COPPE UFRJ
Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia

UFRJ

Relatório atividade 3

Aluno: Lyncoln Sousa de Oliveira

Engenharia de Sistemas e Computação PESC/Coppe

Objetivos

- Buscar uma matriz esparsa simétrica positiva definida
- Aplicar o método lanczos pelo ARPACK para e calcular autopares da matriz
- Comparar tempo de processamento para diferentes estruturas de matrizes esparsas

Engenharia de Sistemas e Computação PESC/Coppe

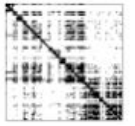
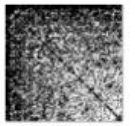
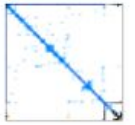
Ambiente de execução

- Super computador SDumont
- Executado na fila gdl (IA)
- CPU: 2x Intel(R) Xeon(R) Gold 6148 2.4GHz
- 384 Gbs de memória ram

Engenharia de Sistemas e Computação PESC/Coppe

Buscar matriz no ssgetpy

```
consulta = ssgetpy.search(rowbounds=(500_000,700_000)
                          ,colbounds=(500_000,700_000),
                          nzbounds = (1_000_000,10_000_000),
                          limit = 10)
consulta
```

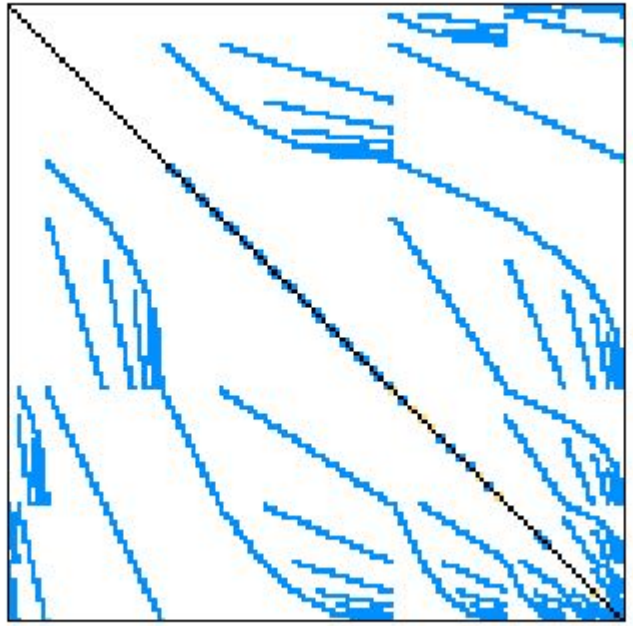
Id	Group	Name	Rows	Cols	NNZ	DType	2D/3D Discretization?	SPD?	Pattern Symmetry	Numerical Symmetry	Kind	Spy Pl
285	ATandT	pre2	659033	659033	5834044	real	No	No	0.33	0.065	frequency-domain circuit simulation problem	
980	Kamvar	Stanford_Berkeley	683446	683446	7583376	binary	No	No	0.25	0.25	directed graph	
1383	Andrianov	lp1	534388	534388	1643420	binary	No	No	1.0	1.0	optimization problem	
1396	Rajat	rajat29	643994	643994	3760246	real	No	No	0.69	0.69	circuit simulation problem	

```
consulta.download(destpath = f'{os.getcwd()}\\matrix',extract=True)[8]
```

Matriz escolhida

Id: 1853 Grupo: Wissgott Nome: parabolic_fem

- Matriz com 525.825 linhas e colunas
- Tipo Real, com 3.674.625 números diferentes de 0
- Origem : Problema de dinâmica de fluidos computacional



Engenharia de Sistemas e Computação PESC/Coppe

Leitura da matriz

- Formato mtx

scipy.io.mmread

`scipy.io.mmread(source)` [source]

Reads the contents of a Matrix Market file-like 'source' into a matrix.

Parameters: **source** : *str or file-like*

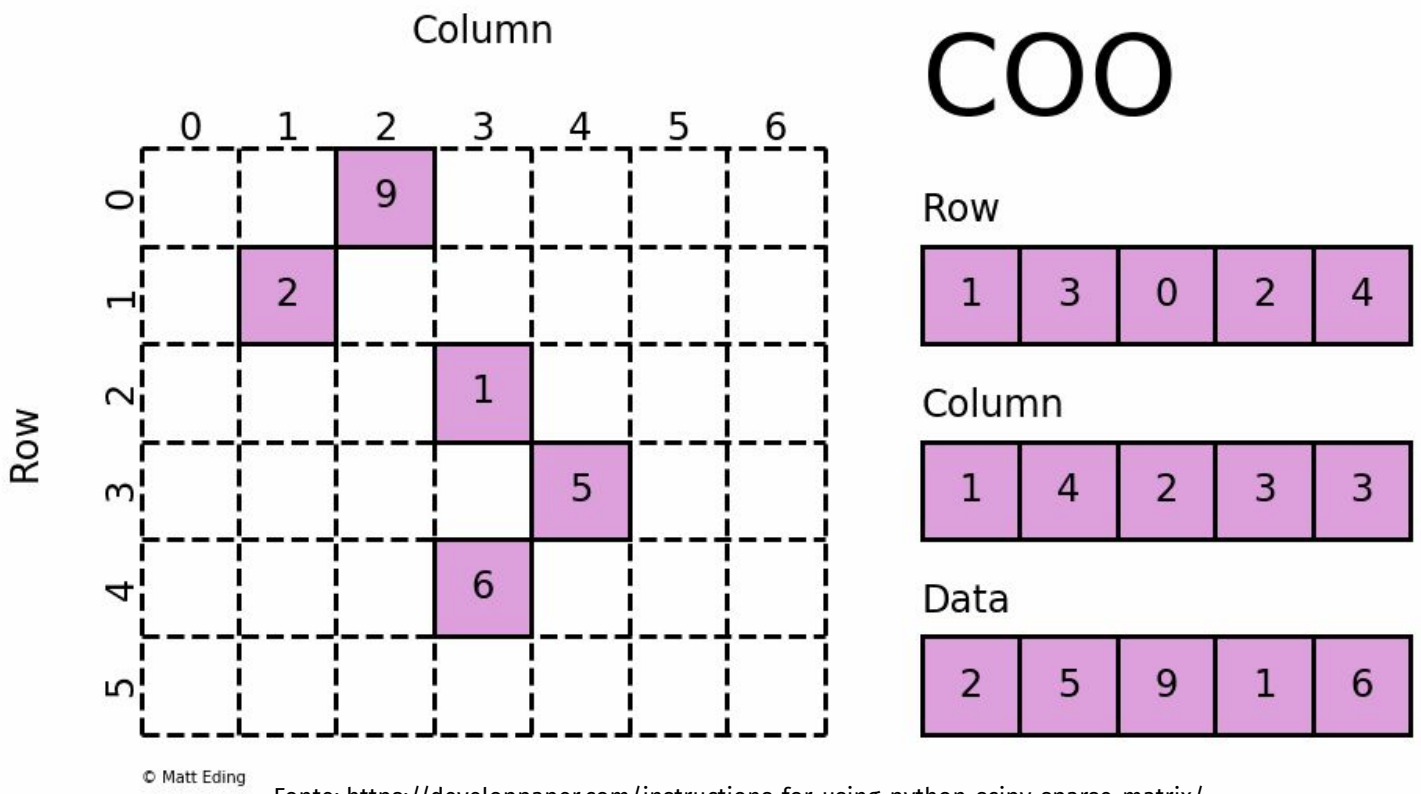
Matrix Market filename (extensions .mtx, .mtz.gz) or open file-like object.

Returns: **a** : *ndarray or coo_matrix*

Dense or sparse matrix depending on the matrix format in the Matrix Market file.

```
A = mmread('matrix\\parabolic_fem\\parabolic_fem.mtx')
A
```

```
<525825x525825 sparse matrix of type '<class 'numpy.float64'>'
        with 3674625 stored elements in COOrdinate format>
```



Fonte: https://developpaper.com/instructions-for-using-python-scipy-sparse-matrix/

scipy.sparse.csr_matrix

class **scipy.sparse.csr_matrix**(*arg1, shape=None, dtype=None, copy=False*)

Compressed Sparse Row matrix

This can be instantiated in several ways:

csr_matrix(S)

with another sparse matrix S (equivalent to S.tocsr())

```
A2 = csr_matrix(A)
A2
```

```
<525825x525825 sparse matrix of type '<class 'numpy.float64'>'
        with 3674625 stored elements in Compressed Sparse Row format>
```

scipy.sparse.csc_matrix

class **scipy.sparse.csc_matrix**(*arg1, shape=None, dtype=None, copy=False*)

Compressed Sparse Column matrix

This can be instantiated in several ways:

csc_matrix(S)

with another sparse matrix S (equivalent to S.tocsc())

```
A3 = csc_matrix(A)
A3
```

```
<525825x525825 sparse matrix of type '<class 'numpy.float64'>'
        with 3674625 stored elements in Compressed Sparse Column format>
```

Engenharia de Sistemas e Computação PESC/Coppe

Autopares

scipy.sparse.linalg.eigsh

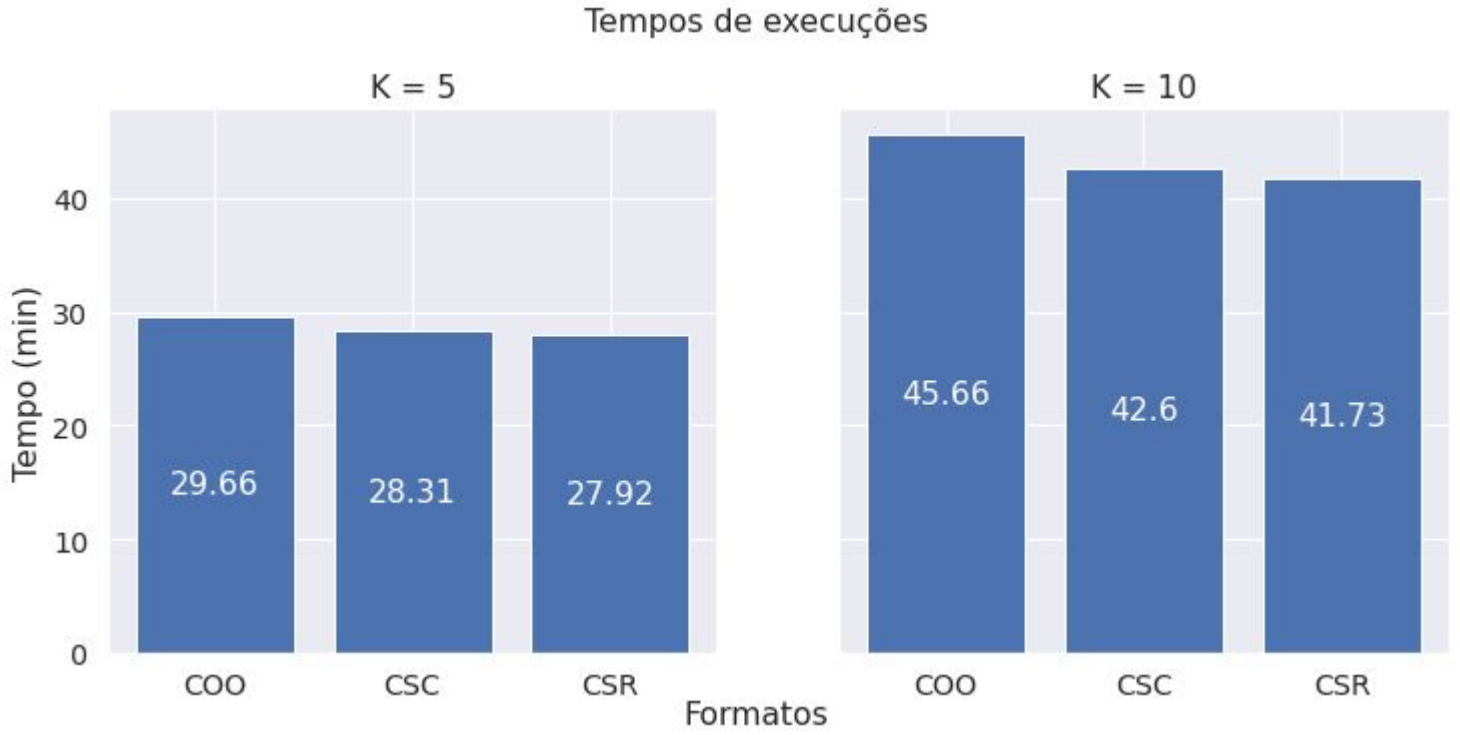
scipy.sparse.linalg.eigsh(*A, k=6, M=None, sigma=None, which='LM', v0=None, ncv=None, maxiter=None, tol=0, return_eigenvectors=True, Minv=None, OPinv=None, mode='normal'*)

Find k eigenvalues and eigenvectors of the real symmetric square matrix or complex Hermitian matrix A. [source]

```
for n in [5,10]:
    print(n)
    np.random.seed(18052022)
    v0 = np.random.rand(min(A.shape))
    start = time.time()
    eigenvalues, eigenvectors = eigsh(A, k=n, v0=v0)
    end = time.time()
    print(end-start)
    np.save("/scratch/rtm-uq/lyncoln.oliveira/lanczos/eigenvalues_COO_"+ str(n) +".npy",eigenvalues)
    np.save("/scratch/rtm-uq/lyncoln.oliveira/lanczos/eigenvectors_COO_"+ str(n) +".npy",eigenvectors)
    np.save("/scratch/rtm-uq/lyncoln.oliveira/lanczos/time_COO_"+ str(n) +".npy",end-start)
```

Resultados

- K = 5 : [0.79998248, 0.7999853 , 0.79998905, 0.79999375, 0.79999657]
- K = 10 : [0.79996651, 0.7999712 , 0.79997401, 0.79997778, 0.79998247,0.79998248, 0.7999853 , 0.79998905, 0.79999375, 0.79999657]



Engenharia de Sistemas e Computação PESC/Coppe

Conclusões

- Economia memória
- Obtenção dos autopares em bom tempo
- Pequena diferença entre os formatos CSR e CSC
- Mais execuções