Cálculo Simbólico
Práctica 3

Complejidad de las bases de Gröebner

José Ignacio Carmona Villegas
Juan Hernández García

Índice de contenido

Planteamiento del Procedimiento....3
Selección del paquete de cálculo....3
Sistema de Pruebas....3
Repeticiones....3
Gráficas y Resultados....5
Tests Cuadráticos....5
Grado fijo....5
Número de Polinomios Fijo....7
Número de variables fijo....9
Tests Lineales....11
Número de polinomios libre....11
Número de variables libre....11
Grado libre....12
Conclusiones Finales....13
Manual de instalación y uso....14

Planteamiento del Procedimiento

Selección del paquete de cálculo

El paquete de cálculo seleccionado para esta práctica será la biblioteca **Java Algebra System (JAS)**. Los motivos de esta selección son varios:

- Conocimiento previo de la misma. Se ha utilizado para la realización de la práctica 2.
- Compilador/Intérprete ya instalado y preparado.
- Sistema de repositorios ya integrado con Eclipse. Esto nos permite la programación colaborativa y eficiente.

Más adelante, en la sección de conclusiones finales, discutiremos con más detalle si ha sido o no una buena elección final y por qué motivos. En cualquier caso, se ha obtenido un buen conocimiento empírico de su comportamiento con respecto a las bases de Gröebner.

Sistema de Pruebas

Ya que esta práctica se centra en los resultados obtenidos, hemos planteado un diverso sistema de pruebas que pueda indicarnos el comportarmiento de **la implementación del algoritmo de bases de Gröebner en JAS**.

Se han programado pruebas, donde se fijan n paramétros de entrada al algoritmo (variables, número de polinomios y grado), quedando los restantes libres. Según el número de parámetros libres tenemos:

- **Test Cúbico:** este test no va a ser de gran utilidad, sobre todo por el tiempo que necesitaría a la hora de ser ejecutado. Tampoco la representación visual será útil. Se ha programado a modo teórico, y por si alguien quisiera ejecutarlo en una máquina potente.
- **Tests Cuadráticos:** tests donde sólo se fija el valor de una componente y las otros dos van cambiando su valor. Cuando se completa el test, se incrementa el valor de la componente fijada y se vuelve a lanzar para obtener una nueva tanda de resultados. Estos tests no son tan costosos en tiempo como el cúbico, pero aún así llevan mucho tiempo de ejecución. Serán útiles para ver comportamiento en las **zonas de bajos valores** y detectar posible **sinergia entre parámetros** de entrada.
- **Tests Lineales:** tests donde sólo un parámetro de entrada está libre. Son muy útiles para terminar de fijar las ideas alcanzadas con los tests cuadráticos y ver el **comportamiento aislado** de los parámetros de entrada y comprobar **comportamiento asintótico.**

Repeticiones

Cada valor obtenido en los tests, es la media de **50 repeticiones** realizadas para dicha configuración.

Para realizar estas repeticiones había dos posibilidades:

- Realizar un tests completo 50 veces.
- Realizar cada paso del tests 50 veces.

Finalmente se ha optado por la segunda opción. Nos hubiera gustado realizar la primera, ya que representaría de forma más real el comportamiento medio de la función, pero en este puntos nos hemos comenzado a encontrar con los problemas de la implementación del cálculo de bases de Gröebner en JAS: **problemas de liberación de memoria.**

La máquina Java, al contrario que otros lenguajes como por ejemplo C, libera memoria cuando considera oportuno. Sacrifica una eficiente liberación de memoria, por librar al usuario de hacerlo él mismo. Normalmente esto no supone demasiado, pero en esta implementación de JAS parece que sí.

Llamar dos veces seguidas a la función con los mismos parámetros da resultados más realistas y uniformes que cambiar parámetros de entrada (incluso que cambios bruscos en los parámetros de entrada). La conclusión a la que se llega, es que realiza una serie de inicializaciones en las clases internas que son bastante ineficientes.

Tras muchas repeticiones y cambios de estructura en la práctica se ha llegado a esta conclusión.

Por otro lado, 50 repeticiones, era una número que ponía en consenso los resultados obtenidos y el tiempo necesario para ejecutar los tests.

Gráficas y Resultados

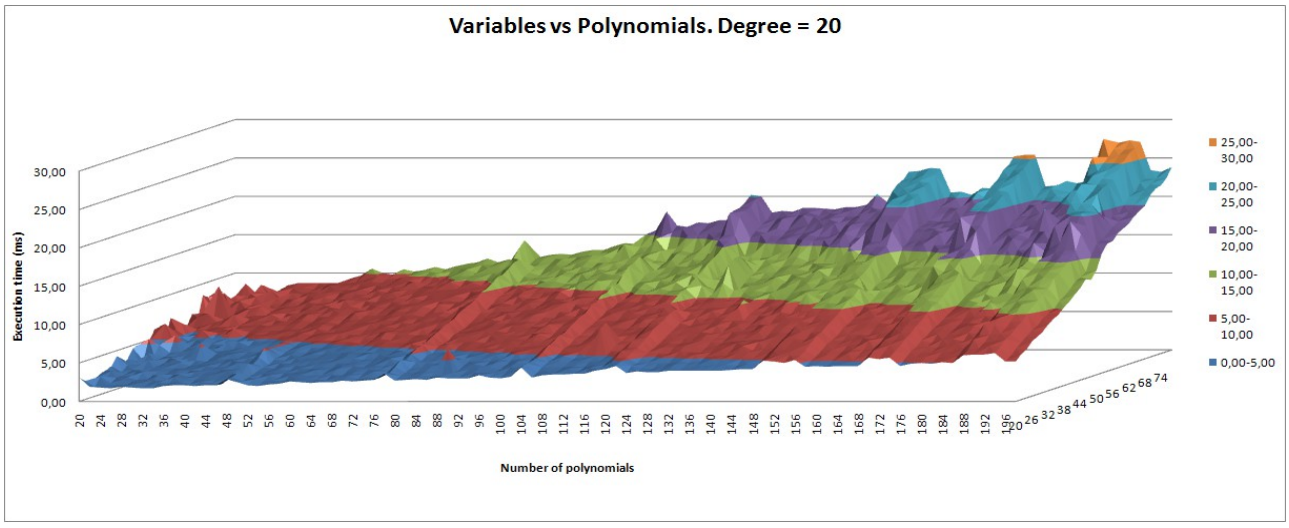
Como ya se ha dicho antes, se han programado tres tipos de tests: cúbicos, cuadráticos y lineales. No se ha llegado a completar una ejecución completa de tests cúbico, pero está a su disposición por si decide ejecutarlo en una máquina lo suficiente potente y con suficiente memoria (en este caso tendrá que indicar manualmente a la máquina virtual de java cuanta memoria máxima le está permitido coger, ya que inicialmente el parámetro está establecido a 512MB).

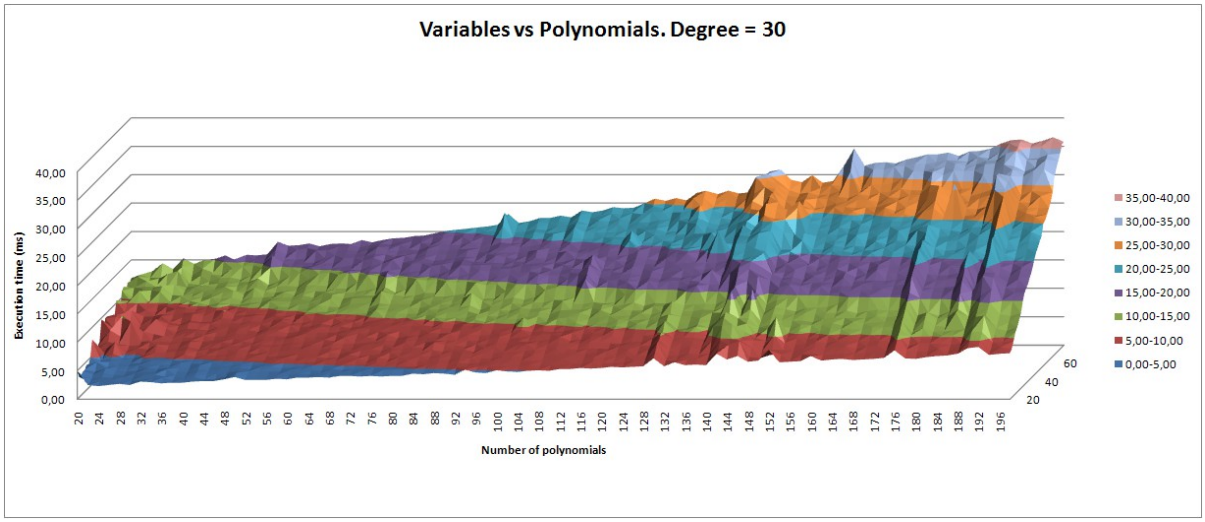
Por cada test cuadrático, se han tomado tres valores para la componente fijada, los más parecidos entre sí posibles, y que al mismo tiempo permitan completar la ejecución a pesar de los problemas de implementación de Gröebner en JAS. Es decir, se han obtenido tres gráficas por cada test cuadrático.

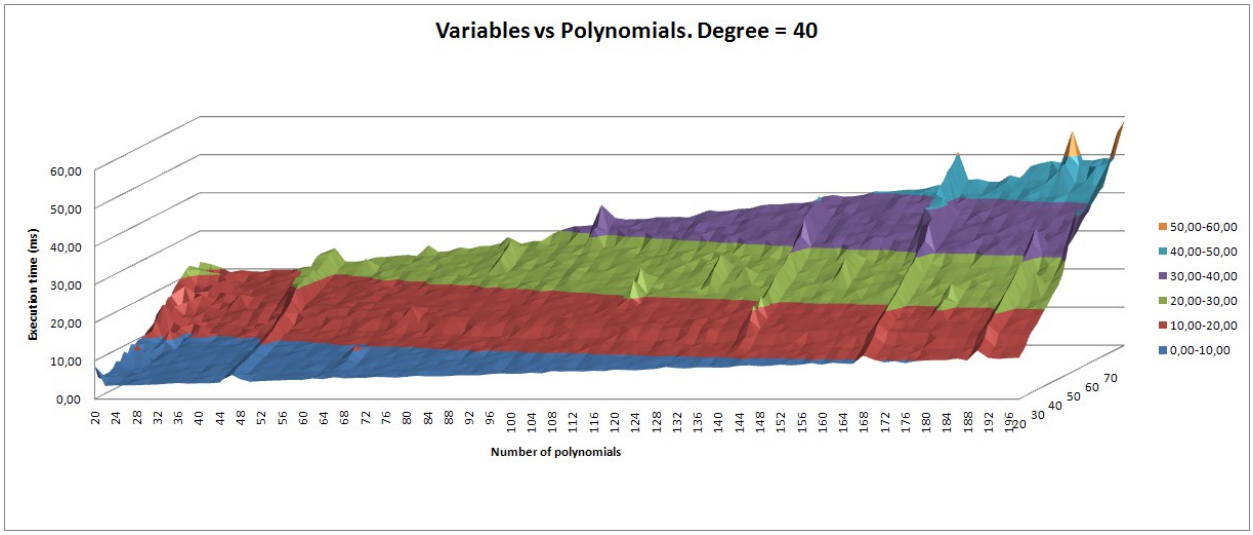
Tests Cuadráticos

Recordar, que estas primeras gráficas nos sirven para hacernos una idea del comportamiento del algoritmo con valores bajos, y detectar posible sinergia entre componentes. No deben ser utilizadas para asegurar un comportamiento global.

Grado fijo





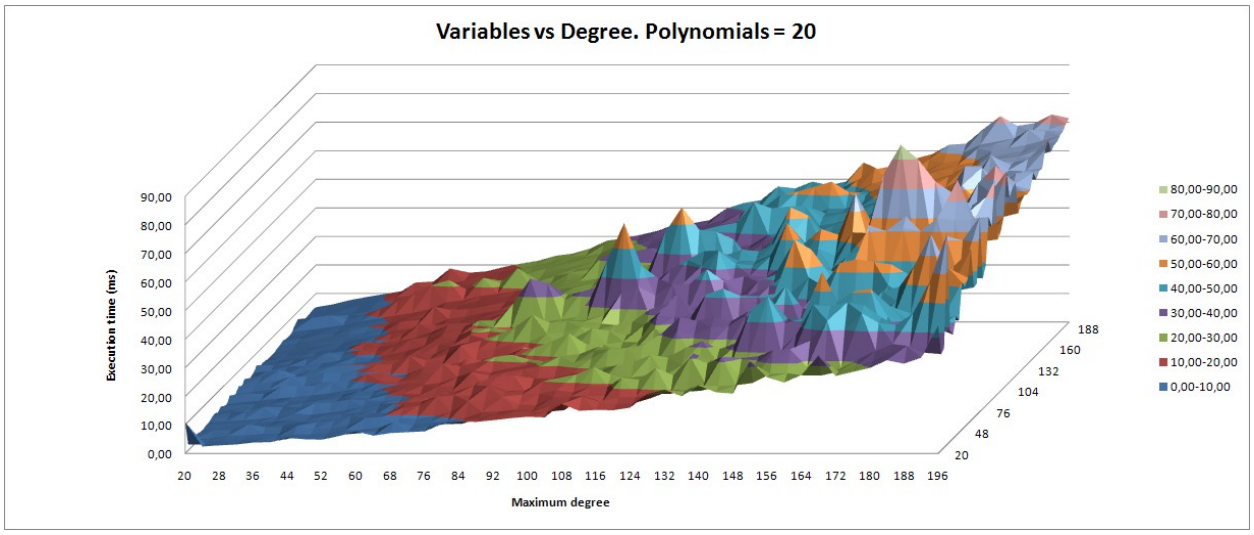


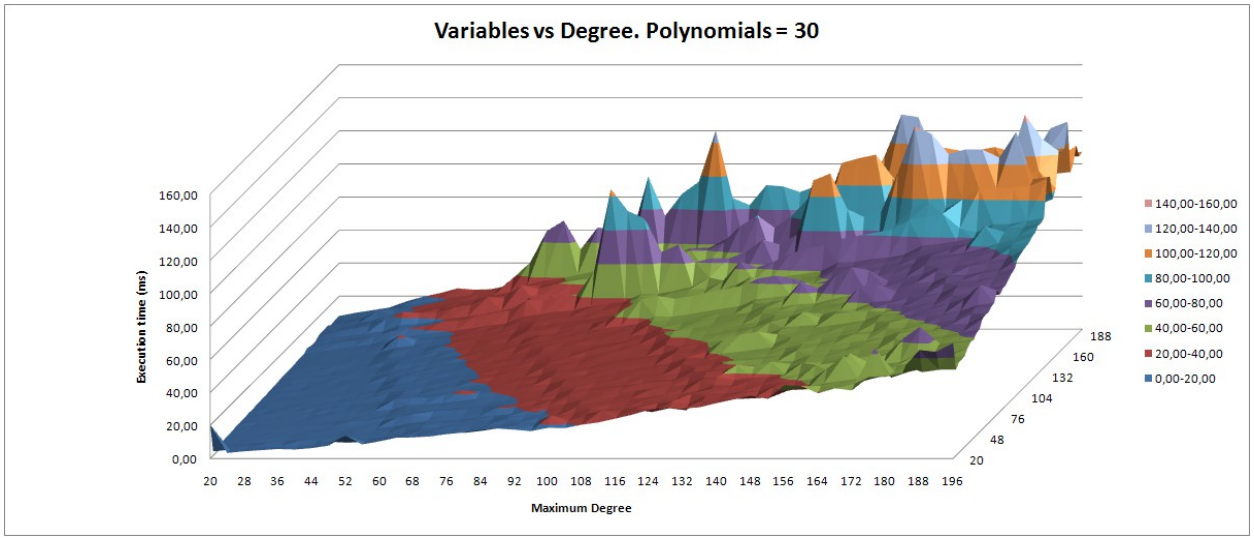
Podemos ver comportamientos similares en las tres gráficas.

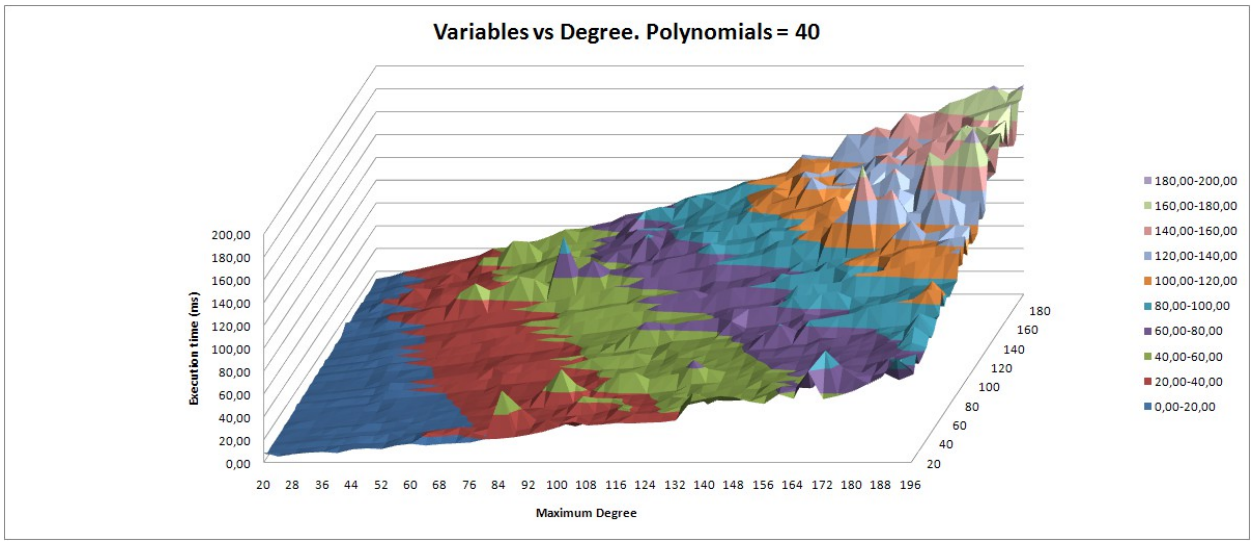
- Conjunto de **picos en las configuraciones iniciales**: esto se debe a las inicializaciones necesarias de las clases internas. Veremos que es un comportamiento estándar para todos los tests.
- Conjunto de **líneas alzadas intermedias**. Esto también va a darse en todos los tests realizados, para todas las configuraciones. Representa estados del procesador, cambios de tarea… en resumen, son consecuencia directa de la máquina utilizada para la obtención de los tiempos.
- La tendencia de la superficie es parecida para los tres valores fijados. Parece que para estos valores bajos los polinomios influyen menos que el número de variables.
- Se puede ver, que al aumentar el grado de los polinomios, la inclinación de las superficies es mayor, y los tiempos más altos.

En este grupo de gráficas, se ha fijado el grado del polinomio a 20, 30, 40. Como ya se ha dicho antes, a la máquina virtual de java no se le puede obligar a la liberación de memoria, como mucho sugerir. Esto hace que al realizar un tests cuadrático, con 50 repeticiones sobre cada valor, se vaya quedando mucha memoria colgada por el camino. Imposibilita la obtención de un número mayor de gráficas por test, o tratar con valores mayores para los parámetros fijos.

Número de Polinomios Fijo



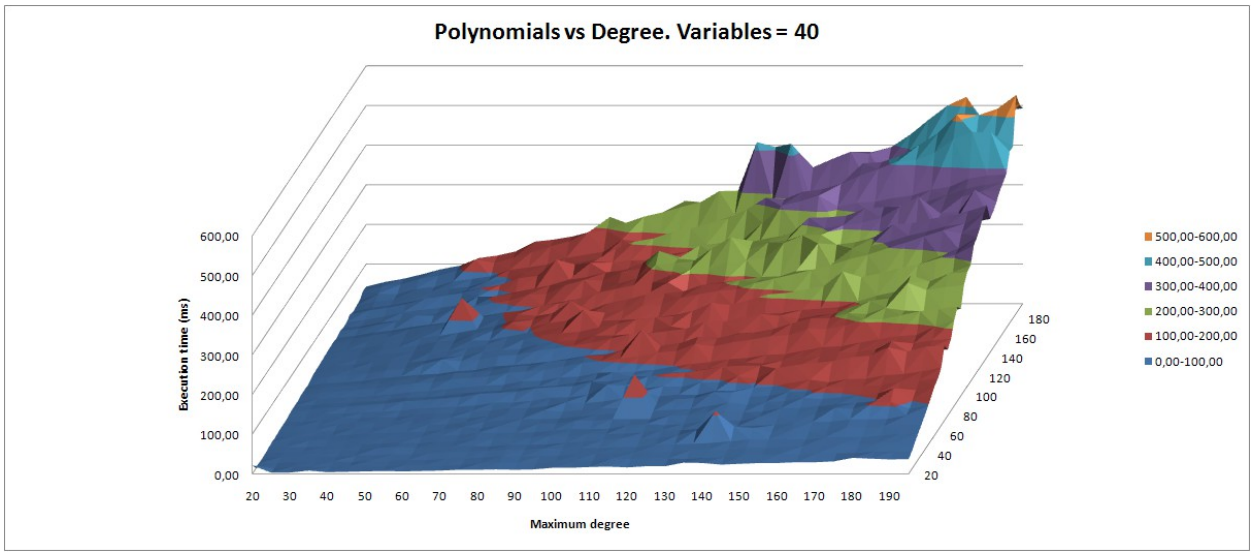


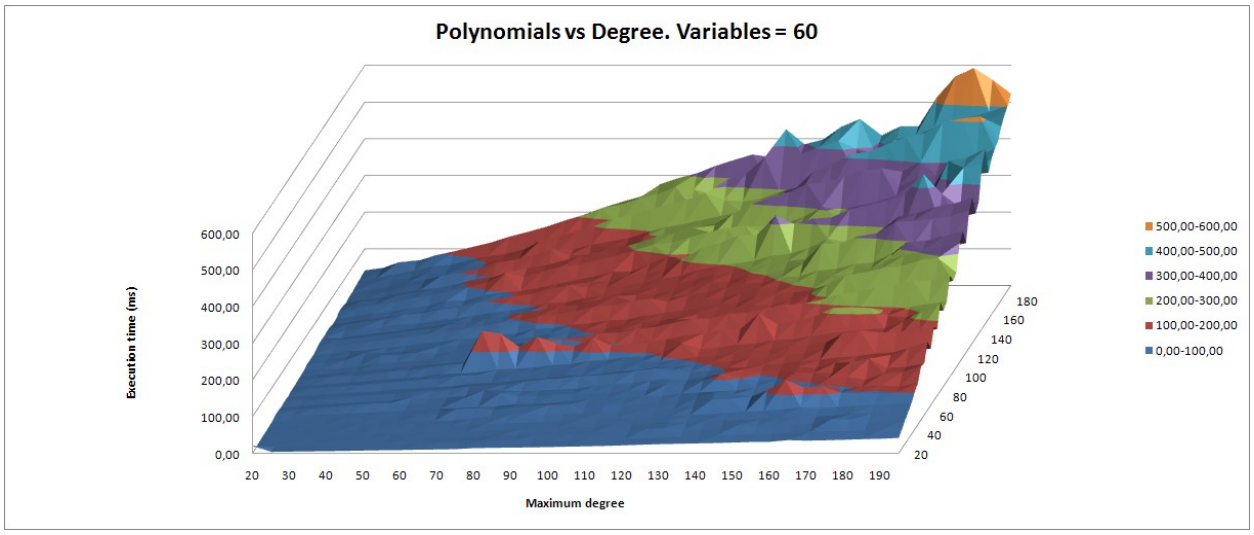


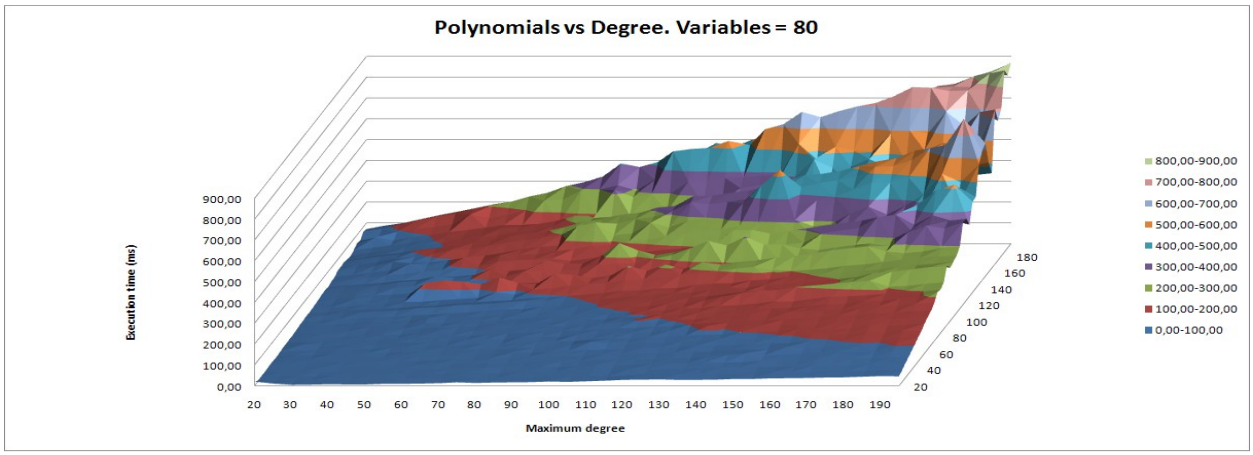
Podemos ver comportamientos similares en las tres gráficas.

- Conjunto de **picos en las configuraciones iniciales**: Siguen ocurriendo, pero se aprecian mucho menos porque nos movemos en una escala de tiempo mayor.
- Conjunto de **“líneas montañosas”**. Representa estados del procesador, cambios de tarea… en resumen, son consecuencia directa de la máquina utilizada para la obtención de los tiempos. Esta vez, el número de irregularidades es mayor, y conforme van aumentando el número de variables y grado, cada vez más frecuentes. Posiblemente, esto se deba a un proceso activo interfiriendo con la máquina. No obstante, se puede ver bien el comportamiento del test.
- La **tendencia de la superficie** es parecida para los tres valores fijados. Parece que para estos valores bajos el grado influye menos que el número de variables.
- Se puede ver, que al aumentar el número de polinomios, la inclinación de las superficies es mayor, y los tiempos más altos. Con respecto a la tanda anterior de gráficas, se puede observar que la magnitud de crecimiento es mayor. Es decir, el cambio en el parámetro fijo actual, influye más en conjunción al crecimiento de los otros dos.

Número de variables fijo







Los comportamientos observados son:

- Conjunto de **picos en las configuraciones iniciales**: Siguen ocurriendo, pero no se aprecian por la escala de tiempos en la que nos movemos.
- Conjunto de **"líneas montañosas"**. Representa estados del procesador, cambios de tarea… en resumen, son consecuencia directa de la máquina utilizada para la obtención de los tiempos. Aunque siguen apareciendo irregularidades, esta vez son mucho menores. Se puede observar una superficie muy uniforme.
- La **tendencia de la superficie** es parecida para los tres valores fijados. Esta vez, polinomios y grado, influyen en el crecimiento de una forma muy similar para valores bajos de los parámetros.
- Si tenemos en consideración el parámetro fijados a 40, al incrementar conjuntamente el grado y los polinomios nos lleva a los tiempos más altos conseguidos hasta el momento.

De los nueve test cuadráticos realizados podemos concluir:

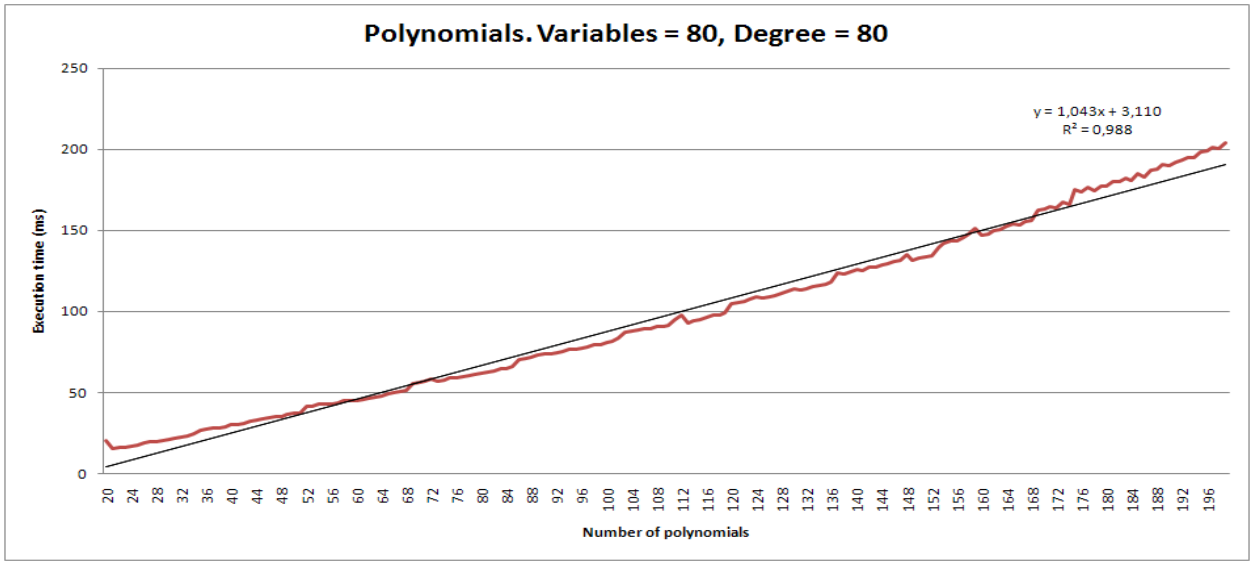
- El aumento de grado y polinomios de forma conjunta nos lleva a los tiempos más elevados. (comparativas con parámetro fijado a 40).
- En todas las gráficas hay un coste de inicialización.
- Parece que para valores bajos el número de variables en los más influyente en el cálculo del tiempo.

Tests Lineales

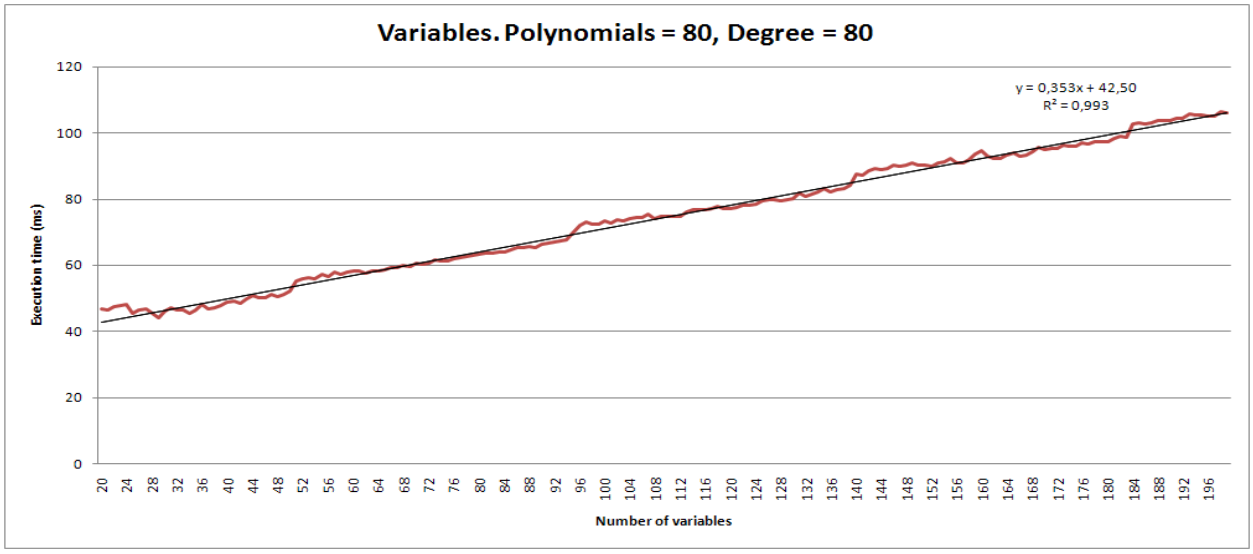
Los test lineales (fijando dos parámetros y dejando una fija), nos van a servir para comprobar el comportamiento en extremos, y para comprobar el aumento de tiempos conforme aumenta de forma aislada uno de los parámetros.

Los resultados obtenidos son los siguientes:

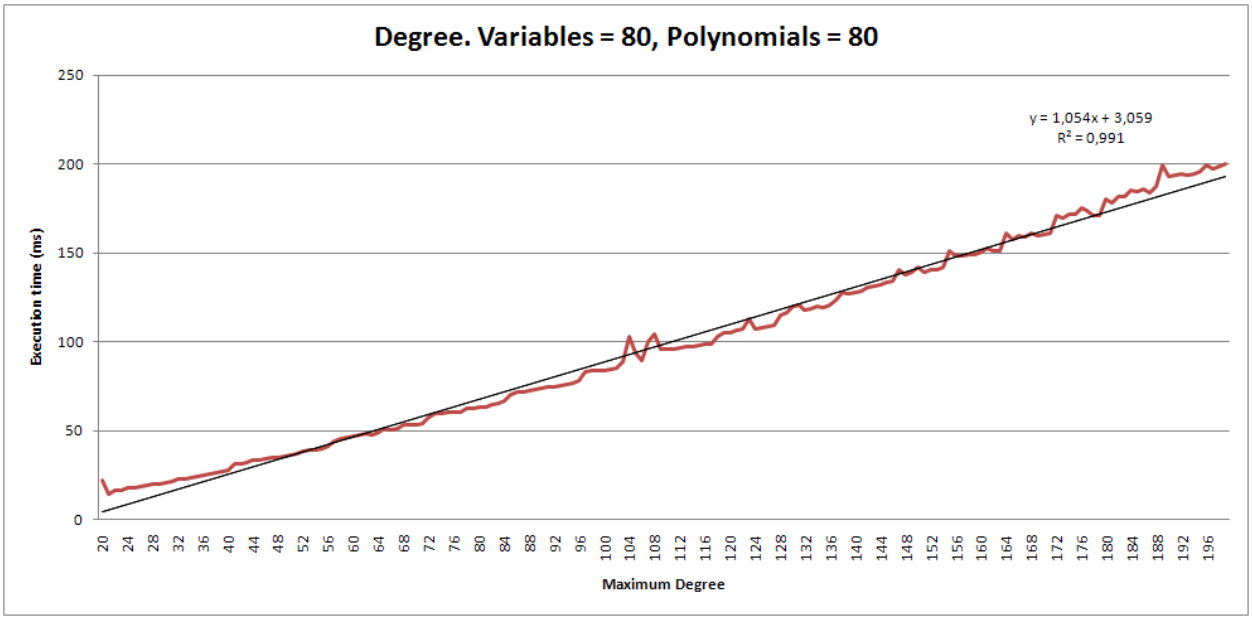
Número de polinomios libre



Número de variables libre



Grado libre



Para todas los casos se ha realizado un ajuste (linea negra) sobre los valores reales obtenidos (linea roja). Nos permite ver que para todos los casos el crecimiento es lineal y se puede obtener una función que nos lo defina (indicada en la parte superior derecha de las gráficas).

Número de polinomios: 1,043x + 3,1 **Número de variables:** 0,353x + 42,50 **Grado máximo:** 1,054x + 3,59

Podemos ver, gracias a los términos independientes, que **inicialmente**, el número de **variables es mucho más influyente** que el número de polinomios o el grado de los mismos. Este dato corrobora los obtenidos con los tests cuadráticos.

Sin embargo, **asintóticamente el número de polinomios y grado tienen mucho más peso**. Además, el crecimiento que provocan es similar (para valores fijos del resto de parámetros).

En estas gráficas también se puede observar el **pequeño sobrecoste inicial** que hemos ido comentando a lo largo de todo el documento.

Los picos y los valles son provocados por la dedicación del procesador a otras tareas.

Conclusiones Finales

Se ha realizado un estudio intensivo del **comportamiento y problemas** de las implementación del cálculo de las bases de Gröebner en JAS.

No causa sorpresa que el número de polinomios o grado máximo provoquen mayor crecimiento. En el cálculo formal de bases de Gröebner, a mayor número de polinomios iniciales hay un crecimiento importante del número de operaciones a realizar entre pares de ellos.

El Grado también provoca un incremento del número de operaciones en la máquina. Se están generando aleatoriamente los polinomios sin coeficientes que sean nulos. Eso implica que a mayor grado, el número de elementos del polinomio es siempre mayor y, por lo tanto, el número de operaciones aritméticas a realizar con ellos crece.

En el algoritmo, esto se nota en el cálculo de los S-Polinomios, que requiere calcular todas las combinaciones entre los polinomios de entrada, por lo que el número de polinomios es muy determinante; e igualmente con el grado de los polinomios, pues dicho grado (y también el número de polinomios) fuerza un mayor número de monomios con los que operar. Sin embargo, el número de variables sólo afecta a la longitud de los monomios, pero no a su número. Se ha escogido esta medida en base a monomios por ser la partícula mínima de un polinomio. Además, en el algoritmo, también se realiza la división de cada S-polinomio contra el resto (de los originales), y de nuevo vuelven a ser definitorios el número de polinomios y su grado máximo. El hecho de que el número de variables sea más influyente para valores bajos, se debe probablemente a un defecto de implementación de la biblioteca usada.

Hemos tenido que lidiar con problemas importantes de la implementación del cálculo de las bases de Gröebner en JAS, de lo que se ha obtenido un conocimiento importante de la misma. Tras todo el trabajo empírico realizado con ella, podemos sacar las siguientes conclusiones:

- **Mala gestión de memoria:** esto es debido a una mala implementación de la función encargada del cálculo de las bases. No tenemos acceso al código original, pero debe tener cuellos de botella importantes.
- **No orientada a ejecuciones masivas:** a causa de lo anterior, la biblioteca no parece estar orientada a repeticiones múltiples de la función. El simple hecho de meter en un bucle dicha función y llamarla muchas veces con una alta alteración de parámetros puede provocar inestabilidad en los resultados.
- **Error en función análoga:** en la biblioteca la función sobre la que se centra esta práctica tiene varias sobrecargas. La sobrecarga más simple tiene una alta tasa de fallo y logra dejar colgada la máquina con facilidad y sin motivo. Esto nos ha provocado un importante retraso.

A causa del desconocimiento general del paquete, y decente funcionamiento en la práctica anterior, se ha elegido JAS. Ha sido una mala elección, pero ha provocado que aprendamos

muchas cosas. En un futuro, se optaría por aprender a utilizar una biblioteca mejor cualificada para el cálculo simbólico.

Manual de instalación y uso

Prerrequisitos:

- Tener instalado java jre7: http://www.oracle.com/technetwork/java/javase/downloads/java-se-jre-7-download-432155.html
- Descargar IDE eclipse Java: http://www.eclipse.org/downloads/moreinfo/java.php
- Asegurarse de que el path de donde toma el jre es correcto.
- Asegurarse de que las bibliotecas includas en "lib" se encuentran asociadas al buildpath del proyecto. En caso de que no, puede configurarse desde las propiedades el proyecto.

Datos de entrada:

- No hay que suministrar datos de entrada. Los test se lanzan y se autogeneran polinomios a partir de los parámetros establecidos.

Datos de salida:

- Las salidas se mostrarán por consola y se almacenarán en ficheros llamados **test1.txt, test2, …, test7.txt**.

Proyecto: puede abrirse como un proyecto convencional. Adicionalmente puede descargarse desde el repositorio directamente.
https://github.com/Zalioth/SC_Gröebner

Ejecución

- Cuando se tengan los datos de entrada correctamente ubicados, bastará con colocarse encima del fichero Main.class, pulsar botón derecho/Run As/ Java Application.
- En ese momento en consola aparecerá un menú con la siguiente estructura:

  0 -> Exit.
  1 -> Test 1. Cubic test, v-p-d.
  2 -> Test 2. Quadratic test, v-p
  3 -> Test 3. Quadratic test, v-d
  4 -> Test 4. Quadratic test, p-d
  5 -> Test 5. Linear test, p
  6 -> Test 6. Linear test, v
  7 -> Test 7. Linear test, d
  Where:
  'v' stands for number of variables,
  'p' stands for number of polynomials
  'd' stands for maximum degree of each polynomial.
  Choose an option:

- Por consola podrá seleccionar el test a realizar.
- Se comenzarán a generar los resultados.