组　　号	**1**
实验题目	数学实验测试文档
队员姓名	**1. A**
	2. B
	3. C
	4. D

目　录

1 模板说明

1.1 引入模板

使用如下 LaTeX 代码：

```
\documentclass[options]{mathexpthesis}
```

拥有几个可选项：

- **bwprint/colorprint** 控制打印时的颜色.

- **withoutpreface** 控制扉页的输出.
- **withouttitlepage** 控制标题页的输出，最好将摘要放在此页.

1.2 模板参数

- **title** 实验文档标题，在 document 中需要使用 **maketitle**，该参数不必须填写.
- **groupno** 小组编号，该参数必须填写.
- **membera,b,c,d** 成员姓名（仅支持恰好四人的小组），该参数必须填写.

1.3 编译

目前仅支持 XƎLaTeX 编译.

2 简单的使用说明

2.1 摘要与关键字

使用方式：

```
\begin{abstract}
    Abstract here.
    \keywords{\LaTeX, typesetting}
\end{abstract}
```

2.2 节标题

模板中的指令经过了重定义，但使用方法不变：

```
\section{A section}
\subsection{A subsection}
\subsubsection{A subsubsection}
```

具体上，修改了一定的间距，重写了一个计数器指令.

2.3 公式与定理环境

使用了 amsthm, amsmath, amssymb, amsfonts 宏包. 使用方式不变.
行内公式使用 $$ 即可. 行间公式使用 equation, equarray, align 等方式.
请注意英文的拼写：

- lemma 引理

- theorem 定理
- proof 证明
- assumption 假设
- definition 定义
- example 例

注意，此处的计数器是**章节全局性**的计数器.

还有一些其它的定义，不过用处不算太大，此处不介绍.

这里使用一个我曾经写过的例子.

定理 2.1 (Risez 定理) 若 $f_n \xrightarrow{m} f$，则 $\exists \{f_{n_k}\} \subset \{f_n\}, s.t. \quad f_{n_k} \to f_n \quad a.e.$

证明： $f_n \xrightarrow{m} f$，则 $\forall \varepsilon > 0, \delta > 0, \exists n \geqslant 1, s.t. \quad n \geqslant N, m(E(|f_n - f| \geqslant \varepsilon)) < \delta$. 于是 $\forall k \in \mathbb{N}^*$，令 $\varepsilon = \frac{1}{k}, \delta = \frac{1}{2^k}$，可以依次选取自然数 $n_1, n_2, \cdots, n_k, \cdots, s.t.$

$$m(E(|f_{n_k} - f| \geqslant \frac{1}{k})) < \frac{1}{2^k}.$$

下面证明 $f_{n_k} \to f \quad a.e.$

令

$$E_0 = \bigcap_{N=1}^{\infty} \bigcup_{k=N}^{\infty} E(|f_k - f| \geqslant \frac{1}{k}).$$

对每个 $N = 1, 2, \cdots$, 有

$$m(E_0) \leqslant m(\bigcup_{k=N}^{\infty} E(|f_{n_k} - f| \leqslant \frac{1}{k})) \leqslant \sum_{k=N}^{\infty} m(E(|f_{n_k} - f| \geqslant \frac{1}{k})) < \sum_{k=N}^{\infty} \frac{1}{2^k} = \frac{1}{2^{N-1}}.$$

令 $N \to \infty$，可知 $m(E_0) = 0$. 由 De Morgan 公式得，

$$E - E_0 = \bigcup_{N=1}^{\infty} \bigcap_{k=N}^{\infty} E(|f_k - f| \geqslant \frac{1}{k}).$$

所以 $\forall x \in E - E_0, \exists N \geqslant 1, s.t. \quad k \geqslant N$，

$$|f_{n_k}(x) - f(x)| < \frac{1}{k}.$$

故 $f_{n_k}(x) \to f(x)$，则在 E 上 $f_{n_k} \to f \quad a.e.$ □

2.4 图片插入

对于未指定后缀的图片，将以**.pdf**，**.eps**，**.jpg**，**.png** 的顺序查找.

图片支持的目录：

```
{figures/}, {figure/}, {pictures/}, {picture/}, {pic/}, {pics/},
↪   {image/}, {images/}
```

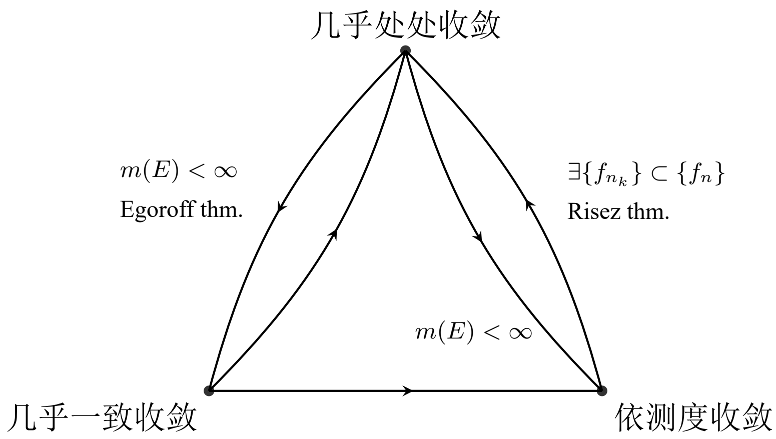

图 1　**Example of TikZ**

插入图片的方式：

```
\begin{figure}
    \centering
    \caption{The caption} % caption above
    \includegraphics{figures/...}
    \label{fig-x}
\end{figure}
```

在 figure 环境中也可以使用 TikZ 作图. 以下是一个例子.

2.5 表格插入

支持简单的表格，推荐使用 Excel2LaTeX 工具一键转换各种表格. 一般表格的创建是用 tabular, table 这样的命令来完成的.

建议采用三线表. 以下是一个三线表的示例.

Type	Biomass($g \cdot m^{-2}$)	Area Required(m^2)
rain forest	45000	6666.67
seasonal forest	35000	8571.43
grassland	4000	75000

表 1　**Minimum area needed in different environment**

LaTeX 代码由插件生成.

```latex
\begin{table}[htbp]
  \centering
    \begin{tabular}{lcc}
    \toprule
    \multicolumn{1}{c}{Type} & Biomass($g \cdot m^{-2}$) & Area
    ↪  Required($m^2$) \\
    \midrule
    rain forest & 45000 & 6666.67 \\
    seasonal forest & 35000 & 8571.43 \\
    grassland & 4000  & 75000 \\
    \bottomrule
    \end{tabular}
  \caption{Minimum area needed in different environment}
  \label{tab:addlabel}
\end{table}
```

3 参考文献

使用以下代码：

```latex
\begin{thebibliography}{99}
    \bibitem{bib:one} ....
    \bibitem{bib:two} ....
\end{thebibliography}
```

使用 \cite 命令可以实现一个“比较大”的引用 [1]. 使用新写的 \upcite 命令则是上标引用[2].

4 附录

使用：

```latex
\appendix
```

即可创建附录.

附录中可以放置一些实验时产生的很占地方的图片，也可以放置代码. 推荐 minted 宏包，需要 Python 安装一些额外文件并将其放入系统的环境变量中：

```
pip install Pygments
```

从代码生成的角度上，minted 的效果优于 lstlistings.

5 其它注意事项

5.1 符号的使用

句号请写成”.”，不要使用“。”.

不需要滥用符号，公式标号与定理环境.

所有符号应使用相应的说明，也需要一个整体的表格概述符号的意义.

5.2 图片与表格的位置

不要忘加 htb.

部分情况不要忘加 **centering**.

5.3 使用许可与鸣谢

模板协议为 LPPL（直接用就行了）.

此文档中除去示例代码的内容，协议为 LPPL（即不要未经许可使用我的 matlab 代码和 tikz 代码）.

由于模板集成了latexstudio/GMCMThesis[3]与latexstudio/CUMCMThesis[4]的文档模板，该部分的版权为版权所属人所有，此模板属于 Derived Work（这是鸣谢）.

5.4 其它

以下是示例的参考文献内容和示例的附录. 如需使用网址，url 宏包可能会更好看一点. 有时 url 可能会导致强制换行的问题，此时可以人工换行.

参考文献

[1] upcite reference example

[2] cite reference example

[3] `https://github.com/latexstudio/GMCMthesis`

[4] `https://github.com/latexstudio/CUMCMthesis`

附录 A 一些 matlab 测试代码

A.1 单纯形法-Core

```
function [xval, fval] = SimplexMethod(c, A, b, epsilon)
    % Simplex method for linear programming.
    % This program is to solve functions like:
    % min f = cx such that Ax = b, x >= 0, b >= 0
    % c, A, b: equation above
    % epsilon: error value when calculating a valid point

    if (nargin == 3)
        epsilon = 1e-6;
    end

    if (size(c, 2) ~= size(A, 2)) || (size(c, 1) ~= 1) || ...
       (size(A, 1) ~= size(b, 1)) || (size(b, 2) ~= 1)
        error("Invalid matrix.");
    end

    % Find a solution first.
    extA = [A, eye(size(A, 1))];
    extB = b;
    extC = [zeros(size(c)), -ones(size(b')), 0];
    inX = size(c, 2) + 1: size(c, 2) + size(A, 1);
    extTable = [extA extB; extC]; % Generate a table
    % Manipulate the table
    for i = 1:size(A, 1)
        extTable(end, :) = extTable(end, :) + extTable(i, :);
    end
    % Pivot, step by step
    [val, cind] = max(extTable(end, 1: end - 1));
    while (val > 0)
        minval = inf;
        for i = 1:size(extTable, 1) - 1
            if (extTable(i, cind) * inf >= 0)
```

```
                val = extTable(i, end) / extTable(i, cind);
                if (val < minval)
                    minval = val;
                    rind = i; % find row index
                end
            end
        end
        if (isinf(minval))
            error("Unable to find a valid initial point.");
        end
        point = extTable(rind, cind); % find the point
        inX(rind) = cind; % calculate inner columns at the same time
        extTable(rind, :) = extTable(rind, :) ./ point;
        % Elimination
        for i = 1:size(extTable, 1)
            if (i ~= rind)
                extTable(i, :) = extTable(i, :) - extTable(i, cind) .*
                ↪  extTable(rind, :);
            end
        end
        [val, cind] = max(extTable(end, 1: end - 1));
    end
    if (abs(extTable(end, end)) >= epsilon)
        error("Unable to find a valid initial point.");
    else
        tempX = zeros(size(extTable, 2), 1);
        tempX(inX) = extTable(1: end - 1, end);
        initX = tempX(1: size(c, 2)); % get initial point of original LP
    end
    % change extTable to let useful vectors in
    for i = 1: size(A, 1)
        if (inX(i) > size(c, 2)) % move condition
            ckey = find(extTable(i, 1: size(c, 2)) ~= 0, 1);
            if (~isempty(ckey)) % get the key, apply the change
                point = extTable(i, ckey);
```

```
            inX(i) = ckey;
            extTable(i, :) = extTable(i, :) ./ point;
            % Elimination without the bottom
            for j = 1:size(extTable, 1) - 1
                if (j ~= i)
                    extTable(j, :) = extTable(j, :) - extTable(j,
                     ↪  ckey) .* extTable(i, :);
                end
            end
        else % don't get the key, remove the line
            warning("Surplus condition found. Removing it.")
            inX(i) = -1; % Tag the key
        end
    end
end
table = extTable(inX > 0, [1: size(A, 2), end]); % construct table
% re-calculate conditional number
table = [table; zeros(1, size(table, 2))]; % make room first
table(end, end) = c * initX;
inX = inX(inX > 0);
w = c(inX) / table(1: end - 1, inX); % get factor w
for j = 1: size(table, 2) - 1
    table(end, j) = w * table(1: end - 1, j) - c(j);
end
% Pivot, step by step
[val, cind] = max(table(end, 1: end - 1));
while (val > 0)
    minval = inf;
    for i = 1:size(table, 1) - 1
        if (table(i, cind) * inf >= 0)
            val = table(i, end) / table(i, cind);
            if (val < minval)
                minval = val;
                rind = i; % find row index
            end
```

```
            end
        end
        if (isinf(minval))
            error("This problem is unbounded.")
        end
        point = table(rind, cind); % find the point
        inX(rind) = cind; % calculate inner columns at the same time
        table(rind, :) = table(rind, :) ./ point;
        % Elimination
        for i = 1:size(table, 1)
            if (i ~= rind)
                table(i, :) = table(i, :) - table(i, cind) .*
                ↪  table(rind, :);
            end
        end
        [val, cind] = max(table(end, 1: end - 1));
    end
    tempX = zeros(size(table, 2), 1);
    tempX(inX) = table(1: end - 1, end);
    xval = tempX(1: size(c, 2)); % get initial point of original LP
    fval = table(end, end);
end
```

A.2 单纯形法-Full

```
function [xval, fval] = SimplexSolver(c, A, b, Aeq, beq, lb, ub, epsilon)
    % Uniform simplex solver for simplex method.
    % Solves problem like:
    % min f = cx, such that A * x <= b, Aeq * x = beq, lb <= x <= ub
    % epsilon: error value when calculating a valid point
    % This program works in larger range than the core simplex method.

    % Autofill
    if (nargin == 3)
        Aeq = []; beq = []; lb = []; ub = []; epsilon = 1e-6;
    elseif (nargin == 5)
```

```
    lb = []; ub = []; epsilon = 1e-6;
elseif (nargin == 6)
    ub = []; epsilon = 1e-6;
elseif (nargin == 7)
    epsilon = 1e-6;
end
origC = c;

% Validate
if (~isempty(A) && ~isempty(b) && ...
   ((size(c, 2) ~= size(A, 2)) || (size(c, 1) ~= 1) || ...
   (size(A, 1) ~= size(b, 1)) || (size(b, 2) ~= 1))) || ...
   (~isempty(Aeq) && ~isempty(beq) && ((size(c, 2) ~= size(Aeq, 2))
   ↪  || ...
   (size(Aeq, 1) ~= size(beq, 1)) || (size(beq, 2) ~= 1))) || ...
   (~isempty(lb) && (size(lb, 2) ~= 1 || size(lb, 1) ~= size(c, 2)))
   ↪  || ...
   (~isempty(ub) && (size(ub, 2) ~= 1 || size(ub, 1) ~= size(c, 2)))
    error("Invalid matrix.");
end

% Integrate lowerbounds and upperbounds into constraints
if (isempty(lb))
    lb = -inf(size(c))';
end
if (isempty(ub))
    ub = +inf(size(c))';
end

% create movement table for recovering the data later
% line 1 demonstrates movement type
% value 0 for x - u type + v - x type
% value 1 for x - u type
% value 2 for v - x type
% value 3 for x1 - x2 type
```

```
% line 2 demonstrates movement place
% value n for place n
% value -1 for not taking a place

movement_table = zeros(2, size(c, 2));
place = size(c, 2) + 1;
for i = 1: size(c, 2)
    if (ub(i) - lb(i) < 0)
        error("lowerbound is greater than upperbound.");
    end
    if (lb(i) ~= -inf) && (ub(i) ~= +inf)
        movement_table(1, i) = 0;
        movement_table(2, i) = -1;
        % Add constraint for full bounded condition
        if (~isempty(A))
            b(:) = b(:) - A(:, i) * lb(i);
        end
        if (~isempty(Aeq))
            beq(:) = beq(:) - Aeq(:, i) * lb(i);
        end
        temprow = zeros(size(c));
        temprow(i) = 1;
        A = [A; temprow];
        b = [b; ub(i) - lb(i)];
    elseif (lb(i) ~= -inf) && (ub(i) == +inf)
        movement_table(1, i) = 1;
        movement_table(2, i) = -1;
        if (~isempty(A))
            b(:) = b(:) - A(:, i) * lb(i);
        end
        if (~isempty(Aeq))
            beq(:) = beq(:) - Aeq(:, i) * lb(i);
        end
    elseif (lb(i) == -inf) && (ub(i) ~= +inf)
        movement_table(1, i) = 2;
```

```
        movement_table(2, i) = -1;
        c(i) = -c(i);
        if (~isempty(A))
            b(:) = b(:) - A(:, i) * ub(i);
            A(:, i) = -A(:, i);
        end
        if (~isempty(Aeq))
            beq(:) = beq(:) - Aeq(:, i) * ub(i);
            Aeq(:, i) = -Aeq(:, i);
        end
    elseif (lb(i) == -inf) && (ub(i) == +inf)
        movement_table(1, i) = 3;
        movement_table(2, i) = place;
        place = place + 1;
        % Add variable for unbounded condition
        c = [c, -c(i)];
        if (~isempty(A))
            A = [A, -A(:, i)];
        end
        if (~isempty(Aeq))
            Aeq = [Aeq, -Aeq(:, i)];
        end
    end
end

% Add surplus variables for inequalities and make it stardardized
for i = 1: size(A, 1)
    tempcol = zeros(size(A, 1), 1);
    if b(i) < 0
        A(i, :) = -A(i, :);
        b(i) = -b(i);
        tempcol(i) = -1;
        A = [A, tempcol];
        c = [c, 0];
    else
```

```
            tempcol(i) = 1;
            A = [A, tempcol];
            c = [c, 0];
        end
        if (~isempty(Aeq))
            Aeq = [Aeq, zeros(size(Aeq, 1), 1)];
        end
    end

    % Make equality constraints startardized
    for i = 1: size(Aeq, 1)
        if (beq(i) < 0)
            Aeq(i, :) = -Aeq(i, :);
            beq(i) = -beq(i);
        end
    end

    % Solve the LP using the core simplex method module
    [tempxval, ~] = SimplexMethod(c, [A; Aeq], [b; beq], epsilon);

    xval = nan(size(origC, 2), 1);
    % From movement_table to recover final value
    for i = 1: size(movement_table, 2)
        switch (movement_table(1, i))
            case {0, 1}
                xval(i) = tempxval(i) + lb(i);
            case 2
                xval(i) = ub(i) - tempxval(i);
            case 3
                xval(i) = tempxval(i) - tempxval(movement_table(2, i));
        end
    end
    fval = origC * xval;

end
```