Numerical Approximation of Eigenvalues

A Comparison of Power and Jacobi Methods

Gage Golish
Department of Mathematics and Computer Science
Indiana State Univeristy

Fall 2019

Contents

1 Power Method

The power method is an iterative method that numerically approximates the largest in magnitude eigenvalue and its corresponding eigenvector of a linear system. The only requirement is that the eigenvalues of the matrix be real. Although the matrix does not have to also be symmetric, we test the algorithm only on symmetric matrices for comparison purposes with the Jacobi method.

The algorithm is shown below. It was implemented as a generator function, therefore the stopping condition is left up to the calling function. This will be demonstrated in the results section. Each iteration takes $O(n^2)$ time due to the matrix multiplication on line 12.

eigen.py (8-16)

```
8  def power_method(A):
9      x = np.zeros(A.shape[0])
10     x[0] = 1.0
11     while True:
12         p = A @ x
13         n = np.max(np.abs(p))
14         x = (1.0 / n) * p
15         yield n
16
```

2 Jacobi Method

The Jacobi method is also an iterative method, but it approximates all eigenvalues of the system; however, for the purposes of comparison, we will only consider the case of the largest eigenvalue. Also the Jacobi method requires a symmetric matrix. The functions below are subroutines of the main Jacobi algorithm. `outer_norm` calculates the $N(A)$ and `outer_argmax` finds the largest in magnitude non-diagonal entry in the matrix. Each iteration of Jacobi costs $O(n^2)$ due to calling `outer_argmax` on line 22. Note that Jacobi is also implemented as a generator and its stopping condition is left up to the calling function.

utils.py (6-10)

```
6  def outer_norm(A):
7      lt = np.tril_indices(A.shape[0], -1)
8      ut = np.triu_indices(A.shape[0], 1)
9      return np.sum(np.square(A[lt])) + np.sum(np.square(A[ut]))
10
```

utils.py (13-17)

```
13 def outer_argmax(A):
14     B = np.abs(A.copy())
```

```
15      B[np.diag_indices(B.shape[0])] = 0
16      return np.unravel_index(B.argmax(), B.shape)
17
```

eigen.py (18-41)

```
18  def jacobi_method(A):
19      norm = utils.outer_norm(A)
20      yield norm
21      while True:
22          i, j = utils.outer_argmax(A)
23          if i != 0 or j != 0:
24              a = A.copy()
25              theta = 0.5 * (A[i, i] - A[j, j]) / A[i, j]
26              t = utils.sign(theta) / (abs(theta) + math.sqrt(theta**2 + 1))
27              c = 1 / math.sqrt(t**2 + 1)
28              s = c * t
29              a[i, j] = a[j, i] = 0
30              a[i, i] = A[i, i] + t * A[i, j]
31              a[j, j] = A[j, j] - t * A[i, j]
32              for l in range(A.shape[0]):
33                  if l != i and l != j:
34                      a[i, l] = a[l, i] = c * A[i, l] + s * A[j, l]
35                      a[j, l] = a[l, j] = c * A[j, l] - s * A[i, l]
36              norm -= 2 * A[i, j]**2
37              A[:] = a
38              yield norm
39          else:
40              yield 0
41
```

3 Results

In order to compare the methods, a 10x10 symmetric matrix with random values between 0 and 20 is generated. The matrix is listed below and is referred to as test case 3 for the purposes of the experiment. The maximum eigenvalue of the matrix is found to be 102.85. The code for running the experiment for each method is shown below.

tests.py (34-43)

```
34  def run_power(A, threshold=0.0001):
35      data = []
36      prev = float("inf")
37      for approx in eigen.power_method(A):
```

```
38          data.append(approx)
39          if abs(approx - prev) < threshold:
40              break
41          prev = approx
42      return data
43
```

```
.............................. tests.py (44-54) ..............................
44  def run_jacobi(A, threshold=0.0001):
45      data = []
46      prev = float("inf")
47      for norm in eigen.jacobi_method(A):
48          data.append(norm)
49          eigenval = max(np.abs(np.diag(A)))
50          if abs(eigenval - prev) < threshold:
51              break
52          prev = eigenval
53      return data
54
```

The threshold value, ϵ, is used in both iterations as the stopping condition. Its default value is $\epsilon = 0.0001$. For the power method, ϵ is interpreted as the accuracy up to a certain number of decimal places. For the Jacobi method, it serves as the accuracy to certain number of decimal places for the largest in magnitude eigenvalue on the diagonal of $A^{(m)}$. Thus the iteration will stop once the largest eigenvalue has been approximated to the same degree of accuracy as in the power method, regardless of how well the other eigenvalues have been approximated.

$$A = \begin{bmatrix} 3.16 & 9.52 & 16.81 & 17.04 & 7.27 & 8.56 & 2.7 & 9.19 & 11.82 & 9.4 \\ 9.52 & 2.74 & 12.47 & 15.97 & 4.54 & 18.59 & 3.57 & 12.65 & 15.87 & 7.27 \\ 16.81 & 12.47 & 1.72 & 8.25 & 12.87 & 7.7 & 4.49 & 3.6 & 8.99 & 8.75 \\ 17.04 & 15.97 & 8.25 & 4.86 & 12.4 & 6.68 & 13.69 & 10.03 & 5.3 & 10.58 \\ 7.27 & 4.54 & 12.87 & 12.4 & 9.1 & 13.06 & 11.31 & 17.05 & 1.86 & 15.18 \\ 8.56 & 18.59 & 7.7 & 6.68 & 13.06 & 6.11 & 13.74 & 4.83 & 5.24 & 10.14 \\ 2.7 & 3.57 & 4.49 & 13.69 & 11.31 & 13.74 & 11.17 & 14.77 & 11.98 & 5.84 \\ 9.19 & 12.65 & 3.6 & 10.03 & 17.05 & 4.83 & 14.77 & 18.12 & 5.32 & 11.24 \\ 11.82 & 15.87 & 8.99 & 5.3 & 1.86 & 5.24 & 11.98 & 5.32 & 15.37 & 11.81 \\ 9.4 & 7.27 & 8.75 & 10.58 & 15.18 & 10.14 & 5.84 & 11.24 & 11.81 & 8.48 \end{bmatrix}$$

Refer to the figure below to see the results of the experiment. As you can see, the power method approximated the eigenvalue to within ϵ in 13 iterations, where Jacobi only took 10. The difference becomes much larger as the required degree of accuracy is increased. In the second figure, we see that the power method took 25 iterations where Jacobi still only took 10 when $\epsilon = 0.0000000001$.

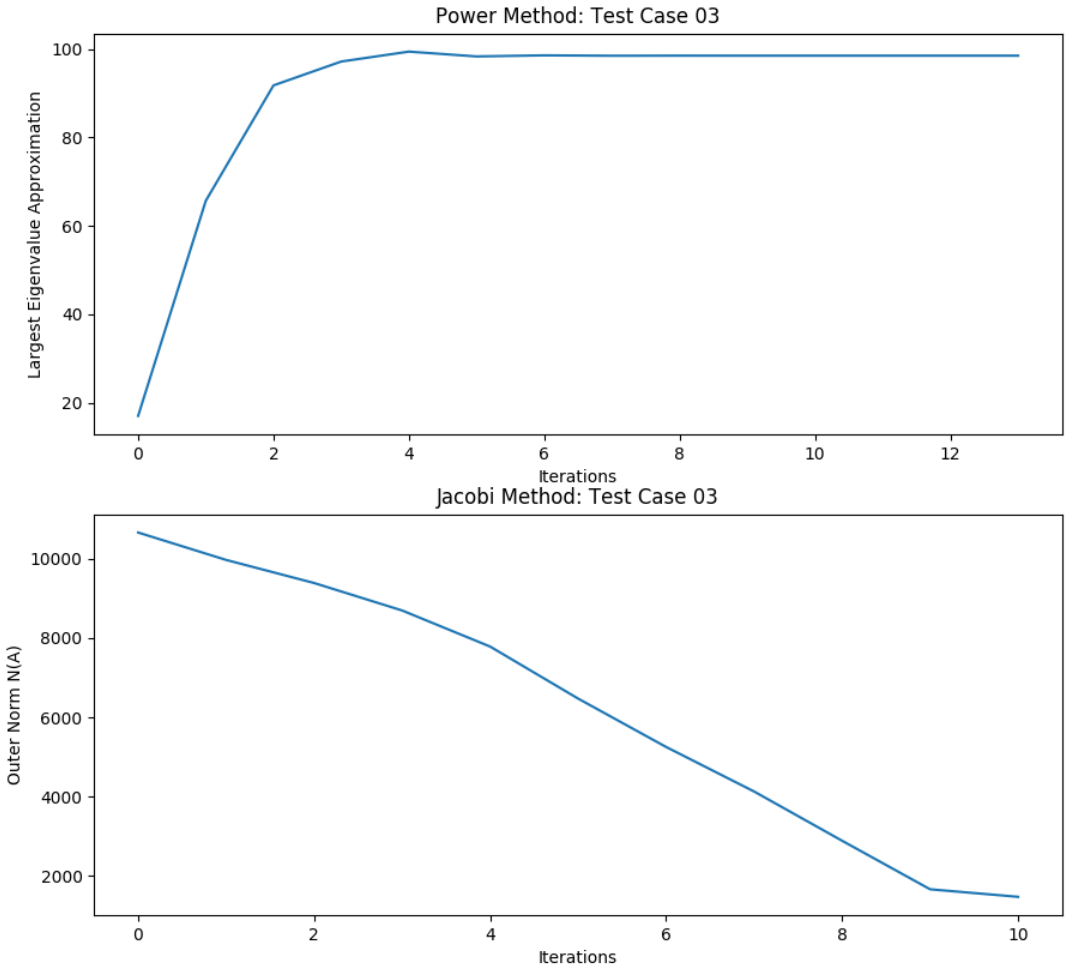

Figure 1: Plot of iterations for both Jacobi and power methods, $\epsilon = 0.0001$.

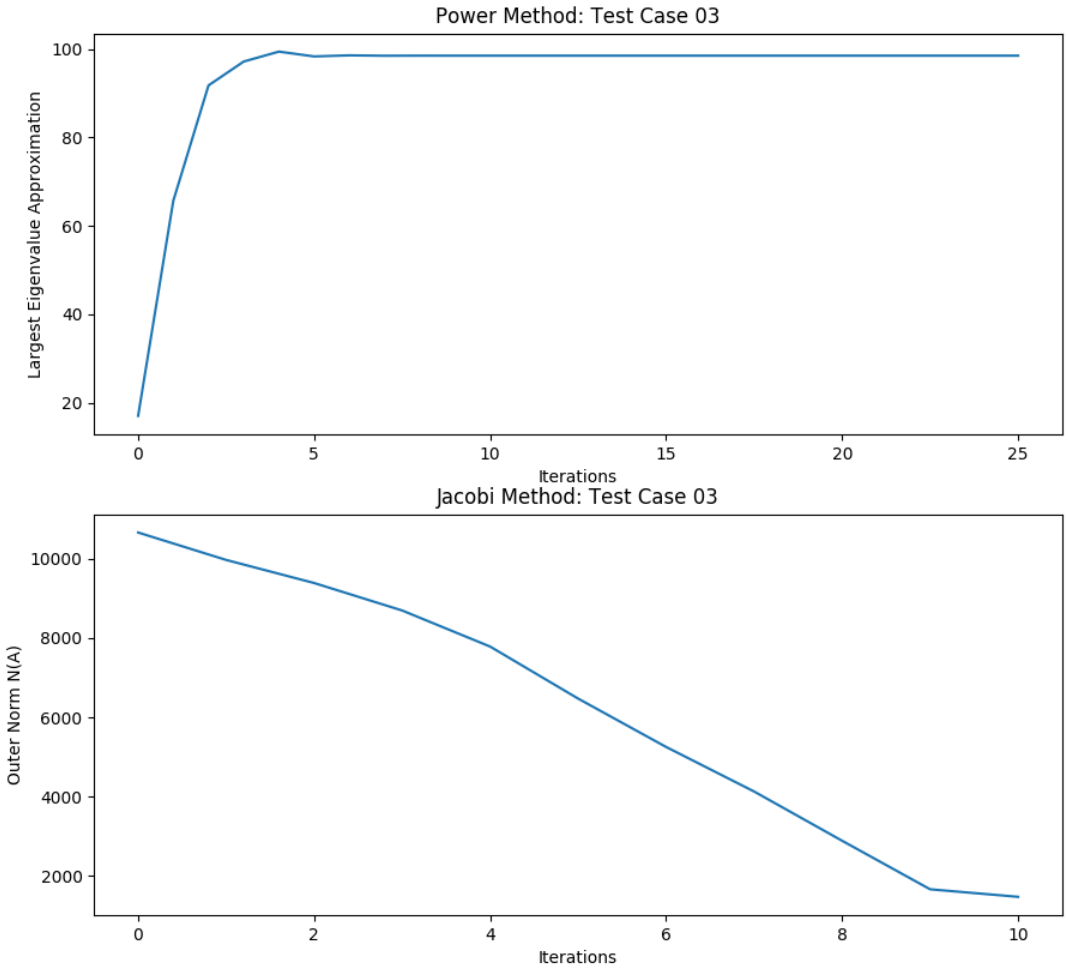

Figure 2: Plot of iterations for both Jacobi and power methods, $\epsilon = 1 \times 10^{-10}$.