PySparse - A sparse linear algebra extension for Python

Roman Geus

January 4, 2007

spmatrix module functions

ll_mat(n, m, sizeHint=1000) Creates a *ll_mat* object, that represents a general, all zero $m \times$

```
>>> from pysparse import spmatrix
>>> A = spmatrix.ll_mat(5, 5)
>>> for i in range(5):
...     A[i,i] = i+1
>>> A[2,1] = A[0,0]
>>> print A
ll_mat(general, [5,5], [(0,0): 1, (1,1): 2, (2,1): 1,
(2,2): 3, (3,3): 4, (4,4): 5])
```

A.export_mtx(fileName, precision=6)

A.update

Example: 2D-Poisson matrix This section illustrates the use of the *spmatrix* module to build the well known 2D-Poisson matrix resulting from a $n \times n$ square grid.

```
def poisson2d(n):
```

Function	$n = 100$	$n = 300$	$n = 500$	$n = 1000$

The performance difference between Python's `poisson2d_sym` and `poisson2d_sym_blk` 751736.5255a.q0.25(s)00rmance985preconce985moduleTf20.42340Td[in07378.679performa

```
class diag_prec:
    def __init__(self, A):
        self.shape = A.shape
        n = self.shape[0]
```

Return value All iterativ6 solv6rs return a tupl6 with thre6 elements (*info*, *iter*, *relres*):

info is an integer that contains the exit status of the iterativ6 solv6r. *info* has one of the following values

2 iteration conv6rged, residual is as small as seems reasonabl6 on this machine.

1 iteration conv6rged, $b = 0$, so 0 the exact solution is

iteration conv6rged,v6 rror6 so6 h(an)]TJ/F459.9626Tf37.353770Td[

Example: Solving the poisson system Let's solve the Poisson system

$$L\,x = 1, \tag{1}$$

Function	Size	t

IinsoIver

Example: Maxwell problem The following code illustrates the use of the *jdsym* module. Two matrices A and M are read from files. A Jacobi preconditioner from $A - \tau M$ is built. Then the JDSYM eigensolver is called, calculating 5 eigenvalues near 25.0 and the associated eigenvalues to an accuracy of 10^{-10}. We set *strategy* = 1 to avoid convergence to the high-dimensional null space of (A, M).

```
from pysparse import spmatrix, itsolvers, jdsym, precon

A = spmatrix.ll_mat_from_mtx('edge6x3x5_A.mtx')
M = spmatrix.ll_mat_from_mtx('edge6x3x5_B.mtx')
tau = 25.0

Atau = A.copy()
Atau.shift(-tau, M)
K = precon.jacobi(Atau)
```

0 natural ordering

1 MMD applied to the structure of $A^T A$

2 MMD applied to the structure of $A^T + A$

3 COLAMD, approximate minimum degree column ordering