Wspólne informacje dotyczące testowych funkcji, ich danych wejściowych i wyjściowych oraz sposobu wymiany danych opisano w sekcji 7.

1 Utworzenie testowych danych wejściowych dla algorytmu Lanczos

Lst. 1. Matlab

```
1 function init_data(work_dir, g, a_prx, startvec_prx);
2 >>> init_data('../data', 10, 'A', 'startvec');
```

Parametry wejściowe:

- `g` – parametr funkcji `numgrid() (MATLAB)` określający rozmiar danych wejściowych
- `a_prx` – prefiks nazwy docelowego pliku z symetryczną macierzą A (o rozmiarze n)
- `startvec_prx` – prefiks nazwy docelowego pliku z wektorem startowym *startvec* (o rozmiarze n)

Dane wyjściowe:

- `<work_dir>/<a_prx>_n-<n>_g-<g>.csv` – plik CSV z macierzą symetryczną A o rozmiarze n i parametrze użytym dla `numgrid()` równym g.
- `<work_dir>/<startvec_prx>_n-<n>.csv` – plik CSV z losowym znormalizowanym wektorem startowym *startvec* o rozmiarze n.

2 Początkowa iteracja algorytmu Lanczos

Lst. 2. Matlab

```
1 function lancno_init(work_dir, m, a_name, startvec_name, a_vec_prx,
      b_vec_prx, anorm_prx);
2 >>> lancno_init('../data', 200, 'A_n-48_g-10', 'startvec_n-48', 'm-a',
      'm-b', 'm-anorm');
```

Lst. 3. C++

```
1 void lancno_init(string work_dir, uint32_t m, string a_name, string
      startvec_name, string a_vec_prx, string b_vec_prx, string anorm_prx,
      bool save_bin, bool save_csv, bool dbg);
2 >>> lancno_init("../data", 200, "A_n-48_g-10", "startvec_n-48", "c-a",
      "c-b", "c-anorm", true, true, false);
```

Parametry wejściowe:

- `m` – liczba iteracji

- `a_name` – nazwa pliku z symetryczną macierzą A (o rozmiarze n)
- `startvec_name` – nazwa pliku z wektorem startowym *startvec* (o rozmiarze n)
- `a_vec_prx` – prefiks nazwy docelowego pliku z wektorem a (rozmiar m), zawierającym przekątną macierzy T
- `b_vec_prx` – prefiks nazwy docelowego pliku z wektorem b (rozmiar m), zawierającym nad-/podprzekątną macierzy T
- `anorm_prx` – prefiks nazwy docelowego pliku z wartością *anorm* (skalar)

Dane wyjściowe:

- `<work_dir>/<a_vec_prx>_n-<n>_m-<m>.{csv|bin}` – plik z wektorem a (rozmiar m).
- `<work_dir>/<b_vec_prx>_n-<n>_m-<m>.{csv|bin}` – plik z wektorem b (rozmiar m).
- `<work_dir>/<anorm_prx>_n-<n>_m-<m>.{csv|bin}` – plik z wartością *anorm* (skalar).

3 Krok MRRR (Multiple Relatively Robust Representations for Tridiagonals)

Lst. 4. C++

```
1 void mrrr(string work_dir, string a_vec_name, string b_vec_name, string
      s_prx, string ritz_prx, bool save_bin, bool save_csv, bool dbg);
2 >>> mrrr("../data", "c-a_n-48_m-100", "c-b_n-48_m-100", "c-S", "c-ritz",
      true, true, false);
```

Parametry wejściowe:

- `a_vec_name` – nazwa pliku z wektorem a (rozmiar m)
- `b_vec_name` – nazwa pliku z wektorem b (rozmiar m)
- `s_prx` – prefiks nazwy docelowego pliku z macierzą S o rozmiarze m (**macierz ta nie jest symetryczna**)
- `ritz_prx` – prefiks nazwy docelowego pliku z wektorem *ritz* (rozmiar m)

Dane wyjściowe:

- `<work_dir>/<s_prx>_m-<m>.{csv|bin}` – plik z niesymetryczną macierzą S (rozmiar m).
- `<work_dir>/<ritz_prx>_m-<m>.{csv|bin}` – plik z wektorem *ritz* (rozmiar m).

4 Residual estimation and removing non-converged and spurious Ritz values

Lst. 5. Matlab

```
1 function rescon(work_dir, s_name, ritz_name, b_name, anorm_name,
      eps_name, s_prx, ritz_prx, lres_prx, cul_prx, idx_prx, k_prx);
2 >>> rescon('../data', 'c-S_m-200', 'c-ritz_m-200', 'm-b_n-48_m-200',
      'm-anorm_n-48_m-200', 'm-rescon-eps', 'm-rescon-S', 'm-rescon-ritz',
      'm-rescon-lres', 'm-rescon-cul', 'm-rescon-idx', 'm-rescon-k');
```

Lst. 6. C++

```
1 void rescon(string work_dir, string s_name, string ritz_name, string
      b_name, string anorm_name, string eps_name, string s_prx, string
      ritz_prx, string lres_prx, string cul_prx, string idx_prx, string
      k_prx, bool save_bin, bool save_csv, bool dbg);
2 >>> rescon("../data", "c-S_m-200", "c-ritz_m-200", "c-b_n-48_m-200",
      "c-anorm_n-48_m-200", "c-rescon-eps", "c-rescon-S", "c-rescon-ritz",
      "c-rescon-lres", "c-rescon-cul", "c-rescon-idx", "c-rescon-k", true,
      true, false);
```

Parametry wejściowe:

- `s_name` – nazwa pliku z niesymetryczną macierzą S (rozmiar m) otrzymaną w kroku MRRR
- `ritz_name` – nazwa pliku z wektorem *ritz* (rozmiar m) otrzymanym w kroku MRRR
- `b_name` – nazwa pliku z wektorem b (rozmiar m) otrzymanym w kroku LANCNO_INIT
- `anorm_name` – nazwa pliku z wartością *anorm* (skalar) otrzymaną w kroku LANCNO_INIT
- `eps_name` – nazwa pliku z wartością *eps* (skalar) – dla Matlaba jest to `2.2204e-16`
- `s_prx` – prefiks nazwy docelowego pliku z niesymetryczną macierzą S o rozmiarze m
- `ritz_prx` – prefiks nazwy docelowego pliku z wektorem *ritz* (rozmiar k, $k \leq m$)
- `lres_prx` – prefiks nazwy docelowego pliku z wektorem *lres* (rozmiar k, $k \leq m$)
- `cul_prx` – prefiks nazwy docelowego pliku z wektorem *cul* (rozmiar k, $k \leq m$)
- `idx_prx` – prefiks nazwy docelowego pliku z wektorem zero-jedynkowym *idx* (rozmiar m)
- `k_prx` – prefiks nazwy docelowego pliku z wartością k (skalar)

Dane wyjściowe:

- `<work_dir>/<s_prx>_m-<m>_k-<k>.{csv|bin}` – plik z niesymetryczną macierzą S (rozmiar m).

- `<work_dir>/<ritz_prx>_m-<m>_k-<k>.{csv|bin}` – plik z wektorem $ritz$ ((rozmiar k, $k \leq m$). `<work_dir>/<lres_prx>_m-<m>_k-<k>.{csv|bin}` – plik z wektorem $lres$ (rozmiar k, $k \leq m$). `<work_dir>/<cul_prx>_m-<m>_k-<k>.{csv|bin}` – plik z wektorem cul (rozmiar k, $k \leq m$). `<work_dir>/<idx_prx>_m-<m>_k-<k>.{csv|bin}` – plik z wektorem idx (rozmiar m). `<work_dir>/<k_pr` – plik z wartością k (skalar).

5 Resztę kodu podzielić na etapy:

1. distinguish the clusters of eigenvalues
2. reflect using Householder
3. compute eigenvectors

Z których każdy będzie zaimplementowany w Matlabie i C++.

6 Dodatkowe, pomocnicze funkcje w C++

Lst. 7. C++

```
1 bool cmp_vec(string work_dir, string x_vec_name, string y_vec_name,
      float eps);
2 >>> cmp_vec("../data", "a_n-48", "b_n-48", 0.0001);
```

Parametry wejściowe:

- `x_vec_name` – nazwa pliku z wektorem x
- `y_vec_name` – nazwa pliku z wektorem y
- `eps` – minimalna wartość różnicy między elementami wektora powodująca nierówność

Zwracana wartość określa czy wektory są jednakowe (true) lub różne (false).

Lst. 8. C++

```
1 bool cmp_mat(string work_dir, string x_mat_name, string y_mat_name,
      float eps);
2 >>> cmp_mat("../data", "a_n-48", "b_n-48", 0.0001);
```

Parametry wejściowe:

- `x_mat_name` – nazwa pliku z macierzą x
- `y_mat_name` – nazwa pliku z macierzą y
- `eps` – minimalna wartość różnicy między elementami macierzy powodująca nierówność

Zwracana wartość określa czy macierze (symetryczne, zredukowane) są jednakowe (true) lub różne (false).

7 Uwagi do implementacji

- W parametrze `work_dir` podawana jest ścieżka do istniejącego katalogu z którego odczytywane są i zapisywane pliki z danymi.
- Parametry o nazwach kończących się na `_name` podają nazwy istniejących plików z danymi wejściowymi (rozszerzenie pomijane)
- Parametry o nazwach kończących się na `_prx` podają prefiks nazw plików z danymi wyjściowymi.
- Implementacja w C++ zapisuje pliki docelowe binarnie i/lub w CSV w zależności od wartości parametrów `save_bin`, `save_csv`.
- Implementacja C++ odczytuje wejściowy plik CSV i konwertuje go do pliku binarnego o takiej samej nazwie i rozszerzeniu `.bin`. Przy kolejnym wywołaniu z tym samym parametrem wejściowym odczytuje plik binarny zamiast CSV.
- Implementacja w Matlabie używa tylko plików CSV dla parametrów wejściowych i wyjściowych.
- Przyjąć założenia lub wprowadzić przełączniki dla sposobu zapisu binarnych macierzy:
  - dla macierzy symetrycznych (`TrMatrixd namespace`) można zapisywać tylko górną połowę wraz z przekątną
  - dla macierzy niesymetrycznych (na razie brak obsługi) należy przeprowadzać zapis pełnej macierzy
  - można rozważyć zapis tylko niezerowych elementów macierzy A jako trójek: indeks wiersza i, indeks kolumny j, wartość A_{ij}
  - powyższe punkty są nieistotne jeśli działania będą przeprowadzane tylko na macierzach symetrycznych
- Parametr `dbg` dla funkcji C++ pozwala na włączenie trybu debugowania, który powoduje wyświetlanie szczegółowych wyników pośrednich.