Universidad Nacional de Colombia
Facultad de Ciencias
Análisis Numérico I
Parcial III

Mateo Andrés Manosalva Amaris
Edgar Santiago Ochoa Quiroga
Sergio Alejandro Bello Torres

1. Problema 1

Dada la función

$$f(x) = x + \frac{1}{x} - 2, \quad f : \mathbb{R}_{>0} \to \mathbb{R},$$

se construye el siguiente algoritmo para aproximar la raíz $r = 1$:

$$x_{n+1} = 2 - \frac{1}{x_n}.$$

- Verificar que si $x_0 > 1$ entonces la sucesión $\{x_n\}$ es monótona decreciente y acotada inferiormente por 1. Concluir que $x_n \to 1$, aunque esta iteración no está en las hipótesis del teorema del punto fijo. ¿Qué hipótesis no se cumple?
- Dar un algoritmo para aproximar la raíz de f que converja cuadráticamente.

2. Problema 2

Sea

$$f(x) = (x - r_1)(x - r_2)\dots(x - r_d),$$

donde $r_1 < r_2 < \dots < r_d$.

- Probar que si $x_0 > r_d$ la sucesión de Newton-Raphson converge a r_d.

  Demostración. Escribamos $f(x) = (x - r_d)g(x)$ con $g(x) = (x - r_1)\cdots(x - r_{d-1})$. De esta manera, $f'(x) = g(x) + (x - r_d)g'(x)$, luego la iteración del metodo de Newton Raphson es de la forma:

  $$x_{k+1} = x_k - \frac{(x_k - r_d)g(x)}{(x_k - r_d)g'(x_k) + g(x_k)}$$

  Ahora asuma que $x_0 > r_d$, es decir $x_0 - r_d > 0$, esto implica que $x_0 - r_i > 0$ para $i = 1, \dots, a$ pues r_d es la mayor raíz de $f(x)$. Con esto en mente note que:

  $$\begin{aligned} x_1 - r_d &= x_0 - r_d - \frac{(x_0 - r_d)g(x)}{(x_0 - r_d)g'(x_0) + g(x_0)} \\ &= (x_0 - r_d)\left(1 - \frac{g(x_0)}{(x_0 - r_d)g'(x_0) + g(x_0)}\right) \end{aligned}$$

Ahora, veamos que $g'(x_0) > 0$:

$$g'(x_0) = \sum_{i=1}^{d-1} \left(\prod_{\substack{j=1 \\ j \neq i}}^{d-1} (x_0 - r_j) \right)$$

Como cada factor de cada producto es positivo, tenemos que cada sumando es positivo, y por lo tanto $g'(x_0) > 0$,luego $(x_0 - r_d)g'(x_0) + g(x) > g(x)$. Así

$$0 < \frac{g(x_0)}{(x_0 - r_d)g'(x_0) + g(x_0)} < 1$$

Y así

$$x_1 - r_d = (x_0 - r_d)\left(1 - \frac{g(x_0)}{(x_0 - r_d)g'(x_0) + g(x_0)}\right) > 0$$

Por lo tanto $x_1 - r_d > 0$. Procediendo de manera inductiva llegamos a que $x_k - r_d > 0$ y además, como

$$0 < \left(1 - \frac{g(x_{k-1})}{(x_{k-1} - r_d)g'(x_{k-1}) + g(x_{k-1})}\right) < 1$$

Vemos que existe una constante $M \in (0,1)$ tal que $x_k - r_d < M(x_{k+1} - r_d)$ lo cual implica que $x_k - r_d < M^k(x_0 - r_d)$, como $M^k \xrightarrow{n\to\infty} 0$, tenemos que x_k converge a r_d

□

- Para un polinomio

$$P(x) = a_d x^d + \cdots + a_0, \quad a_d \neq 0,$$

tal que sus d raíces son reales y distintas, se propone el siguiente método para aproximar todas sus raíces:

  - Se comienza con un valor x_0 mayor que

$$M = \text{máx}\left\{1, \sum_{i=0}^{d-1} \frac{|a_i|}{|a_d|}\right\}.$$

  - Se genera a partir de x_0 la sucesión de Newton-Raphson, que, según el ítem anterior, converge a la raíz más grande de P, llamémosla r_d; obteniéndose de este modo un valor aproximado $\tilde{r_d}$.
  - Se divide P por $x - \tilde{r_d}$ y se desprecia el resto, dado que $r_d \approx \tilde{r_d}$. Se redefine ahora P como el resultado de esta división y se comienza nuevamente desde el primer ítem, para hallar las otras raíces.

- Aplicar este método para aproximar todas las raíces del polinomio

$$P(x) = 2x^3 - 4x + 1.$$

Solución. Para implementar éste metodo en Matlab tendremos en cuenta varios aspectos importantes: si tenemos $P(x) = a_d x^d + \ldots + a_0$, su derivada es $P'(x) = d\,a_d x^{d-1} + \ldots + a_2 x + a_1$, con lo cual podemos definir dos funciones tales que dada una lista de coeficientes y una variable x, podemos evaluar $P(x)$ y $P'(x)$, además, sabemos que evaluar directamente es

muy costoso computacionalmente, pues a medida que aumenta el grado del polinomio, hay que efectuar muchas operaciones tan solo para calcular x^d, por lo tanto escribimos $P(x) = x(x(\cdots(a_d x + a_{d-1})\cdots) + a_1) + a_0$ y hacemos lo mismo con $P'(x)$. Tambien consideramos que al dividir por $x - r_i$, podemos aplicar regla de Ruffini o división sintética, es decir, si tenemos $p(x) = a_n x^n + \cdots + a_0$ y dividimos entre $b(x) = x - r$ obtendremos un polinomio $q(x) = b_{n-1}x^{n-1} + \cdots + b_0$ a partir de la siguiente regla de recurrencia:

$$\begin{aligned} b_{n-1} &= a_n \\ b_{n-2} &= a_{n-1} + r b_{n-1} \\ &\vdots \\ b_0 &= a_1 + r b_1 \end{aligned}$$

Note que en este caso omitimos el residuo de la división, pues solo nos importa obtener los coeficientes.

Ya con todo esto, podemos aplicar el método de Newton-Rapshon al polinomio dado, de donde obtuvimos que sus raíces son $r_1 = -1{,}525687120865518$, $r_2 = 0{,}258652022504153$ y $r_3 = 1{,}267035098361366$. Al evaluar el polinomio en cada raíz obtenemos que $P(r_1) = 0{,}111022302462516 \times 10^{-15}$, $P(r_2) = -0{,}888178419700125 \times 10^{-15}$ y $P(r_3) = -0{,}666133814775094 \times 10^{-15}$. Con esto podemos observar que el método de Newton-Raphson aproxima muy bien las raíces, pues aunque al hacer división sintética nos arriesgamos a perder un coeficiente, las aproximaciones resultantes son muy exactas, con una precisión de 15 cifras decimales.

3. Problema 3

Sea $f \in C^2[a,b]$, y sean $x_0 = a, x_1 = a + h, \ldots, x_n = b$, donde $h = \dfrac{b-a}{n}$. Considerar la poligonal $l(x)$ que interpola a f en los puntos $x_i, i = 0 \ldots n$. Probar que

a)

$$|f(x) - l(x)| \leq \frac{h^2}{2} \max_{x \in [a,b]} |f''(x)|$$

.

b)

$$|f'(x) - l'(x)| \leq h \max_{x \in [a,b]} |f''(x)|$$

.

4. Problema 4: Silueta de la Mano

Para dibujar la silueta de su mano, siga los siguientes pasos:

- Preparamos una tabla de abcisas y ordenadas usando los siguientes comandos de MATLAB:

```
figure('position',get(0,'screensize'))
axes('position',[0 0 1 1])
[x,y] = ginput;
```

- Dibuje su mano en un papel y póngalo sobre la pantalla del computador. Use el ratón para seleccionar alrededor de 37 puntos que delineen su mano (como se muestra en la figura). Termine la instrucción `ginput` oprimiendo enter.
- Grafique los puntos (x, y) obtenidos y la mano correspondiente mediante el comando `plot` de MATLAB.
- Implemente el método de splines cúbicos.
- Interpole por separado los puntos (i, x_i) e (i, y_i) mediante splines cúbicos usando su programa.
- Grafique la curva parametrizada que se obtiene.
- Estime el área de su mano usando la fórmula del área de Gauss:

$$A = \frac{1}{2}\left|\sum_{i=1}^{n-1} x_i y_{i+1} + x_n y_1 - \sum_{i=1}^{n-1} x_{i+1} y_i - x_1 y_n\right|.$$

Solución. Primero comenzamos ejecutando los comandos sugeridos para conocer su funcionamiento, graficamos los puntos seleccionados y obtuvimos un resultado poco suave, el que justamente queremos interpolar por splines cúbicos, realizamos esto primero con el comando predefinido en Matlab. Como la curva de la mano es paramétrica (no necesariamente es función), tomamos las coordenadas (x_i, y_i) y realizamos interpolación componente por componente, es decir encontramos splines para cada componente en función de t, exáctamente lo mismo hicimos con nuestro código.

Recordemos que para splines cúbicos los polinomios deben cumplir varias condiciones, en este caso vamos a implementar una spline cúbica natutal, por lo que estas condiciones se obtienen de solucionar el sistema

$$\begin{bmatrix} a_1 & b_1 & & & \\ b_1 & a_2 & b_2 & & \\ & \ddots & \ddots & \ddots & \\ & & b_{n-3} & a_{n-2} & b_{n-2} \\ & & & b_{n-2} & a_{n-1} \end{bmatrix} \begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \vdots \\ \sigma_{n-2} \\ \sigma_{n-1} \end{bmatrix} = 6 \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_{n-2} \\ d_{n-1} \end{bmatrix}$$

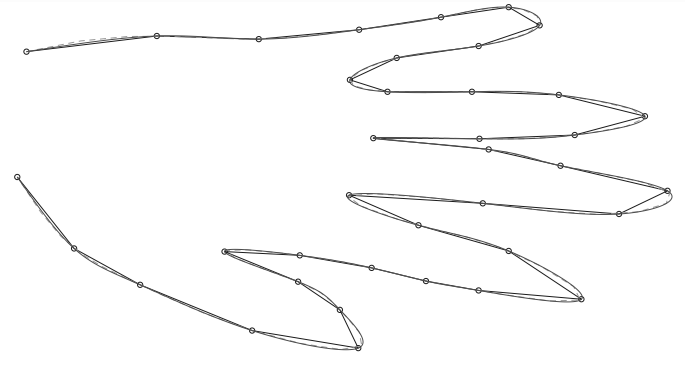
donde $a_k = 2(h_{k-1} + h_k)$, $d_k = \dfrac{y_{k+1} - y_k}{h_k} - \dfrac{y_k - y_{k-1}}{h_{k-1}}, k = 1, \ldots, n-1$, $b_k = h_k$, $k = 1, \ldots, n-2$ y σ_i son los coeficientes de los polinomios, $\sigma_0 = \sigma_n = 0$. En este caso debemos solucionar dos sistemas de este tipo, uno para $x(t)$ y otro para $y(t)$, con lo que vamos a tomar un tamaño de paso uniforme $h = 1$ por lo tanto la matriz se vuelve la siguiente

$$\begin{bmatrix} 4 & 1 & & & \\ 1 & 4 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 4 & 1 \\ & & & 1 & 4 \end{bmatrix}$$

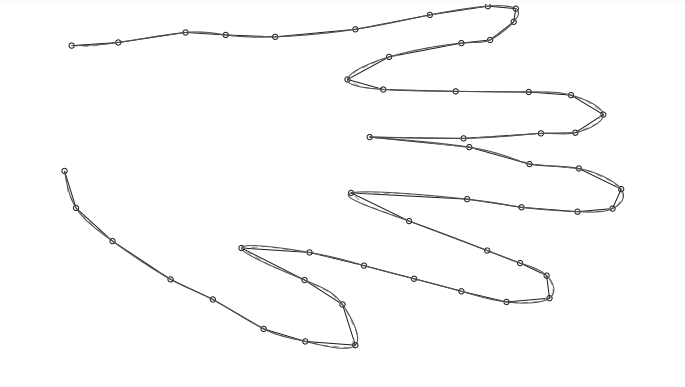
por lo que implementamos el algoritmo de Thomas para matrices tridiagonales para encontrar

los coeficientes de manera óptima, posteriormente generamos los polinomios con estos coeficientes y graficamos la curva paramétrica y finalmente estimamos el área con las manos de los miembros de nuestro grupo. El código final que implementamos lo adjuntamos en un cuaderno de Jupyter. Obtuvimos los siguientes resultados, en donde las lineas puntadas son la solución de Matlab, se evidencia que el código es bueno ya que su diferencia es muy pequeña.

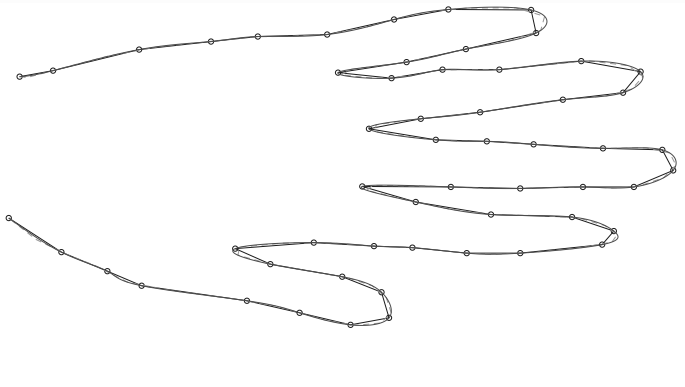
La primera mano con 37 puntos



El área aproximada fue 0.2479, como se observa la precisión es mejorable por lo que de ahora en adelante usaremos más puntos, observe con la misma mano la diferencia al escoger más puntos



y obtenemos un área aproximada de 0.2533, es de esperar que en los parte donde los dedos se curvan se necesiten de más puntos. Para la segunda mano obtuvimos



y un área de 0.3121. Alguno de ustedes hágalo con su mano porfa.

5. Problema 5: Integración Numérica

Se tiene la integral

$$I = \int_0^1 \frac{4}{1+x^2} dx = \pi.$$

- Use las reglas compuestas del punto medio, del trapecio y de Simpson para aproximar I para varios tamaños de paso de integración $h_n = 1/n$, $n = 10, 50, 100, 250, 500, 1000, 1500, 2000$. Grafique el logaritmo del error absoluto versus n para cada paso. Describa el efecto de redondeo de los errores cuando $h \to 0$.
- Implemente el método de integración de Romberg para calcular I. Grafique el logaritmo del error en los términos diagonales en la tabla de extrapolación versus $\log h$. Verifique sus resultados con la teoría.