Ejercicios teóricos para el examen extraordinario de Métodos Numéricos I

Aritmética con redondeo

1. Ejemplo de violación de la ley asociativa de la adición en la aritmética con redondeo al más cercano. Consideremos los números escritos en la notación científica decimal con dos dígitos después del punto flotante. Denotemos por $\oplus, \ominus, \otimes, \oslash$ a las operaciones aritméticas con redondeo al más cercano. Por ejemplo,

$$(9.83 \cdot 10^{-2}) \oplus (7.54 \cdot 10^{-3}) = \text{redond}(1.0584 \cdot 10^{-1}) = 1.06 \cdot 10^{-1}.$$

Encuentre números a, b, c pertenecientes a esta aritmética (es decir, escritos en la notación científica decimal con dos dígitos después del punto flotante) tales que

$$(a \oplus b) \oplus c \neq a \oplus (b \oplus c).$$

Multiplicación y división de los polinomios

2. Multiplicación de un polinomio por un binomio. Escriba un algoritmo que calcule los coeficientes del producto $(x + b)P(x)$, donde $P(x) = a_0 + a_1 x + \ldots + a_{n-1}x^{n-1}$ es un polinomio y b es un número. Calcule el número de las operaciones de multiplicación en este algoritmo.

Entrada: números $a_0, a_1, \ldots, a_{n-1}$ y b.

Salida: coeficientes $c_0, c_1, \ldots, c_n$ del polinomio $(x + b)P(x)$.

3. Algoritmo de Horner. Escriba el *algoritmo de Horner* (también llamado la *división sintética* o la *regla de Ruffini*) que evalue el polinomio $P(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \ldots + a_1 x + a_0$ en el punto b. Calcule el número de las multiplicaciones en este algoritmo.

Entrada: números $a_0, a_1, \ldots, a_{n-1}$ y b.

Salida: número $P(b)$.

Solución de sistemas de ecuaciones lineales

4. Solución de un sistema de ecuaciones lineales con una matriz triangular inferior. Escriba el algoritmo de la *sustitución hacia adelante* que se usa para resolver un sistema de ecuaciones lineales $Lx = b$, donde L es una matriz triangular inferior de orden n con entradas diagonales no nulas y b es un vector de longitud n. Calcule el número de las operaciones de multiplicación y división.

Entrada: matriz L y vector b.

Salida: vector x.

5. Solución de un sistema de ecuaciones lineales con una matriz triangular superior. Escriba el algoritmo de la *sustitución hacia atrás* que se usa para resolver un sistema de ecuaciones lineales $Ux = b$, donde U es una matriz triangular superior de orden n con entradas diagonales no nulas y b es un vector de longitud n. Calcule el número de las operaciones de multiplicación y división.

Entrada: matriz U y vector b.

Salida: vector x.

6. Factorización LU. Escriba el algoritmo de la *factorización LU* de una matriz cuadrada A. Puede suponer que A es invertible (no singular) y posée una factorización LU. Calcule el número de las operaciones de multiplicación y división en este algoritmo.

Entrada: matriz cuadrada A de orden n.

Salida: matrices L y U de orden n tales que $LU = A$, U es triangular superior, L es triangular inferior y las entradas diagonales de L son iguales a 1.

7. Métodos iterativos de Jacobi y de Gauss-Seidel para resolver sistemas de ecuaciones lineales. Escriba las fórmulas de los métodos de Jacobi y de Gauss-Seidel. Explique la diferencia entre estos. Calcule el número de las operaciones de multiplicación y división en una iteración del algoritmo de Jacobi. Denote por n al orden de la matriz del sistema.

Solución de ecuaciones no lineales

8. Métodos de la secante y de la regla falsa. Explique la idea de los los métodos de la secante y de la regla falsa los cuales se utilizan para aproximar un cero de la función dada f. No mencione las condiciones de salida, escriba solamente las fórmulas o reglas que se usan en estos métodos para construir los puntos nuevos a partir de los puntos anteriores. Por ejemplo, para el método de la secante es suficiente expresar el punto x_n a través de los puntos anteriores. Explique la diferencia entre estos dos métodos.

9. Justificación del algoritmo babilónico para sirve para calcular la raíz cuadrada positiva de un número positivo. Sea a un número positivo. Calcule el punto fijo positivo $p > 0$ de la función

$$g(x) = \frac{1}{2}\left(x + \frac{a}{x}\right).$$

Muestre que $g'(p) = 0$ y $g'(x) > 0$ para todo $x > p$. Encuentre un número $k \in (0, 1)$ tal que $g'(x) \leq k$ para todo $x > p$.

10. Algoritmo para calcular la raíz cúbica positiva de un número positivo. Sea a un número positivo y sea $p = \sqrt[3]{a}$. Construya una función $g\colon (0, +\infty) \to (0, +\infty)$ tal que $g(p) = p$ y $g'(p) = 0$. Sugerencia: escriba la fórmula iterativa que da el método de Newton-Raphson aplicado a la función $f(x) = x^3 - a$.

Interpolación polinomial

11. Construcción del polinomio básico de Lagrange. Escriba un algoritmo para calcular los coeficientes del *polinomio básico de Lagrange*. Así llamamos al polinomio que toma valor 1 en el punto x_k y valor 0 en los puntos x_j, $j \in \{1, \ldots, n\} \backslash \{k\}$. Calcule el número de las operaciones de multiplicación y división en este algoritmo.

> Entrada: números diferentes por pares $x_1, x_2, \ldots, x_n$, un índice $k \in \{1, \ldots, n\}$.
>
> Salida: coeficientes del polinomio de grado $n-1$ que toma valor 1 en el punto x_k y toma valor 0 en los puntos x_j, $j \in \{1, \ldots, n\} \setminus \{k\}$.

12. Tabla de las diferencias divididas. Escriba un algoritmo para calcular la *tabla de las diferencias divididas*. Calcule el número de las operaciones de división en este algoritmo.

> Entrada: números diferentes por pares $x_1, x_2, \ldots, x_n$, números $y_1, y_2, \ldots, y_n$.
>
> Salida: diferencias divididas $d_1 = P[x_1], d_2 = P[y_1, y_2], \ldots, d_n = P[y_1, \ldots, y_n]$, donde P es una función tal que $P(x_i) = y_i$ para todo $i \in \{1, \ldots, n\}$.

Observación: aquí la función P sirve para denotar a las diferencias divididas, en realidad las últimas dependen solamente de los puntos $x_1, x_2, \ldots, x_n$ y de los valores $y_1, y_2, \ldots, y_n$.

13. Construcción del polinomio interpolante con la fórmula de Newton. Escriba un algoritmo que calcule los coeficientes del polinomio interpolante usando la *fórmula de Newton*. Suponemos que ya están dadas las diferencias divididas necesarias. Calcule el número de las operaciones de multiplicación en este algoritmo.

> Entrada: números diferentes por pares $x_1, x_2, \ldots, x_n$, números $d_1, d_2, \ldots, d_n$.
>
> Salida: coeficientes del polinomio P que tenga las diferencias divididas dadas:
>
> $$P[x_1] = d_1, \qquad P[x_1, x_2] = d_2, \qquad \ldots, \qquad P[x_1, \ldots, x_n] = d_n.$$