Chapter 5: Optimization

Many econometric methods rely on the maximisation of the (log-) likelihood. There are many solutions to that end in GROCER. You can use Scilab optim function which minimizes a convex function. You can also use GROCER translation-adaptation of all but one minimization functions of James LeSage econometric toolbox, the most efficient being a function named maxlik, as in Gauss. Note also that Scilab contains a function named fsolve which can find a zero of a function (see Scilab manual or on-line help). Part 1 of this chapter is a brief description of these functions and presents the terms of the choice for a user. Part 2 provides, as for every chapter, the specification of all these functions along with a few examples.

1. A brief presentation of the optimization functions[1].

For its own needs, James Lesage has programmed a number of optimization algorithms existing in the literature: the Davidon-Fletcher-Powell, the Powell, Fletcher,Reeves,Polak,Ribiere and (from Mike Cliff) other variants of the Newton-like methods (Gauss-Newton, Marquard, steepest descent and especially BFGS[2]) embodied in one function: the corresponding functions are dfp_min, pow_min, frpr_min and maxlik.

The functions dfp_min, pow_min, frpr_min have all the same specifications: they take as input the function name, starting values for the parameters (these 2 arguments are compulsory in that order) and optional arguments. These optional arguments can be one or more of 4 options for the maximisation: maxit, the number of maximum iterations; tol, the tolerance parameter indicating numerical convergence of the algorithm; delta, the increment for the calculation of the numerical derivative of the function and optprt which, if equal to 1, provides the parameters and function values at each iteration.

An example (the syntax would be exactly the same with dfp_min or frpr_min) is the following:

```
-->dfp_min(to_llike,ones(5,1),'maxit=1000',’tol=1e-6’,’optprt=1’,’delta=1e-
4’,y,x);
```

In this example, the function to maximise is to_llike (see chapter 6). This function has 3 input: the parameters over which the maximisation is done, y and x. The starting values are set to ones(5,1) and all the parameters maxit, delta, optprt and tol are given values different from the default ones.

If instead the user had typed:

```
-->dfp_min(to_llike,ones(5,1),'maxit=500',’tol=1e-5’,’optprt=0’,’delta=1e-
5’,y,x);
```

[1] For a survey of optimization methods see for instance R. Fletcher, An overview of unconstrained optimization, in Algorithms for Continuous Optimization: The State of the Art, E. Spedicato, ed., Kluwer Academic Publishers, Boston, 1994, pp. 109-143, available at http://citeseer.nj.nec.com/fletcher93overview.html

[2] For: Broyden, Fletcher, Goldfarb, Shanno

it would have been equivalent to:

```
-->dfp_min(to_llike,ones(5,1) y,x);
```

It is also possible to change only part of the default values. For instance:

```
-->dfp_min(to_llike,ones(5,1),'maxit=1000',’tol=1e-6’,y,x);
```

works with optprt=0 and delta=1e-5.

The function maxlik is much more sophisticated in one dimension: it allows a great number of options. If one does not want to bother with these options, one can ignore them and let the program uses the default ones: they include the maximum number of iterations (default: 100), the various convergence criterions (on the parameters, the function, the gradient); the gradient function (default: the numerical gradient numz0); the increment used to calculate the numerical derivative; the method used to update the hessian (default is BFGS); the maximum authorized for the condition number of the hessian; the lambda factor in Marquard method.

A call to maxlik for the function garch_like (see chapter 11) could then be:

```
-->maxlik(garch_like,ones(8,1),1,2,y,x,'grad=garch_grad','btol=1e-5','ftol=1e-
5','gtol=1e-5','cond=10000','oprprt=0','hess=marq','lambda=0.02')
```

The function to maximise, garch_like, takes 4 arguments which are parm (the parameter values), the orders p and q of the garch(p,q) models (here 1 and 2 respectively), a vector of endogenous variables (here y) and a matrix of exogenous variables (here x). All optional arguments are here given other values than the default ones: the gradient function is here explicitly provided (garch_gard); all convergence criteria (btol, ftol, gtol) are set to 1E-5; the condition number is relaxed (10000 instead of 1000); the method used to update the Hessian is the Marquard one instead of the BFGS one and the value of lambda is set to 0.02 instead of the 0.01 default value; the cause of the stop is not printed (not recommended: the program could simply stop because the maximum number of iterations has been reached).

A much simpler call to this function could be:

```
-->maxlik(garch_like,ones(8,1),1,2,y,x)
```

Note however that you loose the benefit arising from the programming of the exact gradient. So a recommended call (which is indeed the default call in GROCER function garch) should be:

```
-->maxlik(garch_like,ones(8,1),1,2,y,x,'grad=garch_grad')
```

The empirical literature today favors the BFGS method. This is also the one that Scilab function optim uses. An experiment, similar to the one provided by James Lesage (chapter 9 of his book: “Applied Econometrics using MATLAB”) and extended to deal also with optim, shows the main estimation results for the estimation a tobit model (you can reproduce this experiment by executing the function: optim1_d; obviously the execution time can be different, dependaing on the speed of your processor, but the hierarchy should be the same). Table 5.1 thereafter shows that the estimation results are (happily!) the same, but also that maxlik is the speediest of all

these functions. Scilab optim function is however sometimes more robust than maxlik; so, in some cases, notably when the minimum is difficult to find, it can be safe to use also optim.

So, in most GROCER functions (such as nls, garch, kalman…) that needs a maximisation function, the choice between malik (default) and optim is always offered to the user.

Table 5.1: Comparison of optimization results

	optim	maxlik	powell	frpr	dfp
bhat estimate(1)	.9575415	.9575409	.9575410	.9575318	.9575413
bhat estimate(2)	.9658022	.9658019	.9658025	.9658058	.9658029
bhat estimate(3)	.8956249	.8956247	.8956247	.8956146	.8956250
bhat estimate(4)	.9424002	.9424003	.9423993	.9423898	.9424007
bhat estimate(5)	1.0319505	1.0319506	1.0319502	1.0319564	1.0319499
sige estimate	.8091094	.8091099	.8090889	.8091175	.8091090
likelihood estimate	169.53862	169.53862	169.53862	169.53862	169.53862
execution time	.735	.687	7.188	1.687	1.547

2. The optimization functions and their specifications.

dfp_min_____DFP minimization

CALLING SEQUENCE

[result]=dfp_min(func,pin,varargin)

PARAMETERS

INPUT:
* func = (minus likelihood) function to minimize
* pin = parameter vector fed to func
* varargin = 'maxit=x' to set the maximum # of iterations (default = 500) and arguments of function func others that the parameters

---

OUTPUT:
a result list with:
- result('meth') = 'dfp'
- result('b') = (kx1) minimizing vector
- result('f') = value of func at solution values
- result('hess') = hessian at solution values
- result('iter') = # number of iterations
- result('flag') = 0 for convergence, 1 for non-convergence

DESCRIPTION
DFP minimization routine to minimize func (Converted from Numerical Recipes book dfpmin routine)
NOTE: func must take the form func(b,varargin)
where: b = parameter vector (k x 1)
varargin = arguments passed to the function

Example:
res = ols2(y,x);result4 = dfp_min('to_llike', [res('beta'); res('sige')],'maxit=1000',y,x);

This example, taken from optim1_d, realises the estimation of a tobit model, with initial values taken from on ols estimation. Maximum # of iterations is set to 1000.

frpr_min_____Fletcher, Reeves, Polak, Ribiere minimization

CALLING SEQUENCE

[result]=frpr_min(func,pin,varargin)

PARAMETERS

INPUT:
* func = (minus likelihood) function to minimize
* pin = parameter vector fed to func
* varargin = 'maxit=x' to set the maximum # of iterations (default = 500) and arguments of function func others that the parameters
---
OUTPUT:
result = a result list with:
- result('meth') = 'dfp'
- result('b') = (kx1) minimizing vector
- result('f') = value of func at solution values
- result('hess') = hessian at solution values
- result('iter') = # number of iterations
- result('flag') = 0 for convergence, 1 for non-convergence

DESCRIPTION
Fletcher, Reeves, Polak, Ribiere minimization routine to minimize func (Converted from Numerical Recipes book frprmn routine).
NOTE: func must take the form func(b,varargin)
where: b = parameter vector (k x 1)
varargin = arguments passed to the function

Example:
res = ols2(y,x);result4 = dfp_min('to_llike', [res('beta'); res('sige')],'maxit=1000',y,x);

This example, taken from optim1_d, realises the estimation of a tobit model, with initial values taken from on ols estimation. Maximum # of iterations is set to 1000.

hessian_____finite symmetric difference Hessian

CALLING SEQUENCE

H=hessian(namefunc,p,varargin)

PARAMETERS

INPUT:
* namefunc = name of the function
* p = point where to evaluate the function
* vararagin = arguments other than p of funtion namefunc

OUTPUT:
* H = value of the hessian of namefunc at point p

DESCRIPTION
Computes finite symmetric difference Hessian.

Example:
chh='hessian("filter0",param,'"+func+"',y,x,F)'; vcovt=inv(evstr(chh))

This example, taken from kalman provides the calculation of the variance-covariance matrix of the estimated parameters of a kalman filtering.

linmin_____line minimization routine

CALLING SEQUENCE

[pout,fout]=linmin(pin,xi,tol,f,varargin)

PARAMETERS

INPUT:
* pin = (kx1) vector of starting values
* xi = (kx1) direction vector
* tol = tolerance
* func = function name string
* varargin = arguments passed to func

OUTPUT:
* pout = (kx1) minimizing vector
* fout = (kx1) value of func at minimum pout values

DESCRIPTION
Line minimization routine that performs an ad hoc n-dimensional Golden Section Search for the minimum of a function (Converted from Numerical Recipes book linmin routine).

NOTES:
* func must take the form func(b,varargin)
  where: b = parameter vector (k x 1)
  varargin = arguments passed to the function
* used by: dfp_min(), pow_min(), frpr_min()

Example:
[pin fret] = linmin(pin,xi,tol,func,varargin(:))

This command is used in frpr_min to ajust the value of the searched parameter at each iteration.

maxlik_____maximization function

CALLING SEQUENCE

[results]=maxlik(func,xarg,varargin)

PARAMETERS

INPUT:
* func = function to be minimized
* xarg = parameter vector fed to func
* varargin =
  . optional arguments passed for infoz
    'maxit=x' to set the maximum # iterations (default 100)
    'btol=x' to set the convergence criterium for b (default 1e-5)
    'ftol=x' to set the convergence criterium for function (default 1.49e-8)
    'gtol=x' to set the convergence criterium for gradiant (default 1.49e-8)
    'dirtol=x' to set the maximum change on b authorized during an iteration (default 1)
    'cond=x' to set the maximum condition number authorized for hessian matrix (default 1000)
    'lambda=x' to set lambda in the GN/Marq hess option (default 0.01)
    'delta=x' to set the increment used in the numz func (default 0.01)
    'optprt=0' not to print the cause of the stop (default 1)
    'hess=namefunc' to set the name of an hessian update function (default hessz)
    'grad=namefunc' to set the name of a grdiant function (default numz0)
  or
  . arguments (if any) other than xarg passed to func

RETURNS: results = a results tlist with
  . results('meth') = infoz(hess)
    = 'dfp', 'bfgs', 'gn', 'marq', 'sd' (from input)
  . results('hess') = numerical hessian at the optimum
  . results('bhist') = history of b at each iteration
  . results('b') = parameter value at the optimum
  . results('f') = objective function value at the optimum
  . results('g') = gradient at the optimum
  . results('dg') = change in gradient
  . results('db') = change in b parameters
  . results('df') = change in objective function

. results('iter') = # of iterations taken
. results('time') = time (in seconds) needed to find solution
. results('infoz') = infoz (options used)

DESCRIPTION
Maximizes a (minus log likelihood) function.

Example:
1) rd=maxlik('grocer_f_ssr',grocer_init,grocer_chdelta, varargin(:),grocer_chftol)
2) oresult = maxlik('to_llike',b0,y,x,vtrunc,lopt(:))
3) oresult = maxlik('filter0',param,func,y,x,F,varargin(:))

These examples are taken from the functions nls, tobit and kalman. The functions maximized are grocer_f_ssr, to_llike and filter. The initial parameters are respectively grocer_init, b0 and param. In example 2 and 3, y, x and vtrunc (in example 2) or F (in example 3) are the parameters other than b0 of function to_llike. Grocer_chdelta, grocer_chftol, lopt(:) are varargin(:) are options for maxlik.

numz0_____numerical derivative

CALLING SEQUENCE

[g]=numz0(namefunc,param,k,g,delta,varargin)

PARAMETERS

INPUT:
* namefunc = name of function namefunc with the parameters as first argument
* param = point where the derivative is calculated
* k = dimension of param
* g = a pre-determinated matrix with dimension (k,n) where n is the dimension of namefunc
* delta = the Increment used to evaluate the derivative
* varargin = the arguments other than param in function namefunc
---
OUTPUT:
* g = numerical derivative

DESCRIPTION
Calculates a numerical derivative of function namefunc at point param. The speediest (I think !) possible in scilab.

Examples:
1) eqn=numz0(‘ssr’,ols0(y,x),size(x,2),ones(size(x,2)),1E-5,y,x)
2)grocer_chg='numz0('"+grocer_func+"',grocer_xarg,grocer_k,grocer_ik,grocer_infoz("delta"), varargin(:))';execstr(grocer_chg)

The first example provides the value of the normal equations evaluated at the ols estimator: the results should be approximately (since numz0 calculates a numerical derivative) 0. Example 2,

taken from maxlik, provides a numerical derivative of function grocer_func, at value grocer_xarg, of dimension (grocer_k x 1), with an increment taken from grocer_infoz.

pow_min_____Powell minimization

CALLING SEQUENCE

[result]=pow_min(func,pin,varargin)

PARAMETERS

INPUT:
* func = (minus likelihood) function to minimize
* pin = parameter vector fed to func
* varargin = 'maxit=x' to set the maximum # of iterations
(default = 500)

OUTPUT:
a result list with:
- result('meth') = 'pow'
- result('b') = (kx1) minimizing vector
- result('f') = value of func at solution values
- result('hess') = hessian at solution values
- result('iter') = # number of iterations
- result('flag') = 0 for convergence, 1 for non-convergence

DESCRIPTION
Powell minimization routine to minimize func.
NOTE: func must take the form func(b,varargin)
where: b = parameter vector (k x 1)
varargin = arguments passed to the function

Example:
res = ols2(y,x); result2 = pow_min('to_llike', [res('beta'); res('sige')],'maxit=1000',y,x);

This example, taken from optim1_d, realises the estimation of a tobit model, with initial values taken from on ols estimation. Maximum # of iterations is set to 1000.

stepz_____Step size in maximisation programs

CALLING SEQUENCE

[alpha]=stepz(xarg,chf,infoz,stat,varargin)

PARAMETERS

INPUT:

* xarg = vector of model parameters
* chf = a string that represents a call to a function with all useful arguments; the first one must be called xarg; the other ones if they exists must be called varargin(:)
* infoz = tlist with settings for maxlik
* stat = tlist with minimization status
* varargin = Variable list of arguments passed to func

OUTPUT:
alpha = scalar step size

DESCRIPTION
Determines step size in NUMZ package.
REFERENCES: Numerical Recipes in FORTRAN (LNSRCH, p. 378)

Example:

```
grocer_infoz=tlist(['options';'func';'maxit';'hess';'optprt';'cond';'btol';'ftol';'gtol';'lambda';'H1';'delta'
;'call';'step';'grad';'dirtol'],grocer_func,100,'bfgs',1,1000,sqrt(%eps),sqrt(%eps),sqrt(%eps),.01,1,.0
00001,'other','stepz','numz0',1);
grocer_stat=tlist(['options';'iter';'Hi';'df';'db';'dG';'f';'G';'star';'Hcond';'direc'],0,[],1000,ones(grocer
_k,1)*1000,ones(grocer_k,1)*1000,grocer_f,grocer_G,' ',0,[]);
grocer_chf=grocer_func+"(grocer_xarg,grocer_infoz,grocer_stat,varargin(:))'
grocer_chs=grocer_infoz('step')+'(grocer_xarg,grocer_chf,grocer_infoz,grocer_stat,varargin(:))';
execstr(grocer_chs);
```

This example shows how stepz is called from maxlik. Grocer_xarg and varargin are arguments passed to maxlik.