Tarea 1

Diseño y Programación Orientados a Objetos

"Masas y resortes como objetos de software"

Carlos Apablaza 2521068-9 Rodrigo Mujica 2521003-4

Esta tarea consiste de la simulación, por medio de java, del sistema físico de bloques unidos a resortes en un plano de coordenadas de 2 dimensiones.

Para esto, se ha tomado un modelo ideal de los elementos físicos, o sea, se consideran bloques de masa centralizada en un punto, no existe roce, etc.

El programa se basa en 6 clases:

- PhysicElement que es heredada por Spring y Block.
- Vector2D.
- Simulator.
- Experiment (contiene el método main).

Básicamente al invocar el método main de la clase Experiment, este crea objetos tanto de la clase Spring como Block, dándoles sus atributos físicos correspondientes, definidos manualmente dentro del código. Es necesario también definir las uniones de estos elementos vía código, para poder tener finalmente un sistema físico completo.

Luego crea un objeto de la clase Simulador para que simule el paso del tiempo y las fuerzas que actuarían sobre los elementos previamente creados. Para esto, pide por consola algunos parámetros temporales al usuario.

Una vez que la simulación (método de Simulator) se ha iniciado, este se encarga de ir renovando la posición de cada Bloque definido, por medio del método getForce(), que calcula la fuerza aplicada por los resortes a cada Bloque según la posición de sus extremos, obtenida a partir de la ley de Hooke.

Teniendo la fuerza que aplica cada resorte, se realiza una suma vectorial de las mismas para cada bloque, obteniendo así su aceleración "instantánea" y permitiendo deducir matemáticamente tanto la posición como la velocidad un "deltat" más tarde.

Además de calcular los atributos de cada elemento (cada "deltat" segundos), también el simulador se encarga de invocar (cada "samplet" segundos) al método printSystemState, que nos entregara la información cartesiana de cada uno de los elementos definidos por pantalla o a un archivo (según haya sido la invocación del programa).

Se realizaron simulaciones de 2 sistemas, en ambos casos, se utilizaron los valores:
delta t = 0.001
tiempo de muestreo = 0.1

De acuerdo a lo requerido, los gráficos obtenidos son los siguientes:

a) 1 Dimensión:

Este es el caso más simple, y consta de un bloque 1 de masa infinita, un resorte, de largo en reposo de 0.7 y constante de elasticidad de 1 y un bloque 2 de masa 1. Los bloques inicialmente tienen velocidad 0.

Las posiciones iniciales de los bloques son:
Bloque 1 = (0,0)
Bloque 2 = (1,0)

Estos valores fueron elegidos para recrear el evento simulado en los archivos utilizados como base para esta tarea (solución del problema en R).

Figura 1: Esquemático unidimensional.

Al graficar los datos, el resultado obtenido es el mismo que en el código sin modificaciones:

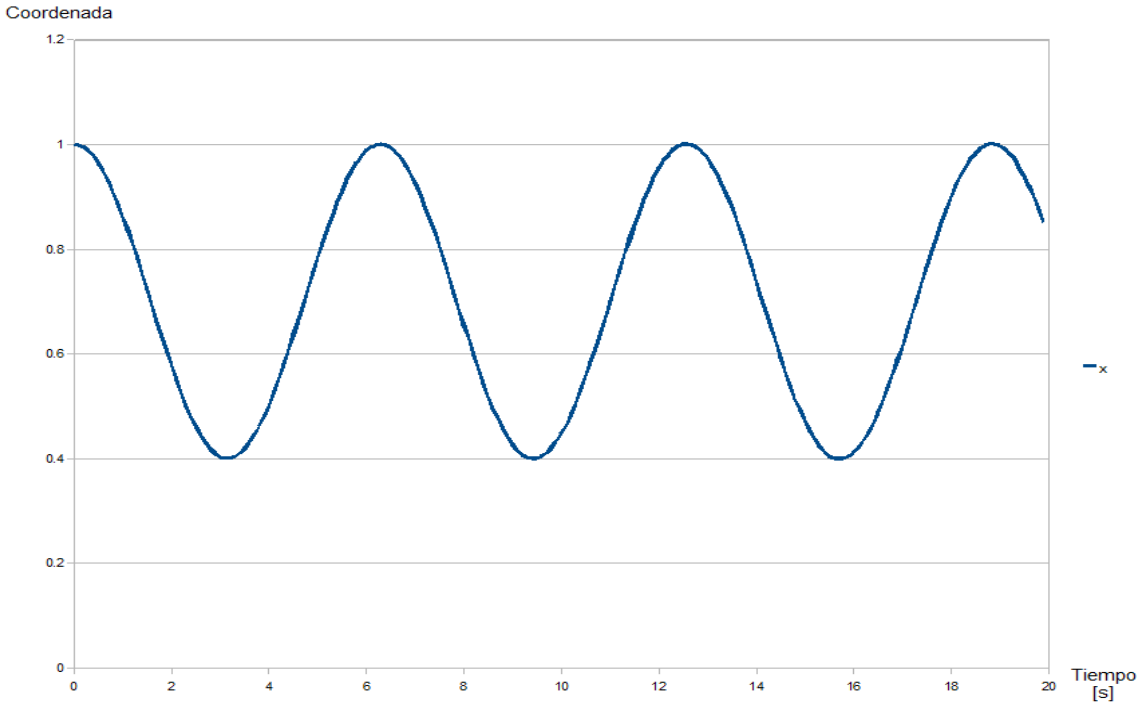

Figura 2: Gráfico de posición del bloque de masa m en función del tiempo.

b) 2 Dimensiones:

En este caso, el sistema consta de 3 bloques, dos en los extremos, y uno al medio. Y se tienen 2 resortes. Esto se muestra en la siguiente figura:

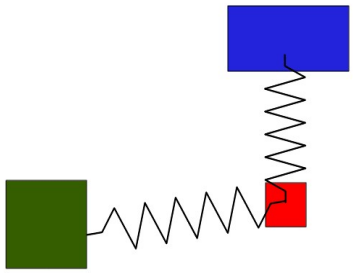

Figura 3: Esquemático bidimensional.

Las posiciones iniciales son:

Bloque 1 = (0,0)
Bloque 2 = (1,0.5)
Bloque 3 = (1,1)

En este caso los bloques 1 y 3 son de masa infinita, y el bloque 2 es de masa 1. Los resortes 1 y 2 tienen constante de elasticidad y largo en reposo iguales a 1.Los bloques tienen velocidad inicial igual a 0.

Al graficar los datos, el resultado obtenido fue el siguiente:

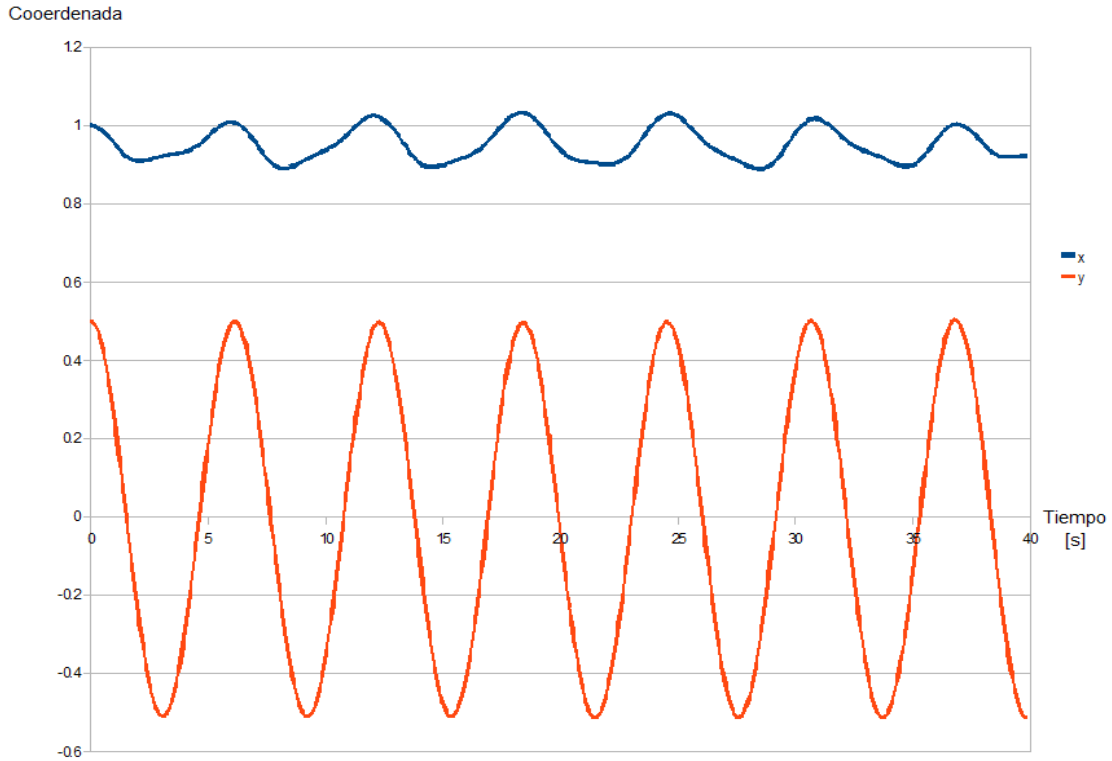

Figura 4: Gráfico de coordenadas del bloque rojo en función del tiempo.

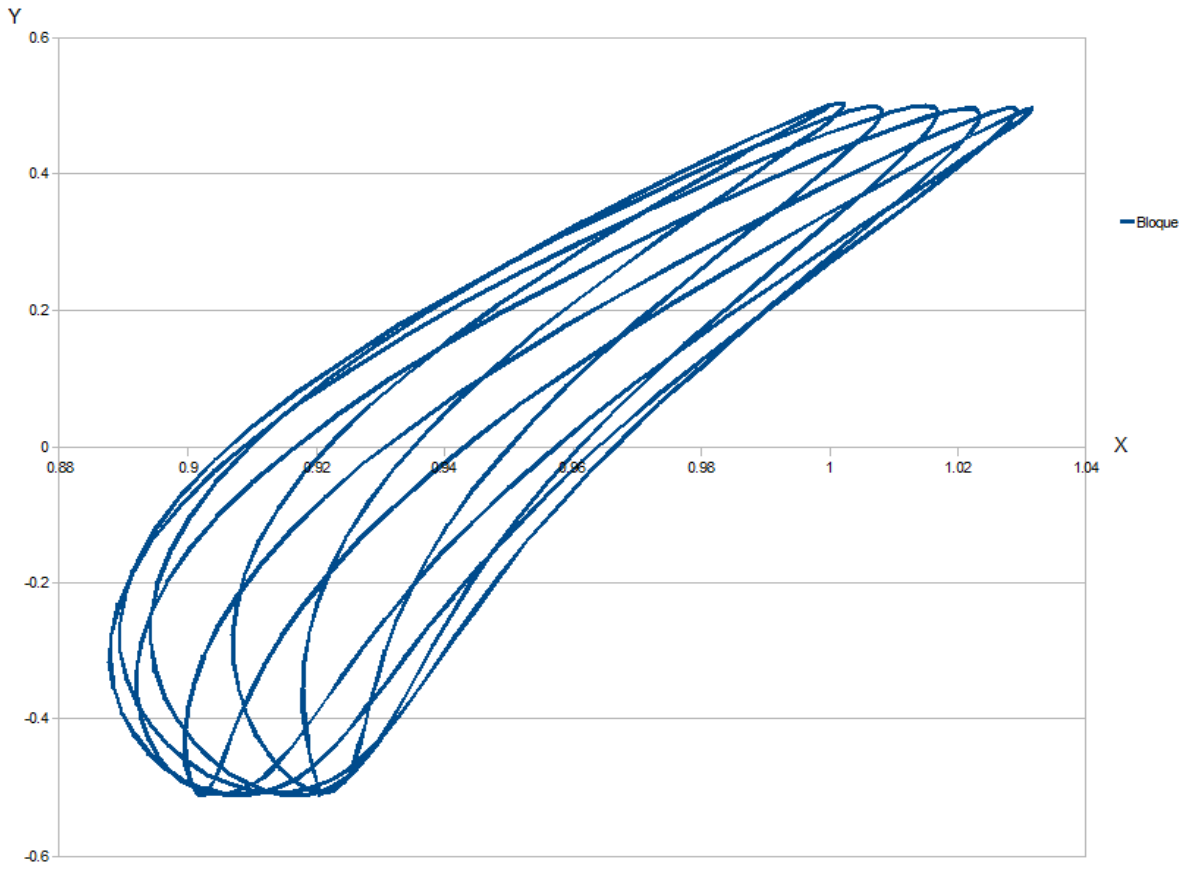

Figura 5: Gráfico de posición del bloque rojo en función del tiempo.

Dificultades:

- Hacer que al rearmar la fuerza aplicada sobre los bloques, luego de haber obtenido las diferencias de vectores, escalado y el resto de operaciones, esta quedara en al dirección correcta para ser aplicada. Principalmente debido a que no nos percatamos en un principio del rango en el que nos entregaba los datos la función Math.atan().
- Como abordar el problema de migrar a 2D, debido a la gran cantidad de variables y métodos que debían ser transformadas, para poder realizar las pruebas a medida que trabajábamos sin perder la funcionalidad inicial del programa.