IPM-407 — Proyecto 2
Métodos basados en transformadas de Fourier

Consideren la siguiente ecuación de Poisson

$$\frac{\partial^2\phi}{\partial x^2}+\frac{\partial^2\phi}{\partial y^2}=\frac{1}{\pi^2}\left[30\left(\left(\frac{x}{\pi}\right)^2-\frac{x}{\pi}\right)+30\left(\left(\frac{y}{\pi}\right)^2-\frac{y}{\pi}\right)-4\pi^2\left(\frac{x}{\pi}-1\right)\sin\left(2y\right)\right] \tag{1}$$

en el cuadrado $0\leq x\leq\pi$ y $0\leq y\leq\pi$ y con condiciones de contorno $\phi(0,y)=\phi(\pi,y)=\phi(x,0)=\phi(x,\pi)=0$. Resuelvan numéricamente esta ecuación usando

1. Método espectral
2. Diferencias finitas aceleradas con FFT en 2D (tipo pregunta 4 de la tarea 2)
3. Diferencias finitas aceleradas con FFT en 1D (tipo pregunta 5 de la tarea)

Para mallas de $N=16,32,64$ y 128 nodos (o más!) En su informe, detalle la implementación de cada técnica y discuta los siguientes puntos

1. Convergencia del error.
2. Complejidad algoritmica (N versus tiempo).
3. Memoria utilizada.
4. Conclusiones con respecto a la conveniencia de cada caso.

La solución analítica de esta ecuación es

$$\phi(x,y)=15\left(\left(\frac{x}{\pi}\right)^2-\frac{x}{\pi}\right)\left(\left(\frac{y}{\pi}\right)^2-\frac{y}{\pi}\right)-\sin(2y)\frac{\sinh\left(2\pi\left(\frac{x}{\pi}-1\right)\right)}{\sinh(2\pi)}+\left(\frac{x}{\pi}-1\right)\sin(2y) \tag{2}$$

Ayuda

- Esta tarea está inspirada en el Ejemplo 6.4 del libro *Fundamentals of Engineering Numerical Analysis* de Parviz Moin.
- Las condiciones de borde homogénea $\phi=0$ en los bordes son concordantes con la función seno, por lo tanto es más conveniente usar la transformada del seno. Hay una implementación de la transformada del seno en `scipy.fftpack`[1]. Otra opción es calcular la transformada discreta del seno con FFTs de la forma:

```
def dst_2D(x):

        M,N = numpy.shape(x)
        x   = x[1:M-1,:]
        M,N = numpy.shape(x)
        y = numpy.zeros((2*(M+1),N))
        y[0,:]      = numpy.zeros(N)[:]
```

[1] `https://docs.scipy.org/doc/scipy/reference/fftpack.html`

```
        y[1:M+1,:] = x[:,:]
        y[M+1,:]    = numpy.zeros(N)[:]
        y[M+2:2*M+2,:] = -numpy.flipud(x)[:,:]
        y_t = fft(numpy.transpose(y))
        y_t = numpy.transpose(y_t)
        y = numpy.real((y_t[1:M+1])/(-1j*(M+1)))

        yy = numpy.zeros((M+2,N))
        yy[1:M+1,:] = y[:,:]

        return yy

def dst_1D(x):

        M = len(x)
        x = x[1:M-1]
        M = len(x)
        y = array([0])
        y = numpy.append(y,x)
        y = numpy.append(y,0)
        y = numpy.append(y, -numpy.flipud(x))
        y_t = numpy.fft.fft(y)

        y = numpy.real(y_t[1:M+1]/(-1j*(M+1)))
        yy = numpy.array([0])
        yy = numpy.append(yy, y)
        yy = numpy.append(yy, 0)

        return yy
```

Considerando que la transformada inversa del seno es igual a la transformada del seno multiplicado por $N/2$