**Funciones para modelar la cinemática 3D**

**V1.1**

Una ecuación de restricción cinemática o conjunto de ellas se describirá mediante una estructura de MATLAB con la siguiente configuración:

struct1={constraint\_class p}

struct2={constraint\_class p fun}

siendo

constraint\_class el nombre de la clase (coincide con el nombre de la función, ver más adelante) entre ‘ ‘

p el vector puntero que indica la posición de las variables en el vector q

c el vector puntero que indica la posición de las variables en el vector c

fun el nombre de la función que permite calcular el valor de una variable conocido el instante temporal (para ecuaciones reónomas)

La cinemática de un mecanismo estará definida por un conjunto de ecuaciones de restricción cinemática agrupadas en estructuras de MATLAB como las indicadas anteriormente, que, a su vez, se almacenan en una cell mediante concatenación:

mech={}

mech=[mech;struct1;struct2 …]

save('kinematic\_def.mat','mech')

Esta variable se guardará en memoria como kinematic\_def.mat y será accesible cuando haya que evaluar el vector phi, su jacobiano o la derivada temporal del mismo así como las derivadas temporales del vector phi.

Se puede definir y guardar, vía menús, la cinemática de un mecanismo con la función def\_mech\_3D.

La librería incluye funciones que utilizan la información del archivo kinematic\_def.mat para calcular de manera automática los siguientes elementos:

 🡪 calc\_phi

 🡪 calc\_jac

 🡪 calc\_djac

 🡪 calc\_dphit

 🡪 calc\_d2phit

Adicionalmente, el método de Newton que calcula la posición está implementado en la función calc\_pos de esta librería.

1. **Sólido rígido definido por una base de 3 vectores unitarios ortogonales**

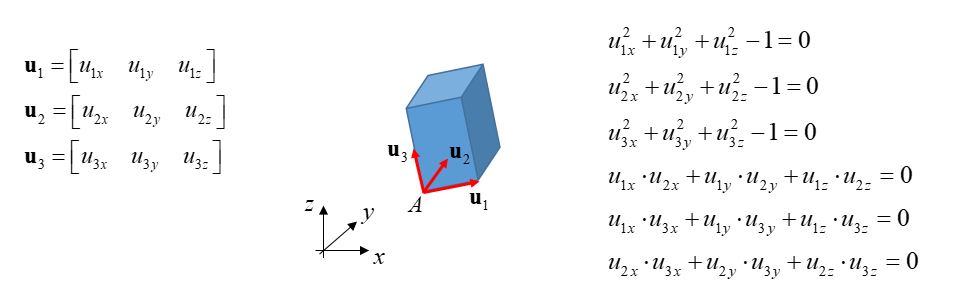
Función: phi\_3D\_rs\_3uv

Número de ecuaciones: 6

Entradas: p,q,c

p=[nu1x nu1y nu1z nu2x nu2y nu2z nu3x nu3y nu3z]

Ecuaciones:



Jacobiano: jacphi\_3D\_rs\_3uv(p,q,c)

Derivada del jacobiano respecto del tiempo: djacphi\_3D\_rs\_3uv(p,q,dq,c)

1. **Punto adicional en un sólido rígido definido por una base de 3 vectores unitarios ortogonales**

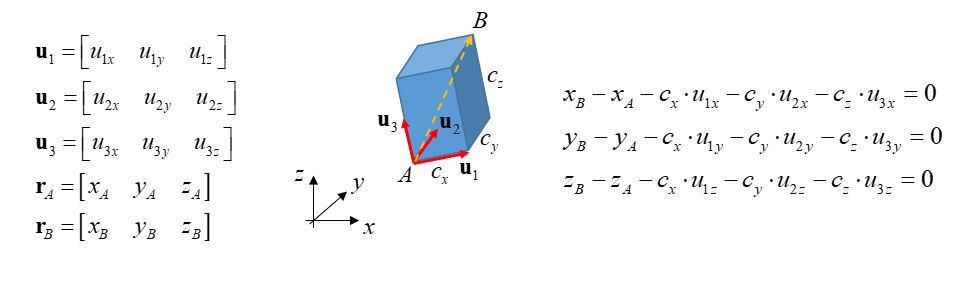
Función: phi\_3D\_rs\_1ap

Número de ecuaciones: 3

Entradas: p,q,c

p= [nu1x nu1y nu1z nu2x nu2y nu2z nu3x nu3y nu3z nxa nya nza nxb nyb nzb ncx ncy ncz]

Ecuaciones:



Jacobiano: jacphi\_3D\_rs\_1ap(p,q,c)

Derivada del jacobiano respecto del tiempo: djacphi\_3D\_rs\_1ap(p,q,dq,c)

1. **Ángulo que forman 2 vectores unitarios**

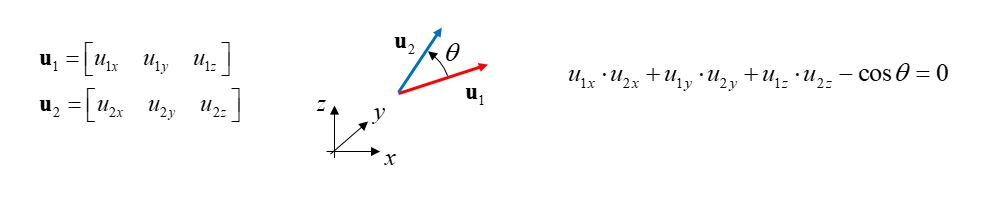
Función: phi\_3D\_2uv\_angle

Número de ecuaciones: 1

Entradas: p,q,c

p= [nu1x nu1y nu1z nu2x nu2y nu2z theta]

Ecuaciones:



Jacobiano: jacphi\_3D\_2uv\_angle(p,q,c)

Derivada del jacobiano respecto del tiempo: djacphi\_3D\_2uv\_angle(p,q,dq,c)

1. **Dos vectores unitarios paralelos**

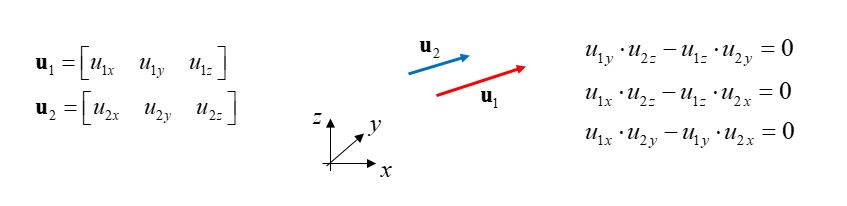
Función: phi\_3D\_2puv

Número de ecuaciones: 2 (+1 redundante)

Entradas: p,q,c

p= [nu1x nu1y nu1z nu2x nu2y nu2z]

Ecuaciones:



Jacobiano: jacphi\_3D\_2puv(p,q,c)

Derivada del jacobiano respecto del tiempo: djacphi\_3D\_2puv(p,q,dq,c)

1. **1 punto alineado con un vector unitario**

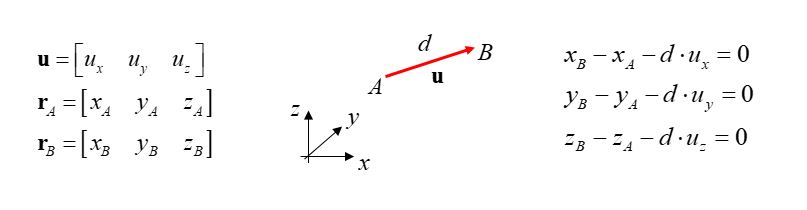
Función: phi\_3D\_1p1uv\_alin

Número de ecuaciones: 3

Entradas: p,q,c

p= [nux nuy nuz nxa nya nza nxb nyb nzb nd]

Ecuaciones:



Jacobiano: jacphi\_3D\_1p1uv\_alin(p,q,c)

Derivada del jacobiano respecto del tiempo: djacphi\_3D\_1p1uv\_alin(p,q,dq,c)

1. **Ángulo que forman un vector unitario respecto a una base plana ortogonal y unitaria**

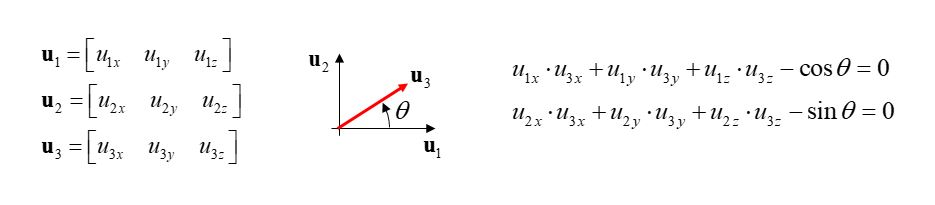
Función: phi\_3D\_1uv\_basis\_angle

Número de ecuaciones: 1 (+1 redundante)

Entradas: p,q,c

p= [nu1x nu1y nu1z nu2x nu2y nu2z nu3x nu3y nu3z theta]

Ecuaciones:



Jacobiano: jacphi\_3D\_1uv\_basis\_angle (p,q,c)

Derivada del jacobiano respecto del tiempo: djacphi\_3D\_1uv\_basis\_angle (p,q,dq,c)

1. **Valor de coordenada**

Función: phi\_coord\_t

Número de ecuaciones: 1

Entradas: p,q,fun,t

p= [ncoord]

Ecuaciones:



Jacobiano: jacphi\_coord\_t(p,q,fun,t)

Derivada del jacobiano respecto del tiempo: djacphi\_coord\_t(p,q,dq,fun,t)

1. **Distancia constante entre 2 puntos**

Función: phi\_3D\_rs\_2p

Número de ecuaciones: 1

Entradas: p,q,c

p= [nxa nya nza nxb nyb nzb nLab]

Ecuaciones:

Gráfico

Descripción generada automáticamente con confianza baja

Jacobiano: jacphi\_3D\_ rs\_2p (p,q,c)

Derivada del jacobiano respecto del tiempo: djacphi\_3D\_ rs\_2p (p,q,dq,c)