SYMBOLIC COMPUTATION APPLIED TO INDUSTRIAL ROBOTS THROUGH SAGE

Elvira Izurrategui, C.; Nájera Canal, S.; García Verde, L. F.; Rico Azagra, J.

Universidad de La Rioja

This paper presents the kinematics development of an industrial robot through SAGE (Software for Algebra and Geometry Experimentation) programming. SAGE is a free alternative to owner computation software such as Maple, Mathematica or MATLAB. SAGE can operate in server mode from any browser, allowing you to apply new information technologies to multiple engineering problems.

The kinematics of industrial robots is one of the applications of SAGE. The kinematics study is dealt considering a set of rigid bars linked together through articulations, making up all of them an open kinematic chain. From here, the robot motion analysis is solved using algorithms based on vector mechanics. The approach is performed following the Denavit-Hartenberg representation that standardizes the coordinate systems which describes spatial relations. After studying the kinematic problem, the symbolic computation programming in SAGE is performed and generic results of the kinematic variables of position, velocity and acceleration are obtained. The programming allows you to particularize the problem to any specific situation and the data can be presented graphically to give a simple computation environment for any particular structure of an industrial robot.

Keywords: *SAGE (Software for Algebra and Geometry Experimentation); Industrial robot; Kinematics; Mechanics; CAE*

CÁLCULO SIMBÓLICO APLICADO A ROBOTS INDUSTRIALES MEDIANTE SAGE

Este artículo presenta el desarrollo de la cinemática de un robot industrial mediante programación en el software SAGE (Software for Algebra and Geometry Experimentation). SAGE es una alternativa libre a programas de cálculo propietarios como Maple, Mathematica o MATLAB. SAGE puede funcionar en modo servidor desde cualquier navegador, permitiendo aplicar las nuevas tecnologías de la información a múltiples problemas en la ingeniería.

Entre las aplicaciones de SAGE se encuentra la cinemática de los robots industriales. Su estudio se aborda considerando una serie de barras rígidas unidas entre sí a través de articulaciones, constituyendo el conjunto una cinemática abierta. A partir de esta situación se plantea el análisis del movimiento del robot que es resuelto mediante algoritmos basados en la mecánica vectorial. El planteamiento se realiza siguiendo la representación de Denavit-Hartenberg que estandariza los sistemas de coordenadas que describen relaciones espaciales. Tras el estudio del problema cinemático, se realiza la programación con cálculo simbólico en SAGE obteniendo resultados genéricos de las variables cinemáticas de posición, velocidad y aceleración. La programación permite particularizar el problema a cualquier situación concreta, y los datos se pueden presentar de forma gráfica, obteniéndose un entorno sencillo de cálculo para una estructura concreta de un robot industrial.

Palabras clave: *SAGE (Software for Algebra and Geometry Experimentation); Robot industrial, Cinemática; Mecánica vectorial; CAE*

Correspondencia: silvano.najerac@unirioja.es

1. Introducción

Los avances tecnológicos en el hardware de los ordenadores y los nuevos desarrollos de software han permitido en las últimas décadas un gran desarrollo de los programas de cálculo simbólico. Sage (*Software for Arithmetic and Geometry Experimentation*) fue creado en 2004 por William Steim como un proyecto de desarrollo de un software libre CAS (*Computer Algebra System*) de código abierto distribuido bajo licencia GPL (*General Public License*). La misión del proyecto fue crear una alternativa viable de código abierto a Magma, Maple, Mathematica y Matlab. Desde sus inicios, Sage ha ido integrando varios paquetes de software libre (PARI, GAP, Máxima, Singular, Pynac, GNUplot, etc.) manteniendo actualmente alrededor y actualmente incorpora cerca de 100 paquetes ("Sage Main Page," 2012), y encontrándose en continuo desarrollo y evolución.

Mientras otros sistemas CAS utilizan su propio lenguaje de programación, Sage ha sido desarrollado en lenguaje Python. Dicho lenguaje está orientado a objetos y su aprendizaje y uso se ha extendido en los ambientes académicos. Entre las razones de su éxito cabe destacar que es un software libre multiplataforma, además de que utiliza un lenguaje de muy alto nivel compacto, legible y elegante.

Las aplicaciones a las que puede ir dirigido este software de cálculo son múltiples. Destacar tanto las aplicaciones docentes dirigidas al ámbito matemático (Botana, Abánades, & Escribano, 2012), así como las dirigidas al ámbito docente dentro de la ingeniería (Elvira, Nájera, Rico, & Gil-Martínez, 2012; Elvira, Rico, & Gil-Martínez, 2011).

1.1 Características fundamentales de Sage

Para acceder a la aplicación hay que tener un registro de usuario con un identificador de usuario y una contraseña. La capacidad de dar de alta a nuevos usuarios recae sobre el administrador de la aplicación, si bien, éste puede configurarla para que sean los propios usuarios los que se registren de forma autónoma.

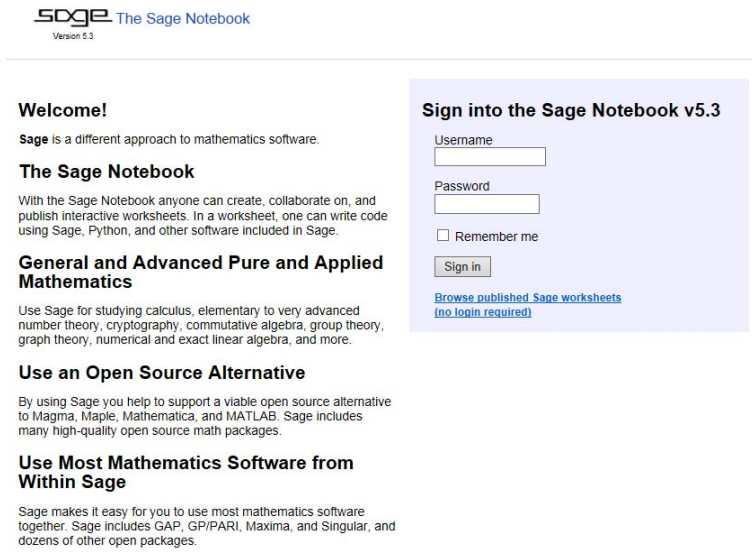
La instalación de Sage es recomendable realizarla sobre un sistema operativo servidor, y preferiblemente, siguiendo la política de software libre, usar cualquier distribución de Linux (Ubuntu, Fedora, OpenSuse, etc.). Existen paquetes binarios precompilados para dichas distribuciones, aunque es recomendable realizar la compilación del código original siguiendo las instrucciones ("Sage Installation," 2012). Es importante señalar que también existen otras alternativas para poder instalar la aplicación en otros sistemas operativos (MAC OSX, Solaris, Windows). En el caso de realizar la instalación sobre un sistema operativo Windows, la alternativa disponible es trabajar sobre una máquina virtual con la pérdida de rendimiento que ello genera.

Uno de los aspectos destacables de la aplicación es su característica cliente-servidor. Por un lado se encuentra la aplicación instalada en un equipo servidor, y por otro lado, múltiples clientes acceden simultáneamente desde otros equipos cliente a la aplicación. Este acceso se realiza mediante un navegador web, simplemente conociendo la dirección web de acceso. En resumen, el uso del software es independiente del tipo plataforma utilizada, siempre y cuando se disponga de un navegador web. El acceso a la aplicación queda restringido a clientes de la misma red privada, a menos que se le conceda una dirección de internet pública al equipo servidor. En este caso, se podrá acceder al equipo desde cualquier ubicación física que disponga conexión a Internet (Figura 1).

Además de poder trabajar con el software mediante la estructura de red cliente-servidor, también se puede trabajar directamente en el servidor, eliminando de este modo los tiempos de latencia provocados por las comunicaciones de red. Mediante esta forma de trabajo local el programador puede prescindir del entorno gráfico utilizado a través del navegador web y

programar directamente las sentencias en línea de comandos, ganando tiempo al eliminar las latencias propias de las transcripciones de código utilizadas por los intérpretes de *HTML*.

Figura 1: Pantalla de entrada al servidor de la aplicación Sage.



Una vez que el usuario ha accedido al software, éste ya tiene a su disposición lo que se conoce como cuaderno de programación (notebook) compuesto de hojas de cálculo (worksheets). El usuario puede gestionar todas sus hojas (Figura 2), las cuales pueden encontrarse en tres estados distintos: activas, archivadas (almacenadas) o eliminadas (en papelera). Señalar que las hojas son archivos *ASCII* con extensión *SWS* y que están codificados en HTML y XML. El usuario puede crear nuevas hojas, descargar las hojas que desee en formato comprimido al ordenador cliente o realizar la carga de hojas desde el cliente local al servidor remoto. Una característica relevante es la posibilidad de trabajar de forma colaborativa dentro de la aplicación. Para ello cada usuario propietario de una o varias hojas del cuaderno, puede compartirlas con otros usuarios (llamados colaboradores) y trabajar conjuntamente sobre ellas.

A su vez, cada una de las hojas está estructurada en celdas donde el usuario interactúa con la aplicación. Existen los siguientes tipos de celdas:

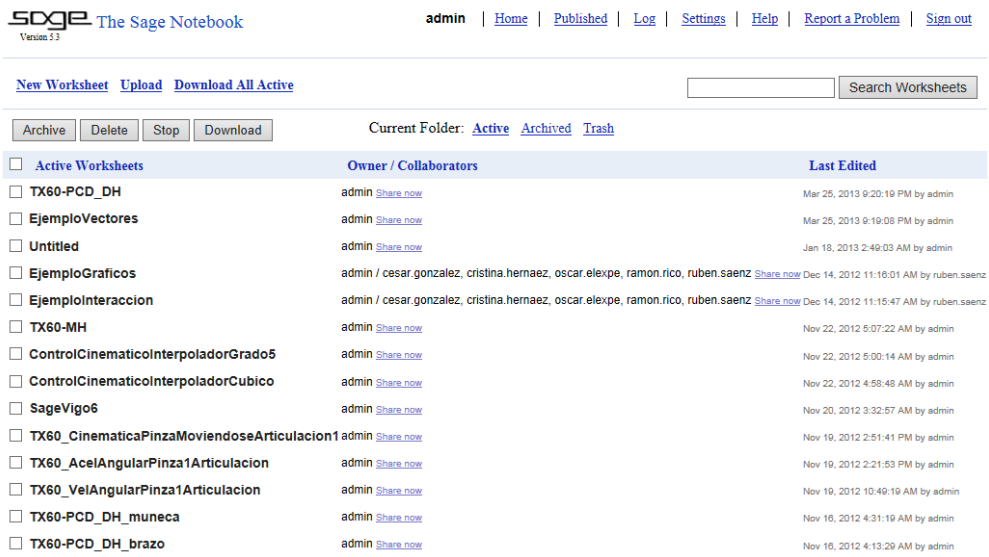
- Celdas de cómputo matemático de entrada: permiten al usuario introducir sentencias de programación que serán enviadas, interpretadas y ejecutadas por el servidor.
- Celdas de cómputo matemático de salida: permiten al usuario recibir desde el servidor los resultados (matemáticos, gráficos, etc.) de la programación realizada.
- Celdas sin cómputo: no permiten realizar programación de tipo matemático. Son celdas con codificación HTML, interesantes para añadir enunciados de problemas, manuales de usuario, foros de debate, etc.

Como en cualquier gestor de contenidos la administración recae sobre un usuario con más privilegios disponibles sobre la aplicación que el resto de usuarios. Entre las capacidades que tiene el administrador de la aplicación cabe destacar:

- Administración de cuentas de usuario (identificador de acceso, contraseñas, etc.).
- Configuración de parámetros de apariencia del entorno (idioma, estructura de filas y columnas, etc.).

- Gestión de parámetros de seguridad (puerto de acceso, tipos de autentificación, *reCAPTCHA*, etc)
- Administración del cuaderno hojas de programación global a todos los usuarios.
- Gestión de los últimos registros de uso de la aplicación.

Figura 2: Pantalla con las hojas del cuaderno del usuario en Sage.



1.2 Infraestructura cliente-servidor en el Departamento de Ingeniería Eléctrica de la Universidad de La Rioja

La docencia impartida por el Departamento de Ingeniería Eléctrica necesita de software de simulación en muchas asignaturas. Para ello la universidad dispone de licencias de software propietario Matlab y Mathematica que pueden ser utilizadas por los profesores del departamento. Uno de los condicionantes para el uso de estas licencias es la limitación de uso de ciertos paquetes adicionales (toolboxes). En concreto para realizar cálculo simbólico no se dispone licencia de dicho paquete adicional, por lo cual, fue necesario buscar una alternativa económica para su uso docente. Sage cumplió todos los requisitos para ser seleccionado.

La infraestructura montada inicialmente en el año 2010 en el Departamento de Ingeniería Eléctrica en la Universidad de La Rioja, estaba constituida por tres equipos servidores obsoletos que daban soporte al profesorado y alumnos desde dentro y fuera de la universidad (Elvira, 2010). A partir de entonces se ha ido mejorando sustancialmente el equipamiento para dar un servicio garantizado y continuado al personal que hace uso de la misma.

En la infraestructura disponible a día de hoy hay dos equipos servidores disponibles para ejecutar el software: uno de ellos está conectado a Internet y el otro sólo permite acceso a ciertos equipos de una VLAN (*Virtual Local Area Network*) creada en la universidad para los equipos de aulas y laboratorios de trabajo de los alumnos. En la Tabla 1 se muestran las características del hardware de los equipos utilizados como servidores.

Los equipos servidores disponen de características y elementos hardware, necesarios para dar servicio continuado de la aplicación Sage a prueba de múltiples fallos:

- Fuentes de alimentación redundantes, con conectividad a líneas independientes de red y con un sistema de alimentación ininterrumpida.

- Sistema de almacenamiento redundante local basado en tarjeta hardware *RAID* con gestión de múltiples discos duros en redundancia *RAID1* o *RAID5*.
- Configuración y gestión automática de copias de seguridad locales y remotas.

Tabla 1. Datos de los servidores Sage en el Departamento de Ingeniería Eléctrica.

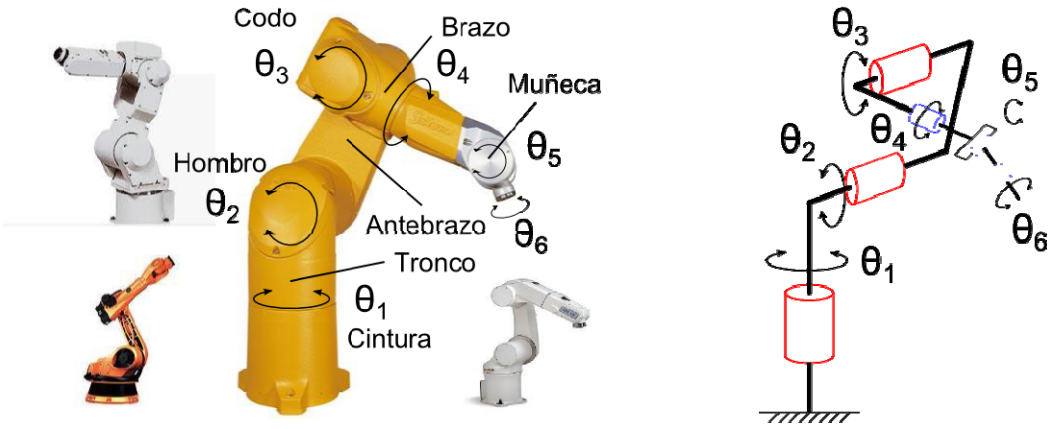
Familia	Hardware µP	Memoria	Discos duros
HP DL585 G1	4 x Opteron DC	24 GBytes	3x300GB (Raid5)
IBM xSeries 360	4 x Xeon 2.0 GHz	4 GBytes	2x32.6GB(Raid1)

Además de los servidores citados anteriormente, señalar hay otras muchos servidores públicos ("Sage Public Server," 2012) con esta aplicación, con grandes prestaciones de cómputo y de velocidad de comunicación de red. Su uso puede ser recomendado para instituciones donde no dispongan de los medios materiales para montar, configurar y mantener su propio servidor.

2. Aplicación de Sage al cálculo simbólico de robots

Un robot es una cadena cinemática abierta constituida por eslabones y articulaciones. Cada una de las articulaciones q_i produce un movimiento relativo entre dos eslabones adyacentes. Dicho movimiento puede ser de tipo lineal en cuyo caso se denominara a la articulación traslacional d_i o de tipo angular en una articulación rotacional θ_i.

Figura 3: Robots industriales con estructura antropomórfica. Representación simbólica.



Una de las estructuras robóticas más utilizadas es la antropomórfica (Figura 3), formada por un brazo con tres articulaciones rotacionales $\{\theta_1, \theta_2, \theta_3\}$, así como una muñeca compuesta por otras tres articulaciones rotacionales $\{\theta_4, \theta_5, \theta_6\}$. Su semejanza a la estructura morfológica humana se puede observar en los elementos que constituyen los tres primeros eslabones (tronco, antebrazo y brazo) movidos a través de las tres primeras articulaciones (cintura, hombro y codo). Si bien la función principal del brazo es posicionar la herramienta, garra o efector terminal (dedos de la mano), la función de la muñeca consiste en realizar su orientación adecuada. El movimiento combinado de las seis articulaciones permite ubicar el efector terminal allí donde se desee y según la aplicación particular del robot a la que va a ir dirigido (desplazamiento de objetos, soldadura, puntura, etc.).

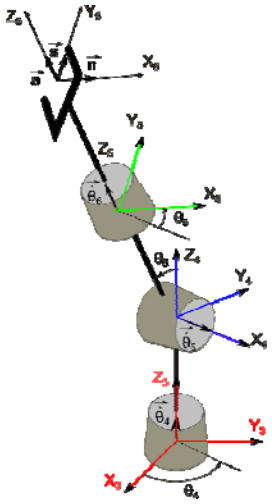
Dado que muchos robots industriales disponen de la misma estructura (antropomórfica, SCARA, cilíndrica, cartesiana), algunos autores (Siciliano, 2009) utilizan una representación simbólica del robot (Figura 3).

2.1 Estudio de cambios de orientación producidos por la muñeca de Euler

El robot anterior posee una muñeca compuesta tipo Euler (Figura 4) caracterizada por la composición de tres movimientos de orientación (Spong & Vidyasagar, 1989):

- Una rotación de valor α en torno a un eje Z.
- Una rotación de valor β en torno a un eje Y.
- Una rotación de valor γ en torno a un eje Z.

Figura 4: Muñeca de Euler compuesta con tres grados de libertad.



Estas tres rotaciones se realizan en sistemas de referencia móviles, y el conjunto formado por los 3 ángulos de Euler $[\alpha, \beta, \gamma]^T$ constituye una representación mínima del atributo de orientación de cualquier sólido.

$$R = R_{Z,\alpha} \cdot R_{Y,\beta} \cdot R_{Z,\gamma} = \begin{bmatrix} c_\alpha c_\beta c_\gamma - s_\alpha s_\gamma & -c_\alpha c_\beta s_\gamma - s_\alpha c_\gamma & c_\alpha s_\beta \\ s_\alpha c_\beta c_\gamma + c_\alpha s_\gamma & -s_\alpha c_\beta s_\gamma + c_\alpha c_\gamma & s_\alpha s_\beta \\ -s_\beta c_\gamma & s_\beta s_\gamma & c_\beta \end{bmatrix} \quad (1)$$

En la Figura 5 se muestra la programación utilizando Sage (Finch, 2011) de las matrices simbólicas tridimensionales así como algunas operaciones algebraicas fundamentales.

Además del ejemplo de descripción de orientación mediante los ángulos de Euler, también se puede obtener otras representaciones de la orientación en el espacio tridimensional conmutando el orden de los giros, modificando los ejes de giro e incluso realizando los cambios de orientación simples en sistemas de referencia fijos. Por último señalar que también se pueden obtener modelos matriciales para buscar cambios de orientación en cualquier plano principal XY, YZ, ZX.

Figura 5: Cálculo de la matriz de orientación de Euler simbólica mediante Sage.

```
var('alpha','beta','gamma');
Rot1=MatrixSpace(SR,3,3);Rot2=MatrixSpace(SR,3,3);Rot3=MatrixSpace(SR,3,3);
Rot1=matrix([[cos(alpha),-sin(alpha),0],[sin(alpha),cos(alpha),0],[0,0,1]])
Rot2=matrix([[cos(beta),0,-sin(beta)],[0,1,0],[sin(beta),0,cos(beta)]]);
Rot3=matrix([[cos(gamma),-sin(gamma),0],[sin(gamma),cos(gamma),0],[0,0,1]]);
MunecaEuler=Rot1*Rot2*Rot3;
html('<font color="blue">Matriz de Euler: $\mathbf{R}_{(\\alpha,\\beta,\\gamma)} = %s$</font>'%latex(MunecaEuler))
```

$$\text{Matriz de Euler: } \mathbf{R}_{(\alpha,\beta,\gamma)} = \begin{pmatrix} \cos(\alpha)\cos(\beta)\cos(\gamma) - \sin(\alpha)\sin(\gamma) & -\sin(\gamma)\cos(\alpha)\cos(\beta) - \sin(\alpha)\cos(\gamma) & -\sin(\beta)\cos(\alpha) \\ \sin(\alpha)\cos(\beta)\cos(\gamma) + \sin(\gamma)\cos(\alpha) & -\sin(\alpha)\sin(\gamma)\cos(\beta) + \cos(\alpha)\cos(\gamma) & -\sin(\alpha)\sin(\beta) \\ \sin(\beta)\cos(\gamma) & -\sin(\beta)\sin(\gamma) & \cos(\beta) \end{pmatrix}$$

2.2 Estudio de la cinemática de la pinza del robot moviéndose una articulación

Utilizando los métodos de la mecánica clásica, se propone en este ejemplo el estudio de los vectores de posición, velocidad y aceleración mediante la derivación de vectores. Para ello se va a estudiar la cinemática de movimiento del punto final del robot (Figura 6) calculando los vectores en un sistema de referencia fijo o estacionario $\{X_0, Y_0, Z_0\}$, y en un sistema de referencia móvil que cambia su orientación $\{X_1, Y_1, Z_1\}$ debido al movimiento exclusivo de la primera articulación θ_1.

En el sistema de referencia fijo la obtención de los vectores velocidad ${}^0\mathbf{v}_p$ y aceleración ${}^0\mathbf{a}_p$ se obtiene por derivación directa del vector de posición ${}^0\mathbf{r}_p$, (Danta, 2010), siendo éste un vector no constante dependiente del valor de la primera articulación θ_1.

$$ {}^0\vec{r}_p = \begin{pmatrix} -bs_1 \\ bc_1 \\ 0 \end{pmatrix} \quad {}^0\vec{v}_p = \begin{pmatrix} -b\dot{\theta}_1 c_1 \\ -b\dot{\theta}_1 s_1 \\ 0 \end{pmatrix} \quad {}^0\vec{a}_p = \begin{pmatrix} -b\ddot{\theta}_1 c_1 + b\dot{\theta}_1^2 s_1 \\ -b\ddot{\theta}_1 s_1 - b\dot{\theta}_1^2 c_1 \\ 0 \end{pmatrix} \tag{2} $$

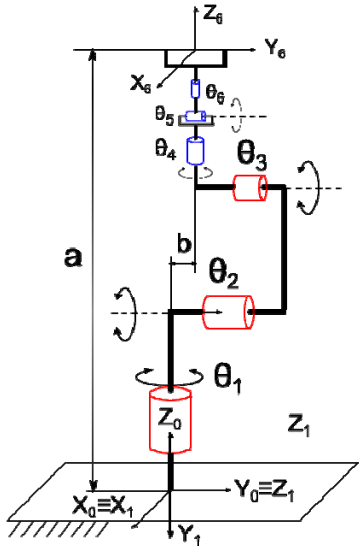
Sin embargo, el vector de la posición del efector final del robot ${}^1\mathbf{r}_p$ será constante en el sistema de referencia móvil $\{X_1, Y_1, Z_1\}$ solidario con dicho movimiento, y los vectores ${}^1\mathbf{v}_p$ y ${}^1\mathbf{a}_p$ conlleva el cálculo de un producto vectorial del vector velocidad angular de la referencia móvil y el vector que se está derivando (Targ, 2008).

$$ {}^1\vec{r}_p = \begin{pmatrix} 0 \\ -a \\ p \end{pmatrix} \quad {}^1\vec{v}_p = \frac{d\,{}^1\vec{r}_p{}^T}{dt} + \left({}^1\vec{\Omega}_1 \wedge {}^1\vec{r}_p\right) = \begin{pmatrix} -b\dot{\theta}_1 c_1 \\ -b\dot{\theta}_1 s_1 \\ 0 \end{pmatrix} \quad {}^1\vec{a}_p = \frac{d\,{}^1\vec{a}_p{}^T}{dt} + \left({}^1\vec{\Omega}_1 \wedge {}^1\vec{v}_p\right) = \begin{pmatrix} -b\ddot{\theta}_1 \\ 0 \\ -b\dot{\theta}_1^2 \end{pmatrix} \tag{3} $$

Ambos resultados anteriores están relacionados mediante la matriz de cambio de orientación ${}^0R_1$ variable según el valor de la primera articulación θ_1, según la expresión siguiente:

$$ \begin{aligned} {}^0\vec{r}_p &= {}^0R_1(\theta_1)\cdot{}^1\vec{r}_p \\ {}^0\vec{v}_p &= {}^0R_1(\theta_1)\cdot{}^1\vec{v}_p \\ {}^0\vec{a}_p &= {}^0R_1(\theta_1)\cdot{}^1\vec{a}_p \end{aligned} \quad donde: {}^0R_1(\theta_1) = R_{Z,\theta_1}\cdot R_{X,-90^\circ} \tag{4} $$

Figura 6: Representación simbólica del robot moviéndose la primera articulación θ_1.



En la Figura 7 se muestra la programación en Sage del código (Finch, 2011) para la obtención de dichos vectores simbólicos. Se puede observar en el código la posibilidad de insertar fórmulas matemáticas escritas en *LaTex*.

Figura 7: Celda de cómputo de entrada para el cálculo de la cinemática de movimiento del efector terminal según el movimiento de la primera articulación.

```
var('theta,t,r,a,b,omega,alpha')
tita(t)=function('theta',t); # Definición de la variable de articulación tita1 = f(t) en función del tiempo
dtita(t)=derivative(tita,t); # Velocidad de la variable de articulación tita1 = f(t) en función del tiempo
ddtita(t)=derivative(dtita,t); # Aceleración de la variable de articulación tita1 = f(t) en función del tiempo
def Rot01(a):
    var("alpha");alpha=a;
    R=MatrixSpace(SR,3,3);
    R1=matrix([[cos(alpha),0,-sin(alpha)],[sin(alpha),0,cos(alpha)],[0,-1,0]]);return R1
R01=Rot01(tita(t))
show('Matriz de cambio de orientación entre la referencia móvil SR1 y la referencia fija SR0')
html('<center><font color="green">$^0\mathbf{R}_1 (\\theta)= %s$</font></center>'%latex(R01))

Omega0=vector(SR,[0,0,0]);Omega1=vector(SR,[0,-dtita,0]);
dtita(t)=omega;
show('Vector de velocidad angular de la referencia móvil expresado en ella misma SR1');
html('<center><font color="green">$^1\mathbf{\Omega}_1 (\\theta)= %s$</font></center>'%latex(Omega1.column()))

rP1=vector(SR,[0,-a,b]);
show('Vector de posición en la referencia móvil no inercial SR1')
html('<center><font color="green">$^1\mathbf{r}_P (\\theta)= %s$</font></center>'%latex(rP1.column()))
show('Vector de posición en la referencia fija SR0')
rP0=R01*rP1 # Vector de posición en la referencia fija inercial SR0
html('<center><font color="green">$^0\mathbf{r}_P (\\theta)= %s$</font></center>'%latex(rP0.column()))

vP0=derivative(rP0,t)+Omega0.cross_product(rP0);
show('Vector de velocidad en la referencia fija SR0');
html('<center><font color="green">$^0\mathbf{v}_P (\\theta)= %s$</font></center>'%latex(vP0.column()))
vP1=derivative(rP1,t)+Omega1.cross_product(rP1);
show('Vector de velocidad en la referencia móvil SR1');
html('<center><font color="green">$^1\mathbf{v}_P (\\theta)= %s$</font></center>'%latex(vP1.column()))

# Cálculo de los vectores de aceleración en ambas referencias
aP0=derivative(vP0,t)+Omega0.cross_product(vP0);
show('Vector de aceleración en la referencia fija SR0');
html('<center><font color="green">$^0\mathbf{a}_P (\\theta)= %s$</font></center>'%latex(aP0.column()))
aP1=derivative(vP1,t)+Omega1.cross_product(vP1);
show('Vector de aceleración en la referencia móvil SR1');
html('<center><font color="green">$^1\mathbf{a}_P (\\theta)= %s$</font></center>'%latex(aP1.column()))
```

Figura 8: Celda de cómputo de salida para el cálculo de la cinemática de movimiento del efector terminal según el movimiento de la primera articulación.

Matriz de cambio de orientación entre la referencia móvil SR1 y la referencia fija SR0

$$ {}^0\mathbf{R}_1(\theta) = \begin{pmatrix} \cos(\theta(t)) & 0 & -\sin(\theta(t)) \\ \sin(\theta(t)) & 0 & \cos(\theta(t)) \\ 0 & -1 & 0 \end{pmatrix} $$

Vector de velocidad angular de la referencia móvil expresado en ella misma SR1

$$ {}^1\mathbf{\Omega}_1(\theta) = \begin{pmatrix} 0 \\ -D[0](\theta)(t) \\ 0 \end{pmatrix} $$

Vector de posición en la referencia fija SR0

$$ {}^0\mathbf{r}_P(\theta) = \begin{pmatrix} -b\sin(\theta(t)) \\ b\cos(\theta(t)) \\ a \end{pmatrix} $$

Vector de posición en la referencia móvil no inercial SR1

$$ {}^1\mathbf{r}_P(\theta) = \begin{pmatrix} 0 \\ -a \\ b \end{pmatrix} $$

Vector de velocidad en la referencia fija SR0

$$ {}^0\mathbf{v}_P(\theta) = \begin{pmatrix} -b\cos(\theta(t))\,D[0](\theta)(t) \\ -b\sin(\theta(t))\,D[0](\theta)(t) \\ 0 \end{pmatrix} $$

Vector de velocidad en la referencia móvil SR1

$$ {}^1\mathbf{v}_P(\theta) = \begin{pmatrix} -bD[0](\theta)(t) \\ 0 \\ 0 \end{pmatrix} $$

Vector de aceleración en la referencia fija SR0

$$ {}^0\mathbf{a}_P(\theta) = \begin{pmatrix} b\sin(\theta(t))\,D[0](\theta)(t)^2 - b\cos(\theta(t))\,D[0,0](\theta)(t) \\ -b\cos(\theta(t))\,D[0](\theta)(t)^2 - b\sin(\theta(t))\,D[0,0](\theta)(t) \\ 0 \end{pmatrix} $$

Vector de aceleración en la referencia móvil SR1

$$ {}^1\mathbf{a}_P(\theta) = \begin{pmatrix} -bD[0,0](\theta)(t) \\ 0 \\ -bD[0](\theta)(t)^2 \end{pmatrix} $$

En la Figura 8 se muestra una celda salida de una hoja en Sage con los resultados gráficos obtenidos en el explorador web a partir de la programación realizada en la celda de entrada anterior.

2.3 Cálculo de matrices de transformación homogéneas entre distintos sistemas de referencia

Una matriz de transformación homogénea es un modelo matemático matricial que describe los siguientes atributos sobre un objeto: posición, orientación, perspectiva y escalado (Fu, González, & Lee, 1990). En la descripción de los sólidos (eslabones y articulaciones) de un robot sólo se utilizan los dos atributos que describen la ubicación, dejando un factor de escala unitario y anulando los tres parámetros de cambio de perspectiva.

$$H = \begin{bmatrix} r_{11} & r_{12} & r_{13} & d_x \\ r_{21} & r_{22} & r_{23} & d_y \\ r_{31} & r_{32} & r_{33} & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

Según los parámetros dimensionales mostrados en la Figura 9 en el robot, se realiza el cálculo de las siguientes matrices de transformación homogéneas:

- Matriz de transformación homogénea que describe el sistema de referencia del centro de masas del eslabón fijo 0 respecto al sistema de referencia fijo $\{X_0, Y_0, Z_0\}$: ${}^{0}H_{cdm0}$.
- Matriz de transformación homogénea que describe el sistema de referencia del centro de masas del eslabón móvil 1 respecto al sistema de referencia colocado en del centro de masas del eslabón fijo 0: ${}^{cdm0}H_{cdm1}$.
- Matriz de transformación homogénea que describe el sistema de referencia del centro de masas del eslabón móvil 1 respecto al sistema de referencia $\{X_0, Y_0, Z_0\}$: ${}^{0}H_{cdm1}$.

Los resultados de la programación (Figura 10) muestran la aplicación de Sage en la creación y manipulación de matrices simbólicas de dimensión 4x4, así como algunas operaciones algebraicas básicas. La aplicación Sage permite incorporar codificación *HTML* dentro de los programas permitiendo mejores resultados gráficos.

Figura 9: Representación de los sistemas de referencia en los eslabones 0 y 1 del robot.

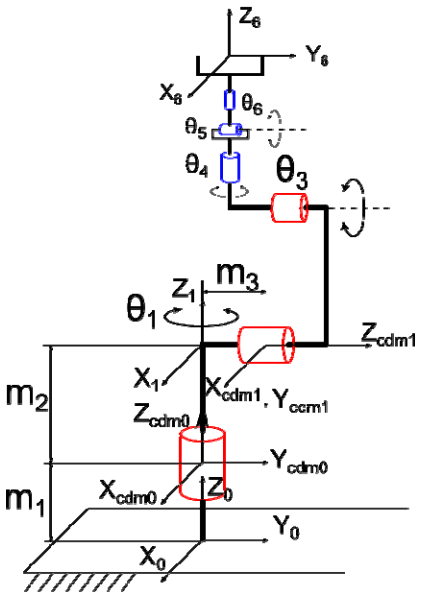


Figura 10: Celdas de cómputo de entrada y salida para el cálculo de las matrices de transformación homogéneas de eslabones.

```
var("theta,m1,m2,m3")    # Defino variables
H0cdm0=matrix(SR,[[1,0,0,0],[0,1,0,0],[0,0,1,m1],[0,0,0,1]]);
show('Matriz de transformación homogénea constante relacionando SR0 y SRcdm0')
html('<center><font color="green">$^0\mathbf{H}_{cdm_0}= %s$</font></center>'%latex(H0cdm0))

R01=matrix(SR,[[cos(theta),0,-sin(theta)],[sin(theta),0,cos(theta)],[0,-1,0]]);
rcdm0cdm1=matrix(SR,[[0],[m2],[m3]])    # Defino el vector columna de posición
Fila=matrix(SR,[[0,0,0,1]])    # Defino la última fila de la matriz de Transformación Homogénea
Hcdm0cdm1=R01.augment(rcdm0cdm1)    # Añado el vector de posición a la matriz de rotación
Hcdm0cdm1=Hcdm0cdm1.stack(Fila)    # Añado la última fila a la matriz de Transformación Homogénea
show('Matriz de transformación homogénea variable relacionando SRcdm0 y SRcdm1 ')
html('<center><font color="green">$^{cdm_0}\mathbf{H}_{cdm_1}= %s$</font></center>'%latex(Hcdm0cdm1))

H0cdm1=H0cdm0*Hcdm0cdm1;
show('Matriz de transformación homogénea variable relacionando SR0 y SRcdm1 ')
html('<center><font color="green">$^{0}\mathbf{H}_{cdm_1}= %s$</font></center>'%latex(H0cdm1))
```

Matriz de transformación homogénea constante relacionando SR0 y SRcdm0

$$^{0}\mathbf{H}_{cdm_0} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & m_1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Matriz de transformación homogénea variable relacionando SRcdm0 y SRcdm1

$$^{cdm_0}\mathbf{H}_{cdm_1} = \begin{pmatrix} \cos(\theta) & 0 & -\sin(\theta) & 0 \\ \sin(\theta) & 0 & \cos(\theta) & m_2 \\ 0 & -1 & 0 & m_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Matriz de transformación homogénea variable relacionando SR0 y SRcdm1

$$^{0}\mathbf{H}_{cdm_1} = \begin{pmatrix} \cos(\theta) & 0 & -\sin(\theta) & 0 \\ \sin(\theta) & 0 & \cos(\theta) & m_2 \\ 0 & -1 & 0 & m_1 + m_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

2.4 Cálculo de la matriz de transformación homogénea siguiendo la notación de Denavit-Hartenberg

En los robots se utiliza un sistema de referencia fijo {X_0, Y_0, Z_0} asociado al eslabón fijo de referencia (tierra). En la cadena cinemática del robot, para cada eslabón disponible i-ésimo se asocia un sistema de referencia móvil {X_i, Y_i, Z_i}. La descripción de la ubicación de dos eslabones adyacentes se realiza utilizando una matriz de transformación homogénea cumpliendo las dos condiciones de Denavit-Hartenberg (Paul, 1984):

- El eje X_i es perpendicular al eje Z_{i-1}.
- Los ejes X_i y Z_{i-1} se cortan.

Dando origen a la matriz de Denavit-Hartenberg:

$$^{i-1}H_i(q_i) = \begin{bmatrix} c_{\vartheta_i} & -s_{\vartheta_i}c_{\alpha_i} & s_{\vartheta_i}s_{\alpha_i} & a_i c_{\vartheta_i} \\ s_{\vartheta_i} & c_{\vartheta_i}c_{\alpha_i} & -c_{\vartheta_i}s_{\alpha_i} & a_i s_{\vartheta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6)$$

En la Figura 11, se muestra el ejemplo de programación en Sage. Destacar en este ejemplo la programación de funciones utilizando Python, así como la programación de estructuras de control para realizar la simplificación trigonométrica de las componentes de la matriz.

Mediante las dos condiciones indicadas anteriormente se logran relacionar ambos sistemas de referencia utilizando únicamente cuatro parámetros, en lugar de seis que son los necesarios para describir una orientación y una posición de forma genérica. De estos cuatro

parámetros de Denavit-Hartenberg tres son constantes y uno de ellos es la propia variable de la articulación de giro θ_i o articulación traslacional d_i. Existen otras represtaciones para la descripción de dos eslabones adyacentes (J.Craig, 2005).

Figura 11: Celdas de cómputo de entrada y salida para el cálculo de la matriz de transformación homogénea de Denavit-Hartenberg.

```
def MHDH(titai,alfai,ai,di):
    var("alpha,theta,a,d");alpha=alfai;theta=titai;a=ai;d=di;
    H=MatrixSpace(SR,4,4);
    H1=matrix([[cos(theta),-sin(theta)*cos(alpha),sin(theta)*sin(alpha),a*cos(theta)],[sin(theta),cos
(theta)*cos(alpha),-cos(theta)*sin(alpha),a*sin(theta)],[0,sin(alpha),cos(alpha),d],[0,0,0,1]]);
    for k1 in srange(4):
        for k2 in srange(4):
            if H1[k1,k2]<> 0:
                H1[k1,k2]=H1[k1,k2].full_simplify()
    return H1;
var('theta,alpha,a,d')
H=MHDH(theta,alpha,a,d);
show('Matriz de transformación homogénea de Denavit-Hartenberg')
html('<center><font color="blue">$^{i-1}\mathbf{H}_{i} (q_i)= %s$</font></center>'%latex(H))
```

Matriz de transformación homogénea de Denavit-Hartenberg

$$ {}^{i-1}\mathbf{H}_i(q_i) = \begin{pmatrix} \cos(\theta) & -\sin(\theta)\cos(\alpha) & \sin(\alpha)\sin(\theta) & a\cos(\theta) \\ \sin(\theta) & \cos(\alpha)\cos(\theta) & -\sin(\alpha)\cos(\theta) & a\sin(\theta) \\ 0 & \sin(\alpha) & \cos(\alpha) & d \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

2.5 Resolución del problema cinemático de ubicación del brazo del robot siguiendo la notación de Denavit-Hartenberg

Colocando los cuatro primeros sistemas de referencia (Figura 12) en el brazo del robot siguiendo la notación de Denavit-Hartenberg (Barrientos, 2007), se obtiene la Tabla 2 de parámetros de Denavit-Hartenberg.

Mediante la multiplicación de las tres matrices de transformación homogéneas de los eslabones del brazo se puede obtener la ubicación del punto final del brazo (sistema de referencia $\{X_3, Y_3, Z_3\}$).

$$ {}^{0}H_3(\theta_1,\theta_2,\theta_3) = {}^{0}H_1(\theta_1)\cdot {}^{1}H_2(\theta_2)\cdot {}^{2}H_3(\theta_3) \qquad (7) $$

Figura 12: Representación de los sistemas de referencia del brazo del robot según la notación Denavit-Hartenberg.

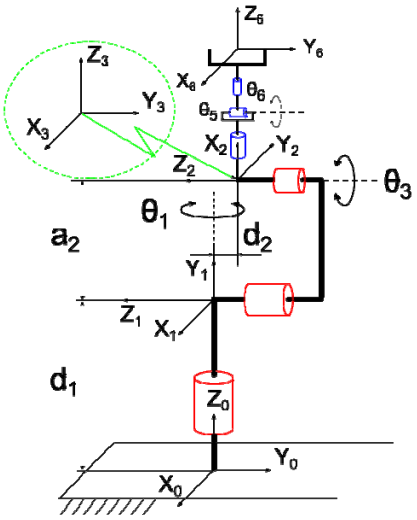


Tabla 2. Parámetros de Denavit-Hartenberg del brazo.

θ_i	d_i	a_i	α_i
θ_1	d_1	0	90º
θ_2+90º	$-d_2$	a_0	0º
θ_3-90º	0	0	-90º

En la Figura 13 se muestran los resultados en Sage. En la programación del ejercicio se ha conseguido realizar la simplificación trigonométrica de la matriz llamando directamente a procedimientos incorporados en el paquete Máxima.

Figura 13: Celdas de cómputo de entrada y salida para la resolución del problema cinemático directo del brazo robótico.

```
def MHDH(titai,alfai,ai,di):
    var("alpha,theta,a,d");alpha=alfai;theta=titai;a=ai;d=di;
    H=MatrixSpace(SR,4,4);
    H1=matrix([[cos(theta),-sin(theta)*cos(alpha),sin(theta)*sin(alpha),a*cos(theta)],[sin(theta),cos(theta)*cos
(alpha),-cos(theta)*sin(alpha),a*sin(theta)],[0,sin(alpha),cos(alpha),d],[0,0,0,1]]);
    for k1 in srange(4):
        for k2 in srange(4):
            if H1[k1,k2]<> 0:
                H1[k1,k2]=H1[k1,k2].full_simplify()
    return H1;
var('theta1,theta2,theta3,d1,d2,a2')
H01=MHDH(theta1,pi/2,0,d1);
H12=MHDH(theta2+pi/2,0,a2,-d2);
H23=MHDH(theta3-pi/2,-pi/2,0,0);
H02=H01*H12;
for k1 in srange(4):
    for k2 in srange(4):
            if H02[k1,k2]<> 0:
                H02[k1,k2]=H02[k1,k2].full_simplify()

H03=H02*H23;
for k1 in srange(4):
    for k2 in srange(4):
            if H03[k1,k2]<> 0:
                H03[k1,k2]=H03[k1,k2].factor();
                H03[k1,k2]=maxima(H03[k1,k2]).trigreduce().sage()
html('<center><font>$^0\mathbf{H}_{3} (\\theta_1,\\theta_2,\\theta_3)= %s$</font></center>'%latex(H03))
```

$$
{}^{0}\mathbf{H}_{3}(\theta_1,\theta_2,\theta_3) = \begin{pmatrix}
\cos(\theta_2+\theta_3)\cos(\theta_1) & -\sin(\theta_1) & -\sin(\theta_2+\theta_3)\cos(\theta_1) & -\frac{1}{2}a_2\sin(-\theta_1+\theta_2)-\frac{1}{2}a_2\sin(\theta_1+\theta_2)-d_2\sin(\theta_1) \\
\sin(\theta_1)\cos(\theta_2+\theta_3) & \cos(\theta_1) & -\sin(\theta_2+\theta_3)\sin(\theta_1) & -\frac{1}{2}a_2\cos(-\theta_1+\theta_2)+\frac{1}{2}a_2\cos(\theta_1+\theta_2)+d_2\cos(\theta_1) \\
\sin(\theta_2+\theta_3) & 0 & \cos(\theta_2+\theta_3) & a_2\cos(\theta_2)+d_1 \\
0 & 0 & 0 & 1
\end{pmatrix}
$$

3. Conclusiones

Sage es una herramienta de apoyo a la enseñanza de cualquier asignatura del ámbito de la ingeniería que requiera de formulación simbólica. Es atractiva al ser software de código libre, permitir la programación en un lenguaje Python compacto y legible, y poder ser utilizada desde cualquier navegador web independiente del hardware utilizado.

Los profesores pueden plantear a los alumnos ejemplos de ejercicios y enunciados de ejercicios. También pueden plantear enunciados de prácticas (en determinadas materias). Los estudiantes pueden utilizar Sage para resolver actividades académicas (ejercicios, prácticas, etc.) planteados por el profesor, comprobar desarrollos complejos matemáticos planteados de forma manuscrita, así como averiguar los errores que se han encontrado. Un correcto uso de la aplicación permite al alumno enriquecerse en el proceso de aprendizaje.

Todos los usuarios del software (profesores y alumnos) pueden trabajar de forma colaborativa compartiendo hojas correspondientes a las tareas que les han sido encomendadas. Si la aplicación de Sage está instalada en un servidor, permite centralizar toda la información en un único equipo, facilitando así el acceso e intercambio de resultados.

Agradecimientos

Los autores agradecen al Gobierno de La Rioja la ayuda prestada en el proyecto de investigación IMPULSA 2010/01 dentro del Plan Riojano de I+D+i 2008-2011.

Referencias

Barrientos, A. (2007). *Fundamentos de robótica*. Mc Graw Hill.

Botana, F., Abánades, M. a., & Escribano, J. (2012). Using a free open source software to teach mathematics. *Computer Applications in Engineering Education. Wiley Periodicals Inc*. doi:10.1002/cae.21565

Danta, M. R. (2010). *Mecánica*. Editorial: Secretariado de publicaciones. Universidad de Sevilla.

Elvira, C. (2010). Control y Programación de robots con Sage. *I Jornadas SAGE/Python*.

Elvira, C., Nájera, S., Rico, J., & Gil-Martínez, M. (2012). Propuesta de prácticas docentes de Ingeniería de Control usando el software Sage. *III Jornadas SAGE/Python*.

Elvira, C., Rico, J., & Gil-Martínez, M. (2011). Primeros pasos en prácticas de control de sistemas dinámicos utilizando Sage. *II Jornadas SAGE/Python*.

Finch, C. (2011). *Sage Beginner's Guide About the Author*. Packt Publishing.

Fu, K. S., González, R. C., & Lee, C. S. G. (1990). *Robótica: Control, detección, visión e inteligencia*. Mc Graw Hill.

J.Craig, J. (2005). *Introduction to robotics. Mechanics and control.* Pearson Education International.

Paul, R. P. (1984). *Robot manipulators. Mechanics, programming and control*. MIT Press.

Sage Installation. (2012). Retrieved from www.sagemath.org/doc/installation

Sage Main Page. (2012). Retrieved from www.sagemath.org

Sage Public Server. (2012). Retrieved from www.sagenb.org

Siciliano, B. (2009). *Robotics. Modelling, planning and control*. Springer.

Spong, M. W., & Vidyasagar, M. (1989). *Robot Dynamics and Control*. *John Wiley & Sons*. John Wiley & Sons.

Targ, S. M. (2008). *Curso breve de mecánica teórica*. MIR.