Reverse Communication Interface

Marcin Krzysztofik [1]

Reverse communication is a means of avoiding procedure arguments in the parameter list of a procedure. Most numerical routines use the alternative, forward (or direct) communication approach, i.e. they are called only once to compute results; they completely specify the problem by including user-provided procedures in the argument list. In the majority of applications this is the easiest and most convenient way to solve numerical problems. In a forward communication routine to minimize a function the user supplies the objective function and/or nonlinear constraints as a procedure, or a set of procedures, which is evaluated for any values of its arguments. This procedure will have a strict format of how the information might be supplied and this can lead to some restrictions. For example the specification of the user-provided procedure might not allow the user to pass some useful information. Alternatively the routine might be called in a mixed language environment and this too can lead to difficulties passing a procedure argument.

A way to get around these problems is to use a reverse communication routine. We can think of it as advancing a numerical algorithm one-step with each call before returning to the user for fresh information or with a completed solution. Instead of calling an algorithm once with all of the information provided, the user calls it several times, each time checking what new information is required. Such routines have a parameter (a flag) which returns a value after each call of the routine. This flag determines whether the solution has been found and algorithm can be stopped, or whether some additional information needs to be supplied by the user and the routine re-entered. A reverse communication interface demands more effort from the user, who is responsible for checking whether a solution has been obtained and, if not, for supplying the requested information and then re-entering the routine. On the other hand it gives more flexibility and allows more complicated data structures to be handled.

The other difference between forward and reverse approaches is in memory access [2]. Since a reverse communication routine does not need to take all input data at once, it can reduce the amount of memory used comparing with forward communication.

Most algorithms in the NAG Fortran Library [5] use the forward com-

[1] Marcin.Krzysztofik@nag.co.uk

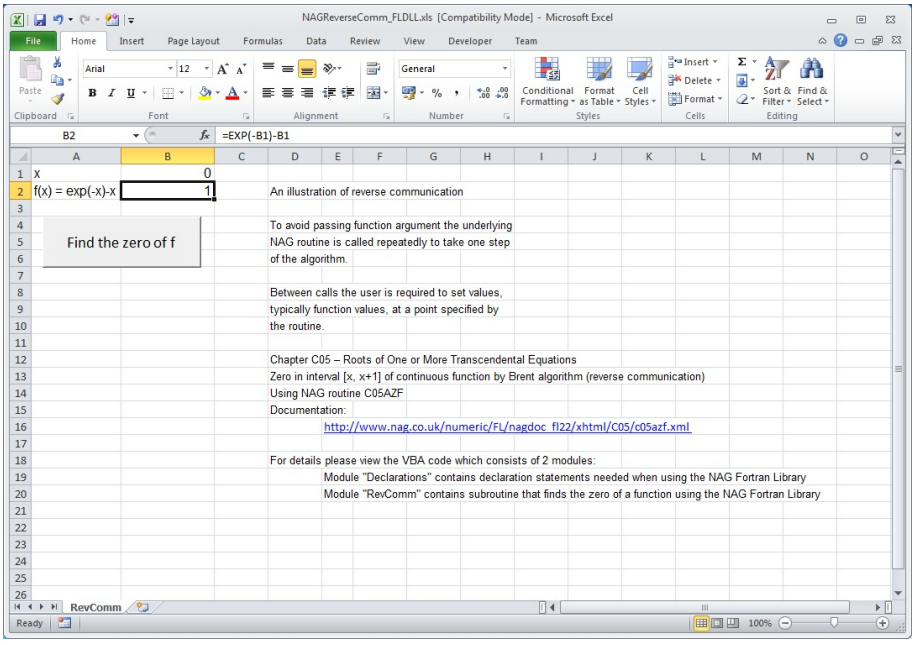

Figure 1: NAG routine C05AZF example program in Excel.

munication which is easy to use and sufficient for the majority of problems. There are, however, several routines using reverse communication to address the needs of more advanced users. They can be used in the following areas:

- Finding roots of equations;
- Solving ordinary differential equations;
- Optimizing a multivariate function;
- Solving large scale eigenproblems.

In order to demonstrate reverse communication let us have a look at an example using NAG routine C05AZF [1] to find a zero of function $f(x) = \exp(-x) - x$ in a given interval using Brent's algorithm. The NAG Library routine is being called from Microsoft Excel. It is coded using Visual Basic for Applications (VBA). $f(x)$ is specified on the worksheet, it does not need to be coded in VBA, because the VBA routine accesses the values of the function from the worksheet. For this example the user also supplies the value of x — the lower bound of the interval.

The Excel workbook is called *NAGReverseComm_FLDLL.xls* and is available for download from *NAG and Excel* [6] webpage. An alternative version

of the workbook (*NAGReverseComm_CLDLL.xls*), which uses the NAG C Library [4] is also available. Both workbooks run with 32 and 64 bit Excel.

Please contact NAG at support@nag.co.uk regarding trial licenses for NAG Library or any other questions. Please also contact NAG with requests for other numerical routines that you would like implemented in a reverse communication form.

References

[1] NAG Library Routine Document: C05AZF. http://www.nag.co.uk/numeric/FL/nagdoc_fl22/pdf/C05/c05azf.pdf.

[2] Mike Dewar. Reverse Communication. http://thenumericalalgorithmsgroup.blogspot.com/2010/01/reverse-communication.html, January 2010.

[3] Jack Dongarra, Victor Eijkhout, and Ajay Kalhan. Reverse Communication Interface for Linear Algebra Templates for Iterative Methods. http://www.netlib.org/lapack/lawnspdf/lawn99.pdf, May 1995.

[4] NAG C Library. http://www.nag.co.uk/numeric/CL/CLdescription.asp.

[5] NAG Fortran Library. http://www.nag.co.uk/numeric/fl/FLdescription.asp.

[6] NAG and Microsoft Excel – Expand your capabilities. http://www.nag.co.uk/numeric/nagandexcel.asp.