LEVEL package

Dimitri Breda[1] Stefano Maset[2] Rossana Vermiglio[1]

[1]*Dipartimento di Matematica e Informatica*
Università degli Studi di Udine
via delle Scienze 208, I-33100 Udine, Italy
e-mail: {dimitri.breda,rossana.vermiglio}@dimi.uniud.it

[2]*Dipartimento di Matematica e Informatica*
Università degli Studi di Trieste
Via Valerio 12, I-34127 Trieste, Italy
e-mail: maset@units.it

Abstract

This manuscript represents a basic guide to the use of the Matlab package LEVEL, a software described in [2], of which this manual consitutes a supplement. The reader is thus referred to that paper for more detailed information.

Contents

1 Installation

LEVEL is a stand-alone Matlab package freely available from Netlib at www.netlib.org/numeralgo/na27. A user needs only standard Matlab, version 7.X (no particular problems were detected with version 6.X and additional

toolboxes are not required). A step-by-step installation for Unix environment follows:

1. download the gzipped tar archive `na27.tgz` in the directory you use for users Matlab packages and toolboxes. Matlab provides a directory usually named `usr/local/matlab/toolbox/` but only an administrator user can write on it. Also the whole contents of this directory is erased when a new Matlab version is installed. Hence, we suggest to create a users directory named e.g. `$HOME/matlab/toolbox/` for installing additional packages;

2. execute

   ```
   cd HOME/matlab/toolbox
   ```

   and

   ```
   tar xvfz na27.tgz
   ```

   which will create the directory LEVEL containing the package;

3. optionally remove the archive by

   ```
   rm na27.tgz
   ```

4. start Matlab;

5. add the path to the package directory, e.g. through the Matlab menu File ← "Set Path";

6. make sure the path is placed at the bottom of Matlab search path by clicking "Move to Bottom";

7. save the changed path.

Alternatively, the directory LEVEL can be saved everywhere and the software can be run in the same directory.

For Windows environment, substitute steps 1, 2 and 3 above by standard download and unzip procedures. For Mac OS X just download and double-click the archive. The remaining steps are the same.

2 List of files

The LEVEL package comes together with a series of m-files necessary to the user in order to perform various tests. According to [*], these are indicated in the sequel as Test 1,Test 2,Test 3,Test 4 and Test 5.

- accompanying files:
  - **manual.pdf**: description of installation and usage of LEVEL;
  - **contents.m**: description of the files contained in LEVEL;
  - **readme.txt**: copyright and authors' disclaim;
- main external m-files:
  - **level.m**: main function for approximating the set of curves corresponding to a single-valued level of a given surface;
  - **lcontour.m**: function using Matlab `contour` for comparison with `level` for a single-valued level of a given surface;
- main internal m-files:
  - **ref.m**: function performing the adaptive triangulation refinement on square cells as described in [*];
  - **adcur.m**: function performing the adaptive curvature technique described in [*];
  - **secant.m**: secant method for refining the level curves segments as described in [*];
- m-files for tests:
  - **test1.m**: script running Test 1, i.e. level curves of a constructed surface;
  - **test2.m**: script running Test 2, i.e. level curves of the Matlab surface `peaks`;
  - **test3.m**: script running Test 3, i.e. computation of a stability chart for a Delay Differential Equation (DDE);
  - **test4.m**: script running Test 4, i.e. computation of ϵ-pseudospectra for a first DDE;
  - **test5.m**: script running Test 5, i.e. computation of ϵ-pseudospectra for a second DDE;

- **fval1.m**: function evaluating the relevant surface for Test 1;
- **fval2.m**: function evaluating the relevant surface for Test 2;
- **fval3.m**: function evaluating the relevant surface for Test 3;
- **fval4.m**: function evaluating the relevant surface for Test 4;
- **fval5.m**: function evaluating the relevant surface for Test 5;

- additional internal m-files ONLY for tests:
  - **model.m**: internal function in `fval3` providing the coefficients and delays of the DDE for Test 3;
  - **root.m**: internal function in `fval3` approximating the characteristic roots of the DDE provided in `model` (see [1]);
  - **spig.m**: internal function in `fval3` providing the pseudospectral discretization of the infinitesimal generator associated to the DDE provided in `model` (see [1]);
  - **cheb.m**: internal function in `root` providing the Chebyshev nodes and differentiation matrix for the pseudospectral discretization performed in `spig` (see [3]);
  - **elle.m**: internal function in `fval3` providing the evaluation of the Lagrange coefficients for the pseudospectral discretization performed in `spig`;
  - **desort.m**: internal function in `fval3` ordering the complex roots produced by `spig` according to their decreasing real part;
  - **ord.m**: internal function in `fval3` ordering the discrete delays and their relevant coefficients for the DDE of Test 3 provided by `model`;
  - **multilevel.m**: main function for approximating the set of curves corresponding to a multiple-valued level of a given surface - used in Test 4 and Test 5 for ϵ-pseudospectra computation (see [*]);
  - **multicontour.m**: function using Matlab `contour` for comparison with `multilevel` for a multiple-valued level - used in Test 4 and Test 5 for ϵ-pseudospectra computation (see [*]) and comparison with `multilevel`.

Remark 1 *Observe that the same algorithms used in* `root` *and its associated functions* `spig`, `cheb`, `elle` *and* `desort` *are also utilized in* `multilevel` *and* `multicontour` *for the DDEs of Test 4 and Test 5. For the numerical methods underlying those algorithms see [1] and [3].*

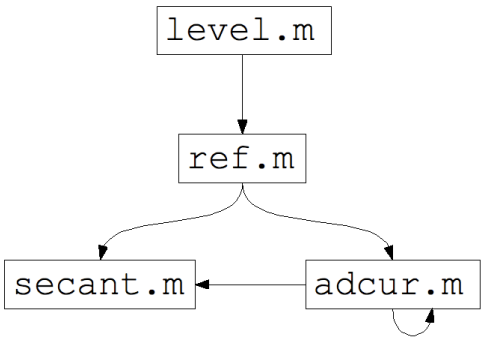

Figure 1: hierarchy of function calls.

3 How to use LEVEL

The LEVEL package is basically made of the main external function `level`, which constitute the user interface. The other main external function `lcontour` is provided just to let the user compare in a simple way with the Matlab `contour` plot algorithm. All other functions reported in the list in the previous section are called internally by the above ones and hence the user is not asked to use them directly.

1. the function **level.m** allows to plot the set of curves of a given level z for a given surface function f in the rectangular region provided through the 4-dimensional vector r containing the (x, y)-extrema coordinates $[r_1, r_2] \times [r_3, r_4]$. A overall tolerance TOL must be given in input and the output is made of the total number of evaluations of f required to draw the level curves, the total CPU time and the plot of the curves. Additional optional tolerance parameters may be provided as variable input as descried by calling for `help level` according to the detailed explanation given in [*].
   Algorithm: the set of z-level curves in a given rectangle of the (x, y)-plane is obtained by evaluating f on a first coarse square grid. Each square cell is given to the internal function `ref.m` performing, when required, an adaptive triangulation which locate the portions of regions where segments of the level curves lie into triangular cells of a minimum size proportional to the desired tolerance given in input. Once this size is reached, a final refinement is performed through the internal function `adcur.m` which executes an adaptive curvature strategy combined with iterations based on the secant method through the other internal function `secant.m` (see Figure 1 for the hierarchy of calls).

2. the function **lcontour.m** allows to plot the same set of level curves and it is

based on the same inputs and outputs as for `level.m` above.
Algorithm: the set of z-level curves in a given rectangle of the (x, y)-plane is obtained by using the Matlab `contour` plot algorithm.

4 Tests

In order to provide a first understanding on how to use the LEVEL package, a set of script files is provided. These files automatically execute the corresponding Test 1, Test 2, Test 3, Test 4 and Test 5 in [*] and produce the relevant outputs such as the figures shown in [*] and the data collected in the relevant tables.

5 Concluding remarks and tips

The main external and test files are suitably commented. The main comments can be seen through the standard Matlab function `help`. Additional comments are provided internally in order to follow the structure of the main algorithms.

The user is recommended to carefully read [*] before attempting to use the LEVEL package, especially the parts relevant to the setting of the concerned (compulsory and optional) tolerance parameters. As a general advise, consider that the human eye is in general able to distinguish in a Matlab figure (of standard size) details of size ranging approximately from $1/100$ to $1/1000$ of the figure size (for instance, the standard grid adopted in `contour` is only 60×60). Therefore, consider that asking for a final accuracy much less than this proportion translates into a considerable increasing of computational efforts (CPU time, memory storage) which is totally useless since we basically cannot distinguish the improvements on the level curves location.

References

[1] D. Breda, S. Maset, and R. Vermiglio. Pseudospectral differencing methods for characteristic roots of delay differential equations. *SIAM J. Sci. Comput.*, 27(2):482–495, 2005.

[2] D. Breda, S. Maset, and R. Vermiglio. An adaptive algorithm for efficient computation of level curves of surfaces. *Numer. Algo.*, 52(4):605–628, 2009.

[3] L. N. Trefethen. *Spectral methods in MATLAB*. Software - Environment - Tools series. SIAM, Philadelphia, USA, 2000.