Universität Regensburg — SS 2016
Institute for Theoretical Physics
Institute of Computational Condensed Matter Theory
Prof. Dr. Ferdinand Evers, Dr. Richard Korytár, B.Sc. Patrick Grössing, **8.4.2016**
`http://www.physik.uni-regensburg.de/forschung/evers/courses/cscn.phtml`

Exercise Sheet No. 0
"Computational Condensed Matter Theory"

$\boxed{1}$ Matrices

a) Initialize a 10×10 tridiagonal matrix D, with the value 2 on the diagonal, and 1 on super- and sub-diagonals. Calculate the trace, determinant, eigenvalues and eigenvectors of the matrix D.

b) Construct a 10×10 band matrix H, with the value 2 on the diagonal, and $H_{k,k+2} = i$, $H_{k,k-2} = -i$. Calculate the transpose and complex transpose matrix of H and show that H is hermitian.

c) Calculate the inverse of D and use the result to solve the equation

$$\hat{D}\vec{x} = \vec{y}, \qquad \vec{y} = (1,2,3,4,5,6,7,8,9,10)^T$$

and calculate $\vec{x}$. Which alternative method does Numpy propose to calculate $\vec{x}$?

Useful functions:

- numpy: `full`, `fill_diagonal`, `trace`, `eye`, `transpose or ndarray.T`, `conjugate or ndarray.conjugate`, `array_equal`, `dot`, `allclose`.
- numpy.linalg: `det`, `eigh`, `inv`

$\boxed{2}$ Slicing and construction of matrices

a) Define two matrices of the following form: (Here j denotes a complex number)

$$\hat{X} = \begin{pmatrix} 0 & 1j \\ -1j & 0 \end{pmatrix} \qquad \hat{Y} = \begin{pmatrix} 2j & 0 \\ 0 & -2j \end{pmatrix}$$

Put each of them into a 8x8 matrix with $\hat{X}$, $\hat{Y}$ on the diagonal. Use those to construct an 800x800 matrix $\hat{H}$ e.g.:

$$\hat{M}_X = \begin{pmatrix} \hat{X} & 0 & 0 & 0 \\ 0 & \hat{X} & 0 & 0 \\ 0 & 0 & \hat{X} & 0 \\ 0 & 0 & 0 & \hat{X} \end{pmatrix} \qquad \hat{H} = \begin{pmatrix} \hat{M}_X & \hat{M}_Y & 0 & \cdots \\ \hat{M}_Y^\dagger & \ddots & \ddots & \cdots \\ 0 & \ddots & \ddots & \ddots \\ \vdots & \vdots & \ddots & \ddots \end{pmatrix}$$

To make sure you put the matrices right print $\hat{M}_X$, $\hat{M}_Y$ and $\hat{M}_Y^\dagger$ from $\hat{H}$ (You have to slice $\hat{H}$ to retrieve the matrices)

***b)** Calculate the eigenvalues and eigenvectors of H and plot the spectrum as a scatterplot. Now filter the first and last entry of each eigenvector and take the norm of those values. Save this in an array and use it as coloring for your scatterplot (This array should have the same length as your eigenvalue array). **Useful functions:**

– numpy: `kron`, `eye`, `array`, `conjugate`, `transpose`

– numpy.linalg: `eigh`, `norm`

$\boxed{3}$ Multiple graphs and animations

a) Visualize the time evolution of an underdamped harmonic oscillator. The amplitude is given by $x(t) = e^{-\zeta t}\cos(\sqrt{1-\zeta^2}t)$. Draw several curves (for different damping rates $0 < \zeta < 1$) in a single graph and add a legend to your plot. Try to place the legend without overlapping with the plot.

***b)** Plot the real part of a complex valued function $f(x,t) = \Re\left[e^{i(x-\omega_0 t)}\right], \omega_0 = 2\pi$ at $t = 0$ using 100 sampling points $0 \leq x_i \leq 2\pi$. Now try to animate the running wave. To achieve this one has to provide a function to `FuncAnimation` as well as an iterator giving the values for which the provided function should be evaluated. If you are stuck search for the `simple_anim` example code, execute it and try to rebuild your code from there. (`http://matplotlib.org/examples/animation/simple_anim.html`)

Useful functions:

– numpy: `mgrid linspace cos sqrt exp`

– matplotlib.pyplot: `plot`, `legend`

– matplotlib.animation: `FuncAnimation`

$\boxed{4}$ Visualization of scalar and vector fields

a) There are many possibilities to visualize two-dimensional scalar fields. Use Matplotlib to visualize the functions

$$\Phi_1(x,y) = xy, \quad \Phi_2(x,y) = \frac{1}{1+r^6}, \quad \Phi_3(x,y) = e^{-r^2} \quad \text{with} \quad r = \sqrt{x^2+y^2}$$

in at least two different ways. Play with different color schemes.

b) 2d vector fields can also be visualized with Pythons Matplotlib. Draw the functions

$$\mathbf{E}_1(x,y) = (-y,x), \quad \mathbf{E}_2(x,y) = -\nabla r, \quad \mathbf{E}_3(x,y) = -(y,x).$$

Useful functions:

– numpy: `mgrid`, `power`, `sqrt`, `exp`

– matplotlib.pyplot: `figure`, `subplot`, `pcolor`, `colorbar`, `show`, `get_cmap`, `contour`

– mpl_toolkits.mplot3d: `add_subplot`, `plot_surface`

$\boxed{*5}$ Functions and scripts

a) The potential of a point charge q at $\mathbf{x_0}$ is given by $\Phi(\mathbf{x} = \frac{q}{|\mathbf{x}-\mathbf{x_0}|}$. Write a function `potential` with the x- and y-coordinates of an grid, as well as the position (x_0, y_0) and charge q of the point charge as arguments. The function should return a (discretized) field of the potential.
The function should be called like e.g.

```
x,y = np.mgrid[-5:0.2:5,-5:0.2:5]
resultat = potential(x,y,-1.5,2.5,1)

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(x, y, resultat, cmap=plt.get_cmap('cool'))
```

to draw the potential of an point charge with $q = 1$ located at the position $(-1.5, 2.5)$.

b) Write a function `potential2` which calculates the potential of a system, now with several point charges. Beside the coordinates of the grid, the function should take a list of coordinates and charges as parameter. This list has the structure of a matrix:

$$\begin{pmatrix} x_1 & y_1 & q_1 \\ x_2 & y_2 & q_2 \\ x_3 & y_3 & q_3 \\ & \vdots & \end{pmatrix}$$

meaning a charge q_1 is at (x_1, y_1), a charge q_2 at (x_2, y_2), etc.

Useful functions:

- numpy: `power`, `sqrt`, `newaxis`, `concatenate`, `putmask`, `empty_like`
- numpy.linalg: `norm`
- numpy.ma: `array`, `masked_less`