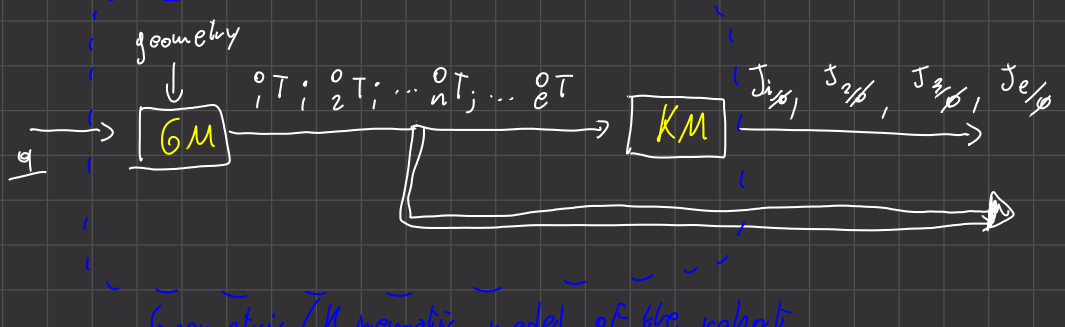
25/09/23

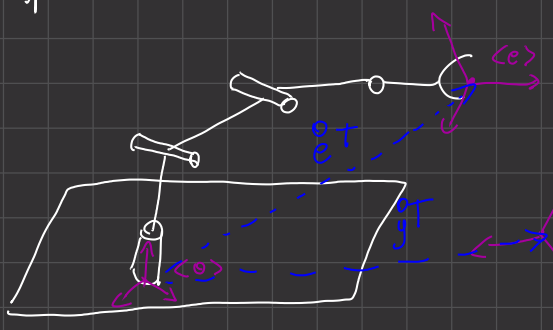
$$J_{i/0}(q) = J_{i/0}\left({}^0_1T(q);\ {}^0_2T(q);\ {}^0_3T(q_1) \dots {}^0_mTT(q)\right)$$

The info that we need to describe a Jacobian is not only the angle of every joint but also the distances and all geometrical distances and all the Transf. matrices contain this information. Inside J there are all these geometric distances. J contain all the parameters of the geometry. This connects to the geometric model.

We can now show a new block that can take the info from the GM.



We have the means for building two blocks. One to know where the frames start and on the basis of that we can work on the kinematic model. The Jacobians provide the link from the cartesian to joint space?



<g> goal frame

${}^g_yT$ desired transformation matrix

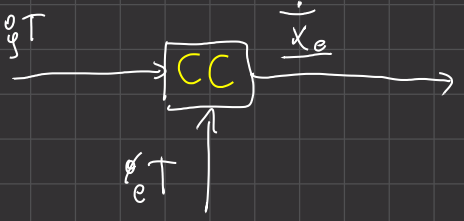
We want to make <e> approach <g>

<e> depends on how the robot moves and is constrained by that. Let's think for a moment that <e> was not constrained. First we want to understand how a frame not constrained should move to reach <g>

If I want to go to position of <g> with <e> how should I move? For this we should have a vector in the direction from <e> to <g> But keep in mind that this is not the only solution because any vector ortogonal to the distance of <e> would turn but get closer if there is a component towards <g>. In addition to this examples there are infinite vectors that work. We will also see solutions for rotation.

The block outputs the desired velocity for the frame

We now introduce the Cartesian Controller



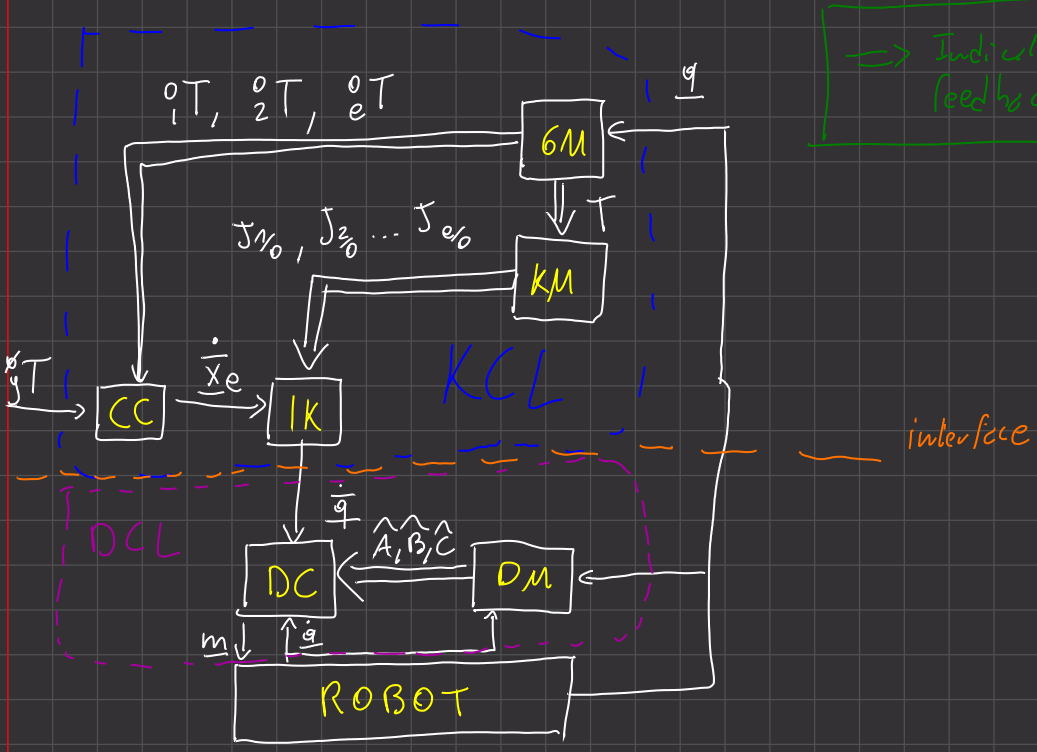
$\dot{\underline{x}}_e \in \mathbb{R}^6$

This outputs the desired cartesian velocity ($\dot{x}$), but we need the desired joint velocity vector ($\dot{q}$).
We have seen that Jacobian maps from joint to cartesian space.
We will see that the inverse of Jacobian will offer in some way the inverse, even if the Jacobian won't be square. This block will output the velocity that best approximate the desired velocity.

If we have a 4-degree of freedom robot we can go in a subset of the 6-dof space.
In the other hand if we have a more than 6-dof robot sometimes we cannot garantee the velocity required. For example think when your arm is stretched and you want to extend even more. You can't.

We are still missing the inverse kinematic block. That provides the best approximation to this problem. Remember the workspace of the robot.

The overall control scheme is the following:



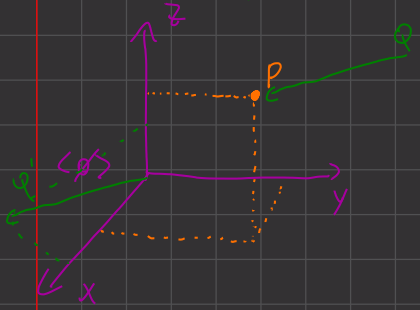
The robot wants torque (m). From the robot we measure $\underline{q}$ and $\dot{\underline{q}}$. The first thing we do in the DCL is building a controller that builds the desired velocity. The objective of the DCL is to follow as close as possible the reference velocity. Once you have done this you need to drive the ee to the goal. Now I need to generate $\dot{\underline{q}}$. From geometry we have

position of the frames and with RM their movements with teachpen. Feeding this to the teachpen we see how the system should move in space. We get the desired velocity that gets mapped to the desired velocity with the inverse Kinematic.

TODO

Ch. 1 Geometric Foundamentals

Coordinate Systems (Frame)



Oriented triad of orthogonal vectors.

X, Y, Z are placed in the right hand rule.
- thumb = x
- Index = y
- Middle = z

This is the right handed frame

With the concept of frame we can describe vectors and points. For example

$$ {}^{\phi}p = \begin{bmatrix} x_p \\ y_p \\ z_p \end{bmatrix} $$

The components are the lenght of the vectors that allow me to reach P in the reference of this particular frame.
We also have vectors.
If we imagine Q in ϕ of the frame it's easy to identify the components this is called a projected vector.

$$ {}^{\phi}(P-Q) = {}^{\phi}P - {}^{\phi}Q $$

TODO ?

The point on with I apply the force is very important, but when I want to know the distance the point on with is applied is not very important.

Often we can also say:

Projected vector ???

$$ {}^{\phi}(P-Q) = {}^{\phi}P - {}^{\phi}Q \triangleq {}^{\phi}\underline{v}_{P,Q} $$

↑ the vector that joins Q to P

We can now recall the grassman rule to join point and vectors

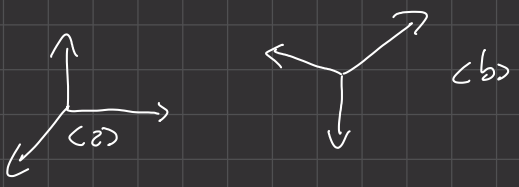
Grassmen Rule

$$ {}^{\phi}P = {}^{\phi}Q + {}^{\phi}\underline{v}_{P,Q} $$

In the frame unit vectors are immediately representable:

x	$\hat{\imath}_\emptyset$	$[1\ 0\ 0]^T$
y	$\hat{\jmath}_\emptyset$	$[0\ 1\ 0]^T$
z	$\hat{k}_\emptyset$	$[0\ 0\ 1]^T$

Let's now consider two frames



Set of information needed to describe how ⟨b⟩ is with respect to ⟨a⟩ is

$$\{ {}^a\underline{O}_b, {}^a\hat{\imath}_b, {}^a\hat{\jmath}_b, {}^a\hat{k}_b \}$$

unit vectors of b projected on ⟨a⟩ to code the rotation of frame ⟨b⟩ in respect to ⟨a⟩.

This information is encoded into the rotation or orientation matrix.

Orientation matrices (Rotation Matrices)

Let's consider a vector v and represent it as a function of unit vectors of ⟨b⟩.

$$\underline{v} = \underline{\hat{\imath}_b}\, x_b + \underline{\hat{\jmath}_b}\, y_b + \underline{\hat{k}_b}\, z_b \qquad \hat{\imath}, \hat{\jmath}, \hat{k} \text{ form a base.}$$

If now I perform projection of $\underline{v}$ on ⟨a⟩

$${}^a\underline{v} = {}^a\underline{\hat{\imath}_b}\, x_b + {}^a\underline{\hat{\jmath}_b}\, y_b + {}^a\underline{\hat{k}_b}\, z_b$$

Now representing as a vector-matrix multiplication we get:

$${}^a\underline{v} = \begin{bmatrix} {}^a\underline{\hat{\imath}_b} & {}^a\underline{\hat{\jmath}_b} & {}^a\underline{\hat{k}_b} \end{bmatrix} \begin{bmatrix} x_b \\ y_b \\ z_b \end{bmatrix}$$

referred frame → a, referring frame → b

$${}^a_b R = \text{ROTATION MATRIX}$$

$${}^a\underline{v} = {}^a_b R\, {}^b\underline{v}$$ ← This is another way of seeing the multiplication.

In Matlab this will be represented like a_R_b.

It's important that in the vector we include the name of the frame on which the vector should be projected on. For example b_distance.

So a multiplication could be:

$$a\_distance = a\_R\_b \quad b\_distance.$$

R is a ortho-normal matrix
- columns are orthogonal to each other
- unitary lenght

$${}^a_b R \, {}^a_b R^T = I_{nxn}$$

Given this the matrix has some properties:
Recalling that:

$$det(AB) = det(A)\, det(B)$$
$$det(A) = det(A^T)$$

Being ortho-normal means:

$$\begin{cases} {}^a_b R \, {}^a_b R^T = I_{3x3} \\ {}^a_b R^T \, {}^a_b R = I_{3x3} \end{cases} \Rightarrow {}^a_b R^T = {}^a_b R^{-1} = {}^b_a R$$

↑ this because of $det(R) = 1$

this is important because:

$$\forall M: \; M M^T = I \Rightarrow det(M) = \pm 1$$

For Rot. Mat the determinant equals 1 $\rightarrow$ $det({}^a_b R) = 1$

this to preserve the direction meaning the right handed frame.

For example

$$\begin{matrix} & x & y & z & \text{on } b \\ \end{matrix}$$
$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \rightarrow \; det = -1$$

We can see that y and z are the same while x is the opposite, but this is impossible if we consider 2 right handed frames. So this cannot happen. $det({}^b_a R) = -1$ does not respect right-hand rule.

From these properties we can also conclude that the transpose is the inverse.
It also makes sense if you think about it geometrically.

$$SO(n) = \{ R \in \mathbb{R}^{n x n} : R^T R = I_{nxn}, \; det(R) = +1 \}$$

↑ Special orthogonal group

Special for this limitation

This group has some properties: N.B

TODO on what? MULT DIV

- CLOSURE: given any two elements R_1, R_2 of $SO(n)$ then also $R_1 \cdot R_2 \in SO(n)$
- IDENTITY: there exists an element I of $SO(n)$ such that $IR \in SO(n)$ for any R
- INVERSE: for any $R \in SO(n)$ there exists R^{-1} such that $RR^{-1} = I$
- ASSOCIATIVITY: given three elements R_1, R_2, R_3 in $SO(n)$ it follows that $(R_1 R_2) R_3 = R_1 (R_2 R_3)$

A member of SO(3) is a Rotation Matrix

A consideration about ${}^2_b R^T = {}^b_2 R$ is that this is also equivalent to:

$$ {}^2_b R = \begin{bmatrix} {}^2\hat{\imath}_b & {}^2\hat{\jmath}_b & {}^2\hat{k}_b \end{bmatrix} = \begin{bmatrix} {}^b\hat{\imath}_2^T \\ {}^b\hat{\jmath}_2^T \\ {}^b\hat{k}_2^T \end{bmatrix} $$

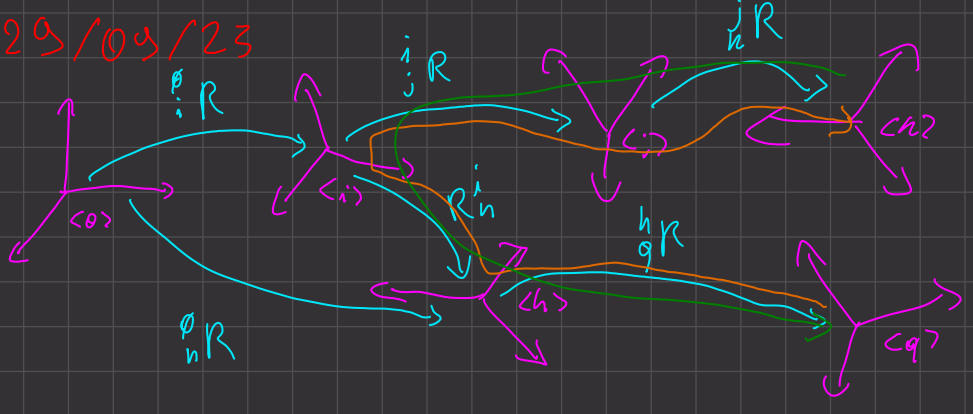
On the transpose you will se what you see on columns now on the rows.

Our manipulator is composed by a tree of frames:

The notation helps to check if everything is done correctly

N.B. This holds for every v so for definition $^0_1R\ ^1_2R\ ^2_3R = {}^0_3R$

29/09/23

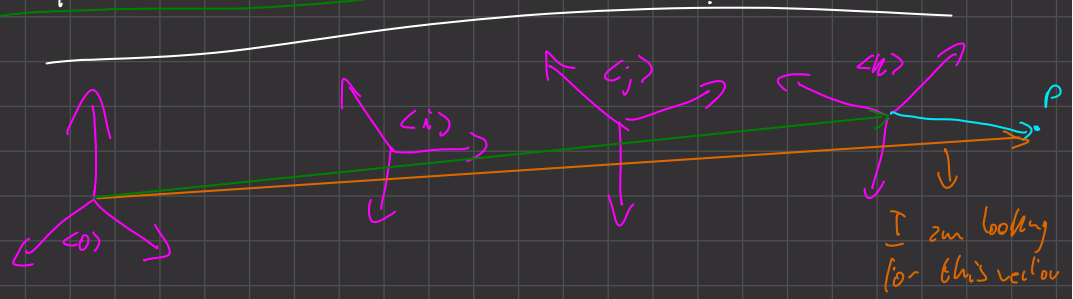


Here we have some reference frames and suppose that we know some rotation matrices.

How do you compute: ${}^q_n R = {}^h_q R^T \, {}^i_h R^T \, {}^j_i R \, {}^j_h R = {}^q_h R \, {}^h_i R \, {}^j_i R \, {}^j_n R$ (POSTMULTIPLICATION)

We can also do the same thing following this path: (PREMULTIPLICATION)

$${}^q_n R = {}^h_q R^T \, {}^i_h R^T \, {}^j_i R \, {}^j_n R$$



To get from P to P the position of all the frames is an essential parameter. I could for example use another vector P and now the description of P becomes P + P. This can be done iteratively going back towards <0>.

$${}^k p = {}^k(P - O_k)$$

$${}^0 P = {}^0 O_k + {}^0(P - O_k)$$

$${}^0 P = {}^0 O_k + {}^0_k R \, {}^k P$$

Omogeneous Coordinates

$${}^k \bar{P} = \begin{bmatrix} {}^k P \\ 1 \end{bmatrix} \in \mathbb{R}^4$$

$${}^0 \bar{P} = {}^0_k T \, {}^k \bar{P} \qquad {}^0_k T \triangleq \begin{bmatrix} {}^0_k R & {}^0 O_k \\ 0_{3\times 1} & 1 \end{bmatrix} \in \mathbb{R}^{4\times 4}$$

So if I do $\begin{bmatrix} {}^0_k R & {}^0 O_k \\ 0_{3\times 1} & 1 \end{bmatrix} \begin{bmatrix} {}^k P \\ 1 \end{bmatrix} = \begin{bmatrix} {}^0_k R \, {}^k P + {}^0 O_k \\ 1 \end{bmatrix}$

$${}^0_k T \quad {}^k \bar{P} = {}^0 \bar{P}$$

Inverse trasformation matrix

The increase in dimension is a gimmick to obtain the previous formula with a matrix by vector multiplication.

We Know that:

$$ {}^a_bT^{-1} = {}^b_aT $$

N.B.

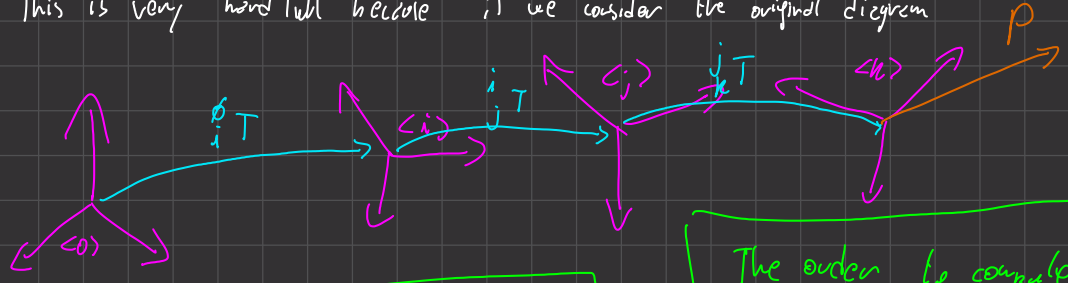
BUT

$$ {}^a_bT^{-1} \neq {}^a_bT^T $$

$$ {}^a_bT^{-1} = \begin{bmatrix} {}^a_bR^T & -{}^a_bR^T\,{}^aO_b \\ 0_{3x1} & 1 \end{bmatrix} $$

The origin of a on b would be the opposite vector and to put it in the right direction I need a minus in front.

This is very handfull because if we consider the original diagram



$$ {}^0\bar{p} = {}^0_1T\,{}^1_2T\,{}^2_3T\,{}^{n-1}_nT\,{}^n\bar{p} $$

$$ {}^0_nT = {}^0_1T\,{}^1_2T\,{}^2_3T\,{}^{n-1}_nT $$

The order to compute the trasformation matrix is from <0> to <e>

It's also possible to describe a vector using homogeneous coordinates. It's done by just apposing $\emptyset$ to the vector.

$$ \bar{p} = \begin{bmatrix} {}^n\bar{p} \\ \emptyset \end{bmatrix}_{4x1} $$

This because the origin in this case does not come into play.

The inertial frame it's not different from any other frame.

three parameter representation of rotation matrices

${}^{a}_{b}R$ - Rotation matrices are orthonormal. (vectors are orthogonal to each other; modulus = 1 for every vector.)

So I have:

$$\begin{cases} {}^{a}i_b^T \, {}^{a}i_b = 1 \\ {}^{a}j_b^T \, {}^{a}j_b = 1 \\ {}^{a}k_b^T \, {}^{a}k_b = 1 \end{cases} \qquad \begin{cases} {}^{a}i_b \cdot {}^{a}j_b^T = 0 \\ {}^{a}j_b^T \, {}^{a}k_b = 0 \\ {}^{a}k_b^T \, {}^{a}i_b^T = 0 \end{cases}$$

(dot product)

${}^{a}_{b}R$ satisfy all these equations

We have 6 constraints and 9 numbers so we only have to pick 3 that are unknown.

For this reason we can talk about 3 number representation of rotation matrices.

The easiest of these representation is called
→ Pronounced Oilar

Euler angles

they express that any rotation can be represented by three rotations between 3 different axes.

We can choose between:
- extrinsic rotation
- intrinsic rotation

- I describe the final orientation with rotation around a fixed coordinate system. This unfortunately is not so easy to implement because when we multiply R matrices we are not doing intrinsic rotation. In this case we first rotate around the original x, then around the original y and finally around the original z.

- this is easier to implement. We first rotate around x, then the rotated y and then the rotated z. When we multiply with Rot. Matrices we are doing intrinsic rotations.

Intrinsic angles can be divided in:

- proper Euler angles
  z-x-z; y-x-y; x-z-x; ... → these represent the sequence of axes around which you do the rotation
- Tait-Bryan angles
  x-y-z; z-y-x; ...

$z-y'-x''$ — this is the most used one and we will use this one

$${}^b_a R = R_z(\psi)\, R_y(\theta)\, R_x(\phi)$$

YAW PITCH ROLL

the apex stresses that we are talking about intrinsic rotation

It's possible to demonstrate that an extrinsic rotation x-y-z it's the same as an intrinsic rotation z-y-x.

02/10/23

$${}^b_a R = R_z(\psi)\, R_y(\theta)\, R_x(\phi) =$$

$$= \begin{bmatrix} c\psi c\theta & -s\psi c\phi + c\psi s\theta s\psi & s\psi s\phi + c\psi c\phi s\theta \\ s\psi c\theta & c\psi c\phi + s\phi s\theta s\psi & -c\psi s\phi + s\theta s\psi c\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix}$$

Not needed to know by heart

It's useful to know the structure to know the inverse.

$$\theta = \underbrace{atan2}_{\text{software function to compute atan on 4 quadrants instead of two.}}\left(-r_{3,1}, \sqrt{r_{11}^2 + r_{21}^2}\right) \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$$

If $\cos\theta \neq 0$:

$$\psi = atan2(r_{21}, r_{11})$$
$$\phi = atan2(r_{32}, r_{33})$$

If $\cos\theta = 0$ it's an interesting configuration because leads to a singularity.

In the sequence of rotation if we rotate in pitch of $\pm\frac{\pi}{2}$ the x axis becomes the same as the z axis of the original frame. We have infinite combination of first and third axis that end up representing the same output. We get a singularity when the pitch is $\pm\frac{\pi}{2}$.

Suppose now that I want to control roll to a certain angle, when i get $\frac{\pi}{2}$ in pitch yaw and roll become the same. It's an ill defined situation.

The use of Euler angles is fine but we must have regard for pitch being close to $\frac{\pi}{2}$

Vector operations

- Linear combination

$$\underline{v} = c_1 \underline{v}_1 + c_2 \underline{v}_2 + \dots c_n \underline{v}_n$$

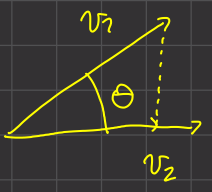
If they are linear indipendent they can form the base of your n-based dimensional frame.

If I take into consideration the projection of a frame I also need to projection the same frame all the components.

$${}^2\underline{v} = c_1 {}^2\underline{v}_1 + c_2 {}^2\underline{v}_2 + \dots c_n {}^2\underline{v}_n$$

- Scalar product

$$(\underline{v}_1 \cdot \underline{v}_2) = (\underline{v}_2 \cdot \underline{v}_1) = |v_2||v_1| \cos\theta$$



$$(\underline{v}_1 \cdot \underline{v}_2) = {}^2\underline{v}_1^T \, {}^2\underline{v}_2$$

projected on $\langle 2 \rangle$

I can consider $(\underline{v}_1 \cdot) = (\underline{v}_1^T)_{1\times 3}$ linear operation

- Vector product

$$(\underline{v}_1 \wedge \underline{v}_2) = (\underline{v}_1 \times \underline{v}_2) = \begin{cases} \text{modulus} \\ |\underline{v}_1 \times \underline{v}_2| = |v_1||v_2| \sin(\theta) \quad \hookrightarrow \text{minimum angle} \\ \underline{v}_1 \times \underline{v}_2 \perp (\underline{v}_1, \underline{v}_2) \end{cases}$$

the direction is ortogonal to both v_1 and v_2 according to the right hand rule

- Properties:

- Anti-commutative: $(\underline{v}_1 \times \underline{v}_2) = -(\underline{v}_2 \times \underline{v}_1)$

- $${}^2(\underline{v}_1 \times \underline{v}_2) = {}^2[\underline{v}_1 \times] \, {}^2\underline{v}_2 = \begin{bmatrix} 0 & -z & y \\ z & 0 & x \\ -y & -x & 0 \end{bmatrix} {}^2\underline{v}_2$$

  Operator 3x3 — Components of ${}^2\underline{v}_1$

  For this reason ↳ This matrix is skew-simmetric

- $${}^2[\underline{v}_1 \times]^T = -{}^2[\underline{v}_1 \times]$$

- Inverse mapping: vex operator → $\text{vex}(S(\underline{v}_1)) = \underline{v}_1$

axial vector

sometimes this can also be represented like: $[\underline{v}_1 \times] = S(v_1)$

$$\forall M: M \in \mathbb{R}^{3\times 3}, \; M = -M^T$$
$$M\underline{y} = \text{vex}(M) \times \underline{y}$$

Any skew-matrix has an axial vector. Vex is needed to get x, y and z

Power of skew-symmetric 3x3 real matrices associated with a unit vector

$h \in \mathbb{R}^{3\times 1}$

$[\underline{h}\,x] = \underline{h} \times$

$[\underline{h}\,x]^2 = (\underline{h}\,\underline{h}^T - I)$

$[\underline{h}\,x]^3 = -[\underline{h}\,x]$

$[\underline{h}\,x]^4 = -[\underline{h}\,x]^2$

$[\underline{h}\,x]^5 = [\underline{h}\,x]$

$[\underline{h}\,x]^6 = [\underline{h}\,x]^2$

$\vdots$

After the 4th power they repeat from the first one.

- In general we state that:

$$[\underline{h}\,x]^{2i+1} = (-1)^i [\underline{h}\,x]$$

Rule for odd numbers (skew symmetric)

$$[h x]^{2(i+1)} = (-1)^i (h h^T - I)$$

Rule for even numbers (symmetric)

$(I - hh^T)$ projects a vector on the plane orthogonal to h (TONO CAMIZLIK)

- $(I - \underline{h}\underline{h}^T)$ is the projection operator defined by h

$(I - \underline{h}\underline{h}^T)\underline{h} = \emptyset$ because $(h - h\underline{h}^T\underline{h}) = \emptyset$

- $\text{trace}([h x]^{2i+1}) = 0$

tr → sum of all the elements on the main diagonal in a matrix

$tr([\underline{h}\,x]^{2(i+1)}) = (-1)^{(i+1)} \cdot 2$

- $A \in \mathbb{R}^{n\times n}$

$A = A^T$ symmetrical

$A = -A^T$ skew symmetrical

$$\forall A \in \mathbb{R}^{n\times n} \quad A = \frac{A + A^T}{2} + \frac{A - A^T}{2}$$

no symm or skew-symm

$\frac{A+A^T}{2}$ symmetric

$\frac{A-A^T}{2}$ skew-symmetric

Linear Operators between Vectors

Consider: • frame $\langle b \rangle$
• any projected vector ${}^b\underline{v}$
• ${}^b\underline{\omega}$ subject to ${}^b\underline{\omega} = {}^b\underline{L}\,{}^b\underline{v}$ linear operation $\underline{L} \in \mathbb{R}^{3\times 3}$

$${}^2\underline{\omega} = {}^2_b\underline{R}\,{}^b\underline{\omega} = {}^2_b\underline{R}\,{}^b\underline{L}\,{}^b\underline{v} = {}^2_b\underline{R}\,{}^b\underline{L}\,{}^2_b\underline{R}^T\,{}^2\underline{v} = {}^2\underline{L}\,{}^2\underline{v}$$

$$\boxed{{}^2\underline{L} = {}^2_b\underline{R}\,{}^b\underline{L}\,{}^2_b\underline{R}^T}$$

This is the operation to change the operator from one frame to another.

For example:

Vector product operator

$${}^b(\underline{\omega} \times \underline{v}) = {}^b[\underline{\omega}\times]\,\underline{v}$$

$${}^2(\underline{\omega} \times \underline{v}) = {}^2_b\underline{R}\,{}^b[\underline{\omega}\times]\,{}^2_b\underline{R}^T\,{}^2\underline{v} = {}^2[\underline{\omega}\times]\,{}^2\underline{v}$$