
PyOrder Documentation

Release 0.1.0

Dominique Orban

November 02, 2009

CONTENTS

1	Introduction	3
1.1	A Few Concepts about Symmetric Sparse Matrices	3
1.2	Availability	4
2	Bringing Nonzero Elements to the Diagonal	5
2.1	The <code>pymc21</code> Module	5
2.2	Examples	5
3	Profile, Wavefront, Bandwidth Reduction	9
3.1	The <code>pymc60</code> Module	9
3.2	Examples	10
4	Input / Output	15
4.1	The <code>hrb</code> Module	15
5	Other Tools	17
5.1	The <code>spy</code> Module	17
6	Indices and tables	19
	Bibliography	21
	Module Index	23
	Index	25

Contents:

INTRODUCTION

PyOrder is a Python package that provides access to several means to reorder sparse matrices. Typically, sparse matrices are reordered prior to factorization so as to preserve sparsity of the factors, ensure that there are as many nonzero elements as possible on the diagonal, or decrease the envelope, wavefront, profile or semi-bandwidth.

Note: Orderings designed to promote sparsity of the factors of a matrix are different in nature from the other types of orderings mentioned in the previous paragraph and are not considered in PyOrder. If you would like to see such orderings (e.g., AMD, CAMD, COLAMD, etc.) included in future releases of PyOrder, please let me know.

1.1 A Few Concepts about Symmetric Sparse Matrices

Although not all ordering functions in this package assume symmetry, most concepts are defined for symmetric matrices only.

If A is a square $n \times n$ symmetric sparse matrix, let a_{ij} be the element at the intersection of the i -th row and the j -th column. The following concepts are important for reordering purposes.

The i -th *wavefront* of A is the number of nonzero rows in the submatrix $A(i : n, 1 : i)$, $i = 1, \dots, n$. It is denoted f_i .

The maximum and mean-square wavefronts are

$$\max_{i=1,\dots,n} f_i \quad \text{and} \quad \frac{1}{n} \sum_{i=1}^n f_i^2.$$

If m_i is the column index of the first nonzero element on the i -th row of A , the *envelope* of A is the set of nonzero elements lying between (i, m_i) and the diagonal, exclusive of the diagonal, i.e.

$$\text{Env}(A) = \{(i, j) \mid 1 \leq i \leq n, m_i \leq j < i\}.$$

The *profile* $P(A)$ of the matrix is the number of elements in $\text{Env}(A)$ plus the number of nonzero elements on the diagonal, i.e.,

$$P = \sum_{i=1}^n (i - m_i + 1).$$

It is possible to show that those quantities are linked via the relationship

$$P = \sum_{i=1}^n f_i.$$

1.2 Availability

PyOrder is essentially a set of interfaces to subroutines from the HSL <http://www.cse.scitech.ac.uk/nag/hsl> (formerly the Harwell Subroutine Library). Subroutines from the HSL may be obtained free of charge under certain conditions. See <http://hsl.rl.ac.uk/hsl2007/hsl20074researchers.html> to decide whether the terms apply to you. The HSL subroutines relevant to PyOrder are not packaged together with the Python interfaces and should be obtained separately.

Note: At this time, only double precision real data is supported. Please let me know if you would like support for single precision and/or complex data.

BRINGING NONZERO ELEMENTS TO THE DIAGONAL

2.1 The `pymc21` Module

A Python interface to the HSL subroutine MC21AD.

References

nonzerodiag (*nrow*, *colind*, *rowptr*)

Given the sparsity pattern of a square sparse matrix in compressed row (csr) format, attempt to find a *row* permutation so the row-permuted matrix has a nonzero diagonal, if this is possible. This function assumes that the matrix indexing is 0-based. Note that in general, this function does not preserve symmetry. The method used is a depth-first search with lookahead described in [Duff81a] and [Duff81b].

- Parameters**
- nrow** The number of rows of the input matrix.
 - colind** An integer array (or list) of length `nnz` giving the column indices of the nonzero elements in each row.
 - rowptr** An integer array (or list) of length `nrow+1` giving the indices of the first element of each row in `colind`.
- Returns**
- perm** An integer array of length `nrow` giving the variable permutation. If `irow` and `jcol` are two integer arrays describing the pattern of the input matrix in triple format, `perm[irow]` and `jcol` describe the permuted matrix.
 - nzdiag** The number of nonzeros on the diagonal of the permuted matrix.

2.2 Examples

2.2.1 Basic Usage

This first example calls the Fortran subroutine directly with the matrix data in compressed sparse column format.

Warning: Keep in mind that when calling the Fortran subroutines directly, all indices in the matrix data must be 1-based, i.e., row indices range from 1 through `nrow` and column indices range from 1 through `ncol`.

```
1 import numpy as np
2 from pyorder.pymc21 import mc21module
3 n = 4
4 icn = np.array([1,4,3,4,1,4,2,4], dtype=np.int32)
5 ip = np.array([1,3,5,7], dtype=np.int32)
6 lenr = np.array([2,2,2,2], dtype=np.int32)
7 iperm,numnz = mc21module.mc21ad(icn,ip,lenr)
8 print 'iperm = ', iperm, ' (1-based)'
9 print 'numnz = ', numnz
```

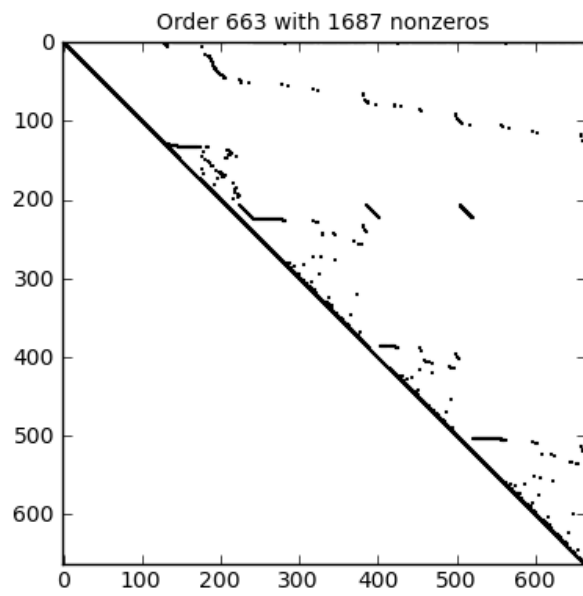
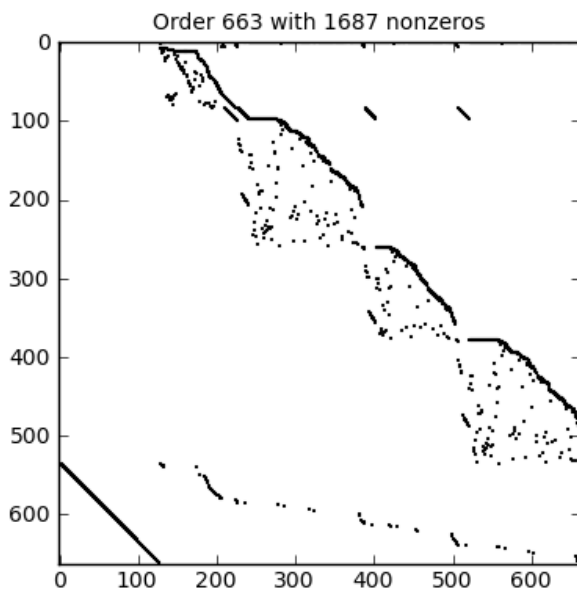
2.2.2 Python Interface

In this second example, Python usage is intended. Matrix indices must therefore be 0-based. The permutation vector returned by `nonzerodiag()` is now also 0-based.

```
1 """
2 Illustrate usage of the pymc21 module, using an input matrix in Harwell-Boeing
3 or Rutherford-Boeing format. Supply a file name as input argument on the
4 command line and uncomment below as appropriate.
5 """
6
7 import sys
8 import numpy as np
9 from pyorder.pymc21.pymc21 import nonzerodiag
10 from pyorder.tools.hrb import HarwellBoeingMatrix, RutherfordBoeingData
11 from pyorder.tools.spy import FastSpy
12 import pylab
13
14 if len(sys.argv) < 2:
15     sys.stderr.write('Supply input matrix as argument\n')
16     sys.exit(1)
17
18 fname = sys.argv[1]
19 #M = HarwellBoeingMatrix(fname, patternOnly=True, readRhs=False)
20 M = RutherfordBoeingData(fname, patternOnly=True, readRhs=False)
21
22 if M.nrow != M.ncol:
23     sys.stderr.write('Input matrix must be square\n')
24     sys.exit(1)
25
26 perm, nzdiag = nonzerodiag(M.nrow, M.ind, M.ip)
27
28 (irow, jcol) = M.find()
29 left = pylab.subplot(121)
30 FastSpy(M.nrow, M.ncol, irow, jcol, sym=M.issym, ax=left.get_axes())
31
32 right = pylab.subplot(122)
33 FastSpy(M.nrow, M.ncol, perm[irow], jcol, sym=M.issym, ax=right.get_axes())
34 pylab.show()
```

Reorganizing a square matrix so it has a nonzero diagonal is useful for instance when factorizing a submatrix of a basis in the Simplex method for linear programming.

The effect of `nonzerodiag()` is illustrated on the following example, `shl_0` from the [University of Florida Sparse Matrix Collection](#). I used the matrix in Rutherford-Boeing format.



PROFILE, WAVEFRONT, BANDWIDTH REDUCTION

3.1 The `pymc60` Module

A Python interface to the HSL subroutine MC60AD.

The functions in this module compute a symmetric permutation of a sparse symmetric matrix so as to reduce its profile, wavefront, or bandwidth via Sloan's method [SLO] or the reverse Cuthill-McKee method [CM].

References

sloan (*n*, *rowind*, *colptr*, *icntl*=, [0, 6], *weight*=, [2, 1])

Apply Sloan's algorithm to reduce the profile and wavefront of a sparse symmetric matrix. Either the lower or the upper triangle of the input matrix should be given in compressed sparse column (csc) or compressed sparse row (csr) format. This includes the diagonal of the matrix. A set of weights can be supplied to define the priority function in Sloan's method.

Parameters **n** The order of the input matrix.

rowind An integer array (or list) of length nnz giving the row indices of the nonzero elements in each column.

colptr An integer array (or list) of length n+1 giving the indices of the first element of each column in rowind.

Note that since either triangle can be given in either csc or csr format, the words 'row' and 'column' may be swapped in the description above. The indexing in rowind and colptr should be zero-based.

Keywords **icntl** An integer array (or list) of length two of control parameters used during the first phase, where the input data is checked. The method terminates if duplicates of out-of-range indices are discovered (*icntl*[0]=0) or ignores them (*icntl*[0]=1). No diagnostic messages will be output if *icntl*[1]=0. If *icntl*[1] is > 0, it gives the unit number (in the Fortran sense) where diagnostic messages are output.

weight An integer array (or list) of length two giving the weights in Sloan's priority function. Reid and Scott (1999) recommend to apply the method twice, with either [2,1] and [16,1], or with [1,2] and [16,1], and to retain the best result.

Returns **perm** An integer array of length n giving the variable permutation. If *irow* and *jcol* are two integer arrays describing the pattern of the input matrix in triple format, *perm*[*irow*] and *perm*[*jrow*] describe the permuted matrix.

rinfo A real array of length 4 giving statistics on the permuted matrix. `rinfo[0]` = profile `rinfo[1]` = maximum wavefront `rinfo[2]` = semi-bandwidth `rinfo[3]` = root-mean-square wavefront.

rcmk (*n*, *rowind*, *colptr*; *icntl*=, [0, 6])

Apply the reverse Cuthill-McKee algorithm to reduce the bandwidth of a sparse symmetric matrix. Either the lower or the upper triangle of the input matrix should be given in compressed sparse column (csc) or compressed sparse row (csr) format. This includes the diagonal of the matrix.

Parameters **n** The order of the input matrix.

rowind An integer array (or list) of length `nnz` giving the row indices of the nonzero elements in each column.

colptr An integer array (or list) of length `n+1` giving the indices of the first element of each column in `rowind`.

Note that since either triangle can be given in either csc or csr format, the words ‘row’ and ‘column’ may be swapped in the description above. The indexing in `rowind` and `colptr` should be zero-based.

Keywords **icntl** An integer array (or list) of length two of control parameters used during the first phase, where the input data is checked. The method terminates if duplicates of out-of-range indices are discovered (`icntl[0]=0`) or ignores them (`icntl[0]=1`). No diagnostic messages will be output if `icntl[1]=0`. If `icntl[1]` is > 0 , it gives the unit number (in the Fortran sense) where diagnostic messages are output.

Returns **perm** An integer array of length `n` giving the variable permutation. If `irow` and `jcol` are two integer arrays describing the pattern of the input matrix in triple format, `perm[irow]` and `perm[jcol]` describe the permuted matrix.

rinfo A real array of length 4 giving statistics on the permuted matrix. `rinfo[0]` = profile `rinfo[1]` = maximum wavefront `rinfo[2]` = semi-bandwidth `rinfo[3]` = root-mean-square wavefront.

reorder_matrix (*n*, *rowind*, *colptr*; *icntl*=, [0, 6], *jcntl*=, [0, 0], *weight*=, [2, 1])

Helper function called by *sloan* and *rcm* performing the bulk of the work when applying Sloan’s method or the reverse Cuthill-McKee algorithm to a symmetric sparse matrix.

3.2 Examples

3.2.1 Basic Usage

The first example is the one from the documentation of the HSL subroutine MC60. In it, we call the Fortran subroutines directly. As before, this means that all indices must be 1-based. The permutation vector and indices of the supervariables are also 1-based.

```
1 "MC60 demo from the HSL MC60 spec sheet"
2 import numpy as np
3 from pyorder.pymc60 import mc60module
4
5 icntl = np.array([0,6], dtype=np.int32) # Abort on error
6 jcntl = np.array([0,0], dtype=np.int32) # Sloan's alg with auto choice
7 weight = np.array([2,1])               # Weights in Sloan's alg
8
9 # Store lower triangle of symmetric matrix in csr format (1-based)
10 n = 5
11 icptr = np.array([1,6,8,9,10,11], dtype=np.int32) # nnz = 10
```

```

12  irn = np.empty(2*(icptr[-1]-1), dtype=np.int32)
13  irn[:icptr[-1]-1] = np.array([1,2,3,4,5,2,3,3,4,5], dtype=np.int32)
14
15  # Check data
16  info = mc60module.mc60ad(irn, icptr, icntl)
17
18  # Compute supervariables
19  nsup, svar, vars = mc60module.mc60bd(irn, icptr)
20  print 'The number of supervariables is ', nsup
21
22  # Permute reduced matrix
23  permsv = np.empty(nsup, dtype=np.int32)
24  pair = np.empty((2, nsup/2), dtype=np.int32)
25  info = mc60module.mc60cd(n, irn, icptr[:nsup+1], vars[:nsup], jcntl, permsv, weight, pair)
26
27  # Compute profile and wavefront
28  rinfo = mc60module.mc60fd(n, irn, icptr[:nsup+1], vars[:nsup], permsv)
29
30  # Obtain variable permutation from supervariable permutation
31  perm, possv = mc60module.mc60dd(svar, vars[:nsup], permsv)
32  print 'The variable permutation is ', perm      # 1-based
33  print 'The profile is ', rinfo[0]
34  print 'The maximum wavefront is ', rinfo[1]
35  print 'The semibandwidth is ', rinfo[2]
36  print 'The root-mean-square wavefront is ', rinfo[3]

```

3.2.2 Python Interface

The Python interface provides smoother and more intuitive application of Sloan's and the reverse Cuthill-McKee methods by way of the `sloan()` and `rcmk()` functions. At the same time, we use input data in either Harwell-Boeing or Rutherford-Boeing format.

```

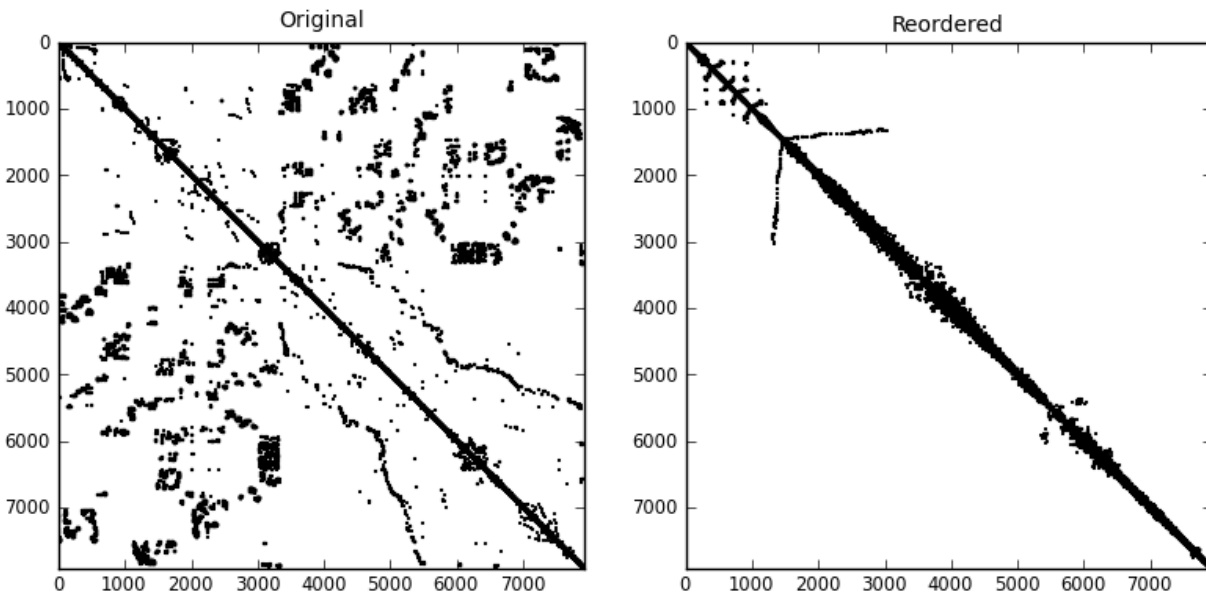
1  "MC60 demo with input matrix in HB or RB format"
2
3  import numpy as np
4  from pyorder.pymc60 import sloan, rcmk
5  from pyorder.tools.hrb import HarwellBoeingMatrix, RutherfordBoeingData
6  from pyorder.tools.spy import FastSpy
7  import pylab
8  import sys
9
10 if len(sys.argv) < 2:
11     sys.stderr.write('Data file name must be supplied\n')
12     sys.exit(1)
13
14 fname = sys.argv[1]
15 #M = HarwellBoeingMatrix(fname, patternOnly=True, readRhs=False)
16 M = RutherfordBoeingData(fname, patternOnly=True, readRhs=False)
17
18 if M.nrow != M.ncol or not M.issym:
19     sys.stderr.write('Input matrix must be square and symmetric\n')
20     sys.exit(1)
21
22 # Compute reverse Cuthill-McKee ordering
23 perm, rinfo = rcmk(M.nrow, M.ind, M.ip)
24

```

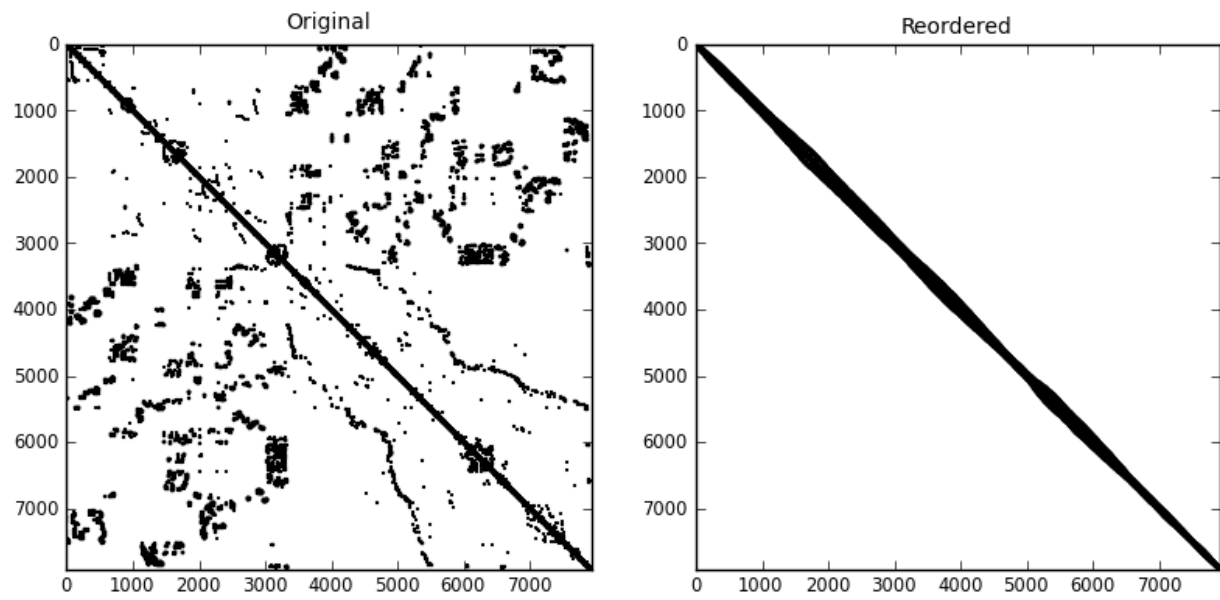
```
25 # Or: Compute Sloan's ordering
26 #perm, rinfo = sloan(M.nrow, M.ind, M.ip)
27
28 # Plot original matrix
29 (irow, jcol) = M.find()
30 left = pylab.subplot(121)
31 FastSpy(M.nrow, M.ncol, irow, jcol, sym=M.issym,
32         ax=left.get_axes(), title='Original')
33
34 # Apply permutation and plot reordered matrix
35 right = pylab.subplot(122)
36 FastSpy(M.nrow, M.ncol, perm[irow], perm[jcol], sym=M.issym,
37         ax=right.get_axes(), title='Reordered')
38 pylab.show()
```

For illustration, here is the result of this example on a problem from the [University of Florida Sparse Matrix Collection](#). I used matrix `commanche_dual` in Rutherford-Boeing format.

Here is the matrix reordered according to Sloan's algorithm.



Here it is reordered according to the reverse Cuthill-McKee algorithm.



INPUT / OUTPUT

4.1 The `hrb` Module

Provides access to sparse linear systems described in Harwell-Boeing or Rutherford-Boeing format. This module exposes the two classes `HarwellBoeingMatrix` and `RutherfordBoeingData`. For more information, see the references below.

References

class `HarwellBoeingMatrix` (*fname*, ***kwargs*)

Imports a sparse matrix from a file in Harwell-Boeing format. The matrix is stored in compressed sparse row format in (`self.ind`, `self.ip`, `self.val`). Right-hand sides, if any, are stored in `self.rhs`. Right-hand sides can be stored as dense vectors, in which case `self.rhs` has shape (`nrow`, `nrhs`), as sparse vectors, in which case they are stored in compressed sparse column format in (`self.rhsptr`, `self.rhsind`, `self.rhs`), or in elemental format (typically when the matrix itself is stored in finite-element format), in which case `self.rhs` has shape (`nnzero`, `nrhs`).

Note that the matrix indices are zero-based, i.e., row indices range from 0 through `nrow-1` and column indices range from 0 through `ncol-1`.

The matrix can be subsequently converted to triple format with (`row`, `col`) = `self.find()`

Keywords **`patternOnly`** do not read matrix element values (False)

`readRhs` read right-hand sides, if any (False)

`readGuess` read starting guess, if any (False)

`realSol` read solution vector, if any (False)

`find()`

`fortranRead` (*stream*, *format*)

`readArray` (*fp*, *which*, *nelm*, *format*)

`readMatrix` (*fp*, ***kwargs*)

class `RutherfordBoeingData` (*fname*, ***kwargs*)

Bases: `pyorder.tools.hrb.HarwellBoeingMatrix`

Imports data from a file in Rutherford-Boeing format. The data is held in (`self.ind`, `self.ip`, `self.val`). If the data represents a sparse matrix, the three arrays represent the matrix stored in compressed sparse row format. Otherwise, the three arrays represent the supplementary data. Refer to the Rutherford-Boeing documentation for more information (reference [4] in the docstring for the present module.)

Note that the matrix indices are zero-based, i.e., row indices range from 0 through `nrow-1` and column indices range from 0 through `ncol-1`.

The data can be subsequently converted to triple format with `(row, col) = self.find()`

Keywords **patternOnly** do not read data values (False)

find()

fortranRead (*stream, format*)

readArray (*fp, which, nelm, format*)

readMatrix (*fp, **kwargs*)

OTHER TOOLS

5.1 The `spy` Module

Plot the sparsity pattern of a sparse matrix.

FastSpy (*nrow, ncol, irow, jcol, **kwargs*)

To plot the sparsity pattern of a sparse matrix in coordinate format.

Arguments **nrow** Number of rows of the matrix.

ncol Number of columns of the matrix.

irow Integer Numpy array of length `nnz` giving the row indices of the nonzero elements.

jcol Integer Numpy array of length `nnz` giving the column indices of the nonzero elements.

Keywords **ax** A Pylab Axes instance used to plot the sparsity pattern. If none is given, this function returns an Axes instance.

sym Should be set to True if the matrix is symmetric and only one triangle is passed in (`irow,jcol`).

title String to be used as title for the plot. If none is given, the title defaults to giving the order of the matrix and the number of nonzero elements.

comment Comment to be appended to the default plot title in case no title is supplied.

showtitle Show or hide plot title.

val Float Numpy array of length `nnz` giving the values of the nonzero elements of the matrix. If supplied, a scatter plot is produced with patches of size proportional to the magnitude of the element. This option can slow down the plot for large values of `nnz`.

INDICES AND TABLES

- *Index*
- *Module Index*
- *Search Page*

BIBLIOGRAPHY

- [Duff81a] I. S. Duff, *On Algorithms for Obtaining a Maximum Transversal*, ACM Trans. Math. Software, **7**, pp. 315-330, 1981.
- [Duff81b] I. S. Duff, *Algorithm 575: Permutations for a zero-free diagonal*, ACM Trans. Math. Software, **7**, pp. 387-390, 1981.
- [CM] E. Cuthill and J. McKee, *Reducing the bandwidth of sparse symmetric matrices* In Proc. 24th Nat. Conf. ACM, pages 157-172, 1969.
- [RS] J. K. Reid and J. A. Scott, *Ordering symmetric sparse matrices for small profile and wavefront*, International Journal for Numerical Methods in Engineering, **45**(12), pp. 1737-1755, 1999.
- [SLO] S. W. Sloan, *An algorithm for profile and wavefront reduction of sparse matrices*, International Journal of Numerical Methods in Engineering, **23**, pp. 239-251, 1986.
- [DGL] I.S. Duff, R.G. Grimes and J.G. Lewis, *Sparse Matrix Test Problems*, ACM Transactions on Mathematical Software, **15**(1), p.1-14, 1989
- [HBUG] ftp://ftp.cerfacs.fr/pub/algo/matrices/harwell_boeing/userguide.ps.Z
- [HB] <http://math.nist.gov/MatrixMarket/data/Harwell-Boeing>
- [RBC] The Rutherford-Boeing Sparse Matrix Collection, I.S. Duff, R.G. Grimes and J.G. Lewis, Technical Report RAL-TR-97-031, Rutherford Appleton Laboratory, Chilton, OX, UK, 1997. (<ftp://ftp.numerical.rl.ac.uk/pub/reports/duglRAL97031.pdf>)
- [RB] <http://www.cerfacs.fr/algor/Softs/RB>

MODULE INDEX

P

`pyorder.pymc21.pymc21`, [5](#)
`pyorder.pymc60.pymc60`, [9](#)
`pyorder.tools.hrb`, [15](#)
`pyorder.tools.spy`, [17](#)

INDEX

F

FastSpy() (in module pyorder.tools.spy), 17
find() (pyorder.tools.hrb.HarwellBoeingMatrix method), 15
find() (pyorder.tools.hrb.RutherfordBoeingData method), 16
fortranRead() (pyorder.tools.hrb.HarwellBoeingMatrix method), 15
fortranRead() (pyorder.tools.hrb.RutherfordBoeingData method), 16

H

HarwellBoeingMatrix (class in pyorder.tools.hrb), 15

N

nonzerodiag() (in module pyorder.pymc21.pymc21), 5

P

pyorder.pymc21.pymc21 (module), 5
pyorder.pymc60.pymc60 (module), 9
pyorder.tools.hrb (module), 15
pyorder.tools.spy (module), 17

R

rcmk() (in module pyorder.pymc60.pymc60), 10
readArray() (pyorder.tools.hrb.HarwellBoeingMatrix method), 15
readArray() (pyorder.tools.hrb.RutherfordBoeingData method), 16
readMatrix() (pyorder.tools.hrb.HarwellBoeingMatrix method), 15
readMatrix() (pyorder.tools.hrb.RutherfordBoeingData method), 16
reorder_matrix() (in module pyorder.pymc60.pymc60), 10
RutherfordBoeingData (class in pyorder.tools.hrb), 15

S

sloan() (in module pyorder.pymc60.pymc60), 9