MATSLEMN
A Matlab package for the numerical solution of Sturm-Liouville eigenvalue problems

May 6, 2009

MATSLEMN is a Matlab package implementing the sixth order modified Neumann method for Sturm-Liouville eigenvalue problems of the form

$$-(p(x)y(x)')' + q(x)y(x) = Ew(x)y(x), \tag{1}$$

with p and w strictly positive, and boundary conditions (at regular points) of the form

$$a_1 y(a) + a_2 p(a) y'(a) = 0, \quad b_1 y(a) + b_2 p(a) y'(a) = 0, \tag{2}$$

where a_1, a_2 are not both zero nor are b_1, b_2. For details on the algorithm used, we refer to the paper *Solution of Sturm-Liouville problems using modified Neumann schemes*. Here we briefly describe the usage of the routines included in MATSLEMN.

calculate_eigenvalues

```
[E,partition] = calculate_eigenvalues(problem,pmin,pmax,tol,indices)
```

INPUT:

- `problem`: structure containing information on the problem
  - `problem.pfun`: function handle to a function defining $p(x)$
  - `problem.qfun`: function handle to a function defining $q(x)$
  - `problem.wfun`: function handle to a function defining $w(x)$
  - `problem.a`
  - `problem.b`: endpoints of the integration interval
  - `problem.bcs`: vector with the coefficients $[a_1\ a_2\ b_1\ b_2]$ of the boundary conditions
  - `problem.partition`: optional, structure containing information on a partition (mesh)
- `pmin` and `pmax`
  - `indices=true`: eigenvalues are computed with index between `pmin` and `pmax`
  - `indices=false`: eigenvalues between `pmin` and `pmax` are computed
- `tol`: an (relative) input tolerance
- `indices`: boolean indicating if `pmin` and `pmax` are indices or energies

OUTPUT:

- `E`: structure containing information on the computed eigenvalues

- `E.eigenvalues`: vector containing the eigenvalue estimates
- `E.errors`: vector containing the corresponding error estimates
- `E.indices`: vector containing the indices of the computed eigenvalues
- `E.status`: vector. If `E.status(i)`$\neq 0$ some difficulties were detected in the computation of `E.eigenvalues(i)`

- `partition`: structure containing information on the mesh, can be passed as input to later calls to `calculate_eigenvalues` in order to avoid the recomputation of the mesh. A `partition` structure must also be passed to `calculate_eigenfunction`.
  - `partition.bp`: structure containing information on the basic mesh
    - `bp.xs`: vector of meshpoints
    - `bp.p0`: vector of constant approximations of 1/p(x). `bp.p0(i)` is the constant approximation of 1/p(x) over the first mesh interval.
    - `bp.q0`: vector of constant approximations of q(x).
    - `bp.w0`: vector of constant approximations of w(x).
    - `bp.p1`: vector of $\hat{P}_1$-values
    - `bp.q1`: vector of $\hat{Q}_1$-values
    - `bp.w1`: vector of $\hat{W}_1$-values
    - `bp.p2`: vector of $\hat{P}_2$-values
    - `bp.q2`: vector of $\hat{Q}_2$-values
    - `bp.w2`: vector of $\hat{W}_2$-values
    - `bp.inf_a`: true if a is infinite
    - `bp.inf_b`: true if b is infinite
  - `partition.rp`: structure containing information on the reference mesh. same fiels as for `partition.bp`.

calculate_eigenfunction

`[x,v,vp]=calculate_eigenfunction(partition,E)`

INPUT

- `partition`: information on the mesh = an output argument of `calculate_eigenvalues`
- `E`: the value of the eigenvalue

OUTPUT

- `x`: vector of x-values (= `partition.rp.xs`)
- `v`: vector of y-values evaluated in x
- `vp`: vector of y'-values evaluated in x

Examples

The package contains some examples/predefined problems, illustrating the usage of the different routines. The M-file `Collatz.m`, e.g., solves the Collatz problem

$$y'' + \frac{3}{4x^2}y = -\frac{1}{x^6}Ey \quad \text{in } [1,2],\ y(1) = y(2) = 0.$$

This problem can be solved in closed form. The solutions are

$$E_k = \frac{64}{9}k^2\pi^2, \quad y_k = \frac{3}{8k\pi}x^{3/2}\sin\frac{4k\pi}{3}\left(1-\frac{1}{x^2}\right), \quad k = 1, 2, 3, \ldots.$$

Collatz.m

Add directory which contains source files (=algorithm):

```
addpath(genpath(pwd))
addpath(genpath('./source'))
```

Define user input tolerance:

```
tol=1e-6;
```

Set `print_messages` *to true if messages (with intermediate results) may be displayed during computation:*

```
print_messages(false);
```

Construct a structure containing information on the problem:

```
problem.pfun=@pfun; %function handles to p,q,w functions
problem.qfun=@qfun;
problem.wfun=@wfun;
problem.a=1;      %integration interval
problem.b=2;
problem.bcs=[1 0 1 0]; %boundary conditions
```

Compute first 21 eigenvalues

```
[E,partition]=calculate_eigenvalues(problem,0,20,tol,true);
```

Display results:

```
disp('first 21 eigenvalues')
disp('  k           E_k              estimated error')
for n=1:length(E.indices)
  disp(sprintf('%3d     %18.12f       %5.1E',E.indices(n),E.eigenvalues(n),...
    abs(E.errors(n))));
end
```

Compute and plot the first eigenfunction:

```
[x,v,vp]=calculate_eigenfunction(partition,E.eigenvalues(1));
figure
plot(x,v,'r*')
legend('y_{1} (tol=10^{-7})')
axis tight
```

Coefficient functions (make sure they are vectorized):

```
function res=pfun(x)
res=1;

function res=qfun(x)
res=3./4./x.^2;

function res=wfun(x)
res=1./x.^6;
```

Results: Add the MATSLEMN directory to the Matlab path and run the Collatz example by typing `Collatz` at the command window. The following results are displayed:

```
first 21 eigenvalues
  k           E_k              exact error
  0        70.183854273169       7.5E-007
  1       280.735440839216       2.7E-005
  2       631.654880819996       2.0E-004
  3      1122.942353498700       7.0E-004
  4      1754.596356102771       1.8E-005
  5      2526.618768848218       4.2E-005
  6      3439.008919663548       9.7E-005
  7      4491.766875107518       2.5E-004
  8      5684.892864004655       7.3E-004
  9      7018.386096731034       7.4E-004
 10      8492.246241985202       3.4E-005
 11     10106.476553079696       1.6E-003
 12     11861.074215636734       3.0E-003
 13     13756.037729506997       2.4E-003
 14     15791.373462344138       6.4E-003
 15     17967.080115461278       1.4E-002
 16     20283.149503909335       1.6E-002
 17     22739.549880838826       1.9E-002
 18     25336.371250100965       1.3E-004
 19     28073.541403491534       4.1E-006
 20     30951.079700567687       3.0E-004
```

and a plot of the eigenfunction is shown.

Contact

Dept. of Applied Mathematics and Computer Science
Ghent University
Krijgslaan 281 (S9) B-9000 Ghent Belgium
Veerle.Ledoux@UGent.be