NLEVP: A Collection of Nonlinear Eigenvalue Problems. Users' Guide

Betcke, Timo and Higham, Nicholas J. and Mehrmann, Volker and Schröder, Christian and Tisseur, Françoise

2010

MIMS EPrint: **2010.99**

ISSN 1749-9097

NLEVP: A Collection of Nonlinear Eigenvalue Problems. Users' Guide.

Timo Betcke[*] Nicholas J. Higham[†] Volker Mehrmann[‡]
Christian Schröder[‡] Françoise Tisseur[†]

November 15, 2010

Abstract

This is the Users' Guide for NLEVP: a collection of nonlinear eigenvalue problems provided in the form of a MATLAB toolbox. A separate paper describes the collection and its organization.

1 Introduction

This document describes how to install and use the NLEVP MATLAB toolbox, which provides a collection of nonlinear eigenvalue problems.

For details of the design and organization of the collection, and the problems it contains, see [1].

This document describes Version 2.0 of the toolbox. The collection will grow and contributions are welcome (see Section 6).

2 Installation and Usage

The collection is available from

`http://www.mims.manchester.ac.uk/research/numerical-analysis/nlevp.html`

It is provided as both a zip file and a tar file. To install the toolbox create the directory `nlevp` in a suitable location and make it the current directory. Download `nlevp.zip` or `nlevp.tar` into this directory. Then use appropriate "unzip" software (making sure to preserve the directory structure) or type `tar xvf nlevp.tar`. This creates the subdirectory `private`. Put the `nlevp` directory on the MATLAB path, which can be done using the `addpath` command (ideally in `startup.m`).

To try the toolbox, run the demonstration script by typing `nlevp_example` at the MATLAB command prompt. Then execute the following commands:

```
help nlevp
nlevp query problems
nlevp query properties

nlevp help railtrack
```

[*]Department of Mathematics and Statistics, University of Reading, UK (`t.betcke@reading.ac.uk`). The work of this author was supported by Engineering and Physical Sciences Research Council Grant EP/H004009/1.

[†]School of Mathematics, The University of Manchester, Manchester, M13 9PL, UK (`{higham,ftisseur}@ma.man.ac.uk`). This work was supported by Engineering and Physical Sciences Research Council grants EP/D079403/1 and EP/E050441/1 (CICADA: Centre for Interdisciplinary Computational and Dynamical Analysis). The work of the second author was also supported by a Royal Society-Wolfson Research Merit Award and the work of the fifth author was also supported by a Leverhulme Research Fellowship.

[‡]Institut für Mathematik, MA 4-5, Technische Universität Berlin, Germany (`{mehrmann,schroed}@math.tu-berlin.de`). Supported by Deutsche Forschungsgemeinschaft through Matheon, the DFG Research Center *Mathematics for key technologies* in Berlin.

```
nlevp query railtrack
coeffs = nlevp('railtrack')
spy(coeffs{2})

coeffs = nlevp('bicycle')
polyeig(coeffs{:})
```

The collection has been tested in MATLAB 7.1 (R14) up to R2010b. It does not work with versions 6.5 (R13) and earlier of MATLAB, since it uses functionality introduced in MATLAB 7.0 (R14).

3 Release History

The first release of the toolbox was version 1.0, 4-Apr-2008, and contained 26 problems. The second release, version 2.0, is dated 15-Nov-2010, contains 46 problems, and has the following changes:

- Problem **string** has been renamed **spring**. **spring** has been generalized to include more parameters but is backward compatible with **string**. Invoking `nlevp('string')` still works: it invokes `nlevp('spring')` and produces a warning message.
- The matrices generated by problems **Acoustic wave 1D** and **Acoustic wave 2D** have been modified in order to more closely match the formulation in the paper from which this problem is taken. The eigenvalues now lie in the upper half-plane instead of the left half-plane.
- New problems are: **fiber**, **foundation**, **genhyper2**, **Hadeler**, **intersection**, **metal strip**, **PDDE stability**, **plasma drift**, **omnicam1**, **omnicam2**, **qep1**, **qep2**, **qep3**, **qep4**, **railtrack2**, **relative pose 5pt**, **relative pose 6pt**, **shaft**, **speaker box**, **surveillance**.
- New functionality: `nlevp('eval',...)` and `[coeffs,fun] = nlevp('name',...)`.
- Automatic testing of problem properties via `nlevp_test`.
- Cosmetic changes have been made to some of the functions.
- Citations to the sources of the problems have been updated, where necessary.

4 The MATLAB Function nlevp

The toolbox has just one main user-callable function, `nlevp`, which is as follows.

```
function varargout = nlevp(name,varargin)
%NLEVP   Collection of nonlinear eigenvalue problems.
%  [COEFFS,FUN,OUT3,OUT4,...] = NLEVP(NAME, ARG1, ARG2,...)
%    generates the matrices and functions defining the problem specified by
%    NAME (a string). ARG1, ARG2,... are problem-specific input arguments.
%    All problems are of the form
%      T(lambda)*x = 0
%    where
%      T(lambda)= f0(lambda)*A0 + f1(lambda)*A1 + ... + fk(lambda)*Ak.
%    The matrices A0, A1, ..., Ak are returned in a cell array:
%    COEFFS = {A0,...,Ak}.
%    FUN is a function handle that can be used to evaluate the functions
%    f1(lambda),...,fk(lambda).  For a scalar lambda,
%    F = FUN(lambda) returns a row vector containing
%      F = [f1(lambda), f2(lambda), ..., fk(lambda)].
%    If lambda is a column vector, FUN(lambda) returns a row per element in
%    lambda.
%    [F,FP] = FUN(lambda) also returns the derivatives
%      FP = [f1'(lambda), f2'(lambda), ..., fk'(lambda)].
%    [F,FP,FPP,FPPP,...] = FUN(lambda) also returns higher derivatives.
%    OUT3, OUT4, ... are additional problem-specific output arguments.
%    See the list below for the available problems.
%
%  PROBLEMS = NLEVP('query','problems') or NLEVP QUERY PROBLEMS
%    returns a cell array containing the names of all problems
%    in the collection.
%  NLEVP('help','name') or NLEVP HELP NAME
%    gives additional information on problem NAME, including number and
%    meaning of input/output arguments.
%  NLEVP('query','name') or NLEVP QUERY NAME
%    returns a cell array containing the properties of the problem NAME.
%  PROPERTIES = NLEVP('query','properties') or NLEVP QUERY PROPERTIES
%    returns the properties used to classify problems in the collection.
%  NLEVP('query',property1,property2,...) or NLEVP QUERY PROPERTY1 ...
%    lists the names of all problems having all the specified properties.
%
%  [T,TP,TPP,...] = NLEVP('eval', NAME, LAMBDA, ARG1, ARG2, ...)
%  evaluates the matrix function T and its derivatives TP, TPP,...
%  for problem NAME at the scalar LAMBDA.
%
%  NLEVP('version') or NLEVP VERSION
%    prints version, release date, and number of problems
%    of the installed NLEVP collection.
%  V = NLEVP('version')
%    returns a structure V containing version information.
%    V consists of the fields v.number, v.date, and v.problemcount.
%
%  Available problems:
%
%  acoustic_wave_1d   Acoustic wave problem in 1 dimension.
%  acoustic_wave_2d   Acoustic wave problem in 2 dimensions.
%  bicycle            2-by-2 QEP from the Whipple bicycle model.
%  bilby              5-by-5 QEP from Bilby population model.
%  butterfly          Quartic matrix polynomial with T-even structure.
%  cd_player          QEP from model of CD player.
%  closed_loop        2-by-2 QEP associated with closed-loop control system.
%  concrete           Sparse QEP from model of a concrete structure.
%  damped_beam        QEP from simply supported beam damped in the middle.
%  dirac              QEP from Dirac operator.
%  fiber              NEP from fiber optic design.
%  foundation         Sparse QEP from model of machine foundations.
%  gen_hyper2         Hyperbolic QEP constructed from prescribed eigenpairs.
%  gun                NEP from model of a radio-frequency gun cavity.
%  hadeler            NEP due to Hadeler.
%  hospital           QEP from model of Los Angeles Hospital building.
%  intersection       10-by-10 QEP from intersection of three surfaces.
%  loaded_string      REP from finite element model of a loaded vibrating
%                     string.
%  metal_strip        QEP related to stability of electronic model of metal
%                     strip.
%  mobile_manipulator QEP from model of 2-dimensional 3-link mobile manipulator.
%  omnicam1           9-by-9 QEP from model of omnidirectional camera.
%  omnicam2           15-by-15 QEP from model of omnidirectional camera.
%  orr_sommerfeld     Quartic PEP arising from Orr-Sommerfeld equation.
%  pdde_stability     QEP from stability analysis of discretized PDDE.
%  plasma_drift       Cubic PEP arising in Tokamak reactor design.
%  power_plant        8-by-8 QEP from simplified nuclear power plant problem.
%  QEP1               3-by-3 QEP with known eigensystem.
%  QEP2               3-by-3 QEP with known, nontrivial Jordan structure.
%  QEP3               3-by-3 parametrized QEP with known eigensystem.
%  QEP4               3-by-4 QEP with known, nontrivial Jordan structure.
%  railtrack          QEP from study of vibration of rail tracks.
%  railtrack2         Palindromic QEP from model of rail tracks.
%  relative_pose_5pt  Cubic PEP from relative pose problem in computer vision.
%  relative_pose_6pt  QEP from relative pose problem in computer vision.
%  schrodinger        QEP from Schrodinger operator.
%  shaft              QEP from model of a shaft on bearing supports with a
%                     damper.
%  sign1              QEP from rank-1 perturbation of sign operator.
%  sign2              QEP from rank-1 perturbation of 2*sin(x) + sign(x)
%                     operator.
%  sleeper            QEP modelling a railtrack resting on sleepers.
%  speaker_box        QEP from finite element model of speaker box.
%  spring             QEP from finite element model of damped mass-spring
%                     system.
%  spring_dashpot     QEP from model of spring/dashpot configuration.
%  surveillance       27-by-20 QEP from surveillance camera callibration.
%  wing               3-by-3 QEP from analysis of oscillations of a wing in
%                     an airstream.
%  wiresaw1           Gyroscopic system from vibration analysis of wiresaw.
%  wiresaw2           QEP from vibration analysis of wiresaw with viscous
%                     damping effect.
%
%  Examples:
%  coeffs = nlevp('railtrack')
%    generates the matrices defining the railtrack problem.
%  nlevp('help','railtrack')
%    prints the help text of the railtrack problem.
%  nlevp('query','railtrack')
%    prints the properties of the railtrack problem.
%
%  For example code to solve all polynomial eigenvalue problems (PEPs)
%  in this collection of dimension < 500 with MATLAB's POLYEIG
%  see NLEVP_EXAMPLE.M.

%  Reference:
%  T. Betcke, N. J. Higham, V. Mehrmann, C. Schroeder, and F. Tisseur.
%  NLEVP: A Collection of Nonlinear Eigenvalue Problems,
%  MIMS EPrint 2010.98, Manchester Institute for Mathematical Sciences,
%  The University of Manchester, UK, 2010.

% Check inputs
if nargin < 1, error('Not enough input arguments'); end
if ~ischar(name), error('NAME must be a string'); end

name = lower(name);

if strcmp(name,'query') && nargin == 1
   error('Not enough input arguments')
end
```

```
if strcmp('string',name)
   name = 'spring';
   warning('NLEVP:string_renamed','Problem string has been renamed spring.')
end

if strcmp('version',name)
   name = 'nlevp_version';
end

switch name
    case 'help'
        if nargin < 2, help nlevp; return, end
        name = varargin{1};
        if ~ischar(name), error('NAME must be a string'); end
        eval(['help ', name]);
        return
    case 'eval'
        [varargout{1:max(nargout,1)}] = nlevp_eval(varargin{:});
    otherwise
        [varargout{1:nargout}] = feval(name,varargin{:});
end
```

5 The MATLAB Function nlevp_example

The toolbox contains a function `nlevp_example.m` that illustrates the use of `nlevp`. Running it provides a quick test that the toolbox is correctly installed. This function can be adapted in order to test the user's own methods on subsets of NLEVP problems.

```
function nlevp_example(fname)
%NLEVP_EXAMPLE  Run POLYEIG on PEP problems from NLEVP.
% NLEVP_EXAMPLE solves all the not-too-large PEP problems in NLEVP
% by POLYEIG, sending output to the screen.
% NLEVP_EXAMPLE(fname) directs partial output to the file named fname
% (intended for generating output for NLEVP paper).

if nargin == 0
   fid = 1;
else
   fid = fopen(fname,'w');
end

nmax = 500;
probs = nlevp('query','pep');
nprobs = length(probs);
nprobs_total = length(nlevp('query','problems'));
fprintf(fid,'NLEVP contains %2.0f problems in total,\n', nprobs_total);
fprintf(fid,'of which %2.0f are polynomial eigenvalue problems (PEPs).\n', nprobs);
fprintf(fid,'Run POLYEIG on the PEP problems of dimension at most %2.0f:\n\n',nmax);

fprintf(fid,'             Problem   Dim  Max and min magnitude of eigenvalues\n');
fprintf(fid,'             -------   ---  ------------------------------------\n');
m = ceil(nprobs/4);
j = 1;
for i=1:nprobs
   if fid ~= 1 && i == 9
      fprintf(fid,'               ...\n');
      fid_save = fid;
      fid = 1;  % Omit output from this point on when writing to file.
   end
   coeffs = nlevp(probs{i});
   [n, nc] = size(coeffs{1});
   if n >= nmax
      fprintf(fid,'%20s   %3.0f is a PEP but is too large for this test.\n', ...
              probs{i}, n);
   elseif n ~= nc
      fprintf(fid,'%20s   %3.0f is a PEP but is nonsquare.\n', probs{i}, n);
   else

      % POLYEIG will convert sparse input matrices to full.
      e = polyeig(coeffs{:});
      fprintf(fid,'%20s   %3.0f  %9.2e, %9.2e\n', ...
               probs{i}, n, max(abs(e)), min(abs(e)));
      subplot(m,4,j)
      plot(real(e), imag(e),'.')
      title(probs{i},'Interpreter','none')
      % Tweaks.
      if strcmp(probs{i},'sign1'), ylim([-1 1]*1.5), end
      if strcmp(probs{i},'damped_beam')
         title(['         ' probs{i}],'Interpreter','none')
      end
      if strcmp(probs{i},'relative_pose_6pt')
         title(['          ' probs{i}],'Interpreter','none')
      end
      if strcmp(probs{i},'speaker_box') || strcmp(probs{i},'intersection')
         title(['     ' probs{i}],'Interpreter','none')
      end
      j = j+1;
   end
end

if nargin > 0, fclose(fid_save); end
```

The output of the function is shown in [1].

6 Contributing to the Collection

Contributions of suggested new problems for the collection are welcome. They can be sent to any of the authors. The following rules should be followed when providing new problems.

Write a LaTeX file called `problem_name.tex`, where `problem_name` is the proposed name of your example, describing the problem. Here, `problem_name` should be a string in lower case without any spaces. The `tex` file should consist of a problem environment, with first line stating the relevant identifiers for the problem (these properties are listed by `nlevp query properties`, and are explained in the companion document [1]):

```
\begin{problem}{problem_phrase}{identifier1,identifier2,...}
This is a xxx-problem of dimension nnn.
It arises in ...
\end{problem}
```

Here, `problem_phrase` is a capitalized phrase possibly containing spaces. For example one problem in the collection has `problem_name = loaded_string` but `problem_phrase = Loaded string`.

Provide your citations in a `bib` file; one `bib` file suffices even if multiple `tex` files are provided.

Write an M-file generating the coefficients of the example called `problem_name.m`. Document the M-file in the leading comment lines with the most important information from the `tex` file. If the problem is parameter dependent, set default values for any parameters not specified when the function is called. If you need extra data files, their names should begin with `problem_name`, e.g., `problem_name.mat`.

To specify a polynomial problem the first output of the M-file should be a cell array containing the coefficient matrices starting with the constant term. Thus if the first output is called `coeffs` and you want to define a PEP $P(\lambda) = \sum_{i=0}^{k} \lambda^i A_i$, then `coeffs{1}`=$A_0$, `coeffs{2}`=$A_1$, ..., `coeffs{k+1}`=$A_k$.

The second output argument must be a function that computes derivatives of the nonlinear scalar functions in the definition of the problem; for a polynomial eigenvalue problem this is trivially provided by a line of the form

```
fun = @(lam) nlevp_monomials(lam,k);
```

Here, `nlevp_monomials.m` is a function provided with NLEVP in the `private` directory.

If a supposed solution is provided it should be returned in a structure `sol` with the following format:

`sol.eval`: an $m \times 1$ vector, where m eigenvalues are provided,

`sol.evec`: an $m \times n$ matrix, where column `j` is the eigenvector corresponding to `sol.eval(j)`.

If both left and right eigenvectors are known, they should be returned in `sol.evec_left` and `sol.evec_right`.

References

[1] T. Betcke, N. J. Higham, V. Mehrmann, C. Schröder, and F. Tisseur. NLEVP: A collection of nonlinear eigenvalue problems. MIMS EPrint 2010.98, Manchester Institute for Mathematical Sciences, The University of Manchester, UK, Nov. 2010. 25 pp.