Metody numeryczne

Komentarz 1. zadania laboratoryjnego

Bartosz Smolarczyk

Poprawność implementacji i złożoność obliczeniowa

- `__init__(self, a:np.ndarray, b:np.ndarray, c:np.ndarray) -> dTridiag`

  Metoda zapisuje w tworzonym obiekcie klasy `dTridiag` wektory `a`, `b` oraz `c` jako `diagonal`, `hyperdiagonal` i `subdiagonal`, wykonując jednocześnie konwersję ich zawartości do typu `float`. Dodatkowo zapisywane są rozmiar diagonalii (`main_len` $= n$), sub- i hiperdiagonalii (`side_len`) oraz odległość wierszowa/kolumnowa `d` między główną diagonalą a subdiagonalą/hiperdiagonalą.
  Na koniec metoda wywołuje `__LU_decompose()`, która wyznacza rozkład LU (bez wyboru elementu dominującego, obliczamy go od razu aby być gotowym na wielokrotne wywołania `dTridiag.solve`). Rozkład jest zapisywany jako wektory `L_subdiag`, `U_diag` oraz `U_hyperdiag` ponieważ w macierzach L oraz U niezerowe są jedynie główne diagonale oraz odpowiednio sub- i hiperdiagonala (nie poświęcamy pamięci na diagonalę macierzy L złożoną z samych 1).

  Złożoność obliczeniowa: O(n) - liniowe zapisywanie wektorów diagonalii, liniowe obliczanie rozkładu LU.
  Złożoność pamięciowa: O(n) - wszystkie dane są trzymane wewnątrz kilku wektorów rozmiaru n.

- `dTridiag.dot(self, v: np.ndarray) -> np.ndarray`

  Metoda wykonuje konwersję wektora `v` do typu `float`. Następnie najpierw mnoży wartości diagonali przez wartości wektora `v`, a na koniec do iloczynu dodaje wyniki mnożeń sub- i hiperdiagonali przez odpowiednie fragmenty wektora `v`.

  Złożoność czasowa: O(n) - liniowe mnożenie elementów wektorów i ich dodawanie.
  Złożoność pamięciowa: O(n) - dodatkowa pamięć na wektor wynikowy rozmiaru n.

- `dTridiag.solve(self, y: np.ndarray) -> np.ndarray`

  Metoda wykonuje konwersję wektora `y` do typu `float`. Następnie rozwiązuje układy równań $Lz = y$ oraz $Ux = z$, ostatecznie zwracając szukany wektor x. Obliczenia wykonuje blokowo, rozwiązując d równań na raz (są to proste równania z jedną niewiadomą), później podstawia wyznaczone wartości do kolejnego bloku d równań z dwiema niewiadomymi sprowadzając je do równań z jedną niewiadomą i tak do rozwiązania całego układu.

  Złożoność czasowa: O(n) - dla każdego wiersza L oraz U wykonujemy pojedyncze operacje zmiennoprzecinkowe.
  Złożoność pamięciowa: O(n) - dodatkowa pamięć na wektor wynikowy rozmiaru n.