CONTENTS

CHAPTER

ONE

INTRODUCTION TO PYSPARSE

PySparse extends the Python interpreter by a set of sparse matrix types holding double precision values. PySparse also includes modules that implement

- Iterative Krylov methods for solving linear systems of equations,
- Diagonal (Jacobi) and SSOR preconditioners,
-

CHAPTER

TWO

SPARSE MATRIX FORMATS

This section describes the sparse matrix storage schemes available in Pysparse. It also covers sparse matrix creation, population and conversion.

-

2.2 Compressed Sparse Row Format

In CSR format, a sparse matrix is represented via three arrays:

va

3.1.2 ll_mat objects

ll_mat

general matrices. If applied to symmetric matrices, only a partial result is returned.

Fancy indexing can also be done with Python lists:

```
>>> print A[ [
```

nnz

Returns the number of non-zero entries stored in matrix **A**
number of non-zero entries in thewber the

ARturns non-zero matrix

tore.Returns matrix
mtvecA

xAheyspashe(matri-vr)15ecstoeproductx

3.2 Example: 2D-Poisson matrix

3.3 Vectorization

The `put` method of `ll_mat` objects allows us to operate on entire arrays at a time. This is advantageous because the loop over the elements of an array is performed at C level instead of in the Pcthon script. For illustration, let's rewrite the `poisson2d`, `poisson2d_sym` and `poisson2d_sym_blk` constructors.

The `put` method can be used in `poisson1d` as so:

CHAPTER

FOUR

PRECONDITIONERS

4.1 The `precon` Module

CHAPTER

FIVE

ITERATIVE SOLVERS

relres the relative residual at the approximate solution computed by the iterative method. What this actually is depends on the actual iterative method used.

The iterative solvers may accept additional parameters, which are passed as keyword arguments.

Note that not all iterative solvers check for all above error conditions.

5.1.1 `itsolvers` Module Functions

The module functions defined in the `precon` module implement various iterative methods (PCG, MINRES, QMRS

`nnz`

The `nnz`

```
    def precon(self, x, y):
        self.LU.solve(x,y)


n = .00
A = poisson.poisson2d_sym_blk(n).to_csr()   # Convert right away
b = numpy.ones(n*n)
x = numpy.empty(n*n)

K = ILU_Precon(A)
info, niter, relres = itsolvers.pcg(A, b, x, 1e-12, 2000, K)
```

Note:

6.2.2 The

`solve(`*rhs, transpose=False*`)`

Solve the linear system `A x = rhs`, where A is the input matrix and `rhs` is a Numpy vector of appropriate dimension. The result is placed in the `sol` member of the class instance.

If the optional argument

scale

PysparsolvePyspar(*rhs, x* **)**PysparSolve the linear systemPyspar $A x = rhs$Pyspar. The result is placed in the

CHAPTER

SEVEN

EIGENVALUE SOLVER

7.1 The `jdsym` Module

The `jdsym`

blkwise is an integer that affects the convergence criterion if `blksize`

```
A = A.to_sss(); M = M.to_sss()
k_conv, lmbd, Q, it  = jdsym.jdsym(A, M, K, 5, tau,
                                   1e-10, 150, itsolvers.qmrs,
                                   jmin=5, jmax=10, clvl=1, strategy=1)
```

CHAPTER

EIGHT

HIGHER-LEVEL SPARSE MATRIX CLASSES

8.1 The `pysparseMatrix` module

class `PysparseMatrix`(***kwargs*)

Bases: `sparseMatrix.SparseMatrix`

```
>>> L = PysparseMatrix(size = 3)
```

8.1.1 Creating an Identity Matrix

class `PysparseIdentityMatrix(`*size*`)`

Bases: `pysparseMatrix.PysparseMatrix`

Represents a sparse identity matrix for pysparse.

CHAPTER

NINE

CHAPTER

TWELVE

INDICES AND TABLES

BIBLIOGRAPHY

[DEGLL99] J. W. Demmel, S. C. Eisenstat, J. R. Gilbert, X. S. Li and J. W. H. Liu, *A supernodal approach to sparse partial pivoting*, SIAM Journal on Matrix Analysis and Applications **20**(3), pp. 720-755, 1999.

[DGL99] J. W. Demmel, J. R. Gilbert and X. S. Li, *An Asynchronous Parallel Supernodal Algorithm for Sparse Gaussian Elimination*, SIAM Journal on Matrix Analysis and Applications **20**(4), pp. 915-952, 1999.

[LD03] X. S. Li and J. W. Demmel, *SuperLU_DIST: A Scacable Distributed-Memory Sparse Direct Solver for Unsymmetric Linear Systems*, ACM Transactions on Mathematical Software **29**(2), pp. 110-140, 2003.

[SLU] http://crd.lbl.gov/~xiaoye/SuperLU

[D04a] T. A. Davis, *A column pre-ordering strategy for the unsymmetric-pattern multifrontal method*, ACM Transactions on Mathematical Software, **30**(2), pp. 165-195, 2004. T. A. Davis, *Algorithm 832: UMFPACK, an unsymmetric-pattern multifrontal method*, Mathematical Software, **30**(2), pp. 196. 2004.

[DD99] T. A. Davis and I. S. Duff, *A combined unifrontal/multifrontal method for unsymmetric s* Transactions on Mathematical Software, **25**(1), pp. 1-19, 1999. T. A. Davis and I. S. Duff, *An unsymm* Journal on Matrix Analysis and Application

[UMF] http://www.cise.ufl.edu/research/sparse/

INDEX

A

C