1. *Back to the Future.*

(a)
- Primero se debe construir el lado derecho de la ecuación, es decir, el vector $\mathbf{w}$. Para asegurar una mejor aproximación, es decir, con la menor cota superior del error en el intervalo, se utilizará el conjunto S_2 de los puntos de Chebyshev: $\mathbf{x}^c$ e $\mathbf{y}^c$. Entonces,
  (I) **[5 puntos]:** se construye un polinomio interpolador $p_{m-1}(x)$ utilizando Interpolación Baricéntrica para reducir la cantidad de operaciones elementales en su evaluación. La interpolación polinomial se realizará con $\mathbf{x}^c$ e $\mathbf{y}^c$, es decir con los puntos de Chebyshev disponibles. Esta elección se hace para reducir el fenómeno de Runge. La interpolación polinomial asegura que $p_{m-1}(x_k^c) = y_k^c$, para así posteriormente aproximar $f(x_j)$ con $p_{m-1}(x_j)$, es decir,
  (II) **[5 puntos]:** se evalúan los n puntos $x_j = \frac{2}{n-1}(j-1)-1$ en $p_{m-1}(x_j)$ para $j = \{1, \dots, n\}$. Notar que el vector $\mathbf{w}$ se define como $[f(x_1), f(x_2), \dots, f(x_j), \dots, f(x_n)]^T$ sin embargo se aproximará con $[p_{m-1}(x_1), p_{m-1}(x_2), \dots, p_{m-1}(x_j), \dots, p_{m-1}(x_n)]^T$
- **[10 puntos]:** Segundo, se debe resolver la ecuación, $T_n\,\mathbf{v} = \mathbf{w}$, para encontrar el vector condensador. Tomamos ventaja de la forma de la matriz T_n, ya que está factorizada en dos matrices triangulares, es decir, conocemos la factorización LU de T_n. Entonces realmente tenemos $L_n\,U_n\,\mathbf{v} = \mathbf{w}$, (i) definimos el vector auxiliar $\mathbf{z} = U_n\,\mathbf{v}$, lo cual genera $L_n\,U_n\,\mathbf{v} = L_n\,\mathbf{z} = \mathbf{w}$. Es decir debemos resolver el sistema de ecuaciones lineales $L_n\,\mathbf{z} = \mathbf{w}$ y dado que tiene una estructura bidiagonal-"triangular inferior", podemos obtener $\mathbf{z}$ utilizando una versión reducida de *forward substitution* para tomar ventaja de la estructura presentada y (ii) luego de obtener $\mathbf{z}$, podemos obtener $\mathbf{v}$ resolviendo el sistema de ecuaciones lineales bidiagonal-"triangular superior" $U_n\,\mathbf{v} = \mathbf{z}$ utilizando una versión reducida de *backward substitution* para tomar ventaja de la estructura presentada.
  Para resolver $L_n\,\mathbf{z} = \mathbf{w}$, mediante una versión reducida de *forward substitution*, debemos realizar los siguientes pasos para tomar ventaja de la estructura de L_n y obtener $\mathbf{z}$:
  - *1ra ecuación*: $z_1 = w_1$, la cual entrega el valor de z_1 directamente, es decir $z_1 = w_1$.
  - *2da ecuación*: $-\dfrac{1}{2}\,z_1 + z_2 = w_2$, despejando z_2 obtenemos $z_2 = w_2 + \dfrac{1}{2}\,z_1$. Recordar que en este paso ya conocemos el valor de z_1.
  - *i-ésima ecuación*: $-\dfrac{(i-1)}{i}\,z_{i-1} + z_i = w_i$, despejando z_i se obtiene $z_i = w_i + \dfrac{(i-1)}{i}\,z_{i-1}$ para $i = \{3, \dots, n-1\}$.
  - *n-ésima ecuación*: $-\dfrac{(n-1)}{n}\,z_{n-1} + z_n = w_n$, finalmente despejando z_n se obtiene $z_n = w_n + \dfrac{(n-1)}{n}\,z_{n-1}$.
- **[10 puntos]:** Por otro lado, para resolver $U_n\,\mathbf{v} = \mathbf{z}$, mediante una versión reducida de *backward substitution*, debemos realizar los siguientes pasos para tomar ventaja de la estructura de U_n y obtener $\mathbf{v}$:
  - *n-ésima ecuación*: $\dfrac{n+1}{n}\,v_n = z_n$, por lo tanto despejando v_n obtenemos $v_n = \dfrac{n}{n+1}\,z_n$.
  - $(n-1)$*-ésima ecuación*: $\dfrac{n}{n-1}\,v_{n-1} - v_n = z_n$, despejando v_{n-1} obtenemos $v_{n-1} = \dfrac{n-1}{n}\,(z_n + v_n)$.
  - *i-ésima ecuación*: $\dfrac{i+1}{i}\,v_i - v_{i+1} = z_i$, despejando v_i obtenemos $v_i = \dfrac{i}{i+1}\,(z_i + v_{i+1})$ para $i = \{2, \dots, n-2\}$.
  - *1ra ecuación*: $2\,v_1 - v_2 = z_1$, despejando v_1 obtenemos $v_1 = \dfrac{1}{2}\,(z_1 + v_2)$.

  Por lo tanto se puede obtener $\mathbf{v}$ primero interpolando la *data* disponibles en los puntos de Chebyshev con interpolación Baricéntrica y luego resolviendo la secuencia de sistemas de ecuaciones lineales generados por L_n y U_n respectivamente, tomando ventaja de las definiciones de L_n y U_n entregadas.

(b)
```
'''
input:
n : (integer) dimension of the matrix T_n.
output:
v : (ndarray) array of dimension n that stores the estimation of capacitor vector "v".
'''
def capacitor_vector(n):
    % Loading m-dimensional vector data $\mathbf{x}^e$ and $\mathbf{y}^e$
    xe = np.load('/ScientificComputing/DrEmmettData_xe.npz')
    ye = np.load('/ScientificComputing/DrEmmettData_ye.npz')
    % Loading m-dimensional vector data $\mathbf{x}^c$ and $\mathbf{y}^c$
    xc = np.load('/ScientificComputing/DrEmmettData_xc.npz')
    yc = np.load('/ScientificComputing/DrEmmettData_yc.npz')
    % Hint: You should only use one pair of vectors
```

- [**5 puntos**] Obtener el vector $\mathbf{w}$ mediante interpolación Baricéntrica

```
j = np.arange(1,n + 1)
xj = (2*(j - 1))/(n - 1) - 1.
pB = BarycentricInterpolation(xc,yc)
w = pB(xj)
```

- [**10 puntos**] Resolver el sistema de ecuaciones $L_n\mathbf{z} = \mathbf{w}$

```
z = np.ones((n,1)) # or better "z = np.ones(n)"
z[0] = w[0]
for i in np.arange(2,n + 1):
  z[i - 1] = w[i - 1] + (i - 1)*z[i - 2]/(i)
```

- [**10 puntos**] Resolver el sistema de ecuaciones $U_n\mathbf{v} = \mathbf{z}$

```
v = np.ones((n,1)) # or better "v = np.ones(n)"
v[n-1] = (n*z[n - 1])/(n + 1)
for i in np.arange(n - 1,0,-1):
  v[i-1] = (i*(z[i-1] + v[i]))/(i + 1)
return v
```

2. *Ambigüedad espacial.*

(a)

- **[10 puntos]:** La primera tarea es construir la aproximación paramétrica:

$$\mathbf{r}(t) = \langle f_1(t), f_2(t)\rangle = \langle a_1 + b_1\, t, a_2 + b_2\, t\rangle$$

Esto significa que debemos obtener los valores de a_1, b_1, a_2 y b_2. Para esto, debemos resolver **dos** problemas de mínimos cuadrados: aproximar $f_1(t)$ que tiene relación con las coordenadas x_k de las mediciones y aproximar $f_2(t)$ que tiene relación con las coordenadas y_k de las mediciones. Entonces debemos resolver los siguientes sistemas de ecuaciones $A\,\mathbf{c}_1 = \mathbf{x}_k$ y $A\,\mathbf{c}_2 = \mathbf{y}_k$, donde los lados derechos $\mathbf{x}_k$ y $\mathbf{y}_k$ corresponden a las coordenadas x_k e y_k y los vectores $\mathbf{c}_1$ y $\mathbf{c}_2$ corresponden a los coeficientes $[a_1, b_1]^T$ y $[a_2, b_2]^T$, respectivamente:

$$\begin{array}{rcl} a_1 + b_1\, t_1 & = & x_1 \\ a_1 + b_1\, t_2 & = & x_2 \\ a_1 + b_1\, t_3 & = & x_3 \\ \vdots & \vdots & \vdots \\ a_1 + b_1\, t_{n-2} & = & x_{n-2} \\ a_1 + b_1\, t_{n-1} & = & x_{n-1} \\ a_1 + b_1\, t_n & = & x_n \end{array} \Rightarrow \begin{bmatrix} 1 & t_1 \\ 1 & t_2 \\ 1 & t_3 \\ \vdots & \vdots \\ 1 & t_{n-2} \\ 1 & t_{n-1} \\ 1 & t_n \end{bmatrix} \begin{bmatrix} a_1 \\ b_1 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n-2} \\ x_{n-1} \\ x_n \end{bmatrix} \Rightarrow A\,\mathbf{c}_1 = \mathbf{x}_k$$

$$\begin{array}{rcl} a_2 + b_2\, t_1 & = & y_1 \\ a_2 + b_2\, t_2 & = & y_2 \\ a_2 + b_2\, t_3 & = & y_3 \\ \vdots & \vdots & \vdots \\ a_2 + b_2\, t_{n-2} & = & y_{n-2} \\ a_2 + b_2\, t_{n-1} & = & y_{n-1} \\ a_2 + b_2\, t_n & = & y_n \end{array} \Rightarrow \begin{bmatrix} 1 & t_1 \\ 1 & t_2 \\ 1 & t_3 \\ \vdots & \vdots \\ 1 & t_{n-2} \\ 1 & t_{n-1} \\ 1 & t_n \end{bmatrix} \begin{bmatrix} a_2 \\ b_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{n-2} \\ y_{n-1} \\ y_n \end{bmatrix} \Rightarrow A\,\mathbf{c}_2 = \mathbf{y}_k$$

Notar que la matriz A asociada a ambos sistemas de ecuaciones lineales sobre-determinados es la misma, dado que se tiene los datos en el mismo tiempo.

- **[5 puntos]:** Luego de construido el sistema de ecuaciones lineales sobre-determinado, se puede encontrar los coeficientes que minimizan el error cuadrático de la matriz A, es decir $A = \widehat{Q}\,\widehat{R}$, de la siguiente forma:

$$\overline{\mathbf{c}}_1 = \begin{bmatrix} \overline{a}_1 \\ \overline{b}_1 \end{bmatrix} = \widehat{R}^{-1}\,\widehat{Q}^* \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad \text{y} \quad \overline{\mathbf{c}}_2 = \begin{bmatrix} \overline{a}_2 \\ \overline{b}_2 \end{bmatrix} = \widehat{R}^{-1}\,\widehat{Q}^* \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}.$$

Notar que $\widehat{R}^{-1}$-por-un-vector nos indica que debemos resolver el sistema de ecuaciones lineales asociado, en particular acá se puede utilizar *backward substitution*. También es posible haber usado las ecuaciones normales.

- **[10 puntos]:** Una vez construida las parametrizaciones solicitadas podemos minimizar la función $g(t)$, la cual la definimos ahora con los parámetros obtenidos, es decir,

$$g(t) = \left(p_x - \overline{a}_1 - \overline{b}_1\, t\right)^2 + \left(p_y - \overline{a}_2 - \overline{b}_2\, t\right)^2.$$

Para encontrar el mínimo, podemos obtener la derivada,

$$\begin{aligned} g'(t) &= 2\,\left(p_x - \overline{a}_1 - \overline{b}_1\, t\right)\,(-\overline{b}_1) + 2\,\left(p_y - \overline{a}_2 - \overline{b}_2\, t\right)\,(-\overline{b}_2), \\ &= -2\,\left(p_x\,\overline{b}_1 - \overline{a}_1\,\overline{b}_1 - \overline{b}_1^2\, t\right) - 2\,\left(p_y\,\overline{b}_2 - \overline{a}_2\,\overline{b}_2 - \overline{b}_2^2\, t\right) \\ &= -2\,\left(p_x\,\overline{b}_1 - \overline{a}_1\,\overline{b}_1 - \overline{b}_1^2\, t + p_y\,\overline{b}_2 - \overline{a}_2\,\overline{b}_2 - \overline{b}_2^2\, t\right) \end{aligned}$$

simplificando e igualando a 0 obtenemos,

$$\begin{aligned} \left(p_x\,\overline{b}_1 - \overline{a}_1\,\overline{b}_1 - \overline{b}_1^2\, t + p_y\,\overline{b}_2 - \overline{a}_2\,\overline{b}_2 - \overline{b}_2^2\, t\right) &= 0, \\ \left(\overline{b}_1^2 + \overline{b}_2^2\right)\,\tilde{t} &= \left(p_x\,\overline{b}_1 - \overline{a}_1\,\overline{b}_1 + p_y\,\overline{b}_2 - \overline{a}_2\,\overline{b}_2\right) \\ \tilde{t} &= \frac{\left(p_x\,\overline{b}_1 - \overline{a}_1\,\overline{b}_1 + p_y\,\overline{b}_2 - \overline{a}_2\,\overline{b}_2\right)}{\overline{b}_1^2 + \overline{b}_2^2}. \end{aligned}$$

- **[5 puntos]:** Por lo tanto la distancia mínima es $\sqrt{g\left(\tilde{t}\right)}$.

(b)
```
'''
input:
p  : (ndarray) 2-dimensional point p.
xk : (ndarray) array of dimension "n" that stores the "x_k" measurements.
yk : (ndarray) array of dimension "n" that stores the "y_k" measurements.
tk : (ndarray) array of dimension "n" that stores the "t_k" measurements.
n  : (integer) Number of measurements.

output:
d  : (float) minimal distance from point p to the parametric representation of street C.
'''
def map_matching(p,xk,yk,tk,n):
```

- **[5 puntos]:** construir la matriz A para los problemas de mínimos cuadrados asociado y obtener su factorización QR reducida.
  ```
  A = np.ones((n,2))
  A[:,1] = tk
  Q,R = np.linalg.qr(A,mode="reduced")
  ```
- **[10 puntos]:** resolver el problema de mínimos cuadrados $A\,\mathbf{c}_1 = \mathbf{x}_k$ y $A\,\mathbf{c}_2 = \mathbf{y}_k$
  ```
  c1 = np.dot(np.transpose(Q),xk)
  a1,b1 = np.linalg.solve(R,c1)
  c2 = np.dot(np.transpose(Q),yk)
  a2,b2  = np.linalg.solve(R,c2)
  ```
- **[5 puntos]:** calcular $\tilde{t}$ donde la distancia es mínima
  ```
  t_tilde = (p[0]*b1 - a1*b1 + p[1]*b2 - a2*b2)/(b1**2 + b2**2)
  ```
- **[5 puntos]:** calcular la distancia mínima
  ```
  xt_tilde = a1 + b1*t_tilde
  yt_tilde = a2 + b2*t_tilde
  d = np.linalg.norm(p - np.array([xt_tilde,yt_tilde]))
  return d
  ```