Métodos Computacionales

Tarea 2 — 2019–10

Completa

Los archivos:
`hw2.mk`,
`Fourier.py`
`Edificio.cpp`
`Plots_hw2.py`
y `Resultados_hw2.tex`. deben estar comprimidos en `ApellidoNombre_hw2.zip` y deben descomprimirse en un directorio `ApellidoNombre_hw2` (5pts). Este directorio comprimido debe contener únicamente los archivos mencionados anteriormente (más los archivos de la parte 2) y debe subirlo a SICUA antes de las **10:00 pm del viernes 10 de mayo de 2019**. Adicionalmente debe haber trabajado su tarea en un repositorio de GitHub cuyo enlace debe también subir a sicua. Su repositorio debe tener varios commits hechos durante las 4 semanas que tiene para trabajar en la tarea. Recuerde que es un trabajo completamente individual (no puede usar códigos que usted no haya escrito en su totalidad ni puede recibir ayuda de nadie cuando esté escribiendo su código). Recuerde también que los códigos deben correr sin botar errores en los computadores de compufísica (Y110B). Si su código no corre la nota de ese ejericicio será de cero. Recuerde que es buena práctica comentar el código y elegir buenos nombres para sus variables. Eso será tenido en cuenta durante el proceso de corrección de esta tarea. No se aceptarán trabajos entregados tarde ni por otro medio distinto a SICUA. El comando `gmake -f hw4.mk` debe ser suficiente para que todos los códigos de la tarea corran, se generen las gráficas y se produzca el archivo `Resultados_hw2.pdf`.

1. (5 points) **Github**
   En esta tarea se evaluará el uso de Github de la siguiente manera:

   - Los archivos de solución deben, además de subirse a SICUA, estar en un repositorio cuyo enlace debe ser enviado por SICUA cuando se envíe la solución de la tarea.
   - Debe haber al menos 15 commits de la tarea en dónde se vea claramente la **evolución** del trabajo (no es válido hacer varios "commits" justo antes de la hora de entrega de la tarea).
   - Los commits se deben hacer para los archivos individuales de la tarea a medida que su trabajo en cada archivo vaya avanzando. Se espera que haya commits durante al menos 3 de las 4 semanas que tienen para trabajar en la tarea.

2. (37 points) **Transformada de Fourier: implementación propia, paquetes de scipy y espectrogramas.**
   La idea de este ejercicio es implementar un código que les permita entender y hacer espectrogramas de señales (ver por ejemplo: `https://pnsn.org/spectrograms/what-is-a-spectrogram`). En este ejercicio deben primero usar su implementación propia de la transformada de fourier para analizar dos señales isimples y luego usar paquetes de scipy y de matplotlib para hacer espectrogramas de una señal real de un sismo. No olviden poner labels y leyendas en las gráficas. **NOTA importante**: TODAS las graficas mencionadas a continuacion deben guardarse **sin ser mostradas** y ser incluidas en el pdf de resultados. Para esto debe escribir un script de python llamado `Fourier.py` que:

- Almacene los datos de `signalSuma.dat` y de `signal.dat`. Estas dos señales estan conformadas por dos ondas sinusoidales. En el primer caso la señal es la suma de las dos ondas, en el segundo la señal está conformada por, primero una de las ondas y luego la siguiente.
- Haga una gráfica con dos subplots uno con los datos de `signal.dat` y otro con los de `signalSuma.dat`.
- Haga la transformada de Fourier de ambas señales usando su **implementación propia** de la transformada discreta de fourier.
- Haga una gráfica de las transformadas de Fourier de ambas señales.
  Esta gráfica debe ser en función de las frecuencias (bono si no usa el paquete fftfreq. Indique esto con un mensaje en la terminal.)
- Usando el paquete matplotlib.pyplot.specgram (ver: `https://matplotlib.org/api/_as_gen/matplotlib.pyplot.specgram.html`) haga un espectrograma de las dos señales.
- Almacene los datos de `temblor.txt`. Estos datos son datos reales de una señal sísmica
- Haga una gráfica de la señal en función del tiempo.
- Haga la transformada de Fourier de la señal usando paquetes de scipy y grafíquela.
- haga un espectrograma de la señal.

3. (37 (+15) points) **Ecuaciones diferenciales ordinarias: un edificio en un sismo.**
   La idea de este ejercicio es hacer un modelo (muy simplificado) de un edificio para estudiar cómo éste responde a una perturbación externa, por ejemplo a un sismo. La descripción detallada de cómo modelar un edificio de tres pisos la pueden encontrar en `https://www.ijee.ie/articles/Vol15-6/ijee1107.pdf`. En las gráficas (tomadas de `https://www.ijee.ie/articles/Vol15-6/ijee1107.pdf`) pueden ver un esquema del edificio representado a partir de tres masas unidas por resortes.

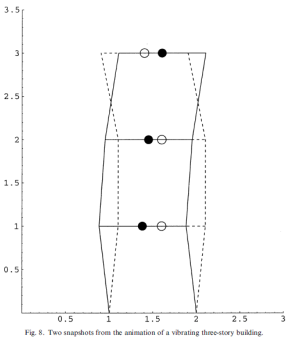

Fig. 8. Two snapshots from the animation of a vibrating three-story building.

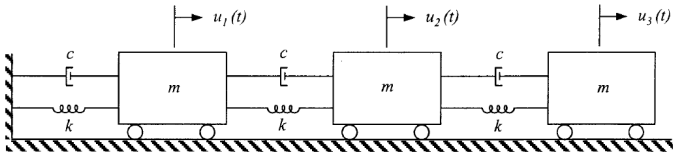

Fig. 7. Spring-mass-damper system with 3 masses.

Este sistema está descrito por un sistema de ecuaciones diferenciales de segundo orden:

$$m\frac{d^2u_1(t)}{dt^2} = -\gamma v_1(t) - 2ku_1(t) + ku_2(t) + F(t) \tag{1}$$

$$m\frac{d^2u_2(t)}{dt^2} = -\gamma v_2(t) + ku_1(t) - 2ku_2(t) + ku_3(t) \tag{2}$$

$$m\frac{d^2u_3(t)}{dt^2} = -\gamma v_3(t) + ku_2(t) - ku_3(t) \tag{3}$$

Donde $u_i(t)$ y $v_i(t)$ son las posiciones y velocidades respectivas de cada uno de los tres pisos del edificio, k es una constante que da cuenta de la rigidez de la estructura, m es la masa de cada piso y γ es un coeficiente de fricción. $F(t)$ respresenta un forzamiento externo a la estructura.

Usted debe escribir un script de C++ llamado `Edificio.cpp` que:

- con el método de Leap-Frog o el de Runge Kutta, resuelva el sistema de ecuaciones para los siguientes parámetros y forzamiento: $m = 1000.0$, $\gamma = 0.0$, $k = 2000$, $\omega = 1.0 * \sqrt{(k/m)}$, $F(t) = sin(\omega t)$, y tome como condiciones iniciales $u_i(t) = 0$ y $v_i(t) = 0$ (Note que los valores de los parámetros son arbitrarios y no tienen unidades. Hay un bono al final de la tarea si busca y usa valores realistas, incluye amortiguamiento y/o usa un forzamiento más realista.)
- Una vez haya verificado que su código funciona, varíe la frecuencia ω de forzamiento, tomando 100 valores entre $\omega = 0.2 * \sqrt{(k/m)}$ y $\omega = 3.0 * \sqrt{(k/m)}$. Genere datos que le permitan graficar la amplitud máxima de desplazamiento de los pisos $(u_i(t)_{max})$ en funcion de ω.
- BONO (15pts): En este ejercicio hay un bono abierto. La idea es que usted use el código para estudiar mejor el problema. Puede por ejemplo usar parámetros realistas, o usar un forzamiento mas realista (datos de temblor.txt?), o incluir amortiguamiento ($\gamma > 0$) o estudiar el efecto de pisos con distintas masas, o comprarar edificios de distinto número de pisos, etc... El bono debe ser una sección en el pdf de resultados dónde usted explique qué hizo, muestre sus resultados y los analice. También puede hacer una animación en python

NOTA importante: En el pdf de resultados debe incluir cinco graficas: Una de $u_i(t)_{max})$ en funcion de ω y cuatro de $u_i(t)$ en funcion del tiempo. Debe incluir además un análisis de sus resultados. Por ejemplo hable de resonancia, del número de picos, describa las graficas de amplitudes para resonancia y no resonancia, etc....
Estas expliaciones deben ser cortas y **precisas**.

4. (16 points) **Gráficas, makefiles y pdf de resultados.**
El código `Plots_hw2.py` debe (4pts):

   - Leer y guardar los datos generados por el código en C++ del ejercicio de ecuaciones diferenciales.
   - Graficar: Las amplitudes $u_i(t)_{max}$ en funcion de ω. Use esta grafica para seleccionar cuatro valores de ω que le parezcan interesantes y grafique las amplitudes en función del tiempo (una grafica por cada valor de ω). Guarde dichas gráficas sin mostrarlas (para poder luego incluirlas en el pdf de resultados.)

   Modifique el archivo `Resultados_hw2.tex` para poder (4pts):

   - Organizar las gráficas obtenidas Haga una sección por cada ejercicio de la tarea y describa brevemente sus resultados.
   En el ejercicio de Fourier, debe incluir todas las gráficas y debe explicar lo que observa en las gráficas. Debe en particular describir lo qué observa en las gráficas de espectrogramas (tambien llamados time frequency plots). Este archivo debe estar incluido dentro de las dependencias del makefile y debe permitir generar un archivo `Resultados_hw2.pdf`

El archivo `hw2.mk` debe (8pts):

- Incluir todas las dependencias y reglas necesarias para generar y actualizar el archivo: `Resultados_hw2.pdf`.
  Los archivos que deben subir a Sicua (comprimidos en `ApellidoNombre_hw2.zip`) son:
  `hw2.mk`,
  `Fourier.py`
  `Plots_hw2.py`
  `Resultados_hw2.tex`. y `Los que falta por anunciar...`,