libceigs

Generated by Doxygen 1.6.3

Tue Feb 28 12:31:43 2012

Contents

Chapter 1

ceigs doxygen documentation

Author

Edgar Simo-Serra <esimo@iri.upc.edu>

Version

1.1

Date

January 2012

1.1 License

Please note that if linked to UMFPACK this library becomes GPLv3+ licensed and not LGPLv3+ licensed.

1.2 Overview

This is a simple C frontend for ARPACK. This allows easy access to calculating a subset of eigenvectors and eigenvalues of sparse matrices. Specifically it can solve two problems:

- $Av = dv$
- $Av = dMv$

Where A, M are sparse matrices, v is the subset of eigenvectors and d is the diagonal matrix of eigenvalues.

Currently only symmetric real matrices are supported.

1.3 Usage

ARPACK uses a Reverse Communication Interface to be extremely flexible. This wrapper aims to keep that flexibility while also making it much easier to use. A small example of usage would be:

```
#include <ceigs.h>
#include <suitesparse/csparse.h>
#include <stdio.h>

int main( int argc, char *argv[] )
{
   cs *A; // Matrix to work with
   int n; // Order of the matrix
   int nev; // Number of eigenvectors to calculate, nev < n+1
   A = cs_spalloc( ... );
   // Fill A here

   // Output data
   double *v, *d;
   v = malloc( n * nev * sizeof(double) );
   d = malloc( nev * sizeof(double) );

   // Actual algorithm
   int ret;
   ret = eigs( n, nev, d, v, // Output and problem information
         A, NULL, // We are solving Av = vd, so no need for second matrix
         EIGS_ORDER_SM, // Order to get eignvalues in
         EIGS_MODE_I_REGULAR, // Mode of operation of ARPACK
         NULL, NULL ); // Use default driver (csparse) and default options
   if (ret != 0)
      fprintf( stderr, "An error occurred while running eigs!\n" );

   // Print some output
   int i;
   for (i=0; i<nev; i++)
      printf( "Eigenvalue  %d: %f\n"
              "Eigenvector %d: ( %f, %f, %f, %f, ... )\n",
              i, d[i], i, v[i*n+0], v[i*n+1], v[i*n+2], v[i*n+3] );
```

```
    // Clean up
    cs_spfree( A );
    free( v );
    free( d );

    return 0;
}
```

This example would calculate nev eigenvectors and eigenvalues of the matrix A in the order of smallest magnitude first. To compile you would have to do:

```
$ gcc -lceigs -larpack -lcxsparse ceigs_test.c -o ceigs_test
```

1.4 Changelog

- Version 1.2, (unreleased)
  - Changed default number of Lanczos vectors to a faster default.
  - Tweaked ARPACK parameters to behave like MATLAB.
  - Improved error reporting.
- Version 1.1, January 2012
  - Invert the eigenvector/value order to match octave/matlab's eigs(...) function.
  - Support for EIGS_MODE_I_SHIFTINVERT with default driver backend.
  - Support for EIGS_MODE_G_SHIFTINVERT with default driver backend.
  - Added number of Lanczos vectors to use as a parameter.
  - Added driver that uses UMFPACK backend (default).
  - Added driver that tries Cholesky factorization, then LU and finally.
  - Added driver that tries LU factorization then QR.
  - Added driver that tries QR factorization.
- Version 1.0, December 2011
  - Initial Revision.
  - Support for EIGS_MODE_I_REGULAR with default driver backend.
  - Support for EIGS_MODE_G_REGINVERSE with default driver backend.

1.5 References

- Rich Lehoucq, Kristi Maschhoff, Danny Sorensen and Chao Yang. ARPACK. http://www.caam.rice.edu/software/ARPACK/

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Data Structure Documentation

4.1 EigsDriver_t Struct Reference

Driver backend.

`#include <ceigs.h>`

Data Fields

- EigsInitdrv_t init
- EigsFreedrv_t free
- EigsDsdrv_t dsdrv

4.1.1 Detailed Description

Driver backend.

4.1.2 Field Documentation

4.1.2.1 EigsDsdrv_t EigsDriver_t::dsdrv

Actual driver implementation. For exact specification please refer to the ARPACK user guide. Set to NULL to indicate this backend is not supported.

4.1.2.2 EigsFreedrv_t EigsDriver_t::free

Driver clean up function. Set to NULL if not needed.

4.1.2.3 EigsInitdrv_t EigsDriver_t::init

Driver initialization function. Set to NULL if not needed.

The documentation for this struct was generated from the following file:

- ceigs.h

4.2 EigsDriverGroup_t Struct Reference

List of drivers that can be used. Any can be set to NULL to indicate unsupported.

`#include <ceigs.h>`

Data Fields

- EigsDriver_t driver1
- EigsDriver_t driver2
- EigsDriver_t driver3
- EigsDriver_t driver4
- EigsDriver_t driver5
- EigsDriver_t driver6

4.2.1 Detailed Description

List of drivers that can be used. Any can be set to NULL to indicate unsupported.

4.2.2 Field Documentation

4.2.2.1 EigsDriver_t EigsDriverGroup_t::driver1

Driver for EIGS_MODE_I_REGULAR.

4.2.2.2 EigsDriver_t EigsDriverGroup_t::driver2

Driver for EIGS_MODE_I_SHIFTINVERT.

4.2.2.3 EigsDriver_t EigsDriverGroup_t::driver3

Driver for EIGS_MODE_G_REGINVERSE.

4.2.2.4 EigsDriver_t EigsDriverGroup_t::driver4

Driver for EIGS_MODE_G_SHIFTINVERT.

4.2.2.5 EigsDriver_t EigsDriverGroup_t::driver5

Driver for EIGS_MODE_G_BUCKLING.

4.2.2.6 EigsDriver_t EigsDriverGroup_t::driver6

Driver for EIGS_MODE_G_CAYLEY.

The documentation for this struct was generated from the following file:

- ceigs.h

4.3 EigsOpts_t Struct Reference

Options to use.

`#include <ceigs.h>`

Data Fields

- int iters
- double tol
- double sigma
- int ncv

4.3.1 Detailed Description

Options to use.

4.3.2 Field Documentation

4.3.2.1 int EigsOpts_t::iters

Maximum iterations during algorithm execution. Default is 3000.

4.3.2.2 int EigsOpts_t::ncv

Number of Lanczos vectors to computer (0 to autoset). This value must be larger than one plus the number of eigenvalues being calculated. Default is 0.

4.3.2.3 double EigsOpts_t::sigma

Value used for the shift-invert modes to choose where to calculate eigenvalues near. Default is 0.0.

4.3.2.4 double EigsOpts_t::tol

Tolerance to use. A value of 0.0 indicates to use maximum machine precision. Default is 0.0.

The documentation for this struct was generated from the following file:

- ceigs.h

Chapter 5

File Documentation

5.1 ceigs.h File Reference

The main include of the ceigs library.

Data Structures

- struct EigsOpts_t

  Options to use.

- struct EigsDriver_t

  Driver backend.

- struct EigsDriverGroup_t

  List of drivers that can be used. Any can be set to NULL to indicate unsupported.

Defines

- #define EIGS_VERSION_MAJOR 1
- #define EIGS_VERSION_MINOR 1

Typedefs

- typedef void *(* EigsInitdrv_t)(int n, const void *data_A, const void *data_M, const EigsOpts_t *opts)

  Prototype for driver initialization.

- typedef void(* EigsFreedrv_t)(void *data, const EigsOpts_t *opts)

  Prototype for driver clean up.

- typedef int(∗ EigsDsdrv_t)(int ido, int n, double ∗workd, const int ∗ipntr, const void ∗data_A, const void ∗data_M, void ∗extra)

  Prototype for a dsaupd_ driver.

Enumerations

- enum EigsOrder_t {

  EIGS_ORDER_LA, EIGS_ORDER_SA, EIGS_ORDER_LM, EIGS_-ORDER_SM,

  EIGS_ORDER_BE }

  Ordering to return eigenvalues as.

- enum EigsMode_t {

  EIGS_MODE_I_REGULAR, EIGS_MODE_I_SHIFTINVERT, EIGS_-MODE_G_REGINVERSE, EIGS_MODE_G_SHIFTINVERT,

  EIGS_MODE_G_BUCKLING, EIGS_MODE_G_CAYLEY }

  Mode of operation of the ARPACK backend.

Functions

- void eigs_optsDefault (EigsOpts_t ∗opts)

  Sets the default parameters.

- int eigs (int n, int nev, double ∗lambda, double ∗vec, const void ∗data_A, const void ∗data_M, EigsOrder_t order, EigsMode_t mode, const EigsDriverGroup_t ∗drvlist, const EigsOpts_t ∗opts)

  Main interface to the ARPACK eigen vector calculator.

- void eigs_version (int ∗major, int ∗minor)

  Gets the version of the library during runtime.

Variables

- const EigsDriverGroup_t eigs_drv_umfpack

  Driver group using the UMFPACK library.

- const EigsDriverGroup_t eigs_drv_cholesky

  Driver group using Cholesky factorization.

- const EigsDriverGroup_t eigs_drv_lu

Driver group using LU factorization.

- const EigsDriverGroup_t eigs_drv_qr

  Driver group using QR factorization.

5.1.1 Detailed Description

The main include of the ceigs library.

5.1.2 Define Documentation

5.1.2.1 #define EIGS_VERSION_MAJOR 1

Major version of the ceigs library.

5.1.2.2 #define EIGS_VERSION_MINOR 1

Minor version of the ceigs library.

5.1.3 Typedef Documentation

5.1.3.1 typedef int(∗ EigsDsdrv_t)(int ido, int n, double ∗workd, const int ∗ipntr, const void ∗data_A, const void ∗data_M, void ∗extra)

Prototype for a dsaupd_ driver.

These drivers are the core of the reverse communication interface used by ARPACK. Their objective is to provide an implementation to manipulate the data type used by the matrices with ARPACK. For exact consultation please refer to the ARPACK user guide.

5.1.3.2 typedef void(∗ EigsFreedrv_t)(void ∗data, const EigsOpts_t ∗opts)

Prototype for driver clean up.

This function should free all data allocated by it's reciprocal driver initialization function. The only input parameter is the pointer returned by the driver initialization function.

See also

EigsInitdrv_t

5.1.3.3 typedef void*(* EigsInitdrv_t)(int n, const void *data_A, const void *data_M, const EigsOpts_t *opts)

Prototype for driver initialization.

This function can generate auxiliary data structures for the backend driver if necessary. The only input parameter is the size of the matrix.

See also

EigsFreedrv_t

5.1.4 Enumeration Type Documentation

5.1.4.1 enum EigsMode_t

Mode of operation of the ARPACK backend.

Note

Note that currently neither EIGS_MODE_G_BUCKLING nor EIGS_MODE_G_-CAYLEY are implemented yet.
By default all the CSPARSE drivers use LU factorization.

Enumerator:

EIGS_MODE_I_REGULAR For solving Av=vd in regular mode.

EIGS_MODE_I_SHIFTINVERT For solving Av=vd in shift-invert mode.

EIGS_MODE_G_REGINVERSE For solving Av=dMv in regular inverse mode.

EIGS_MODE_G_SHIFTINVERT For solving Av=dMv in shift-invert mode.

EIGS_MODE_G_BUCKLING For solving Av=dMv in Buckling mode.

EIGS_MODE_G_CAYLEY For solving Av=dMv in Cayley mode.

5.1.4.2 enum EigsOrder_t

Ordering to return eigenvalues as.

Enumerator:

EIGS_ORDER_LA Largest algebraic eigenvalues first.

EIGS_ORDER_SA Smallest algebraic eigenvalues first.

EIGS_ORDER_LM Largest eigenvalues in magnitude first.

EIGS_ORDER_SM Smallest eigenvalues in magnitude first.

EIGS_ORDER_BE Compute nev eigenvalues, half from each end of the spectrum. When nev is odd, compute one more from the high end than from the low end.

5.1.5 Function Documentation

5.1.5.1 int eigs (int *n*, int *nev*, double * *lambda*, double * *vec*, const void * *data_A*, const void * *data_M*, EigsOrder_t *order*, EigsMode_t *mode*, const EigsDriverGroup_t * *drvlist*, const EigsOpts_t * *opts*)

Main interface to the ARPACK eigen vector calculator.

This either solves the problem "Ax=lx" or "Ax=Mlx".

For the algorithm to work the following conditions must be met:

- nev <= n
- nev+1 <= ncv

Various alternative dirvers are provided, these include:

- eigs_drv_lu: LU factorization driver. Works for non-singular matrices.
- eigs_drv_cholesky: Cholesky factorization driver. Works for non-singular symmetric positive definite matrices.
- eigs_drv_qr: QR factorization driver. Always works, but has worse precision.

Note

When using the default drivers, matrix A and matrix B should be be compressed csparse matrices.

Parameters

← ***n*** Order of the matrix.

← ***nev*** Number of eigenvectors and eigenvalues to calculate.

→ ***lambda*** Eigenvalues calculated (should be of size nev).

→ ***vec*** Eigenvectors calculated (should be of size nev∗n). If NULL ARPACK will not calculate the eigenvectors and only compute eigenvalues.

← ***data_A*** A matrix data.

← ***data_M*** M matrix data if applicable.

← ***order*** Ordering to find eigenvalues in.

← ***mode*** Mode of operation.

← ***drvlist*** List of drivers to use or NULL to use defaults.

← ***opts*** Options to use or NULL to use defaults.

Returns

0 on success.

See also

eigs_optsDefault

5.1.5.2 void eigs_optsDefault (EigsOpts_t * *opts*)

Sets the default parameters.

This function should generally be used before manipulating the options of the ARPACK for future compatibility.

Parameters

→ ***opts*** Default options for eigs.

5.1.5.3 void eigs_version (int * *major*, int * *minor*)

Gets the version of the library during runtime.

The version takes the form of major.minor.

Parameters

→ ***major*** Major version of the library.

→ ***minor*** Minor version of the library.

See also

EIGS_VERSION_MAJOR
EIGS_VERSION_MINOR

5.1.6 Variable Documentation

5.1.6.1 const EigsDriverGroup_t eigs_drv_cholesky

Driver group using Cholesky factorization.

This driver will first attempt Cholesky factorization, then LU factorization and finally QR factorization if the previous fails. This driver is the default.

Note

For Cholesky factorization to work, the A matrix must be Hermitian, positive definite, and non-singular.

5.1.6.2 const EigsDriverGroup_t eigs_drv_lu

Driver group using LU factorization.

This driver will fall back to QR factorization if LU factorization fails.

Note

For LU factorization to work, the A matrix must be non-singular.

5.1.6.3 const EigsDriverGroup_t eigs_drv_qr

Driver group using QR factorization.

Note

This works with least-squares so the A matrix can be singular.

5.1.6.4 const EigsDriverGroup_t eigs_drv_umfpack

Driver group using the UMFPACK library.

This is the most complete and powerful driver of them all.

Index