[問題] V-B

5×5 の行列 A および 5 次元の縦ベクトル $\boldsymbol{b}$ がそれぞれ以下のように与えられたとする。

$$A = \begin{bmatrix} 14 & 14 & -9 & 3 & -5 \\ 14 & 52 & -15 & 2 & -32 \\ -9 & -15 & 36 & -5 & 16 \\ 3 & 2 & -5 & 47 & 49 \\ -5 & -32 & 16 & 49 & 79 \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} -15 \\ -100 \\ 106 \\ 329 \\ 463 \end{bmatrix}.$$

この時行列 A に対してガウス-ザイデル法を実行するプログラムを作成し連立一次方程式

$A\boldsymbol{x} = \boldsymbol{b}$ の解 $\boldsymbol{x} = [x_1, x_2, x_3, x_4, x_5]^{\mathsf{T}}$ を有効数字 10 進 3 桁まで求め 4 桁を四捨五入して答えよ。

作成したプログラムも提出すること。プログラミング言語は問わない。

[略解] V-B

$$\boldsymbol{x} = \begin{bmatrix} -4.01 \times 10^{-14} \\ 1.00 \\ 2.00 \\ 3.00 \\ 4.00 \end{bmatrix}$$

1つ目の要素はほぼ 0

[手法] 疎行列を扱うための安い方法

ガウス-ザイデル法 (Gauss-Seidel iteration)

Input: A, $k_{\max}$, $\boldsymbol{x}$ (初期ベクトル)
Output: $\boldsymbol{x}$

1: **for** $k = 1, 2, \ldots, k_{\max}$ **:**
2: **for** $i = 1, \ldots, n$ **:**
3: $sum = 0$
4: **for** $j = 1, \ldots, i-1$ **:**
5: $sum \leftarrow sum + a_{ij}x_j$
6: **for** $j = i+1, \ldots, n$ **:**
7: $sum \leftarrow sum + a_{ij}x_j$
8: $x_i = (-sum + b_i)/a_{ii}$

※ 疎行列を安く扱うにはさらに行列 $A = (a_{ij})_{1 \le i \le n, 1 \le j \le n}$ を適した形で保存する必要があり、上記のアルゴリズムも非零の a_{ij} のみについての演算を行うように修正する必要がある。

[手法解説]

ガウス-ザイデル法 (Gauss-Seidel iteration)

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(-\sum_{j<i} a_{ij} x_j^{(k+1)} - \sum_{j>i} a_{ij} x_j^{(k)} + b_i \right), \quad 1 \le i \le n,\ 1 \le j \le n$$

- $n = 3$ の例
  - $x_1^{(k+1)} = \frac{1}{a_{11}} \left(-a_{12} x_2^{(k)} - a_{13} x_3^{(k)} + b_1 \right)$ ($a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1$ の変形)
  - $x_2^{(k+1)} = \frac{1}{a_{22}} \left(-a_{21} x_1^{(k+1)} - a_{23} x_3^{(k)} + b_2 \right)$ ($a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2$ の変形)
  - $x_3^{(k+1)} = \frac{1}{a_{33}} \left(-a_{31} x_1^{(k+1)} - a_{32} x_2^{(k+1)} + b_3 \right)$ ($a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3$ の変形)

[数値解析 第6回]

関数近似と補間 (1)

データ点間をつなぐ関数を速く求める

関数近似 (Approximation)

- 関数 $f(x)$ または点の集合をより性質の良い関数を用いた有限項の級数として近似的に表す方法
- 関数 $f(x)$ に対する関数近似の例
  - 多項式近似 $f(x) \sim p_n(x) = \sum_{i=0}^{n} a_n x^n$
    - ★ テイラー展開 $f(x) \sim \sum_{i=0}^{n} \frac{f^{(n)}(a)}{n!}(x-a)^n$
  - フーリエ級数展開 $f(x) \sim \frac{a_0}{2} + \sum_{i=1}^{n} (a_n \cos nx + b_n \sin nx)$
- 点の集合に対する関数近似の例
  - **多項式補間、最小 2 乗法**

補間 (Interpolation)

- 2 つのデータ点 (x_a, y_a), (x_b, y_b) の間の x_c $(x_a \leq x_c \leq x_b)$ での値 y_c を近似的に求める方法
  - データ点をつなぐ関数 $q(x)$ を**関数近似**で求め、x_c での近似値を $y_c \sim q(x_c)$ として求める
- 補間の例
  - **多項式補間** (例. ラグランジュ補間、ニュートン補間)
  - **区分的多項式補間** (例. スプライン補間)

多項式補間の問題設定

- **多項式補間**

問題設定 1 x 軸上の値の集合 $\{x_i,\ 0 \leq i \leq n\}$ と連続関数 $f(x)$ が与えられた時、$p_n(x_i) = f(x_i),\ 0 \leq i \leq n$ を満たす高々 n 次の多項式 $p_n(x)$ を求める問題 (**関数近似**の一種)

問題設定 2 x 軸上の値の集合 $\{x_i, 0 \leq i \leq n\}$ とそれぞれの x_i に対応するデータの値 $\{y_i,\ 0 \leq i \leq n\}$ が与えられた時、$p_n(x_i) = y_i,\ 0 \leq i \leq n$ を満たす高々 n 次の多項式 $p_n(x)$ を求める問題

[問題] VI-A

(1) 2つのデータ点 $(x_0, y_0) = (0, 1)$, $(x_1, y_1) = (1, 2)$ を通る1次の補間多項式 $p_1(x) = \alpha_1 x + \alpha_0$ を以下の3通りの方法により求めよ。(a) 連立方程式を解く方法、(b) ラグランジュ補間を用いる方法、(c) ニュートン補間を用いる方法。

(2) 問1の2点に $(x_2, y_2) = (4, 2)$ を加えた3点を通る2次の補間多項式 $p_2(x) = \beta_2 x^2 + \beta_1 x + \beta_0$ を問1と同様の3通りの方法により求めよ。

[略解] VI-A

(1) $p_1(x) = x + 1$

(a) $p_1(x_0) = y_0$, $p_1(x_1) = y_1$ を連立させて解く、(b) $p_1(x) = y_0 L_0^{(1)} + y_1 L_1^{(1)}$ を計算、(c) $p_1(x) = a_0 + a_1\, w_0(x)$ を計算。

(2) $p_2(x) = -\frac{1}{4}x^2 + \frac{5}{4}x + 1$

(a) $p_1(x_0) = y_0$, $p_1(x_1) = y_1$, $p_1(x_2) = y_2$ を連立させて解く、

(b) $p_2(x) = y_0 L_0^{(2)} + y_1 L_1^{(2)} + y_2 L_2^{(2)}$ を計算、(c) $p_2(x) = p_1(x) + a_2\, w_1(x)$ を計算。

[手法解説]

ラグランジュ補間 (Lagrange interpolation)

x_i に対応する n 次式 $L_i^{(n)}(x) = \prod_{k=0, k\neq i}^{n} \frac{x - x_k}{x_i - x_k}$ を $n+1$ 個足し合わせることで補間多項式 $p_n(x) = \sum_{k=0}^{n} y_k L_k^{(n)}(x)$ を構成する方法

- 点 (x_j, y_j) を通ることを確認
  - $L_k^{(n)}(x_j) = \delta_{kj} = \begin{cases} 1 & (k = j) \\ 0 & (k \neq j) \end{cases}$ より $p_n(x_j) = \sum_{k=0}^{n} y_k L_k^{(n)}(x_j) = \sum_{k=0}^{n} y_k \delta_{kj} = y_j$
- 乗除算の回数は $n^2 + \mathcal{O}(n)$
  - 連立方程式を LU 分解 ($\frac{1}{3}n^3 + \mathcal{O}(n^2)$) で解くよりも速い (補間多項式の係数をすべて求める場合は $\frac{1}{2}n^3 + \mathcal{O}(n^2)$ で LU 分解より遅い)

[手法] ニュートン補間 (1)

ニュートン補間 (差分商の計算)

Newton interpolation via divided difference coefficients (construction)

Input: $n, x_0, x_1, \ldots, x_n, y_0, y_1, \ldots, y_n$

Output: $a_0, a_1, \ldots, a_n$

1: $a_0 \leftarrow y_0$
2: **for** $k = 1, 2, \ldots, n$ **:**
3: $\quad w \leftarrow 1$
4: $\quad p \leftarrow 0$
5: $\quad$ **for** $j = 0, 1, \ldots, k-1$ **:**
6: $\quad\quad p \leftarrow p + a_j w$
7: $\quad\quad w \leftarrow w\,(x_k - x_j)$
8: $\quad a_k \leftarrow (y_k - p)/w$

乗除算の回数は n^2

[手法] ニュートン補間 (2)

ニュートン補間 ($x = \xi$ での値 $p_n(\xi)$ の評価)

Newton interpolation via divided difference coefficients (evaluation)

Input: $n, \xi, x_0, x_1, \ldots, x_n, a_0, a_1, \ldots, a_n$

Output: p_ξ ($x = \xi$ での補間多項式の値)

1: $p_\xi \leftarrow a_n$

2: **for** $k = n-1, n-2, \ldots, 0$ **:**

3: $\quad p_\xi \leftarrow a_k + p_\xi\,(\xi - x_k)$ **(ホーナー法)**

乗除算の回数は n

[手法解説]

差分商を用いたニュートン補間 (Newton interpolation via divided difference coefficients)

通過する点 (x_k, y_k) を 1 点ずつ増やしながら n 次補間多項式 $p_n(x)$ を構成する方法。$w_k(x) = \prod_{i=0}^{k}(x - x_i)$ としたとき、**差分商** $a_{k+1} = \dfrac{y_{k+1} - p_k(x_{k+1})}{w_k(x_{k+1})}$ を含む漸化式 $p_{k+1}(x) = p_k(x) + a_{k+1}\, w_k(x)$ を用いる。$p_0(x) \equiv a_0 = y_0$ とする。

- ホーナー法による書き換え
  - $$p_n(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \cdots + a_n(x - x_0)\cdots(x - x_{n-1})$$
    $$= a_0 + (x - x_0)(a_1 + (x - x_1)(a_2 + \cdots + (x - x_{n-1})a_n)\cdots)$$
- 乗除算の回数
  - 差分商の計算: n^2、$x = \xi$ での値 $p_n(\xi)$ の評価: n
  - 新たな点 (x_{n+1}, y_{n+1}) での差分商 a_{n+1} を求める: n

[問題] VI-B

$f(x) = e^x$, $x \in [-1, 1]$ とした時、点 $(x_0, f(x_0)), (x_1, f(x_1)), \ldots, (x_n, f(x_n))$ に対してニュートン補間を行い補間多項式 $p_n(x)$ を差分商を用いて表す場合を考える。

$x_0 = -1$, $x_n = 1$ とし $x_1, \ldots, x_{n-1}$ を区間 $[-1, 1]$ を n 等分するように取った時、$n = 2, 4, 8, 16, 32$ の場合の $p_n(x)$ について相対誤差 $E(x)$ を区間 $[-1, 1]$ を 500 等分する点 $(x = -1.0, -0.996, -0.992, \ldots, -0.004, 0.0, 0.004, \ldots, 0.992, 0.996, 1.0)$ で求め、$E(x)$ が最大となる x とその時の $E(x)$ の値をそれぞれ有効数字 10 進 3 桁で 4 桁目を四捨五入して答えよ。

作成したプログラムも提出すること。プログラミング言語は問わない。

おまけ

※ 配列のインデックスが 0 から始まるプログラミング言語用のアルゴリズム

[手法] 疎行列を扱うための安い方法

ガウス-ザイデル法 (Gauss-Seidel iteration)

※ 配列のインデックスが 0 から始まるプログラミング言語用

Input: A, $k_{\max}$, $\boldsymbol{x}$ (初期ベクトル)
Output: $\boldsymbol{x}$

```
1: for k = 1, 2, ..., k_max :
2:     for i = 0, ..., n − 1 :
3:         sum = 0
4:         for j = 0, 1, ..., i − 1 :
5:             sum ← sum + a_ij x_j
6:         for j = i + 1, i + 2, ..., n − 1 :
7:             sum ← sum + a_ij x_j
8:         x_i = (−sum + b_i)/a_ii
```