Numerical Software III

GNU Scientific Library
Nonlinear Modern Family Calibration

Viktor Mosenkis

Informatik 12:
Software and Tools for Computational Engineering (STCE)

RWTH Aachen University
Contents

Objectives and Learning Outcomes

GNU Scientific Library (GSL)
   Overview
   Compiling and Linking

Multidimensional Optimization with GSL

Nonlinear Least-Squares Fitting with GSL

Summary and Next Steps
Objectives and Learning Outcomes

GNU Scientific Library (GSL)
  Overview
  Compiling and Linking

Multidimensional Optimization with GSL

Nonlinear Least-Squares Fitting with GSL

Summary and Next Steps
Objective

▶ Learn how to use GNU Scientific Library (GSL),
▶ Solve nonlinear Modern Family problem with multidimensional optimization algorithm,
▶ Solve nonlinear Modern Family problem with nonlinear least-squares fitting algorithm

Learning Outcomes

▶ You will understand
  ▶ difference between multidimensional optimization and nonlinear least-squares fitting,
  ▶ their advantages and disadvantages,
  ▶ their usage with GSL

▶ You will be able to use GSL to solve Modern Family problem with
  ▶ multidimensional optimization algorithm,
  ▶ nonlinear least-squares fitting algorithm.
Outline

Objectives and Learning Outcomes

GNU Scientific Library (GSL)
  Overview
  Compiling and Linking

Multidimensional Optimization with GSL

Nonlinear Least-Squares Fitting with GSL

Summary and Next Steps
The **GNU Scientific Library (GSL)** is a collection of routines for numerical computing for C and C++ programmers. GSL is free software under the GNU General Public License.

The library provides routines for a wide range of topics in numerical computing such as:

- Random Numbers
- Quadrature
- Root-Finding
- Least-Squares Fitting
- Minimization
- Linear Algebra
- Sparse Linear Algebra
GSL can be downloaded from
  ▶️ https://www.gnu.org/software/gsl/

The documentation can be obtained

Precompiled binary packages are included in most GNU/Linux distributions

A compiled version of GSL is available as part of Cygwin on Windows
The following program demonstrates the use of library to compute the value of the Bessel function for $x = 5$.

```c
#include <stdio.h>
#include <gsl/gsl_sf_bessel.h>

int main (void)
{
    double x = 5.0;
    double y = gsl_sf_bessel_J0 (x);
    printf ("J0(%g) = %.18e\n", x, y);
    return 0;
}
```
To compile a source file main.cpp you must tell the compiler the location of gsl directory containing the GSL header files. The default location of the gsl directory is /usr/local/include. Thus a typical compilation command with g++ is:

```
$ g++ -Wall -l/usr/local/include -c main.cpp
```

The library is installed as a single static library libgsl.a. A shared version of the library libgsl.so is also installed on systems that support shared libraries. The default location of these files is /usr/local/lib. If this directory is not on the search path of your linker you have to provide its location as a command line flag -L/usr/local/lib. To link against the library you also need to specify a supporting CBLAS library (basic linear algebra subroutines). GSL provides a suitable CBLAS implementation libgslcblas.a (static) libgslcblas.so (shared). On some machines you must use the option -lm to link against the system math library. Hence the following command links the application with the library:

```
$ g++ -L/usr/local/lib main.o -lgsl -lgslcblas -lm
```
Outline

Objectives and Learning Outcomes

GNU Scientific Library (GSL)
  Overview
  Compiling and Linking

Multidimensional Optimization with GSL

Nonlinear Least-Squares Fitting with GSL

Summary and Next Steps
We state the Modern Family example for model

\[ y = f(p, x) : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R} \]

as minimization of the error function

\[ E(p, X, y) = \sum_{i=0}^{m-1} (f(p, x_i^T) - y_i)^2 \]

In this module only nonlinear (in \( p \))

\[ y = f(p, x) = (p^T \cdot x)^2 \]

model is considered.
Multidimensional Optimization

Model

GSL provides several multidimensional minimization methods. All of them aim to solve the following problem.
Consider

\[ y = f(x) : \mathbb{R}^n \to \mathbb{R} \ . \]

Find \( x_0 \in \mathbb{R}^n \) such that

\[ f(x_0) \leq f(x) \]

for all \( x \) in the neighborhood of \( x_0 \). Hence these methods allow to solve our Modern Family problem. By reformulating the error function as

\[ f(p) = E(p, X, y) = \sum_{i=0}^{m-1} (f(p, x_i^T) - y_i)^2 \]
Multidimensional Optimization

Initialising the Multidimensional Minimizer

The following function initializes a multidimensional minimizer. The minimizer itself depends only on the dimension of the problem and the algorithm and can be reused for different problems.

```c
// Workspace for minimizing functions using derivatives
#include <gsl/multimin.h>

// This function returns a pointer to a newly allocated instance of a minimizer of type T for an n−dimension function.
#include <gsl/multimin.h>
gsl_multimin_fdfminimizer* gsl_multimin_fdfminimizer_alloc(const gsl_multimin_fdfminimizer_type *T, size_t n)

// Initializes the minimizer s to minimize the function fdf starting from the initial point x.
// The size of the first trial step is given by step_size. The accuracy of the line minimization
// is specified by tol
int gsl_multimin_fdfminimizer_set(gsl_multimin_fdfminimizer *s, gsl_multimin_function_fdf *fdf,
                                   const gsl_vector *x, double step_size, double tol)

// Frees all the memory associated with the minimizer s
void gsl_multimin_fdfminimizer_free(gsl_multimin_fdfminimizer *s)

// This type specifies a minimization algorithm using gradients.
gsl_multimin_fdfminimizer_type
```
Multidimensional Optimization

Providing a function to minimize

The minimizer with derivatives requires the user to provide a function that calculates the value of $f(x)$ and a function that calculates its gradient. These function must be defined by the following data type:

```c
//Defines a general function of n variables with parameters and the corresponding gradient
gsl_multimin_function_fdf

//Should return the result f(x,params) for argument x and parameters params.
double (* f) (const gsl_vector * x, void * params)

//Should the gradient of f(x,params) for argument x and parameters params.
void (* df) (const gsl_vector * x, void * params, gsl_vector * g)

//The number of components of vector x
size_t n

//A pointer to the parameters of the function
void * params
```
Multidimensional Optimization

Iteration and Stopping Criteria

**Iteration**: The following function performs one iteration to update the state of the minimizer.

```c
// Perform a single iteration of the minimizer s
int gsl_multimin_fdfminimizer_iterate(gsl_multimin_fdfminimizer * s)
```

The minimizer maintains a current best estimate of the minimum at all times. This information can be accessed with auxiliary functions. E.g.

```c
// Return the gradient at the current best estimate
gsl_vector * gsl_multimin_fdfminimizer_gradient(const gsl_multimin_fdfminimizer * s)
```

**Stopping Criteria**: A minimization procedure should stop when one of the following conditions is true:

- A minimum has been found to within the user-specified precision
- A user-specified maximum number of iterations has been reached
- An error has occurred

These conditions are under control of the user. Several functions are available to test the precision of the current result. E.g.

```c
// Tests the norm of the gradient g against the absolute tolerance epsabs
int gsl_multimin_test_gradient(const gsl_vector * g, double epsabs)
```
Outline

Objectives and Learning Outcomes

GNU Scientific Library (GSL)
   Overview
   Compiling and Linking

Multidimensional Optimization with GSL

Nonlinear Least-Squares Fitting with GSL

Summary and Next Steps
Nonlinear Least-Squares Fitting

Model

The nonlinear least-squares fitting aims to minimize the squared residuals of \( m \) functions \( f_i : \mathbb{R}^n \rightarrow \mathbb{R} \),

\[
F(x) = \frac{1}{2} \| f(x) \|^2 = \frac{1}{2} \sum_{i=1}^{m} f_i(x)^2
\]

where \( x \in \mathbb{R}^n \).

Hence this method also allows to solve our nonlinear Modern Family problem, by setting

\[
f_i(x) = f(p, x_i^T) - y_i = (p^T \cdot x_i^T)^2
\]
Nonlinear Least-Squares Fitting
Initializing the Solver

// This structure specifies the type of algorithm which will be used to solve a nonlinear least squares problem.
gsl_multifit_nlinear_type

// Return a pointer to a newly allocated instance of a derivative solver of type T for n observations and p parameters.
gsl_multifit_nlinear_workspace * gsl_multifit_nlinear_alloc(const gsl_multifit_nlinear_type * T, 
const gsl_multifit_nlinear_parameters * params, const size_t n, const size_t p)

// Return a set of recommended default parameters for use in solving nonlinear least squares problems.
gsl_multifit_nlinear_parameters gsl_multifit_nlinear_default_parameters(void)

// Initialize an existing workspace w to use the system fdf and the initial guess x.
int gsl_multifit_nlinear_init(const gsl_vector * x, gsl_multifit_nlinear_fdf * fdf, 
gsl_multifit_nlinear_workspace * w)

// Free all the memory associated with the workspace w.
void gsl_multifit_nlinear_free(gsl_multifit_nlinear_workspace * w)

// Return a pointer to the name of the solver
const char * gsl_multifit_nlinear_name(const gsl_multifit_nlinear_workspace * w)
const char * gsl_multifit_nlinear_trs_name(const gsl_multifit_nlinear_workspace * w)
Nonlinear Least-Squares Fitting

Providing a function to minimize

The minimizer with derivatives requires the user to provide \( n \) functions that calculate \( f_i(x) \). These functions must be defined by the following data type:

```c
// Defines a general system of functions with arbitrary parameters, the corresponding
// Jacobian matrix of derivatives.

int (* f) (const gsl_vector * x, void * params, gsl_vector * f)
```

```c
// Should store the n components of the vector f(x) in f for argument x and arbitrary
// parameters params

int (* df) (const gsl_vector * x, void * params, gsl_matrix * J)
```

```c
// The number of components of vector f

size_t n
```

```c
// The number of components of vector x

size_t p
```

```c
// A pointer to the parameters of the function

void * params
```
Nonlinear Least-Squares Fitting

High Level Driver

The following routine provide a high level wrapper that combines the iteration and convergence testing for easy use.

```c
// iterate the nonlinear least squares solver w for a maximum of maxiter iterations. After each iteration, the system is tested for convergence with the error tolerances xtol, gtol and ftol. Additionally, the user may supply a callback function callback which is called after each iteration, so that the user may save or print relevant quantities for each iteration. The parameter callback_params is passed to the callback function

int gsl_multifit_nlinear_driver(const size_t maxiter, const double xtol, const double gtol, const double ftol, void (*callback)(const size_t iter, void *params, const gsl_multifit_linear_workspace *w, void *callback_params, int *info, gsl_multifit_nlinear_workspace *w)
```

The solver workspace contains information which can be used to track the progress/result of the solution. This information can be accessed via auxiliary functions. E.g.

```c
//Return the current position x (i.e. best-fit parameters) of the solver w

gsl_vector * gsl_multilarge_nlinear_position(const gsl_multilarge_nlinear_workspace * w)
```

```c
//Return the current residual vector f(x) of the solver w

gsl_vector * gsl_multifit_nlinear_residual(const gsl_multifit_nlinear_workspace * w)
```
Outline

Objectives and Learning Outcomes

GNU Scientific Library (GSL)
   Overview
   Compiling and Linking

Multidimensional Optimization with GSL

Nonlinear Least-Squares Fitting with GSL

Summary and Next Steps
Summary

▶ Solve the nonlinear Modern Family problem with
   ▶ multidimensional minimization method from GSL
   ▶ nonlinear least-squares fitting method from GSL

Next Steps

▶ Play with sample code.
▶ Implement the derivative function for both methods using symbolic differentiation and AD. Play around with different parameters and compare the results. Use iteration driver in the nonlinear least-squares fitting.
▶ Continue the course to find out more ...