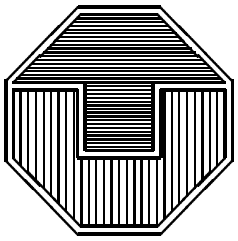

Dresden University of Technology

Institute for Fluid Mechanics

`LASPack` Reference Manual[1][2]

(version 1.12.2)

Tomáš Skalický

August 13, 1995

[1]The development of `LASPack` was supported in part by the German Bundesministerium für Forschung und Technologie under contract number 0329016D.

[2]A HTML version of the manual is available at `http://www.tu-dresden.de/mwism/skalicky-/laspack/laspack.html`.

Author's address:

Tomáš Skalický
Dresden University of Technology
Institute for Fluid Mechanics
Mommsenstraße 13
D-01062 Dresden
Germany
e-mail: skalicky@msmfs1.mw.tu-dresden.de
http://www.tu-dresden.de/mwism/skalicky/home.html

Contents

---

1 Introduction

`LASPack` is a package for solving large sparse systems of linear equations like those which arise from discretization of partial differential equations.

Main features:

- The primary aim of `LASPack` is the implementation of efficient iterative methods for the solution of systems of linear equations. All routines and data structures are optimized for effective usage of resources especially with regard to large sparse matrices. The package can be accessed from an application through a straightforward interface defined in the form of procedure calls.
- Beside the obligatory Jacobi, succesive over-relaxation, Chebyshev, and conjugate gradient solvers, `LASPack` contains selected state-of-the-art algorithms which are commonly used for large sparse systems:

  - CG-like methods for non-symmetric systems: CGN, GMRES, BiCG, QMR, CGS, and BiCGStab,
  - multilevel methods such as multigrid and conjugate gradient method preconditioned by multigrid and BPX preconditioners.

  All above solvers are applicable not only to the positive definite or non-symmetric matrices, but are also adopted for singular systems (e.g. arising from discretization of Neumann boundary value problems).

- The implementation is based on an object-oriented approach (although it has been programmed in C). Vectors and matrices are defined as new data types in connection with the corresponding supporting routines. The basic operations are implemented so that they allow the programming of linear algebra algorithms in a natural way.

- `LASPack` is extensible in a simple manner. An access to the internal representation of vectors and matrices is not necessary and is, as required of the object-oriented programming, avoided. This allows an improvement of algorithms or a modification of data structures with no adjustment of application programs using the package.

- `LASPack` is written in ANSI C and is thus largely portable.

2 What's New

2.1 Changes in Version 1.12

- Fixed some bugs.

- Thoroughly tested all solvers in module **ITERSOLV** for symmetric systems (by Marc Niemann).

- Generated the HTML version of the manual.

2.2 Changes in Version 1.11

- Fixed some bugs.

- Changed the parameter lists of solvers in modules **ITERSOLV** and **MLSOLV** as well as of preconditions in modules **PRECOND**.

- Extended the structure of the type `QMatrix` for storage of null space information for singular matrices.

- Adopted the iterative solvers to systems of equations with a singular matrix.

- Corrected the implementation of the Chebyshev method.

- Extended the estimation of eigenvalues to preconditioned matrices.

- Revised the `LASPack` error handling.

- Introduced the symbolic names for variables of the types `Vector`, `Matrix`, and `QMatrix`.
- Thoroughly revised the manual.

2.3 Changes in Version 1.10

- Fixed some bugs.
- Implemented the BPX Preconditioned Conjugate Gradient method. Renamed the module **MGSOLV** and the program `mgtest` in **MLSOLV** and `mlstest`, respectively, because of their true multilevel character.
- Improved the estimation of extremal eigenvalues by the Lanczos method.
- Accelerated the iterative solvers for zero initial solution (which arises often within the multigrid algorithm).
- Changed the defaults for accuracy in modules **RTC** and **EIGENVAL** to 10^{-8} and 10^{-4}, respectively.
- Extended the structure of type `QMatrix`, e.g. by reciprocal values of diagonal elements which are needed in classical solvers (Jacobi, SOR method) are stored as auxiliary variables.
- Implemented the procedure `V_SetRndCmp` in module **VECTOR** which initializes vector components by random values.
- Modified the stopping criterion for GMRES.
- Changed the names of some procedures and variables.

3 Downloading

The source code and the documentation of `LASPack` is available in World Wide Web at the following URLs:

- the distribution file

  `http://www.tu-dresden.de/mwism/skalicky/laspack-1.12.2.tar.Z`
  `ftp://netlib.att.com/netlib/linalg/laspack-1.12.2.tar.Z`
  or at other netlib sites:
  - `netlib.att.com` (Murray Hill, NJ, USA)
  - `ftp.netlib.org` (Oak Ridge, TN, USA)
  - `netlib.no` (Oslo, Norway)
  - `unix.hensa.ac.uk` (Lancaster, UK)
  - `elib.zib-berlin.de` (Berlin, Germany)

  - `nchc.edu.tw` (Taiwan)
  - `ftp.cs.uow.edu.au` (Wollongong, Australia)

- this manual as HTML document

  `http://www.tu-dresden.de/mwism/skalicky/laspack/laspack.html`

- the postscript version of the manual

  `http://www.tu-dresden.de/mwism/skalicky/laspack1.ps.Z`
  `http://www.tu-dresden.de/mwism/skalicky/laspack2.ps.Z`

You may also contact the author at

Dresden University of Technology
Institute for Fluid Mechanics
Mommsenstraße 13
D-01062 Dresden
Germany

or by e-mail under `skalicky@msmfs1.mw.tu-dresden.de`.

4 Installation

In order to simplify the description of the installation procedure, we assume that you are working on a UNIX system and have already copied the `LASPack` distribution file `laspack-1.12.2.tar.Z`.

The complete installation consists of following five steps:

1. Move the file `laspack-1.12.2.tar.Z` to the directory that will become the top-level directory for the source files.

2. Uncompress and untar the file:

   ```
   uncompress laspack-1.12.2.tar.Z
   tar xvf laspack-1.12.2.tar
   ```

   This should create the entire distribution tree.

3. Make sure that you have appropriate set the following variables of your UNIX environment (for compilation of the source files an ANSI C compiler is needed):

`HOME`	home directory
`CC`	name of the C compiler
`CFLAGS`	compiler options
`LDFLAGS`	linker options

4. There are two possibilities for installing the package:

User's installation: This should be carried out if you want to install `LASPack` in your home directory so that it is available only to you.

At this stage, you could use the environment variable `ARCH_EXT` in order to install the library and the test programs in different subdirectories depending on computer architecture. It may be advantageous, if you share your home directory across a heterogeneous computer network, for the management of several versions of binaries. In this case, set the variable `ARCH_EXT` to an appropriate value (e.g. `/sunos` on Sun workstations, `/hp-ux` on HP workstations, etc.) and make sure that the directory `$HOME/bin$ARCH_EXT` is contained in your `PATH` variable.

Run the installation script:

```
./install
```

This will generate the library `liblaspack.a` (and the library `libxc.a` which is no a part of `LASPack` but required by some test programs) and install it in the directory `$HOME/lib$ARCH_EXT`. Furthermore, it will build the test programs and install them in the directory `$HOME/lib$ARCH_EXT`.

Local installation: This will install `LASPack` library and corresponding header files in the directory `/usr/local/lib` and the test programs in the directory `/usr/local/bin` which are usually used for such kind of software. To do this you need permission to write in these directories.

The installation script should be started with an additional parameter:

```
./install local
```

5. Finally, running the installation script, the `LASPack` library created is automatically checked. By means of the program `mlstest` described in section 5.4, all combinations of multilevel solvers, plain iterative procedures, and preconditioners available in `LASPack` are tested on a two-dimensional Poisson problem. If all goes well, you get the message:

   ```
   LASPack tested successfully.
   ```

   Therewith `LASPack` installation is completed.

   In order to run test programs in a C shell, do not forget to renew the hash tables by the command:

   ```
   rehash
   ```

`LASPack` as well as the installation script was successfully tested on several machines: Sun Sparc5, HP 9000/735, IBM RS/6000 550, DEC 3000/800 M, SGI IRIS Indigo and PC 486 (running Linux). Nevertheless, if the installation fails, please contact the author.

5 Optimizing and Example Programs

In this section, the programs `lastest`, `vectopt`, `matropt`, and `msltest` are described.

The first three programs have been developed in order to test and optimize selected operations in the LASPack library. Their syntax

```
lastest [-d<dim>] [-c<cycles>] [-h]
vectopt [-d<dim>] [-c<cycles>] [-h]
matropt [-d<dim>] [-c<cycles>] [-h]
```

and options

```
-d  set vector and matrix dimensions to <dim>, default is 10000
-c  set number of cycles to <cycles>, default is 10
-h  print the help
```

are the same. The dimensions as well as the number of cycles should be chosen high enough in order to avoid undesired caching effects and inaccuracy of time determination.

5.1 Program lastest

The program lastest shown efficiency of basic operations implemented in LASPack. The output contains the following data:

```
Results:    (for dimension 100000, cycles 100)
--------
```

#	with LASPack	in-line code	efficiency	description
1	---	---	---	vector constr. & destr.
2	0.00900 s	---	---	vector generation
3	0.06316 s	---	---	matrix constr. & destr.
4	1.32628 s	---	---	matrix generation
5	0.02283 s	0.02283 s	100.00 %	a = b
6	0.04033 s	0.03850 s	95.45 %	a = b + c
7	0.02283 s	0.02283 s	100.00 %	a = s * b
8	0.04316 s	0.04333 s	100.39 %	s = b * c
9	0.26699 s	0.27116 s	101.56 %	a = L * b
10	0.03000 s	0.02983 s	99.44 %	a = a + b
11	0.01933 s	0.01967 s	101.72 %	a = s * a
12	0.04250 s	0.04183 s	98.43 %	a = b + s * c
13	0.03267 s	0.03300 s	101.02 %	a = a + s * b

Here s is a scalar, a, b, c are vectors, and L is a matrix. The in-line routines are in comparison with LASPack simplified and can be therefore a little faster.

5.2 Program vectopt

The program vectopt could be applied to check efficiency of several implementations of the vector operation:

```
  <Vector 1> += <Vector 2>.
```

It produces results like these:

```
Results:    (for dimension 1000000, cycles 100)
--------

 implementation    CPU time
-----------------------------------------
    LASPack       2.518e-01 s = 100.0 %
       1          3.548e-01 s = 140.9 %
       2          3.196e-01 s = 126.9 %
       3          2.481e-01 s =  98.5 %
       4          2.392e-01 s =  95.0 %
       5          2.395e-01 s =  95.1 %
-----------------------------------------
```

For details to the implementations look at the file `laspack/examples/vectopt/testproc.c`. The current `LASPack` version corresponds to the implementation 2 and 3 depending on the computer architecture, respectively. They are, in comparison with `LASPack`, simplified and can be therefore a little faster.

5.3 Program `matropt`

This program `matropt` could be used to check efficiency of several implementations of the matrix by vector product:

```
  <Vector 2> = <Matrix> <Vector 1>.
```

The results may shown as follows:

```
Results:    (for dimension 100000, cycles 100)
--------

 implementation    CPU time
-----------------------------------------
    LASPack       1.967e-01 s = 100.0 %
       1          2.836e-01 s = 144.2 %
       2          2.326e-01 s = 118.3 %
       3          1.928e-01 s =  98.0 %
       4          2.005e-01 s = 101.9 %
-----------------------------------------
```

For details to the implementations look at the file `laspack/examples/matropt/testproc.c`. The current `LASPack` version corresponds to the implementation 3. This is, in comparison with `LASPack`, simplified and can be therefore a little faster.

5.4 Program `mlstest`

The program `mlstest` is an extensive example code demonstrating possibilities of usage of `LASPack` routines. It solves a Poisson problem

$$-\triangle u = 1$$

in a 1D or 2D unit domain by a specified multilevel solver.

All input parameters have to be set interactively. The output contains the convergence history as well as the determined middle convergence rates.

For example, a three-grid solution by means of the V-cycle multigrid method with the Gauss-Seidel smoothing (two pre-smoothing and one post-smoothing iteration) and the ILU preconditioned CG method on the coarsest grid (20 iterations) yields for a 65×65 grid on the finest level:

```
  0. iteration ... accuracy =  1.0000e+00
  1. iteration ... accuracy =  4.3033e-02
  2. iteration ... accuracy =  3.3534e-03
  3. iteration ... accuracy =  2.8452e-04
  4. iteration ... accuracy =  2.4934e-05
  5. iteration ... accuracy =  2.2565e-06
  6. iteration ... accuracy =  2.0996e-07
  7. iteration ... accuracy =  1.9860e-08
  8. iteration ... accuracy =  7.6893e-09

  CPU time:    0.67 s

Middle contraction rate
  referred to one iteration:  9.677e-02
  referred to 1 s CPU time:   7.751e-13
```

The input data corresponds in this case to: `2`, `3`, `64`, `1`, `n`, `1`, `2`, `2`, `2`, `1`, `0`, `1.0`, `6`, `20`, `3`, `1.0`, `100`, `1e-8`.

6 Copyright

3. All modifications to the source code must be clearly marked as such. Binary redistributions based on modified source code must be clearly marked as modified versions in the documentation and/or other materials provided with the distribution.

4. No fees may be charged for distribution of the codes, other than a fee to cover the cost of the media and a reasonable handling fee.

5. This code, and any derivative of this code, may not be used in a commercial package without the prior explicit written permission of the author.

This code is provided "as is", without any warranty of any kind, either expressed or implied, including but not limited to, any implied warranty of merchantibility or fitness for any purpose. In no event will the author or any party who distributed the code be liable for damages or for any claim(s) by any other party, including but not limited to, any lost profits, lost data or data rendered inaccurate, losses sustained by third parties, or any other special, incidental or consequential damages arising out of the use or inability to use the program, even if the possibility of such damages has been advised against. The entire risk as to the quality, the performance, and the fitness of the program for any particular purpose lies with the party using the code.

ANY USE OF THIS CODE CONSTITUTES ACCEPTANCE OF THE TERMS OF THE COPYRIGHT NOTICE

7 Acknowledgement

The author would like to thank Markus Rösler, who encouraged the development of this package and was the first one to use it. He also wishes to acknowledge Andreas Auge, Thomas Biesinger and Daniel Spirn for many helpful discussions concerning implementation details and suggestions for improvement to this manual.

8 Manual Pages, Index