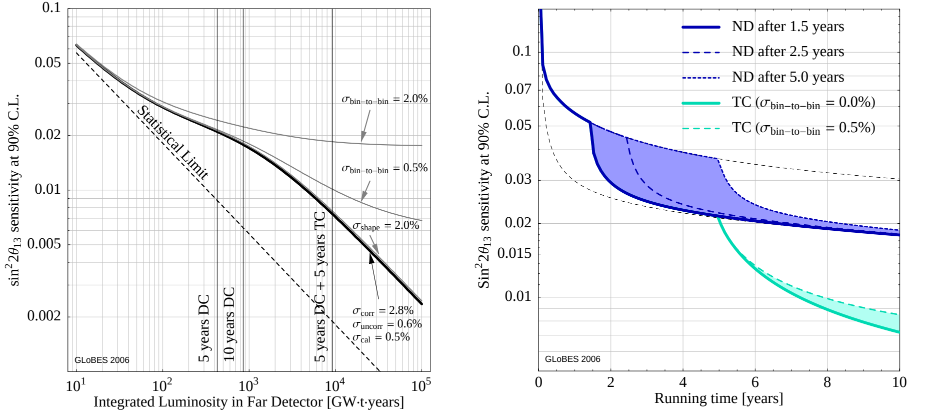
$\Rightarrow$ Write efficient χ^2 functions

- Use analytical simplifications as far as possible
- Avoid unnecessary code repetition
- Avoid numerically expensive expressions (sin, cos, exp, ...)
- Avoid conditional branches

Outline

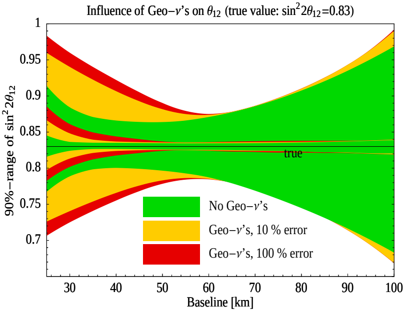
1. Reactor neutrino experiments
   - Important systematical error contributions
   - χ^2 analysis inlcuding systematical errors
2. Systematical errors in GLoBES
   - Internal χ^2 functions
   - User-defined χ^2 functions
   - Example: Implementation of Double Chooz
   - Performance considerations
3. Some physics results

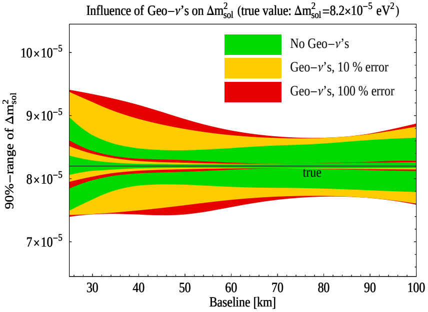
θ_{13} sensitivity of next-generation reactor experiments



P. Huber, JK, M. Lindner, M. Rolinec, W. Winter, hep-ph/0601266

Measurement of θ_{12} and Δm^2_{21} in a reactor experiment





JK, M. Lindner, A. Merle, M. Rolinec, hep-ph/0606151

Conclusions

- Systematical errors are crucial for reactor experiments

Conclusions

- Systematical errors are crucial for reactor experiments
- Correlated errors cancel in a near/far setup

Conclusions

- Systematical errors are crucial for reactor experiments
- Correlated errors cancel in a near/far setup
- Careful design of χ^2 expression is mandatory

Conclusions

- Systematical errors are crucial for reactor experiments
- Correlated errors cancel in a near/far setup
- Careful design of χ^2 expression is mandatory
- Several simple χ^2 functions are already implemented in GLoBES

Conclusions

- Systematical errors are crucial for reactor experiments
- Correlated errors cancel in a near/far setup
- Careful design of χ^2 expression is mandatory
- Several simple χ^2 functions are already implemented in GLoBES
- New in version 3.0: Flexible interface for user-defined χ^2

Conclusions

- Systematical errors are crucial for reactor experiments
- Correlated errors cancel in a near/far setup
- Careful design of χ^2 expression is mandatory
- Several simple χ^2 functions are already implemented in GLoBES
- New in version 3.0: Flexible interface for user-defined χ^2
- Keep performance in mind when implementing your own χ^2