

通用报告模板

清华大学学位论文 L^AT_EX 模板
使用示例文档

Xue Ruini 12345678

January, 2019

南京大學

通用报告模板

清华大学学位论文 L^AT_EX 模板 使用示例文档

系 别：计算机科学与技术系

专 业：计算机科学与技术

姓 名：薛瑞尼

指导教师：郑纬民教授

辅导教师：陈文光教授

二〇一九年一月

摘 要

论文的摘要是对论文研究内容和成果的高度概括。摘要应对论文所研究的问题及其研究目的进行描述，对研究方法和过程进行简单介绍，对研究成果和所得结论进行概括。摘要应具有独立性和自明性，其内容应包含与论文全文同等量的主要信息。使读者即使不阅读全文，通过摘要就能了解论文的总体内容和主要成果。

论文摘要的书写应力求精确、简明。切忌写成对论文书写内容进行提要的形式，尤其要避免“第 1 章……；第 2 章……；……”这种或类似的陈述方式。

本文介绍清华大学论文模板 `ThuThesis` 的使用方法。本模板符合学校的本科、硕士、博士论文格式要求。

本文的创新点主要有：

- 用例子来解释模板的使用方法；
- 用废话来填充无关紧要的部分；
- 一边学习摸索一边编写新代码。

关键词是为了文献标引工作、用以表示全文主要内容信息的单词或术语。关键词不超过 5 个，每个关键词中间用分号分隔。（模板作者注：关键词分隔符不用考虑，模板会自动处理。英文关键词同理。）

关键词：`TEX`；`LATEX`；CJK；模板；论文；`TEX`；`LATEX`；CJK；模板；论文

目 录

0.1 出题	6
0.2 首试	7
0.3 突破	8
0.4 真神技也!	10
0.5 面试之外: MATLAB 的反除函数	11
0.6 结语	12
第 1 章 带 English 的标题	13
1.1 封面相关	13
1.2 字体命令	13
1.3 表格样本	14
1.3.1 基本表格	14
1.3.2 复杂表格	15
1.3.3 其它	18
1.4 定理环境	19
1.5 参考文献	22
1.6 公式	23
第 2 章 中华人民共和国	25
2.1 其它例子	25
2.1.1 绘图	25
2.1.2 插图	25
第 3 章 实现功能	29
3.1 硬件部分	29
3.2 软件部分	29
3.3 其它	29
第 4 章 组内分工	30
4.1 郑樊巍同学	30
4.2 沈天琪同学	30

第 5 章 硬件部分	31
5.1 五级流水线 CPU 的结构	31
5.1.1 取指模块	31
5.1.2 译码模块	31
5.1.3 执行模块	33
5.1.4 访存模块	33
5.1.5 寄存器与回写模块	34
5.2 指令实现	34
5.2.1 逻辑指令	34
5.2.2 移位指令	35
5.2.3 算术指令	35
5.2.4 跳转指令	35
5.2.5 访存指令	36
5.3 指令测试	37
5.4 ROM, RAM 与 IO	37
5.4.1 储存空间构建与 IO 配置	37
5.4.2 MMIO	38
5.5 运行时环境 AM	38
5.6 遇到的问题与解决方案	40
5.7 实验启示	40
第 6 章 软件部分	41
6.1 操作系统的定义	41
6.2 操作系统中的约定	41
6.3 屏幕显示功能	41
6.4 斐波那契数	42
6.5 Hello World	42
6.6 电子琴	42
6.7 单步调试	42
6.8 遇到的问题与解决方案	44
6.9 实验启示	44
第 7 章 文件结构	45
第 8 章 代码样式	46
插图索引	48

表格索引	49
公式索引	50
参考文献	51
致 谢	53
声 明	54
附录 A 外文资料原文	55
A.1 Single-Objective Programming	55
A.1.1 Linear Programming.....	56
A.1.2 Nonlinear Programming.....	57
A.1.3 Integer Programming	58
附录 B 外文资料的调研阅读报告或书面翻译	60
B.1 单目标规划	60
B.1.1 线性规划.....	60
B.1.2 非线性规划	61
B.1.3 整数规划.....	62
附录 C 其它附录	63

主要符号对照表

HPC	高性能计算 (High Performance Computing)
cluster	集群
Itanium	安腾
SMP	对称多处理
API	应用程序编程接口
PI	聚酰亚胺
MPI	聚酰亚胺模型化合物, N-苯基邻苯酰亚胺
PBI	聚苯并咪唑
MPBI	聚苯并咪唑模型化合物, N-苯基苯并咪唑
PY	聚吡咙
PMDA-BDA	均苯四酸二酐与联苯四胺合成的聚吡咙薄膜
ΔG	活化自由能 (Activation Free Energy)
χ	传输系数 (Transmission Coefficient)
E	能量
m	质量
c	光速
P	概率
T	时间
v	速度
劝学	<p>君子曰：学不可以已。青，取之于蓝，而青于蓝；冰，水为之，而寒于水。木直中绳。輮以为轮，其曲中规。虽有槁暴，不复挺者，輮使之然也。故木受绳则直，金就砺则利，君子博学而日参省乎己，则知明而行无过矣。吾尝终日而思矣，不如须臾之所学也；吾尝跂而望矣，不如登高之博见也。登高而招，臂非加长也，而见者远；顺风而呼，声非加疾也，而闻者彰。假舆马者，非利足也，而致千里；假舟楫者，非能水也，而绝江河，君子生非异也，善假于物也。积土成山，风雨兴焉；积水成渊，蛟龙生焉；积善成德，而神明自得，圣心备焉。故不积跬步，无以至千里；不积小流，无以成江海。骐骥一跃，不能十步；弩马十驾，功在不舍。锲而舍之，朽木不折；锲而不舍，金石可镂。</p>

蚓无爪牙之利，筋骨之强，上食埃土，下饮黄泉，用心一也。蟹六跪而二螯，非蛇鳝之穴无可寄托者，用心躁也。——荀况

清华大学学位论文 L^AT_EX 模板

使用示例文档

姓名：薛瑞尼


学号：12345678

日期：2019 年 1 月 30 日

PART I HOMEWORK

❓ 问题 26.1.1

We will show that each

 解： We will show that each flow in G' is mapping to a flow of same value in G and vice versa. Thus the maximum flow in G' has the same value as a maximum flow in G . this


For a flow in G , consider the flow f , in which there exist antiparallel edge $v_i \rightarrow v_j$ and $v_j \rightarrow v_i$. Suppose the flow on $v_i \rightarrow v_j$ is F and F' in another direction. The flowmargin- f' in G' has same flow in each edge except that $v_i \rightarrow v_j$ is replaced by $v_i \rightarrow v'$ andpar $v' \rightarrow v_j$ and the flow is F . It is obvious that Capacity constraint and Flow conservation meets. The value of f' is

$$|f'| = \sum_{v \in V'} f'(s, v) - \sum_{v \in V'} f'(v, s)$$

If there is no antiparallel edges incident with s , the value is same. Otherwise, $f(s, v) = f'(s, v')$ the value is still the same.

For the other side, due to Flow conservation, $f'(v_1, v') = f'(v', v_2)$. By deleting v' and add edge $v_1 \rightarrow v_2$ with $f(v_1, v_2) = f'(v_1, v')$ we can get the graph G . Thus we have built a bijection $G : \mathbb{F} \rightarrow \mathbb{F}'$ where \mathbb{F} denotes all possible flow and with the same value. Hence we have proved this problem.

❓ 问题 26.1.2


 解： Deleting the source and the sink, we get a unique multiple-source, multiple-sink flow network.

For the other side, for a multiple-source, multiple-sink flow network, we add two kinds of edges $s \rightarrow s_i$ and $t \rightarrow t_i$. We will prove the flow on each kind of each is unique.

For the first kind, by flow conservation, $f(s, s_i) = \sum f(s_i, v)$, where v is vertices incident with s_i except s . Thus the value is unique. The same proof applied for the second kind.


Thus we have proved the problem.

问题 26.1.6

 解: Build the graph in which corners are vertices and edges (u, v) and (v, u) are added if there is a road connecting them. Constrain the edge to have capacity 1, and the problem is reverted to find a maximum flow and check if it can be 2 and with integer flow value in each edge.


However, this leads to a problem. The two may take a road in different direction, namely one pass (u, v) and the other (v, u) . A direct algorithm is to exhaust all possible direction of each edge and run the above procedure, namely for one road connecting u, v , adding edge (u, v) in one time and (v, u) in another. However, this costs a lot of time, I wonder if there is a better way.

问题 26.1.7

 解: Build network G' in the following way: for a vertex $v \in G$ with vertex constraint C , add two vertices v_1, v_2 into G' , all edge connected from u to v will lead to edge $u \rightarrow v_1$ in G' and all edge connected from v to u will lead to edge $v_2 \rightarrow u$ in G' . Two edges $(v_1, v_2), (v_2, v_1)$ with capacity C is also added, which has flow $\sum_u f(u, v) - \sum_u f(v, u)$. In this way, the constraint of edge between v_1 and v_2 serves as constraint of v .

The graph G' has $2|V|$ vertices and $|E| + |V|$ edges.

问题 26.2.2

 解: The flow is $11 + 1 - 4 + 7 + 4 = 19$, the capacity of this cut is $16 + 4 + 7 + 4 = 31$

问题 26.2.6

✍ 解: We construct a new flow network by adding a super source s with edge (s, s_i) of capacity p_i and a super sink t with edge (t_i, t) of capacity q_i . We will prove the original problem is to find the maximum flow in the new network and check whether it can be $sum p_i$

First, by conservation of flow in s_i , we have $\sum_{v \in V} f(s_i, v) = f(s, s_i) = p_i$. The same proof applies for each t_i . Sum over all flow in and out of $V - \{s, t\}$, we get $\sum_i f(s, s_i) = \sum_i f(t_i, t)$. In the case we find the maximum flow in the new network and check whether it can be $sum p_i$, we can have $\sum_i p_i = \sum_j q_j$. So the conversion is right.

❓ 问题 26.2.8

✍ 解: To prove this problem, we will prove that in the while loop, if we find the path p from s to t will never contain edge (v, s) . Otherwise, since the path begins at s and reaches s again, thus the path is not a simple path which contradict the definition of augment path.

Since (v, s) will not appear in any augment path, the path finded each time by Ford-Fulkerson method remains the same. Thus this procedure is not influenced and can get the right result.

❓ 问题 26.2.10

✍ 解: Suppose we already find a maximum flow f . Make a new network where the capacity of each edge equals the flow in the original network. We only need to prove by $|E|$ times of finding augment path we can find the maximum flow of the new network. Note that the maximum flow of the new network is full on every edge.

Run Ford-Fulkerson algorithm on the new network with one modification that each time when we find the minimal capacity of n argument path, instead of update the corresponding edges in residual network, we just delete (u, v) . This will not influence the augment path finded afterwards for it is has the same value of the final flow. Otherwise, there is an augment path contain (v, u) , then another augment path containing (u, v) must exist. Denote the first as $s \rightarrow v \rightarrow u \rightarrow t$ and the other $s \rightarrow u \rightarrow v \rightarrow t$, we can construct two augment path $s \rightarrow t$ without traversing (u, v) .

Since each time an edge is deleted, at most $|E|$ augment path can be finded. In this way, we can get at most $|E|$ augment path to get a maximum flow.

问题 26.2.12

解： Since there is an edge (v,s) that $f(v,s)=1$, we hope to find a circle from s to s that there is flow on each edge. Since the capacity is all integer, if $f(v,u) > 0$ then $f(v,u) \geq 1$. Thus by deleting 1 from flow on the edges of the circle, we get a flow f' with the same value of f . In a flow, each vertex is connected with s , so is v . Thus there exists circle $s \sim v \rightarrow s$.

To get f' , we use a deep-first search on f starting at s . When we find this circle, we can perform above operations to get the desired flow f' .

问题 26.2.13

解： Add $\frac{1}{2|E|}$ to each edge to form a network G' . To prove the minimal cut in G' is the minimal cut in G with the minimal edges, we will prove that the minimal cut in G' has corresponding minimal value of cut in G and in all minimal value cut it contains the least edges.

For the first thing, for all the cut $f(S, T)$ in G , the corresponding cut in G' is $f(S, T) + \frac{1}{2|E|}E(S, T) < f(S, T) + 1$. Since the flow is integer, two flow with different value differ at least one. Thus the minimal one remains minimal.

For the second thing, the minimal cut in G' has minimal value $f(S, T) + \frac{1}{2|E|}E(S, T)$. Since $f(S, T)$ is the same for minimal cut in G , thus $E(S, T)$ is minimal, namely it contains minimal edges in G .

问题 26.3.3

解： The augment path is a simple path. Since the graph is a partite graph, the path must go like: $s \rightarrow l_1 \rightarrow r_1 \cdots \rightarrow t$. Thus the over bounding of the augment path is $2 \min(V(L), V(R)) + 1$.

问题 26-1

✍ 解: (a). We have already discussed this problem in 26.1.7. The graph G' has $2|V|$ vertices and $|E| + |V|$ edges.

(b) Construct the flow network as follows: add super source s and edges connecting s to all initial points. If two node in the grid can be reached from each other, then add edges in the flow. Finally, add edges from boundary point to super sink t . All edges and vertices have unit capacity. Then the problem converted to determine whether the maximum flow can be m .

With the relabeled-to-front algorithm, the running time is $O(V^3) = O(n^6)$.

❓ 问题 26-2

✍ 解: (a). Construct the network G' as suggested and let all edges have capacity 1. This network solves a matching problem and we have the partite graph X, Y . We can construct a path cover from a flow as follows:

- select a node not in path unless all nodes are in the path.
- if there is no $f(x_i, y_j) > 0$, put it into path cover as a path of length.
- else put x_i, y_j into path, and select x_j back to step two.

In this way, we construct a path for each flow. Actually, each edge in the path cover is an edge in the matching of the partite network. Thus we get $|V| = \sum_{p \in P} v(p) = \sum_{p \in P} 1 + e(p) = P + \sum_p e(p) = |P| + M = |P| + |f|$ (Note the graph is DAG). When $|f|$ is maximal, we will get the minimal path cover.

(b) If the graph contains circle, the deduction $\sum_{p \in P} v(p) = \sum_{p \in P} 1 + e(p)$ will not hold.

An counter-example is graph with $E = \{(1, 2), (2, 3), (3, 4), (4, 1)\}$. With our algorithm, (x_1, y_2) and (x_4, y_1) can be either chosen. If the latter one is chosed, then a false answer of 2 is generated. Thus in this case we cannot guarantee the algorithm to get the right answer.

PART II CORRECTION

PART III FEEDBACK

- Question in 26.1.6

0.1 出题

话说当今计算机业事，机器学习当道，码农趋之若鹜。遇投机者，以数月速学之，可夸夸其谈不下数个时辰，然则凡细节、原理之处，一概不通。更有好事者炒作概念，一时间机器学习从业者良莠不齐。为探求职者功力之深，G 公司取一巧计，出题如下：

线性方程组：

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ \cdots & \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &= b_m \end{cases}$$

可矩阵表示为 $\mathbf{Ax} = \mathbf{b}$ 其中 $\mathbf{A} \in \mathbf{M}_{m \times n}$, $\mathbf{x}, \mathbf{b} \in \mathbb{R}^n$ 。在 $m > n$ 时，上述方程组多数情况下无解。但如果允许引入残差，将方程求解问题转化为最小化残差和问题，则仍可解。

引入定义在 \mathbb{R}^n 上的函数：残差平方和 $S(\mathbf{x}) = \|\mathbf{Ax} - \mathbf{b}\|^2$ 。由此给出最小二乘解的形式化定义：对于线性方程组 $\mathbf{Ax} = \mathbf{b}$ ，求最小二乘解 $\hat{\mathbf{x}} = \operatorname{argmin}(S(\mathbf{x}))$ ，即使 $S(\mathbf{x})$ 最小的 \mathbf{x} 。这便是经典的最小二乘问题。

问题 0.1： 试计算最小二乘解，并考虑提供算法的实际应用情况

解 0.1： 这里只考虑较为简单的情况。首先设 \mathbf{A} ：行向量分别为 a_1, a_2, \dots, a_m ，则有

$$\begin{aligned} S(\mathbf{x}) &= \|\mathbf{Ax} - \mathbf{b}\|^2 \\ &= \sum_{i=1}^m (a_i x - b_i)^2 \\ &= \operatorname{tr}((\mathbf{Ax} - \mathbf{b})^\top (\mathbf{Ax} - \mathbf{b})) \end{aligned}$$

将求导运算推广到矩阵的迹上：下证对矩阵 $\mathbf{A} \in \mathbf{M}_{m \times n}, \mathbf{B} \in \mathbf{M}_{n \times m}$ ，有

$$\frac{\partial \operatorname{tr}(\mathbf{AB})}{\partial \mathbf{A}} = \frac{\partial \operatorname{tr}(\mathbf{BA})}{\partial \mathbf{A}} = \mathbf{B}^\top$$

首先，有 $\operatorname{tr}(\mathbf{AB}) = \sum_{i=1}^m \sum_{j=1}^n a_{ij} b_{ji}$ ，进而 $\frac{\partial \operatorname{tr}(\mathbf{AB})}{\partial a_{ij}} = b_{ji}$ 。从而可得上述命题。

并且类似地，我们有

$$\frac{\partial \text{tr}(\mathbf{A}^\top \mathbf{B})}{\partial a_{ij}} = \frac{\partial \text{tr}(\mathbf{B} \mathbf{A}^\top)}{\partial a_{ij}} = \mathbf{B}$$

进而

$$\begin{aligned} \frac{\partial S(\mathbf{x})}{\partial \mathbf{x}} &= \frac{\partial \text{tr}((\mathbf{A}\mathbf{x} - \mathbf{b})^\top (\mathbf{A}\mathbf{x} - \mathbf{b}))}{\partial \mathbf{x}} \\ &= \frac{\partial \text{tr}(\mathbf{x}^\top \mathbf{A}^\top \mathbf{A} \mathbf{x} - \mathbf{b}^\top \mathbf{A} \mathbf{x} - \mathbf{b} \mathbf{x}^\top \mathbf{A}^\top + \mathbf{b}^\top \mathbf{b})}{\partial \mathbf{x}} \\ &= 2(\mathbf{A}^\top \mathbf{A} \mathbf{x} - \mathbf{A}^\top \mathbf{b}) \end{aligned}$$

令导数为 0，得到 $\mathbf{A}^\top \mathbf{A} \mathbf{x} - \mathbf{A}^\top \mathbf{b} = 0$ ， $\mathbf{A}^\top \mathbf{A}$ 可逆时，有

$$\mathbf{x} = (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top \mathbf{b}$$

0.2 初试

三人之中，最先面试者为贾有才。其方见此题，心中窃喜，遂写下 MATLAB 代码：

解 0.2: `$\mathbf{x} = \text{inv}(\mathbf{A}' * \mathbf{A}) * \mathbf{A}' * \mathbf{b};$` //MATLAB codes

贾抬头，却见面试官微微摇头。贾大恐，遂解释其中数学推导云云，皆与上同，不复再言。然面试官不以为意。贾面试之结果可知也。

贾之解法固然正确，何不得面试官之心耶？

可以看到，在这种直接计算最小二乘解的过程中，我们进行了 3 次矩阵乘法，和一次矩阵求逆。矩阵乘法利用 $O(n^3)$ 的暴力做法已足够快，即使利用 $O(n^{2.8})$ 算法却因常数巨大、实现复杂且无法降低整体复杂度而常不被采用。虽然在矩阵乘法的优化空间小，矩阵求逆却有比最简单的高斯消元法更好的选择。

回顾高斯消元求逆，假设矩阵为 n 阶方阵，一般我们先进行下三角消元。对于第 i 个主元，进行下三角消元时我们需要进行 $(n-i) * (n+1-i)$ 次乘法和加法（消 $n-i$ 行，每一行 $n+1-i$ 个非 0 元素），所以将一次乘法和加法作为单位时间后，进行下三角消元消耗的时间为

$$\sum_{i=1}^n (n-i) * (n+1-i) = \sum_{i=0}^{n-1} i * (i+1) = \frac{(n+1)n(n-1)}{3}$$

而类似的，进行上三角消元时，我们要进行 n 次行变换。但由于此时每行只有一个非 0 元，所以总消耗为

$$\sum_{i=0}^{n-1} i = \frac{(n-1) * n}{2}$$

但是计算矩阵逆时同时要对 I_n 进行相同的行变换，由于操作完全对称，所以将最后时间乘二，我们就得到了高斯消元求逆的一个比较精确的时间相对值：

$$T(n) = \left(\frac{(n+1)n(n-1)}{3} + \frac{(n-1) * n}{2} \right) * 2 = \frac{2n^3 + 3n^2 - 5n}{3} = \frac{2}{3}n^3 + O(n^2)$$

而用 A^{-1} 求矩阵逆采用是最朴素的方法，其相对时间和上述计算相差无几。实际上存在更好的方法。

0.3 突破

其后面试者为甄汇。审题有饷，方落笔

解 0.3: $x = (A' * A) \backslash (A' * b); // \text{MATLAB codes}$

面试官微颌首。此句何意也？

在 *MATLAB* 中， \backslash 是矩阵左除的意思。即左乘矩阵的逆。在早期的 *MATLAB* 版本中，左除判断矩阵是否能够 *LU* 分解，若可以则使用 *LU* 分解求逆。

该算法的思路为，将待求逆矩阵 A 分解成下三角矩阵 L 与 U 的乘积，再通过求解两个上（下）三角形方程组进行求解：

$$LUx = b \Rightarrow \begin{cases} Ly = b \\ Ux = y \end{cases}$$

对于下三角形方程组，易知有 $O(n^2)$ 的计算方法 $x_i = \frac{b_i - \sum_{j=1}^{i-1} l_{i,j} x_j}{l_{i,i}}$ 。若 *LU* 分解能在 $O(n^3)$ 完成，则渐进时间复杂度与高斯消元相同。这看起来好像是画蛇添足，但实际不然。

首先有如下命题：

命题 0.1: 设方阵 A 的各阶顺序主子式都非 0，则存在下三角矩阵 L 与上三角矩阵 U 使得 $A = L * U$ 。

为了证明这个命题，我们有如下引理

引理 0.1: 若 \mathbf{A} 与 \mathbf{B} 都是 n 阶上（下）三角方阵，则 $\mathbf{A} * \mathbf{B}$ 也是 n 阶上（下）三角方阵。

此处略去证明，因此引理只需用矩阵乘法的定义验证即可，即验证 $\mathbf{A} * \mathbf{B}$ 的下方（上方）元素都为 0，而这是显然的。这样我们就可以给出命题的证明

证明 考虑对 \mathbf{A} 进行高斯消元。注意到一个事实：进行下三角消元时，不需要用到行交换。也就是说，每一行在向下消元后，下一行的主元只会向右移一位。假设第 i 次消元将下一行主元右移了两位以上，则考虑 \mathbf{A} 的 $i+1$ 顺序主子式，其行向量线性相关，为 0，矛盾。（这里同时也意味着 $\mathbf{A}_{1,1} \neq 0$ ）

由此知此高斯消元相当于左乘上三角初等阵 \mathbf{P}_i ，而 \mathbf{P}_i 为下三角初等阵。于是 $\mathbf{A} = \mathbf{P}_i^{-1} * \mathbf{P}_i * \mathbf{A} = \mathbf{P}_i^{-1} * \mathbf{A}^\top$ ，此为消元过程。经过 n 次消元后， \mathbf{A} 变为上三角阵，左乘了 n 个下三角初等阵，所以 $\mathbf{A} = \mathbf{P} * \mathbf{A}^\top$ ，即 $\mathbf{A} = \mathbf{L} * \mathbf{U}$ 。证毕 \square

如何快速计算 \mathbf{L} 与 \mathbf{U} 呢？假设 \mathbf{L} 的对角元素都为 1（对应杜立特分解），下面给出一种递推算算法：

首先由 $\mathbf{L} * \mathbf{U} = \mathbf{A}$ ，设 $\mathbf{L} = (l_{i,j})$, $\mathbf{U} = (u_{i,j})$, $\mathbf{A} = (a_{i,j})$ ，显然有 $u_{1,j} = a_{1,j}$ ，从而 $l_{i,1} = \frac{a_{i,1}}{u_{1,1}}$ 。得到这两个初始条件后，如下递推计算剩下元素：

$$u_{r,j} = \frac{a_{r,j} - \sum_{k=1}^{r-1} l_{i,k} u_{k,j}}{l_{r,r}} = a_{r,j} - \sum_{k=1}^{r-1} l_{i,k} * u_{k,j}$$

$$l_{i,r} = \frac{a_{i,r} - \sum_{k=1}^{r-1} l_{i,k} u_{k,r}}{u_{r,r}}$$

此法复杂度如何？注意到对于 l_{ir} ，需要最多 r 次乘除运算。 u_{rj} 计算的时间复杂度与 l_{ir} 相当。总时间复杂度为

$$2 * \sum_{i=1}^n \sum_{r=1}^{n-i} r = 2 * \sum_{i=1}^n \frac{i(i-1)}{2} = \sum_{i=1}^n i^2 + O(n^2) = \frac{1}{3}n^3 + O(n^2)$$

再者是是是两个三角形矩阵对应的方程求解，如上分析，时间为 $O(n^2)$ 将所有步骤复杂度相加，可以发现相对于普通的高斯消元而言，其最高项系数确实减少了：

$$T(n) = \frac{1}{3}n^3 + O(n^2)$$

在实际应用中， LU 分解的高处甚于此，更有

- 在 MATLAB 代码的具体实现中, LU 分解 (可以分解的情况下) 解方程组的理论时间复杂度系数是高斯消元的 $\frac{1}{3}$ 。当 $n = 1000$ 时, 平均每次高斯消元耗时 30s, LU 分解法耗时 22s。(机器性能类于学校机房机)
- 节省内存开销。根据 LU 分解时的递推方法, 我们不需要同时保存所有信息在内存中, 求出 u_{rj} 后 $a_{rj}, j \geq r$ 的储存位置不再需要, 求出 l_{ir} 后 $a_{ir}, i \geq r+1$ 的储存位置不再需要。对于超大矩阵, 这可以节省大量内存空间。
- 对于求解 $Ax = b$ 其中 A 固定, b 取多个值的问题, 同过一次 $O(n^3)$ 的 LU 分解, 每个 b 只需 $O(n^2)$ 的三角矩阵求解即可。当 b 的数量 m 足够大时, 时间复杂度为 $O(mn^2)$ 优于 $O(mn^3)$ 。

0.4 真神技也!

最后登场者为超勇也。其人气宇轩昂, 仪表不凡, 洋洋洒洒落笔写道:

解 0.4: //MATLAB code

$$\begin{aligned} [Q, R] &= qr(A); \\ x &= R \setminus (Q' * b); \end{aligned}$$

面试官见此代码, 有面露疑色而待解释者, 有舒容展眉而称赞者。何以如此也? 此非习线代者所不能得之方法也。可见此人定非碌碌之辈, 或将效大力于公司。

这首行代码, 用的是 QR 分解, 又称正交三角分解。它是把一个实可逆矩阵 A 化成正交矩阵 Q 和实非奇异上三角矩阵 R 的乘积, $A = QR$ 。对于稀疏矩阵, 可以使用 Givens 方法, 对于一般矩阵, 可以使用 Householder 方法求解。对于 n 阶矩阵, 其时间复杂度为 $O(n^3)$ 。篇幅有限, 这里不展开讨论, 有兴趣者可自行搜索相关资料。

对于正交矩阵 Q , 我们有 $Q^T = Q^{-1}$, 故问题可转换为求 $Rx = Q^T b$ 。又 R 为上三角阵, 故只需 $O(n^2)$ 即可得到解。对于最小二乘问题, 由 $Q^T Q = 1$, 我们更有惊喜的发现:

$$A^T A x = A^T b \Rightarrow (QR)^T QR = (QR)^T b \Rightarrow R^T R x = R^T Q^T b \Rightarrow R x = Q^T b$$

如此一来, 我们首先绕过了矩阵的乘法。为了了解其进一步的优点, 我们引入一个新的概念: 条件数

定义 0.1: 对任意一个矩阵 \mathbf{A} ，我们可以对它进行 SVD 分解 $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ ，其中奇异值 $\sigma_1, \sigma_2, \dots$ 是 $\mathbf{\Sigma}$ 矩阵的对角线上的元素。定义矩阵 \mathbf{A} 的条件数为

$$\text{cond } \mathbf{A} = \frac{\max_i \sigma_i}{\min_i \sigma_i}.$$

这里我们不给出条件数的推导和性质，只介绍其直观含义。当 $\text{cond } \mathbf{A}$ 越大时，计算的误差也就越大。想象高斯消元中，一个超大值 (e.g. 10^{16}) 除以一个小值 (e.g. 10^{-16}) 将会导致数值溢出，反之则会导致精度丢失。故保证计算时有较小的条件数是很重要的。

对于前两种方法，都计算了 $\mathbf{A}^T \mathbf{A}$ ，这导致 $\text{cond } \mathbf{A}^T \mathbf{A} = \text{cond } \mathbf{V} \mathbf{\Sigma}^2 \mathbf{V}^T = (\text{cond } \mathbf{A})^2$ 。由此一来，条件数被平方了。

而超氏的方法中，没有计算改矩阵乘法，避免了条件数被平方。配合以数值稳定的 *Household* 方法计算 QR 分解，将得到数值稳定的答案。

0.5 面试之外：MATLAB 的反除函数

事实上，作为一款专业（付费）软件，MATLAB 的函数一直在完善升级。当前的反除 \ 对大型矩阵运算时过程相当复杂。实际上，反除是使用一系列判断确定矩阵的特有性质（类型）选择最优的分解方法后进行求解。具体的判断如图所示。

简单的概括，反除法下，MATLAB 先判断矩阵是否为方阵，否则用 QR 分解，是则判断是否是三角阵或交换行的位置后构成三角阵。再判断是否（共轭）对称，对于对称阵，若可以三角分解，则使用 Cholesky 分解，其计算量约为 LU 分解的一半，否则使用 LDL 分解。若不对称，当矩阵是 Hessenberg 矩阵时（若 $a_{ij} = 0 \forall i > j + 1$ ，则 \mathbf{A} 称为上 Hessenberg 矩阵），可以针对其使用 Hessenberg 分解加速，否则才使用我们上述的 LU 分解。

注意到对于 n 阶方阵，MATLAB 并不会去选择 QR 分解，而对于最小二乘问题，QR 分解有其特殊的好处。故超勇的方法确实比甄汇之方法优。

而在 MATLAB R2017b 中，通过引入函数 `decomposition` 再次对反除法进行了优化。使用 `tA = decomposition(A); x = tA \ b`，可以先对 \mathbf{A} 进行合适的分解，再进行计算。这可以方便的实现前面提到的 LU 分解优点中的第三点。实践检验，对于 \mathbf{A} 不变 \mathbf{b} 改变的情况下，仅 1000 次运算 $n = 100$ 的矩阵，普通的反除效率和先使用 `decomposition` 的反除效率相乘达 9 倍。

希望以上内容也能帮助你更好的使用 MATLAB 的函数。

0.6 结语

无名氏评曰：当今世事，群雄逐鹿，一位难求。非有扎实学问，超人之能，难以臻高手之境，一招一回间，马脚尽露。数学者，计算机之基础也；线代、矩阵者，机器学习之基石也。仅一最小二乘问题，三法共解，高下立断。从计算机者，因以此为鉴，既应通过学习好的数学工具提升算法，也应不满足于仅于数学中得解，而应考虑何以快速的计算答案。此双剑合璧，可冠一方也。不学可乎？不精学可乎？

参考资料

- 1 方保镕，周继东，李医民《矩阵论》
- 2 张皓 <https://www.zhihu.com/question/62482926/answer/526304253>
- 3 菡父 <https://zhuanlan.zhihu.com/p/30958676>
- 4 线代启示录 <https://ccjou.wordpress.com>

第 1 章 带 English 的标题

这是 NJUrepo 的示例文档，基本上覆盖了模板中所有格式的设置。建议大家在使用模板之前，除了阅读《NJUrepo 用户手册》，这个示例文档也最好能看一看。

小老鼠偷吃热凉粉；短长虫环绕矮高粱^①。

1.1 封面相关

封面的例子请参看 `cover.tex`。主要符号表参看 `denotation.tex`，附录和个人简历分别参看 `appendix01.tex` 和 `resume.tex`。里面的命令都很直观，一看即会^②。

1.2 字体命令

苏轼 (1037-1101)，北宋文学家、书画家。字子瞻，号东坡居士，眉州眉山（今属四川）人。苏洵子。嘉佑进士。神宗时曾任祠部员外郎，因反对王安石新法而求外职，任杭州通判，知密州、徐州、湖州。后以作诗“谤讟朝廷”罪贬黄州。哲宗时任翰林学士，曾出知杭州、颖州等，官至礼部尚书。后又贬谪惠州、儋州。北还后第二年病死常州。南宋时追谥文忠。与父洵弟辙，合称“三苏”。在政治上属于旧党，但也有改革弊政的要求。其文汪洋恣肆，明白畅达，为“唐宋八大家”之一。其诗清新豪健，善用夸张比喻，在艺术表现方面独具风格。少数诗篇也能反映民间疾苦，指责统治者的奢侈骄纵。词开豪放一派，对后代很有影响。《念奴娇·赤壁怀古》、《水调歌头·丙辰中秋》传诵甚广。

坡仙擅长行书、楷书，取法李邕、徐浩、颜真卿、杨凝式，而能自创新意。用笔丰腴跌宕，有天真烂漫之趣。与蔡襄、黄庭坚、米芾并称“宋四家”。能画竹，学文同，也喜作枯木怪石。论画主张“神似”，认为“论画以形似，见与儿童邻”；高度评价“诗中有画，画中有诗”的艺术造诣。诗文有《东坡七集》等。存世书迹有《答谢民师论文帖》、《祭黄几道文》、《前赤壁赋》、《黄州寒食诗帖》等。画迹有《枯木怪石图》、《竹石图》等。

^①韩愈 (768-824)，字退之，河南河阳（今河南孟县）人，自称郡望昌黎，世称韩昌黎。幼孤贫刻苦好学，德宗贞元八年进士。曾任监察御史，因上疏请免关中赋役，贬为阳山县令。后随宰相裴度平定淮西迁刑部侍郎，又因上表谏迎佛骨，贬潮州刺史。做过吏部侍郎，死谥文公，故世称韩吏部、韩文公。是唐代古文运动领袖，与柳宗元合称韩柳。诗力求险怪新奇，雄浑重气势。

^②你说还是看不懂？怎么会呢？

易与天地准，故能弥纶天地之道。仰以观於天文，俯以察於地理，是故知幽明之故。原始反终，故知死生之说。精气为物，游魂为变，是故知鬼神之情状。与天地相似，故不违。知周乎万物，而道济天下，故不过。旁行而不流，乐天知命，故不忧。安土敦乎仁，故能爱。范围天地之化而不过，曲成万物而不遗，通乎昼夜之道而知，故神无方而易无体。

[无 \youyuan 字体。] 有天地，然后万物生焉。盈天地之间者，唯万物，故受之以屯；屯者盈也，屯者物之始生也。物生必蒙，故受之以蒙；蒙者蒙也，物之穉也。物穉不可不养也，故受之以需；需者饮食之道也。饮食必有讼，故受之以讼。讼必有众起，故受之以师；师者众也。众必有所比，故受之以比；比者比也。比必有所畜也，故受之以小畜。物畜然后有礼，故受之以履。

履而泰，然后安，故受之以泰；泰者通也。物不可以终通，故受之以否。物不可以终否，故受之以同人。与人同者，物必归焉，故受之以大有。有大者不可以盈，故受之以谦。有大而能谦，必豫，故受之以豫。豫必有随，故受之以随。以喜随人者，必有事，故受之以蛊；蛊者事也。

[无 \lishu 字体。] 有事而后大，故受之以临；临者大也。物大然后可观，故受之以观。可观而后有所合，故受之以噬嗑；嗑者合也。物不可以苟合而已，故受之以贲；贲者饰也。致饰然后亨，则尽矣，故受之以剥；剥者剥也。物不可以终尽，剥穷上反下，故受之以复。复则不妄矣，故受之以无妄。

有无妄然后可畜，故受之以大畜。物畜然后可养，故受之以颐；颐者养也。不养则不可动，故受之以大过。物不可以终过，故受之以坎；坎者陷也。陷必有所丽，故受之以离；离者丽也。

1.3 表格样本

1.3.1 基本表格

模板中关于表格的宏包有三个：**booktabs**、**array** 和 **longtabular**，命令有一个 **\hlinewd**。三线表可以用 **booktabs** 提供的 **\toprule**、**\midrule** 和 **\bottomrule**。它们与 **longtable** 能很好的配合使用。如果表格比较简单的话可以直接用命令 **\hlinewd{<width>}** 控制。

首先来看一个最简单的表格。表 1.1 列举了本模板主要文件及其功能。请大家注意三线表中各条线对应的命令。这个例子还展示了如何在表格中正确使用脚注。由于 **L^AT_EX** 本身不支持在表格中使用 **\footnote**，所以我们不得不将表格放在小页中，而且最好将表格的宽度设置为小页的宽度，这样脚注看起来才更美观。

表 1.1 模板文件。如果表格的标题很长，那么在表格索引中就会很不美观，所以要像 `chapter` 那样在前面用中括号写一个简短的标题。这个标题会出现在索引中。

文件名	描述
<code>thuthesis.ins</code>	L ^A T _E X 安装文件，DocStrip ^①
<code>thuthesis.dtx</code>	所有的一切都在这里 ^② 。
<code>thuthesis.cls</code>	模板类文件。
<code>thuthesis.cfg</code>	模板配置文。 <code>cls</code> 和 <code>cfg</code> 由前两个文件生成。
<code>thuthesis-numeric.bst</code>	参考文献 BIB _T E _X 样式文件。
<code>thuthesis-author-year.bst</code>	参考文献 BIB _T E _X 样式文件。
<code>thuthesis.sty</code>	常用的包和命令写在这里，减轻主文件的负担。

②表格中的脚注

②再来一个

1.3.2 复杂表格

我们经常会在表格下方标注数据来源，或者对表格里面的条目进行解释。前面的脚注是一种不错的方法，如果不喜欢脚注，可以在表格后面写注释，比如表 1.2。

表 1.2 复杂表格示例 1。这个引用 Knuth (1989) 不会导致编号混乱。

x \ y	First Half		Second Half	
	1st Qtr	2nd Qtr	3rd Qtr	4th Qtr
East*	20.4	27.4	90	20.4
West**	30.6	38.6	34.6	31.6

注：数据来源 《NJUrepo 使用手册》。

*：东部

**：西部

此外，表 1.2 同时还演示了另外两个功能：1) 通过 `tabularx` 的 `|X|` 扩展实现表格自动放大；2) 通过命令 `\diagbox` 在表头部分插入反斜线。

为了使我们的例子更接近实际情况，我会在必要的时候插入一些“无关”文字，以免太多图表同时出现，导致排版效果不太理想。第一个出场的当然是我的最爱：风流潇洒、骏马绝尘、健笔凌云的李太白了。

李白，字太白，陇西成纪人。凉武昭王暠九世孙。或曰山东人，或曰蜀人。白少有逸才，志气宏放，飘然有超世之心。初隐岷山，益州长史苏颋见而异之，曰：“是

子天才英特，可比相如。”天宝初，至长安，往见贺知章。知章见其文，叹曰：“子滴仙人也。”言于明皇，召见金銮殿，奏颂一篇。帝赐食，亲为调羹，有诏供奉翰林。白犹与酒徒饮于市，帝坐沉香亭子，意有所感，欲得白为乐章，召入，而白已醉。左右以水頰面，稍解，援笔成文，婉丽精切。帝爱其才，数宴见。白常侍帝，醉，使高力士脱靴。力士素贵，耻之，摘其诗以激杨贵妃。帝欲官白，妃辄沮止。白自知不为亲近所容，恳求还山。帝赐金放还。乃浪迹江湖，终日沉饮。永王璘都督江陵，辟为僚佐。璘谋乱，兵败，白坐长流夜郎，会赦得还。族人阳冰为当涂令，白往依之。代宗立，以左拾遗召，而白已卒。文宗时，诏以白歌诗、裴旻剑舞、张旭草书为三绝云。集三十卷。今编诗二十五卷。——《全唐诗》诗人小传

浮动体的并排放置一般有两种情况：1) 二者没有关系，为两个独立的浮动体；2) 二者隶属于同一个浮动体。对表格来说并排表格既可以像图 1.3、图 1.4 使用小页环境，也可以如图 1.5 使用子表格来做。图的例子参见第 2.1.2.2 节。

表 1.3 第一个并排子表格

111	222
222	333

表 1.4 第二个并排子表格

111	222
222	333

然后就是忧国忧民，诗家楷模杜工部了。杜甫，字子美，其先襄阳人，曾祖依艺为巩令，因居巩。甫天宝初应进士，不第。后献《三大礼赋》，明皇奇之，召试文章，授京兆府兵曹参军。安禄山陷京师，肃宗即位灵武，甫自贼中遁赴行在，拜左拾遗。以论救房琯，出为华州司功参军。关辅饥乱，寓居同州同谷县，身自负薪采椽，餽糒不给。久之，召补京兆府功曹，道阻不赴。严武镇成都，奏为参谋、检校工部员外郎，赐绯。武与甫世旧，待遇甚厚。乃于成都浣花里种竹植树，枕江结庐，纵酒啸歌其中。武卒，甫无所依，乃之东蜀就高适。既至而适卒。是岁，蜀帅相攻杀，蜀大扰。甫携家避乱荆楚，扁舟下峡，未维舟而江陵亦乱。乃溯沿湘流，游衡山，寓居耒阳。卒年五十九。元和中，归葬偃师首阳山，元稹志其墓。天宝间，甫与李白齐名，时称李杜。然元稹之言曰：“李白壮浪纵恣，摆去拘束，诚亦差肩子美矣。至若铺陈终始，排比声韵，大或千言，次犹数百，词气豪迈，而风调清深，属对律切，而脱弃凡近，则李尚不能历其藩翰，况堂奥乎。”白居易亦云：“杜诗贯穿古今，尽工尽善，殆过于李。”元、白之论如此。盖其出处劳佚，喜乐悲愤，好贤恶恶，一见之于诗。而又以忠君忧国、伤时念乱为本旨。读其诗可以知其世，故当时谓之“诗史”。旧集诗文共六十卷，今编诗十九卷。

不可否认 L^AT_EX 的表格功能没有想象中的那么强大，不过只要足够认真，足够

表 1.5 并排子表格

(a) 第一个子表格		(b) 第二个子表格	
111	222	111	222
222	333	222	333

细致，同样可以排出来非常复杂非常漂亮的表格。请参看表 1.6。

表 1.6 复杂表格示例 2

Network Topology		# of nodes	# of clients			Server
GT-ITM	Waxman Transit-Stub	600	2%	10%	50%	Max. Connectivity
Inet-2.1		6000				
Xue	Rui	Ni	NJUrepo			
	ABCDEF					

最后就是清新飘逸、文约意赅、空谷绝响的王大侠了。王维，字摩诘，河东人。工书画，与弟缙俱有俊才。开元九年，进士擢第，调太乐丞。坐累为济州司仓参军，历右拾遗、监察御史、左补阙、库部郎中，拜吏部郎中。天宝末，为给事中。安禄山陷两都，维为贼所得，服药阳喑，拘于菩提寺。禄山宴凝碧池，维潜赋诗悲悼，闻于行在。贼平，陷贼官三等定罪，特原之，责授太子中允，迁中庶子、中书舍人。复拜给事中，转尚书右丞。维以诗名盛于开元、天宝间，宁薛诸王驸马豪贵之门，无不拂席迎之。得宋之问辋川别墅，山水绝胜，与道友裴迪，浮舟往来，弹琴赋诗，啸咏终日。笃于奉佛，晚年长斋禅诵。一日，忽索笔作书数纸，别弟缙及平生亲故，舍笔而卒。赠秘书监。宝应中，代宗问缙：“朕常于诸王坐闻维乐章，今存几何？”缙集诗六卷，文四卷，表上之。敕答云，卿伯氏位列先朝，名高希代。抗行周雅，长揖楚辞。诗家者流，时论归美。克成编录，叹息良深。殷璠谓维诗词秀调雅，意新理惬。在泉成珠，著壁成绘。苏轼亦云：“维诗中有画，画中有诗也。”今编诗四卷。

要想用好论文模板还是得提前学习一些 $\text{T}_\text{E}\text{X}/\text{L}_\text{A}\text{T}_\text{E}\text{X}$ 的相关知识，具备一些基本能力，掌握一些常见技巧，否则一旦遇到问题还真是比较麻烦。我们见过很多这样的同学，一直以来都是使用 Word 等字处理工具，以为 $\text{L}_\text{A}\text{T}_\text{E}\text{X}$ 模板的用法也应该类似，所以就沿袭同样的思路来对待这种所见非所得的排版工具，结果被折腾的焦头烂额，疲惫不堪。

如果您要排版的表格长度超过一页，那么推荐使用 `longtable` 或者 `supertabular` 宏包，模板对 `longtable` 进行了相应的设置，所以用起来可能简单一些。表 1.7

就是 `longtable` 的简单示例。

表 1.7 实验数据

测试程序	正常运行 时间 (s)	同步 时间 (s)	检查点 时间 (s)	卷回恢复 时间 (s)	进程迁移 时间 (s)	检查点 文件 (KB)
CG.A.2	23.05	0.002	0.116	0.035	0.589	32491
CG.A.4	15.06	0.003	0.067	0.021	0.351	18211
CG.A.8	13.38	0.004	0.072	0.023	0.210	9890
CG.B.2	867.45	0.002	0.864	0.232	3.256	228562
CG.B.4	501.61	0.003	0.438	0.136	2.075	123862
CG.B.8	384.65	0.004	0.457	0.108	1.235	63777
MG.A.2	112.27	0.002	0.846	0.237	3.930	236473
MG.A.4	59.84	0.003	0.442	0.128	2.070	123875
MG.A.8	31.38	0.003	0.476	0.114	1.041	60627
MG.B.2	526.28	0.002	0.821	0.238	4.176	236635
MG.B.4	280.11	0.003	0.432	0.130	1.706	123793
MG.B.8	148.29	0.003	0.442	0.116	0.893	60600
LU.A.2	2116.54	0.002	0.110	0.030	0.532	28754
LU.A.4	1102.50	0.002	0.069	0.017	0.255	14915
LU.A.8	574.47	0.003	0.067	0.016	0.192	8655
LU.B.2	9712.87	0.002	0.357	0.104	1.734	101975
LU.B.4	4757.80	0.003	0.190	0.056	0.808	53522
LU.B.8	2444.05	0.004	0.222	0.057	0.548	30134
EPA.2	123.81	0.002	0.010	0.003	0.074	1834
EPA.4	61.92	0.003	0.011	0.004	0.073	1743
EPA.8	31.06	0.004	0.017	0.005	0.073	1661
EP.B.2	495.49	0.001	0.009	0.003	0.196	2011
EP.B.4	247.69	0.002	0.012	0.004	0.122	1663
EP.B.8	126.74	0.003	0.017	0.005	0.083	1656

1.3.3 其它

如果不想让某个表格或者图片出现在索引里面，请使用命令 `\caption*`。这个命令不会给表格编号，也就是出来的只有标题文字而没有“表 XX”，“图 XX”，否则索引里面序号不连续就显得不伦不类，这也是 \LaTeX 里星号命令默认的规则。

有这种需求的多是本科同学的英文资料翻译部分，如果觉得附录中英文原文中的表格和图片显示成“表”和“图”不协调的话，一个很好的办法就是用 `\caption*`，参数随便自己写，比如不守规矩的表 1.111 和图 1.111 能满足这种特殊需要（可以

参看附录部分)。

表 1.111 这是一个手动编号，不出现在索引中的表格。

NJUrepo

Figure 1.111 这是一个手动编号，不出现在索引中的图。

薛瑞尼

如果的确想让它编号，但又不想让它出现在索引中的话，目前模板上不支持。最后，虽然大家不一定会独立使用小页，但是关于小页中的脚注还是有必要提一下。请看下面的例子。

柳宗元，字子厚（773-819），河东（今永济县）人^①，是唐代杰出的文学家，哲学家，同时也是一位政治改革家。与韩愈共同倡导唐代古文运动，并称韩柳^②。

^①山西永济水校。

^②唐宋八大家之首二位。

唐朝安史之乱后，宦官专权，藩镇割据，土地兼并日渐严重，社会生产破坏严重，民不聊生。柳宗元对这种社会现实极为不满，他积极参加了王叔文领导的“永济革新”，并成为这一运动的中坚人物。他们革除弊政，打击权奸，触犯了宦官和官僚贵族利益，在他们的联合反扑下，改革失败了，柳宗元被贬为永州司马。

1.4 定理环境

给大家演示一下各种和证明有关的环境：

假设 1.1： 待月西厢下，迎风户半开；隔墙花影动，疑是玉人来。

$$c = a^2 - b^2 \quad (1-1)$$

$$= (a + b)(a - b) \quad (1-2)$$

千辛万苦，历尽艰难，得有今日。然相从数千里，未曾哀戚。今将渡江，方图百年欢笑，如何反起悲伤？（引自《杜十娘怒沉百宝箱》）

定义 1.1： 子曰：「道千乘之国，敬事而信，节用而爱人，使民以时。」

千古第一定义！问世间、情为何物，只教生死相许？天南地北双飞客，老翅几回寒暑。欢乐趣，离别苦，就中更有痴儿女。君应有语，渺万里层云，千山暮雪，只影向谁去？

横汾路，寂寞当年箫鼓，荒烟依旧平楚。招魂楚些何嗟及，山鬼暗谿风雨。天也妒，未信与，莺儿燕子俱黄土。千秋万古，为留待骚人，狂歌痛饮，来访雁丘处。

命题 1.1: 曾子曰：「吾日三省吾身——为人谋而不忠乎？与朋友交而不信乎？传不习乎？」

多么凄美的命题啊！其日牛马嘶，新妇入青庐，奄奄黄昏后，寂寂人定初，我命绝今日，魂去尸长留，揽裙脱丝履，举身赴清池，府吏闻此事，心知长别离，徘徊庭树下，自挂东南枝。

注释 1.1: 天不言自高，水不言自流。

$$\begin{aligned}\varphi(x, z) &= z - \gamma_{10}x - \gamma_{mn}x^m z^n \\ &= z - Mr^{-1}x - Mr^{-(m+n)}x^m z^n\end{aligned}$$

$$\zeta^0 = (\xi^0)^2, \quad (1-3)$$

$$\zeta^1 = \xi^0 \xi^1, \quad (1-4)$$

$$\zeta^2 = (\xi^1)^2, \quad (1-5)$$

天尊地卑，乾坤定矣。卑高以陈，贵贱位矣。动静有常，刚柔断矣。方以类聚，物以群分，吉凶生矣。在天成象，在地成形，变化见矣。鼓之以雷霆，润之以风雨，日月运行，一寒一暑，乾道成男，坤道成女。乾知大始，坤作成物。乾以易知，坤以简能。易则易知，简则易从。易知则有亲，易从则有功。有亲则可久，有功则可大。可久则贤人之德，可大则贤人之业。易简，而天下矣之理矣；天下之理得，而成位乎其中矣。

公理 1.1: 两点间直线段距离最短。

$$\begin{aligned}x &\equiv y + 1 \pmod{m^2} & (1-6) \\ x &\equiv y + 1 \pmod{m^2} & (1-7) \\ x &\equiv y + 1 \pmod{m^2} & (1-8)\end{aligned}$$

《彖曰》：大哉乾元，万物资始，乃统天。云行雨施，品物流形。大明始终，六位时成，时乘六龙以御天。乾道变化，各正性命，保合大和，乃利贞。首出庶物，万国咸宁。

《象曰》：天行健，君子以自强不息。潜龙勿用，阳在下也。见龙再田，德施普也。终日乾乾，反复道也。或跃在渊，进无咎也。飞龙在天，大人造也。亢龙有悔，盈不可久也。用九，天德不可为首也。

引理 1.1: 《猫和老鼠》是我最爱看的动画片。

$$\begin{aligned} \int_a^b \left\{ \int_a^b [f(x)^2 g(y)^2 + f(y)^2 g(x)^2] - 2f(x)g(x)f(y)g(y) dx \right\} dy \\ = \int_a^b \left\{ g(y)^2 \int_a^b f^2 + f(y)^2 \int_a^b g^2 - 2f(y)g(y) \int_a^b fg \right\} dy \end{aligned}$$

行行重行行，与君生别离。相去万余里，各在天一涯。道路阻且长，会面安可知。胡马依北风，越鸟巢南枝。相去日已远，衣带日已缓。浮云蔽白日，游子不顾返。思君令人老，岁月忽已晚。弃捐勿复道，努力加餐饭。

定理 1.1: 犯我强汉者，虽远必诛 ——陈汤（汉）

$$y = 1 \tag{1-9a}$$

$$y = 0 \tag{1-9b}$$

道可道，非常道。名可名，非常名。无名天地之始；有名万物之母。故常无，欲以观其妙；常有，欲以观其徼。此两者，同出而异名，同谓之玄。玄之又玄，众妙之门。上善若水。水善利万物而不争，处众人之所恶，故几于道。曲则全，枉则直，洼则盈，敝则新，少则多，多则惑。人法地，地法天，天法道，道法自然。知人者智，自知者明。胜人者有力，自胜者强。知足者富。强行者有志。不失其所者久。死而不亡者寿。

证明 燕赵古称多感慨悲歌之士。董生举进士，连不得志于有司，怀抱利器，郁郁适兹土，吾知其必有合也。董生勉乎哉？

夫以子之不遇时，苟慕义强仁者，皆爱惜焉，矧燕、赵之士出乎其性者哉！然吾尝闻风俗与化移易，吾恶知其今不异于古所云邪？聊以吾子之行卜之也。董生勉乎哉？

吾因子有所感矣。为我吊望诸君之墓，而观于其市，复有昔时屠狗者乎？为我谢曰：“明天子在上，可以出而仕矣！” ——韩愈《送董邵南序》 □

推论 1.1: 四川话配音的《猫和老鼠》是世界上最好看最好听最有趣的动画片。

$$V_i = v_i - q_i v_j, \quad X_i = x_i - q_i x_j, \quad U_i = u_i, \quad \text{for } i \neq j; \tag{1-10}$$

$$V_j = v_j, \quad X_j = x_j, \quad U_j u_j + \sum_{i \neq j} q_i u_i. \tag{1-11}$$

迢迢牵牛星，皎皎河汉女。纤纤擢素手，札札弄机杼。终日不成章，泣涕零如雨。河汉清且浅，相去复几许。盈盈一水间，脉脉不得语。

例 1.1: 大家来看这个例子。

$$\begin{cases} \nabla f(x^*) - \sum_{j=1}^p \lambda_j \nabla g_j(x^*) = 0 \\ \lambda_j g_j(x^*) = 0, \quad j = 1, 2, \dots, p \\ \lambda_j \geq 0, \quad j = 1, 2, \dots, p. \end{cases} \quad (1-12)$$

练习 1.1: 请列出 Andrew S. Tanenbaum 和 W. Richard Stevens 的所有著作。

猜想 1.1: *Poincare Conjecture* If in a closed three-dimensional space, any closed curves can shrink to a point continuously, this space can be deformed to a sphere.

问题 1.1: 回答还是不回答，是个问题。

如何引用定理 1.1 呢? 加上 \label 使用 \ref 即可。妾发初覆额，折花门前剧。郎骑竹马来，绕床弄青梅。同居长干里，两小无嫌猜。十四为君妇，羞颜未尝开。低头向暗壁，千唤不一回。十五始展眉，愿同尘与灰。常存抱柱信，岂上望夫台。十六君远行，瞿塘滟滪堆。五月不可触，猿声天上哀。门前迟行迹，一一生绿苔。苔深不能扫，落叶秋风早。八月蝴蝶来，双飞西园草。感此伤妾心，坐愁红颜老。

1.5 参考文献

当然参考文献可以直接写 \bibitem，虽然费点功夫，但是好控制，各种格式可以自己随意改写。

本模板推荐使用 BIB_TE_X，分别提供数字引用 (thuthesis-numeric.bst) 和作者年份引用 (thuthesis-author-year.bst) 样式，基本符合学校的参考文献格式 (如专利等引用未加详细测试)。看看这个例子，关于书的 Knuth (1989); Goosens et al. (1994); Gröning et al. (2004)，还有这些 Krasnogor (2004); 阎真 (2001); 班固 (1998)，关于杂志的 Chafik El Idrissi et al. (1994); Mellinger et al. (1996); Shell (2002)，硕士论文 猪八戒 (2005); Jeyakumar (2004)，博士论文 沙和尚 (2005); Zadok (2001)，标准文件 IEEE Std 1363-2000 (2000)，会议论文 Kim et al. (2003); Kocher et al. (1999)，技术报告 Woo et al. (1995)，电子文献 萧钰; Online Computer Library Center, Inc.。若使用著者-出版年制，中文参考文献 贾宝玉 等 (1800) 应增加 key={pinyin} 字段，以便正确进行排序 王重阳 等 (2006)。另外，如果对参考文献有不如意的地方，请手动修改 bbl 文件。

有时候不想要上标，那么可以这样 (沙和尚, 2005)，这个非常重要。

有时候一些参考文献没有纸质出处，需要标注 URL。缺省情况下，URL 不会在连字符处断行，这可能使得用连字符代替空格的网址分行很难看。如果需要，可以将模板类文件中

```
\RequirePackage{hyperref}
```

一行改为：

```
\PassOptionsToPackage{hyphens}{url}
\RequirePackage{hyperref}
```

使得连字符处可以断行。更多设置可以参考 url 宏包文档。

1.6 公式

贝叶斯公式如式 (1-13)，其中 $p(y|\mathbf{x})$ 为后验； $p(\mathbf{x})$ 为先验；分母 $p(\mathbf{x})$ 为归一化因子。

$$p(y|\mathbf{x}) = \frac{p(\mathbf{x}, y)}{p(\mathbf{x})} = \frac{p(\mathbf{x}|y)p(y)}{p(\mathbf{x})} \quad (1-13)$$

论文里面公式越多， $\text{T}_{\text{E}}\text{X}$ 就越 happy。再看一个 `amsmath` 的例子：

$$\det \mathbf{K}(t = 1, t_1, \dots, t_n) = \sum_{I \in \mathbf{n}} (-1)^{|I|} \prod_{i \in I} t_i \prod_{j \in \bar{I}} (D_j + \lambda_j t_j) \det \mathbf{A}^{(\lambda)}(\bar{I}|\bar{I}) = 0. \quad (1-14)$$

前面定理示例部分列举了很多公式环境，可以说把常见的情况都覆盖了，大家在写公式的时候一定要好好看 `amsmath` 的文档，并参考模板中的用法：

$$\begin{aligned} & \int_a^b \left\{ \int_a^b [f(x)^2 g(y)^2 + f(y)^2 g(x)^2] - 2f(x)g(x)f(y)g(y) dx \right\} dy \\ &= \int_a^b \left\{ g(y)^2 \int_a^b f^2 + f(y)^2 \int_a^b g^2 - 2f(y)g(y) \int_a^b fg \right\} dy \end{aligned}$$

其实还可以看看这个多级规划：

$$\left\{ \begin{array}{l} \max_x F(x, y_1^*, y_2^*, \dots, y_m^*) \\ \text{subject to:} \\ G(x) \leq 0 \\ (y_1^*, y_2^*, \dots, y_m^*) \text{ solves problems } (i = 1, 2, \dots, m) \\ \left\{ \begin{array}{l} \max_{y_i} f_i(x, y_1, y_2, \dots, y_m) \\ \text{subject to:} \\ g_i(x, y_1, y_2, \dots, y_m) \leq 0. \end{array} \right. \end{array} \right. \quad (1-15)$$

这些跟规划相关的公式都来自于刘宝碇老师《不确定规划》的课件。

第 2 章 中华人民共和国

2.1 其它例子

在第 1 章中我们学习了贝叶斯公式 (1-13)，这里我们复习一下：

$$p(y|\mathbf{x}) = \frac{p(\mathbf{x}, y)}{p(\mathbf{x})} = \frac{p(\mathbf{x}|y)p(y)}{p(\mathbf{x})} \quad (2-1)$$

2.1.1 绘图

本模板不再预先装载任何绘图包（如 `pstricks`，`pgf` 等），完全由用户来决定。个人觉得 `pgf` 不错，不依赖于 `Postscript`。此外还有很多针对 `LaTeX` 的 GUI 作图工具，如 `XFig(jFig)`, `WinFig`, `Tpx`, `Ipe`, `Dia`, `Inkscape`, `LaTeXPiX`, `jPicEdt`, `jaxdraw` 等等。

2.1.2 插图

强烈推荐《`LaTeX 2ε` 插图指南》！关于子图形的使用细节请参看 `subcaption` 宏包的说明文档。

2.1.2.1 一个图形

一般图形都是处在浮动环境中。之所以称为浮动是指最终排版效果图形的位置不一定与源文件中的位置对应^①，这也是刚使用 `LaTeX` 同学可能遇到的问题。如果要强制固定浮动图形的位置，请使用 `float` 宏包，它提供了 `[H]` 参数，比如图 2.1。

^①This is not a bug, but a feature of `LaTeX`!



图 2.1 利用 Xfig 制图

大学之道，在明明德，在亲民，在止于至善。知止而后有定；定而后能静；静而后能安；安而后能虑；虑而后能得。物有本末，事有终始。知所先后，则近道矣。古之欲明明德于天下者，先治其国；欲治其国者，先齐其家；欲齐其家者，先修其身；欲修其身者，先正其心；欲正其心者，先诚其意；欲诚其意者，先致其知；致知在格物。物格而后知至；知至而后意诚；意诚而后心正；心正而后身修；身修而后家齐；家齐而后国治；国治而后天下平。自天子以至于庶人，壹是皆以修身为本。其本乱而未治者否矣。其所厚者薄，而其所薄者厚，未之有也！

——《大学》

2.1.2.2 多个图形

如果多个图形相互独立，并不共用一个图形计数器，那么用 `minipage` 或者 `parbox` 就可以。否则，请参看图 2.2，它包含两个小图，分别是图 2.2(a)和图 2.2(b)。推荐使用 `\subcaptionbox`，因为可以像图 2.2 那样对齐子图的标题，也可以使用 `subcaption` 宏包的 `\subcaption` (放在 `minipage` 中,用法同 `\caption`)或是 `subfigure`

、`subtable` 环境，像图 2.3，不要再用 `\subfloat`、`\subfigure` 和 `\subtable`。

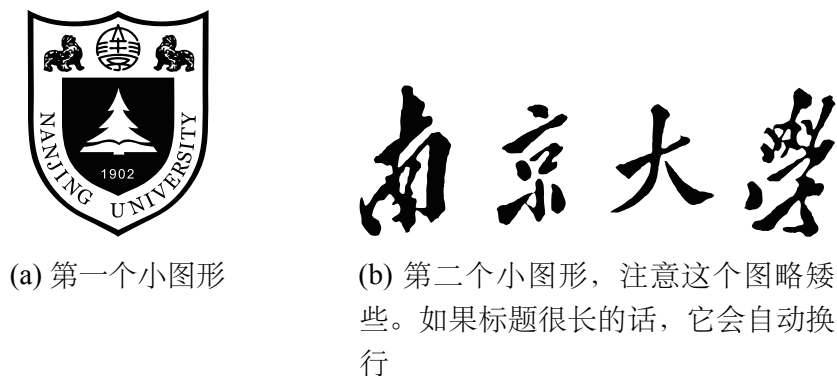


图 2.2 包含子图形的大图形 (`subcaptionbox` 示例)

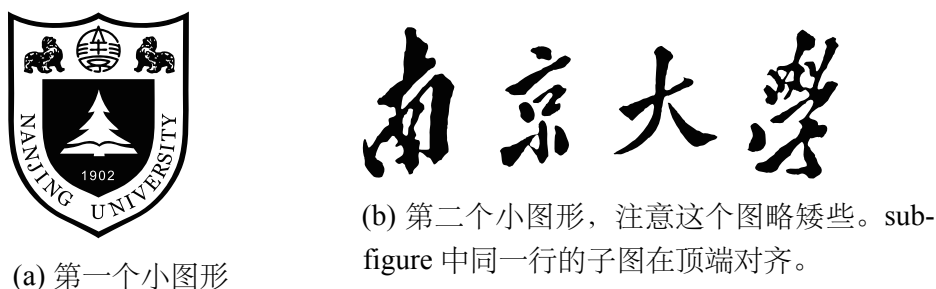


图 2.3 包含子图形的大图形 (`subfigure` 示例)

古之学者必有师。师者，所以传道受业解惑也。人非生而知之者，孰能无惑？惑而不从师，其为惑也，终不解矣。生乎吾前，其闻道也固先乎吾，吾从而师之；生乎吾後，其闻道也亦先乎吾，吾从而师之。吾师道也，夫庸知其年之先後生於吾乎！是故无贵无贱无长无少，道之所存，师之所存也。

嗟乎！师道之不传也久矣，欲人之无惑也难矣。古之圣人，其出人也远矣，犹且从师而问焉；今之众人，其下圣人也亦远矣，而耻学於师。是故圣益圣，愚益愚。圣人之所以为圣，愚人之所以为愚，其皆出於此乎？爱其子，择师而教之，於其身也，则耻师焉，惑焉。彼童子之师，授之书而习其句读者，非吾所谓传其道、解其惑者也。句读之不知，惑之不解，或师焉，或不焉，小学而大遗，吾未见其明也。巫医、乐师、百工之人不耻相师，士大夫之族曰“师”曰“弟子”之云者，则群聚而笑之。问之，则曰：彼与彼年相若也，道相似也，位卑则足羞，官盛则近谏。呜呼！师道之不复，可知矣。巫医、乐师、百工之人。吾子不齿，今其智乃反不能及，其可怪也欤！圣人无常师。孔子师郯子、苌子、师襄、老聃。郯子之徒，其贤不及孔子。孔子曰：“三人行，必有我师。”是故弟子不必不如师，师不必贤於弟子。闻



图 2.4 并排第一个图



图 2.5 并排第二个图

道有先後，术业有专攻，如是而已。

如果要把编号的两个图形并排，那么小页就非常有用：

李氏子蟠，年十七，好古文、六艺，经传皆通习之，不拘於时，学於余。余嘉其能行古道，作师说以贻之。

——韩愈（唐）

第 3 章 实现功能

3.1 硬件部分

1. 五级流水线 CPU，实现除特殊访存指令外所有的逻辑、算术、移位、访存、跳转分支指令
2. 解决数据相关、控制相关问题。实现跳转指令仅延迟槽中内容一定被执行，第二条以后的指令不受影响；实现访存指令无需延迟槽。
3. 128KB 内存同步读写
4. VGA，键盘，音频多 IO 的调度
5. 代码段、数据段、栈段和 IO 段的内存划分
6. MMIO 实现多 IO 管理

3.2 软件部分

1. 实现了 750 行汇编代码构成的操作系统，包含三个被调用 40 次左右的系统函数。以 `printanasciicode` 为例，仅适用 90 条指令实现了软件控制写入，节省了大量存储器空间，
2. 代码实现显示字符功能
3. 解析 `./gdb ./fib ./hello ./mu` 四条用户指令
4. gdb 单步调试功能，仿造实际控制流的跳转并输出
5. 计算斐波那契数列，实现由用户控制运算第几项
6. 通过 `./mu` 开启电子琴使能，进入电子琴模式
7. 开机欢迎动画

3.3 其它

1. 搭建 MIPS32 运行时环境 (AM)，可将 C 程序编译到 MIPS32 上，并支持字符输出。结合 ICS 课程的库函数和测试程序，可提供对大型程序的支持接口
2. 项目地址：https://github.com/zhengzangw/NJU_MIPS，提供了项目的说明文档、测试样例、开发工具 (Makefile) 和硬件实现的 git 记录。
3. 由于文件内容多，我们特别为老师批改方便准备了一份文件目录7结构于本报告的最后。老师可根据该结构验收项目文件，减小验收负担。

第 4 章 组内分工

4.1 郑奘巍同学

- 工作量 >35 小时
- 完成了整个五级流水线 CPU 的编写，实现大量指令，解决数据相关和结构相关
- 实现了 RAM, ROM 和可读写 ROM
- 实现了 VGA, 键盘, 音频等 IO 的连接和 MMIO
- 制作 CPU 测试样例和初始化 IP 核文件，完成硬件测试
- 运行时环境的搭建，包括 Turing Machine 接口, ld 文件和 Makefile 文件的编写

4.2 沈天琪同学

- 工作量约 30 小时
- 完成了整个操作系统的编写
- 前期资料收集与系统结构设计，编写小部分测试样例
- 欢迎动画的初始化文件编写，键盘模块与显存模块的设计
- 共同完成最终硬件测试

第 5 章 硬件部分

5.1 五级流水线 CPU 的结构

五级流水线 CPU 分为：取指，译码，执行，访存，回写五个阶段，使得一个周期可以同时进行五条指令。基本结构如图??所示，五级流水线执行流程如图??所示。

可以看到，整个流水线由时钟 CLOCK 控制，各个模块之间设置传输模块，在每个时钟周期的上升沿将前一阶段的数据传递到下一个模块。本图中还省略了从执行阶段将跳转地址传递到 pc、为解决数据冲突从执行和访存连接到译码模块等连线。

5.1.1 取指模块

取指模块中，ce 控制整个 CPU 的开关，pc 为当前指令位置。jmp_flag_i 和 stall 分别是跳转地址和控制流水线暂停的变量，将在后文讨论。

```
always @ (posedge clk) begin
  if (rst == `RSTENABLE) begin
    ce <= `CHIPDISABLE;
  end else begin
    ce <= `CHIPENABLE;
  end
end

always @ (posedge clk) begin
  if (ce == `CHIPDISABLE) begin
    pc <= 32'h00000000;
  end else if (stall[0] == `NOSTOP) begin
    if (jmp_flag_i == `JMP) begin // 跳转
      pc <= jmp_target_address_i;
    end else begin // 地址增加
      pc <= pc + 4'h4;
    end
  end
end
```

5.1.2 译码模块

译码模块中，如果是 R 类型指令，我们使用如下的划分

```
wire[5:0] op = inst_i[31:26]; // Instruction Code
wire[4:0] sa = inst_i[10:6]; // sa
```



```

wire[5:0] func = inst_i[5:0]; //Function Code
wire[4:0] rt = inst_i[20:16]; //rt
wire[4:0] rs = inst_i[25:21]; //rs
wire[4:0] rd = inst_i[15:11]; //rd

```

如果是 I 类型指令，我们将 immediate 的值无符号扩展后放入变量 imm 中；如果是 J 型指令，我们将前 26 位放入 jmp_target_address 中传入下个模块。整个译码模块为一个选择器，先更根据 op 确定第一级，如果为 EXE_SPECIAL, 再根据 func 确定具体指令。如果为 EXE_REGIMM, 再根据 rt 确定具体指令。

对于每条指令，通过设置 wreg, reg1_read, reg2_read 控制寄存器读写开关，如果读入无效，则将立即数传递给第二个操作数。以 addu 指令和第二操作数的加载为例，代码如下。

```

case (op)
  `EXE_SPECIAL: begin
    case (func)
      `EXE_ADDU: begin
        wreg_o <= `WRITEENABLE; //需要写入，默认为rd
        aluop_o <= `EXE_ADDU_OP; //操作为无符号加法
        alusel_o <= `EXE_RES_ARITH; //取算术运算结果
        reg1_read_o <= `READENABLE; //需要读第一个寄存器，默认为rs
        reg2_read_o <= `READENABLE; //需要读第二个寄存器，默认为rt
        instvalid <= `INSTVALID; //指令有效
      end
      ...
    endcase
    ...
  endcase

always @(*) begin
  stallreq_for_reg2_loadrelate <= `NOSTOP;
  if (rst==`RSTENABLE) begin
    reg2_o <= `ZEROWORD;
    // Data Hazzard
  end else if ( pre_inst_is_load == 1'b1 && ex_wd_i == reg2_addr_o
    && reg2_read_o == 1'b1 ) begin
    stallreq_for_reg2_loadrelate <= `STOP; //load相关
  end else if ((reg2_read_o==`READENABLE)&&(ex_wreg_i==`WRITEENABLE)&&
    (ex_wd_i==reg2_addr_o)) begin
    reg2_o <= ex_wdata_i; //来自执行模块的数据
  end else if ((reg2_read_o==`READENABLE)&&(mem_wreg_i==`WRITEENABLE)&&
    (mem_wd_i==reg2_addr_o)) begin
    reg2_o <= mem_wdata_i; //来自访存模块的数据
  end else if (reg2_read_o == `READENABLE) begin
    reg2_o <= reg2_data_i; //需要读数据
  end else if (reg2_read_o == `READDISABLE) begin
    reg2_o <= imm; //无需读寄存器数据，默认提供立即数
  end else begin
    reg2_o <= `ZEROWORD;

```

```

end
end

```

5.1.3 执行模块

执行模块由逻辑运算、移位运算、转移、算术模块、乘法模块、除法模块和跳转六个部分并行组成，根据取指模块传递进来的指令类型，选择相应的输出。其中乘法和除法向 hi,lo 输出，这里没有给出。具体的指令实现在 5.2 节中给出。同时，由于除法实现的特殊性，我们专门设置了 div 模块来实现除法指令。

```

always @(*) begin
    wd_o <= wd_i;
    if (((aluop_i == `EXE_ADD_OP) || (aluop_i == `EXE_SUB_OP)) && (overflow) )
begin
    wreg_o <= `WRITEDISABLE; // 无需写入
end else begin
wreg_o <= wreg_i; // 写入与译码模块要求相同
end
    case (alusel_i)
        `EXE_RES_LOGIC: begin
            wdata_o <= logicout; // 逻辑模块
        end
        `EXE_RES_SHIFT: begin
            wdata_o <= shiftres; // 移位模块
        end
        `EXE_RES_MOVE: begin
            wdata_o <= moveres; // 转移模块
        end
        `EXE_RES_ARITH: begin
            wdata_o <= arithres; // 计算模块
        end
        `EXE_RES_JUMP: begin
            wdata_o <= link_address_i; // 跳转模块
        end
        default: begin
            wdata_o <= `ZEROWORD;
        end
    endcase
end
end

```

5.1.4 访存模块

对于非访存指令，在访存模块只是进行数据的传递。对于访存指令，则通过译码模块传递进来的参数进行处理。该模块将向外连接到 RAM 中，以 4 字节为单位，向 RAM 提供地址，是否写入，写入数据等信息。其中，mem_sel 将控制更改的内容属于以 4 字节为单位的 RAM 中的哪几位。

5.1.5 寄存器与回写模块

回写模块没有单独实现，而是直接写在寄存器内。寄存器模块 `regfile` 实现了 0-31 号寄存器，`hilo_reg` 则实现了 `hi` 和 `lo` 寄存器。

5.2 指令实现

5.2.1 逻辑指令

5.2.1.1 基本实现

逻辑指令是最简单的部分，共十七条指令，列举如下（`uext` 为对立即数进行零扩展，逻辑位移与算术位移在表中没有体现，可自行区分）

<code>and</code>	$rd \leftarrow rs \& rt$	<code>or</code>	$rd \leftarrow rs rt$
<code>xor</code>	$rd \leftarrow rs \text{ xor } rt$	<code>nor</code>	$rd \leftarrow (rs \text{ xor } rt)$
<code>andi</code>	$rt \leftarrow rs \& \text{uext}(\text{imm})$	<code>xori</code>	$rt \leftarrow rs \text{ xor } \text{uext}(\text{imm})$
<code>lui</code>	$rt \leftarrow \{\text{imm}, 16'b0\}$	<code>sll</code>	$rd \leftarrow rt \ll sa$
<code>srl</code>	$rd \leftarrow t \gg sa$	<code>sra</code>	$rd \leftarrow st \gg sa$
<code>sllv</code>	$rd \leftarrow rt \ll rs[4:0]$	<code>rd ← srlv</code>	$rt \gg rs[4:0]$
<code>srav</code>	$rd \leftarrow rt \gg rs[4:0]$	<code>pref</code>	<code>nop</code>

实现了这些指令的同时，也实现了 `nop`, `ssnop`, `move` 等指令（语法糖）。这些指令的解耦方式基本上就是经历上述了五个模块，在取码后译码执行并回写。

5.2.1.2 数据相关问题

数据冲突指的是流水线中执行的几条指令中，一条指令依赖于前面指令的执行结果，具体又分为 `RAW`, `WAR`, `WAW` 三种情况。以如下代码为例：

```
ori $1, $0, 0x1100
ori $2, $1, 0x0020
```

则在第二条指令的译码阶段，第一条指令还没到达回写阶段导致取得的值不正确。译码阶段和回写阶段相差了两个时钟周期，如果使用插入空指令的方法解决，则浪费时间与空间。我们通过数据前推，将计算结果从执行和访存阶段直接更新到后面指令的译码阶段，从而使其获得正确的结果。代码实现可见5.1.2节，原理如下图所示。

5.2.2 移位指令

移位指令共 6 条如下。实现时，在译码阶段确定是否需要转移。对于 hi,lo 的移位操作，也需要考虑上述的数据相关问题。

movn	$rd \leftarrow rs \text{ if } rt \neq 0$	movz	$rd \leftarrow rs \text{ if } rt = 0$
mfhi	$rd \leftarrow hi$	mflo	$rd \leftarrow lo$
mthi	$hi \leftarrow rs$	mtlo	$lo \leftarrow rs$

5.2.3 算术指令

5.2.3.1 基本实现

简单的算术指令共 15 条，实现时注意注意手册对指令的定义。**add** 指令在加法溢出时不保存结果（溢出异常没有实现），**addu** 则不进行溢出检查。其它指令也类似如此。比较特别的指令有 **slt** 保存大小比较结果以及 **clo** 保存高位 1 的个数，**clz** 保存高位 0 的个数。

add(u)	$rd \leftarrow rs + rt$	sub(u)	$rd \leftarrow rs - rt$
slt(u)	$rd \leftarrow rs < rt$	addi(u)	$rd \leftarrow rs + \text{uext}(\text{imm})$
slti(u)	$rt \leftarrow rs < \text{uext}(\text{imm})$	clo	高位连续 1 的个数
clz	高位连续 0 的个数	mul	$rd \leftarrow rs * rt$
mult(u)	$\{hi, lo\} \leftarrow rs * rt$	div(u)	$\{hi, lo\} \leftarrow rs / rt$

5.2.3.2 除法

我们使用了一种朴素的方法来实现除法。试商法需要 32 个周期进行除法运算，流程如下。为实现除法模块，我们参考了试商法的状态机设计，并添加流水线控制模块，当进入除法模块后停止整个流水线的运行，直到除法完成再恢复。可见本设计中除法的代价是很大的。

5.2.4 跳转指令

5.2.4.1 基本实现

跳转指令共 4 条，分支指令 10 条这里不具体列出，指令的区别在于分支条件不同以及是否保存返回地址。在取指阶段判断是否跳转，如果需要则暂停流水线的运行，以解决结构冲突。以 **bltzal** 指令为例，该指令发生跳转当且仅当 $rs < 0$ ，且保存指令后地址为返回地址到第 31 号寄存器。跳转部分代码可见 5.1.1 节。

```

`EXE_BLTZAL: begin
    wreg_o <= `WRITEENABLE; //需要写入寄存器
    aluop_o <= `EXE_BLTZAL_OP; //操作指令
    alusel_o <= `EXE_RES_JUMP; //操作类型
    reg1_read_o <= `READENABLE; //需要读一个寄存器
    reg2_read_o <= `READDISABLE;
    wd_o <= 5'b11111; //需要将返回地址写入31号寄存器
    link_addr_o <= pc_plus_8; //返回地址
    instvalid <= `INSTVALID; //指令有效
    if (reg1_o[31]==1'b1) begin
        jmp_target_address_o <= pc_plus_4 + imm_sll2_signedext; //跳转地址
        jmp_flag_o <= `JMP; //需要跳转
        next_inst_in_delayslot_o <= `INDELAYSLOT; //下一条指令处于延迟槽
    end
end

```

和

```

always @(*) begin
    if (rst == `RSTENABLE) begin
        stall <= 6'b000000;
    end else if (stallreq_ex == `STOP) begin
        stall <= 6'b001111; //来自执行模块的停止流水线请求
    end else if (stallreq_id == `STOP) begin
        stall <= 6'b000111; //来自译码模块的停止流水线请求
    end else begin
        stall <= 6'b000000;
    end
end

```

5.2.5 访存指令

访存过程中，我们使用 IP 核作为储存器，让 IP 核的频率为 CPU 的两倍，可实现同步访存。访存指令共 8 条如下：

lb	字节加载	lbu	无符号字节加载
lh	半字加载	lhu	半字无符号加载
lw	字加载	sw	字储存
sb	字节储存	sh	半字储存

以字节加载指令为例，其代码如下

```

case (aluop_i)
`EXE_LB_OP: begin
    mem_addr_o <= mem_addr_i;
    mem_we <= `WRITEDISABLE;
    mem_ce_o <= `CHIPENABLE;

```

```

    case (mem_addr_i[1:0]) //根据最后两位选择写入值和选择变量
      2'b00: begin
        wdata_o <= {{24{mem_data_i[31]}}, mem_data_i[31:24]};
        mem_sel_o <= 4'b1000;
      end
      2'b01: begin
        wdata_o <= {{24{mem_data_i[23]}}, mem_data_i[23:16]};
        mem_sel_o <= 4'b0100;
      end
      2'b10: begin
        wdata_o <= {{24{mem_data_i[15]}}, mem_data_i[15:8]};
        mem_sel_o <= 4'b0010;
      end
      2'b11: begin
        wdata_o <= {{24{mem_data_i[7]}}, mem_data_i[7:0]};
        mem_sel_o <= 4'b0001;
      end
      default: begin
        wdata_o <= `ZEROWORD;
      end
    end
  ...
endcase
end

```

在访存指令中，也有一类数据相关，称为 **load** 相关，当需要使用的数据是由访存阶段读入时，就会出现此类问题。为此，当下一条指令需要用到上一条加载指令的结果时，我们也暂停流水线。

最终，我们的 CPU 的数据转移流大致如下。

5.3 指令测试

为了测试指令的正确性，我们利用仿真文件 `sopc_tst.v` 对文件夹 `program/tst` 下的汇编测试样例进行仿真，并编写了 `showreg.do` 的脚步文件显示需要观测的变量值。

5.4 ROM, RAM 与 IO

5.4.1 储存空间构建与 IO 配置

我们利用 IP 核构建储存空间。储存设备如下：

储存设备	大小	功能	初始内容
ROM	4KB	指令存储器	指令 (.text 节)
Writable ROM	1KB	可读写指令存储器 (支持 gdb)	无
RAM	128KB	内存	无
GRAM	1KB	显存 (存储字符)	开机画面

为实现 Writable ROM, 我们直接使用了 `reg` 型变量。对于其它储存器, 我们利用了 IP 核进行构成。类似于内存条的位扩展, 我们使用四个 IP 核构成 ROM。每个 ROM 中相邻的两位在地址上相差 4 位。

IO 设备复用了之前实验的代码, VGA 以字符为单位显示, 键盘支持全键位、组合键和大小写, 电子琴支持和声。将储存器与 CPU 打包成 `sopc`, 我们构筑整个系统结构如下。

5.4.2 MMIO

为了提供对设备的访问, 我们使用了内存映射管理所有 IO。通过一系列判断语句, 模拟了一个 MMIO 管理器, 使得可以通过访存指令获得键盘输入、向 VGA 写入字符、控制电子琴。具体分配如下。

ADDRESS	DESCRIPTION	FUNCTION
0x000000 - 0x003900	ROM	for instruction executable; lw only
0x003900 - 0x004000	RAM	for instruction executable; lw,sw only
0x004000 - 0x005000	GRAM	for video readable, writable; access by vga(640*480)
0x005000	keyboard enable	lb only
0x005004	keyboard ascii	lb only
0x005008	audio enable	wb only
- 0x006000	Reserve	
0x006000 - 0x008000	RAM:Data Section	
0x008000 - 0x010000	RAM:Heap	

5.5 运行时环境 AM

AM 的概念是在拔尖班余子濠老师在计算机系统课程的 PA 作业中介绍的, 我们摘录 PA 讲义部分如下:

在 PA 中, 我们补充完成了 `x86-nemu` 的 AM, 并完成了虚拟机上的虚拟机 `x86-navy` 的 AM。我们设想, 通过完成相应的 MIPS32 的 AM 的设计, 我们可以把 PA

中的小程序搬运到我们的 CPU 上（甚至仙剑奇侠传）。这个想法让我激动不已。然而由于没有实现中断以及时间问题，最终我们没有完成全部的设计。

我们实现了 AM 的 Turing Machine 的接口。

```
int video = 0x4001; //显存起始位置
void _putc(char ch) {
    asm volatile(
        "move $24, %0\n\t"
        "sb %1, 0x0($24)\n\t"
        :
        : "r"(video), "r"(ch)
    );
    video++;
}

void _trm_init() {
    asm volatile("ori $sp,$0,0x9000"); //设置栈区
    _halt(main());
}

void _halt(int code) {
    if (code) {
        _putc('W'); _putc('A'); //HIT THE BAD TRAP
    } else {
        _putc('A'); _putc('C'); //HIT THE GOOD TRAP
    }
    while (1);
}
```

在 loader.ld 文件中，将代码节，数据节设置到与上节表格相对应的位置，将起始符号改为 _trm_init，修改 Makefile 文件和 bash 脚本使其支持 mips32 AM 的编译，编译 cputest 下的 dummy.c 文件或 sum.c 文件，可以看到屏幕上出现了 AC。这就相当与 PA 实验中的 HIT THE GOOD TRAP. 由于已经有了 _putc 函数，我们可以调用在 PA 中自己写的 klib 中的 stdio.h 中的 printf 来输出我们想要的格式了。

理论上如果实现了系统调用，那么就可以把操作系统 Nanos-lite 和仙剑程序跑在我们的 CPU 上。可惜的是，余老师告诉我们由于没有 cache 和一些其它原因，这个过程还需要额外调用板上的设备以及及时跑起来可能也只能显示第一张图片就卡死了。考虑时间因素在内，我们最后放弃了这个实现。

如此一来，借助 AM 的力量，我们便可以把这两门课程的内容打通，从而更好的理解计算机的体系结构。

5.6 遇到的问题与解决方案

- 实现 CPU 时的困难：流水线停止，div 指令，累乘指令实现等，这些问题都有一定难度，通过阅读资料可以看到前人优秀的实现方法。通过学习可以用到我们的 CPU 中。
- gdb 的支持：原先的设计中代码区和数据区分类，代码区无法读写。为了实现对 gdb 的支持，强行用 reg 类型实现一个支持读写的代码段，以支持操作系统的设计。
- IP 核接口不足，无法同时支持 CPU 读写和 VGA 访问：由于显存本身较小，可以同时保存两份 IP 核的储存空间。虽然不是好方法，但是可以 work。
- MMIO 的调试：MMIO 的调试比其它指令的调试要复杂，因为此时仿真能提供的帮助很小。通过设计对于不同情况有不同反应的汇编代码和将一些信息直接从 LED 灯中输出可以帮助调试。

5.7 实验启示

- 项目可以提升自己的多方面能力：通过完成本次实验，不仅提高了自己的 Verilog 代码能力，加深对 CPU 和计算机体系的理解，还提高了自己的团队合作能力，项目维护能力，Makefile,loader 文件甚至是报告制作的能力。相比与老师提供的往年优秀实验报告的内容，我们两人组队不仅完成了其中三个人队伍的所有内容，还将 CPU 从单周期换位流水线并在各方面进行了拓展。
- RTFM, STFW：遇到问题时，读帮助文档和上网搜索问题资料往往能够解决。
- 课程的融汇贯通十分重要：将 ICS 课程中 AM 引入到数电实验中来，开启了新世界的大门。虽然我们并没有实现太多具体的程序，但是在以后学弟学妹的制作中，可以尝试把仙剑搬入数电实验之中。

第 6 章 软件部分

6.1 操作系统的定义

操作系统（英语：operating system，缩写作 OS）是管理计算机硬件与软件资源的计算机程序，同时也是计算机系统的内核与基石。操作系统需要处理如管理与配置内存、决定系统资源供需的优先次序、控制输入与输出设备、操作网络与管理文件系统等基本事务。操作系统也提供一个让用户与系统交互的操作界面。

6.2 操作系统中的约定

\$0	恒为 0	\$8	Fib temp data	\$16	gdb temp data	\$24	列标号
\$1	留作编译器使用	\$9	Fib temp data	\$17	gdb/testcase ret addr	\$25	行标号
\$2	上一个 ASCII 输入	\$10	Fib temp data	\$18	gdb temp data	\$26	Mu temp data
\$3	sys temp data	\$11	Fib temp data	\$19	gdb temp data	\$27	未处理行指针寄存器
\$4	sys temp data	\$12	Fib temp data	\$20	gdb temp data	\$28	ENTER 信号
\$5	sys temp data	\$13	Fib temp data(10)	\$21	Testcase use	\$29	恒为 4
\$6	sys temp data	\$14	gdb temp data(addr)	\$22	Testcase use	\$30	栈指针
\$7	在用户程序中保存 \$ra	\$15	gdb temp data(addr)	\$23	ret data	\$31	ret addr(\$ra)

6.3 屏幕显示功能

最主要的是实现控制流的可控，保证其在主循环的轨道内运行。我们将输入分为三种情形：1. 不以./开头，我们认为这是普通文本，不执行比对。2. 某一个命令，则此时执行对应命令。3. 错误命令，则此时输出错误信息。最终均会回到主循环。

代码如下（这里省略了有关具体比对的细节）。值得注意的是，我们系统中没有一个通常意义上的键盘缓冲区，我们是借显存作为键盘缓冲区的，这就是为什么我们在寄存器堆中需要一个未处理行指针寄存器了。

读取键盘输入的函数十分简单，如图??所示。而输出函数逻辑较为复杂，如??所示。函数主体约为如上。这部分的功能几乎与实验 11 等价。

6.4 斐波那契数

逻辑十分简单，首先接受输入，检测到 ENTER 信号之后从显存中读取数据，之后迭代运算，最后将结果除 10，余数放入栈中，结果参与下一次除法直到为 0，最后从栈中获得数据输出，形成十进制输出的情形。第一部分，接收数据，等待 ENTER 输入，如图????。第二部分，ENTER 信号产生并被接收后跳转于此，此时数据已经完整，进入迭代，如图??。第三部分，另一个将数据写入栈中的循环，为输出做准备，如图??。最后一部分，从栈中得到数据并不断将其写入显存（调用系统函数）由于最后写入了 ENTER 信号，这意味着我们最终要将这一个额外的信号清除，将未处理行指针寄存器写入正确的值。

6.5 Hello World

Hello world 开机界面和命令展示如????

6.6 电子琴

电子琴程序的功能就是打开电子琴使能，并构建一个检测到 backspace 输入退出的程序

6.7 单步调试

我们的单步调试实现受到了操作系统异常处理程序的启发。

而在我们的实现中，由于没有完成异常处理相关的协处理器与指令，这使得我们不得不采用一种模拟的手段来实现相关机制。在实现中，有 testcase 存放在 0x2f00 的位置，大体流程如??

这三条特殊的指令为:

```
.org 0x2ef0
    nop
    jr  $ra          # jump to $ra
    ori $17,$ra,0x0
```

`nop` 的位置就是将来会被替代的位置。

`gdb` 相当于一个监视者，一次只允许一条指令的运行，当然这样实现是不完美的，他无法处理程序中有跳转的情况，在顺序执行的指令中，它等价于图??

6.8 遇到的问题与解决方案

MARS 不支持高级编译选项，所以我们搭建了一个交叉编译工具链。运用 `makefile` 实现了流程化的操作，当然，我们自己也写了一些小工具用以将各种初始化文件格式转化为了 `mif`，见图??(这部分请参考我写的 `mips` 交叉编译工具链的配置与使用——基于Ubuntu18.04)

有时寄存器在使用时的未初始化会造成相当令人意外的结果，这时就要严格执行所用寄存器的保存于初始化。

MARS 不支持高级编译选项，`gdb` 无法在此调试：这使得我们只好在实机上调试，唯一问题就是每一次编译都十分缓慢，这就要求我们预先写好 `testcase`，从简单开始，这样可以大大降低复杂度，有效排除硬件问题。

6.9 实验启示

一定要多交流，有时你想不到的，或许别人就有一个很好的解决方案，互相查漏补缺，效率是很高的。

第 7 章 文件结构

以下列出了重要文件及其内容。在主文件夹中大量文件属于 IP 核，为自动生成的文件。各部分文件夹下有 Verilog 文件和 mif 初始化文件，cpu 文件夹下为 CPU 模块，具体见硬件内容描述。os.s 为操作系统代码。

```

NJU_MIPS
├── README.md
├── macro.v..... 宏定义
├── NJU_MIPS.v..... 顶层模块
├── sopc.v..... CPU 和储存器模块
├── cpu..... MIPS 五级流水线 CPU 代码
│   ├── cpu.v..... CPU 顶层模块
│   ├── data_RAM..... 128K RAM 代码
│   ├── inst_RAM..... 指令储存器代码
│   ├── keyboard..... 键盘代码
│   ├── vga..... VGA 代码
│   ├── audio..... 电子琴代码
│   ├── simulation..... 仿真文件
│   │   └── modelsim
│   │       └── sopc_tst.v..... 对 sopc.v 的测试文件
│   ├── program..... 初始化代码及生成工具
│   ├── tst..... 测试样例
│   ├── os.s..... 操作系统代码
│   ├── inst_rom.mif..... 初始化文件
├── AM..... 运行时环境（修改自 PA 实验）
│   ├── am
│   │   └── arch
│   │       └── mips32..... MIPS32 运行时环境
│   └── tests..... 测试文件

```

第 8 章 代码样式

Algorithm 1 test

```

if  $i \geqslant \text{maxval}$  then
     $i \leftarrow 0$ 
else
    if  $i + k \leqslant \text{maxval}$  then
         $i \leftarrow i + k$ 
    end if
end if

```

```

>> hello
>> python
shell code

```

```

root $>    ls -al
root $>    cd /usr/lib

```

```

\setvar{\myname}{\name}
\newcommand\myaddress{}
\newcommand{\address}[1]{\renewcommand{\myaddress}{\#1}}

```

```

1  for (int k=0;k<n;++k){
2      for (int i=0;i<n;++i){
3          for (int j=0;j<n;++j){
4              a[i][j][k] = 0;
5          }
6      }
7  }
8  // 这个不是Floyed

```

```

#include <bits/stdc++.h>
using namespace std;
int main(){
    cout << "Hello World!" << endl;
    return 0;
}

```

```

mov %eax, $ebx
sub 3, %eax

```

Algorithm 2 代码标题

```
1   for (int k=0;k<n;++k){
2       for (int i=0;i<n;++i){
3           for (int j=0;j<n;++j){
4               a[i][j][k] = 0;
5           }
6       }
7   }
8   // 这个不是Floyed
9
```

```
import pandas as pd
pd.read_csv("name.csv")
```

```
always @ (posedge clk) begin
if (rst == `RSTENABLE) begin
    ce <= `CHIPDISABLE;
end else begin
    ce <= `CHIPENABLE;
end
end
```


插图索引

图 2.1	利用 Xfig 制图	26
图 2.2	包含子图形的大图形 (subcaptionbox 示例).....	27
图 2.3	包含子图形的大图形 (subfigure 示例)	27
图 2.4	并排第一个图	28
图 2.5	并排第二个图	28

表格索引

表 1.1	模板文件	15
表 1.2	复杂表格示例 1。这个引用 Knuth (1989) 不会导致编号混乱。	15
表 1.3	第一个并排子表格	16
表 1.4	第二个并排子表格	16
表 1.5	并排子表格.....	17
表 1.6	复杂表格示例 2.....	17
表 1.7	实验数据	18

公式索引

公式 1-1	19
公式 1-2	19
公式 1-3	20
公式 1-4	20
公式 1-5	20
公式 1-6	20
公式 1-7	20
公式 1-8	20
公式 1-9a	21
公式 1-9b	21
公式 1-10	21
公式 1-11	21
公式 1-12	22
公式 1-13	23
公式 1-14	23
公式 1-15	24
公式 2-1	25
公式 A-1	55
公式 A-2	56

参考文献

- Knuth D E. The \TeX book[M]. 15th ed. Reading, MA: Addison-Wesley Publishing Company, 1989
- Goosens M, Mittelbach F, Samarin A. The \LaTeX companion[M]. Reading, MA: Addison-Wesley Publishing Company, 1994: 112-125
- Gröning P, Nilsson L, Ruffieux P, et al. Encyclopedia of nanoscience and nanotechnology: volume 1 [M]. USA: American Scientific Publishers, 2004: 547-579
- Krasnogor N. Towards robust memetic algorithms[M]//Hart W, Krasnogor N, Smith J. Studies in Fuzziness and Soft Computing: volume 166 Recent Advances in Memetic Algorithms. New York: Springer Berlin Heidelberg, 2004: 185-207
- 阎真. 沧浪之水[M]. 北京: 人民文学出版社, 2001: 185-207
- 班固. 苏武传[M]//郑在瀛, 汪超宏, 周文复. 新古文观止丛书: 第2卷 传记散文英华. 武汉: 湖北人民出版社, 1998: 65-69
- Chafik El Idrissi M, Roney A, Frigon C, et al. Measurements of total kinetic-energy released to the $N = 2$ dissociation limit of H_2 — evidence of the dissociation of very high vibrational Rydberg states of H_2 by doubly-excited states[J]. Chemical Physics Letters, 1994, 224(10):260-266.
- Mellinger A, Vidal C R, Jungen C. Laser reduced fluorescence study of the carbon-monoxide nd triplet Rydberg series-experimental results and multichannel quantum-defect analysis[J]. J. Chem. Phys., 1996, 104(5):8913-8921.
- Shell M. How to use the IEEEtran \LaTeX class[J]. Journal of \LaTeX Class Files, 2002, 12(4):100-120.
- 猪八戒. 论流体食物的持久保存[D]. 北京: 广寒宫大学, 2005.
- Jeyakumar A R. Metamori: A library for incremental file checkpointing[D]. Blacksburg: Virginia Tech, 2004.
- 沙和尚. 论流沙河的综合治理[D]. 北京: 清华大学, 2005.
- Zadok E. FiST: A System for Stackable File System Code Generation[D]. USA: Computer Science Department, Columbia University, 2001.
- IEEE Std 1363-2000. IEEE standard specifications for public-key cryptography[M]. New York: IEEE, 2000
- Kim S, Woo N, Yeom H Y, et al. Design and Implementation of Dynamic Process Management for Grid-enabled MPICH[C]//Dongarra J, Laforenza D, Orlando S. the 10th European PVM/MPI Users' Group Conference. Venice, Italy: Springer-Verlag, 2003.
- Kocher C, Jaffe J, Jun B. Differential power analysis[C]//Wiener M. Lecture Notes in Computer Science: volume 1666 Advances in Cryptology (CRYPTO '99). Germany: Springer-Verlag, 1999: 388-397.
- Woo A, Bailey D, Yarrow M, et al. The NAS parallel benchmarks 2.0[R/OL]. The Pennsylvania State University CiteSeer Archives, 1995. <http://www.nasa.org/>.
- 萧钰. 出版业信息化迈入快车道[EB/OL]. <http://www.creader.com/news/200112190019.htm>.

Online Computer Library Center, Inc. History of OCLC[EB/OL]. <http://www.oclc.org/about/history/default.htm>.

贾宝玉, 林黛玉, 薛宝钗, 等. 论刘姥姥食量大如牛之现实意义[J]. 红楼梦杂谈, 1800, 224:260-266.

王重阳, 黄药师, 欧阳峰, 等. 武林高手从入门到精通[C]//第 N 次华山论剑. 西安, 中国: 金大庸, 2006.

薛瑞尼. ThuThesis: 清华大学学位论文模板[EB/OL]. 2017. <https://github.com/xueruini/thuthesis>.

致 谢

衷心感谢导师 xxx 教授和物理系 xxx 副教授对本人的精心指导。他们的言传身教将使我终生受益。

在美国麻省理工学院化学系进行九个月的合作研究期间，承蒙 xxx 教授热心指导与帮助，不胜感激。感谢 xx 实验室主任 xx 教授，以及实验室全体老师和同学们的热情帮助和支持！本课题承蒙国家自然科学基金资助，特此致谢。

感谢 L^AT_EX 和 thuthesis 薛瑞尼 (2017)，帮我节省了不少时间。

声 明

本人郑重声明：所呈交的学位论文，是本人在导师指导下，独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。

签 名：_____ 日 期：_____

附录 A 外文资料原文

The title of the English paper

Abstract: As one of the most widely used techniques in operations research, *mathematical programming* is defined as a means of maximizing a quantity known as *objective function*, subject to a set of constraints represented by equations and inequalities. Some known subtopics of mathematical programming are linear programming, nonlinear programming, multiobjective programming, goal programming, dynamic programming, and multilevel programming^[1].

It is impossible to cover in a single chapter every concept of mathematical programming. This chapter introduces only the basic concepts and techniques of mathematical programming such that readers gain an understanding of them throughout the book^[2,3].

A.1 Single-Objective Programming

The general form of single-objective programming (SOP) is written as follows,

$$\begin{cases} \max f(x) \\ \text{subject to:} \\ g_j(x) \leq 0, \quad j = 1, 2, \dots, p \end{cases} \quad (123)$$

which maximizes a real-valued function f of $x = (x_1, x_2, \dots, x_n)$ subject to a set of constraints.

Definition A.1: In SOP, we call x a decision vector, and x_1, x_2, \dots, x_n decision variables. The function f is called the objective function. The set

$$S = \{x \in \mathbf{R}^n \mid g_j(x) \leq 0, j = 1, 2, \dots, p\} \quad (456)$$

is called the feasible set. An element x in S is called a feasible solution.

Definition A.2: A feasible solution x^* is called the optimal solution of SOP if and only if

$$f(x^*) \geq f(x) \quad (\text{A-1})$$

for any feasible solution x .

One of the outstanding contributions to mathematical programming was known as the Kuhn-Tucker conditions A-2. In order to introduce them, let us give some definitions. An inequality constraint $g_j(x) \leq 0$ is said to be active at a point x^* if $g_j(x^*) = 0$. A point x^* satisfying $g_j(x^*) \leq 0$ is said to be regular if the gradient vectors $\nabla g_j(x)$ of all active constraints are linearly independent.

Let x^* be a regular point of the constraints of SOP and assume that all the functions $f(x)$ and $g_j(x)$, $j = 1, 2, \dots, p$ are differentiable. If x^* is a local optimal solution, then there exist Lagrange multipliers λ_j , $j = 1, 2, \dots, p$ such that the following Kuhn-Tucker conditions hold,

$$\begin{cases} \nabla f(x^*) - \sum_{j=1}^p \lambda_j \nabla g_j(x^*) = 0 \\ \lambda_j g_j(x^*) = 0, \quad j = 1, 2, \dots, p \\ \lambda_j \geq 0, \quad j = 1, 2, \dots, p. \end{cases} \quad (\text{A-2})$$

If all the functions $f(x)$ and $g_j(x)$, $j = 1, 2, \dots, p$ are convex and differentiable, and the point x^* satisfies the Kuhn-Tucker conditions (A-2), then it has been proved that the point x^* is a global optimal solution of SOP.

A.1.1 Linear Programming

If the functions $f(x)$, $g_j(x)$, $j = 1, 2, \dots, p$ are all linear, then SOP is called a *linear programming*.

The feasible set of linear is always convex. A point x is called an extreme point of convex set S if $x \in S$ and x cannot be expressed as a convex combination of two points in S . It has been shown that the optimal solution to linear programming corresponds to an extreme point of its feasible set provided that the feasible set S is bounded. This fact is the basis of the *simplex algorithm* which was developed by Dantzig as a very efficient method for solving linear programming.

Roughly speaking, the simplex algorithm examines only the extreme points of the feasible set, rather than all feasible points. At first, the simplex algorithm selects an extreme point as the initial point. The successive extreme point is selected so as to improve the objective function value. The procedure is repeated until no improvement in objective function value can be made. The last extreme point is the optimal solution.

Table 1 This is an example for manually numbered table, which would not appear in the list of tables

Network Topology		# of nodes	# of clients			Server
GT-ITM	Waxman Transit-Stub	600	2%	10%	50%	Max. Connectivity
Inet-2.1		6000				
Xue	Rui	Ni	ThuThesis			
	ABCDEF					

A.1.2 Nonlinear Programming

If at least one of the functions $f(x), g_j(x), j = 1, 2, \dots, p$ is nonlinear, then SOP is called a *nonlinear programming*.

A large number of classical optimization methods have been developed to treat special-structural nonlinear programming based on the mathematical theory concerned with analyzing the structure of problems.



Figure 1 This is an example for manually numbered figure, which would not appear in the list of figures

Now we consider a nonlinear programming which is confronted solely with maximizing a real-valued function with domain \mathbf{R}^n . Whether derivatives are available or not, the usual strategy is first to select a point in \mathbf{R}^n which is thought to be the most likely place where the maximum exists. If there is no information available on which to base such a selection, a point is chosen at random. From this first point an attempt is made to construct a sequence of points, each of which yields an improved objective function value over its predecessor. The next point to be added to the sequence is chosen by analyzing the behavior of the function at the previous points. This construction continues until some termination criterion is met. Methods based upon this strategy are called *ascent methods*, which can be classified as *direct methods*, *gradient methods*, and *Hessian methods* according to the information about the behavior of objective function f . Direct methods require only that the function can be evaluated at each point. Gradient methods require the evaluation of first derivatives of f . Hessian methods require the evaluation of second derivatives. In fact, there is no superior method for all problems. The efficiency of a method is very much dependent upon the objective function.

A.1.3 Integer Programming

Integer programming is a special mathematical programming in which all of the variables are assumed to be only integer values. When there are not only integer variables but also conventional continuous variables, we call it *mixed integer programming*. If all the variables are assumed either 0 or 1, then the problem is termed a *zero-one programming*. Although integer programming can be solved by an *exhaustive enumeration* theoretically, it is impractical to solve realistically sized integer programming problems. The most successful algorithm so far found to solve integer programming is called the *branch-and-bound enumeration* developed by Balas (1965) and Dakin (1965). The other technique to integer programming is the *cutting plane method* developed by Gomory (1959).

Uncertain Programming (BaoDing Liu, 2006.2)

References

NOTE: These references are only for demonstration. They are not real citations in the original text.

- [1] Donald E. Knuth. The \TeX book. Addison-Wesley, 1984. ISBN: 0-201-13448-9
- [2] Paul W. Abrahams, Karl Berry and Kathryn A. Hargreaves. \TeX for the Impatient. Addison-

Wesley, 1990. ISBN: 0-201-51375-7

- [3] David Salomon. The advanced \TeX book. New York : Springer, 1995. ISBN:0-387-94556-3

附录 B 外文资料的调研阅读报告或书面翻译

英文资料的中文标题

摘要：本章为外文资料翻译内容。如果有摘要可以直接写上来，这部分好像没有明确的规定。

B.1 单目标规划

北冥有鱼，其名为鲲。鲲之大，不知其几千里也。化而为鸟，其名为鹏。鹏之背，不知其几千里也。怒而飞，其翼若垂天之云。是鸟也，海运则将徙于南冥。南冥者，天池也。

$$p(y|\mathbf{x}) = \frac{p(\mathbf{x}, y)}{p(\mathbf{x})} = \frac{p(\mathbf{x}|y)p(y)}{p(\mathbf{x})} \quad (123)$$

吾生也有涯，而知也无涯。以有涯随无涯，殆已！已而为知者，殆而已矣！为善无近名，为恶无近刑，缘督以为经，可以保身，可以全生，可以养亲，可以尽年。

B.1.1 线性规划

庖丁为文惠君解牛，手之所触，肩之所倚，足之所履，膝之所倚，砉然响然，奏刀騞然，莫不中音，合于桑林之舞，乃中经首之会。

表 1 这是手动编号但不出现在索引中的一个表格例子

Network Topology		# of nodes	# of clients			Server
GT-ITM	Waxman Transit-Stub	600	2%	10%	50%	Max. Connectivity
Inet-2.1		6000				
Xue	Rui	Ni	ThuThesis			
	ABCDEF					

文惠君曰：“嘻，善哉！技盖至此乎？”庖丁释刀对曰：“臣之所好者道也，进乎技矣。始臣之解牛之时，所见无非全牛者；三年之后，未尝见全牛也；方今之时，臣以神遇而不以目视，官知止而神欲行。依乎天理，批大郤，导大窾，因其固然。技经肯綮之未尝，而况大瓠乎！良庖岁更刀，割也；族庖月更刀，折也；今臣之刀十九年矣，所解数千牛矣，而刀刃若新发于硎。彼节者有间而刀刃者无厚，以无厚

入有间，恢恢乎其于游刃必有余地矣。是以十九年而刀刃若新发于硎。虽然，每至于族，吾见其难为，怵然为戒，视为止，行为迟，动刀甚微，謦然已解，如土委地。提刀而立，为之而四顾，为之踌躇满志，善刀而藏之。”

文惠君曰：“善哉！吾闻庖丁之言，得养生焉。”

B.1.2 非线性规划

孔子与柳下季为友，柳下季之弟名曰盗跖。盗跖从卒九千人，横行天下，侵暴诸侯。穴室枢户，驱人牛马，取人妇女。贪得忘亲，不顾父母兄弟，不祭先祖。所过之邑，大国守城，小国入保，万民苦之。孔子谓柳下季曰：“夫为人父者，必能诏其子；为人兄者，必能教其弟。若父不能诏其子，兄不能教其弟，则无贵父子兄弟之亲矣。今先生，世之才士也，弟为盗跖，为天下害，而弗能教也，丘窃为先生羞之。丘请为先生往说之。”



图 1 这是手动编号但不出现索引中的图片的例子

柳下季曰：“先生言为人父者必能诏其子，为人兄者必能教其弟，若子不听父之诏，弟不受兄之教，虽今先生之辩，将奈之何哉？且跖之为人也，心如涌泉，意

如飘风，强足以距敌，辩足以饰非。顺其心则喜，逆其心则怒，易辱人以言。先生必无往。”

孔子不听，颜回为驭，子贡为右，往见盗跖。

B.1.3 整数规划

盗跖乃方休卒徒大山之阳，脍人肝而脯之。孔子下车而前，见谒者曰：“鲁人孔丘，闻将军高义，敬再拜谒者。”谒者入通。盗跖闻之大怒，目如明星，发上指冠，曰：“此夫鲁国之巧伪人孔丘非邪？为我告之：尔作言造语，妄称文、武，冠枝木之冠，带死牛之胁，多辞缪说，不耕而食，不织而衣，摇唇鼓舌，擅生是非，以迷天下之主，使天下学士不反其本，妄作孝弟，而侥幸于封侯富贵者也。子之罪大极重，疾走归！不然，我将以子肝益昼脯之膳。”

附录 C 其它附录

前面两个附录主要是给本科生做例子。其它附录的内容可以放到这里，当然如果你愿意，可以把这部分也放到独立的文件中，然后将其 `\input` 到主文件中。