Máster Universitario en Nuevas Tecnologías de la Informática

TRABAJO FIN DE MÁSTER

Optimización de algoritmos paralelos para análisis cinemático de sistemas multicuerpo basado en Ecuaciones de Grupo

Autor:

José Carlos Cano Lorente

Tutores:

Antonio Javier Cuenca Muñoz

Domingo Giménez Cánovas

Cotutor:

Mariano Saura Sánchez

Departamento de Ingeniería Mecánica

Universidad Politécnica de Cartagena

Murcia, Julio 2017

Índice

Introducción (1/2)

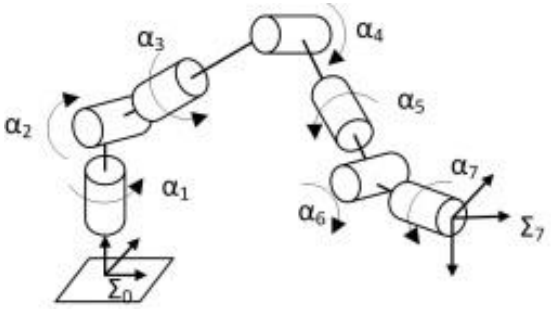
Algoritmos paralelos para **análisis cinemático** de **sistemas multicuerpo** basado en **Ecuaciones de Grupo**

Sistemas multicuerpo: conjunto de cuerpos conectados entre sí mediantes juntas cinemáticas, donde los elementos contiguos tratan de desplazarse unos respecto a otros.

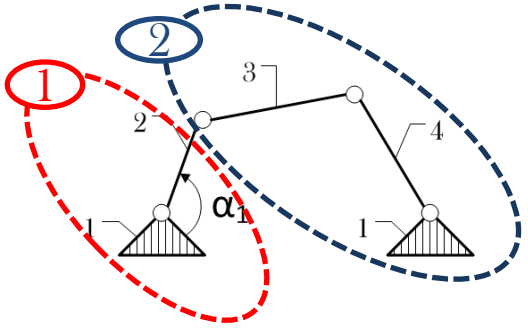
Análisis cinemático: estudio que permite evaluar el movimiento de un mecanismo obteniendo información sobre la posición, velocidad y aceleración de los cuerpos.

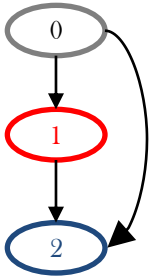
Análisis estructural: descomposición del sistema multicuerpo en Grupos Estructurales.

- Cada Grupo aporta sus propias **Ecuaciones de Grupo**.
- Se obtiene un Grafo Estructural.



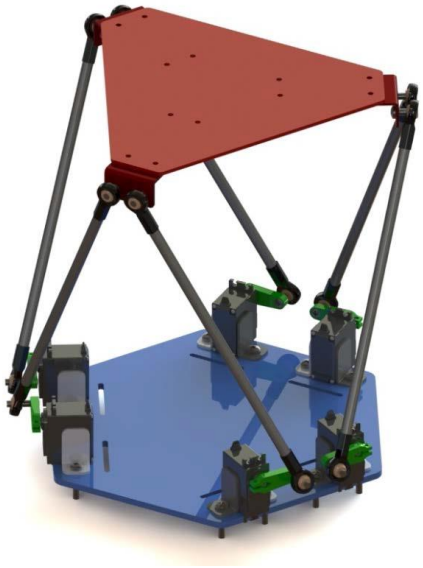
$$(x_1 - x_A)^2 + (y_1 - y_A)^2 - L_1^2 = 0$$
$$(x_2 - x_1)^2 + (y_2 - y_1)^2 - L_2^2 = 0$$
$$(x_3 - x_D)^2 + (y_3 - y_D)^2 - L_3^2 = 0$$





Introducción (2/2)

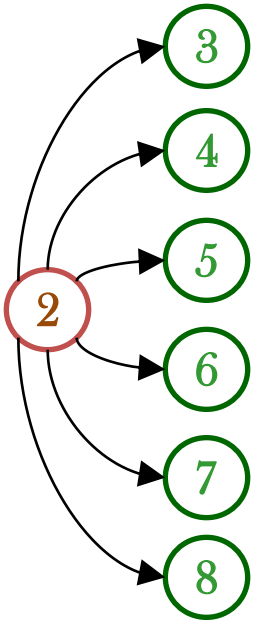
Plataforma de Stewart: sistema multicuerpo, donde se controla la posición (x y z) y la orientación (θ ψ φ) de una parte del mecanismo, en concreto el Terminal.



Las ecuaciones del **Terminal** se expresan como matrices de **12x12**

Las ecuaciones de las **Manivelas** se expresan como matrices de **15x15**

Las matrices obtenidas son **simétricas** y **dispersas**



- Según el diagrama estructural obtenido, **las 6 manivelas se pueden calcular de manera independiente (paralelismo)**, tras resolver el Terminal.

Objetivos

- **Analizar** el software de control de un robot manipulador paralelo de carácter general desarrollado en la UPCT, con un enfoque secuencial y empleando MKL para los cálculos.
- Desarrollar un **simulador** que herede del anterior los algoritmos de resolución de sistemas mecánicos y que permita variar el tamaño del problema para poder extenderlo a soluciones generales.
- Incorporar las rutinas ofrecidas por varias **librerías matriciales**, y analizar su rendimiento en función del factor de dispersión y tamaño de las matrices.
- Aplicar **técnicas de programación paralela**, estudiando las mejoras respecto al método secuencial que ofrecen los sistemas multicore con memoria compartida y multicore con GPUs.
- Crear un proceso de **autotuning** que encuentre las configuraciones del software óptimas para el tamaño del problema a resolver.

Obtener versiones mejoradas del software para:

- **Control** de prototipos reales, haciéndolos susceptibles de ejecutarse en sistemas de placa reducida.
- **Simulación** de nuevos productos (desarrollo virtual del producto).

Herramientas

Compiladores

- **FORTRAN**
- **ANSI C**

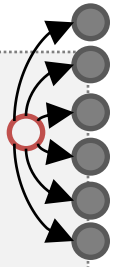
Librerías matriciales

- **MKL:** solver denso (Intel)
- **PARDISO:** solver disperso (Intel)
- **MA27:** solver disperso (parte de la Harwell Subroutine Library)
- **MAGMA: s**olver denso, explotando GPUs (mismos desarrolladores de LAPACK)

Hardware

- **JUPITER:** dos hexa-cores (12 cores) Intel Xeon E5-2620 y 6 GPUs NVIDIA
- **SATURNO:** cuatro hexa-cores (24 cores) Intel Xeon E7530 y 1 GPU NVIDIA

Rendimiento de librerías matriciales (1/3)



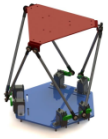
```
1  Cargar en memoria las coordenadas geométricas del modelo
2  for número de iteraciones (tFin) do
3     Resolver la cinemática del Terminal (dimensión nEQTerminal)
4     for all Manivela-Barra (numGE) do
5        for número de iteraciones (tFin2) do
6           Resolver la cinemática del par Manivela-Barra (dimensión nEQManivela)
7        end for
8     end for
9  end for
```

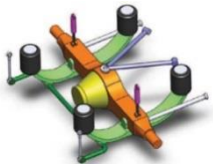
tFin: número de ejecuciones
tFin2: número de veces que se resuelven los sistemas de ecuaciones
numGE: número de grupos paralelizables
nEQTerminal: dimensión de la matriz del terminal
nEQManivela: dimensión de la matriz de las manivelas

- Sin paralelismo explícito, los grupos se calculan secuencialmente.
- Explotaremos paralelismo en las rutinas de las librerías MKL y PARDISO.

Rendimiento de librerías matriciales (2/3)

Influencia del **factor de dispersión** de las matrices en la selección de la librería





nEQTerminal	nEQManivela	Ejecución secuencial 10%	30%	50%	85%	Con paralelismo en rutinas 10%	30%	50%	85%
12	15	MA27	MA27	MA27	MA27	MA27	MA27	MA27	MA27
36	54	MKL	MKL	MKL	MA27	MA27	MKL	MKL	MA27
60	90	MKL	MKL	MKL	MA27	MKL	MKL	MKL	MA27
400	400	MKL	MKL	MKL	MA27	MKL	MKL	MKL	MKL
1000	1000	MKL	MKL	MKL	MA27	MKL	MKL	MKL	MKL
2000	2000	MKL	MKL	MKL	PARD	MKL	MKL	MKL	MKL
3000	3000	MKL	MKL	PARD	PARD	MKL	MKL	MKL	PARD

Experimentos en SATURNO **tFin**=10 **tFin2**=3 **numGE**=6

- Se recomienda MA27 para matrices de pequeñas dimensiones, independientemente de la dispersión.
- MA27 hasta tamaño medio con dispersión alta.
- PARDISO, muy eficiente en matrices grandes dispersas (>85%).

Rendimiento de librerías matriciales (3/3)

Rendimiento con matrices **dispersas** asignando threads las rutinas de las librerías

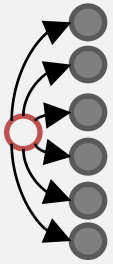
		MA27	MKL		PARDISO	
nEQTerminal	nEQManivela	Secuencial	6 threads	12 threads	4 threads	6 threads
12	15	**0,0022**	0,0088	0,0091	0,0549	0,0592
36	54	**0,0104**	0,0149	0,0149	0,1123	0,1041
60	90	**0,0272**	0,0281	0,0351	0,1505	0,1361
400	400	0,5167	**0,3366**	0,4312	0,9705	1,1421
1000	1000	5,4734	2,8099	**2,2937**	5,6505	5,1878
2000	2000	34,5328	15,4426	**12,2056**	17,0302	20,1799
3000	3000	110,0977	48,9069	35,4511	**34,5061**	35,5385
4000	4000	257,0304	104,3095	80,5172	68,378	**62,5897**

Experimentos en JUPITER **Disp**=85% **tFin**=10 **tFin2**=3 **numGE**=6

- MA27, la más rápida con matrices de pequeñas dimensiones.
- MKL eficiente en tamaños medios, buena respuesta al aumentar los threads
- PARDISO, recomendada en matrices grandes.

Paralelismo en sistemas multicore (1/3)

```
1  for número de iteraciones (tFin) do
2     Resolver la cinemática del Terminal (dimensión nEQTerminal)
3     omp_num_threads ← ArgThreads
4     !$OMP PARALLEL PRIVATE idgrupo
5     !$OMP DO
6     for all Manivela-Barra (numGE) do
7        for número de iteraciones (tFin2) do
8           mkl_num_threads ← ArgMKLThreads
9           Resolver la cinemática del par Manivela-Barra (dimensión nEQManivela)
10       end for
11    end for
12    !$OMP END PARALLEL
13 end for
```



- Creamos **ArgThreads** threads OpenMP, que resuelven cada grupo Manivela-Barra.
- Los cálculos matriciales también pueden explotar paralelismo en MKL y PARDISO, iniciando **ArgMKLThreads** threads anidados (MA27 no implementa paralelismo en sus rutinas).

Paralelismo en sistemas multicore (2/3)

Mejores tiempos de ejecución (segundos) con paralelismo en dos niveles

nEQTerminal	nEQManivela	JUPITER: 12 cores MA27 6 x 1	JUPITER: 12 cores PARDISO 6 x 2	SATURNO: 24 cores MA27 6 x 1	SATURNO: 24 cores PARDISO 6 x 4
12	15	**0,0009**	0,0234	**0,0027**	0,0571
36	54	**0,0043**	0,0288	**0,0034**	0,0475
60	90	**0,0089**	0,043	**0,0088**	0,0682
400	400	**0,1559**	0,4945	**0,2746**	0,6976
1000	1000	**1,6476**	2,4988	**2,6428**	3,5935
2000	2000	10,3387	**7,759**	16,9799	**10,7895**
3000	3000	32,8752	**16,732**	53,5882	**24,8652**
4000	4000	76,5881	**31,6888**	120,8349	**52,3393**

Disp=85% **tFin**=10 **tFin2**=3 **numGE**=6

- MA27, la más rápida con matrices de pequeñas dimensiones, con todos los threads asignados a OpenMP al no disponer de versión paralela.
- PARDISO, recomendada en matrices grandes.

Paralelismo en sistemas multicore (3/3)

Paralelismo con **MKL** en dos niveles, problemas con **más grupos** estructurales **numGE**

numGE	nEQTerminal	nEQManivela	th. OMP x th. MKL 2 x 6	3 x 4	4 x 3	6 x 1	6 x 2	12 x 1
16	12	15	0,0037	0,0029	0,0026	0,0016	0,0022	**0,0016**
	36	54	0,017	0,0127	0,0149	0,0068	0,0076	**0,0052**
	60	90	0,0508	0,0375	0,0335	0,015	0,0196	**0,0114**
	2000	2000	28,8608	24,9983	**24,6512**	41,6372	31,4104	34,685
	3000	3000	77,3555	76,6553	**74,9888**	130,8196	82,6664	106,8146
	4000	4000	**174,6793**	176,9465	175,3777	303,7148	195,164	262,1968
21	12	15	0,0099	0,0037	0,0029	0,0021	0,0021	**0,0019**
	36	54	0,0351	0,0147	0,013	0,0085	0,0149	**0,0079**
	60	90	0,0714	0,0436	0,0369	0,0191	0,0252	**0,0174**
	2000	2000	33,7653	**27,4524**	35,1464	51,3259	34,7575	37,9592
	3000	3000	108,381	**97,2876**	101,759	164,1671	106,8036	122,938
	4000	4000	215,0848	**214,949**	238,1862	378,7186	239,7968	260,5157

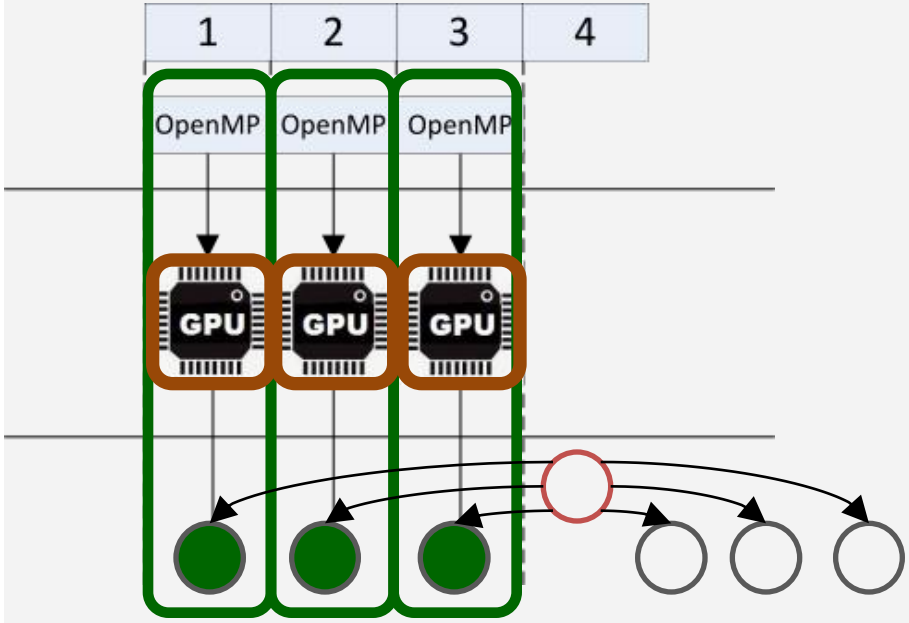
Experimentos en JUPITER **Disp**=85% **tFin**=10 **tFin2**=3

- Matrices pequeñas, mejor paralelismo OpenMP.
- Matrices grandes, mejor las combinaciones de OpenMP y threads MKL que mantiene ocupados todos los cores.

Paralelismo con GPU (1/3)

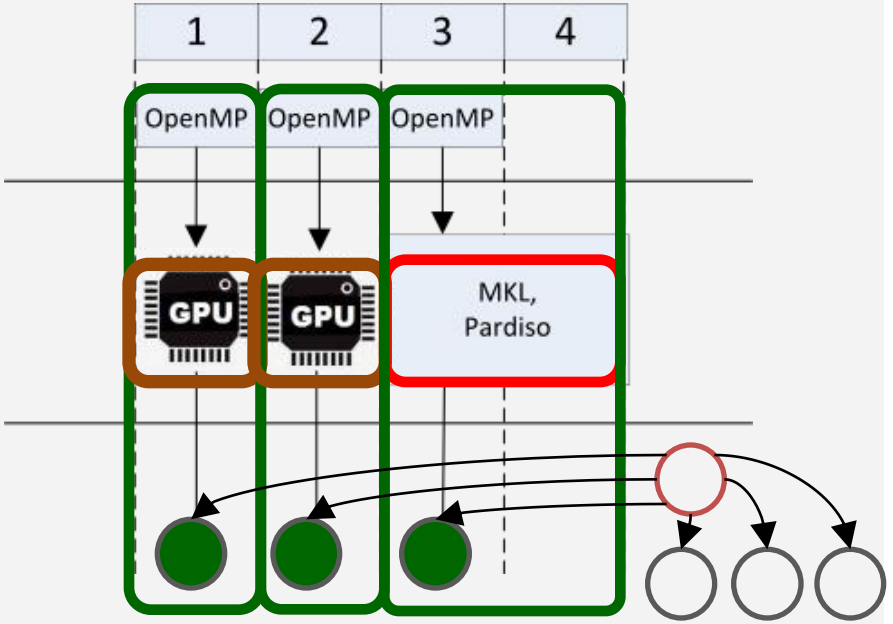
Caso 1: mismos threads que GPUs → MAGMA

- Lanzamos threads para resolver cada grupo Manivela-Barra.
- Cada thread selecciona una GPU, que es utilizada por **MAGMA** para realizar los cálculos matriciales, **combinando recursos de la CPU y la GPU.**



Caso 2: más threads que GPUs → MAGMA + MKL/PARDISO

- Lanzamos threads para resolver cada grupo Manivela-Barra.
- Cada thread selecciona una GPU disponible, que es utilizada por **MAGMA.**
- El resto de threads usarán **MKL** o **PARDISO**, en función del factor de dispersión.



Paralelismo con GPU (2/3)

Mejores tiempos de ejecución (segundos) con paralelismo en dos niveles y GPU

nEQTerminal	nEQManivela	MA27 6 x 1 CPU	PARDISO 6 x 2 CPU	MAGMA 6 CPU + GPU
12	15	**0,0009**	0,0234	4,1789
36	54	**0,0043**	0,0288	0,1242
60	90	**0,0089**	0,043	0,2362
400	400	**0,1559**	0,4945	1,262
1000	1000	**1,6476**	2,4988	2,2188
2000	2000	10,3387	7,759	**5,9685**
3000	3000	32,8752	16,732	**14,0435**
4000	4000	76,5881	31,6888	**28,2109**

Experimentos en JUPITER: 12 CORES + 6 GPU **Disp**=85% **tFin**=10 **tFin2**=3 **numGE**=6

- MA27, la más rápida con matrices de pequeñas dimensiones, con todos los threads asignados a OpenMP (MA27 no dispone de versión paralela).
- El uso de MAGMA, combinando los cálculos matriciales entre la CPU y las GPUs supera a PARDISO en el manejo de matrices grandes (cuando se usan todas la GPUs).

Paralelismo con GPU (3/3)

Problemas con más grupos estructurales **numGE**, comparación MKL, PARDISO, MAGMA

			th. OMP x th. MKL				PARDISO	MAGMA	MAGMA 12
numGE	**nEQTerminal**	**nEQManivela**	2 x 6	3 x 4	4 x 3	12 x 1	12 x 1	6	6 GPU + 6 PARDISO
16	12	15	0,0037	0,0029	0,0026	**0,0016**	0,0171	0,6000	0,3325
	36	54	0,017	0,0127	0,0149	**0,0052**	0,0305	0,2416	0,1407
	60	90	0,0508	0,0375	0,0335	**0,0114**	0,0501	0,6168	0,3657
	2000	2000	28,8608	24,9983	**24,6512**	34,685	**14,6420**	**12,5071**	**9,9266**
	3000	3000	77,3555	76,6553	**74,9888**	106,8146	**35,1489**	**28,8774**	**23,8957**
	4000	4000	**174,6793**	176,9465	175,3777	262,1968	**69,3354**	**57,8293**	**49,3556**
21	12	15	0,0099	0,0037	0,0029	**0,0019**	0,0219	0,7937	0,3816
	36	54	0,0351	0,0147	0,013	**0,0079**	0,0431	0,3345	0,1592
	60	90	0,0714	0,0436	0,0369	**0,0174**	0,0681	0,7619	0,4347
	2000	2000	33,7653	**27,4524**	35,1464	37,9592	**15,8987**	**15,6777**	**13,1037**
	3000	3000	108,381	**97,2876**	101,759	122,938	**38,4195**	**36,0209**	**32,5612**
	4000	4000	215,0848	**214,949**	238,1862	260,5157	**74,0894**	**72,7266**	**70,3011**

Experimentos en JUPITER **Disp**=85% **tFin**=10 **tFin2**=3

- Matrices pequeñas, mejor MKL con paralelismo OpenMP.
- Matrices grandes, el uso de MAGMA supera a PARDISO cuando se usan todas la GPUs.
- La ejecución con 12 threads añade mejoras adicionales

Autotuning (1/4)

¿Por qué?

- Varias configuraciones de paralelismo: ¿Cuál es la mejor?.
- Depende del hardware, del tamaño del problema y número de grupos paralelos.
- Se quieren seleccionar los mejores parámetros en tiempo de ejecución.

Metodología

- Selección de un conjunto de **escenarios de aprendizaje**.
- Selección del **conjunto de parámetros** paralelos válidos para el hardware.
- **Simulación** y almacenamiento de los **mejores tiempos** de ejecución y las configuraciones asociadas.
- En tiempo de ejecución, **seleccionar la mejor configuración** para un problema concreto, en base a la información de aprendizaje.

Autotuning: (2/4)

Aprendizaje: encontrar los mejores parámetros para cada tamaño de problema

Escenarios de aprendizaje

tfin	Tfin2	numGE	nEQTerminal	nEQManivela	nEQMatMul	neslabones	sparsity	symmetric
10	3	6	12	18	3	3	85	1
10	3	6	36	54	3	3	85	1
10	3	6	60	90	3	3	85	1
10	3	6	400	400	3	3	85	1
10	3	6	1000	1000	3	3	85	1
10	3	12	12	18	3	3	85	1
10	3	12	36	54	3	3	85	1
10	3	12	60	90	3	3	85	1
10	3	12	400	400	3	3	85	1
10	3	12	1000	1000	3	3	85	1

Parámetros posibles

	Rango	Lista de valores
OMP	{"1","12","1"}	
MKL		{"1"}
LOOP		{"0"}
LIBRARY		{"1","2","3"}
NESTED		{"2"}

Simulación

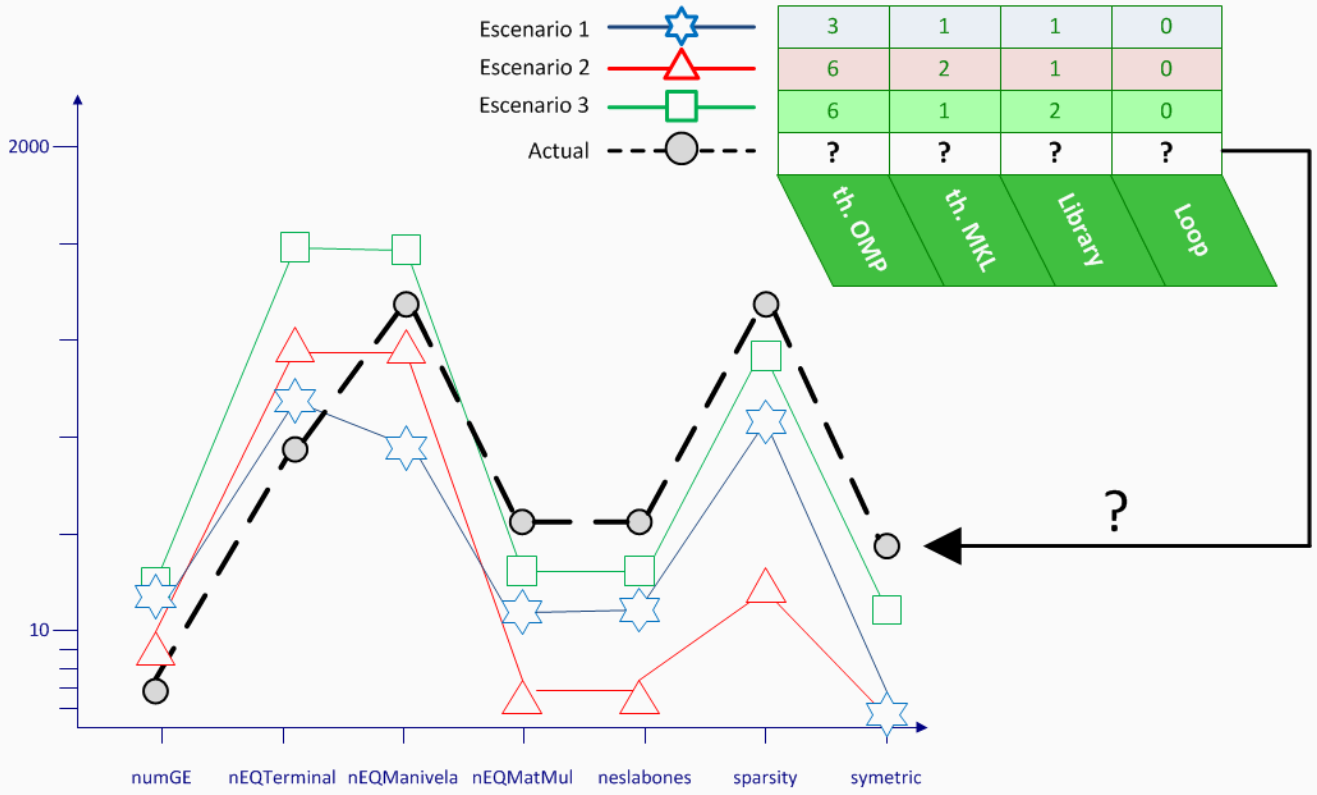
Par. óptimos

OMP	MKL	LOOP	LIBRAR	NESTED
6	1	0	3	2
8	1	0	3	2
8	1	0	3	2
7	1	0	3	2
6	1	0	3	2
12	1	0	3	2
12	1	0	3	2
12	1	0	3	2
12	1	0	3	2

Autotuning_data_base.csv

Autotuning: (3/4)

Ejecución: selección de un escenario similar al tamaño de problema actual



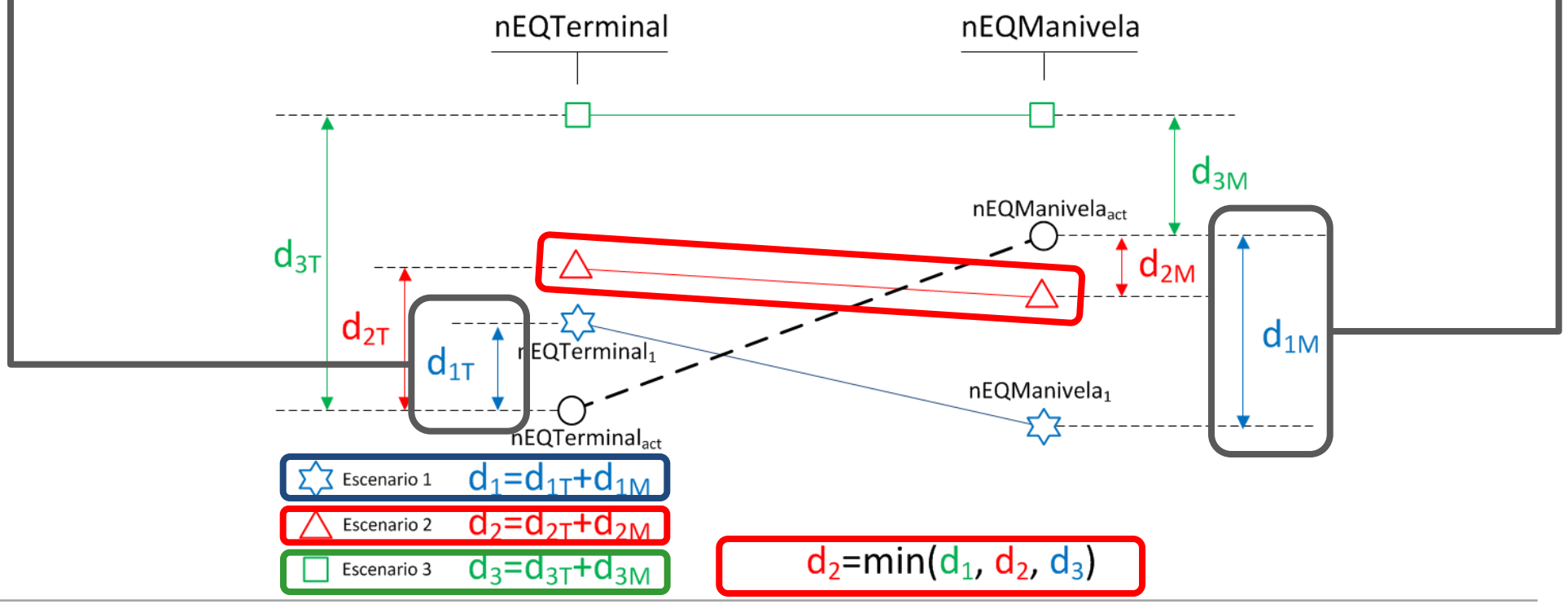
Método de ejemplo: **Distancias mínimas relativas**

Autotuning : (4/4)

Método de distancias mínimas relativas usando **nEQTerminal** y **nEQManivela**

$$d_{1T} = \left(\frac{\texttt{nEQTerminal}_1 - \texttt{nEQTerminal}_{act}}{\texttt{nEQTerminal}_1} \right)^2 \qquad d_{1M} = \left(\frac{\texttt{nEQManivela}_1 - \texttt{nEQManivela}_{act}}{\texttt{nEQManivela}_1} \right)^2$$

$$d_1 = d_{1T} + d_{1M}$$



Conclusiones

- En sistemas mecánicos con **grupos estructurales independientes**, la aplicación de paralelismo durante su resolución ofrece mejoras de rendimiento respecto al enfoque secuencial.
- En los tamaños de problema de la plataforma de Stewart, una librería especializada en matrices dispersas, como es **MA27**, es notablemente más eficaz, aún sin contar con implementación paralela.
- Con tamaños de problema medianos o grandes, librerías comerciales, como **MKL** para matrices densas, o **PARDISO** para matrices dispersas, consiguen reducir aún más los tiempos de ejecución aprovechando su paralelismo interno.
- El uso de **GPUs** a través de librerías especializadas como **MAGMA** se recomienda para matrices de dimensiones superiores a 2000 x 2000.
- Con paralelismo en dos niveles, los mejores rendimientos se obtienen cuando la **combinación de threads** asignados a OpenMP y a las librerías matriciales mantiene todos los cores disponibles en uso.
- Un proceso de **autotuning** permite seleccionar los parámetros paralelos óptimos tomando como base el conocimiento obtenido a través de experimentos preliminares.

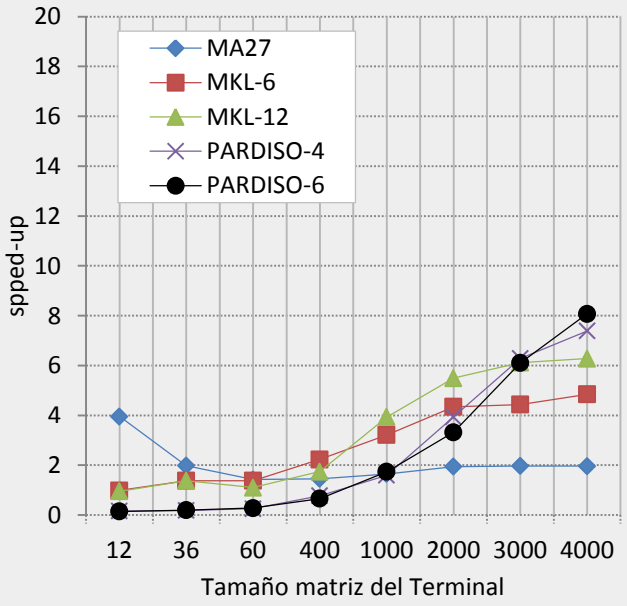
Trabajo futuro

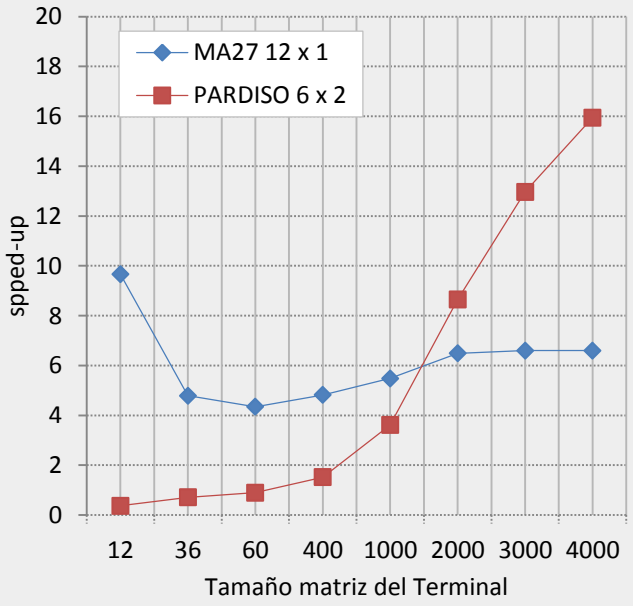
- El estudio de nuevas **estrategias de paralelismo** en función del balanceo entre el número de cores y el de grupos estructurales paralelizables (objetivo: mantener todos los recursos activos).
- Estudio de nuevas librerías que implementan **solvers dispersos** (p.e. MAGMA sparse).
- Salto a un esquema de **paso de mensajes**, para aprovechar los recursos de cómputo disponibles en un cluster.
- Autotuning:
  - ✓ Perfeccionar el **método de exploración** de parámetros óptimos (no exhaustivo).
  - ✓ Nuevos métodos de **selección de escenarios** en la base de datos de autotuning (y comparación con el de distancias mínimas).

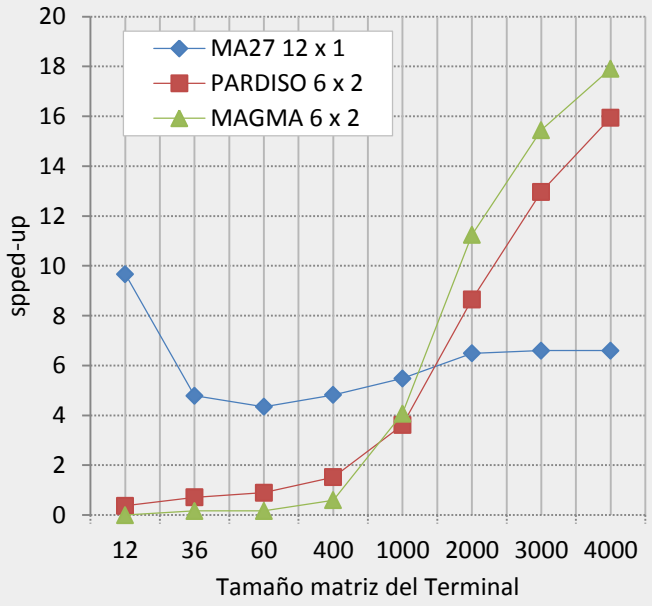
http://pixabay.com/es/pregunta-signo-de-interrogaci%C3%B3n-298479/

Speed-ups

Speed-up respecto a la versión secuencial con MKL (versión inicial del software de control)







Experimentos en JUPITER: 12 CORES + 6 GPU **Disp**=85% **tFin**=10 **tFin2**=3 **numGE**=6