FFTs in Graphics and Vision

Fast String Matching
and
Math Review

Fast Pattern Matching in Strings
Knuth *et al.*, 1977

Outline

Fast Substring Matching

Math Review

- Complex Numbers
- Vector Spaces
- Linear Operators

Fast Substring Matching

Challenge:

Given strings *S* and *T*, find all occurrences of *T* as a substring of *S*.

Fast Substring Matching

Challenge:

Given strings *S* and *T*, find all occurrences of *T* as a substring of *S*.

Example:

S=ACDBEFCDBE *T*=CDB

Fast Substring Matching

<u>Challenge</u>:

Given strings *S* and *T*, find all occurrences of *T* as a substring of *S*.

<u>Brute Force</u>:

- For each position in *S*:
  - Test if the next $|T|$ letters in *S* match those in *T*

S=ACDBEFCDBE *T*=CDB

CDB

$$O(|S|*|T|)$$

Fast Substring Matching

Challenge:

Given strings *S* and *T*, find all occurrences of *T* as a substring of *S*.

Brute Force:

- For each position in *S*:
  - Test if the next |*T*| letters in *S* match those in *T*

Can we do this more efficiently?

Fast Substring Matching

Challenge:

Given strings *S* and *T*, find all occurrences of *T* as a substring of *S*.

Observation:

On a failed match, we don't have to compare all |*T*| letters in *T*:

S=ACDBEFCDBE   *T*=CDB

CDB CDB CDB CDB   **Comparisons: 3**

Fast Substring Matching

<u>Challenge</u>:

Given strings *S* and *T*, find all occurrences of *T* as a substring of *S*.

<u>Observation</u>:

What if the situation is more complex?

S=AAAAAAAAAB

AAAB

T=AAAB

Comparisons: 4

Fast Substring Matching

<u>Challenge</u>:

Given strings *S* and *T*, find all occurrences of *T* as a substring of *S*.

<u>Knuth *et al*. (1977)</u>:

On a failed match, we don't have to re-start the matching.

S=AAAAAAAAAB *T*=AAAB

AAABBBBBBB

$O(|S|+|T|)$

Fast Substring Matching

Challenge:

Given strings *S* and *T*, find all occurrences of *T* as a substring of *S*.

Knuth *et al*. (1977):

On a failed match, we don't have to re-start the matching.

The key is to know where in *T* we have to start comparing.

Fast Substring Matching

Challenge:

Given strings *S* and *T*, find all occurrences of *T* as a substring of *S*.

Knuth *et al*. (1977):

The size of the shift on a mismatch is determined by the repetitions in *T*, is independent of *S*, and can be computed in O(|*T*|) time.

For more details, see:

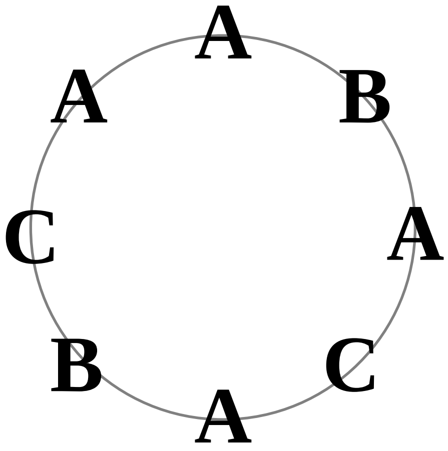
Fast Pattern Matching in Strings.

Fast Substring Matching

Applications:

If we think of a string as a signal on a circle:

S=ABACABAC



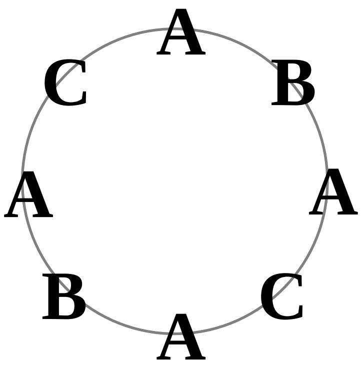
Fast Substring Matching

Applications:

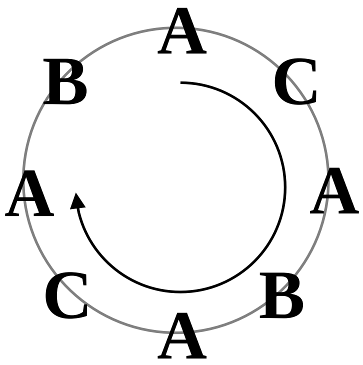
If we think of a string as a signal on a circle:

- We can test if signal *T* is a rotation of *S* by testing if *T* is a substring of *SS*

S=ABACABAC



T=ACABACAB



SS=ABACABACABACABAC
T= ACABACAB

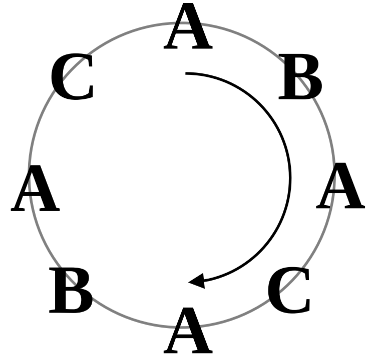
Fast Substring Matching

Applications:

If we think of a string as a signal on a circle:

- We can test if signal *T* is a rotation of *S* by testing if *T* is a substring of *SS*
- We can test if *S* has rotational symmetry by testing if *S* is a substring of *SS*

S=ABACABAC



SS=ABACABACABACABAC

T=ABACABAC

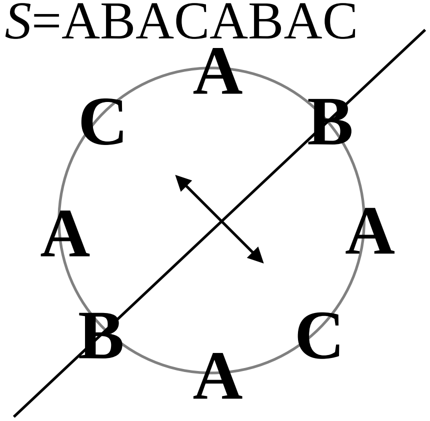
Fast Substring Matching

Applications:

If we think of a string as a signal on a circle:

- We can test if signal *T* is a rotation of *S* by testing if *T* is a substring of *SS*:
- We can test if *S* has rotational symmetry by testing if *S* is a substring of *SS*.
- We can test if *S* has reflective symmetry by testing if *S* is a substring of $(SS)^t$

S=ABACABAC



$(SS)^t$=CABACABACABACABA
T= ABACABAC

Fast Substring Matching

Advantages:

- A fast (linear time) algorithm for performing pattern detection on discrete signals.

Disadvantages:

- Can only tell us if there is a perfect match
  - We need a continuous measure of similarity for real-world data
- Only works for signals on a circle (or a line)
  - Hard to generalize to signals on more complex / interesting domains

Outline

Fast Substring Matching

Math Review

- Complex Numbers
- Vector Spaces
- Linear Operators

Complex Numbers

- A complex number *c* is any number that can be written as:

$$c = a + ib$$

where *a* and *b* are real numbers and *i* is the square root of -1:

$$i^2 = -1$$

Complex Numbers

Given two complex numbers, $c_1=a_1+ib_1$ and $c_2=a_2+ib_2$:

- The sum of the numbers is:

$$c_1+c_2=(a_1+a_2)+i(b_1+b_2)$$

Complex Numbers

Given two complex numbers, $c_1=a_1+ib_1$ and $c_2=a_2+ib_2$:

- The sum of the numbers is:

$$c_1+c_2=(a_1+a_2)+i(b_1+b_2)$$

- The product of the numbers is:

$$\begin{aligned} c_1c_2 &= (a_1+ib_1)(a_2+ib_2) \\ &= a_1a_2+ib_1ib_2+a_1ib_2+ib_1a_2 \\ &= (a_1a_2-b_1b_2)+i(a_1b_2+b_1a_2) \end{aligned}$$

Complex Numbers

Given a complex numbers, $c=a+ib$:

- The negation of the number is:

$$-c=-a-ib$$

Complex Numbers

Given a complex numbers, $c=a+ib$:

- The negation of the number is:

$$-c = -a - ib$$

- The conjugate of the number is:

$$\bar{c} = a - ib$$

Complex Numbers

Given a complex numbers, $c=a+ib$:

- The negation of the number is:

$$-c=-a-ib$$

- The conjugate of the number is:

$$\overline{c}=a-ib$$

- The reciprocal of the number is:

$$\frac{1}{c}=\frac{1}{c}\frac{\overline{c}}{\overline{c}}=\left(\frac{a}{a^2+b^2}\right)-i\left(\frac{b}{a^2+b^2}\right)$$

Complex Numbers

Why do we care?

Complex Numbers

Why do we care?

Fundamental Theorem of Algebra

Given any polynomial:

$$p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$

there always exists a complex number c_0 s.t.:

$$p(c_0) = 0$$

Vector Spaces

A (real/complex) vector space V is a <u>set</u> of elements $v \in V$, with:

- An <u>addition</u> operator "+", and
- A <u>scaling</u> operator "·"

(i.e. we can add any two vectors together to get a vector and if we scale a vector by a number we also get a vector.)

Vector Spaces (Formal Properties 1)

For all u, v, and w in V:

- Associative Addition:

  $$(u+v)+w=u+(v+w)$$

- Commutative Addition:

  $$u+v=v+u$$

- Additive Identity:
  There exists a unique vector 0 in V such that:

  $$u+0=u$$

- Additive Inverse:
  There exists a vector $(-u)$ in V such that:

  $$u+(-u)=0$$

Vector Spaces (Formal Properties 2)

For all u, and v in V, and all (real / complex) scalars a and b:

- Distributive over vector addition:

  $$a(u+v) = (au)+(av)$$

- Distributive over scalar addition:

  $$(a+b)u=(au)+(bu)$$

- Compatible scalar multiplication:

  $$a(bu) = (ab)u$$

- Scalar Identity:

  $$1u=u$$

Vector Spaces: Examples

Real Vector Spaces:

- The real / complex numbers
- The space of *n*-dimensional arrays with real / complex entries
- The space of *mxn* matrices with real / complex entries
- The space of real / complex valued functions on a circle / line / plane / sphere / etc.

Complex Vector Spaces:

- The complex numbers
- The space of *n*-dimensional arrays with complex entries
- The space of *mxn* matrices with complex entries
- The space of complex valued functions on a circle / line / plane / sphere / etc.

Vector Space Basis

A basis of V is a finite set $\{v_1,\ldots,v_n\}$ of vectors such that:

1. Any vector v in V can be expressed as:

$$v = a_1 v_1 + \cdots + a_n v_n$$

   where the a_i are (real / complex) scalars.

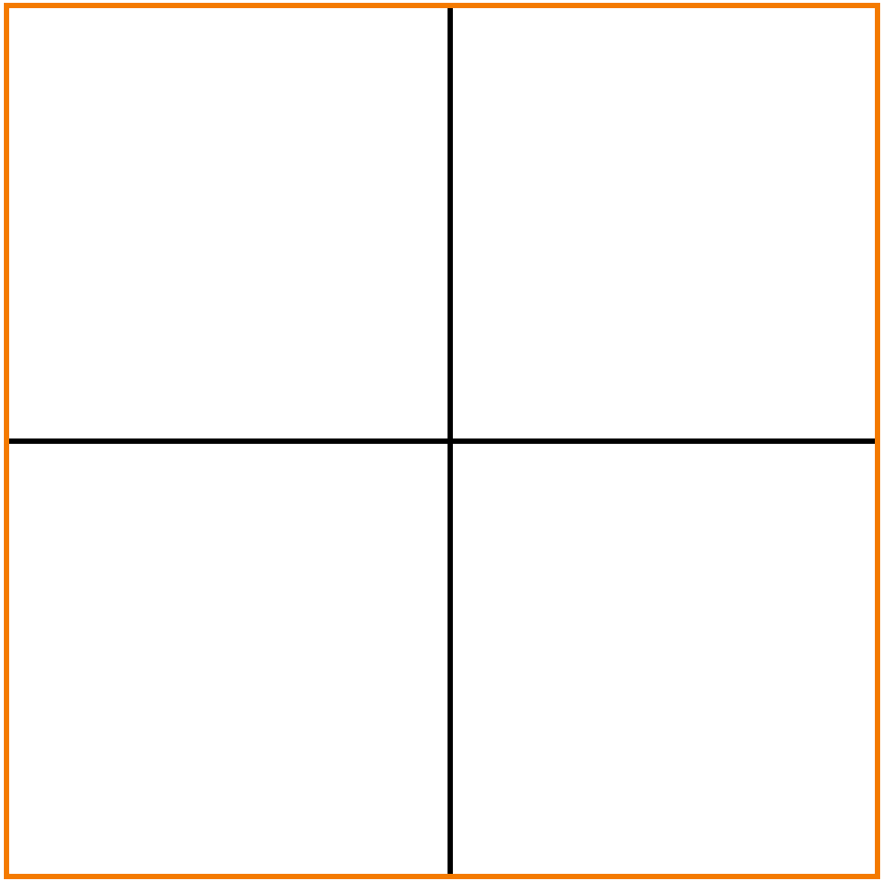
2. No basis vector v_i can be expressed as the linear sum of the other basis vectors.

Vector Space Basis

Many different bases can be used to represent the same vector space.

Example:

Consider the set of points in 2D Euclidean space.



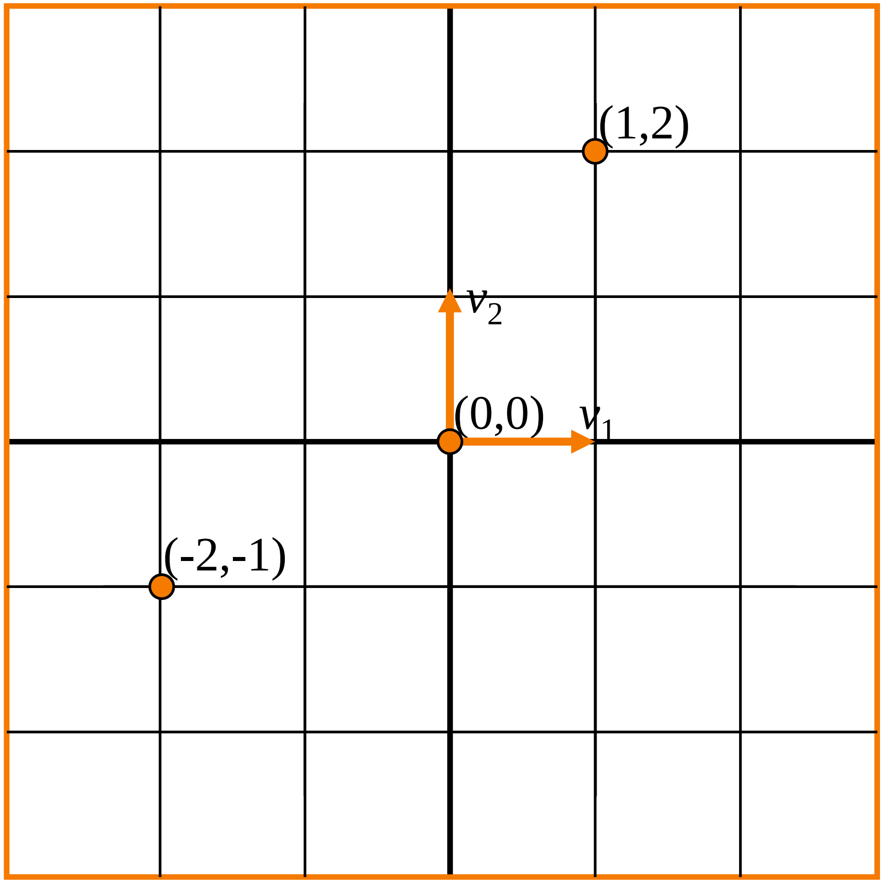
Vector Space Basis

Many different bases can be used to represent the same vector space.

<u>Example</u>:

Consider the set of points in 2D Euclidean space.

We can represent each vector in terms of its (x,y)-coordinates.



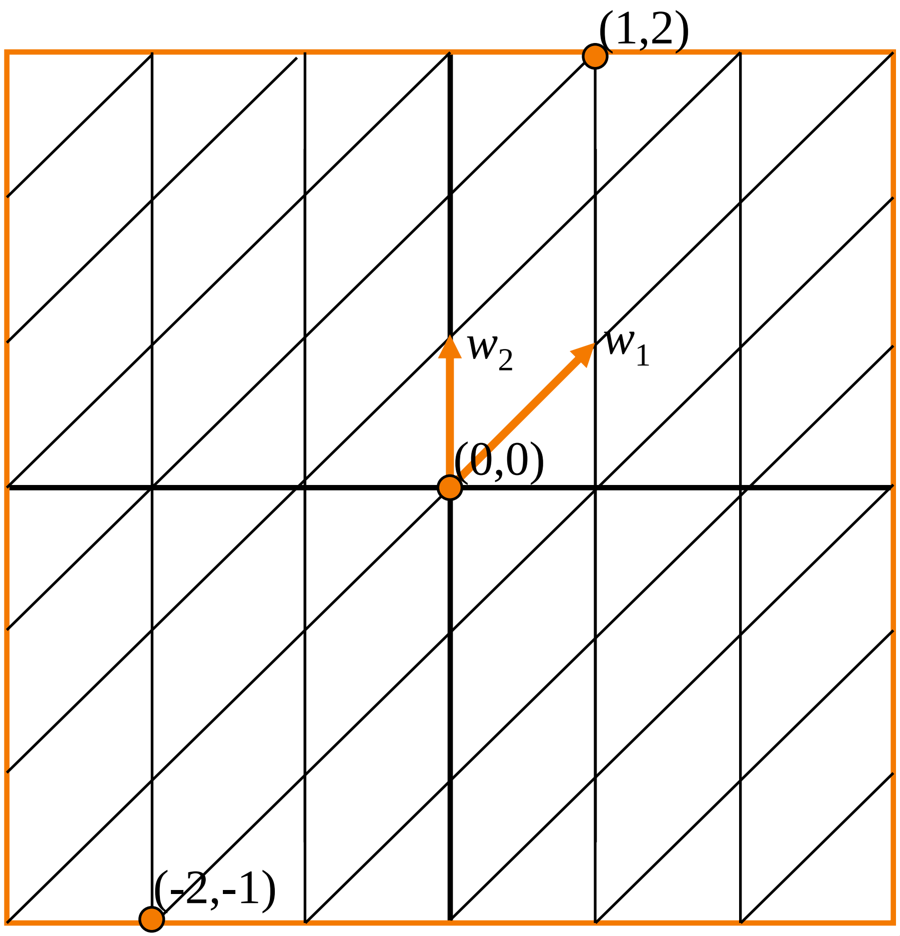
Vector Space Basis

Many different bases can be used to represent the same vector space.

Example:

Consider the set of points in 2D Euclidean space.

Or we could use a different basis…



Linear Maps

A function $L:V\rightarrow W$, is a <u>linear map</u> if for all v_1 and v_2 in V and all (real / complex) scalars a and b:

$$L(av_1+bv_2)=a\cdot L(v_1)+b\cdot L(v_2)$$

Linear Maps

A function $L:V\rightarrow W$, is a <u>linear map</u> if for all v_1 and v_2 in V and all (real / complex) scalars a and b:

$$L(av_1+bv_2)=a\cdot L(v_1)+b\cdot L(v_2)$$

If it exists, the inverse of a linear map L is the map L^{-1} with the property that:

$$L^{-1}\left(L(v)\right)=v$$

Linear Maps

If $L: V \rightarrow W$, is a linear map:

The set of vectors:

$$K = \{v \in V \mid L(v) = 0\}$$

is a vector subspace called the kernel.

The set of vectors:

$$I = \{w \in W \mid \exists v \in V \text{ s.t. } L(v) = w\}$$

is a vector subspace called the image.

Matrices

Given a vector space V, with basis $\{v_1,\ldots,v_n\}$, a linear map can be expressed as an nxn matrix M such that for any vector $v=a_1v_1+\ldots+a_nv_n$ in V:

$$L(v) = b_1v_1 + \cdots + b_nv_n$$

with:

$$\begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix} = \begin{pmatrix} M_{11} \cdots M_{1n} \\ \vdots \quad \ddots \quad \vdots \\ M_{n1} \cdots M_{nn} \end{pmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix}$$

Change of Basis

Given a vector space V, and given two bases $\{v_1,\ldots,v_n\}$ and $\{w_1,\ldots,w_n\}$, then since $\{v_1,\ldots,v_n\}$ is a basis, there exist values B_{ij} such that:

$$\begin{gathered} w_1 = B_{11}v_1 + \cdots + B_{1n}v_n \\ \vdots \\ w_n = B_{n1}v_1 + \cdots + B_{nn}v_n \end{gathered}$$

Change of Basis

Given a vector space V, and given two bases $\{v_1,\ldots,v_n\}$ and $\{w_1,\ldots,w_n\}$, the matrix B is the change of basis matrix.

If v is any vector in V, we can write out v in terms of the basis $\{v_1,\ldots,v_n\}$ as $v=a_1v_1+\ldots+a_nv_n$.

We can also write out v in terms of the basis $\{w_1,\ldots,w_n\}$ as $v=b_1w_1+\ldots+b_nw_n$.

The coefficients are related by:

$$\begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix} = \begin{pmatrix} B_{11} \cdots B_{1n} \\ \vdots \quad \ddots \quad \vdots \\ B_{n1} \cdots B_{nn} \end{pmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix}$$

Change of Basis

Given:

- A vector space V,
- Two bases $\{v_1,\ldots,v_n\}$ and $\{w_1,\ldots,w_n\}$,
- A linear operator L represented by the matrix M in terms of the basis $\{v_1,\ldots,v_n\}$.

The matrix representation for L in terms of the basis $\{w_1,\ldots,w_n\}$ is given by:

$$BMB^{-1}$$

Change of Basis

Why do we care?

Change of Basis

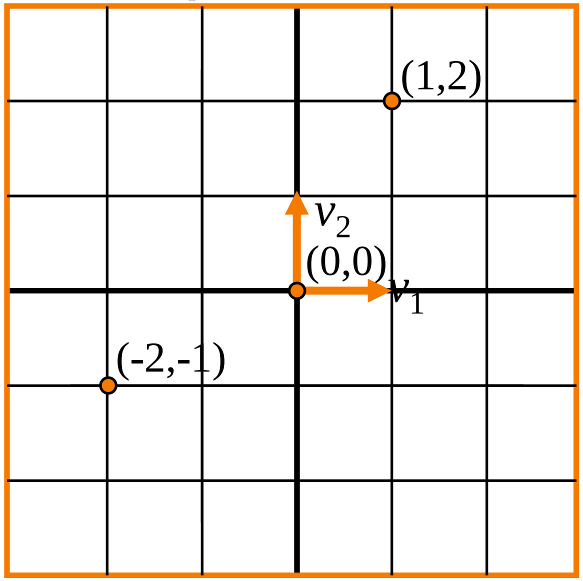
Why do we care?

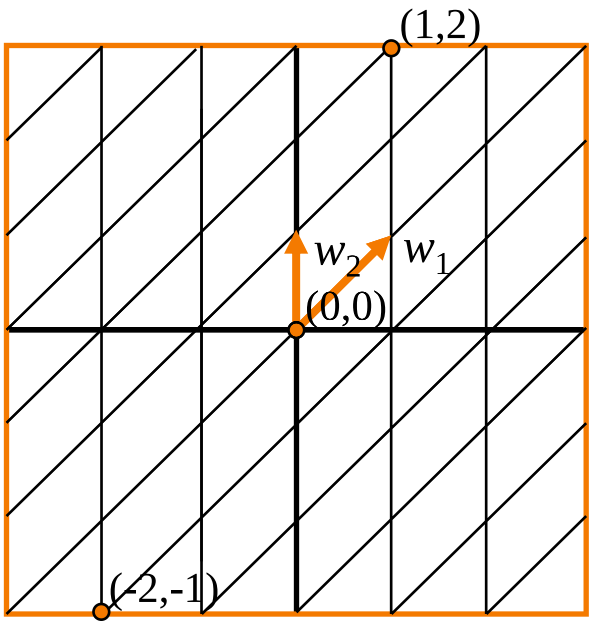
Choosing the appropriate basis can make it much easier to understand a linear operator.

Change of Basis

Why do we care?

Example:





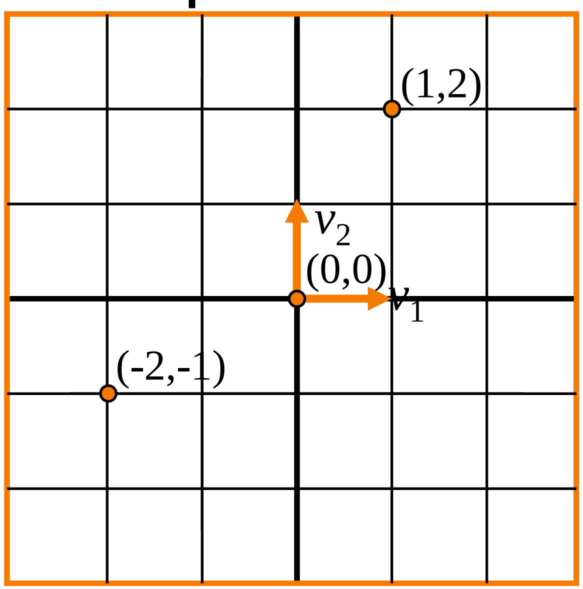
$$B = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$$

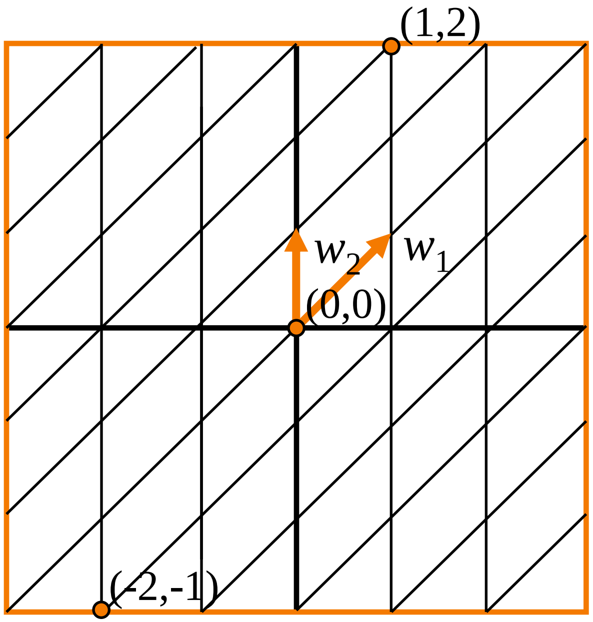
$$B^{-1} = \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix}$$

Change of Basis

Why do we care?

Example:





$$B = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$$

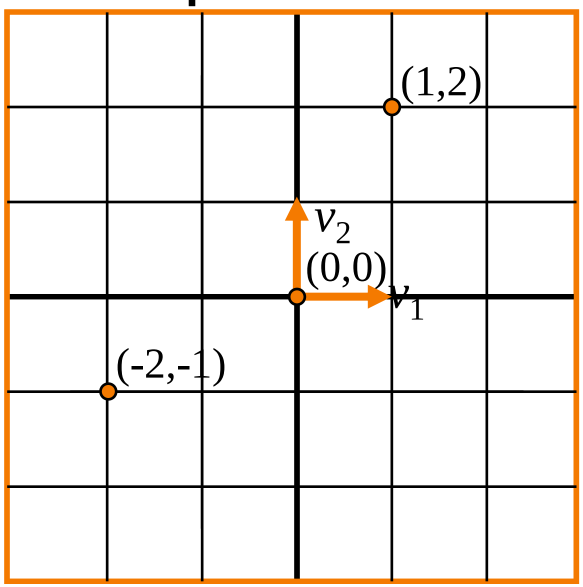
$$B^{-1} = \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix}$$

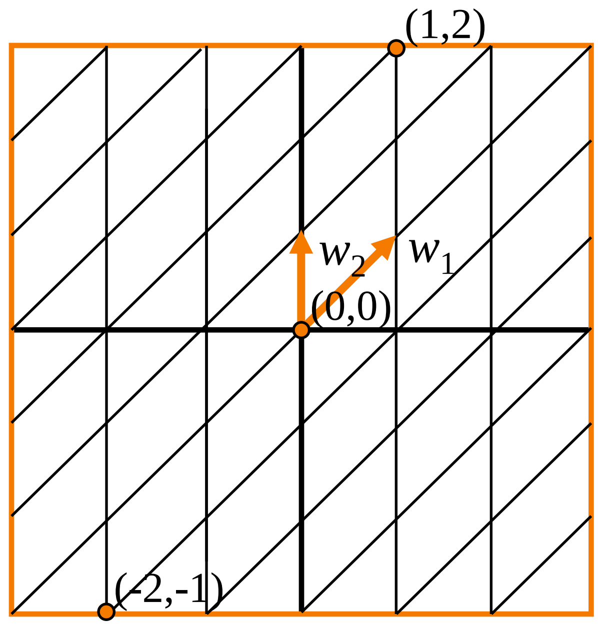
$$M = \begin{pmatrix} 2 & 0 \\ -1 & 1 \end{pmatrix} \Rightarrow BMB^{-1}$$

Change of Basis

Why do we care?

Example:





$$B = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$$

$$B^{-1} = \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix}$$

$$M = \begin{pmatrix} 2 & 0 \\ -1 & 1 \end{pmatrix} \Rightarrow BMB^{-1} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}\begin{pmatrix} 2 & 0 \\ -1 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}$$

Determinants

The determinant is a function that associates a scalar value to every square (*n*x*n*) matrix.

One way to think about this is to write out the matrix as a set of column vectors:

$$M = \left(w_1 | \cdots | w_n \right)$$

Then the determinant of *M* is the (signed) volume of the parallelepiped with sides $\{w_1,\ldots,w_n\}$

Determinants

The determinant of a matrix *M* is equal to zero if and only if there exists a vector *v* in *V*, with $v \neq 0$, such that $M(v)=0$.

Eigenvalues and Eigenvectors

The scalar λ is an eigenvalue of a matrix *M* if there exists a vector *v* in *V* such that:

$$\lambda v = M(v)$$

In this case, *v* is an eigenvector of *M*.

Eigenvalues and Eigenvectors

If *M* has an eigenvector *v* with eigenvalue λ, this must mean that:

$$0 = (M - \lambda)(v)$$

Thus, the matrix:

$$M - \lambda \cdot \mathrm{Id} = \begin{pmatrix} (M_{11} - \lambda) & \cdots & M_{1n} \\ \vdots & \ddots & \vdots \\ M_{n1} & \cdots & (M_{nn} - \lambda) \end{pmatrix}$$

must have zero determinant.

Characteristic Polynomials

If we treat λ as a variable, then the determinant:

$$\det(M - \lambda \cdot \mathrm{Id})$$

is a polynomial of degree *n* in λ. This polynomial is the characteristic polynomial of *M*.

Characteristic Polynomials

The roots of the characteristic polynomial of *M*:

$$\det(M - \lambda \cdot \mathrm{Id})$$

are precisely the eigenvalues of the matrix *M*.

Thus, if we are considering *M* as a matrix acting on a complex vector space, *M* must always have at least one eigenvalue.