Bound-constrained quasi-Newton optimization by LBFGSB can be used to minimize a user-defined model, but for this procedure it requires starting estimates and the partial derivatives to be provided as well as the function to be minimized.

So the user supplied model must define $n + 1$ functions of n variables as follows

$$\begin{aligned} f(1) &= F(x_1, x_2, \ldots, x_n) \\ f(2) &= \partial F/\partial x_1 \\ f(3) &= \partial F/\partial x_2 \\ &\ldots \\ f(n+1) &= \partial F/\partial x_n. \end{aligned}$$

The limited memory quasi-Newton optimization procedure also requires several other parameters, as now listed.

- $MHESS$ is the number of limited memory corrections to the Hessian that are stored. The value of 5 is recommended but, for difficult problems, this can be varied in the range 4 to 17.
- $FACTR$ should be about 1.0e+12 for low precision, 1.0e+07 for medium precision, and 1.0e+01 for high precision. Convergence is controlled by $FACTR$ and $PGTOL$ and will be accepted if

  $$|F_k - F_{k+1}|/\max(|F_k|, |F_{k+1}|, 1) \leq FACTR * EPSMCH$$

  at iteration $k + 1$, where $EPSMCH$ is machine precision, or if

  $$\max_i(\text{Projected Gradient}(i)) \leq PGTOL.$$

- Starting estimates and bounds on the variables can be set by editing the defaults or by installing from a data file.
- The parameter $IPRINT$ allows intermediate output every $IPRINT$ iterations, and the final gradient vector can also be printed if required.
- The program opens two files at the start of each optimization session, `w_usermod.err` stores intermediate output every $IPRINT$ iterations plus any error messages, while `iterate.dat` stores all iteration details, as for SIMFIT programs **qnfit** and **deqsol** when they use the LBFGSB suite for optimization.
- Note that, when $IPRINT > 100$ full output, including intermediate coordinates, is written to `w_usermod.err` at each iteration.

As an example, input the model file `optimum_e.mod` defining Rosenbruck's two dimensional test function

$$F(x, y) = 100(y - x^2)^2 + (1 - x)^2$$

which has a unique minimum at $x = 1, y = 1$ and is represented as follows.

```
 %
f(1) = 100(y - x^2)^2 + (1 - x)^2
f(2) = -400x(y - x^2) - 2(1 - x)
f(3) =  200(y - x^2)
%
 3 equations
 2 variables
 0 parameters
 %
 begin{expression}
 A = y - x^2
 B = 1 - x
 f(1) = 100*A^2 + B^2
 f(2) = -400A*x - 2B
 f(3) = 200A
 end{expression}
 %
```

In order to locate a minimum of this function it is necessary to specify the starting estimates along with parameters controlling the optimization and the output.

For example, the iteration using the default optimization parameters and good starting estimates, in particular

$$-10 \le x \le 10, \quad x_{start} = 0$$
$$-10 \le y \le 10, \quad y_{start} = 0$$

with IPRINT = 5 for output at every fifth iteration proceeds as in the next table.

Iterate	$F(x,y)$	$\|prj.grd.\|$	$Task$
1	6.9219E-01	5.0534E+00	NEW_X
6	2.1146E-01	3.1782E+00	NEW_X
11	1.7938E-02	3.5920E-01	NEW_X
16	1.7768E-04	4.4729E-02	NEW_X
20	5.5951E-13	7.2120E-06	$CONVERGENCE:$ $NORM\ OF\ PROJECTED$ $GRADIENT \le PGTOL$
Solution	Derivatives		
$x = 1$	$\partial F/\partial x$	7.21198E-06	
$y = 1$	$\partial F/\partial y$	-2.87189E-06	

The parameter $Task$ informs users of the action required after each intermediate iteration, then finally it records the reason for termination of the optimization.

As this type of minimization can only locate a local minimum and success depends critically on scaling the problems and choosing good starting estimates, it will usually be necessary to experiment with alternative settings until a reliable convergence has been achieved.

Actually program **usermod** opens two files `iterate.dat` and `w_usermod.err` in the user folder which can be consulted retrospectively to follow the iterations.

In particular, choosing the maximum value of IPRINT = 101 causes output to `w_usermod.err` for every iteration as well as error reports, and these data can be used retrospectively to create contour maps with the optimization trajectory overlaid. This will be discussed in another tutorial.