



WIKIPEDIA
The Free Encyclopedia

[Main page](#)
[Contents](#)
[Featured content](#)
[Current events](#)
[Random article](#)
[Donate to Wikipedia](#)
[Wikipedia store](#)

Interaction
[Help](#)
[About Wikipedia](#)
[Community portal](#)
[Recent changes](#)
[Contact page](#)

Tools
[What links here](#)
[Related changes](#)
[Upload file](#)
[Special pages](#)
[Permanent link](#)
[Page information](#)
[Wikidata item](#)
[Cite this page](#)


Print/export
[Create a book](#)
[Download as PDF](#)
[Printable version](#)

Languages
[Deutsch](#)
[日本語](#)
[Edit links](#)

[Create account](#) [Log in](#)

Article [Talk](#)

[Read](#) [Edit](#) [View history](#)

Search 

Inverse iteration

From Wikipedia, the free encyclopedia

In **numerical analysis**, **inverse iteration** is an **iterative eigenvalue algorithm**. It allows one to find an approximate **eigenvector** when an approximation to a corresponding **eigenvalue** is already known. The method is conceptually similar to the **power method** and is also known as the **inverse power method**. It appears to have originally been developed to compute resonance frequencies in the field of structural mechanics. ^[1]

The inverse power iteration algorithm starts with an approximation μ for the **eigenvalue** corresponding to the desired **eigenvector** and a vector b_0 , either a randomly selected vector or an approximation to the eigenvector. The method is described by the iteration

$$b_{k+1} = \frac{(A - \mu I)^{-1} b_k}{C_k},$$

where C_k are some constants usually chosen as $C_k = \|(A - \mu I)^{-1} b_k\|$. Since eigenvectors are defined up to multiplication by constant, the choice of C_k can be arbitrary in theory; practical aspects of the choice of C_k are discussed below.

At every iteration, the vector b_k is multiplied by the inverse of the matrix $(A - \mu I)$ and normalized. It is exactly the same formula as in the **power method**, except replacing the matrix A by $(A - \mu I)^{-1}$. The closer the approximation μ to the eigenvalue is chosen, the faster the algorithm converges; however, incorrect choice of μ can lead to slow convergence or to the convergence to an eigenvector other than the one desired. In practice, the method is used when a good approximation for the eigenvalue is known, and hence one needs only few (quite often just one) iterations.

Contents [hide]

- 1 Theory and convergence
 - 1.1 Speed of convergence
 - 1.2 Complexity
- 2 Implementation options
 - 2.1 Calculate inverse matrix or solve system of linear equations
 - 2.2 Tridiagonalization, Hessenberg form
 - 2.3 Choice of the normalization constant C_k
- 3 Usage
- 4 See also
- 5 References
- 6 External links

Theory and convergence [\[edit\]](#)

The basic idea of the **power iteration** is choosing an initial vector b (either an **eigenvector** approximation or a **random** vector) and iteratively calculating Ab, A^2b, A^3b, \dots . Except for a set of zero **measure**, for any initial vector, the result will converge to an **eigenvector** corresponding to the dominant **eigenvalue**.

The inverse iteration does the same for the matrix $(A - \mu I)^{-1}$, so it converges to eigenvector corresponding to the dominant eigenvalue of the matrix $(A - \mu I)^{-1}$. Eigenvalues of this matrix are $(\lambda_1 - \mu)^{-1}, \dots, (\lambda_n - \mu)^{-1}$, where λ_i are eigenvalues of A . The largest of these numbers corresponds to the smallest of $(\lambda_1 - \mu), \dots, (\lambda_n - \mu)$. It is obvious that the eigenvectors of A and of $(A - \mu I)^{-1}$ are the same.

Conclusion: The method converges to the eigenvector of the matrix A corresponding to the closest eigenvalue to μ .

In particular taking $\mu = 0$ we see that $(A)^{-k}b$ converges to the eigenvector corresponding to the eigenvalue of A with the smallest absolute value.

Speed of convergence [\[edit\]](#)

Let us analyze the [rate of convergence](#) of the method.

The [power method](#) is known to [converge linearly](#) to the limit, more precisely:

$$\text{Distance}(b^{\text{ideal}}, b_{\text{Power Method}}^k) = O\left(\left|\frac{\lambda_{\text{subdominant}}}{\lambda_{\text{dominant}}}\right|^k\right),$$

hence for the inverse iteration method similar result sounds as:

$$\text{Distance}(b^{\text{ideal}}, b_{\text{Inverse iteration}}^k) = O\left(\left|\frac{\mu - \lambda_{\text{closest to } \mu}}{\mu - \lambda_{\text{second closest to } \mu}}\right|^k\right).$$

This is a key formula for understanding the method's convergence. It shows that if μ is chosen close enough to some eigenvalue λ , for example $\mu - \lambda = \epsilon$ each iteration will improve the accuracy $|\epsilon|/|\lambda + \epsilon - \lambda_{\text{closest to } \lambda}|$ times. (We use that for small enough ϵ "closest to μ " and "closest to λ " is the same.) For small enough $|\epsilon|$ it is approximately the same as $|\epsilon|/|\lambda - \lambda_{\text{closest to } \lambda}|$. Hence if one is able to find μ , such the ϵ will be small enough, then very few iterations may be satisfactory.

Complexity [\[edit\]](#)

The inverse iteration algorithm requires solving a [linear system](#) or calculation of the inverse matrix. For non-structured matrices (not sparse, not Toeplitz,...) this requires $O(n^3)$ operations.

Implementation options [\[edit\]](#)

The method is defined by the formula:

$$b_{k+1} = \frac{(A - \mu I)^{-1} b_k}{C_k},$$

There are, however, multiple options for its implementation.

Calculate inverse matrix or solve system of linear equations [\[edit\]](#)

We can rewrite the formula in the following way:

$$(A - \mu I)b_{k+1} = \frac{b_k}{C_k},$$

emphasizing that to find the next approximation b_{k+1} we may solve a system of linear equations. There are two options: one may choose an algorithm that solves a linear system, or one may calculate the inverse $(A - \mu I)^{-1}$ and then apply it to the vector. Both options have complexity $O(n^3)$, the exact number depends on the chosen method.

The choice depends also on the number of iterations. Naively, if at each iteration one solves a linear system, the complexity will be $k \cdot O(n^3)$, where k is number of iterations; similarly, calculating the inverse matrix and applying it at each iteration is of complexity $k \cdot O(n^3)$. Note, however, that if the eigenvalue estimate μ remains constant, then we may reduce the complexity to $O(n^3) + k \cdot O(n^2)$ with either method. Calculating the inverse matrix once, and storing it to apply at each iteration is of complexity $O(n^3) + k \cdot O(n^2)$. Storing an [LU decomposition](#) of $(A - \mu I)$ and using [forward and back substitution](#) to solve the system of equations at each iteration is also of complexity $O(n^3) + k \cdot O(n^2)$.

Inverting the matrix will typically have a greater initial cost, but lower cost at each iteration. Conversely, solving systems of linear equations will typically have a lesser initial cost, but require more operations for each iteration.

Tridiagonalization, [Hessenberg form](#) [\[edit\]](#)

If it is necessary to perform many iterations (or few iterations, but for many eigenvectors), then it might be wise to bring the matrix to the upper [Hessenberg form](#) first (for symmetric matrix this will be [tridiagonal form](#)). Which costs $\frac{10}{3}n^3 + O(n^2)$ arithmetic operations using a technique based on [Householder reduction](#), with a finite sequence of orthogonal similarity transforms, somewhat like a two-sided QR decomposition.^{[2][3]} (For QR decomposition, the Householder rotations are multiplied only on the left, but for the Hessenberg case they are multiplied on both left and right.) For [symmetric matrices](#) this procedure costs $\frac{4}{3}n^3 + O(n^2)$ arithmetic operations using a technique based on Householder reduction.^{[2][3]}

Solution of the system of linear equations for the [tridiagonal matrix](#) costs $O(n)$ operations, so the complexity grows like $O(n^3) + k \cdot O(n)$, where k is the iteration number, which is better than for the direct inversion. However

for few iterations such transformation may not be practical.

Also transformation to the [Hessenberg form](#) involves square roots and the division operation, which are not universally supported by hardware.

Choice of the normalization constant C_k [\[edit\]](#)

On general purpose processors (e.g. produced by Intel) the execution time of addition, multiplication and division is approximately equal. But on embedded and/or low energy consuming hardware ([digital signal processors](#), [FPGA](#), [ASIC](#)) division may not be supported by hardware, and so should be avoided. Choosing $C_k=2^{n_k}$ allows fast division without explicit hardware support, as division by a power of 2 may be implemented as either a [bit shift](#) (for [fixed-point arithmetic](#)) or subtraction of k from the exponent (for [floating-point arithmetic](#)).

When implementing the algorithm using [fixed-point arithmetic](#), the choice of the constant C_k is especially important. Small values will lead to fast growth of the norm of b_k and to [overflow](#); large values of C_k will cause the vector b_k to tend toward zero.

Usage [\[edit\]](#)

The main application of the method is the situation when an approximation to an eigenvalue is found and one needs to find the corresponding approximate eigenvector. In such situation the inverse iteration is the main and probably the only method to use. So typically the method is used in combination with some other methods which allows to find approximate eigenvalues: the standard example is the [bisection eigenvalue algorithm](#), another example is the [Rayleigh quotient iteration](#) which is actually the same inverse iteration with the choice of the approximate eigenvalue as the [Rayleigh quotient](#) corresponding to the vector obtained on the previous step of the iteration.

There are some situations where the method can be used by itself, however they are quite marginal.

Dominant eigenvector. The dominant eigenvalue can be easily estimated for any matrix. For any [induced norm](#) it is true that $\|A\| \geq |\lambda|$, for any eigenvalue λ . So taking the norm of the matrix as an approximate eigenvalue one can see that the method will converge to the dominant eigenvector.

Estimates based on statistics. In some real-time applications one needs to find eigenvectors for matrices with a speed may be millions matrices per second. In such applications typically the statistics of matrices is known in advance and one can take as approximate eigenvalue the average eigenvalue for some large matrix sample, or better one calculates the mean ratio of the eigenvalue to the trace or the norm of the matrix and eigenvalue is estimated as trace or norm multiplied on the average value the their ratio. Clearly such method can be used with much care and only in situations when the mistake in calculations is allowed. Actually such idea can be combined with other methods to avoid too big errors.

See also [\[edit\]](#)

- [Power iteration](#)
- [Rayleigh quotient iteration](#)
- [List of eigenvalue algorithms](#)

References [\[edit\]](#)

1. [^] Ernst Pohlhausen, *Berechnung der Eigenschwingungen statisch-bestimmter Fachwerke*, ZAMM - Zeitschrift für Angewandte Mathematik und Mechanik 1, 28-42 (1921).
2. [^] ^a ^b Demmel, James W. (1997), *Applied Numerical Linear Algebra*, Philadelphia, PA: [Society for Industrial and Applied Mathematics](#), ISBN 0-89871-389-7, [MR 1463942](#) [↗](#).
3. [^] ^a ^b Lloyd N. Trefethen and David Bau, *Numerical Linear Algebra* (SIAM, 1997).

External links [\[edit\]](#)

- [Inverse Iteration to find eigenvectors](#) [↗](#), physics.arizona.edu
- [The Power Method for Eigenvectors](#) [↗](#), math.fullerton.edu

v · t · e	Numerical linear algebra [hide]
Key concepts	Floating point · Numerical stability
Problems	Matrix multiplication (algorithms) · Matrix decompositions · Linear equations · Sparse problems
Hardware	CPU cache · TLB · Cache-oblivious algorithm · SIMD · Multiprocessing
Software	BLAS · Specialized libraries · General purpose software

This page was last modified on 16 June 2015, at 15:03.

Text is available under the [Creative Commons Attribution-ShareAlike License](#); additional terms may apply. By using this site, you agree to the [Terms of Use](#) and [Privacy Policy](#). Wikipedia® is a registered trademark of the [Wikimedia Foundation, Inc.](#), a non-profit organization.

[Privacy policy](#) [About Wikipedia](#) [Disclaimers](#) [Contact Wikipedia](#) [Developers](#) [Mobile view](#)

