Tarea 3: Diferencias Finitas para EDPs y Visualización Científica

Universidad de Chile **CC3501**
FCFM - DCC **16 de junio de 2020**

Equipo Docente:
Daniel Calderón, Alonso Utreras, Nelson Marambio, Beatriz Graboloza, Heinich Porro, Nadia Decar, Tomás Calderón

Instrucciones Generales

- La tarea es estrictamente individual.
- Este enunciado presenta varias opciones, escoja e implemente solo una de ellas.
- El plazo de entrega se indica en tareas/u-cursos.
- Esta tarea debe ser trabajada y entregada en un repositorio Git remoto privado, proporcionando acceso al equipo docente. Instrucciones detalladas disponibles en material docente.
- Guarde todo su trabajo para esta tarea en una carpeta de nombre tarea3x, con x indicando la opción que haya escogido.
- DEBE utilizar: Python 3.5 (o superior), Numpy, OpenGL core profile, GLFW. Si lo desea, puede resolver el problema numérico utilizando Matlab, Octave, u otro similar. Indique esto apropiadamente en su reporte.

Entregables

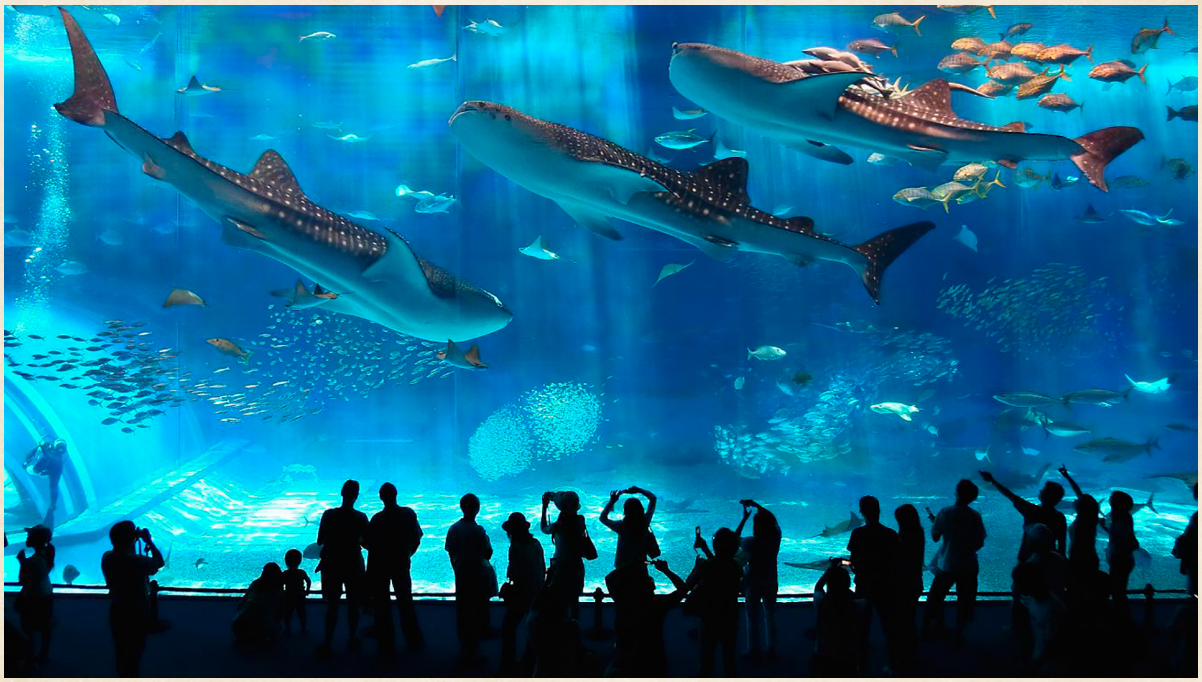
- Código que implemente su solución (4.5 puntos)
  - Su versión final DEBE utilizar archivos *.py, NO jupyter notebooks.
  - Incluya TODO lo que su código necesita, incluyendo archivos proporcionados en cátedras o auxiliares.
  - Al menos 5 commits en su repositorio git remoto ilustrando progreso en su trabajo. Nota 1 si no cumple con este item. *Importante: el mínimo de 5 commits deben corresponder a progresos en el código. Agregar reportes o videos al repositorio no contabiliza.*
- Reporte de documentación de entre 1 y 2 planas. Formato pdf. Detalles disponibles en primera clase. (1.2 puntos)
- Todo lo anterior debe estar disponible en su repositorio Git remoto.
- *Importante:* Video demostrativo de 20-30 segundos vía Tareas/u-cursos en formato mp4. (0.3 puntos)

Objetivos

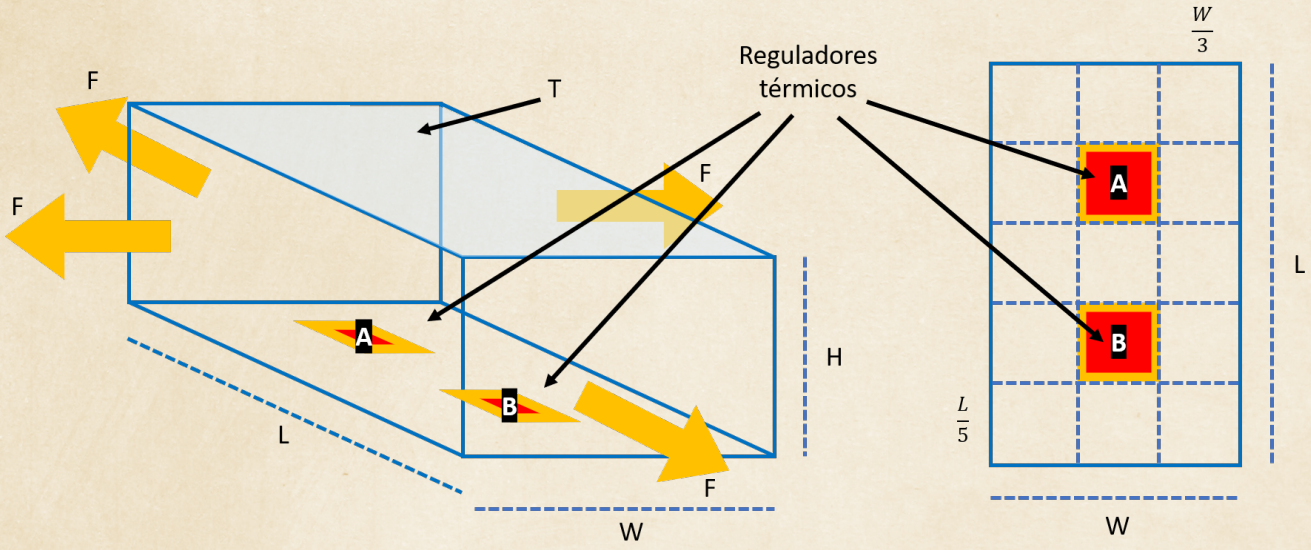
- Consolidar el uso de OpenGL core profile y GLFW en el desarrollo de una aplicación científica.
- Ejercitar y consolidar conocimientos adquiridos en el curso relativos a: discretizaciones, visualización científica y método de diferencias finitas para ecuaciones diferenciales parciales.
- Poner en práctica conocimientos de mallas poligonales, modelos en base a voxeles, animaciones simples, degradaciones de colores, visualización 3D, etc.

Opción A: Acuario

En esta tarea usted debe resolver la ecuación del calor para el agua contenida en un acuario. Distintos tipos de peces preferirán distintos sectores de la misma.



Monterey Bay Aquarium, California, US.



Lo primero, es resolver el problema de *Laplace* $\nabla u = 0$ para la temperatura u sobre el espacio ocupado por el acuario. Consideremos un acuario con forma de paralelepípedo como se indica en la figura.

Como la parte superior se encuentra destapada, tiene una fuga de calor importante. Es este sector, consideraremos la temperatura de la superficie del agua igual a la temperatura ambiente T. Los sectores laterales presentan una menor fuga térmica F (condición tipo Neumann), y finalmente, asumiremos que la parte inferior de la pecera es perfectamente aislante (condición tipo Neumann nula). La temperatura ambiente varía lentamente durante el día, por lo que la dejaremos como entrada del usuario de modo de poder realizar distintas simulaciones.

En el piso del acuario se encuentran dos reguladores térmicos que fijan las temperaturas según sea especificado por el usuario. El efecto

de estos reguladores puede ser modelado como condiciones Dirichlet para la temperatura en dichas zonas. La figura indica la disposición de estos reguladores en el fondo del acuario.

Para resolver el problema, considere:

- Usted debe definir una discretización apropiada.
- Todos los valores numéricos requeridos vendrán especificados en un archivo *json*, cuyo nombre es dado como argumento al programa.
- Almacene sus resultados para las temperaturas en un archivo. Se sugiere utilizar *np.save* y *np.load* por simplicidad.
- Considere que necesitará visualizar sus resultados en el ítem siguiente.

Ejemplo de archivo de configuración *json*:

```
{
    "height" : 4,
    "width" : 3,
    "lenght" : 6,
    "window_loss" : 0.01,
    "heater_a" : 5,
    "heater_b" : 30,
    "ambient_temperature" : 25
    "filename" : "solution.npy"
}
```

La llamada a su programa de cálculo debe ser como sigue:

```
python aquarium-solver.py problem-setup.json
```

Se requiere generar una segunda aplicación que nos permita visualizar como se distribuirán distintos tipos de peces en el interior del acuario. Considere lo siguiente:

- Tenemos 3 tipos de peces, cada uno con una personalidad distinta prefiriendo distintas temperaturas: T_a, T_b y T_c.
- Con las teclas A, B y C visualizará en voxeles los sectores del acuario aptos para cada tipo de pez. Los voxeles serán aptos para un tipo de pez si la temperatura en dicha región se encuentra dentro del rango $[T_i - 2, T_i + 2]$, donde $T_i \in \{T_a, T_b, T_c\}$. Resalte los voxeles de alguna manera conveniente.
- Genere modelos simples para cada tipo de pez y una animación simple del movimiento de su cola. El único requerimiento es que puedan ser identificados como peces y que se distingan claramente entre los distintos tipos.
- El acuario posee N_a peces de tipo A, N_b peces de tipo B y N_c de tipo C.
- Queremos visualizar como se distribuyen los peces dentro del acuario. Para esto, ubíquelos dentro de zonas cuya temperatura sea apta para cada tipo de pez. Se recomienda utilizar una distribución aleatoria dentro de las celdas factibles, pero puede utilizar otra estrategia si lo estima conveniente. El objetivo es no superponer todos los peces en la misma posición. Si hay pocas superposiciones no es relevante.
- Con el acuario en el centro de la escena, debemos poder rotar la cámara a su alrededor y poder acercarnos u alejarnos como si fuéramos espectadores del mismo. La rotación se controla con las flechas del teclado. El objetivo es poder visualizar el acuario desde cualquier ángulo.

Del mismo modo que para el item anterior. Su programa debe recibir un archivo de configuración *json* con los parámetros requeridos. Ejemplo de archivo *json*:

```
{
    "filename" : "solution.npy"
    "t_a" : 15,
    "t_b" : 10,
    "t_c" : 25,
    "n_a" : 5,
    "n_b" : 3,
    "n_c" : 7
}
```

Su programa de visualización se debe ejecutar como sigue:

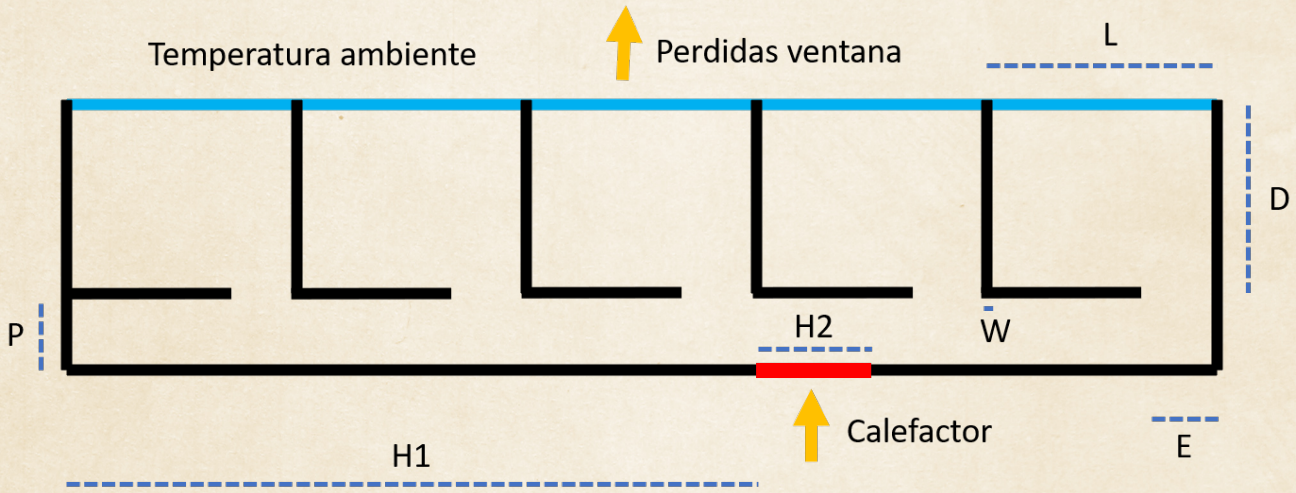
```
python aquarium-view.py view-setup.json
```

Puntuación

- Solución EDP en 3D: 2.5 puntos
- Voxeles de sectores de preferencia: 0.5 puntos
- Modelos y animación de peces: 0.5 puntos
- Visualización de acuario y peces: 1.0 punto

Opción B: Hotel de Don Pedro

Don Pedro quiere instalar un sistema de calefacción centralizado para su nuevo hotel. El objetivo es temperar "eficientemente" todas las habitaciones. La distribución de las piezas es simple, un pasillo da acceso a ellas, y cada una posee una ventana con una vista espectacular. El hotel posee 5 habitaciones.



Respecto de la solución del problema mismo, considere lo siguiente:

- Consideraremos que la temperatura no varía en altura, por lo que se trata de un problema en 2D.
- Resolveremos la ecuación de *Laplace* $\nabla u = 0$ para la temperatura u al interior del hotel. Usted debe determinar una discretización apropiada.
- El gran problema: Los habitantes viven abriendo las ventanas. Dependiendo de si la vantana está abierta o cerrada, cambian las condiciones de borde del problema.
- Cuando la ventana está cerrada, tenemos una condición de borde tipo Neumann igual a *window_loss*. Cuando está abierta, tenemos una condición de borde tipo Dirichlet igual a *ambient_temperature*.
- El hotel posee un poderoso calefactor que se encarga de generar calor vía una condición de Neumann entrante igual a *heater_power* en el sector indicado.
- Los muros del hotel son perfectamente aislantes (condición de Neumann nula), por lo que deben ser considerados en la especificación del dominio del problema.
- Lamentablemente, al menos una ventana del hotel siempre se encontrará abierta. Por lo que siempre habrá una condición tipo Dirichlet dejando el problema bien definido.
- Almacene sus resultados para las temperaturas en un archivo. Se sugiere utilizar *np.save* y *np.load* por simplicidad.
- Considere que necesitará visualizar sus resultados en el ítem siguiente.

Adicionalmente, queremos visualizar la solución convenientemente sobre el modelo tridimensional del hotel.

- Visualizaremos el hotel de Don Pedro como un modelo 3D. Se deben ver muros, suelo, techo y ventanas.
- La cámara debe poder moverse simulando que el usuario camina al interior del hotel. No se preocupe por colisiones con los muros, puede atravesarlos como si fuera un fantasma... Defina un control de cámara apropiado para navegar por el hotel.
- En el piso, dibujaremos un mapa de calor con escala de colores, desde el valor más frío (blanco), pasando por temperaturas intermedias (amarillo-anaranjado), hasta el más cálido (rojo). La escala de colores debe ser una degradación suave. Si lo desea puede utilizar otros 3 colores para definir su escala de temperaturas. En la terminal indique los 3 colores y las 3 temperaturas utilizadas para generar la degradación.
- Al presionar *espacio*, se visualizarán curvas de nivel en el hotel. La elevación de la curva de nivel debe ser proporcional al valor de la temperatura que representa. Considere 10

curvas de nivel entre el suelo y el techo.
- Al presionar *control-derecho*, debe visualizar el gradiente de temperatura como flechas a una pequeña distancia sobre el suelo.
- Al presionar *espacio* o *control-derecho*, debe alternar la visualización de curvas de nivel de temperatura y/o flechas del gradiente de temperaturas.

Ambos programas, tanto el de solución como el de visualización deben recibir un archivo *json* con la configuración del problema. El siguiente es un ejemplo.

```
{
    "filename" : "solution.npy",
    "window_loss" : 0.01
    "ambient_temperature" : 20,
    "heater_power" : 3,
    "P" : 1,
    "L" : 4,
    "D" : 5,
    "W" : 0.1,
    "E" : 1,
    "H1" : 12,
    "H2" : 2,
    "windows" : [0,0,1,1,0]
}
```

En el campo *windows*, se indica con 0 o 1 si la ventana de la habitación i-ésima se encuentra abierta o cerrada.

Finalmente, la llamada a su programa de resolución debe ser:

```
python hotel-solver.py hotel.json
```

y a su programa de visualización:

```
python hotel-viewer.py hotel.json
```

Puntuación

- Solución EDP en 2D: 2 puntos
- Modelo 3D y control de cámara: 0.5 puntos
- Mapa de calor: 0.5 puntos

- Gradiente de temperaturas: 0.5 puntos
- Curvas de nivel: 1.0 punto