連続系アルゴリズム
レポート課題7

連絡先：mail@myuuuuun.com

平成 27 年 12 月 7 日

レポート内で使用したプログラムは GitHub にアップロードしています。 https://github.com/myuuuuun/various/tree/master/ContinuousAlgorithm/HW7/

問題 1

三重対角行列の LU 分解を考える。l_i, d_i, u_i を他の変数で表わせ。計算量はいくらか。

$$\begin{pmatrix} 1 & & & & \\ l_2 & 1 & & & \\ & l_3 & 1 & & \\ & & \ddots & \ddots & \\ & & & l_n & 1 \end{pmatrix} \begin{pmatrix} d_1 & u_1 & & & \\ & d_2 & u_2 & & \\ & & d_3 & \ddots & \\ & & & \ddots & u_{n-1} \\ & & & & d_n \end{pmatrix} = \begin{pmatrix} a_1 & b_1 & & & \\ c_2 & a_2 & b_2 & & \\ & c_3 & a_3 & \ddots & \\ & & \ddots & \ddots & b_{n-1} \\ & & & c_n & a_n \end{pmatrix}$$

$$d_i = \begin{cases} a_1 & if\ i = 1 \\ a_i - l_i\ u_{i-1} & otherwise \end{cases}$$

$$u_i = b_i$$
$$l_i = \frac{c_i}{d_{i-1}} \quad (i \geq 2)$$

を、$i = 1$ から n まで順に求めていけばよい。
各 $i = 1, 2, \ldots, n$ に対して 3 回ずつ計算を行っているので、計算量は $O(n)$ ていど。

問題 2

行列の積を求めるプログラムを書け。

ソースコード: hw7-1.cpp、重要部のみ転記

```
#include <iostream>
#include <array>
#include <cmath>
#include <limits>
#include <iomanip>
using namespace std;

template <size_t row, size_t col>
  using array_2d = std::array<std::array<double, col>, row>;
```

```
// matrix class
template <size_t r_num, size_t c_num> class matrix{
  const int r = r_num;
  const int c = c_num;

public:
  array_2d<r_num, c_num> mat;
  const int row(void){return r;};
  const int col(void){return c;};
  matrix();
  matrix(array_2d<r_num, c_num> &base_mat);
  std::array<double, c_num>& operator[](const int i);
  const matrix<r_num, c_num> &operator=(const matrix<r_num, c_num> &mat1);
  template <size_t c2_num>
  const matrix<r_num, c2_num> operator*(const matrix<c_num, c2_num> &mat1);
};


// constructer
template <size_t r_num, size_t c_num>
matrix<r_num, c_num>::matrix(){};

template <size_t r_num, size_t c_num>
matrix<r_num, c_num>::matrix(array_2d<r_num, c_num> &base_mat)
{
  for(int i=0; i<r_num; ++i){
    for(int j=0; j<c_num; ++j){
      (*this).mat[i][j] = base_mat[i][j];
    }
  }
};


// print function for matrix
template <size_t r_num, size_t c_num>
void print(matrix<r_num, c_num> mat){
  cout.precision(8);
  for(int i=0; i<mat.row(); ++i){
    for(int j=0; j<mat.col(); ++j){
      cout << setw(9) << mat[i][j];
    }
    cout << endl;
  }
  cout << endl;
}


// [] operator
template <size_t r_num, size_t c_num>
std::array<double, c_num> &matrix<r_num, c_num>::operator[](const int i)
{
  return mat[i];
}
```

```
// * operator
template <size_t r_num, size_t c_num>
template <size_t c2_num>
const matrix<r_num, c2_num> matrix<r_num, c_num>
        ::operator*(const matrix<c_num, c2_num> &mat1)
{
  matrix<r_num, c2_num> result;

  for(int i=0; i<result.row(); ++i){
    for(int j=0; j<result.col(); ++j){
      for(int k=0; k<(*this).col(); ++k){
        result[i][j] += (*this).mat[i][k] * mat1.mat[k][j];
      }
    }
  }
  return result;
}


int main(){
  array_2d<2, 3> c_base={{{1, 2, 3}, {4, 5, 6}}};
  array_2d<3, 2> d_base={{{1, 2}, {3, 4}, {5, 6}}};
  matrix<2, 3> c(c_base);
  matrix<3, 2> d(d_base);
  print(c);
  print(d);
  print(c*d);
  return 0;
}
```

出力結果:

```
1   2   3
4   5   6

1   2
3   4
5   6

22   28
49   64
```

この結果は正しい。

オプション

定式化ができないので、LU 分解のコードだけ書きます……

ソースコード: hw7-1.cpp、重要部のみ転記

```
// n 行n 列の要素に対して, 以降の行でpivot 選択を実行する
// 絶対値がよりも大きな0が無ければを返すpivotfalse
template <size_t row, size_t col>
bool partial_pivot(matrix<row, col> &mat1, const int n)
{
  double v;

  // 列の絶対値が最大の行を探索n
  double max_v = abs(mat1.mat[n][n]);
  int max_r = n;
  for(int i=n; i<row; ++i){
    if(abs(mat1.mat[i][n]) > max_v){
      max_v = mat1.mat[i][n];
      max_r = i;
    }
  }

  // 有効なpivot がない場合、その方程式は解けない(rank 落ちの可能性が高い)
  if(max_v < EPS){
    return false;
  }

  // swap
  if(max_r != n){
    for(int i=0; i<col; ++i){
      v = mat1.mat[n][i];
      mat1.mat[n][i] = mat1.mat[max_r][i];
      mat1.mat[max_r][i] = v;
    }
  }

  return true;
}


// LU decomposition(with partial pivoting)
template <size_t size>
const matrix<size, size+1> lu_decomposition(const matrix<size, size> &mat1)
{
  matrix<size, size+1> result;

  // をmat1copy
  for(int i=0; i<size; ++i){
    for(int j=0; j<size; ++j){
      result[i][j] = mat1.mat[i][j];
    }
  }

  // swap する前の元の行列番号を最終列に記録
  for(int i=0; i<size; ++i){
    result[i][size] = i;
  }

  // LU 分解
  for(int i=0; i<size; ++i){
    // pivoting
```

```
    if (partial_pivot(result, i) == false){
      exit(-1);
    };

    double d = 1.0 / result.mat[i][i];
    for(int j=i+1; j<size; ++j){
      result.mat[j][i] = result.mat[j][i] * d;
      for(int k=i+1; k<size; ++k){
        result.mat[j][k] -= result.mat[j][i] * result.mat[i][k];
      }
    }
  }

  return result;
}


// solve LU*x=v using LU matrix and v vector
template <size_t size>
const std::array<double, size> lu_solve(const matrix<size, size+1> &mat1,
        const std::array<double, size> &v)
 {
  // vector を(pivot 選択の結果に従って並び替え)
  std::array<double, size> ordered_v;
  int ordered_r;
  for(int i=0; i<size; ++i){
    ordered_r = mat1.mat[i][size];
    ordered_v[i] = v[ordered_r];
  }

  // 前進代入
  for(int i=0; i<size; ++i){
    double sub = 0;
    for(int j=0; j<i; ++j){
      sub += ordered_v[j] * mat1.mat[i][j];
    }
    ordered_v[i] = ordered_v[i] - sub;
  }

  // 後退代入
  for(int i=size-1; i>=0; --i){
    double sub = 0;
    for(int j=size-1; j>i; --j){
      sub += ordered_v[j] * mat1.mat[i][j];
    }
    ordered_v[i] = (ordered_v[i] - sub) / mat1.mat[i][i];
  }

  return ordered_v;
}

int main(){
  std::array<double, 3> b = {10, 13, 6}, x;
  array_2d<3, 3> A_base={{{2, 3, -1}, {4, 4, -3}, {-2, 3, -1}}};
  matrix<3, 3> A(A_base);
  matrix<3, 4> LU;
  LU = lu_decomposition(A);
  print(LU);
```

```
  x = lu_solve(LU, b);
  print(x);
  return 0;
}
```