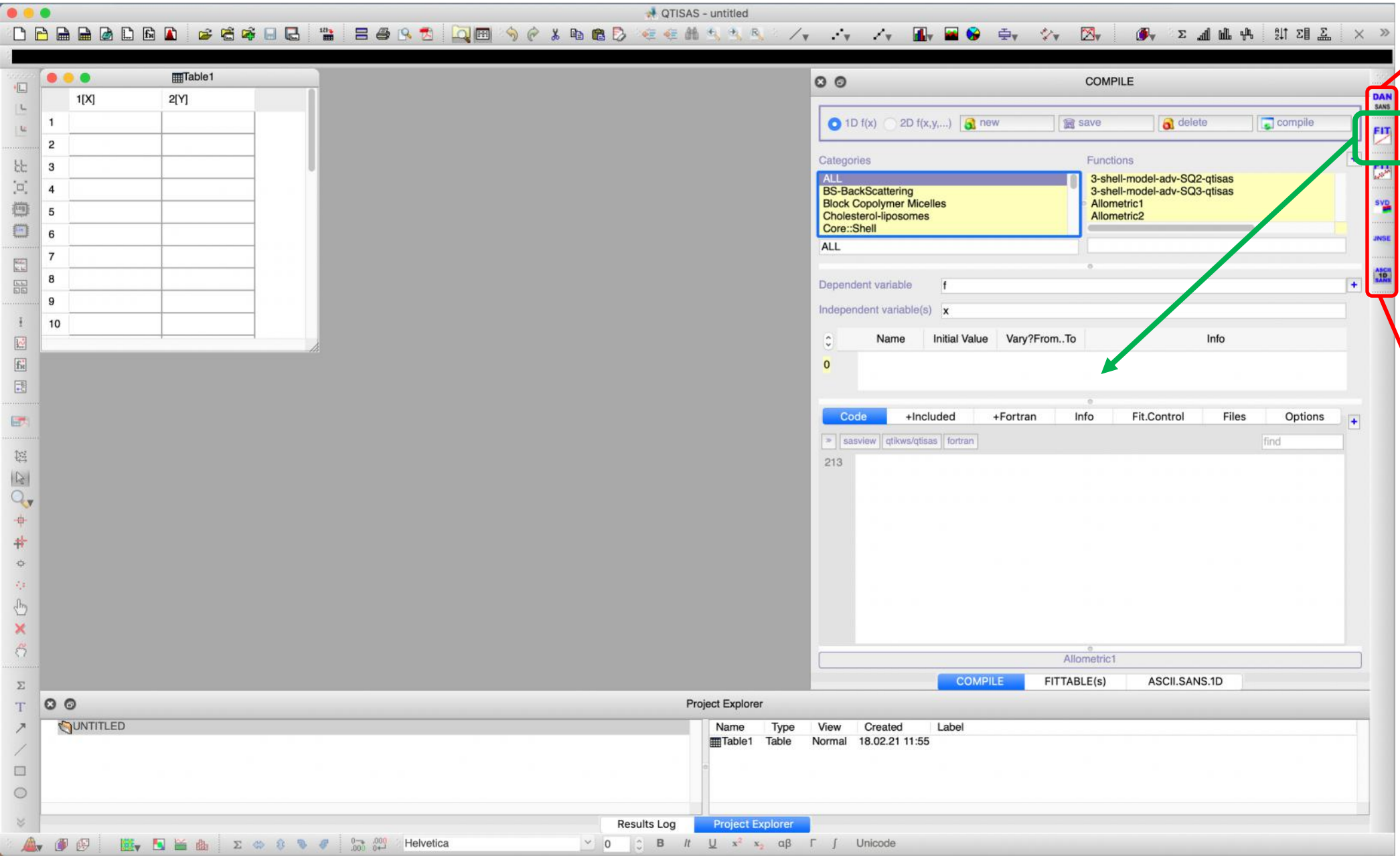
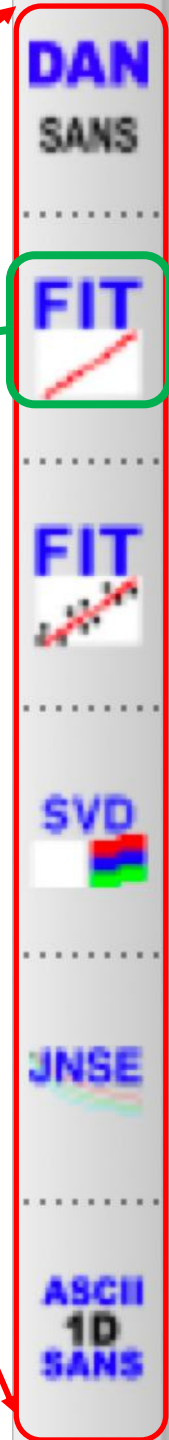
Fit-Compile : Introduction . Screenshots . Example

Fit-Compile :: GUI to compile fitting functions

Fit-Compile :: how to open





Main Elements

1. Button Group

2. Function Explorer

2.1 Category List
2.2 Function List (in Active Category)
2.3 Active Category Line
2.4 Active Function Line

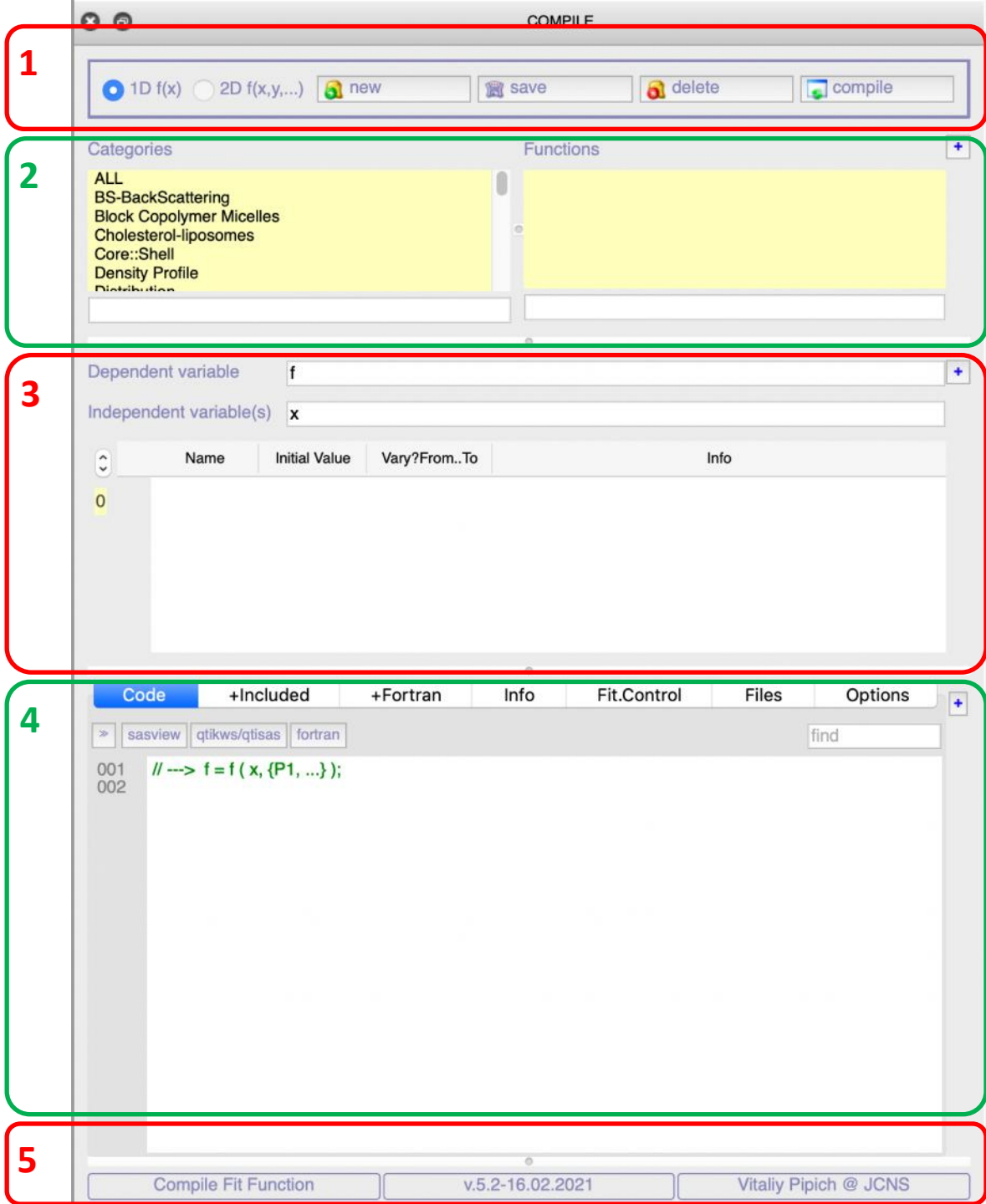
3. Browser of Parameters

3.1 Dependent Variable
3.2 Independent Variable
3.3 Number-of-Parameter Spin-Box
3.4 Table-of-Parameters

4. Tab of Tools

4.1 „Code“: C++ Function Editor (Main Function)
4.2 „+Include“: C++ Function Editor (Included Functions)
4.3 „+Fortran“: Fortran Function Editor (Included Functions)
4.4 „Info“: Information Editor (Current Function)
4.5 “Fit.Control”: Fitting Options (Current Function)
4.6 “Files”: Generated Files Viewer (Current Function)
4.7 “Options”: Fit-Compile Interface Options

5. Info Panel: Name – Version – Author

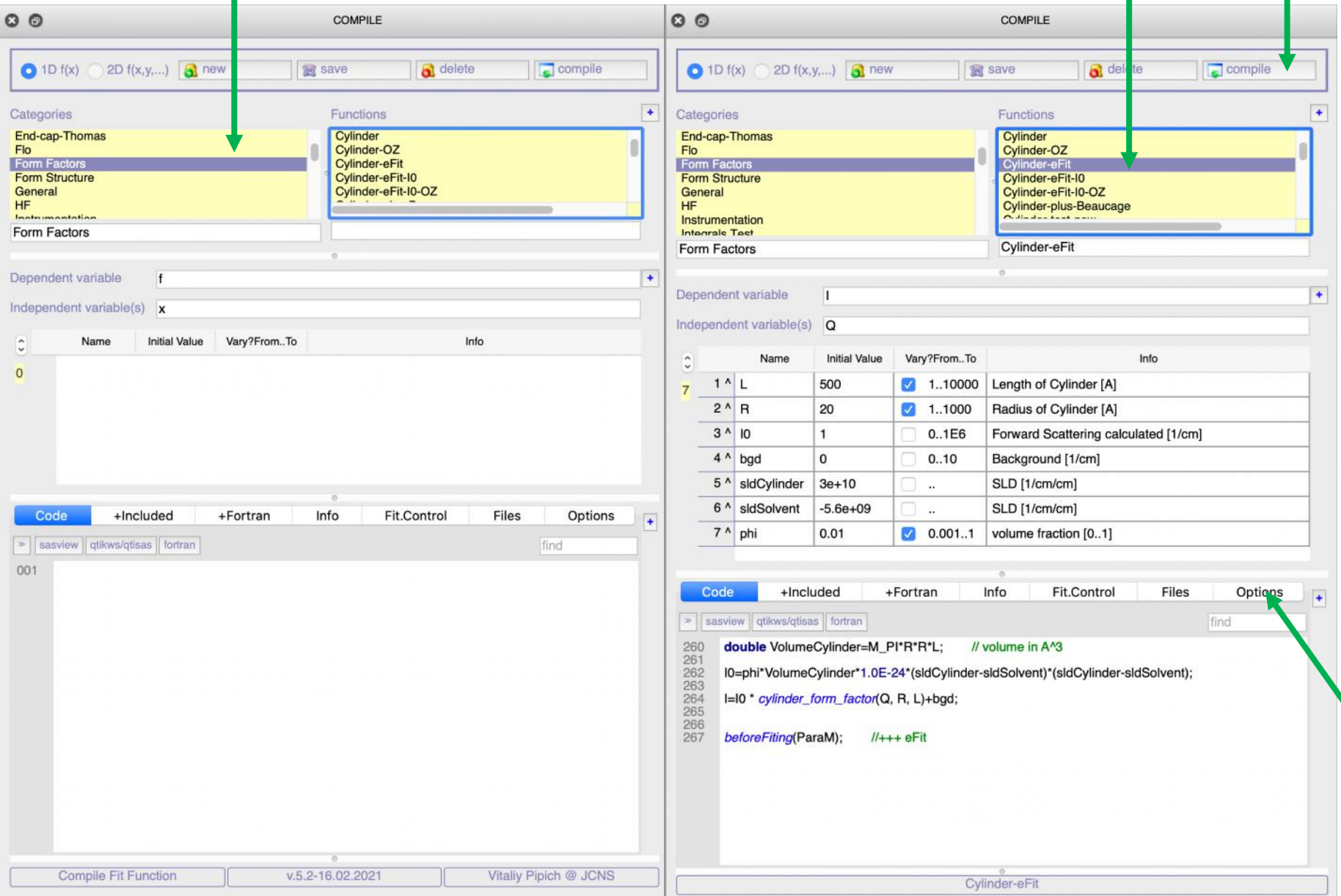


2. Function Explorer: Open & Compile an Existing Function

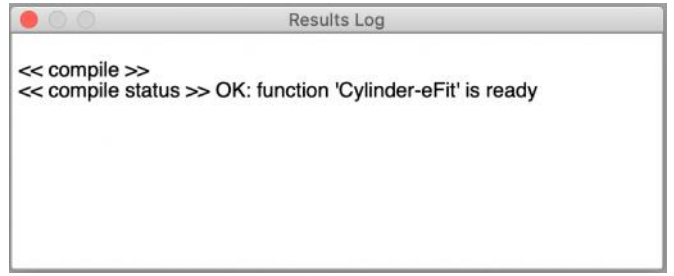
1. Category Selection

2. Function Selection

3. Compile Current Function



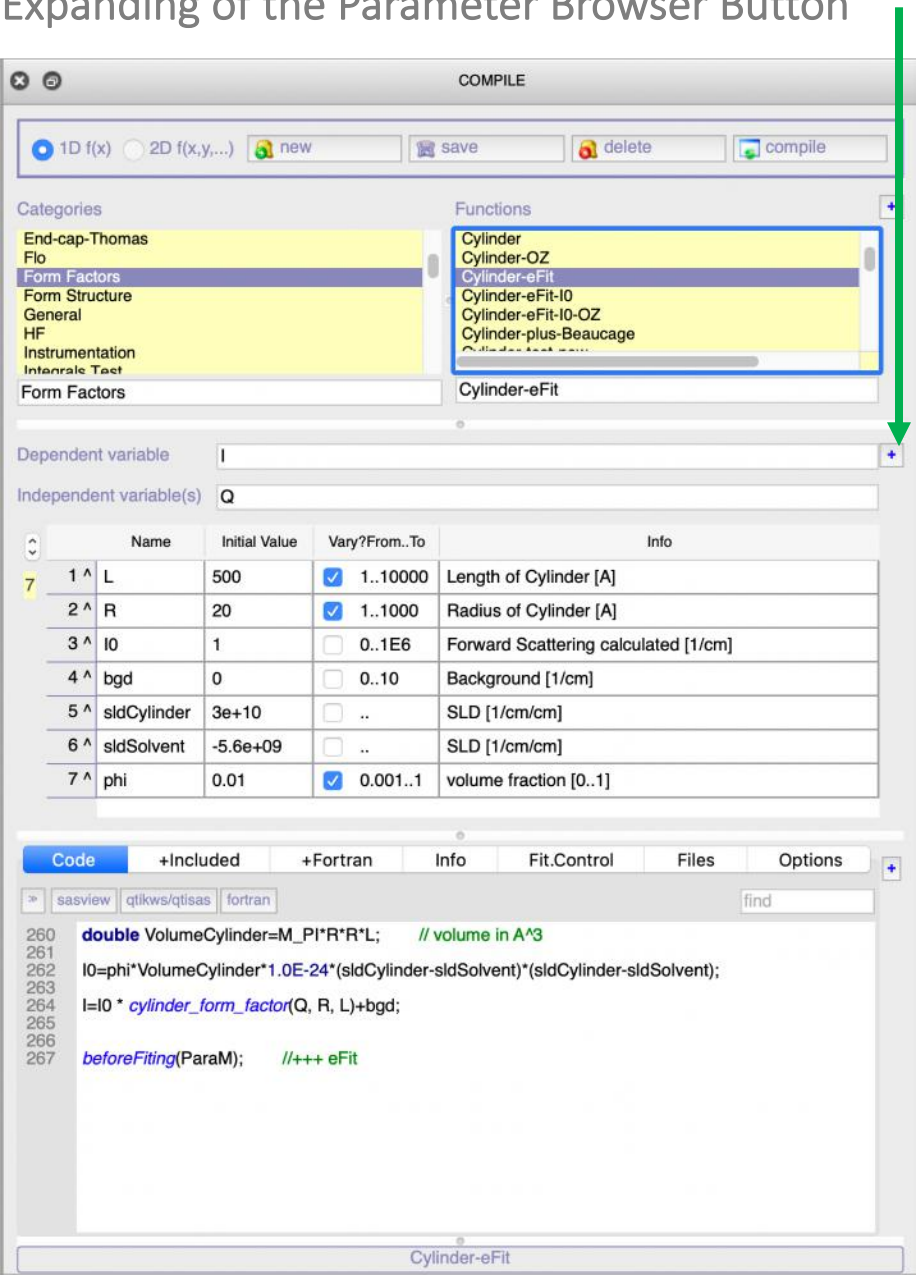
4. Results Log :
check compilation status

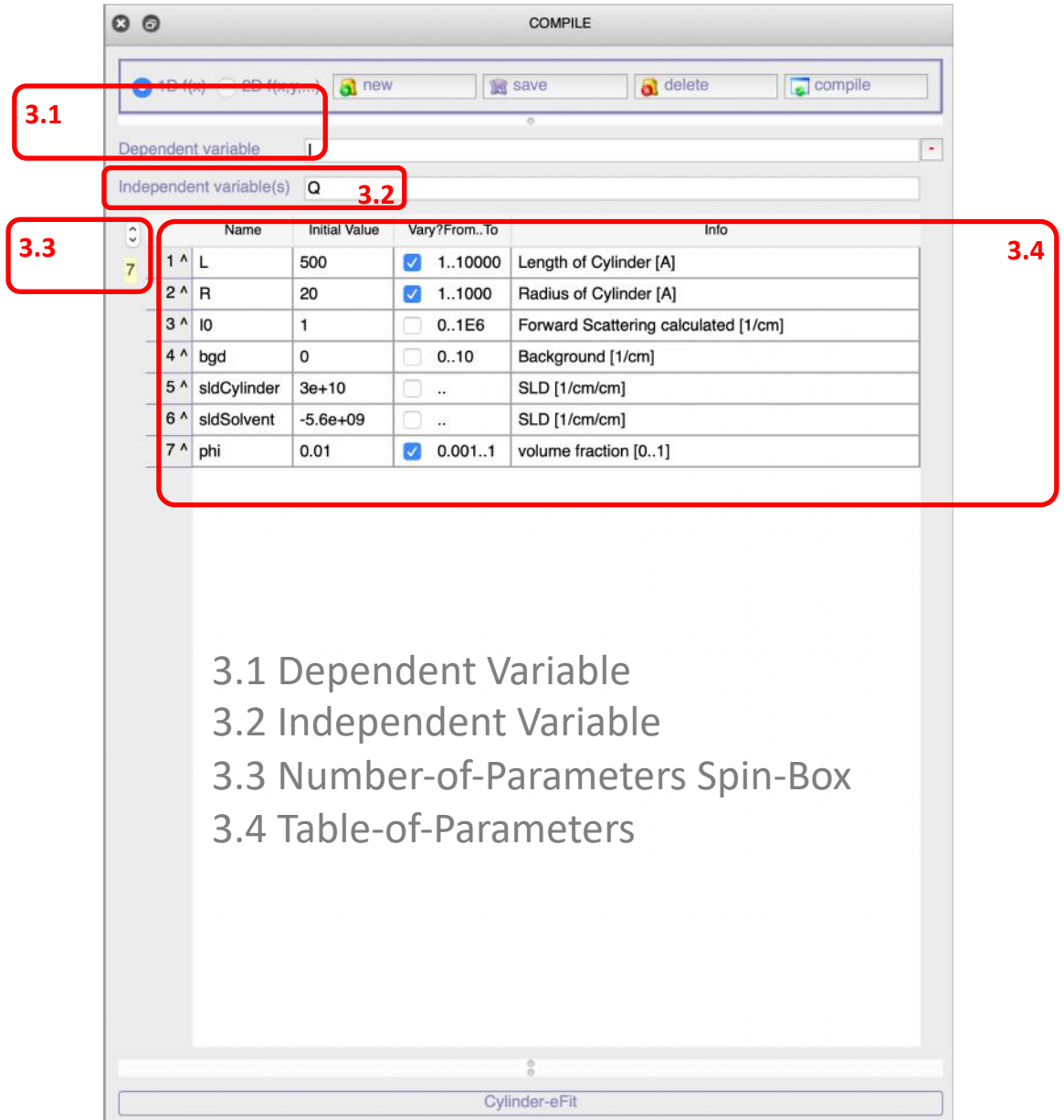


5. Check Options

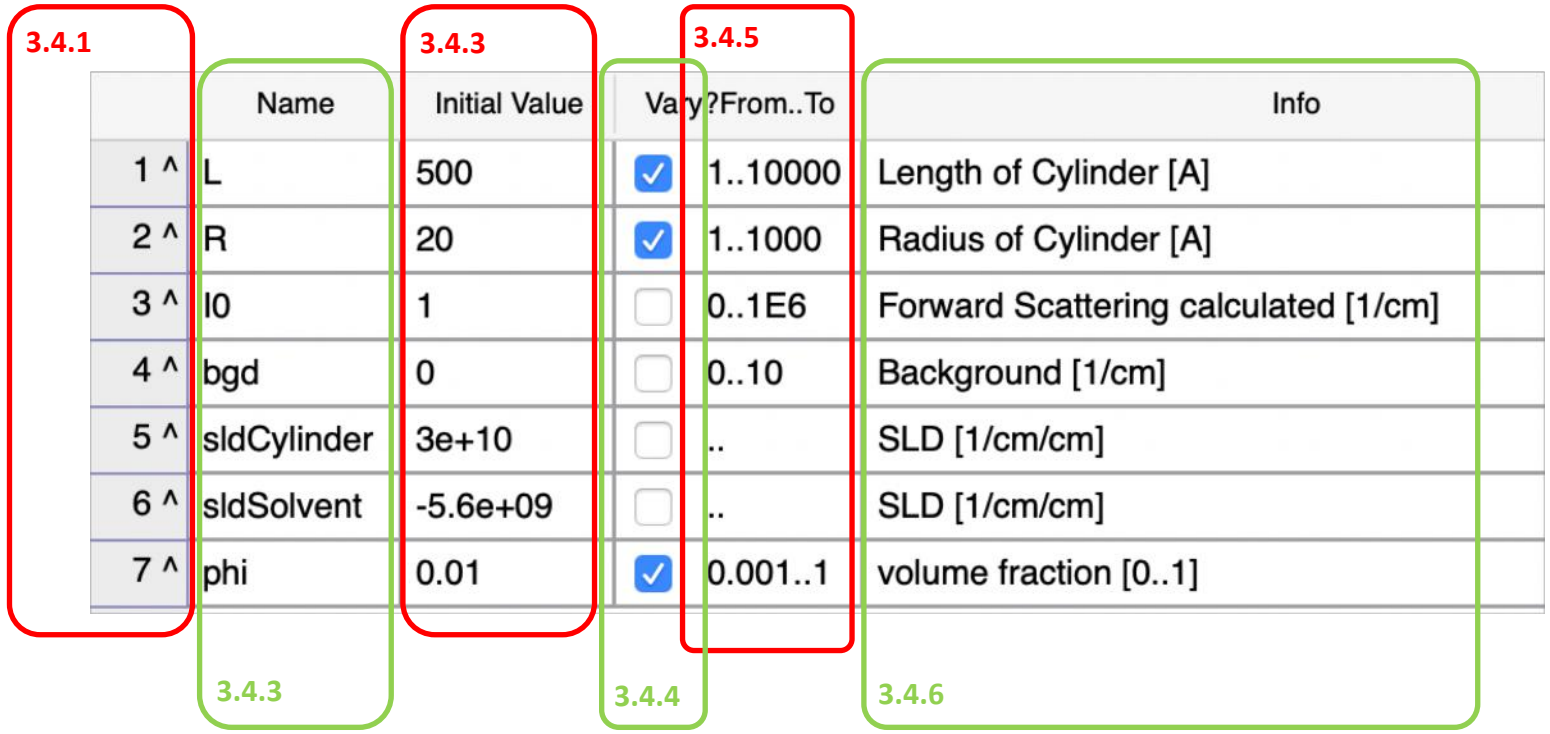
3. Parameters Browser: Example : Cylinder Form Factor

Expanding of the Parameter Browser Button



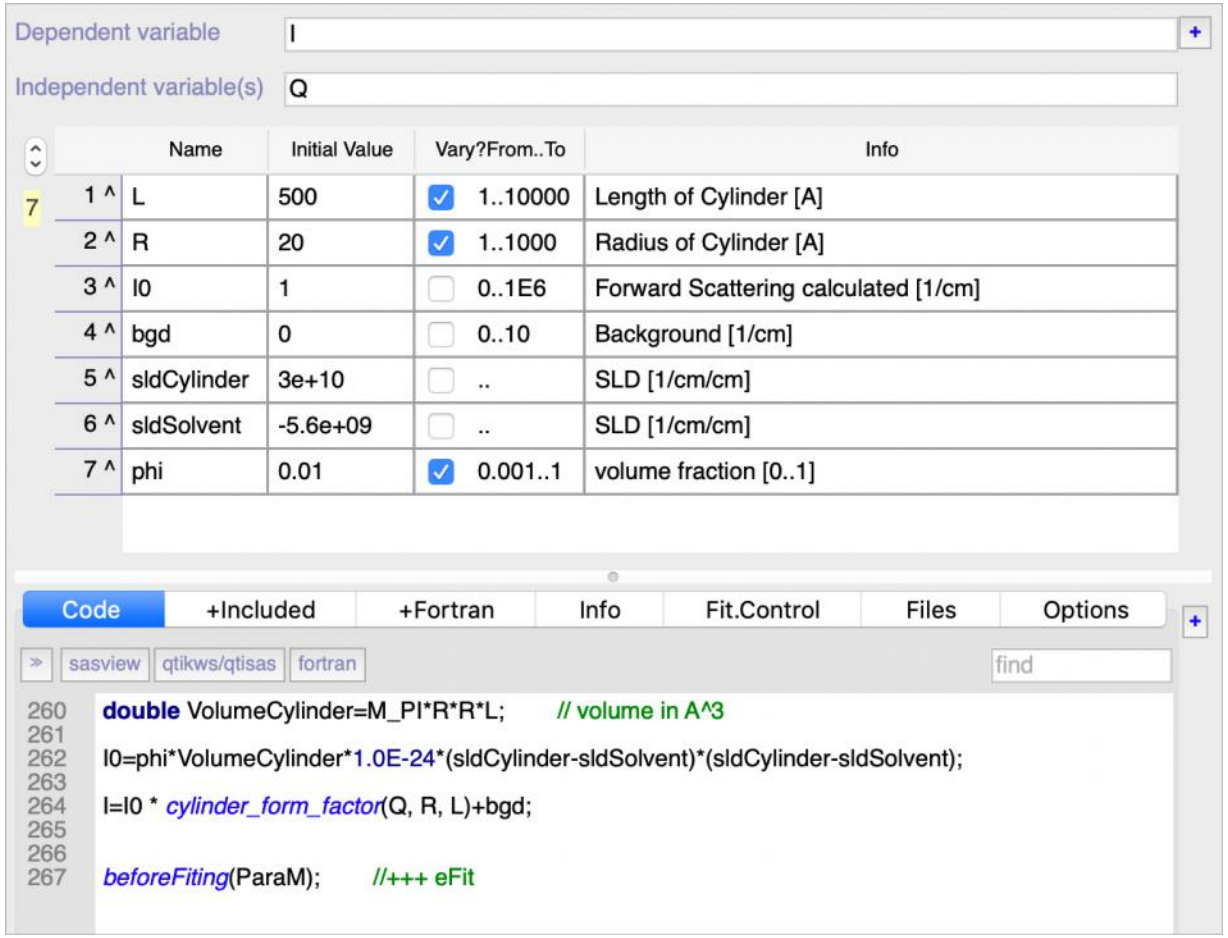


3.4 Table-of-Parameters



3.4.1 Number of the current Parameter; pushing of the "^" symbol will shift pushed row up.
3.4.2 Names of parameters: defined in c++ code.
3.4.3 Initial Values of Parameters.
3.4.4 Check-box of initial adjustability of the parameters
3.4.5 Range of Parameters: if empty (".."), range is [–inf . .+inf]
3.4.6 Info-Column: Description of Parameters

- **L, R, phi** : initially **fittable** parameters (adjustability could be changed in the fitting interface later)
- **bgd, sldCylinder, sldSolvent**: initially **fixed** parameters (adjustability could be changed in the fitting interface later)
- **I0**: parameter **calculated** inside of the function -> `I0=phi*VolumeCylinder*1.0E-24*(sldCylinder-sldSolvent)*(sldCylinder-sldSolvent);`
- **VolumeCylinder**:: Internal Variable (volume): `double VolumeCylinder=M_PI*R*R*L;  // volume in A^3`



Functions

- *cylinder_form_factor*(Q, R, L)
- *beforeFiting*(ParaM)

Defined in **4.2 „+Include“: C++ Function Editor**

4.1 „Code“: C++ Function Editor

Inside of the “Code” we defined next function:
I = I(Q, L, R, I0, bgd, sldCylinder, sldSolvent, phi)

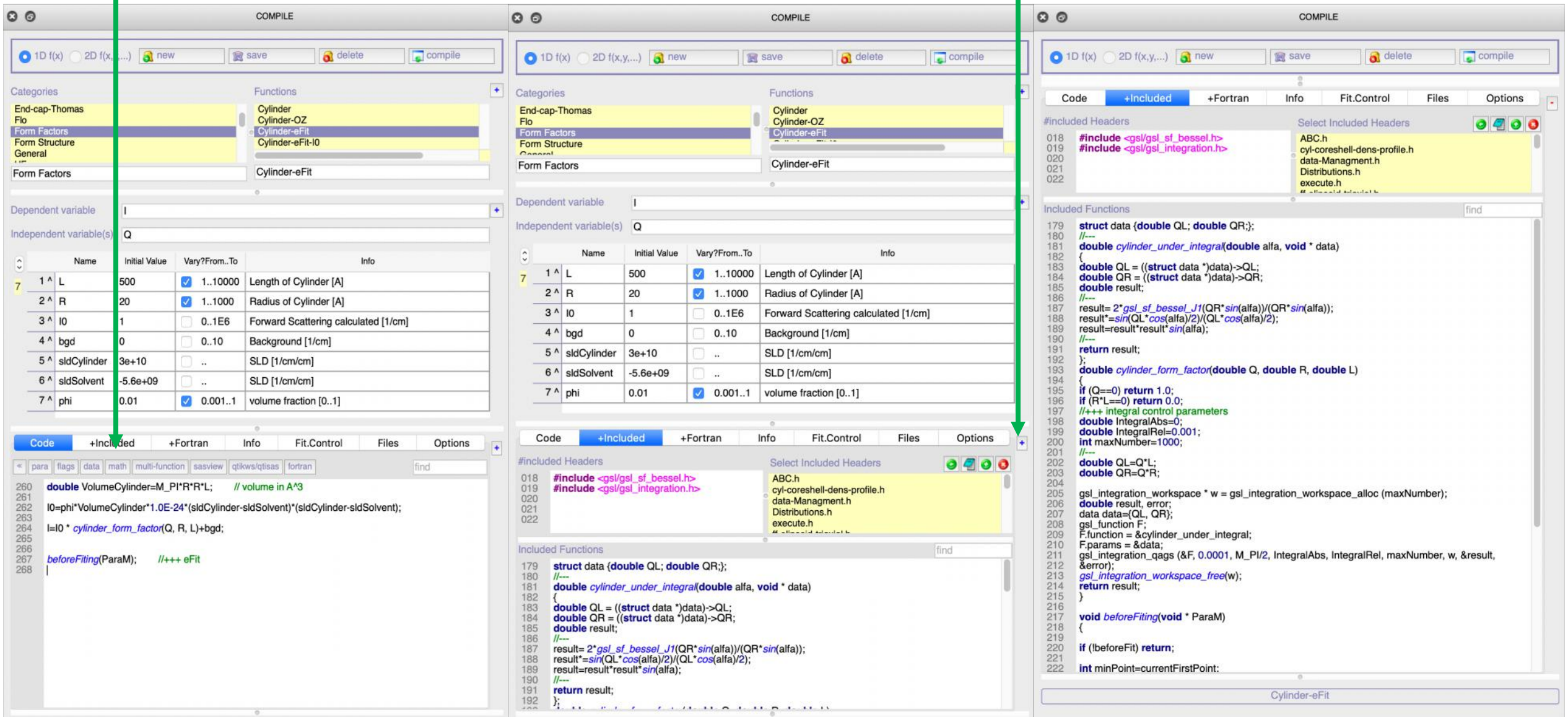
4.1 „Code“: C++ Function Editor

4.1.1 Useful (non-critical) tools [will be explained later]
4.1.2 Line Number in the generated C++ file (important during compilation/debugging)
4.1.3 C++ code editor I=I(Q, parameters)
4.1.4 Find text in the code editor (type text + “enter”)
4.1.5 Tab of Tools: expansion/shrinking button

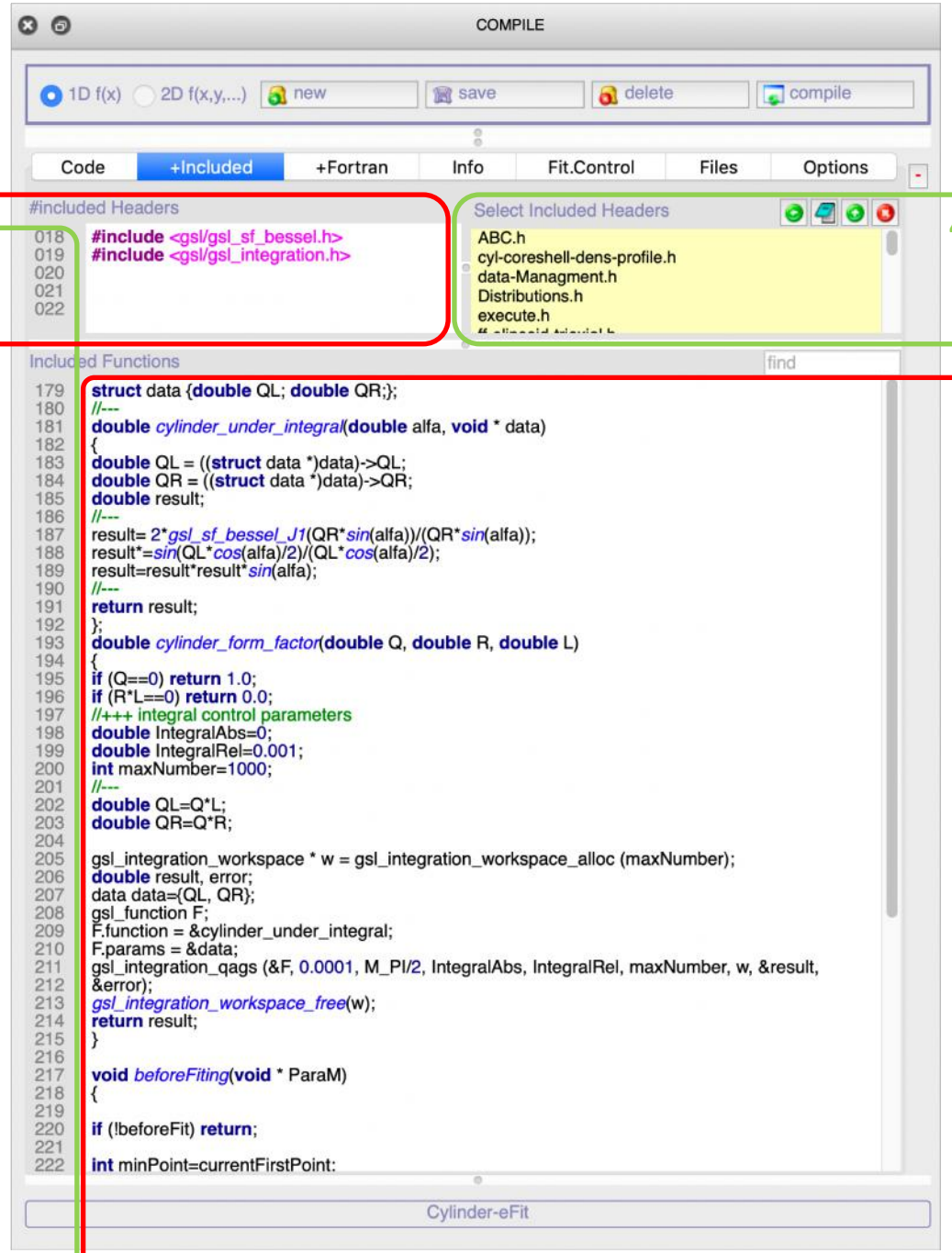
4.2 „+Included“: C++ Editor of Functions

1. Change Active Tab to +Included

2. Push: Expanding of Tools Tab Button



4.2 „+Included“: C++ Editor of Functions



4.2.1 Place for declaration of needed c++ headers

4.2.2 Line Number in the generated C++

4.2.3 C++ Function editor
[functions defined here could be called in the 4.1 function editor]

4.2.4 Headers Browser of header files
[headers path: **./qtiSAS/FitFunctions/ IncludedFunctions/**]
[not used in this example]
[will be explained somewhere else]

GSL - GNU Scientific Library

!!! integrated in Fit-Compile v 1.15

The GNU Scientific Library (GSL) is a numerical library for C and C++ programmers. It is free software under the GNU General Public License.

The library provides a wide range of mathematical routines such as random number generators, special functions and least-squares fitting. There are over 1000 functions in total with an extensive test suite.

Unlike the licenses of proprietary numerical libraries the license of GSL does not restrict scientific cooperation. It allows you to share your programs freely with others.

https://www.gnu.org/software/gsl/

```
#included Headers
018  #include <gsl/gsl_sf_bessel.h>
019  #include <gsl/gsl_integration.h>
020
021
022
```

In the Example we use 2 modules from GSL library

- Bessel Functions module (we use J1(x) function)
  - gsl/gsl_sf_bessel.h
  - https://www.gnu.org/software/gsl/doc/html/specfunc.html?highlight=bessel

- Numerical integral module
  - gsl/gsl_integration.h
  - https://www.gnu.org/software/gsl/doc/html/integration.html

cylinder_form_factor (Q, R, L) = $\int_0^{\pi} F^2(Q, L, R, \alpha) sin(\alpha) d\alpha$

```
double cylinder_form_factor(double Q, double R, double L)
{
if (Q==0) return 1.0;
if (R*L==0) return 0.0;
//+++ integral control parameters
double IntegralAbs=0;
double IntegralRel=0.001;
int maxNumber=1000;
//---
double QL=Q*L;
double QR=Q*R;

gsl_integration_workspace * w = gsl_integration_workspace_alloc (maxNumber);
double result, error;
data data={QL, QR};
gsl_function F;
F.function = &cylinder_under_integral;
F.params = &data;
gsl_integration_qags (&F, 0.0001, M_PI/2, IntegralAbs, IntegralRel, maxNumber, w, &result, &error);
gsl_integration_workspace_free(w);
return result;
}
```

$F(Q, L, R, \alpha)$ = **cylinder_under_integral** (α, data) = $2 \times \frac{J_1[QRsin(\alpha)]}{QRsin(\alpha)} \times \frac{sin\left[{}^{QLcos(\alpha)}/_{2}\right]}{{}^{QLcos(\alpha)}/_{2}}$

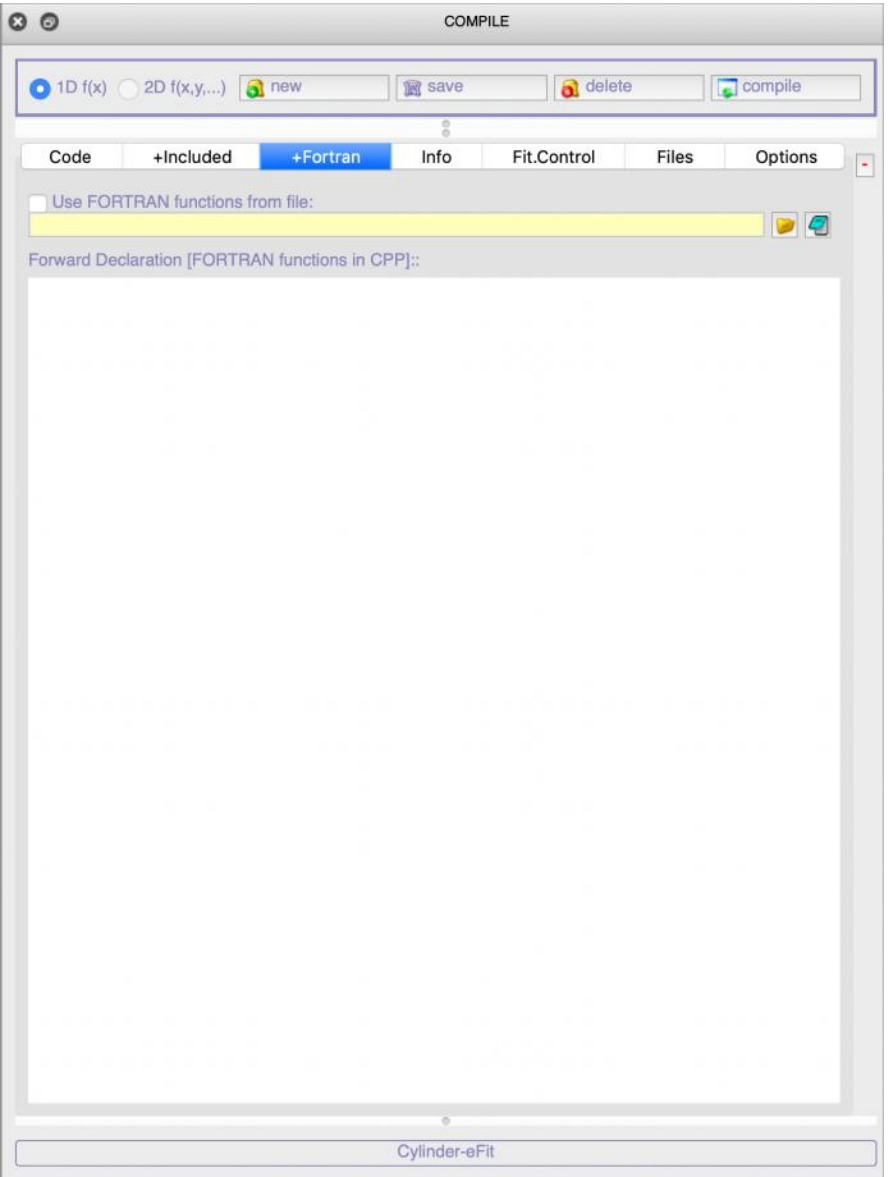
```
double cylinder_under_integral(double alfa, void * data)
{
double QL = ((struct data *)data)->QL;
double QR = ((struct data *)data)->QR;
double result;
//---
result= 2*gsl_sf_bessel_J1(QR*sin(alfa))/(QR*sin(alfa));
result*=sin(QL*cos(alfa)/2)/(QL*cos(alfa)/2);
result=result*result*sin(alfa);
//---
return result;
};
```

- Bessel Functions module (we use J1(x) function)
  - https://www.gnu.org/software/gsl/doc/html/specfunc.html?highlight=bessel
- Numerical integral module
  - https://www.gnu.org/software/gsl/doc/html/integration.html

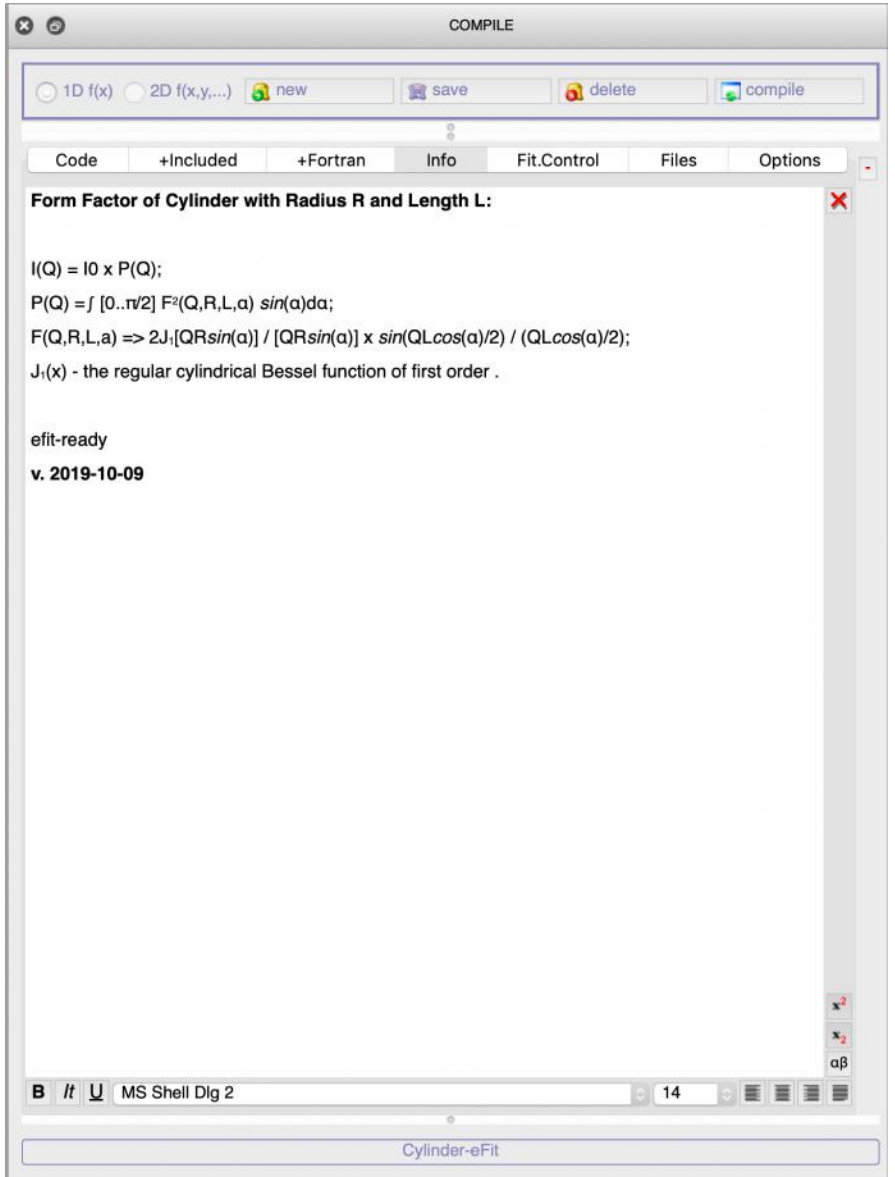
4.3 „+Fortran“: Fortran Function Editor

In current example we do not use any Fortran function.

https://www.qtisas.com/compile/fortran

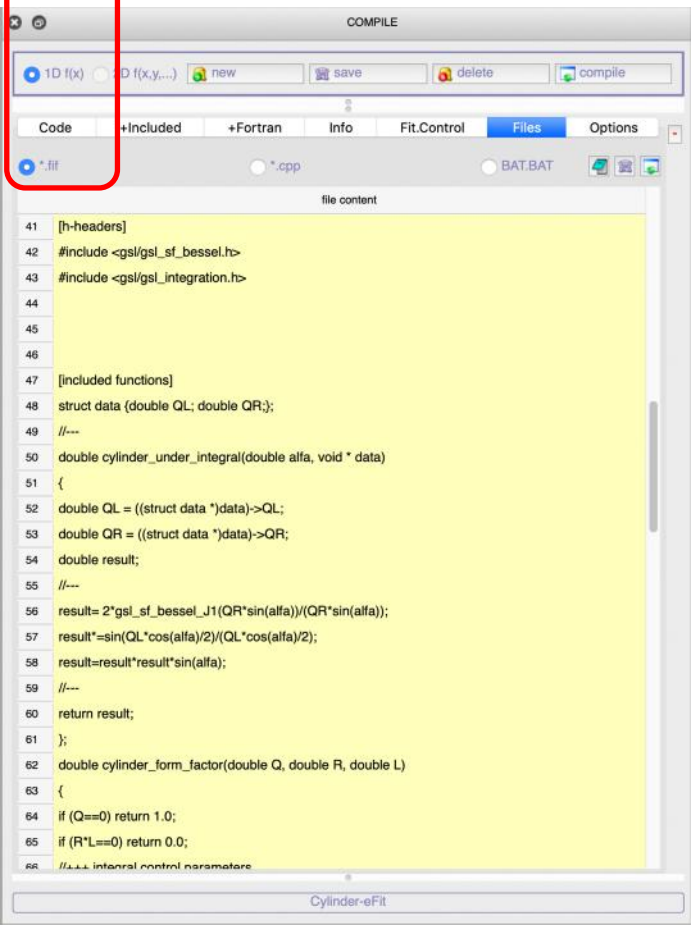


4.4 „Info“: Information Editor

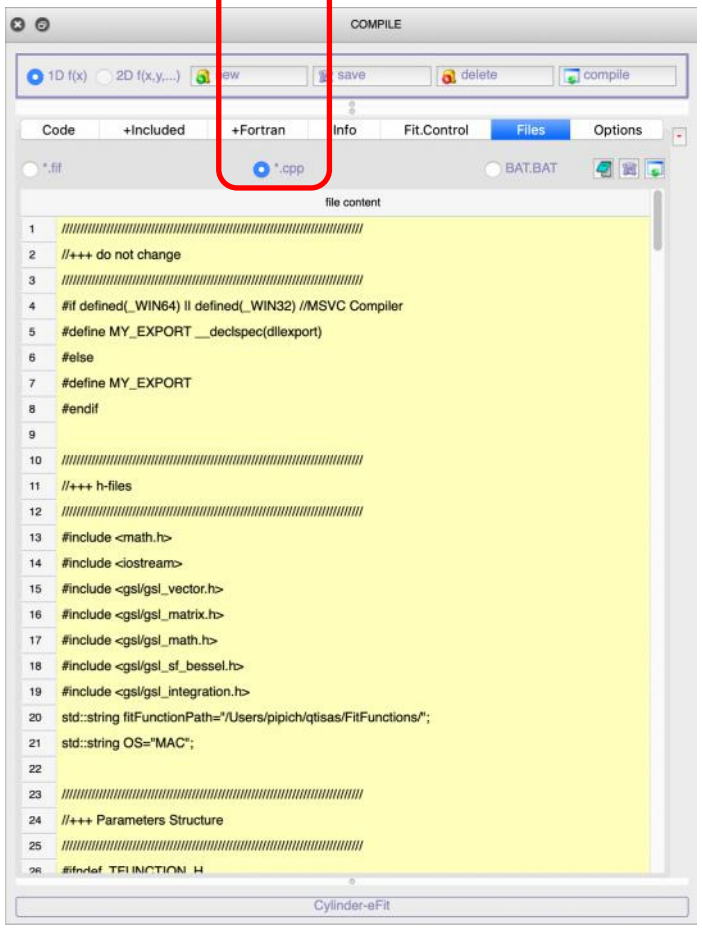


4.6 “Files”: Generated Files Viewer

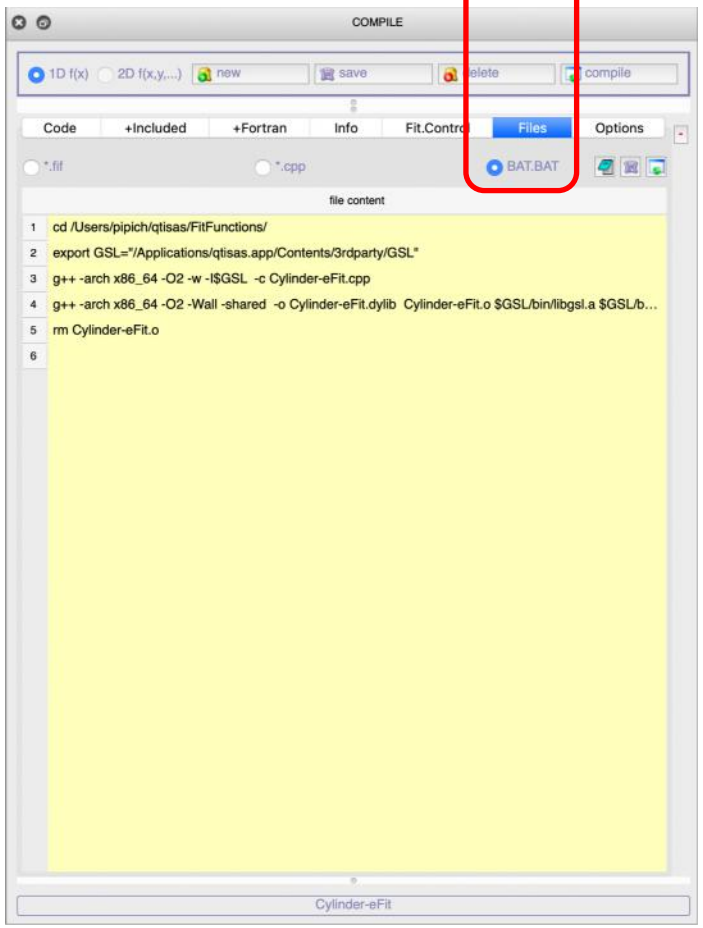
4.6.1



4.6.2



4.6.3

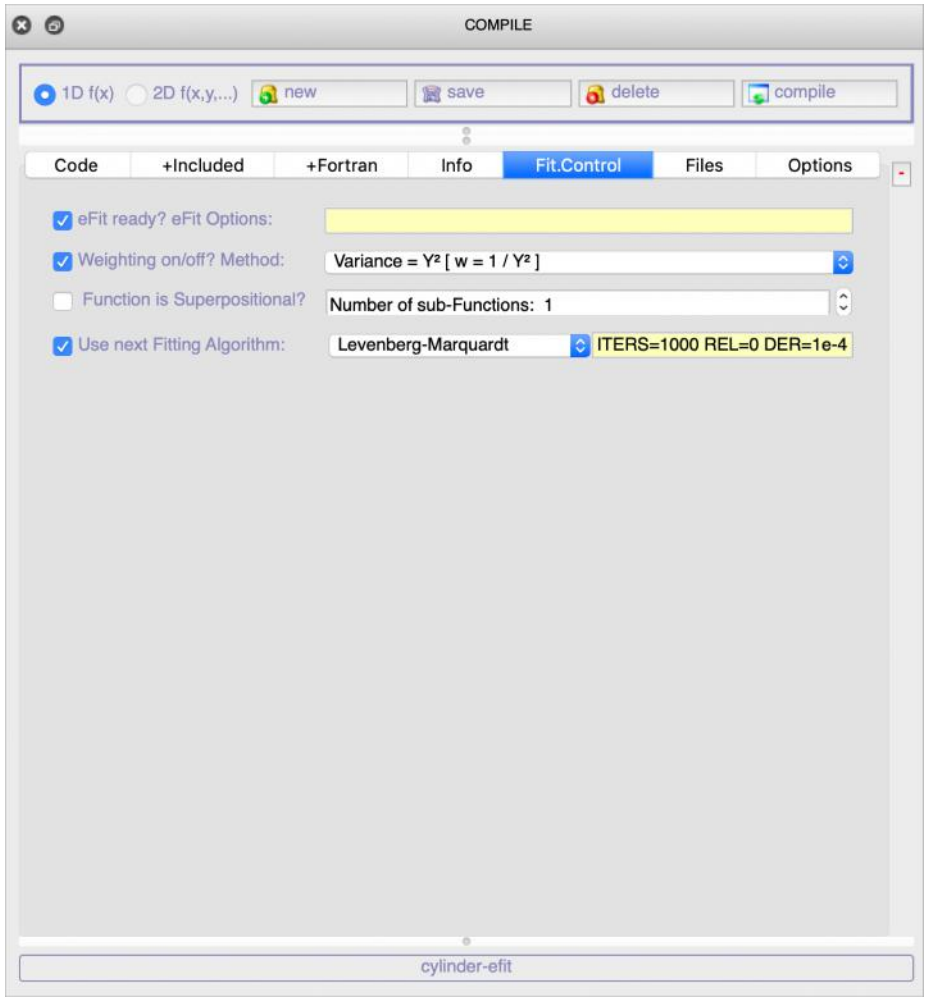


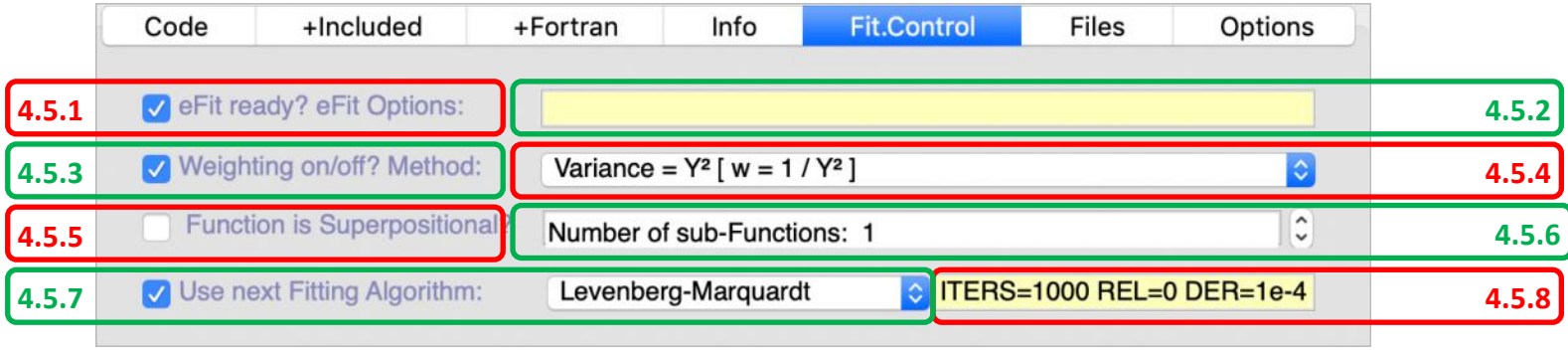
4.6.1 Information from the interface is saved in ***.fif** file

4.6.2 FIF-file is converted to c++ file: ***.cpp**

4.6.3 c++ file is compiled by **BAT.BAT** script file

4.5 “Fit.Control”: Fitting Options





Fit-control for the active function.

4.5.1 Check-Box: eFit-ready
4.5.2 Line-Edit: eFit-Options [see next page]
4.5.3 Check-Box: By default use **Weighting** for the active function if checked
4.5.4 Combo-box: default **Weighting** Method for the active function
4.5.5 Check-Box: super positional Function: if checked function has several sub-functions. Info will be here: https://www.qtisas.com/fitting/superpositional-fit
4.5.6 Number of sub-functions
4.5.7 Check-Box: default **Fitting** algorithm for the active function
4.5.8 Options for active fitting algorithm

4.5.1 “Fit.Control”: Check-Box: eFit-Option

e-FIT ☑ e-Fit+++

default eFit procedure:

[1] Script :: Before Fit

"Before Fit"; to skip it: write here **"no-before"**

[2] Fit

"Fit"; to skip it: write here **"no-fit"**

"color=color_name"; select color of the fitting curve

[3] Script :: After Fit

"After Fit"; to skip it: write here **"no-after"**

[4] Simulate

"Simulate"; by default after fit fitting curve is calculated and plotted in the plot;

name of the default table "fitCurve-FUNCTION-NAME";

if you want to have "unique" name write here **"name=TABLE-NAME"**

In case of **"no-fit":** if you want just to simulate data, select option **"yes-simulate"**

[5] Results of the fit is shown in Res-Log:

"no-reslog": to skip it

"yes-res-in-plot": to show results in graph

4.5.8 “Fit.Control”: Options for active fitting algorithm

MODE=3 ITERS=1000 REL=0 DER=1e-4

simplex:

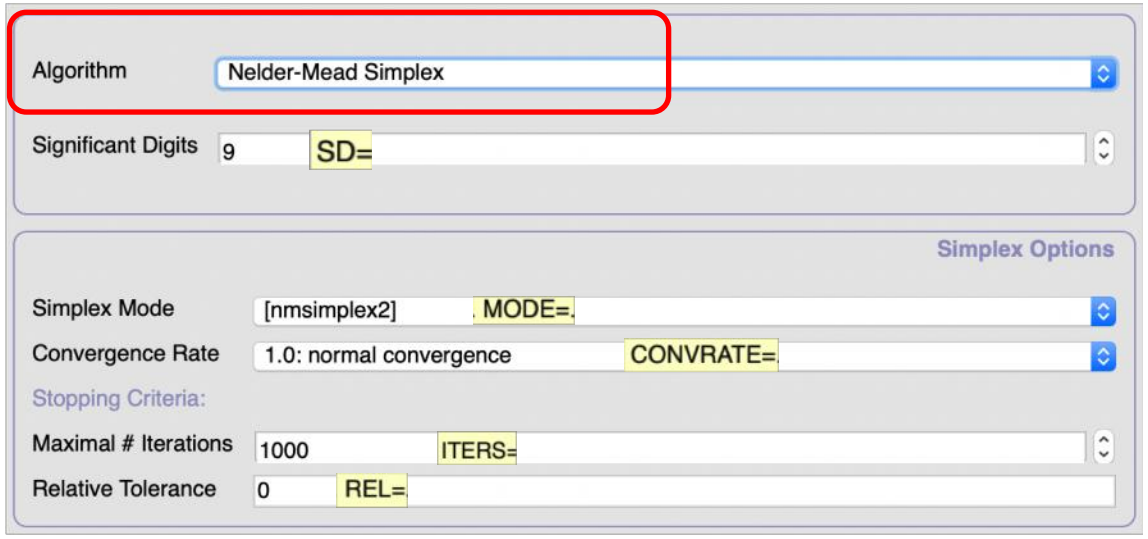
SD=... MODE=... ITERS=... REL=... CONVRATE=...

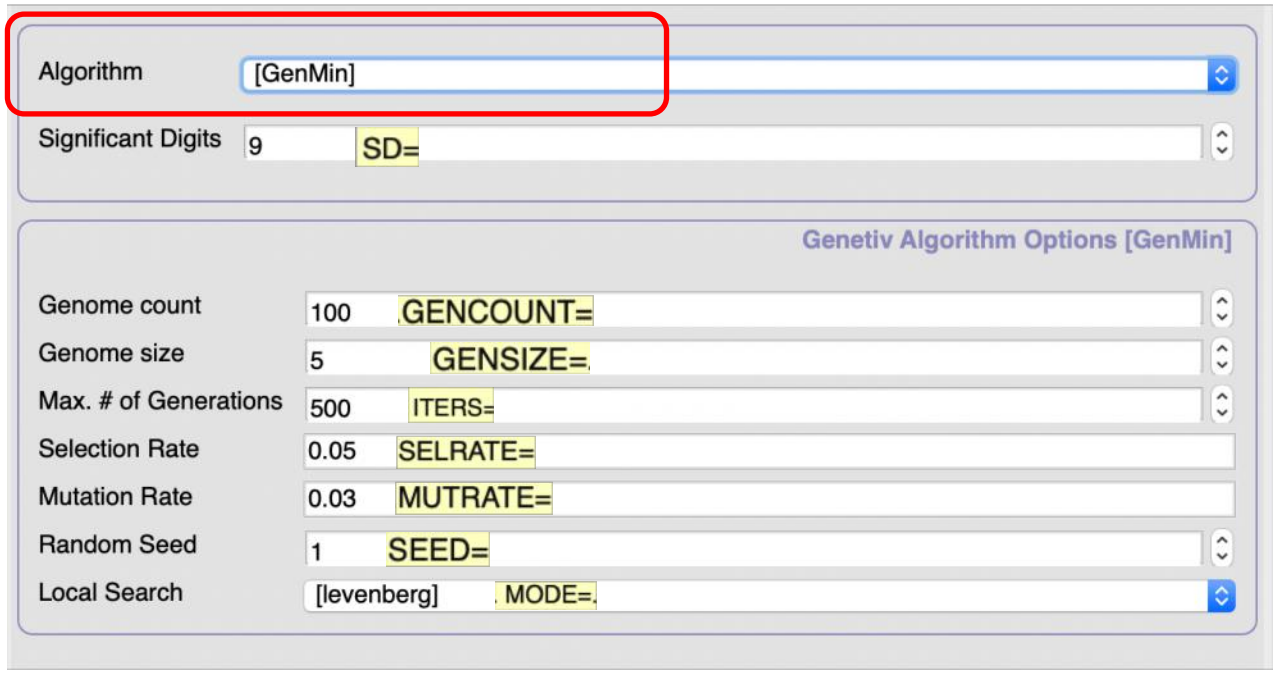
levenberg:

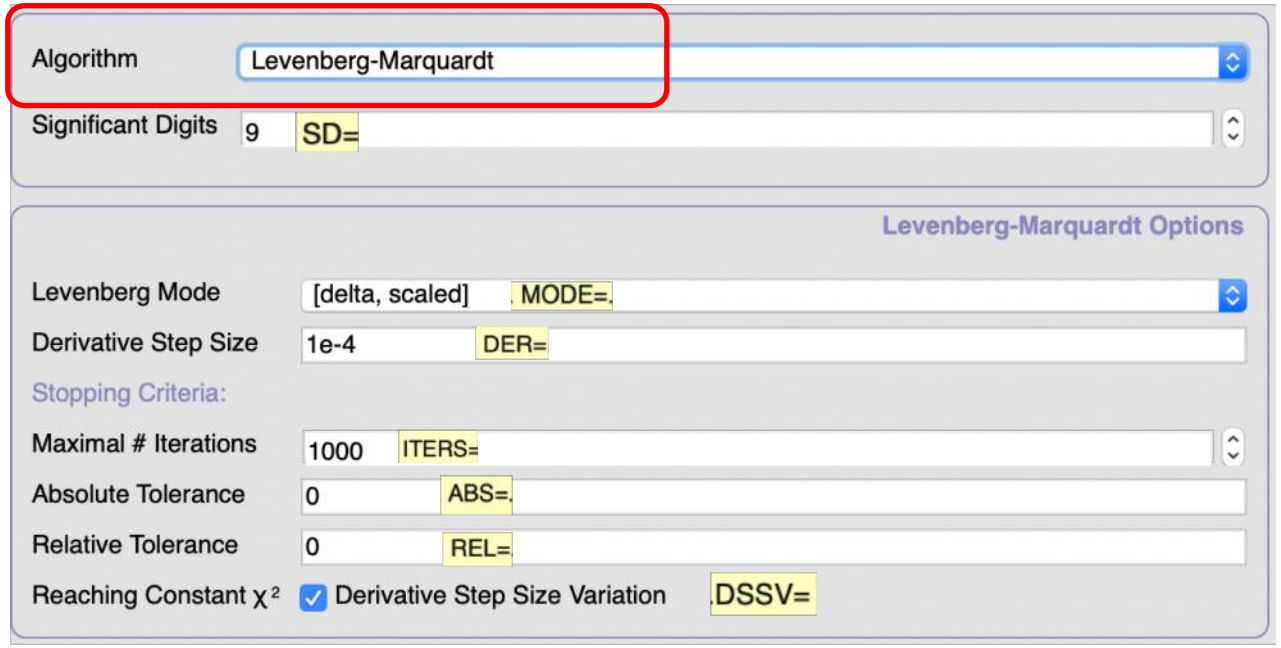
SD=... MODE=... ITERS=... REL=... ABS=... DER=...DSSV=

genetic:

SD=...GENCOUNT=...GENSIZE=...ITERS=...SELRATE=...MUTRATE=...SEED=...MODE=...







4.7 “Options”: Fit-Compile Interface Options