НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. Ігоря СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

Протокол
до комп'ютерного практикуму №3

# РОЗВ'ЯЗАННЯ СИСТЕМ ЛІНІЙНИХ АЛГЕБРАЇЧНИХ РІВНЯНЬ ІТЕРАЦІЙНИМИ МЕТОДАМИ

Виконав студент
групи ФІ-81

Шкаліков О.В.

Перевірила:
Стьопочкіна І.В.

Київ — 2021

# Практична частина

Розглянемо застосування описаних методів на прикладі наступної системи:

$$A = \begin{pmatrix} 5.5 & 7 & 6 & 5.5 \\ 7 & 10.5 & 8 & 7 \\ 6 & 8 & 10.5 & 9 \\ 5.5 & 7 & 9 & 10.5 \end{pmatrix} \qquad b = \begin{pmatrix} 23 \\ 32 \\ 33 \\ 31 \end{pmatrix}$$

Дана матриця не є матрицею з діагональною перевагою, тому нам необхідно алгебраїчними перетвореннями привести нашу задачу до вигляду з матрицею, яка має діагональну перевагу. Для цього ми запустимо одну ітерацію метода Жордана-Гауса. Отримаємо наступну матрицю:

$$A^{'} = \begin{pmatrix} 1 & 1.27273 & 1.09091 & 1 \\ 0 & 1.59091 & 0.363636 & 0 \\ 0 & 0.363636 & 3.95455 & 3 \\ 0 & 0 & 3 & 5 \end{pmatrix}$$

Нескладно побачити, що якщо ми віднімемо від першого рядка другий рядок та четвертий, помножений на $\frac{1}{5}$, то отримаємо матрицю з діагональною перевагою. Таким чином ми маємо наступну систему рівнянь:

$$A^{''} = \begin{pmatrix} 1 & -0.318182 & 0.127273 & 0 \\ 0 & 1.59091 & 0.363636 & 0 \\ 0 & 0.363636 & 3.95455 & 3 \\ 0 & 0 & 3 & 5 \end{pmatrix} \qquad b^{''} = \begin{pmatrix} -0.145455 \\ 2.72727 \\ 7.90909 \\ 8 \end{pmatrix}$$

Приведемо систему до вигляду $x = Cx + d$:

$$C = \begin{pmatrix} 0 & 0.318182 & -0.127273 & 0 \\ 0 & 0 & -0.228571 & 0 \\ 0 & -0.091954 & 0 & -0.758621 \\ 0 & 0 & -0.6 & 0 \end{pmatrix} \qquad d = \begin{pmatrix} -0.145455 \\ 1.71429 \\ 2 \\ 1.6 \end{pmatrix}$$

| **k** | $x_k$ | $r = Ax - b$ | $r^{''} = A^{''}x - b^{''}$ | **Значення критерія** |
|---|---|---|---|---|
| 1 | $\begin{pmatrix} 0.145455 \\ 1.25714 \\ 0.628571 \\ 0.4 \end{pmatrix}$ | $\begin{pmatrix} 7.42857 \\ 9.95325 \\ 11.8701 \\ -11.5429 \end{pmatrix}$ | $\begin{pmatrix} 0.0290909 \\ 0.498701 \\ 3.76623 \\ -4.11429 \end{pmatrix}$ | 7.80659 |
| 2 | $\begin{pmatrix} 0.174545 \\ 1.57061 \\ 1.58095 \\ 1.22286 \end{pmatrix}$ | $\begin{pmatrix} 5.16571 \\ 6.92087 \\ 8.21788 \\ 8.02286 \end{pmatrix}$ | $\begin{pmatrix} 0.0214719 \\ 0.34632 \\ 2.58256 \\ 2.85714 \end{pmatrix}$ | 5.42125 |
| 3 | $\begin{pmatrix} 0.153074 \\ 1.35293 \\ 0.927891 \\ 0.651429 \end{pmatrix}$ | $\begin{pmatrix} 3.53741 \\ 4.73964 \\ 5.65244 \\ -5.4966 \end{pmatrix}$ | $\begin{pmatrix} 0.0138528 \\ 0.237477 \\ 1.79344 \\ -1.95918 \end{pmatrix}$ | 3.71743 |
| 4 | $\begin{pmatrix} 0.166926 \\ 1.5022 \\ 1.38141 \\ 1.04327 \end{pmatrix}$ | $\begin{pmatrix} 2.45986 \\ 3.29565 \\ 3.91328 \\ 3.82041 \end{pmatrix}$ | $\begin{pmatrix} 0.0102247 \\ 0.164914 \\ 1.22979 \\ 1.36054 \end{pmatrix}$ | 2.58155 |

| | | | | |
|---|---|---|---|---|
| 5 | $\begin{pmatrix} 0.156702 \\ 1.39854 \\ 1.07042 \\ 0.771156 \end{pmatrix}$ | $\begin{pmatrix} 1.68448 \\ 2.25697 \\ 2.69164 \\ -2.61743 \end{pmatrix}$ | $\begin{pmatrix} 0.00659658 \\ 0.113084 \\ 0.854021 \\ -0.932945 \end{pmatrix}$ | 1.7702 |
| 6 | $\begin{pmatrix} 0.163298 \\ 1.46962 \\ 1.28638 \\ 0.957745 \end{pmatrix}$ | $\begin{pmatrix} 1.17136 \\ 1.56936 \\ 1.86347 \\ 1.81924 \end{pmatrix}$ | $\begin{pmatrix} 0.0048689 \\ 0.0785307 \\ 0.585615 \\ 0.647878 \end{pmatrix}$ | 1.22931 |
| 7 | $\begin{pmatrix} 0.158429 \\ 1.42026 \\ 1.1383 \\ 0.82817 \end{pmatrix}$ | $\begin{pmatrix} 0.802135 \\ 1.07475 \\ 1.28173 \\ -1.24639 \end{pmatrix}$ | $\begin{pmatrix} 0.00314123 \\ 0.0538496 \\ 0.406677 \\ -0.444259 \end{pmatrix}$ | 0.842954 |
| 8 | $\begin{pmatrix} 0.161571 \\ 1.4541 \\ 1.24114 \\ 0.917022 \end{pmatrix}$ | $\begin{pmatrix} 0.557792 \\ 0.747313 \\ 0.887365 \\ 0.866306 \end{pmatrix}$ | $\begin{pmatrix} 0.00231853 \\ 0.0373956 \\ 0.278864 \\ 0.308513 \end{pmatrix}$ | 0.585384 |
| 9 | $\begin{pmatrix} 0.159252 \\ 1.4306 \\ 1.17062 \\ 0.855319 \end{pmatrix}$ | $\begin{pmatrix} 0.381969 \\ 0.511785 \\ 0.610349 \\ -0.593521 \end{pmatrix}$ | $\begin{pmatrix} 0.00149582 \\ 0.0256427 \\ 0.193656 \\ -0.211552 \end{pmatrix}$ | 0.401406 |
| 10 | $\begin{pmatrix} 0.160748 \\ 1.44672 \\ 1.21959 \\ 0.897629 \end{pmatrix}$ | $\begin{pmatrix} 0.265615 \\ 0.355863 \\ 0.422555 \\ 0.412527 \end{pmatrix}$ | $\begin{pmatrix} 0.00110406 \\ 0.0178074 \\ 0.132792 \\ 0.146911 \end{pmatrix}$ | 0.278755 |
| 11 | $\begin{pmatrix} 0.159644 \\ 1.43552 \\ 1.18601 \\ 0.868247 \end{pmatrix}$ | $\begin{pmatrix} 0.18189 \\ 0.243707 \\ 0.290642 \\ -0.282629 \end{pmatrix}$ | $\begin{pmatrix} 0.000712297 \\ 0.0122108 \\ 0.092217 \\ -0.100739 \end{pmatrix}$ | 0.191146 |
| 12 | $\begin{pmatrix} 0.160356 \\ 1.4432 \\ 1.20933 \\ 0.888395 \end{pmatrix}$ | $\begin{pmatrix} 0.126484 \\ 0.169459 \\ 0.201216 \\ 0.196441 \end{pmatrix}$ | $\begin{pmatrix} 0.000525743 \\ 0.00847972 \\ 0.0632345 \\ 0.0699577 \end{pmatrix}$ | 0.13274 |
| 13 | $\begin{pmatrix} 0.15983 \\ 1.43787 \\ 1.19334 \\ 0.874403 \end{pmatrix}$ | $\begin{pmatrix} 0.0866143 \\ 0.116051 \\ 0.138401 \\ -0.134585 \end{pmatrix}$ | $\begin{pmatrix} 0.000339189 \\ 0.00581467 \\ 0.0439128 \\ -0.047971 \end{pmatrix}$ | 0.0910219 |
| 14 | $\begin{pmatrix} 0.16017 \\ 1.44152 \\ 1.20444 \\ 0.883998 \end{pmatrix}$ | $\begin{pmatrix} 0.0602302 \\ 0.0806946 \\ 0.0958174 \\ 0.0935434 \end{pmatrix}$ | $\begin{pmatrix} 0.000250354 \\ 0.00403796 \\ 0.0301117 \\ 0.0333132 \end{pmatrix}$ | 0.0632096 |
| 15 | $\begin{pmatrix} 0.159919 \\ 1.43898 \\ 1.19683 \\ 0.877335 \end{pmatrix}$ | $\begin{pmatrix} 0.0412449 \\ 0.0552624 \\ 0.0659053 \\ -0.0640882 \end{pmatrix}$ | $\begin{pmatrix} 0.000161518 \\ 0.00276889 \\ 0.0209109 \\ -0.0228433 \end{pmatrix}$ | 0.0433438 |
| 16 | $\begin{pmatrix} 0.160081 \\ 1.44073 \\ 1.20212 \\ 0.881904 \end{pmatrix}$ | $\begin{pmatrix} 0.0286811 \\ 0.038426 \\ 0.0456273 \\ 0.0445445 \end{pmatrix}$ | $\begin{pmatrix} 0.000119216 \\ 0.00192284 \\ 0.0143389 \\ 0.0158634 \end{pmatrix}$ | 0.0300998 |

| | | | | |
|---|---|---|---|---|
| 17 | $\begin{pmatrix} 0.159962 \\ 1.43952 \\ 1.19849 \\ 0.878731 \end{pmatrix}$ | $\begin{pmatrix} 0.0196404 \\ 0.0263154 \\ 0.0313835 \\ -0.0305182 \end{pmatrix}$ | $\begin{pmatrix} 7.69136e-05 \\ 0.00131852 \\ 0.00995756 \\ -0.0108778 \end{pmatrix}$ | 0.0206399 |
| 18 | $\begin{pmatrix} 0.160038 \\ 1.44035 \\ 1.20101 \\ 0.880906 \end{pmatrix}$ | $\begin{pmatrix} 0.0136577 \\ 0.0182981 \\ 0.0217273 \\ 0.0212117 \end{pmatrix}$ | $\begin{pmatrix} 5.67695e-05 \\ 0.000915638 \\ 0.00682804 \\ 0.00755401 \end{pmatrix}$ | 0.0143333 |
| 19 | $\begin{pmatrix} 0.159982 \\ 1.43977 \\ 1.19928 \\ 0.879396 \end{pmatrix}$ | $\begin{pmatrix} 0.00935258 \\ 0.0125312 \\ 0.0149445 \\ -0.0145325 \end{pmatrix}$ | $\begin{pmatrix} 3.66255e-05 \\ 0.000627866 \\ 0.0047417 \\ -0.00517989 \end{pmatrix}$ | 0.00982852 |
| 20 | $\begin{pmatrix} 0.160018 \\ 1.44016 \\ 1.20048 \\ 0.880432 \end{pmatrix}$ | $\begin{pmatrix} 0.00650364 \\ 0.00871338 \\ 0.0103463 \\ 0.0101008 \end{pmatrix}$ | $\begin{pmatrix} 2.70331e-05 \\ 0.000436018 \\ 0.00325145 \\ 0.00359715 \end{pmatrix}$ | 0.00682536 |
| 21 | $\begin{pmatrix} 0.159991 \\ 1.43989 \\ 1.19966 \\ 0.879712 \end{pmatrix}$ | $\begin{pmatrix} 0.00445361 \\ 0.00596722 \\ 0.00711644 \\ -0.00692023 \end{pmatrix}$ | $\begin{pmatrix} 1.74407e-05 \\ 0.000298984 \\ 0.00225795 \\ -0.00246662 \end{pmatrix}$ | 0.00468025 |
| 22 | $\begin{pmatrix} 0.160009 \\ 1.44008 \\ 1.20023 \\ 0.880206 \end{pmatrix}$ | $\begin{pmatrix} 0.00309697 \\ 0.00414923 \\ 0.00492682 \\ 0.0048099 \end{pmatrix}$ | $\begin{pmatrix} 1.28729e-05 \\ 0.000207628 \\ 0.00154831 \\ 0.00171293 \end{pmatrix}$ | 0.00325017 |
| 23 | $\begin{pmatrix} 0.159996 \\ 1.43995 \\ 1.19984 \\ 0.879863 \end{pmatrix}$ | $\begin{pmatrix} 0.00212077 \\ 0.00284153 \\ 0.00338878 \\ -0.00329535 \end{pmatrix}$ | $\begin{pmatrix} 8.3051e-06 \\ 0.000142373 \\ 0.00107521 \\ -0.00117458 \end{pmatrix}$ | 0.00222869 |
| 24 | $\begin{pmatrix} 0.160004 \\ 1.44004 \\ 1.20011 \\ 0.880098 \end{pmatrix}$ | $\begin{pmatrix} 0.00147475 \\ 0.00197582 \\ 0.00234611 \\ 0.00229043 \end{pmatrix}$ | $\begin{pmatrix} 6.12996e-06 \\ 9.88703e-05 \\ 0.00073729 \\ 0.00081568 \end{pmatrix}$ | 0.0015477 |
| 25 | $\begin{pmatrix} 0.159998 \\ 1.43998 \\ 1.19992 \\ 0.879935 \end{pmatrix}$ | $\begin{pmatrix} 0.00100989 \\ 0.00135311 \\ 0.0016137 \\ -0.00156921 \end{pmatrix}$ | $\begin{pmatrix} 3.95481e-06 \\ 6.77968e-05 \\ 0.000512007 \\ -0.000559323 \end{pmatrix}$ | 0.00106128 |
| 26 | $\begin{pmatrix} 0.160002 \\ 1.44002 \\ 1.20005 \\ 0.880047 \end{pmatrix}$ | $\begin{pmatrix} 0.000702261 \\ 0.000940868 \\ 0.00111719 \\ 0.00109068 \end{pmatrix}$ | $\begin{pmatrix} 2.91903e-06 \\ 4.70811e-05 \\ 0.00035109 \\ 0.000388419 \end{pmatrix}$ | 0.000737 |
| 27 | $\begin{pmatrix} 0.159999 \\ 1.43999 \\ 1.19996 \\ 0.879969 \end{pmatrix}$ | $\begin{pmatrix} 0.0004809 \\ 0.000644338 \\ 0.000768431 \\ -0.000747244 \end{pmatrix}$ | $\begin{pmatrix} 1.88324e-06 \\ 3.22842e-05 \\ 0.000243813 \\ -0.000266344 \end{pmatrix}$ | 0.000505371 |
| 28 | $\begin{pmatrix} 0.160001 \\ 1.44001 \\ 1.20002 \\ 0.880022 \end{pmatrix}$ | $\begin{pmatrix} 0.00033441 \\ 0.000448033 \\ 0.000531997 \\ 0.000519372 \end{pmatrix}$ | $\begin{pmatrix} 1.39001e-06 \\ 2.24196e-05 \\ 0.000167186 \\ 0.000184961 \end{pmatrix}$ | 0.000350952 |

| | | | | |
|---|---|---|---|---|
| 29 | $\begin{pmatrix} 0.16 \\ 1.43999 \\ 1.19998 \\ 0.879985 \end{pmatrix}$ | $\begin{pmatrix} 0.000229 \\ 0.000306828 \\ 0.000365919 \\ -0.000355831 \end{pmatrix}$ | $\begin{pmatrix} 8.96783e-07 \\ 1.53734e-05 \\ 0.000116101 \\ -0.000126831 \end{pmatrix}$ | 0.000240653 |
| 30 | $\begin{pmatrix} 0.16 \\ 1.44 \\ 1.20001 \\ 0.880011 \end{pmatrix}$ | $\begin{pmatrix} 0.000159243 \\ 0.000213349 \\ 0.000253332 \\ 0.00024732 \end{pmatrix}$ | $\begin{pmatrix} 6.61911e-07 \\ 1.0676e-05 \\ 7.96123e-05 \\ 8.80769e-05 \end{pmatrix}$ | 0.00016712 |
| 31 | $\begin{pmatrix} 0.16 \\ 1.44 \\ 1.19999 \\ 0.879993 \end{pmatrix}$ | $\begin{pmatrix} 0.000109048 \\ 0.000146108 \\ 0.000174247 \\ -0.000169443 \end{pmatrix}$ | $\begin{pmatrix} 4.27039e-07 \\ 7.32067e-06 \\ 5.52863e-05 \\ -6.03956e-05 \end{pmatrix}$ | 0.000114597 |
| 32 | $\begin{pmatrix} 0.16 \\ 1.44 \\ 1.20001 \\ 0.880005 \end{pmatrix}$ | $\begin{pmatrix} 7.583e-05 \\ 0.000101595 \\ 0.000120634 \\ 0.000117771 \end{pmatrix}$ | $\begin{pmatrix} 3.15196e-07 \\ 5.0838e-06 \\ 3.79106e-05 \\ 4.19414e-05 \end{pmatrix}$ | 7.9581e-05 |

# Додатки

Далі наведено програмний код імплементованих алгоритмів. Вихідний код, який було створено для даного практикума (у тому числі LaTeX), можна знайти за наступним посиланням.

**Класс для інкупсуляції роботи з пам'ятю**

```
#pragma once

#include <cstddef>
#include <algorithm>

using std::size_t;


namespace LSE
{
    template <typename T>
    class Buffer
    {
    public:
        Buffer(size_t size)
        {
            _size = size;
            _data = new T[size]();
        }

        Buffer(const Buffer &lhs)
        {
            _data = new T[lhs._size];
            _size = lhs._size;
            std::copy(lhs._data, lhs._data + lhs._size, _data);
        }

        Buffer(Buffer &&lhs) noexcept
        {
            _data = std::exchange(lhs._data, nullptr);
            _size = std::exchange(lhs._size, 0);
        }

        ~Buffer()
        {
            delete[] _data;
        }

        Buffer& operator=(const Buffer &lhs)
        {
            T *_newData = new T[lhs._size];
```

```
            std::copy(lhs._data, lhs._data + lhs._size, _newData)
               ↪ ;

            _size = lhs._size;
            delete[] _data;
            _data = _newData;

            return *this;
        }

        Buffer& operator=(Buffer &&lhs) noexcept
        {
            std::swap(_data, lhs._data);
            std::swap(_size, lhs._size);
            return *this;
        }

        T& operator[](std::size_t pos)
        {
            return _data[pos];
        }

        const T& operator[](std::size_t pos) const
        {
            return _data[pos];
        }

        inline size_t size() const noexcept { return _size; }

    private:
        T *_data;
        size_t _size;
    };
}
```

**Клас векторів**

```
#pragma once

#include <cmath>
#include "Buffer.h"

namespace LSE
{
    template <typename T>
    class Vector
    {
    public:
        Vector(size_t nval) : _buff(nval) {}

        template<typename U>
```

```
    Vector(const Vector<U>& vector) : _buff(vector.nval())
    {
        for (size_t i = 0; i < vector.nval(); i++)
        {
            _buff[i] = vector(i);
        }
    };

    inline size_t nval() const noexcept { return _buff.size()
      ↪ ; }

    T& operator()(size_t i);
    const T& operator()(size_t i) const;

    auto norm() const noexcept;

private:
    Buffer<T> _buff;
};

template <typename T>
T& Vector<T>::operator()(size_t i)
{
    if (i >= _buff.size())
        throw std::invalid_argument("Position␣is␣out␣of␣range
          ↪ ");
    return _buff[i];
}

template <typename T>
const T& Vector<T>::operator()(size_t i) const
{
    if (i >= _buff.size())
        throw std::invalid_argument("Position␣is␣out␣of␣range
          ↪ ");
    return _buff[i];
}

template <typename T>
auto Vector<T>::norm() const noexcept
{
    T sum = 0;
    for (size_t i = 0; i < _buff.size(); i++)
    {
        sum += _buff[i] * _buff[i];
    }
    return std::sqrt(sum);
}

template <typename T, typename U>
```

```
    auto operator+(const Vector<T> &rhs, const Vector<U> &lhs)
    {
        if (rhs.nval() != lhs.nval())
            throw std::invalid_argument("Different size");

        Vector<typename std::common_type<T, U>::type> result(rhs.
            ↪ nval());
        for (size_t i = 0; i < rhs.nval(); i++)
        {
            result(i) = rhs(i) + lhs(i);
        }
        return result;
    }

    template <typename T, typename U>
    auto operator-(const Vector<T> &rhs, const Vector<U> &lhs)
    {
        if (rhs.nval() != lhs.nval())
            throw std::invalid_argument("Different size");

        Vector<typename std::common_type<T, U>::type> result(rhs.
            ↪ nval());
        for (size_t i = 0; i < rhs.nval(); i++)
        {
            result(i) = rhs(i) - lhs(i);
        }
        return result;
    }

    template <typename T>
    T getMaxAbsValue(const Vector<T> &vector)
    {
        T max = std::abs(vector(0));

        for (size_t i = 1; i < vector.nval(); i++)
        {
            if (max < std::abs(vector(i)))
                max = std::abs(vector(i));
        }

        return max;
    }
}
```

**Клас матриць**

```
#pragma once

#include "Buffer.h"
#include "Vector.h"
```

```
namespace LSE
{
    template <typename T>
    class IMatrix
    {
    public:
        size_t nrow() const noexcept { return _nrow; }
        size_t ncol() const noexcept { return _ncol; }

        virtual T &operator()(size_t i, size_t j) = 0;
        virtual const T& operator()(size_t i, size_t j) const =
            ↪ 0;

    protected:
        size_t _nrow;
        size_t _ncol;
    };

    template <typename T>
    class Matrix : public IMatrix<T>
    {
    public:
        Matrix(size_t nrow, size_t ncol) : _buff(nrow * ncol)
        {
            this->_nrow = nrow;
            this->_ncol = ncol;
        }

        template<typename U>
        Matrix(const Matrix<U>& matrix) : _buff(matrix._buff.size
            ↪ ())
        {
            for (size_t i = 0; i < matrix.nrow(); i++)
            {
                for (size_t j = 0; j < matrix.ncol(); j++)
                {
                    _buff[i] = matrix(i, j);
                }
            }
        };

        T &operator()(size_t i, size_t j) override
        {
            if (i >= this->_nrow || j >= this->_ncol)
                throw std::invalid_argument("Position␣is␣out␣of␣
                    ↪ range");
            return _buff[i * this->_ncol + j];
        }

        const T& operator()(size_t i, size_t j) const override
```

```
    {
        if (i >= this->_nrow || j >= this->_ncol)
            throw std::invalid_argument("Position is out of
                ↪ range");
        return _buff[i * this->_ncol + j];
    }

    Matrix<T> transpose()
    {
        Matrix<T> result(this->_nrow, this->_ncol);

        for (size_t i = 0; i < this->_nrow; i++)
        {
            for (size_t j = 0; j < this->_ncol; j++)
            {
                result(i, j) = operator()(j, i);
            }
        }

        return result;
    }

    bool isSymmetric() const noexcept
    {
        if (this->_ncol != this->_nrow)
        return false;

        for (size_t i = 0; i < this->_nrow; i++)
        {
            for (size_t j = 0; j < i; j++)
            {
                if (operator()(i, j) != operator()(j, i))
                    return false;
            }
        }

        return true;
    }

private:
    Buffer<T> _buff;
};

template <typename T, typename U>
auto operator*(const IMatrix<T> &rhs, const IMatrix<U> &lhs)
{
    if (rhs.ncol() != lhs.nrow())
        throw std::invalid_argument("Number of column !=
            ↪ number of row");
```

```
        Matrix<typename std::common_type<T, U>::type> result(rhs.
            ↪ nrow(), lhs.ncol());

        for (size_t i = 0; i < rhs.nrow(); i++)
        {
            for (size_t j = 0; j < lhs.ncol(); j++)
            {
                for (size_t k = 0; k < rhs.ncol(); k++)
                {
                    result(i, j) += rhs(i, k) * lhs(k, j);
                }
            }
        }

        return result;
    }

    template <typename T, typename U>
    auto operator*(const IMatrix<T> &matrix, const Vector<U> &
        ↪ vector)
    {
        if (matrix.ncol() != vector.nval())
            throw std::invalid_argument("Number␣of␣column␣!=␣size
                ↪ ␣of␣vector");

        Vector<typename std::common_type<T, U>::type> result(
            ↪ matrix.nrow());

        for (size_t i = 0; i < matrix.nrow(); i++)
        {
            for (size_t j = 0; j < matrix.ncol(); j++)
            {
                result(i) += vector(j) * matrix(i, j);
            }
        }

        return result;
    }
}
```

**Класи для оптимізації роботи з матрицями та векторами**

```
#pragma once

#include <unordered_map>

#include "Vector.h"
#include "Matrix.h"

namespace LSE
{
```

```
template<typename T>
class IMatrixView : public IMatrix<T>
{
public:

    void swapRows(size_t i, size_t j)
    {
        swapLines(i, j, _rowMap);
    }

protected:
    size_t getRow(size_t i) const noexcept
    {
        auto it = _rowMap.find(i);
        if (it != _rowMap.end())
        {
            i = it->second;
        }
        return i;
    };

    void swapLines(size_t i, size_t j, std::unordered_map<
       ↪ size_t, size_t> &map)
    {
        size_t iNew, jNew;

        auto it = map.find(i);
        if (it != map.end())
            jNew = it->second;
        else
            jNew = i;

        it = map.find(j);
        if (it != map.end())
            iNew = it->second;
        else
            iNew = j;

        map.insert({i, iNew});
        map.insert({j, jNew});
    }

    std::unordered_map<size_t, size_t> _rowMap;
    IMatrix<T>* _matrix;
};

template <typename T>
class TransposeView : public IMatrixView<T>
{
public:
```

```
    TransposeView(IMatrix<T> &matrix)
    {
        this->_matrix = &matrix;
        this->_nrow = matrix.nrow();
        this->_ncol = matrix.ncol();
    }

    T &operator()(size_t i, size_t j) override
    {
        if (j >= this->_nrow || i >= this->_ncol)
            throw std::invalid_argument("Position is out of
                ↪ range");
        return this->_matrix->operator()(this->getRow(j), i);
    }

    const T& operator()(size_t i, size_t j) const override
    {
        if (j >= this->_nrow || i >= this->_ncol)
            throw std::invalid_argument("Position is out of
                ↪ range");
        return this->_matrix->operator()(this->getRow(j), i);
    }
};

template<typename T>
class MatrixView : public IMatrixView<T>
{
public:
    MatrixView(IMatrix<T> &matrix)
    {
        this->_matrix = &matrix;
        this->_nrow = matrix.nrow();
        this->_ncol = matrix.ncol();
    }

    const T& operator()(size_t i, size_t j) const override
    {
        return this->_matrix->operator()(this->getRow(i), j);
    }

    T& operator()(size_t i, size_t j) override
    {
        return this->_matrix->operator()(this->getRow(i), j);
    }
};

template<typename T>
class MatrixMatrixView : public IMatrixView<T>
{
public:
```

```
    MatrixMatrixView(IMatrix<T> &matrix1, IMatrix<T> &matrix2
        ↪ )
    {
        this->_matrix = &matrix1;
        this->_matrix2 = &matrix2;
        this->_nrow = matrix1.nrow();
        this->_ncol = matrix1.ncol() + matrix2.ncol();
    }

    const T& operator()(size_t i, size_t j) const override
    {
        if (j >= this->_ncol)
            throw std::invalid_argument("Position␣is␣out␣of␣
                ↪ range");

        if (j >= this->_matrix->ncol())
        {
            return this->_matrix2->operator()(this->getRow(i)
                ↪ , j - this->_matrix->ncol());
        }

        return this->_matrix->operator()(i, j);
    }

    T& operator()(size_t i, size_t j) override
    {
        if (j >= this->_ncol)
            throw std::invalid_argument("Position␣is␣out␣of␣
                ↪ range");

        if (j >= this->_matrix->ncol())
        {
            return this->_matrix2->operator()(this->getRow(i)
                ↪ , j - this->_matrix->ncol());
        }

        return this->_matrix->operator()(i, j);
    }

private:
    IMatrix<T>* _matrix2;
};

template<typename T>
class MatrixVectorView : public IMatrixView<T>
{
public:
    MatrixVectorView(IMatrix<T> &matrix, Vector<T> &vector)
    {
        this->_matrix = &matrix;
```

```
            _vector = &vector;
            this->_nrow = matrix.nrow();
            this->_ncol = matrix.ncol() + 1;
        }

        const T& operator()(size_t i, size_t j) const override
        {
            if (j >= this->_ncol)
                throw std::invalid_argument("Position is out of
                    ↪ range");

            i = this->getRow(i);
            if (j >= this->_matrix->ncol())
            {
                return _vector->operator()(i);
            }

            return this->_matrix->operator()(i, j);
        }

        T& operator()(size_t i, size_t j) override
        {
            if (j >= this->_ncol)
                throw std::invalid_argument("Position is out of
                    ↪ range");

            i = this->getRow(i);
            if (j >= this->_matrix->ncol())
            {
                return _vector->operator()(i);
            }

            return this->_matrix->operator()(i, j);
        }

    private:
        Vector<T>* _vector;
    };
}
```

**Утілити для представлення матриць та векторів у консолі**

```
#pragma once

#include <cmath>

#include "Matrix.h"
#include "Vector.h"

namespace LSE
{
```

```
#ifndef LATEX
    template <typename T>
    void print(const Matrix<T> &matrix)
    {
        for (size_t i = 0; i < matrix.nrow(); i++)
        {
            for (size_t j = 0; j < matrix.ncol(); j++)
            {
                std::cout << matrix(i, j) << "\t";
            }
            std::cout << std::endl;
        }
    }

    template <typename T>
    void print(const Vector<T> &vector)
    {
        for (size_t i = 0; i < vector.nval(); i++)
        {
            std::cout << vector(i) << "\t";
        }
        std::cout << std::endl;
    }

    template <typename T>
    void printAbs(const Vector<T> &vector)
    {
        for (size_t i = 0; i < vector.nval(); i++)
        {
            std::cout << std::abs(vector(i)) << "\t";
        }
        std::cout << std::endl;
    }
#else
    template <typename T>
    void print(const Matrix<T> &matrix)
    {
        std::cout << "\\begin{pmatrix}" << std::endl;
        for (size_t i = 0; i < matrix.nrow(); i++)
        {
            std::cout << "\t";
            for (size_t j = 0; j < matrix.ncol() - 1; j++)
            {
                std::cout << matrix(i, j) << "␣&␣";
            }
            std::cout << matrix(i, matrix.ncol() - 1);
            if (i != matrix.nrow() - 1)
                std::cout << "␣\\\\";
            std::cout << std::endl;
        }
```

```
        std::cout << "\\end{pmatrix}" << std::endl;
    }

    template <typename T>
    void print(const Vector<T> &vector)
    {
        std::cout << "\\begin{pmatrix}" << std::endl << "\t";;
        for (size_t i = 0; i < vector.nval() - 1; i++)
        {
            std::cout << vector(i) << "␣\\\\␣";
        }
        std::cout << vector(vector.nval() - 1) << std::endl << "
           ↪ \\end{pmatrix}" << std::endl;
    }

    template <typename T>
    void printAbs(const Vector<T> &vector)
    {
        std::cout << "\\begin{pmatrix}" << std::endl << "\t";;
        for (size_t i = 0; i < vector.nval() - 1; i++)
        {
            std::cout << std::abs(vector(i)) << "␣\\\\␣";
        }
        std::cout << vector(vector.nval() - 1) << std::endl << "
           ↪ \\end{pmatrix}" << std::endl;
    }
#endif
}
```

**Методи Гаусса**

```
#pragma once

namespace LSE
{
    template <typename T>
    void gaussJordanEliminationStep(IMatrix<T>& matrix, size_t i,
       ↪  size_t j)
    {
        for (size_t k = 0; k < matrix.nrow(); k++)
        {
            auto pivot = matrix(i, j);
            auto mult = matrix(k, j) / pivot;
            for (size_t l = 0; l < matrix.ncol(); l++)
            {
                if (i == k)
                    matrix(k, l) = matrix(k, l) / pivot;
                else if (j == l)
                    matrix(k, l) = 0;
                else
                    matrix(k, l) = matrix(k, l) - mult * matrix(i
```

```
                    ↪ , l);
            }
        }
    }

    template <typename T>
    void gaussEliminationStep(IMatrix<T>& matrix, size_t i,
       ↪ size_t j)
    {
        for (size_t k = i + 1; k < matrix.nrow(); k++)
        {
            auto pivot = matrix(i, j);
            auto mult = matrix(k, j) / pivot;
            for (size_t l = j; l < matrix.ncol(); l++)
            {
                if (j == l)
                    matrix(k, l) = 0;
                else
                    matrix(k, l) = matrix(k, l) - mult * matrix(i
                       ↪ , l);
            }
        }
    }
}
```

**Процедура обрахунку коренів лінійної системи рівнянь**

```
#pragma once
#include <vector>

#include "Matrix.h"
#include "Vector.h"
#include "MatrixView.h"
#include "Cholesky.h"
#include "Gauss.h"
#include "Substitution.h"

#ifdef PRINT
#include "Utils.h"
#endif

namespace LSE
{
    template <typename T, typename U>
    auto choleskySolve(const Matrix<T> &A, const Vector<U> &b)
    {
        auto chol = cholesky(A);
        auto y = forward(chol, b);
        auto x = backward(TransposeView(chol), y);

#ifdef PRINT
```

```
        std::cout << "cholesky solver y" << std::endl;
        print(y);
        std::cout << std::endl;
#endif

        return x;
    }

    template <typename T, typename U>
    auto ldlSolve(const Matrix<T> &A, const Vector<U> &b)
    {
        auto [l, d] = ldl(A);
        auto z = forward(l, b);

#ifdef PRINT
        std::cout << "LDL solver z" << std::endl;
        print(z);
        std::cout << std::endl;
#endif

        for (size_t i = 0; i < z.nval(); i++)
        {
            z(i) = z(i) / d(i);
        }

#ifdef PRINT
        std::cout << "LDL solver y" << std::endl;
        print(z);
        std::cout << std::endl;
#endif

        auto x = backward(TransposeView(l), z);
        return x;
    }

    template <typename T, typename U>
    auto gaussJordanSolve(const Matrix<T> &A, const Vector<U> &b)
    {
        using ret_type = typename std::common_type<T, U>::type;
        Matrix<ret_type> m(A);
        Vector<ret_type> v(b);
        MatrixVectorView<ret_type> aug(m, v);

        for (size_t i = 0; i < m.nrow(); i++)
        {
            LSE::gaussJordanEliminationStep(aug, i, i);

#ifdef PRINT
            std::cout << "Gauss Jordan solver. Step: " << i + 1
              ↪ << std::endl;
```

```
            std::cout << "Matrix A: " << std::endl;
            print(m);
            std::cout << "Vector b: " << std::endl;
            print(v);
#endif
        }

        return v;
    }

    template <typename T, typename U>
    auto gaussSolve(const Matrix<T> &A, const Vector<U> &b)
    {
        using ret_type = typename std::common_type<T, U>::type;
        Matrix<ret_type> m(A);
        Vector<ret_type> v(b);
        MatrixVectorView<ret_type> aug(m, v);

        for (size_t i = 0; i < m.nrow(); i++)
        {
            LSE::gaussEliminationStep(aug, i, i);

#ifdef PRINT
            std::cout << "Gauss solver. Step: " << i + 1 << std::
               ↪ endl;
            std::cout << "Matrix A: " << std::endl;
            print(m);
            std::cout << "Vector b: " << std::endl;
            print(v);
#endif
        }

        auto x = backward(m, v);

        return x;
    }

    template <typename T, typename U>
    auto gaussPivotSolve(const Matrix<T> &A, const Vector<U> &b)
    {
        using ret_type = typename std::common_type<T, U>::type;
        Matrix<ret_type> m(A);
        Vector<ret_type> v(b);
        MatrixVectorView<ret_type> aug(m, v);

        for (size_t i = 0; i < m.nrow(); i++)
        {
            auto max = i;
            for (size_t l = i + 1; l < m.nrow(); l++)
            {
```

```
                if (std::abs(aug(l, i)) > std::abs(aug(max, i)))
                    max = l;
            }
            if (max != i)
            {
                aug.swapRows(i, max);
            }

            LSE::gaussEliminationStep(aug, i, i);

#ifdef PRINT
            std::cout << "Gauss with pivoting solver. Step: " <<
               ↪ i + 1 << std::endl;
            std::cout << "Matrix A: " << std::endl;
            print(m);
            std::cout << "Vector b: " << std::endl;
            print(v);
#endif
        }

        size_t i = A.nrow();
        Vector<ret_type> res(b.nval());
        do
        {
            i--;
            T sum = 0;
            for (size_t j = i + 1; j < A.nrow(); j++)
            {
                sum += aug(i, j) * res(j);
            }
            res(i) = (aug(i, A.ncol()) - sum) / aug(i, i);
        } while (i != 0);

        return res;
    }

namespace detail
{
    template <typename T, typename U>
    auto computeCdq(const Matrix<T> &A, const Vector<U> &b)
    {
        using ret_type = typename std::common_type<T, U>::type;
        Matrix<ret_type> C(A.nrow(), A.ncol());
        Vector<ret_type> d(b.nval());
        double q;

        for (size_t i = 0; i < C.nrow(); i++)
        {
            const auto &a = A(i, i);
            double qTemp = 0;
```

```
            for (size_t j = 0; j < C.ncol(); j++)
            {
                if (i != j)
                {
                    auto value = -A(i, j) / a;
                    C(i, j) = value;
                    qTemp += std::abs(value);
                }
            }

            d(i) = b(i) / a;

            if (q < qTemp)
                q = qTemp;
        }

        return std::make_tuple(C, d, q);
    }
}

    template <typename T, typename U>
    auto simpleIteration(const Matrix<T> &A, const Vector<U> &b,
       ↪ double precision = 0.0001, int maxIteration = 100)
    {
        if (A.ncol() != A.nrow() || A.nrow() != b.nval())
            throw std::invalid_argument("Invalid sizes");

        using ret_type = typename std::common_type<T, U>::type;
        auto [C, d, q] = detail::computeCdq(A, b);

#ifdef PRINT
        std::cout << "Matrix C: " << std::endl;
        print(C);
        std::cout << "Vector d: " << std::endl;
        print(d);
#endif

        Vector<ret_type> x = d;
        Vector<ret_type> xprev(b.nval());

        int iteration = 0;
        double criterion;
        precision = precision * (1-q) / q;

        do
        {
            xprev = x;
            x = C * xprev + d;
            criterion = getMaxAbsValue(xprev - x);
```

```
#ifdef PRINT
            std::cout << "Iteration: " << iteration + 1 << std::
               ↪ endl;
            std::cout << "x: " << std::endl;
            print(x);
            std::cout << "Residue: "<< std::endl;
            printAbs(A*x - b);
            std::cout << "Criteria: " << criterion / (1-q) * q <<
               ↪  std::endl;
#endif

            iteration++;
        } while (criterion > precision && iteration <
           ↪ maxIteration);

        return x;
    }

    template <typename T, typename U>
    auto gaussSeidel(const Matrix<T> &A, const Vector<U> &b,
       ↪ double precision = 0.0001, int maxIteration = 100)
    {
        if (A.ncol() != A.nrow() || A.nrow() != b.nval())
            throw std::invalid_argument("Invalid sizes");

        using ret_type = typename std::common_type<T, U>::type;
        auto [C, d, q] = detail::computeCdq(A, b);

#ifdef PRINT
        std::cout << "Matrix C: " << std::endl;
        print(C);
        std::cout << "Vector d: " << std::endl;
        print(d);
#endif

        Vector<ret_type> x = d;
        Vector<ret_type> xprev(b.nval());

        int iteration = 0;
        double criterion;

        do
        {
            xprev = x;

            for (size_t i = 0; i < C.nrow(); i++)
            {
                x(i) = d(i);
                for (size_t j = 0; j < C.ncol(); j++)
```

```
                {
                    if (i > j)
                    {
                        x(i) += C(i, j) * x(j);
                    }
                    else
                    {
                        x(i) += C(i, j) * xprev(j);
                    }
                }
            }

            criterion = getMaxAbsValue(xprev - x);

#ifdef PRINT
            std::cout << "Iteration: " << iteration + 1 << std::
               ↪ endl;
            std::cout << "x: " << std::endl;
            print(x);
            std::cout << "Residue: "<< std::endl;
            printAbs(A*x - b);
            std::cout << "Criteria: " << criterion << std::endl;
#endif

            iteration++;
        } while (criterion > precision && iteration <
           ↪ maxIteration);

        return x;
    }
}
```

**Розв'язання рівняння з практичної частини**

```
#include <iostream>

#include "Matrix.h"
#include "Vector.h"
#include "Solvers.h"
#include "Utils.h"

template <typename T>
void substractRows(LSE::IMatrix<T>& matrix, size_t fromIdx,
   ↪ size_t whatIdx, T coeff = 1)
{
    for (size_t j = 0; j < matrix.ncol(); j++)
    {
        matrix(fromIdx, j) -= coeff * matrix(whatIdx, j);
    }
}
```

```
int main()
{
    LSE::Matrix<double> A(4, 4);
    A(0, 0) = A(0, 3) = A(3, 0) = 5.5;
    A(0, 1) = A(1, 0) = A(3, 1) = A(1, 3) = 7.0;
    A(1, 1) = A(2, 2) = A(3,3) = 10.5;
    A(1, 2) = A(2, 1) = 8.0;
    A(2, 0) = A(0,2) = 6.0;
    A(3, 2) = A(2,3) = 9.0;

    std::cout << "Matrix A" << std::endl;
    print(A);
    std::cout << std::endl;

    LSE::Vector<double> b(4);
    b(0) = 23;
    b(1) = 32;
    b(2) = 33;
    b(3) = 31;

    std::cout << "Vector b" << std::endl;
    print(b);
    std::cout << std::endl;

    LSE::Matrix<double> mat = A;
    LSE::Vector<double> vec = b;
    LSE::MatrixVectorView<double> tmat(mat, vec);
    LSE::gaussJordanEliminationStep(tmat, 0, 0);

    std::cout << "Matrix A after one Gauss-Jordan elimination
      ↪ iteration:" << std::endl;
    print(mat);
    std::cout << std::endl;

    substractRows(tmat, 0, 1);
    substractRows(tmat, 0, 3, 1/5.);

    std::cout << "Our task with diagonally dominant matrix" <<
      ↪ std::endl;
    std::cout << "Matrix A" << std::endl;
    print(mat);
    std::cout << std::endl;
    std::cout << "Vector b" << std::endl;
    print(vec);
    std::cout << std::endl;

    auto xsi = LSE::simpleIteration(mat, vec);
    std::cout << "Simple iteration. Solution: " << std::endl;
    print(xsi);
    std::cout << std::endl;
```

```
    auto xgs1 = LSE::gaussSeidel(A, b);
    auto xgs2 = LSE::gaussSeidel(mat, vec);
    std::cout << "Gauss-Seidel. Solutions: " << std::endl;

    std::cout << "Without diagonally dominant matrix: " << std::
      ↪ endl;
    print(xgs1);
    std::cout << std::endl;

    std::cout << "With diagonally dominant matrix: " << std::endl
      ↪ ;
    print(xgs2);

    return 0;
}
```