Numerical Methods for CSE

AUTUMN 2023

Q+A 22.09.2023

VIDEOS

LECTURE NOTES

HAND SCRIPT

every week.

interactive

Q+A

discuss

HOMEWORK*

TUTORIALS

very important

very important

NO BONUS

A MOCK EXAM "BYOD" + "DIY" NOVEMBER

"CODE EXPERT"

Def Kronecker produkt of two matrices

$\underline{\underline{A}} \in \mathbb{R}^{m \times n}$ $\quad \underline{\underline{B}} \in \mathbb{R}^{l \times k}$

$m, n, l, k \in \mathbb{N}$

$\underline{\underline{A}} \otimes \underline{\underline{B}} \in \mathbb{R}^{(ml) \times (nk)}$

Block of size $l \times k$

$$\begin{bmatrix} A_{11}\underline{\underline{B}} & A_{12}\underline{\underline{B}} & \dots & A_{1n}\underline{\underline{B}} \\ \vdots & & & \\ A_{m1}\underline{\underline{B}} & A_{m2}\underline{\underline{B}} & \dots & A_{mn}\underline{\underline{B}} \end{bmatrix}$$

$\hookrightarrow (ml) \times (nk)$

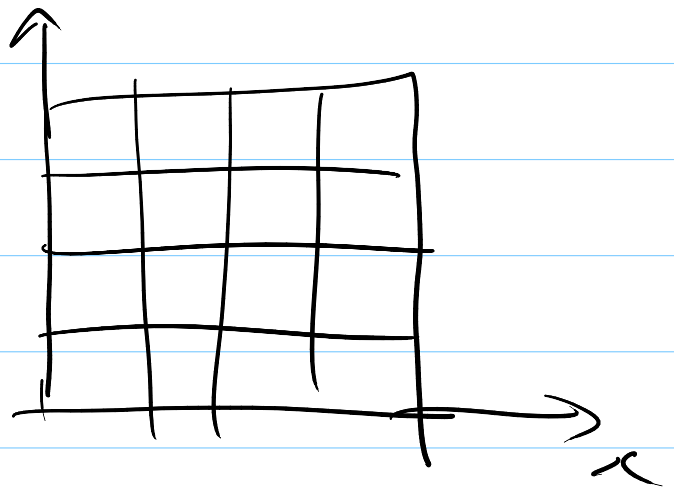
Example $\quad I_m \otimes \underline{\underline{A}} = \begin{bmatrix} 1\underline{\underline{A}} & & 0 \\ & 1\underline{\underline{A}} & \\ 0 & & \ddots \\ & & & 1\underline{\underline{A}} \end{bmatrix}$

$\underline{\underline{A}} \times I_m = \begin{bmatrix} a_{11}\underline{\underline{I}} & a_{12}\underline{\underline{I}} & \dots & a_{1k} \\ & a_{22}\underline{\underline{I}} & & \\ \vdots & & & \\ a_{n1} & a_{n2} & \dots & a_{nk}\underline{\underline{I}} \end{bmatrix}$

$\underline{\underline{A}} \quad n \times k$

$$\left[\begin{matrix} a_{11} & 0 & \dots & 0 \\ 0 & a_{11} & & \vdots \\ \vdots & & & 0 \\ 0 & \dots & & a_{11} \end{matrix} \quad \begin{matrix} a_{12} & 0 & \dots & 0 \\ 0 & a_{12} & \ddots & 0 \\ \vdots & & & 0 \\ 0 & \dots & & a_{12} \end{matrix} \right.$$

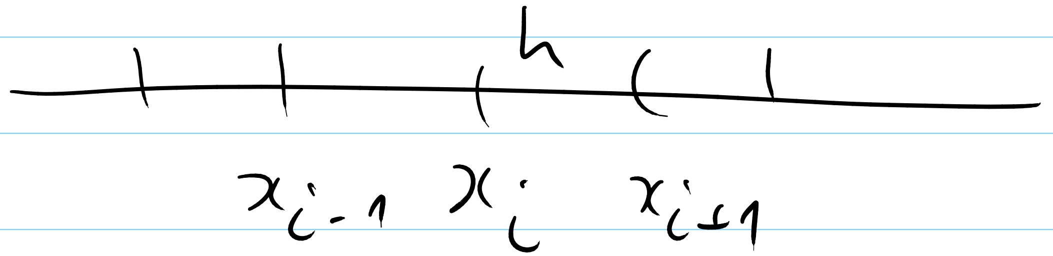
Partial Differential Equations



$\boxed{\frac{\partial}{\partial x}}$ $\frac{\partial}{\partial y}$

$\downarrow$

matrix



$$\underline{u} = \begin{bmatrix} u(x_0) \\ \vdots \\ u(x_{N-1}) \end{bmatrix} \qquad \underline{\underline{A}} \qquad \underline{\underline{D}}_x \qquad \underline{\underline{D}}_y$$

$$\frac{\partial}{\partial x} u(x_i) \simeq \frac{u_{i+1} - u_{i-1}}{2h}$$

$\frac{\partial}{\partial x}$ $\frac{\partial}{\partial y}$

$\underline{D}_x \otimes \underline{D}_y$

29.09.2023

Q3 0.1.11.C.

(Q3.0.1.11.C) [A $\int_0^1 e^x\,dx$-type problem] We know the solution $\mathbf{x} \in \mathbb{R}^n$ and the right-hand-side vector $\mathbf{b} \in \mathbb{R}^n$ of the $n \times n$ (Toeplitz) tridiagonal linear system of equations

$$\begin{bmatrix} \alpha & \beta & 0 & \dots & & \dots & 0 \\ \beta & \alpha & \ddots & & & & \vdots \\ 0 & \beta & \ddots & \ddots & & & \\ \vdots & & \ddots & \ddots & \ddots & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & \ddots & \ddots & \vdots \\ & & & & \ddots & \beta & 0 \\ \vdots & & & & \beta & \alpha & \beta \\ 0 & \dots & & \dots & 0 & \beta & \alpha \end{bmatrix} \mathbf{x} = \mathbf{b}\,.$$

Which overdetermined linear system of equations of maximal size has the vector $[\alpha, \beta]^\top \in \mathbb{R}^2$ as its solution?

$$\begin{cases} \alpha x_1 + \beta x_2 = b_1 \\ \beta x_1 + \alpha x_2 + \beta x_3 = b_2 \\ \beta x_2 + \alpha x_3 + \beta x_4 = b_3 \\ \beta x_3 + \alpha x_4 + \beta x_5 = b_4 \\ \beta x_4 + \alpha x_5 + \beta x_6 = b_5 \\ \vdots \\ \beta x_{n-2} + \alpha x_{n-1} + \beta x_n = b_{n-1} \\ \beta x_{n-1} + \alpha x_n = b_n \end{cases}$$

$$\begin{bmatrix} x_1 & x_2 \\ x_2 & x_1 + x_3 \\ x_3 & x_2 + x_4 \\ x_4 & x_3 + x_5 \\ \vdots & \vdots \\ x_{n-1} & x_{n-2} + x_n \\ x_n & x_{n-1} \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ \\ b_n \end{bmatrix}$$

(Q3.1.1.14.C) Given a matrix $\mathbf{B} \in \mathbb{R}^{m,n}$, a vector $\mathbf{c} \in \mathbb{R}^{m}$, and $\lambda > 0$, define

$$\{\mathbf{x}^*\} := \underset{\mathbf{x}\in\mathbb{R}^n}{\operatorname{argmin}} \|\mathbf{B}\mathbf{x} - \mathbf{b}\|_2^2 + \lambda\|\mathbf{x}\|_2^2 \subset \mathbb{R}^n .$$

looks like a penalisation:

$x \neq 0 \Rightarrow \lambda\|x\|_2^2$ clear increase!

but actually no deeper meaning than for trick

State an overdetermined linear system of equations $\mathbf{A}\mathbf{x} = \mathbf{b}$, of which $\mathbf{x}^*$ is a least-squares solution.

↳ a method for address linear least squares problem for $\underline{B}$ with $\underline{B}$ not full rank

(i.e. (~~some of~~) the columns of $\underline{B}$ are linear dependent)

Pick a small $\lambda > 0$

$$\underline{A} = \begin{bmatrix} \underline{B} \\ \sqrt{\lambda}\,\boxed{\underline{I}_n} \end{bmatrix} \qquad \underline{b} = \begin{bmatrix} \underline{c} \\ \underline{0} \end{bmatrix} \in \mathbb{R}^{m+n}$$

↳ linear indep. columns of $\underline{A}$

$$\lambda\|\underline{x}\|_2^2 = \langle \sqrt{\lambda}\,\underline{I}\,\underline{x}, \sqrt{\lambda}\,\underline{I}\,\underline{x} \rangle$$

$$\underset{\underline{x}\in\mathbb{R}^n}{\operatorname{argmin}} \|\underline{A}\,\underline{x} - \underline{b}\|_2^2$$

$(m+n) \times n$ has Rank n.

$\Rightarrow$ QR-decomposition will work. !

Possible advantage: if $\underline{B}$ is sparse, so is $\underline{A}$, so QR-decomposition via Givens-Rotations might be much less expensive than SVD($\underline{B}$).

Question: difference in advantages CSC / CRS

CSS: good for slicing colons

CRS: good for slicing rows

both are good for internal +, * (pointwise) Matrix * vector (might be faster)

Remark other formats are better for a fast construction.

(Q2.7.1.5.E) For a given matrix $\mathbf{A} \in \mathbb{R}^{m,n}$, $m, n \in \mathbb{N}$, we define the square matrix

$$\mathbf{W}_{\mathbf{A}} := \begin{bmatrix} \mathbf{O}_{m,m} & \mathbf{A} \\ \mathbf{A}^\top & \mathbf{O}_{n,n} \end{bmatrix} \in \mathbb{R}^{m+n,m+n} .$$

Outline the implementation of an efficient C++ function

```
void crsAtoW(std::vector<double> &val,
std::vector<unsigned int> &col_ind,
std::vector<unsigned int> &row_ptr);
```

whose arguments supply the three vectors defining the matrix $\mathbf{A}$ in CRS format and which overwrites them with the corresponding vectors of the CRS-format description of $\mathbf{W}_{\mathbf{A}}$.

Remember that the CRS format of a matrix $\mathbf{A} \in \mathbb{R}^{m,n}$ is defined by

$$\texttt{val}[k] = (\mathbf{A})_{i,j} \quad \Leftrightarrow \quad \begin{cases} \texttt{col\_ind}[k] = j , \\ \texttt{row\_ptr}[i] \leq k < \texttt{row\_ptr}[i+1] , \end{cases} \quad 1 \leq k \leq \mathrm{nnz}(\mathbf{A}) .$$

It may be convenient to use `std::vector::resize(n)` that resizes a vector so that it contains n elements. If n is smaller than the current container size, the content is reduced to its first n elements, removing those beyond (and destroying them). If n is greater than the current container size, the content is expanded by inserting at the end as many elements as needed to reach a size of n using their default value.

Most important: how to implement A^T?

This function does it:

```
CRSMatrix sparse_transpose(const CRSMatrix& input) {
    CRSMatrix res{
        input.m,
        input.n,
        input.nz,
        std::vector<double>(input.nz, 0.0),
        std::vector<int>(input.nz, 0),
        std::vector<int>(input.m + 2, 0) // one extra
    };

    // count per column
    for (int i = 0; i < input.nz; ++i) {
        ++res.rowPtr[input.colIndex[i] + 2];
    }

    // from count per column generate new rowPtr (but shifted)
    for (int i = 2; i < res.rowPtr.size(); ++i) {
        // create incremental sum
        res.rowPtr[i] += res.rowPtr[i - 1];
    }

    // perform the main part
    for (int i = 0; i < input.n; ++i) {
        for (int j = input.rowPtr[i]; j < input.rowPtr[i + 1]; ++j) {
            // calculate index to transposed matrix at which we should p
            const int new_index = res.rowPtr[input.colIndex[j] + 1]++;
            res.val[new_index] = input.val[j];
            res.colIndex[new_index] = i;
        }
    }
    res.rowPtr.pop_back(); // pop that one extra

    return res;
}
```

(Q2.6.0.25.F) [Loss of stability] By direct block-wise Gaussian elimination we found the following solution formulas for a block-partitioned linear system of equations with $\mathbf{D} \in \mathbb{R}^{n,n}$, $\mathbf{c}, \mathbf{b} \in \mathbb{R}^n$, $\alpha \in \mathbb{R}$, $\mathbf{y} \in \mathbb{R}^{n+1}$:

$$\mathbf{A}\mathbf{x} = \begin{bmatrix} \mathbf{D} & \mathbf{c} \\ \mathbf{b}^\top & \alpha \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \xi \end{bmatrix} = \mathbf{y} := \begin{bmatrix} \mathbf{y}_1 \\ \eta \end{bmatrix}, \tag{2.6.0.7}$$

$$\boxed{\xi = \frac{\eta - \mathbf{b}^T \mathbf{D}^{-1} \mathbf{y}_1}{\alpha - \mathbf{b}^\top \mathbf{D}^{-1} \mathbf{c}}}, \quad \mathbf{x}_1 = \mathbf{D}^{-1}(\mathbf{y}_1 - \xi \mathbf{c}). \tag{2.6.0.8}$$

Use these formulas to compute the solution of the 2×2 linear system of equations

$$\begin{bmatrix} \delta & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ \xi \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix},$$

assuming $|\delta| < \frac{1}{2}\mathrm{EPS}$ and using floating point arithmetic.

Hint. Remember that, if $|\delta| < \frac{1}{2}\mathrm{EPS}$, in floating point arithmetic

$$1 \tilde{+} \delta = 1 \quad \text{and} \quad 2 \tilde{+} \delta^{-1} = \delta^{-1}.$$

This is compatible with the "Axiom" of roundoff ana[illegible] Ass. 1.5.3.11

$$\xi = \frac{2 - \frac{1}{\delta}}{1 - \frac{1}{\delta}} = \frac{1}{1 - \delta^{-1}} + 1$$

$$1 + \delta = 1$$

$$-\ 2 - \frac{1}{\delta} = -\frac{1}{\delta^{-1}} \Rightarrow \quad 1 - 2 - \frac{1}{\delta} = 1 - \frac{1}{\delta}$$

$$\underline{2 - \delta^{-1}} = 2 - \underline{(2 + \delta^{-1})} = -\delta^{-1}$$

$$1 - \delta^{-1} = 1 - (2 + \delta^{-1}) = -1 - \delta^{-1}$$

$$\xi = \frac{-\delta^{-1}}{-\delta^{-1} - 1} = \frac{+1}{+1 + \delta} = \frac{1}{1} = 1$$

$$x_1 = \delta^{-1}(1 - 1) = 0 \Rightarrow \begin{bmatrix} -1 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} \delta & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

:(

If we compute "correctly", we get the wrong answer! :)