In [1]:

```
from sympy import *
```

In [2]:

```
def build_polynomial(matrix, iter):
    x = Symbol('x')
    if iter == -1:
        return 1
    if iter == 0:
        return poly(matrix[1][1] - x)
    return poly(
        (matrix[iter][iter] - x)*build_polynomial(matrix, iter - 1) -
        matrix[iter - 1][iter]*matrix[iter][iter - 1]*build_polynomial(matrix, i
ter - 2))
```

In [3]:

```
def halleys_method(function, initial, precision):
    x = Symbol('x')
    iters = 0

    lfunction = lambdify(x, function, 'numpy')

    dfunction = function.diff(x)
    ldfunction = lambdify(x, dfunction, 'numpy')

    d2function = dfunction.diff(x)
    ld2function = lambdify(x, d2function, 'numpy')

    while abs(lfunction(initial)) > precision:
        t = - lfunction(initial)/ldfunction(initial)
        r = ld2function(initial)*t**2/ldfunction(initial)
        initial = initial + t**2/(t + 0.5*r)
        iters += 1

    residual = lfunction(initial)
    return {'root': initial, 'iterations': iters, 'residual': residual}
```

In [4]:

```
def eigenvalues(matrix, initial, precision):
    x = Symbol('x')
    eigenvalues = {'value':[], 'residual':[]}
    n = len(matrix)
    polynomial = build_polynomial(matrix, n-1)
    for _ in range(n):
        root = halleys_method(polynomial.as_expr(), initial, precision)
        eigenvalues['value'].append(root['root'])
        eigenvalues['residual'].append(root['residual'])
        polynomial = pquo(polynomial, poly(x - root['root']))
    return eigenvalues
```

In [5]:

```
matrix = [[1, 2, 0, 0],
          [3, 1, 2, 0],
          [0, 3, 1, 2],
          [0, 0, 3, 2]]
initial = 0
precision = 0.001
```

In [6]:

```
# результат написанного алгоритма с невязкой для каждого значения
eigenvalues(matrix, initial, precision)
```

Out[6]:

```
{'value': [-0.1881911215025469,
  2.8998266041121847,
  -2.8518950613854894,
  5.14025957877585],
 'residual': [8.34833535634516e-09,
  1.497466683986204e-08,
  -7.894759500359783e-06,
  0.0]}
```

In [7]:

```
import numpy
# результат встроенной функции
numpy.linalg.eigvals(numpy.array(matrix))
```

Out[7]:

```
array([-2.85189605, -0.18819112,  5.14026057,  2.89982661])
```