MAT237 Cálculo Numérico

Manuel Loaiza Vasquez

Septiembre 2021

Pontificia Universidad Católica del Perú
Lima, Perú
manuel.loaiza@pucp.edu.pe

Solucionario de la Práctica Calificada 1 del curso Cálculo Numérico de la especialidad de Matemáticas de la Facultad de Ciencias e Ingeniería dictado por el profesor Rubén Agapito durante el ciclo $2021-2$.

1. Use la regla de la cadena para obtener los números de condicionamiento de las siguientes funciones:

a. $f(x) = \cos(2\pi x)$

Solución. Primero probemos el siguiente lema:

Lema 1. *Sean* $f : \mathbb{R} \to \mathbb{R}$ *y* $g : \mathbb{R} \to \mathbb{R}$ *funciones continuas diferenciables y* $h : \mathbb{R} \to \mathbb{R}$ *con* $h(x) = f(g(x))$*, luego*

$$\kappa_h(x) = \kappa_f(g(x)) \cdot \kappa_g(x).$$

Prueba. Aplicamos la fórmula de condicionamiento a h y la regla de la cadena

$$\begin{aligned}
\kappa_h(x) &= \left| x \cdot \frac{h'(x)}{h(x)} \right| \\
&= \left| x \cdot \frac{f'(g(x))g'(x)}{f(g(x))} \right| \\
&= \left| g(x) \cdot \frac{f'(g(x))}{f(g(x))} \right| \cdot \left| x \cdot \frac{g'(x)}{g(x)} \right| \\
&= \kappa_f(g(x)) \cdot \kappa_g(x).
\end{aligned}$$

obteniendo la expresión buscada. □

Sean $g : \mathbb{R} \to \mathbb{R}$ y $h : \mathbb{R} \to \mathbb{R}$ con $g(x) = cos(x)$ y $h(x) = 2\pi x$. Tenemos que $f = g \circ h$, por lo que

$$\begin{aligned}
\kappa_f(x) &= \kappa_g(h(x)) \cdot \kappa_h(x) \\
&= \left| 2\pi x \cdot \frac{-\sin(2\pi x)}{\cos(2\pi x)} \right| \cdot \left| x \cdot \frac{2\pi}{2\pi x} \right| \\
&= 2\pi |x \tan(2\pi x)|.
\end{aligned}$$

b. $f(x) = e^{-x^2}$

Solución. Sean $g : \mathbb{R} \to \mathbb{R}$ y $h : \mathbb{R} \to \mathbb{R}$ con $g(x) = e^x$ y $h(x) = -x^2$. Tenemos que $f = g \circ h$, por lo que

$$\begin{aligned}\kappa_f(x) &= \kappa_g(h(x)) \cdot \kappa_h(x) \\ &= \left| -x^2 \cdot \frac{e^{-x^2}}{e^{-x^2}} \right| \cdot 2 \\ &= 2x^2.\end{aligned}$$

2. Suponga que f es una función con número de condicionamiento κ_f y que f^{-1} es su función inversa. Demuestre que el número de condicionamiento de f^{-1} satisface

$$\kappa_{f^{-1}(x)} = \frac{1}{\kappa_f(f^{-1}(x))}$$

provisto que el denominador es diferente de cero.

Prueba. Analicemos el producto $\kappa_f(f^{-1}(x)) \cdot \kappa_{f^{-1}}(x)$. De acuerdo a Lema 1

$$\kappa_f(f^{-1}(x)) \cdot \kappa_{f^{-1}}(x) = \kappa_{f \circ f^{-1}(x)}(x) = \kappa_x(x) = 1.$$

Ya que nos garantizan que la expresión de la izquierda es distinta de cero, pasamos a dividir esta y concluimos con la prueba obteniendo lo requerido. □

3. El polinomio $x^2 - 2x + 1$ tiene una raíz doble en $r = 1$.

a. Use MATLAB para hacer una tabla de las raíces de

$$x^2 - (2 + \varepsilon)x + 1,$$

para $\varepsilon = 10^{-n}$ con $n = 4, 6, \ldots, 12$.

Solución. Utilicemos el siguiente script en MATLAB y el comando `fzero`

```
clc;
fprintf('Roots of the polynomial x^2 - (2 + eps) x + 1\n');
for n = 4:2:12
    eps = 10^(-n);
    f = @(x) x^2 - (2 + eps) * x + 1;
    root = fzero(f, 1);
    fprintf('eps = %e, root = %.10f\n', eps, root);
end
```

para obtener la tabla de raíces

```
Roots of the polynomial x^2 - (2 + eps) x + 1
eps = 1.000000e-04, root = 0.9900498750
eps = 1.000000e-06, root = 0.9990004999
eps = 1.000000e-08, root = 0.9999000050
eps = 1.000000e-10, root = 0.9999900001
eps = 1.000000e-12, root = 0.9999989999
```

b. ¿Qué puede inferir de los resultados del inciso (a) sobre el número de condicionamiento de la raíz?

Solución. Consideremos el problema de hallar las raíces de $at^2 + bt + c$ cuando solo b varía y a y c son constantes. Derivamos implícitamente con respecto a b:

$$\begin{aligned}
\frac{d}{db}(at^2 + bt + c) &= 0 \\
2at\frac{dt}{db} + t + b\frac{dt}{db} &= 0 \\
\frac{dt}{db}(2at + b) &= -t \\
\frac{dt}{db} &= \frac{-t}{2at + b}.
\end{aligned}$$

Ahora calculamos

$$\kappa_t(b) = \left| b \cdot \frac{dt/db}{t} \right| = \left| \frac{b}{\sqrt{b^2 - 4ac}} \right| = \left| \frac{b}{a(t_1 - t_2)} \right|.$$

Para nuestro caso particular, tenemos $a = 1$ y $b = 2$, por lo que tendremos un gran número de condicionamiento si la diferencia entre las dos raíces de nuestra ecuación cuadrática es mucho menor que 2. Con el comando `fzero` hemos encontrado una de las dos raíces alrededor de 1 y notamos que las dos raíces son muy cercanas (si en vez de usar el comando usábamos la fórmula general era evidente) respecto a cómo hemos perturbado el coeficiente lineal, por lo que pasamos de tener dos raíces iguales a 1, a dos raíces que distan al menos 10^{-3} casi simétricamente de 1 cuando nuestra perturbación es de tan solo 10^{-4}, pero cuando hemos hecho la perturbación más pequeña, hemos estado más cerca de 1, lo cual es algo bueno a pesar de que nuestras raíces sean muy cercanas y un poco fuera de lo intuitivo respecto al análisis hecho, lo cual tiene sentido ya que el número de condicionamiento al final se comporta como una cota y al estar dividiendo entre números muy pequeños se hace difícil de obtener buenos estimados.

4. Use el método de bisección para encontrar dos números reales x, con seis cifras de exactitud, que hagan que el determinante de la matriz

$$A = \begin{pmatrix} 1 & 2 & 3 & x \\ 4 & 5 & x & 6 \\ 7 & x & 8 & 9 \\ x & 10 & 11 & 12 \end{pmatrix}$$

sea igual a 1000. Para cada solución, calcule el determinante correspondiente y reporte cuántos decimales correctos (después del punto decimal) tiene el determinante cuando su solución x es usada.

Solución. Primero, necesitamos definir una función objetivo $f : \mathbb{R} \to \mathbb{R}$ con

$$f(x) = \det(A(x)) - 1000$$

reduciendo así nuestro problema a hallar las raíces de esta. Esbocemos f en el intervalo $[-20, 20)$ para poder hacernos una idea de en qué intervalos la función es monótona y tiene raíces para poder aplicar búsqueda binaria.

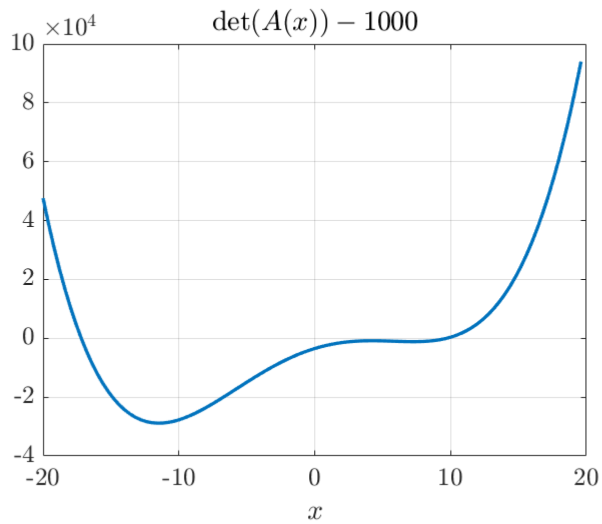

Figure 1: Esbozo de f

De la gráfica observamos que una raíz se encuentra en el intervalo $[-20, -15]$ y la otra en el intervalo $[9, 10]$.

- Aplicamos búsqueda binaria en el intervalo $[-20, -15]$, obteniendo la raíz aproximada -17.188498. Evaluamos $f(-17.188498) = -0.000853$,cuyo valor absoluto al compararse con cero tiene tres decimales correctos.

- Aplicamos búsqueda binaria en el intervalo $[9, 10]$, obteniendo la raíz aproximada 9.708299. Evaluamos $f(9.708299) = -0.000230$,cuyo valor absoluto al compararse con cero tiene tres decimales correctos.

Lo descrito previamente fue conseguido vía el siguiente script:

```
clc;

set(0,'defaulttextInterpreter','latex')
set(0, 'defaultAxesTickLabelInterpreter','latex');
set(0, 'defaultLegendInterpreter','latex');

% For images
set(0, 'defaultAxesFontSize', 15)
set(0, 'defaultLineLineWidth', 2.0);

% Statement matrix
```

```
A = @(x) [
    1, 2, 3, x;
    4, 5, x, 6;
    7, x, 8, 9;
    x, 10, 11, 12];

% Objective function
f = @(x) det(A(x)) - 1000;

% First, I plot some points to find an interval in which I have to do
% the binary search
test_size = 100;
x_axis = zeros(1, test_size);
y_axis = zeros(1, test_size);

for i = 1:test_size
    x_axis(i) = -20 + 0.4 * (i - 1);
    y_axis(i) = f(x_axis(i));
end

plot(x_axis, y_axis)
grid on;

roots = zeros(1, 2);
% According to my plot, I will choose two intervals
% First interval: [-20, -15]
roots(1) = binary_search(-20, -15, f, 1e-6);
% Second interval: [9, 10]
roots(2) = binary_search(9, 10, f, 1e-6);

fprintf('roots = [%.6f, %.6f]', roots(1), roots(2));
fprintf('values = [%.6f, %.6f]', f(roots(1)), f(roots(2)));

% Returns a root of the function f in the interval [l, r] in case it is
% found. Otherwise, it returns the midpoint of the interval, which is
% small enough according to our theory.
% Usage: root = binary_search(left, right, f, eps);

function root = binary_search(left, right, f, eps)
    root = NaN;
    if f(left) * f(right) > 0
        return
```

```
    end
    steps = round((log(right - left) - log(eps)) / log(2)) + 1;
    for step = 1:steps
        mid = left + (right - left) / 2;
        if f(left) * f(mid) > 0
            left = mid;
        elseif f(left) * f(mid) < 0
            right = mid;
        else
            root = mid;
            return
        end
    end
    root = left + (right - left) / 2;
end
```

5. Encuentre tres formas diferentes $g(x)$ para encontrar raíces con seis cifras de exactitud de $f(x) = 0$ por el método de iteración de punto fijo. Cada ecuación $f(x) = 0$ tiene tres raíces. Encuentre más $g(x)$ si fuese necesario hasta que todas las raíces sean encontradas por el método. Para cada iteración convergente, estime el valor de $|g'(r)|$ en base a los errores e_{k+1}/e_k y compárelo con el valor $|g'(r)|$, donde r es la raíz aproximada obtenida y $g'(x)$ es calculada analíticamente.

Solución. Modifiquemos el algoritmo clásico que teníamos para poder mostrar el error en cada iteración y el ratio entre los errores de dos iteraciones consecutivas. Para esto, primero he resuelto el problema sin calcular los errores porque aún no conocía el valor de r, el candidato a raíz, para cada g asociado a un f. Una vez resuelto el problema, he añadido el parámetro r a la función para poder generar la tabla con todo lo solicitado, dándole como input los valores de r hallados previamente.

```
% Returns an approximate solution to f(x) = x with a tolerance of eps
% and at most max_steps steps (to prevent infinite loop)
% Usage: root = fixed_point_iteration(f, guess, eps, max_steps);

function x = fixed_point_iteration(g, guess, eps, max_steps, r)
  % Base case
  x = guess;
  fprintf('step: %d, x: %.6f, g: %.6f, e_i = %.6f\n', 0, x, g(x), abs(x - r));
  steps = 0;
  prev_e = abs(x - r);
```

```matlab
    while abs(x - g(x)) >= eps && steps < max_steps
        steps = steps + 1;
        x = g(x);
        e = abs(x - r);
        fprintf('step: %d, x: %.6f, g: %.6f, e_i = %.6f, e_i / e_{i - 1} = %.6f\n', steps, x, g(x), e, e / prev_e);
        prev_e = e;
    end
end
```

a. $f(x) = 2x^3 - 6x - 1$

Solución. Utilicemos el siguiente script en MATLAB para hallar las tres raíces aproximadas de f utilizando el método de iteración de punto fijo

```matlab
clc;

% We want six digits of accuracy
eps = 1e-6;
max_steps = 100;

fprintf('Roots of f(x) = 2x^3-6x-1\n');

fprintf('\n');
g = @(x) (2 * x^3 - 1) / 6;
root = fixed_point_iteration(g, 1, eps, max_steps, -0.168254);
fprintf('First root: %.6f\n', root);

fprintf('\n');
g = @(x) ((6 * x + 1) / 2)^(1/3);
root = fixed_point_iteration(g, 2, eps, max_steps, 1.810038);
fprintf('Second root: %.6f\n', root);

fprintf('\n');
g = @(x) (4 * x^3 + 1) / (6 * x^2 - 6);
root = fixed_point_iteration(g, -2, eps, max_steps, -1.641784);
fprintf('Third root: %.6f\n', root);
```

obteniendo como salida

```
Roots of f(x) = 2x^3-6x-1

step: 0, x: 1.000000, g: 0.166667, e_i = 1.168254
step: 1, x: 0.166667, g: -0.165123, e_i = 0.334921, e_i / e_{i - 1} = 0.286685
```

```
step: 2, x: -0.165123, g: -0.168167, e_i = 0.003131, e_i / e_{i - 1} = 0.009347
step: 3, x: -0.168167, g: -0.168252, e_i = 0.000087, e_i / e_{i - 1} = 0.027661
step: 4, x: -0.168252, g: -0.168254, e_i = 0.000002, e_i / e_{i - 1} = 0.023786
step: 5, x: -0.168254, g: -0.168254, e_i = 0.000000, e_i / e_{i - 1} = 0.161229
First root: -0.168254

step: 0, x: 2.000000, g: 1.866256, e_i = 0.189962
step: 1, x: 1.866256, g: 1.827037, e_i = 0.056218, e_i / e_{i - 1} = 0.295941
step: 2, x: 1.827037, g: 1.815212, e_i = 0.016999, e_i / e_{i - 1} = 0.302379
step: 3, x: 1.815212, g: 1.811616, e_i = 0.005174, e_i / e_{i - 1} = 0.304354
step: 4, x: 1.811616, g: 1.810519, e_i = 0.001578, e_i / e_{i - 1} = 0.304953
step: 5, x: 1.810519, g: 1.810185, e_i = 0.000481, e_i / e_{i - 1} = 0.305116
step: 6, x: 1.810185, g: 1.810083, e_i = 0.000147, e_i / e_{i - 1} = 0.305101
step: 7, x: 1.810083, g: 1.810052, e_i = 0.000045, e_i / e_{i - 1} = 0.304886
step: 8, x: 1.810052, g: 1.810042, e_i = 0.000014, e_i / e_{i - 1} = 0.304128
step: 9, x: 1.810042, g: 1.810039, e_i = 0.000004, e_i / e_{i - 1} = 0.301617
step: 10, x: 1.810039, g: 1.810038, e_i = 0.000001, e_i / e_{i - 1} = 0.293259
Second root: 1.810039

step: 0, x: -2.000000, g: -1.722222, e_i = 0.358216
step: 1, x: -1.722222, g: -1.647363, e_i = 0.080438, e_i / e_{i - 1} = 0.224552
step: 2, x: -1.647363, g: -1.641813, e_i = 0.005579, e_i / e_{i - 1} = 0.069360
step: 3, x: -1.641813, g: -1.641784, e_i = 0.000029, e_i / e_{i - 1} = 0.005273
step: 4, x: -1.641784, g: -1.641784, e_i = 0.000000, e_i / e_{i - 1} = 0.016033
Third root: -1.641784
```

Ahora analicemos analíticamente cada una de las funciones g:

- $g(x) = (2x^3 - 1)/6$. Tenemos $g'(x) = x^2$, por lo que

  $$|g'(-0.168254)| \approx 0.028309408516 < 1$$

  y el MIPF converge localmente.

- $g(x) = ((6x + 1)/2)^{1/3}$. Tenemos $g'(x) = 1/(3x + 1/2)^{2/3}$, por lo que

  $$|g'(1.810039)| \approx 0.305228 < 1$$

  y el MIPF converge localmente.

- $g(x) = (4x^3+1)/(6x^2-6)$. Tenemos $g'(x) = (x(2x^3-6x-1))/(2(x^2-1)^2)$, por lo que

  $$|g'(-1.641784)| \approx 9.15176 \times 10^{-7} < 1$$

  y el MIPF converge localmente.

b. $f(x) = e^{x-2} + x^3 - x$

Solución. Utilicemos el siguiente script en MATLAB para hallar las tres raíces aproximadas de f utilizando el método de iteración de punto fijo

```matlab
clc;

% We want six digits of accuracy
eps = 1e-6;
max_steps = 100;

fprintf('Roots of f(x) = e^(x - 2) + x^3 - x\n');

fprintf('\n');
g = @(x) exp(x - 2) + x^3;
root = fixed_point_iteration(g, 0.16, eps, max_steps, 0.163822);
fprintf('First root: %.6f\n', root);

fprintf('\n');
g = @(x) (x - exp(x - 2))^(1 / 3);
root = fixed_point_iteration(g, 0.78, eps, max_steps, 0.788940);
fprintf('Second root: %.6f\n', root);

fprintf('\n');
g = @(x) (-exp(x - 2) + 2 * x^3) / (3 * x^2 - 1);
root = fixed_point_iteration(g, -1.2, eps, max_steps, -1.023482);
fprintf('Third root: %.6f\n', root);
```

obteniendo como salida

```
Roots of f(x) = e^(x - 2) + x^3 - x

step: 0, x: 0.160000, g: 0.162913, e_i = 0.003822
step: 1, x: 0.162913, g: 0.163605, e_i = 0.000909, e_i / e_{i - 1} = 0.237722
step: 2, x: 0.163605, g: 0.163770, e_i = 0.000217, e_i / e_{i - 1} = 0.239217
step: 3, x: 0.163770, g: 0.163810, e_i = 0.000052, e_i / e_{i - 1} = 0.238964
step: 4, x: 0.163810, g: 0.163819, e_i = 0.000012, e_i / e_{i - 1} = 0.236349
step: 5, x: 0.163819, g: 0.163822, e_i = 0.000003, e_i / e_{i - 1} = 0.224870
step: 6, x: 0.163822, g: 0.163822, e_i = 0.000000, e_i / e_{i - 1} = 0.172959
First root: 0.163822

step: 0, x: 0.780000, g: 0.785558, e_i = 0.008940
step: 1, x: 0.785558, g: 0.787666, e_i = 0.003382, e_i / e_{i - 1} = 0.378245
step: 2, x: 0.787666, g: 0.788462, e_i = 0.001274, e_i / e_{i - 1} = 0.376632
step: 3, x: 0.788462, g: 0.788761, e_i = 0.000478, e_i / e_{i - 1} = 0.375660
step: 4, x: 0.788761, g: 0.788874, e_i = 0.000179, e_i / e_{i - 1} = 0.374323
step: 5, x: 0.788874, g: 0.788916, e_i = 0.000066, e_i / e_{i - 1} = 0.371217
step: 6, x: 0.788916, g: 0.788932, e_i = 0.000024, e_i / e_{i - 1} = 0.362990
step: 7, x: 0.788932, g: 0.788938, e_i = 0.000008, e_i / e_{i - 1} = 0.340100
```

```
step: 8, x: 0.788938, g: 0.788940, e_i = 0.000002, e_i / e_{i - 1} = 0.270405
step: 9, x: 0.788940, g: 0.788941, e_i = 0.000000, e_i / e_{i - 1} = 0.014544
Second root: 0.788940

step: 0, x: -1.200000, g: -1.053242, e_i = 0.176518
step: 1, x: -1.053242, g: -1.024060, e_i = 0.029760, e_i / e_{i - 1} = 0.168593
step: 2, x: -1.024060, g: -1.023470, e_i = 0.000578, e_i / e_{i - 1} = 0.019436
step: 3, x: -1.023470, g: -1.023482, e_i = 0.000012, e_i / e_{i - 1} = 0.021482
step: 4, x: -1.023482, g: -1.023482, e_i = 0.000000, e_i / e_{i - 1} = 0.038756
Third root: -1.023482
```

Ahora analicemos analíticamente cada una de las funciones g:

- $g(x) = e^{x-2} + x^3$. Tenemos $g'(x) = e^{x-2} + 3x^2$, por lo que

$$|g'(0.163822)| \approx 0.239939 < 1$$

  y el MIPF converge localmente.

- $g(x) = (x - e^{x-2})^{1/3}$. Tenemos $g'(x) = (1 - e^{x-2})/(3(x - e^{x-2})^{2/3})$, por lo que

$$|g'(0.788940)| \approx 0.376011 < 1$$

  y el MIPF converge localmente.

- $g(x) = (-e^{x-2} + 2x^3)/(3x^2 - 1)$. Tenemos

$$g'(x) = \frac{6x^2 - e^{x-2}}{3x^2 - 1} - \frac{6x(2x^3 - e^{x-2})}{(3x^2 - 1)^2},$$

  por lo que

$$|g'(-1.023482)| \approx 0.0226986 < 1$$

  y el MIPF converge localmente.