Numerical Methods

**Report**

Date of the exercise: **25/04/2019**

Exercise: **Determining eigenvalues and eigenvectors of matrix**

Group: 2, Team:

Subsection (names):

1. Mateusz Nowotnik

2. Dawid Tomala

1. Krylov method

The matrix we have chosen:

Taking as the initial vector , yields

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |

From that we can create a system of linear equations

Solving the system, we obtained the following characteristic polynomial coefficients:

The characteristic equation of matrix **A** is of the form:

Solving that polynomial, we obtained:

Calculating coefficients for :

and setting

Using the formula

Eigenvector was obtained.

|  |
| --- |
|  |
|  |

2. Power method

For the program to work properly - a matrix should be 3x3 and its rank (number of linearly independent eigenvectors) equal to 3.

1. #include <iostream>
2. **using** **namespace** std;
4. // initialize the initial vector
5. **void** initVec(**int** v, **double** y[1][3])
6. {
7. **switch**(v) {
8. **case** 1:
9. y[0][0] = 1; y[0][1] = 0; y[0][2] = 0;
10. **break**;
11. **case** 2:
12. y[0][0] = 0; y[0][1] = 1; y[0][2] = 0;
13. **break**;
14. **case** 3:
15. y[0][0] = 0; y[0][1] = 0; y[0][2] = 1;
16. **break**;
17. **default**:
18. cout << "Something's wrong!" << endl;
19. }
20. }
22. **int** main()
23. {
24. **double** A[3][3];
25. **double** y[9][3] = {0};
26. **double** eigenvalue, quot[3];
27. **bool** oscQuot = **true**;
28. **int** v = 0;
30. cout << "3x3 matrix A:" << endl;

33. **for** (**int** i = 0; i < 3; i++)
34. **for** (**int** j = 0; j < 3; j++)
35. cin >> A[i][j];
37. **while** (oscQuot == **true**){
38. // initial vector y(0)
39. initVec(++v, y);
41. // obtaining the iterations of vector 'y'
42. **for** (**int** i = 1; i < 9; i++)
43. **for** (**int** j = 0; j < 3; j++)
44. **for** (**int** k = 0; k < 3; k++)
45. y[i][j] += y[i-1][k]\*A[k][j];
47. // calculating the quotients
48. quot[0] = y[8][0]/y[7][0];
49. quot[1] = y[8][1]/y[7][1];
50. quot[2] = y[8][2]/y[7][2];
52. // changing the value of initial vector if the values of quotients are oscillating
53. **if** (quot[0] != quot[2])
54. oscQuot = **false**;
55. }
57. // calculating the eigenvalue as the arithmetic mean of all quotients
58. eigenvalue = (quot[0] + quot[1] + quot[2])/3.0;
60. /\*
61. // Krylov
62. for (int i = 0; i < 3; i++) {
63. cout << endl;
64. for (int j = 0; j < 4; j++)
65. cout << y[j][i] << " ";
66. }
67. cout << endl << endl;
68. \*/
70. cout << "\nHighest modulus eigenvalue: " << eigenvalue << endl;
72. cout << "\nHighest corresponding eigenvector: " << endl;
73. **for** (**int** i = 0; i < 3; i++)
74. cout << y[8][i] << endl;
76. cout << "\nAfter normalization: " << endl;
77. cout << y[8][0]/ max(max(y[8][0],y[8][1]),y[8][2]) << endl;
78. cout << y[8][1]/ max(max(y[8][0],y[8][1]),y[8][2]) << endl;
79. cout << y[8][2]/ max(max(y[8][0],y[8][1]),y[8][2]) << endl;
81. **return** 0;
82. }

Determining the highest-modulus eigenvalue and corresponding eigenvector of the given matrix:

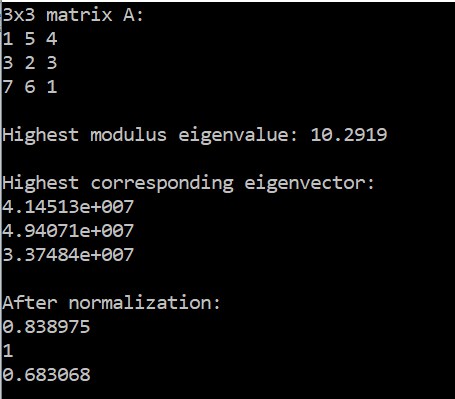
Input to the program:

1 5 4

3 2 3

7 6 1

The input/output from the console:



Obtained highest modulus eigenvalue: **10.2919**

Obtained corresponding eigenvector: