Laboratorio de Métodos Numéricos - Primer cuatrimestre 2011
Trabajo Práctico Número 2: La Guerra Lineal, Episodio 5 ...

Introducción

Corría el año 4957 de la era de paz iniciada luego de la expulsión de los mutantes invasores en el sistema solar de la estrella HAL-9000 cuando los sistemas de defensa alertaron a la comandancia por la intrusión de naves espías provenientes de lejanas galaxias dominadas por especies hostiles. Luego de estas primeras incursiones comenzaron los devastadores ataques alienígenas. Las fuerzas de dicho sistema solar se enfrentaban al ataque de un enemigo muy superior, que amenazaba con conquistar a todos los planetas del sistema y comenzar así una avanzada sobre la República Galáctica.

Los sistemas de defensa convencionales pronto fueron insuficientes para resistir el ataque y, además, las inspecciones telemétricas dieron cuenta de que, fuera de los límites del sistema solar, el enemigo estaba ensamblando un Cañón Warp, el arma más devastadora jamás construida por seres inteligentes. Los registros de la República solamente consignaban cuatro ataques previos por medio de este tipo de armas: en el primer ataque al sistema Z-80[1], en la batalla de la estrella 80286[2], en las guerras de los clones de PCs[3], y en el segundo ataque al sistema Z-80[4]. La única resistencia posible contra un Cañón Warp es utilizar otro Cañón Warp, con lo cual se hizo un llamado desesperado a las fuerzas de línea de la República para responder a este ataque. En menos de dos semanas el Cañón Warp del Ejército de la República estaba preparado para dar batalla, y el combate decisivo estaba por comenzar...

La Guerra Lineal

La batalla entre dos Cañones Warp no es un combate entre bandoleros espaciales, sino una caballerosa justa algebraica entre dos naves que respetan ciertas reglas. Para no ser directamente visible, cada nave se sitúa en un punto del hiperespacio de n dimensiones. Llamemos $x \in \mathbb{R}^n$ e $y \in \mathbb{R}^n$ a los puntos donde se ubican la primera y la segunda nave, respectivamente.

A intervalos regulares, cada nave realiza un disparo de su Cañón Warp con el objetivo de desintegrar al enemigo. Debido a que nos encontramos en el hiperespacio, la única forma de dirigir el disparo hacia la posición de la nave enemiga es mediante una transformación del espacio–tiempo, llamada *transformación warp*. Esta transformación modifica el espacio–tiempo de manera tal que la bomba lanzada por la nave en la posición x y sometida a la transformación warp A "explotará" en la posición $d = Ax$ luego del disparo. Una transformación warp está dada, entonces, por una matriz *inversible* $A \in \mathbb{R}^{n \times n}$. La nave enemiga es destruida si y solo si $\|d - y\|_2 \leq 1$.

Ahora bien, supongamos que la nave y no es destruida. Esta nave puede ver el punto d donde se produjo la explosión y, analizando las distorsiones del espacio–tiempo sobre el fondo de estrellas fijas, también puede deducir la transformación warp A que usó el enemigo. Entonces, al menos en teoría, puede resolver el sistema $Ax = d$ para averiguar la posición x del enemigo y eliminarlo en su siguiente disparo. Aquí entra en juego la habilidad de cada contendiente para evitar ser fácilmente descubierto luego de cada disparo propio, y para descubrir la ubicación del rival luego de cada disparo enemigo.

La Guerra Lineal consiste en una sucesión de disparos simultáneos de cada contendiente, hasta que uno de ellos es destruido (o ambos son destruidos al mismo tiempo). Luego de cada disparo, cada nave recibe la información telemétrica de la posición del disparo enemigo y la transformación warp que usó el adversario, con lo cual podrá ajustar su siguiente disparo para acercarse más a la nave enemiga.

Detalles y requerimientos

El objetivo del trabajo práctico es implementar un Control de Cañón Warp: un programa que desarrolle automáticamente una batalla de la Guerra Lineal contra el programa de otro grupo, de acuerdo con

[1] La Guerra Lineal, Episodio 1: TP1 del segundo cuatrimestre de 2000.
[2] La Guerra Lineal, Episodio 2: TP2 del primer cuatrimestre de 2001.
[3] La Guerra Lineal, Episodio 3: TP2 del primer cuatrimestre de 2005.
[4] La Guerra Lineal, Episodio 4: TP2 del segundo cuatrimestre de 2008.

las siguientes especificaciones.

Cada ejecución del programa corresponde a un disparo del cañón. Cada programa se ejecuta por línea de comandos y toma como parámetros los nombres de varios archivos. Algunos archivos contienen la posición de la nave y los disparos enemigos, y otro archivo contendrá la salida del programa.

Una vez leído el o los archivos de entrada el programa debe seleccionar una transformación warp $A \in \mathbb{R}^{n \times n}$ y realizar un disparo $d = Ax$. A partir del segundo turno el programa contará con información que le permitirá inferir la posición del oponente y podrá elegir una transformación de manera que el disparo se acerque a la nave enemiga. Una vez decidida la transformación y calculado el disparo el programa debe informar los mismos en el archivo de salida.

Luego de escribir este archivo, el programa debe detener su ejecución y será vuelto a ejecutar cuando llegue el momento de realizar un nuevo disparo (en el siguiente turno). Este proceso continúa hasta que alguno de los dos contendientes es eliminado. Cualquier información que sea necesario registrar entre disparos se debe almacenar en archivos auxiliares (en el "directorio actual"), a criterio del grupo.

Los programas serán invocados en el primer turno con la siguiente línea de comandos:

```
<programa> <posicion> <salida>
```

y del segundo turno en adelante, una vez que se cuenta con los datos que generó el programa rival, serán invocados con la siguiente línea de comandos:

```
<programa> <posicion> <salida> <ultimo> <anteriores>
```

Por ejemplo:

```
lineal.exe datos.txt disparo.txt d001.txt otros.txt
```

De acuerdo a los siguientes detalles:

programa Es el programa de cada grupo.

posicion Es un archivo con el siguiente formato:

- La primera línea contiene un cero[5].
- La segunda línea contiene el valor de n, especificando la cantidad de dimensiones del espacio en el que se juega.
- La tercera línea contiene la posición del jugador, especificada por n valores separados por espacios.

Por ejemplo, el siguiente es un archivo válido de entrada:

```
0
2
-9.5101358720664081000e+001 3.1796508444471566000e+001
```

salida Es el archivo que debe generar el programa, conteniendo la transformación warp elegida y el disparo resultante. Su formato debe ser el siguiente:

- La primera línea debe contener el número del turno en que se efectuó el disparo (el primer turno es el número 1).
- La segunda línea debe contener la dimensión n (este valor ya es conocido, pero se incluye en el archivo para verificar que no haya inconsistencias).
- La tercera línea debe contener las n coordenadas del vector d, separadas por espacios.

[5]Por razones históricas.

- Las siguientes n líneas contienen los coeficientes de la matriz A, ubicando en cada línea una fila completa de la matriz (con los coeficientes separados por espacios).

ultimo Contiene el disparo generado por el programa oponente en el turno anterior, con el formato descripto anteriormente.

anteriores Es una concatenación de todos los archivos de disparos del oponente, salvo el último. En el segundo turno es un archivo vacío. En el tercer turno contiene el disparo del turno 1, en el cuarto turno contiene el disparo del turno 1 seguido del disparo del turno 2, etc.

En todo momento el combate estará supervisado por un programa que oficiará de árbitro entre los dos programas en juego (por razones históricas, este árbitro será llamado La Fuerza), que se encarga de ejecutar ambos programas y de pasarles como parámetros los archivos que correspondan en cada caso.

Para asegurar el desarrollo normal de la competencia, el programa jugador se debe implementar bajo el sistema operativo Windows, de manera que sea posible su ejecución en las máquinas del Laboratorio 6.

El combate debe desarrollarse en aritmética de punto flotante de doble precisión (8 bytes, correspondiente al tipo de datos `double`). La Fuerza tomará los datos como `doubles`, con lo cual es importante tomar los recaudos necesarios para la lectura y escritura de los archivos. Todos los archivos generados, tanto por La Fuerza como por los contrincantes, serán archivos ASCII conteniendo sólo números, espacios y fin de línea. Todos los números de punto flotante serán escritos usando 20 dígitos y notación científica[6], para evitar errores de representación.

Comentarios destacables

Es importante que el programa implementado cuente con estrategias inteligentes de defensa, que no le permitan al enemigo deducir fácilmente nuestra posición. Por ejemplo, no sería muy inteligente utilizar como transformación warp una matriz diagonal, dado que en este caso el enemigo puede encontrar nuestra posición con mucha precisión. El informe debe contener todas las alternativas que el grupo haya considerado con relación a este punto, junto con una discusión que justifique la opción finalmente adoptada. Además, el informe debe justificar que la matriz generada en cada turno es inversible. También es importante que el programa esté preparado para actuar ante las posibles estrategias de defensa del enemigo. El programa debe contemplar que el enemigo puede estar generando transformaciones warp que dificulten nuestro análisis de su posición, y debe implementar algún mecanismo que intente manejar esta situación. No se aceptarán transformaciones warp que no sean inversibles, disparos que están mal calculados, o el uso indebido de espadas láser entre los contrincantes. Recuerden que la Guerra Lineal es un combate entre caballeros.

Entregas parciales

6 de mayo: Implementación del (o de los) método(s) de resolución de sistemas de ecuaciones.
13 de mayo: Implementación completa y casos de prueba.

Entrega Final

Formato Electrónico: 19 de mayo de 2011, hasta las 23:59 hs, a la dirección:
metnum.lab2011@gmail.com
Formato físico: 20 de mayo de 2011, de 17 a 21 hs.
Batalla lineal entre grupos: 20 de mayo de 2011, de 20 a 21 hs.

[6] Una opción es `precision(20)` y `setf(std::ios_base::scientific,std::ios_base::floatfield)` de `std::ostream`. Ejemplo: -1.64741740540787990000e+002 .