服务器风冷的模型预测控制

看见两分钟后的温度

从能量守恒到线性 MPC，再到自适应、深度学习与强化学习。
全文只用一台风冷服务器作为例子，只假设读者熟悉微积分。

2026 年 10 月

目录

前言 4

一 问题 5

1 风扇的两难 5

2 温度服从的方程 6

2.1 能量守恒 6

2.2 风扇固定时温度怎样变化 7

3 只看温度的控制器 8

二 构造 MPC 10

4 MPC 的工作方式 10

5 离散化：从微分方程到一步公式 11

6 预测：把一步公式串起来 11

7 代价函数：把目标写成一个数 12

三 求解：为什么要线性化 13

8 求最小值：令导数为零 13

8.1 一个变量 13

8.2 两个变量 15

8.3 导数为零的点一定是最低点吗 16

9 线性化：在当前工况附近取切平面 17

9.1 展开点怎么选 17

9.2 一步公式的切平面 17

10 线性预测：温度成为风量的一次函数 18

11 线性化之后：一组一次方程 20

11.1 代入代价函数，令偏导数为零 20

11.2 凸性得到保证 21

11.3 两种模型的比较与线性化的代价 21

四 MPC 在运行中 23

12 滚动执行：每个周期只动一次风扇 23

13 提前动作：负载到来之前加速风扇 24

13.1 仿真中的现象 24

13.2 一个两步的例子 . . . 24
13.3 提前动作的三个条件 . . . 26
14 仿真比较 . . . 26
15 负载预报从哪里来 . . . 28
15.1 三类负载信息 . . . 28
15.2 没有预报时：预测负载与预测温度 . . . 28
15.3 突变的安全余量 . . . 29
15.4 预报出错的代价 . . . 30
15.5 把不确定性写进约束：鲁棒与随机 MPC . . . 30
16 预测 N 步的前提：惯性 . . . 31
16.1 没有惯性会怎样 . . . 31
16.2 惯性决定预测要看多远 . . . 31
16.3 只有惯性还不够 . . . 32
16.4 MPC 的价值在哪里 . . . 32
五 拓展：把数据放进 MPC 34
17 MPC 里哪些东西是手写的 . . . 34
17.1 把 MPC 写成一行 . . . 34
17.2 每一样都可以从数据中学 . . . 34
17.3 两个求最小值的问题 . . . 35
18 学参数：自适应 MPC . . . 36
18.1 模型为什么会过时 . . . 36
18.2 参数不准的后果 . . . 36
18.3 从数据估计参数 . . . 37
18.4 注意事项 . . . 37
19 学形状：用神经网络补全一步公式 . . . 38
19.1 物理公式漏掉的东西 . . . 38
19.2 为什么调一个参数不够 . . . 38
19.3 神经网络：一个可以调的函数 . . . 39
19.4 训练：又一个求最小值的问题 . . . 39
19.5 放进 MPC：照第三部分的步骤走 . . . 40
19.6 结果 . . . 42
19.7 为什么保留物理公式 . . . 43
20 学负载：预测未来的 P_k . . . 43

21 学终端代价：强化学习的位置 . . . 43
21.1 窗口太短会怎样 . . . 43
21.2 终端代价：替窗口之外的事情记账 . . . 44
21.3 终端代价从哪里来：强化学习 . . . 44
21.4 MPC 与强化学习是一条线的两端 . . . 45
21.5 阶段代价里能学什么 . . . 45
22 学整个控制律：模仿学习与安全层 . . . 46
23 从哪里开始：先看预测误差 . . . 46

附录 **48**

A 一阶线性常微分方程的解 . . . 48
B 泰勒展开、偏导数与求导法则 . . . 48
C 向量、矩阵与线性方程组 . . . 49
D 凸函数与凸集 . . . 49
E 牛顿法 . . . 50
F PI 控制 . . . 50
G 二次规划与求解器 . . . 51
H 欧拉法的精度与采样周期 . . . 51
I 无约束线性 MPC 等价于固定增益的反馈 . . . 52
J 带遗忘因子的最小二乘 . . . 52
K 神经网络与梯度下降 . . . 53
L 仿真设置 . . . 53

前言

服务器的风扇控制器面对的是一个时间错位的问题。风扇转速一秒之内就能改变，CPU 的功耗一毫秒之内就能翻倍，可散热器和 CPU 这块金属的温度要一到两分钟才跟得上。一个只盯着当前温度的控制器，看到的总是一两分钟以前的决定所造成的结果。

模型预测控制（Model Predictive Control，MPC）应对这种错位的办法很直接：用物理模型把未来两分钟的温度算出来，在所有可能的风扇方案中挑出最省电又不超温的一个，执行它的第一步，到下一个采样时刻再重新算一遍。本文的标题指的就是这件事。

全文以一台风冷服务器为唯一的例子，从能量守恒写起，依次回答五个问题（图 1）。

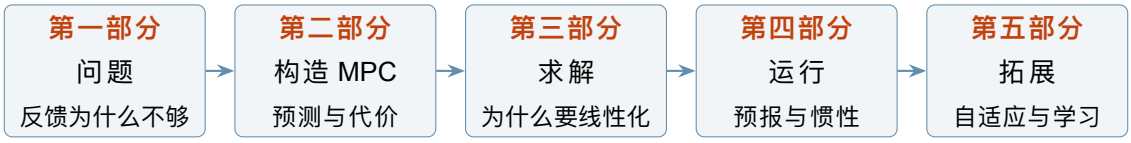


图 1 全文结构。

1. 只看温度的反馈控制差在哪里，MPC 多了什么？（第一、二部分）
2. 为什么实际使用的 MPC 几乎都要先线性化？（第三部分）
3. MPC 在服务器上能做到什么、做不到什么？负载预报从哪里来？预测 N 步依赖系统的什么性质？（第四部分）
4. 自适应、深度学习、强化学习这些技术，分别替换了 MPC 中的哪一部分，又是怎样接进去的？（第五部分）
5. 面对一台具体的服务器，怎样从运行数据判断该用哪一种方法？（第五部分末尾）

正文只假设读者熟悉微积分。矩阵、凸函数、牛顿法、最小二乘等补充知识放在附录中，需要时再查。全文的数字都来自同一组参数和同一套仿真，前后可以互相对照。

第一部分

问题

一台服务器的散热可以用一条一阶微分方程描述。本部分写出这条方程，读出它的三个关键性质，然后检验两种常用的反馈控制方法，找出它们共同的局限。

1　风扇的两难

服务器中的 CPU 在工作时持续发热。热量先传到贴在 CPU 上的散热器，散热器是一块带有许多鳍片的金属。风扇把冷空气吹过鳍片，空气被加热后从机箱后部排出，热量也就随之被带走（图 2）。

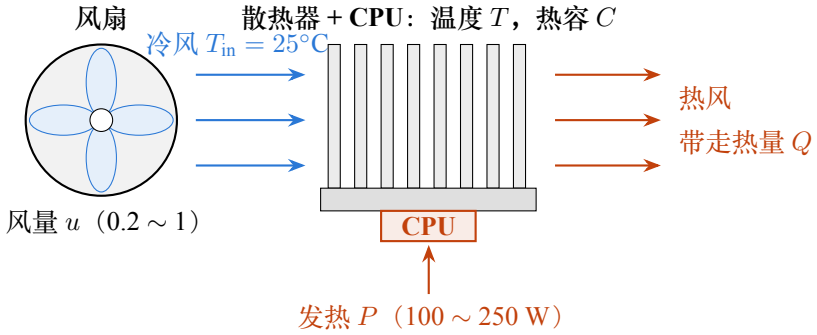


图 2　风冷服务器示意图。把 CPU 与散热器看成一块温度处处相同的金属，温度为 T。热量以功率 P 进入，以功率 Q 被风带走。

在这个过程中，我们能调节的只有风扇。用 u 表示归一化风量：$u = 1$ 为满速，$u = 0.2$ 为允许的最低转速。我们希望风扇的运行同时满足三个目标：

1. **安全**：温度不得超过 80°C。为留出余量，控制时按 78°C 执行。
2. **省电**：风扇功耗近似与 u^3 成正比（风机相似律），转得越快越费电。
3. **平稳**：转速不应忽高忽低，否则噪声大、磨损快。

前两个目标彼此冲突。风扇开得越小越省电，温度却越高。这个冲突有多尖锐，图 3 给出了答案：风量从 0.5 开到 1，散热能力只增加到 1.62 倍，风扇功耗却增加到 8 倍。风扇开大的回报越来越小，代价越来越大，多开的每一分都要算清楚值不值。

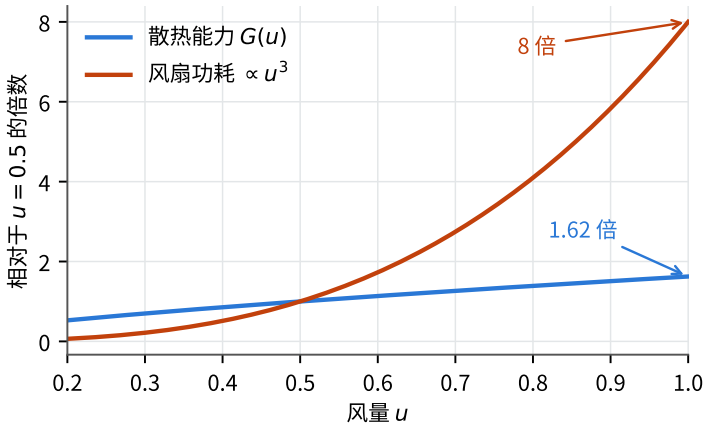


图 3　散热能力与风扇功耗随风量的变化，均以 $u = 0.5$ 为基准。

于是本文要回答的问题是：**在每一时刻，风扇应当以多大风量运行，才能在不超温的前提下尽量省电？**

2　温度服从的方程

2.1　能量守恒

要决定风扇怎样运行，先得知道风扇这样运行时温度会怎样变化。这一关系由能量守恒给出。

考察一段很短的时间 dt。这段时间内 CPU 送入热量 $P\,dt$，风带走热量 $Q\,dt$，两者之差使金属升温 dT。升温所需的热量为 $C\,dT$，其中 C（J/K）是热容，即 mc_p。于是

$$C\,dT = (P - Q)\,dt \quad\Longrightarrow\quad C\frac{dT}{dt} = P - Q. \tag{1}$$

风带走的热量服从牛顿冷却定律，与金属比进风高出的温度成正比：

$$Q = G(u)\,\bigl(T - T_{\text{in}}\bigr). \tag{2}$$

比例系数 $G(u)$ 称为**散热能力**，单位 W/K，表示温差每 1 K 每秒能带走多少焦耳。风量越大，G 越大，但两者并不成正比。强迫对流的经验关系为

$$G(u) = G_{\text{ref}}\left(\frac{u}{0.5}\right)^{0.7}, \qquad G_{\text{ref}} = 3.5\ \text{W/K}. \tag{3}$$

G_{ref} 是风量为 0.5 时的散热能力，由散热器和风道决定，第五部分会再次用到它。

合并以上各式，就得到本文唯一的物理方程：

$$\boxed{C\frac{dT}{dt} = P - G(u)\,\bigl(T - T_{\text{in}}\bigr)} \tag{4}$$

表 1　模型参数

符号	含义	取值
C	CPU 与散热器的热容	400 J/K
T_{in}	进风温度	25°C
P	CPU 功耗	空闲 100 W，繁忙 250 W
u	归一化风量	$0.2 \sim 1$
$G(u)$	散热能力	$1.84 \sim 5.69$ W/K

2.2　风扇固定时温度怎样变化

风量 u 和功耗 P 都不变时，方程 (4) 是一阶线性常微分方程，其解为（推导见附录 A）

$$T(t) = T_\infty + \left(T_0 - T_\infty\right) e^{-t/\tau}, \qquad T_\infty = T_{\text{in}} + \frac{P}{G(u)}, \qquad \tau = \frac{C}{G(u)}. \tag{5}$$

温度从初值 T_0 出发，按指数规律趋向稳态温度 T_∞。τ 称为时间常数，经过 τ 秒，温度走完全程的 63%。

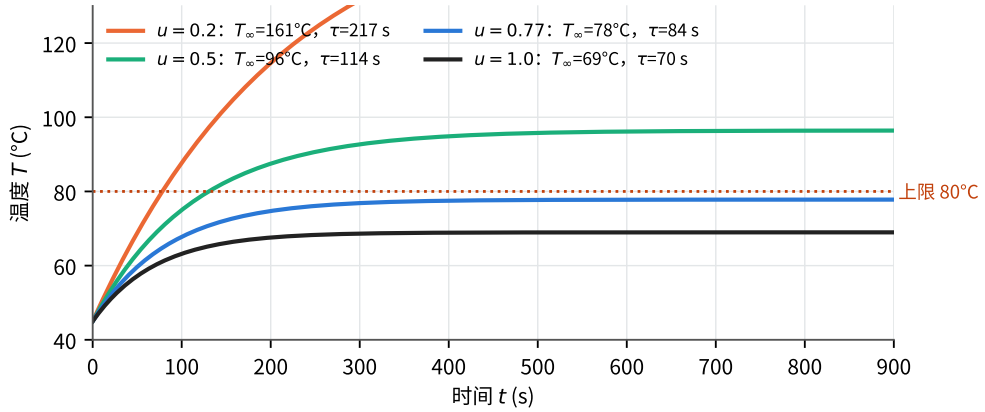


图 4　功耗 250 W、初始温度 45°C 时，不同风量下的温度变化。

从式 (5) 和图 4 可以读出三个性质，它们贯穿全文。

稳态温度只取决于功耗和风量。 繁忙时 $P = 250$ W，要使 $T_\infty \le 78$°C，需要 $G(u) \ge 250/53$，即 $u \ge 0.77$。空闲时 $P = 100$ W，即使风扇处于最低转速，稳态温度也有 79.3°C，所以风扇不能一直停在最低档。

温度变化很慢。 时间常数在 70 s（满速）到 217 s（最低速）之间。风扇调整之后，温度要一到两分钟才跟得上。

热容是一个缓冲池。 温度不会突变，热量可以暂时存在金属里，冷却也可以。这意味着风扇今天的动作，一两分钟之后仍在起作用。第四部分会看到，MPC 提前动作的能力全部建立在这一点上。

3　只看温度的控制器

如果简单方法已经够好，就不必引入 MPC。所以先考察两种常用的方法。它们都属于**反馈控制**，即只根据当前测得的温度决定风扇：

- **风扇曲线**：服务器管理控制器（BMC）的默认做法。温度 60°C 时风量 0.2，80°C 时风量 1，中间按直线插值。
- **PI 控制**：设定目标温度 T_{set}，根据误差 $T - T_{\text{set}}$ 及其累积量调节风扇（附录 F）。

测试场景在后文的所有仿真中保持不变：0 ~ 300 s 空闲（100 W），300 ~ 600 s 繁忙（250 W），之后恢复空闲。PI 的参数经过系统搜索，选取峰值温度不超过 78.5°C 条件下风扇能耗最低的一组，尽量对它公平。

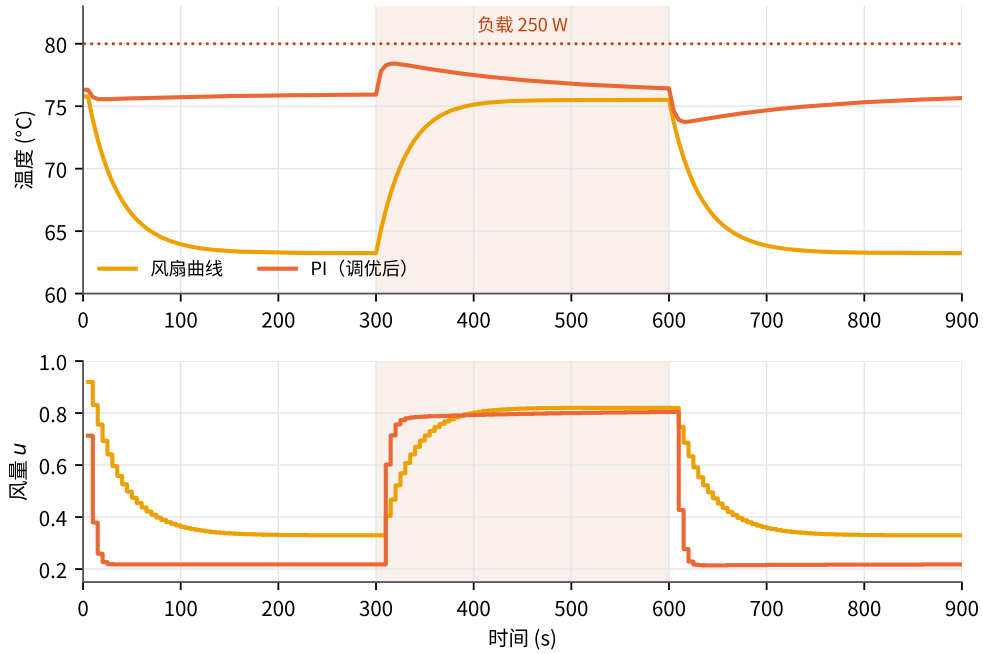


图 5　风扇曲线与 PI 控制。上：温度；下：风量。浅色区域为繁忙时段。

风扇曲线在空闲时把温度压在 63°C 左右，离 78°C 的限值还很远，风扇做了大量不必要的功。它只是一条固定的映射，并不知道限值在哪里。

PI 控制好一些，但负载到来时，温度先升上去，风扇才跟着加速。反馈只能在偏差出现之后补救。为了让升上去的峰值不超过限值，PI 的目标温度只能设在 76°C，于是平时一直多花风扇的电。

> **小结**　反馈控制有三个局限：看不到即将到来的变化；不知道限值在哪里，只能靠留余量间接处理；不会权衡多开一点风扇是否值得。要弥补这三点，控制器需要两样它目前没有的东西：一个能推算未来温度的模型，以及一个在不同方案之间取舍的标准。MPC

> 就是把这两样东西组合在一起。

第二部分

构造 MPC

MPC 每个采样时刻都在做同一件事：预测、评价、选择、执行一步。本部分依次搭起它需要的三样东西：把微分方程变成控制器可以使用的一步公式，把一步公式串起来做预测，把控制目标写成一个可以比较大小的代价函数。

4　MPC 的工作方式

MPC 每隔 t_s 秒执行一次以下步骤：

1. 测量当前温度 T_0。
2. 设未来 N 步的风扇方案为 $U = (u_0, u_1, \dots, u_{N-1})$，用模型预测相应的未来温度 $T_1, T_2, \dots, T_N$。
3. 用代价函数 $J(U)$ 评价这个方案：是否费电、是否平稳、是否超温。
4. 求出使 J 最小的方案 U^*。
5. 只执行第一个值 u_0^*。t_s 秒后回到第 1 步，重新计算。

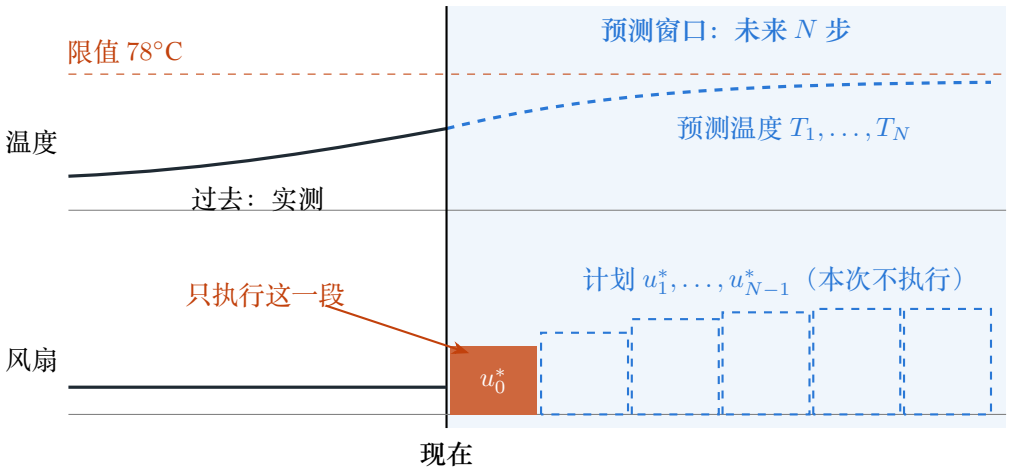


图 6　滚动时域。每次规划未来 N 步，只执行第一步；t_s 秒后窗口整体右移一格，用新的测量值重新规划。

要让这套流程运转起来，需要依次解决三个问题：

- **如何预测**：给定方案 U，算出 $T_1, \dots, T_N$（第 5 节、第 6 节）；

- **如何评价**：写出代价函数 $J(U)$（第 7 节）；
- **如何求最小值**：而且要在每个采样周期内可靠地完成（第三部分）。

5　离散化：从微分方程到一步公式

控制器是一段程序，每隔 t_s 秒运行一次，两次运行之间风扇保持不变。它需要的不是一条连续曲线 $T(t)$，而是一个递推关系：**已知当前温度 T_k，风扇保持 u_k 不变 t_s 秒，求 t_s 秒后的温度 T_{k+1}**。这里 t_s 称为采样周期，第 k 个采样时刻为 $t_k = k\,t_s$。

最直接的做法是把方程 (4) 中的导数近似为差商：

$$\frac{dT}{dt} \approx \frac{T_{k+1} - T_k}{t_s}. \tag{6}$$

代入并整理，得到一步公式：

$$\boxed{T_{k+1} = T_k + \frac{t_s}{C}\Big[P_k - G(u_k)\,\big(T_k - T_{\text{in}}\big)\Big] \;\equiv\; f(T_k, u_k, P_k)} \tag{7}$$

这就是欧拉法。它假设在一个采样周期内，温度始终按周期起点的斜率变化。只要 t_s 远小于系统的时间常数，这一假设就基本成立（附录 H）。本例最小的时间常数约为 70 s，实际运行取 $t_s = 5$ s。为了让算例的数字少一些，第三部分的手算算例取 $t_s = 30$ s，此时欧拉法每步的误差约为 0.6 K，不影响论证。

6　预测：把一步公式串起来

MPC 要评价未来一段时间的表现，因此需要未来若干时刻的温度。从当前测得的 T_0 出发，反复使用一步公式：

$$T_1 = f(T_0, u_0, P_0), \qquad T_2 = f(T_1, u_1, P_1), \qquad T_3 = f(T_2, u_2, P_2), \qquad \dots \tag{8}$$

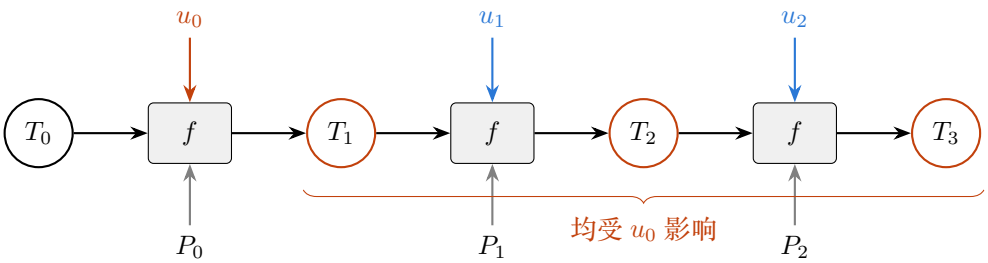


图 7　预测就是把同一个一步公式 f 串联起来。第一步的风量 u_0 影响之后所有温度；最后一步的 u_2 只影响 T_3。未来负载 P_k 也在这里进入预测。

把 T_1 代入 T_2 的表达式，得到 $T_2 = f\big(f(T_0, u_0, P_0), u_1, P_1\big)$，它同时依赖 u_0 和 u_1。一般地，T_k 依赖它之前的所有风量 $u_0, \dots, u_{k-1}$，也依赖之前的所有负载 $P_0, \dots, P_{k-1}$。所以每个预测温度都是风扇方案 U 的函数，记作 $T_k(U)$，而且是层层嵌套的复合函数。

预测要用到未来的负载 $P_0, P_1, \dots$。这些值从哪里来，是第 15 节专门讨论的问题。在那之前，先把它们当作已知。

7　代价函数：把目标写成一个数

代价函数 $J(U)$ 用一个数衡量方案 U 的好坏，MPC 寻找使它最小的方案。设计的方法是：把第 1 节的三个目标逐条翻译成数学表达式（表 2）。

表 2　从控制目标到代价函数

目标	数学表达	理由
省电	$\sum_k u_k^3$	风扇功耗与 u^3 成正比，求和就是这段时间的总能耗。
平稳	$\sum_k (u_k - u_{k-1})^2$	平方使大跳变的代价远高于小调整：跳变 0.2 的代价是跳变 0.1 的 4 倍。u_{-1} 是上一次实际执行的值。
安全	约束 $T_k(U) \le 78$	温度是 70°C 还是 75°C 并不重要，只要不超过 78°C，所以写成不可越过的界限。

合并起来，实际运行中每个采样时刻要求解的问题是

$$\min_{U} \; J(U) = \sum_{k=0}^{N-1} \Big[w_f \, u_k^3 + w_d \, (u_k - u_{k-1})^2 \Big] \quad \text{满足} \quad T_k(U) \le 78, \quad 0.2 \le u_k \le 1. \tag{9}$$

权重 w_f 与 w_d 是两项之间的汇率，只有相对大小有意义。求和覆盖整个预测窗口，因为 u_0 会影响之后的所有温度（图 7）；只看下一步，就发现不了“现在多花一点、后面省很多”的机会。

问题 (9) 带有约束，手算繁琐。为了把“如何求最小值”讲透，第三部分统一使用下面这个更简单的算例。它保留了问题的关键特征：预测温度依赖风扇方案，代价在温度与风扇用力之间权衡。

算例　负载保持 100 W，当前温度 $T_0 = 72$°C，采样周期 $t_s = 30$ s，预测两步（$N = 2$）。希望温度回到 70°C，同时不让风扇用力过大。代价函数取

$$J(u_0, u_1) = \underbrace{(T_1 - 70)^2 + (T_2 - 70)^2}_{\text{温度偏离目标}} + \underbrace{20\,(u_0^2 + u_1^2)}_{\text{风扇用力}}, \tag{10}$$

其中 T_1、T_2 由一步公式 (7) 算出。为使代数简洁，这里用 u^2 代表风扇用力，系数 20 是两部分之间的汇率。

第三部分

求解：为什么要线性化

有了预测和代价函数，MPC 剩下的任务是求最小值。由微积分，最小值点处导数为零。本部分用同一个算例，先在原始的非线性模型下求解，看清困难在哪里；再把模型线性化，看困难怎样消失，以及付出了什么代价。

8　求最小值：令导数为零

可导函数在最小值点处导数为零；多元函数在最小值点处，对每个变量的偏导数都为零。所以 MPC 中的“求最小值”，就是解方程

$$\frac{\partial J}{\partial u_0}=0,\qquad \frac{\partial J}{\partial u_1}=0,\qquad \dots \tag{11}$$

下面先看一个变量的情形，再推广到两个变量，最后讨论导数为零的点是否一定是最低点。

8.1　一个变量

只看第一段（$N=1$），代价函数为 $J(u)=\big(T_1(u)-70\big)^2+20u^2$。

第一步：把 T_1 写成 u 的函数。 在一步公式 (7) 中代入 $T_0=72$、$P=100$、$t_s=30$、$C=400$、$T_{\text{in}}=25$ 和 $G(u)=3.5(2u)^{0.7}$：

$$T_1(u)=72+\frac{30}{400}\Big[100-3.5\,(2u)^{0.7}\,(72-25)\Big]=79.5-12.34\,(2u)^{0.7}. \tag{12}$$

第二步：求 J 的导数。 由链式法则（附录 B），

$$J'(u)=2\big(T_1(u)-70\big)\,T_1'(u)+40u,\qquad T_1'(u)=-17.27\,(2u)^{-0.3}. \tag{13}$$

第三步：令导数为零。 代入 T_1 与 T_1'，方程 $J'(u)=0$ 成为

$$2\Big(9.5-12.34\,(2u)^{0.7}\Big)\Big(-17.27\,(2u)^{-0.3}\Big)+40u=0. \tag{14}$$

u 同时出现在 $(2u)^{0.7}$、$(2u)^{-0.3}$ 和 $40u$ 中，无法通过代数变形单独解出，只能用数值方法逐步逼近方程的根。

第四步：用牛顿法解 $J'(u)=0$。 牛顿法是求方程根的方法。设当前猜测为 u，在 u 处用切线近似函数 J'：

$$J'(u+d)\approx J'(u)+J''(u)\,d. \tag{15}$$

令右边为零，得到步长 $d = -J'(u)/J''(u)$，新的猜测为

$$u \leftarrow u - \frac{J'(u)}{J''(u)}. \tag{16}$$

几何上，这就是从当前点作 J' 曲线的切线，取切线与横轴的交点作为下一个猜测（图 8b）。公式需要二阶导数，对式 (13) 再求一次导数（乘积法则）：

$$J''(u) = 2\Big[T_1'(u)^2 + \big(T_1(u) - 70\big)\,T_1''(u)\Big] + 40, \qquad T_1''(u) = 10.36\,(2u)^{-1.3}. \tag{17}$$

第五步：计算。 从 $u = 0.9$ 出发：$T_1 = 60.88$，$T_1' = -14.48$，$T_1'' = 4.83$，代入得 $J'(0.9) = 300.0$，$J''(0.9) = 371.3$，所以下一个猜测为 $0.9 - 300.0/371.3 = 0.092$。继续迭代，结果见表 3。

表 3　用牛顿法解 $J'(u) = 0$，从 $u = 0.9$ 出发

迭代次数	u	$J'(u)$	$J''(u)$	下一个 u
0	0.900	300.0	371.3	0.092
1	0.092	−325.1	2759.9	0.210
2	0.210	−116.3	1223.1	0.305
3	0.305	−18.8	873.3	0.326
4	0.326	−0.6	823.2	0.327
5	0.327	0.0	821.7	收敛

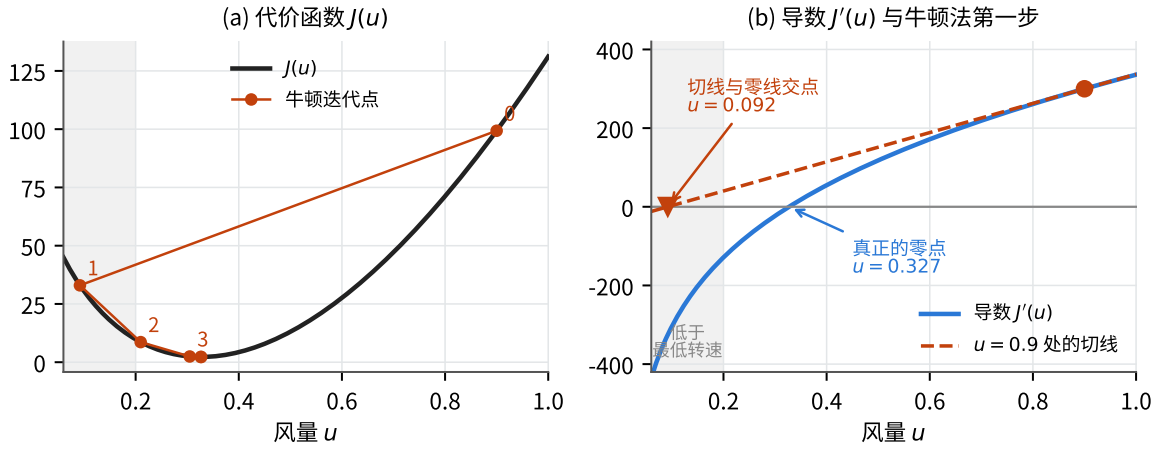


图 8　一元牛顿法。(a) 代价函数 $J(u)$ 及各次迭代点。(b) 导数 $J'(u)$；虚线是它在 $u = 0.9$ 处的切线，与零线交于 $u = 0.092$，而 J' 真正的零点在 $u = 0.327$。

图 8b 说明了第一步为什么跳得太远。$J'(u)$ 是一条弯曲的曲线，在 $u = 0.9$ 处比较平缓，切线斜率偏小，与零线的交点便落得很远，到了 $u = 0.092$，低于风扇的最低转速。此后又走了四步才收敛到 $u = 0.327$。

牛顿法也可以从另一个角度理解：用二阶泰勒展开把 J 在当前点近似成一条抛物线，然后跳到抛物线的最低点。两种理解给出同一个公式 (16)（附录 E）。

8.2 两个变量

回到算例 (10)，未知数是 u_0 和 u_1。第一段末的温度仍是式 (12)，记作 $T_1(u_0)$；第二段末的温度由一步公式再算一次：

$$T_2 = T_1 + 0.075\Big[100 - 3.5\,(2u_1)^{0.7}\,\big(T_1 - 25\big)\Big]. \tag{18}$$

T_2 通过 T_1 依赖 u_0，同时直接依赖 u_1。

求偏导数。 由链式法则，

$$\frac{\partial J}{\partial u_0} = 2\big(T_1 - 70\big)\frac{\partial T_1}{\partial u_0} + 2\big(T_2 - 70\big)\frac{\partial T_2}{\partial u_0} + 40u_0, \tag{19}$$

$$\frac{\partial J}{\partial u_1} = 2\big(T_2 - 70\big)\frac{\partial T_2}{\partial u_1} + 40u_1, \tag{20}$$

其中各温度的偏导数为

$$\frac{\partial T_1}{\partial u_0} = -17.27\,(2u_0)^{-0.3}, \qquad \frac{\partial T_2}{\partial u_0} = \Big[1 - 0.2625\,(2u_1)^{0.7}\Big]\frac{\partial T_1}{\partial u_0},$$
$$\frac{\partial T_2}{\partial u_1} = -0.3675\,(2u_1)^{-0.3}\big(T_1 - 25\big). \tag{21}$$

中间一式的方括号是 $\partial T_2/\partial T_1$：$u_0$ 先改变 T_1，T_1 再改变 T_2。

令式 (19) 和 (20) 都等于零，得到两个方程。它们同样含有 $(2u_0)^{0.7}$、$(2u_1)^{-0.3}$ 以及它们的乘积，无法直接解出，仍然只能迭代。

二元牛顿法。 做法与一元相同：在当前点用切平面分别近似两个偏导数，令近似值为零。记 $J_0 = \partial J/\partial u_0$，$J_1 = \partial J/\partial u_1$，$J_{00} = \partial^2 J/\partial u_0^2$，$J_{01} = \partial^2 J/\partial u_0 \partial u_1$，$J_{11} = \partial^2 J/\partial u_1^2$，则步长 (d_0, d_1) 满足

$$\begin{aligned} J_0 + J_{00}\,d_0 + J_{01}\,d_1 &= 0, \\ J_1 + J_{01}\,d_0 + J_{11}\,d_1 &= 0. \end{aligned} \tag{22}$$

这是关于 d_0, d_1 的二元一次方程组。解出步长，更新 (u_0, u_1)，再重复。该算例的真正最小值点为 $(0.337,\, 0.255)$，从不同起点出发的结果见表 4。

表 4　二元牛顿法从不同起点出发的结果

起点 (u_0, u_1)	迭代次数	结果
$(0.3,\ 0.3)$	2	收敛到 $(0.337,\ 0.255)$
$(0.5,\ 0.5)$	5	收敛到 $(0.337,\ 0.255)$
$(0.2,\ 0.8)$	5	收敛到 $(0.337,\ 0.255)$
$(0.9,\ 0.9)$	1	第一步跳到 $(-0.20,\ -1.06)$，离开允许范围

实际的求解器会限制步长来防止越界（附录 E），但迭代次数仍取决于起点和函数形状，无法事先确定。对一个必须在每个采样周期内给出结果的控制器，这是实实在在的问题。

8.3 导数为零的点一定是最低点吗

导数为零只是最小值的必要条件。一元时，$J'(u)=0$ 的点也可能是最高点，判断要看二阶导数：$J''>0$ 表示函数在该点向上弯曲。如果 J'' 处处大于零，函数处处向上弯，形状像一个碗，导数为零的点就是唯一的最低点。这样的函数称为**凸函数**。

两个变量时，$J(u_0,u_1)$ 是平面上方的一张曲面：平面上每一点 (u_0,u_1) 代表一个风扇方案，曲面的高度就是它的代价（图 9a）。曲面可能沿一个方向向上弯、沿另一个方向向下弯（如马鞍），所以要判断它是否向上弯，必须考察每一个方向。

具体做法如下。在平面上取一点 $U=(u_0,u_1)$，再取一个方向 $\boldsymbol{v}=(v_0,v_1)$，也就是平面上的一个箭头：沿它走单位距离，u_0 增加 v_0，u_1 增加 v_1。从 U 沿 $\boldsymbol{v}$ 走距离 s，到达点 $(u_0+s\,v_0,\ u_1+s\,v_1)$。这条直线正上方的曲面被切出一条曲线

$$g(s)=J\big(u_0+s\,v_0,\ u_1+s\,v_1\big), \tag{23}$$

它是只有一个变量 s 的普通函数（图 9b）。对 s 求导，由链式法则：

$$\begin{aligned} g'(s) &= J_0\,v_0+J_1\,v_1,\\ g''(s) &= J_{00}\,v_0^2+2J_{01}\,v_0v_1+J_{11}\,v_1^2. \end{aligned} \tag{24}$$

g'' 就是曲面沿方向 $\boldsymbol{v}$ 的弯曲程度。若对每个点、每个方向都有 $g''\ge 0$，则 J 是凸函数：沿任何直线切下去都是向上弯的曲线，整张曲面是一个碗。凸函数有一个关键性质：**所有偏导数同时为零的点，一定是全局最低点**，不存在假的谷底（证明见附录 D）。

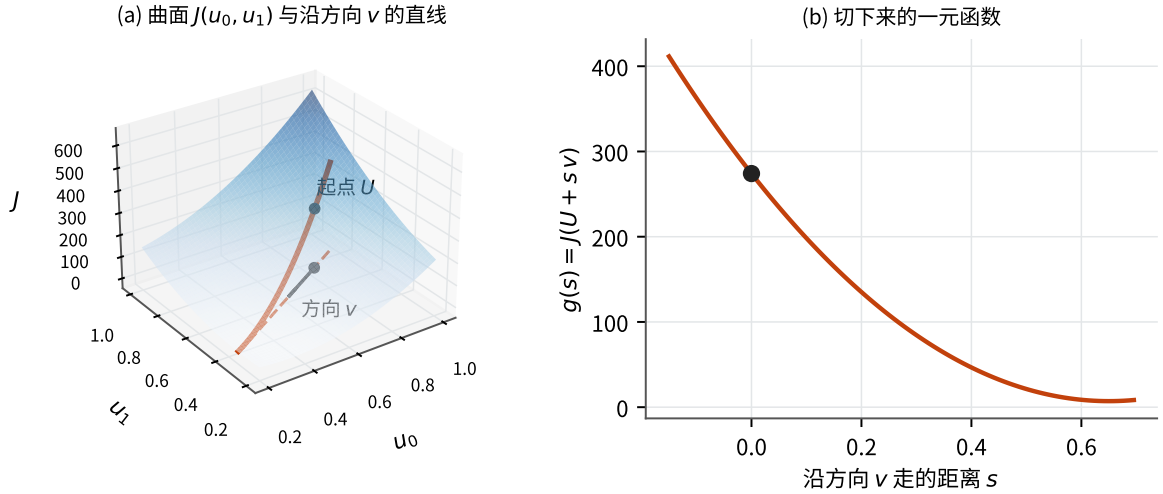


图 9　沿一个方向考察曲面的弯曲。(a) 代价曲面 $J(u_0,u_1)$；底面虚线是从起点 U 出发、沿方向 $\boldsymbol{v}$ 的直线，曲面上的实线是这条直线正上方的截线。(b) 截线画成一元函数 $g(s)$，它的二阶导数就是曲面沿 $\boldsymbol{v}$ 的弯曲程度。

非线性模型下，凸性有没有保证？　看代价中的一项 $(T_k-70)^2$。方案沿上述直线移动时，预测温度 T_k 随之变化，记为 $T_k(s)$。注意 s 是方案沿直线移动的距离，不是时间，$T_k(s)$ 也不是

新的物理方程，只描述方案变动时预测温度怎样跟着变。对 s 求两次导数（链式法则与乘积法则）：

$$\frac{d}{ds}\big(T_k-70\big)^2=2\big(T_k-70\big)T_k',\qquad \frac{d^2}{ds^2}\big(T_k-70\big)^2=2\Big[T_k'^2+\big(T_k-70\big)\,T_k''\Big]. \tag{25}$$

第一项 $T_k'^2$ 总是非负的，第二项的符号却不确定。由于 $G(u)\propto u^{0.7}$ 回报递减，预测温度随风量的变化是一条曲线，$T_k''\neq 0$；当温度低于目标（$T_k<70$）时，$(T_k-70)T_k''$ 可能为负。一旦它压过其余各项，代价函数在这个方向上就向下弯。本算例在允许的风量范围内恰好仍是凸的，但这只是这组参数和工况下的结果。换一个工况、多一个温度节点、或改用更复杂的模型，都必须重新检验。

> **小结**　用非线性模型做预测，求最小值有两个困难。其一，令导数为零得到的是非线性方程，只能迭代求解，迭代次数取决于起点，还可能跳出允许范围。其二，代价函数是否为凸函数没有一般性的保证，导数为零的点未必是最低点。两个困难有同一个根源：**预测温度是风量的弯曲函数**，归根结底是一步公式 f 对风量 u 是弯曲的。

9　线性化：在当前工况附近取切平面

既然困难来自一步公式 f 的弯曲，就在当前工况附近用一个平直的函数代替它。工具是一阶泰勒展开。

9.1　展开点怎么选

泰勒展开总是围绕某一点进行，离这一点越近，近似越准确。展开点记为 $(\bar T,\bar u,\bar P)$，每个采样时刻按当前工况选取：

- $\bar T$：当前测得的温度，即 T_0；
- $\bar u$：上一个周期正在执行的风量；
- $\bar P$：当前负载。

新的风扇方案通常与当前风量相差不大，预测温度也在当前温度附近，这样选择能让近似在最需要准确的地方最准确。到下一个采样时刻，工况变了，展开点随之更新。算例中 $\bar T=72^{\circ}\mathrm{C}$，$\bar P=100\ \mathrm{W}$，并设上一周期风量 $\bar u=0.5$。

9.2　一步公式的切平面

一元函数在 $\bar x$ 处的一阶泰勒展开是切线 $f(x)\approx f(\bar x)+f'(\bar x)\,(x-\bar x)$。一步公式 $f(T,u,P)$ 有三个输入，对应的一阶展开是每个输入各贡献一项“偏导数乘以偏离量”（附录 B）：

$$\boxed{T_{k+1}\approx c+a\,(T_k-\bar T)+b\,(u_k-\bar u)+e\,(P_k-\bar P)} \tag{26}$$

其中 $c = f(\bar T, \bar u, \bar P)$ 是展开点处的函数值，三个系数是 f 在展开点处的偏导数，由式 (7) 直接求得：

$$a = \frac{\partial f}{\partial T} = 1 - \frac{t_s\,G(\bar u)}{C}, \qquad b = \frac{\partial f}{\partial u} = -\frac{t_s}{C}\,G'(\bar u)\,(\bar T - T_{\text{in}}), \qquad e = \frac{\partial f}{\partial P} = \frac{t_s}{C}, \tag{27}$$

这里 $G'(u) = 0.7\,G(u)/u$。三个系数各有物理含义：a 是风扇不变时温度偏差经过一个周期后保留的比例；b 是风量增加 1 个单位，一个周期后温度的变化量，为负值；e 是功耗增加 1 W，一个周期后温度的升高量。

算例中的数值为

$$c = 67.16^{\circ}\text{C}, \qquad a = 0.7375, \qquad b = -17.27\ \text{K/单位风量}, \qquad e = 0.075\ \text{K/W}. \tag{28}$$

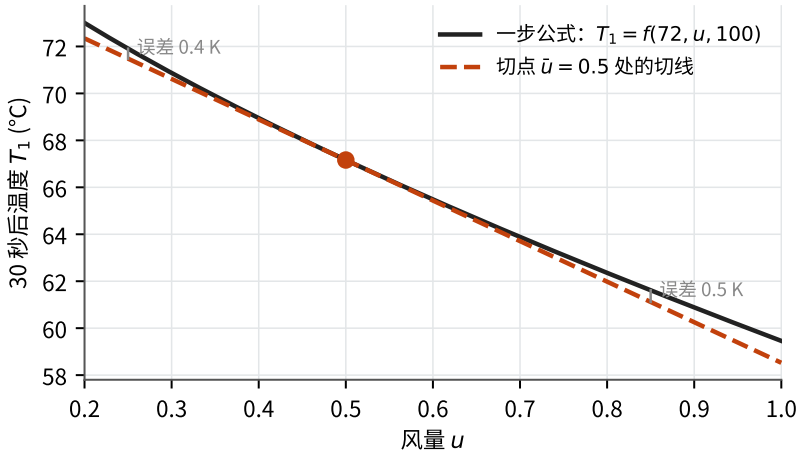


图 10　一步公式（实线）及其在 $\bar u = 0.5$ 处的切线（虚线）。起点 72°C，负载 100 W，$t_s = 30$ s。

图 10 比较了一步公式与它的切线：展开点附近两者几乎重合，离展开点越远误差越大。

10　线性预测：温度成为风量的一次函数

现在重复第 6 节的预测过程，只是每一步改用切平面 (26)。算例中负载不变，$P_k = \bar P$，含 e 的项为零。

第一段。 由于 $T_0 = \bar T$，a 项也为零：

$$T_1 = c + b\,(u_0 - \bar u) = 67.16 - 17.27\,(u_0 - 0.5) = 75.80 - 17.27\,u_0. \tag{29}$$

第二段。 把 T_1 代入切平面：

$$\begin{aligned} T_2 &= c + a\,(T_1 - \bar T) + b\,(u_1 - \bar u) \\ &= 67.16 + 0.7375\,(75.80 - 17.27\,u_0 - 72) - 17.27\,(u_1 - 0.5) \\ &= 78.60 - 12.74\,u_0 - 17.27\,u_1. \end{aligned} \tag{30}$$

第三段。若预测三步，同样得到

$$T_3 = c + a\,(T_2 - \bar{T}) + b\,(u_2 - \bar{u}) = 80.67 - 9.39\,u_0 - 12.74\,u_1 - 17.27\,u_2. \tag{31}$$

与非线性预测 (12)、(18) 对比，本质区别一目了然。非线性预测中，风量以 $(2u)^{0.7}$ 的形式出现，还与温度相乘；线性预测中，**每个预测温度都是“常数加上各段风量的倍数”**，即风扇方案的一次函数（图 12a）。

风量前的系数有清晰的规律。对 T_k 而言，紧挨着的前一段风量系数为 $b = -17.27$，再往前一段为 $ab = -12.74$，再往前为 $a^2 b = -9.39$。某一段风扇多开一点，效果会延续到之后的每一个温度，每经过一段打一次折扣 a。这正是第 2 节所说的热容缓冲：金属记住了过去的冷却。

矩阵写法。N 较大时逐行书写很繁琐，可以用矩阵简写（附录 C）。三步预测写成

$$\begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix} = \underbrace{\begin{bmatrix} 75.80 \\ 78.60 \\ 80.67 \end{bmatrix}}_{\boldsymbol{F}} + \underbrace{\begin{bmatrix} -17.27 & 0 & 0 \\ -12.74 & -17.27 & 0 \\ -9.39 & -12.74 & -17.27 \end{bmatrix}}_{\Phi} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \end{bmatrix}. \tag{32}$$

矩阵乘以向量的规则是：矩阵每一行与向量对应相乘再求和。例如第二行给出 $-12.74\,u_0 - 17.27\,u_1 + 0 \cdot u_2$，加上 78.60 正好是 T_2。所以式 (32) 只是上面三行的紧凑写法，没有新的内容。Φ 的右上角全为零，因为后面的风扇改变不了前面的温度；每一列都是同一个衰减序列向下平移（图 11）。

若负载有变化，切平面中 $e\,(P_k - \bar{P})$ 不再为零，预测式中会多出一项 $\Psi \boldsymbol{D}$，$\boldsymbol{D}$ 由负载预报 $P_k - \bar{P}$ 组成。这一项在第 13 节中起关键作用。

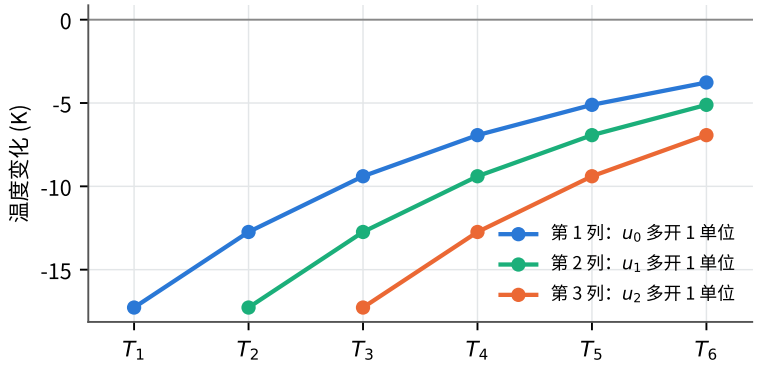


图 11　Φ 的前三列（$N = 6$）。每一列都是同一条衰减曲线向后平移。

> 控制理论中常见的离散状态方程 $x_{k+1} = A\,x_k + B\,u_k + E\,d_k$，在只有一个温度时，$A$、$B$、$E$ 就是这里的 a、b、e。若有多个温度节点（例如芯片与散热器分开建模），它们成为矩阵，推导步骤不变。

11　线性化之后：一组一次方程

11.1　代入代价函数，令偏导数为零

把线性预测代入算例的代价函数 (10)。因为 $T_1 - 70 = 5.80 - 17.27\,u_0$，$T_2 - 70 = 8.60 - 12.74\,u_0 - 17.27\,u_1$，所以

$$J = \big(5.80 - 17.27\,u_0\big)^2 + \big(8.60 - 12.74\,u_0 - 17.27\,u_1\big)^2 + 20\big(u_0^2 + u_1^2\big), \tag{33}$$

这是 u_0、u_1 的二次多项式。与第 8.2 节完全相同，对 u_0 和 u_1 分别求偏导数并令其为零：

$$\frac{\partial J}{\partial u_0} = 2\big(5.80 - 17.27u_0\big)(-17.27) + 2\big(8.60 - 12.74u_0 - 17.27u_1\big)(-12.74) + 40u_0 = 0,$$
$$\frac{\partial J}{\partial u_1} = 2\big(8.60 - 12.74u_0 - 17.27u_1\big)(-17.27) + 40u_1 = 0.$$

展开整理，得到两个一次方程：

$$\begin{aligned} 961.2\,u_0 + 440.0\,u_1 &= 419.4, \\ 440.0\,u_0 + 636.7\,u_1 &= 297.1. \end{aligned} \tag{34}$$

这是普通的二元一次方程组，消元即可解出（附录 C）：

$$u_0 = 0.326, \qquad u_1 = 0.241. \tag{35}$$

这就是线性化最直接的好处。非线性模型下，令偏导数为零得到含有幂函数和乘积的方程，只能迭代；线性模型下，得到一次方程组，一步解出，计算量固定。图 12b 用一元情形把这一点画了出来：非线性模型的导数 $J'(u)$ 是一条曲线，只能逐步逼近它的零点；线性模型的导数是一条直线，零点可以直接写出：

$$J'(u) = -34.54\,(5.80 - 17.27\,u) + 40u = 0 \quad\Longrightarrow\quad u = \frac{17.27 \times 5.80}{17.27^2 + 20} = 0.315.$$

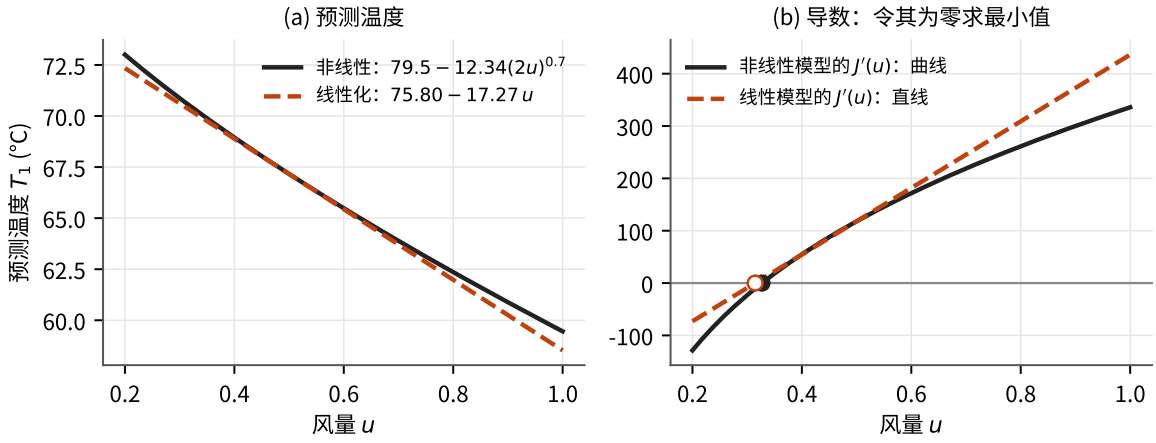


图 12　非线性模型与线性化模型（一元情形）。(a) 预测温度：曲线与它的切线。(b) 代价函数的导数：非线性时是曲线，线性时是直线，令其为零即可直接求出最小值点（圆点）。

换成牛顿法的语言也一样：线性模型下 J' 本身就是直线，它的切线就是它自己，牛顿法从任何起点出发，第一步就落在零点上。表 4 中从 $(0.9, 0.9)$ 出发跳出范围的情况不会再发生。

11.2　凸性得到保证

沿任意方向 $\boldsymbol{v} = (v_0, v_1)$ 考察式 (33) 的弯曲程度。由式 (24)，

$$g'' = 2\,(17.27\,v_0)^2 + 2\,(12.74\,v_0 + 17.27\,v_1)^2 + 40\,\left(v_0^2 + v_1^2\right). \tag{36}$$

这是平方和，只要方向不为零就严格大于零，所以 J 一定是凸函数，方程组 (34) 的解一定是唯一的全局最低点。

对照式 (25) 就能看出原因：那里可能为负的项是 $(T_k - 70)\,T_k''$；现在预测温度是风扇方案的一次函数，沿任何直线都按直线变化，$T_k'' = 0$，这一项不复存在。这个结论与 a、b 的具体数值无关，所以在任何工况下都成立。

省电型代价 (9) 的各项同样是凸的：u^3 在 $u > 0$ 时二阶导数为 $6u > 0$；$(u_k - u_{k-1})^2$ 的二阶导数是常数。约束 $T_k \le 78$ 在线性模型下是风扇方案的一次不等式，在 (u_0, u_1) 平面上是一条直线的一侧（附录 D）。这种“凸代价加直线约束”的问题有成熟算法，能在毫秒量级、在有上界的迭代次数内可靠求解（附录 G）。

11.3　两种模型的比较与线性化的代价

表 5　非线性模型与线性化模型下求最小值的比较

	非线性模型	线性化模型
预测温度对风量	曲线：含 $(2u)^{0.7}$ 及其与温度的乘积	直线：常数加各段风量的倍数
令导数为零得到	非线性方程，无法直接解出	一次方程组，直接解出
求解过程	迭代，次数取决于起点，可能越界	一步完成，计算量固定
是否为凸函数	需逐个工况检验	任何工况下都成立
温度约束的形状	曲线边界	直线边界
代价	预测精确	有线性化误差

线性化并非没有代价。算例中线性模型给出 $(0.326,\ 0.241)$，非线性模型的真正最小值点是 $(0.337,\ 0.255)$，接近但不相同。把线性解代回非线性模型，两段末温度为 70.36 和 70.71°C，而线性模型预测的是 70.17 和 70.28°C。这个误差留给下一部分的滚动执行去修正。

> **小结**　线性化把“解一个只能迭代的非线性方程”变成了“解一组一次方程”，并且保证解出的就是唯一的最低点。换来的是展开点附近的一点误差，而这点误差会在每个采样时

> 刻重新测量、重新展开时被纠正。这就是实际使用的 MPC 几乎都从线性 MPC 开始的原因。

第四部分

MPC 在运行中

前三部分讲的是 MPC 在一个时刻做什么。本部分把时间轴拉开：MPC 怎样一个周期接一个周期地运行，为什么会在负载到来之前就加速风扇，实际能省多少电。然后回头审视两个前提：负载预报从哪里来，以及预测 N 步依赖系统的什么性质。

12　滚动执行：每个周期只动一次风扇

MPC 在每个采样时刻都求出未来 N 步的完整方案，但**只把第一个值 u_0 发给风扇**。风扇在接下来的 t_s 秒内保持这个值，到下一个采样时刻再重新计算。因此风扇每 t_s 秒最多改变一次，其余 $N-1$ 个值从不直接执行（表 6）。

表 6　滚动执行的时间线（$t_s = 5$ s，$N = 24$）

时刻	控制器做什么	风扇做什么
0 s	测温度，取展开点，求出方案 A：$u_0^A, \dots, u_{23}^A$，覆盖 0 ∼ 120 s	0 ∼ 5 s 保持 u_0^A
5 s	重新测温度，更新展开点和负载信息，求出方案 B，覆盖 5 ∼ 125 s	5 ∼ 10 s 保持 u_0^B
10 s	同上，求出方案 C	10 ∼ 15 s 保持 u_0^C
⋮	⋮	⋮

后面的值既然不执行，为什么还要算？因为当前这一步的后果要过一段时间才显现。温度的时间常数是一到两分钟，只看 5 秒后的温度，无法判断“现在开小一点”会不会在一分钟后造成超温。后面的 $N-1$ 个值，是控制器用来评估当前决定的长远后果的一份计划，随时会被下一次计算改写。

只执行第一步，是因为模型有误差（例如上一节的线性化误差），负载信息也会更新。每次用新的测量值重新计算，误差就不会累积。测量值 T_0 决定了每次预测的起点，这是 MPC 中的**反馈**；负载信息 P_k 决定了预测途中会遇到什么，这是**前馈**。

13　提前动作：负载到来之前加速风扇

13.1　仿真中的现象

图 13 是仿真中 MPC 在几个时刻求得的方案（$t_s = 5$ s，$N = 24$，预测窗口 120 s）。负载在 300 s 时从 100 W 升到 250 W，控制器事先知道这一点。

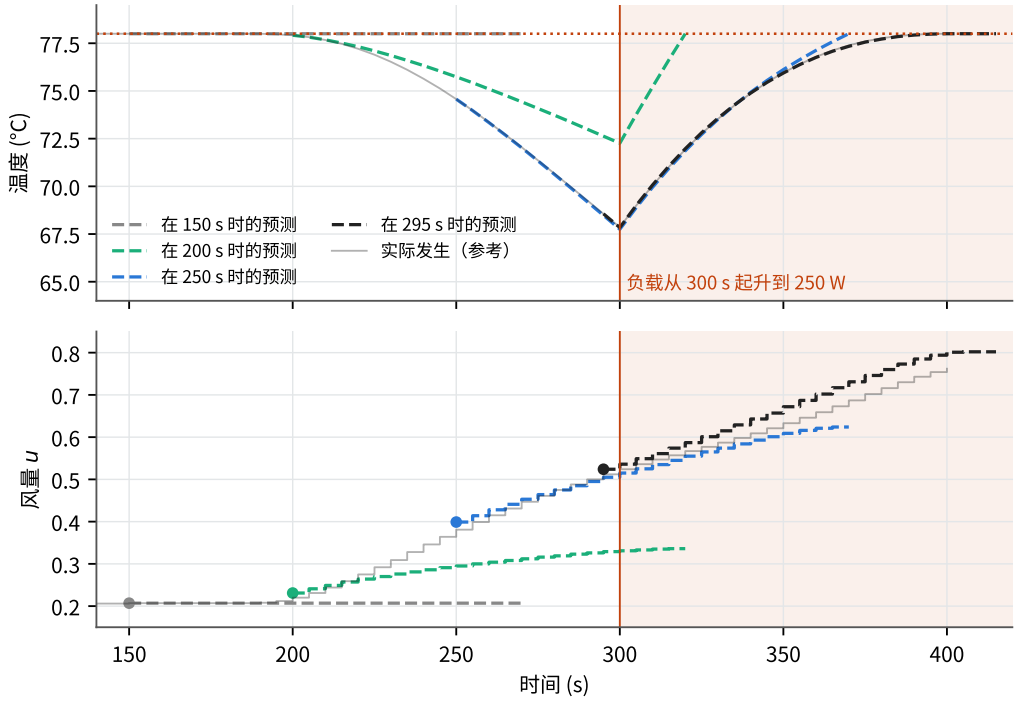


图 13　MPC 在 150、200、250、295 s 时求得的预测温度（上）与风扇方案（下）。圆点是该时刻实际执行的值，虚线是尚未执行的计划。

150 s 时，负载跳变还在 120 s 的窗口之外，方案一直是最低够用的 0.21。负载一进入窗口（180 s 以后），方案就开始爬升：200 s 时从 0.23 爬到 0.34，250 s 时从 0.40 爬到 0.62，295 s 时从 0.52 爬到 0.80。负载还没有来，风扇已经在加速。下面用一个可以手算的例子说明其中的道理。

13.2　一个两步的例子

情境。 当前温度 77°C，风量 0.21，负载 100 W，并且已知 30 s 后负载升到 250 W。要求 60 s 时温度 $T_2 \le 78$°C。取 $t_s = 30$ s，在当前工况 $(\bar{T}, \bar{u}, \bar{P}) = (77,\ 0.21,\ 100)$ 处线性化，由式 (27) 得

$$a = 0.857, \qquad b = -24.8, \qquad e = 0.075.$$

两段的风量记为 $u_0 = 0.21 + v_0$、$u_1 = 0.21 + v_1$，$v_0, v_1 \ge 0$ 是在当前风量之上的额外风量。

第一步：风扇不变时的预测。 令 $v_0 = v_1 = 0$，按切平面 (26) 预测：

$$T_1 = c = 77.06,$$
$$T_2 = c + a\,(T_1 - 77) + e\,(250 - 100) = 77.06 + 0.05 + 11.25 = 88.37.$$

第二段的负载变化通过 $e\,(250-100) = 11.25$ 进入预测，正是这一项让 T_2 超出限值 10.37 K。如果不知道负载要来，控制器会假设第二段仍是 100 W，预测出 $T_2 = 77.1°\mathrm{C}$，看不到任何问题。

第二步：额外风量对 T_2 的作用。 由第 10 节的规律，T_2 中 u_1 的系数是 b，u_0 的系数是 ab：

$$T_2 = 88.37 - 21.2\,v_0 - 24.8\,v_1. \tag{37}$$

关键在于 v_0 的系数不为零：第一段多开的风扇，到第二段末仍然有效，只打了一个折扣 $a = 0.857$，因为金属在第一段被冷却后，到第二段仍记得这份冷却。要满足 $T_2 \le 78$，需要

$$21.2\,v_0 + 24.8\,v_1 \;\ge\; 10.37. \tag{38}$$

第三步：比较两种方案。 用 $v_0^2 + v_1^2$ 衡量额外的风扇用力（凸函数；换成 u^3 结论相同）。

方案一：不提前。第一段不动，$v_0 = 0$，全部由第二段承担：$v_1 = 10.37/24.8 = 0.418$，风量为 $(0.21,\ 0.63)$，代价 $0.418^2 = 0.175$。

方案二：提前分摊。在约束 (38) 取等号的条件下使 $v_0^2 + v_1^2$ 最小。由约束解出 $v_1 = (10.37 - 21.2\,v_0)/24.8$，代入代价得到一元函数

$$h(v_0) = v_0^2 + \Big(\frac{10.37 - 21.2\,v_0}{24.8}\Big)^2.$$

令 $h'(v_0) = 2v_0 - 2\cdot\dfrac{21.2}{24.8}\cdot\dfrac{10.37 - 21.2\,v_0}{24.8} = 0$，解得 $v_0 = 0.207$，$v_1 = 0.241$，风量为 $(0.42,\ 0.45)$，代价 $0.207^2 + 0.241^2 = 0.101$，比方案一少 42%。

用符号推导可以得到一般结论：最优分摊满足 $v_0 = a\,v_1$，并且

$$\text{方案二的代价} = \frac{1}{1 + a^2} \times \text{方案一的代价}. \tag{39}$$

a 越接近 1，即热容越大、冷却保留得越久，提前分摊越划算。

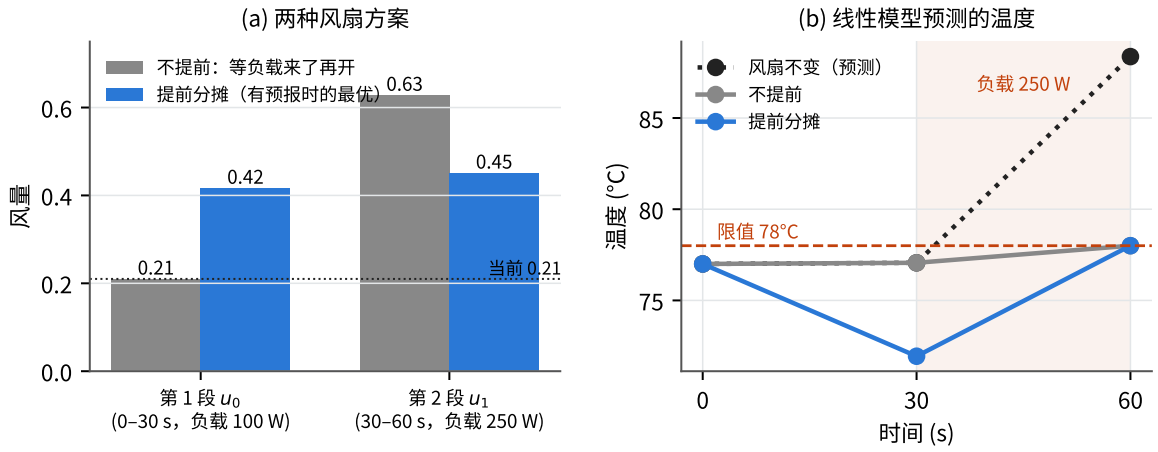


图 14 两步例子。(a) 两种风扇方案。(b) 线性模型预测的温度：风扇不变时 60 s 温度将达 88.4°C；两种方案都恰好把 60 s 温度压到 78°C，提前分摊的方案先把温度降下去，用更小的代价达到同样效果。

13.3 提前动作的三个条件

这个例子说明，提前加速风扇需要三个条件同时成立：

1. **知道负载要来**，未来的超温才会出现在预测里。不知道，预测里就没有超温，优化器没有理由动风扇。
2. **早动作对后面仍然有效**，即 T_2 中 u_0 的系数 $ab \neq 0$。这来自热容。若 $a = 0$，提前开风扇毫无用处。
3. **代价函数是凸的**，一次大动作比几次小动作之和更贵（$0.418^2 > 0.207^2 + 0.241^2$）。风扇功耗 $\propto u^3$ 正好满足这一条。若代价与风量成正比，分摊就没有好处。

仿真中 $N = 24$，约束作用在每一步上，道理完全相同：负载跳变一进入预测窗口，预测中就出现超温，优化器把所需的冷却分摊到之前的多个周期。越早的动作折扣越多（系数 $a^k b$ 随 k 衰减），所以分摊不会无限往前，结果就是图 13 中负载到来前一到两分钟开始的平缓爬升。

线性模型有误差：按非线性模型计算，上述两种方案下 60 s 时的温度都约为 79.7°C，并非恰好 78°C。实际运行中 MPC 只执行第一步，下一次用新的测量重新计算，误差会被逐步纠正。

14 仿真比较

在同一测试场景、同一 80°C 限值下比较四种控制方式（设置见附录 L）。被控对象使用非线性模型，MPC 内部用的是每步重新线性化的模型，两者之间存在误差。“只用实测功耗”的 MPC 不知道未来，假设未来负载等于当前测得的功耗；“有负载预报”的 MPC 事先知道 300 s 时负载会升高。

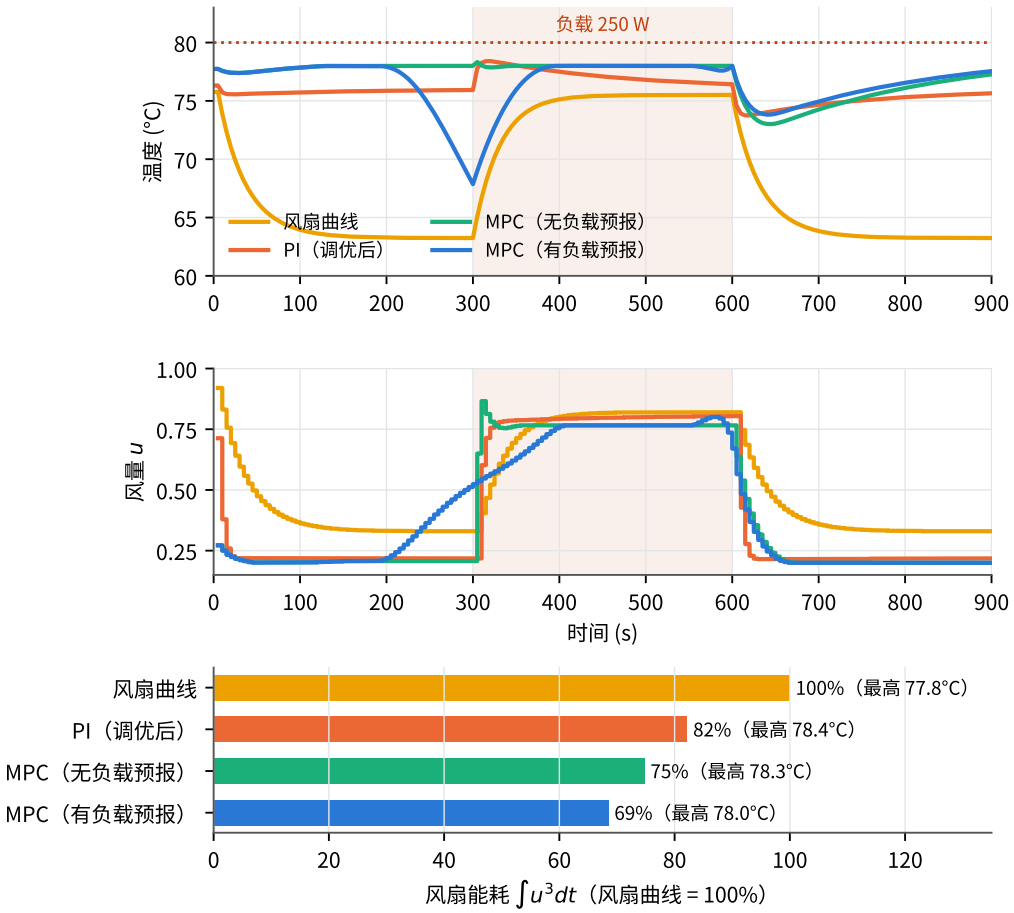


图 15 四种控制方式的温度（上）、风量（中）与风扇能耗（下）。能耗以风扇曲线为 100%。图中“MPC（无负载预报）”即只用实测功耗的 MPC。

表 7 风扇能耗的减少及其来源（以风扇曲线的能耗为 100%）

从	到	减少	来源
风扇曲线	PI	18%	允许温度更高。任何不过度冷却的方法都能得到，与 MPC 无关。
PI	MPC（只用实测功耗）	7%	显式约束：MPC 知道限值在 78°C，可以贴着限值运行；PI 只能降低目标温度留出余量。
MPC（只用实测功耗）	MPC（有负载预报）	6%	预报、热容与 u^3 的凸性：提前平缓加速比事后猛然加速便宜。

需要说明，这一仿真条件比较理想：没有传感器噪声，模型参数准确，负载预报完全正确。节省的也只是风扇的能耗，风扇只占整台服务器功耗的一部分。后两节分别审视负载预报和预测窗口这两个前提，第五部分再讨论模型参数不准时怎么办。

15　负载预报从哪里来

表 7 的最后一行依赖一个强假设：控制器事先知道 300 s 时负载会从 100 W 跳到 250 W。现实中谁来告诉它？如果负载是随机突变的，没有人能事先知道。这一节回答三个问题：负载信息有哪些来源；没有预报时 MPC 还剩下什么；预报出错的代价有多大。

15.1　三类负载信息

表 8　负载信息的来源

场景	信息从哪里来	性质
调度器派发的任务（批处理、高性能计算、AI 训练集群）	任务在队列中排着，调度器知道几秒或几十秒后要启动它，同类任务的功耗有历史记录	不是预测。跳变本来就由调度器引起，它当然先知道
有规律的负载（定时备份、每日业务高峰）	历史数据中的周期规律	统计预测。需要系统运行一段时间、积累数据，有误差
随机突变（用户请求突然涌入）	无法事先获得	只能在发生之后测到

第一类在数据中心里比想象中常见：调度器不仅知道任务何时开始，还可以把任务推迟几秒启动，主动给风扇留出提前量。第二类需要积累数据，正是第五部分中时间序列与深度学习模型的用武之地。第三类没有任何办法事先知道。

15.2　没有预报时：预测负载与预测温度

即使负载完全无法预报，MPC 也不是白搭。关键是区分两件事：预测负载，与预测温度。

负载突变的瞬间，功耗立刻就能测到：CPU 的功率计数器（如 RAPL）或 BMC 读数，延迟在毫秒到秒级。温度却要过很久才反映出来：功耗从 100 W 跳到 250 W，多出的 150 W 全部进入 400 J/K 的热容，温度最初的上升速度只有

$$\left.\frac{dT}{dt}\right|_{\text{跳变瞬间}} = \frac{250-100}{C} = \frac{150}{400} \approx 0.38\ \text{K/s}, \tag{40}$$

要一两分钟才接近新的平衡。于是在跳变后的第一个采样时刻，两类控制器看到的世界完全不同：

- PI 看到的温度几乎没变，一切如常，要等温度升上来才开始加速；
- MPC 把测到的 250 W 代入模型，假设未来保持 250 W，预测出“照现在的风量，两分钟后会到 88°C”，当场加大风扇。

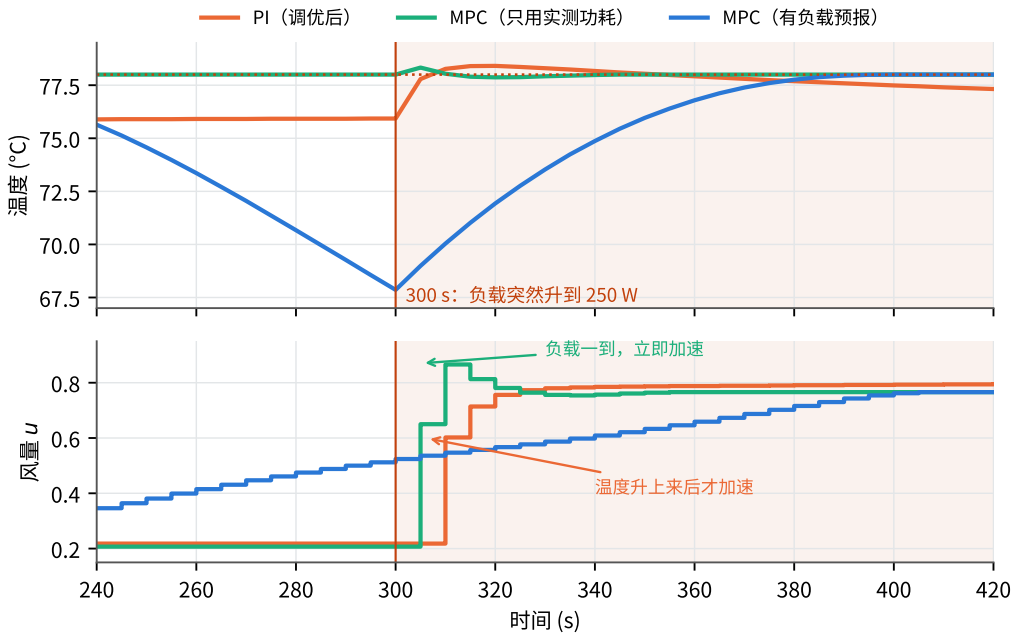


图 16　负载跳变前后的放大图。只用实测功耗的 MPC 不知道负载要来，但在跳变后的第一个采样时刻就把风量提到 0.87；PI 要等温度升上来，风量在随后十几秒内才逐步升高。有预报的 MPC 则提前一到两分钟开始爬升。

这就是图 16 中只用实测功耗的 MPC 的行为。它对未来一无所知，却仍比调好的 PI 省 7% 的风扇能耗。这部分收益来自两样与预报无关的东西：它知道限值在哪里，所以能贴着限值运行；它能从“当前功耗加物理模型”推算出温度的走向，不必等温度自己升上来。

所以 MPC 需要预测的始终是温度，负载预报只是让温度预测更准、看得更远的一项额外输入。有预报，可以吃到提前预冷的那 6%；没有预报，这 6% 拿不到，其余的收益仍在。如果连功耗都测不到、只有温度可用，MPC 相对 PI 的优势就只剩约束处理一项了。

15.3　突变的安全余量

负载无法预报时，安全靠两样东西。

第一是热容争取到的反应时间。设系统原本稳定在 76°C，负载在一个采样周期刚开始时跳到 250 W，而风扇要到下一个采样时刻才会改变。这 5 秒内温度最多上升约 $0.38 \times 5 \approx 1.9$ K。下一个时刻风扇开到足够大，温度就会回落：满速时散热能力为 5.69 W/K，在 78°C 时能带走 $5.69 \times 53 \approx 302$ W，大于 250 W。因此只要温度平时离限值留约 2 K，最坏的突变也不会越界。

第二是把这种最坏情况直接写进优化问题：要求“即使下一刻负载跳到最大值，温度也不超限”。这叫鲁棒 MPC（第 15.5 小节）。代价是平时多留一点余量，多耗一点电。

15.4 预报出错的代价

预报会错。最典型的错误是虚警：预报说有任务，任务却没有来。仿真这种情形：被控对象的负载始终是 100 W，而控制器在 300 s 之前一直收到“300 ～ 600 s 负载为 250 W”的预报，到 300 s 发现任务没来，才改回实测值。

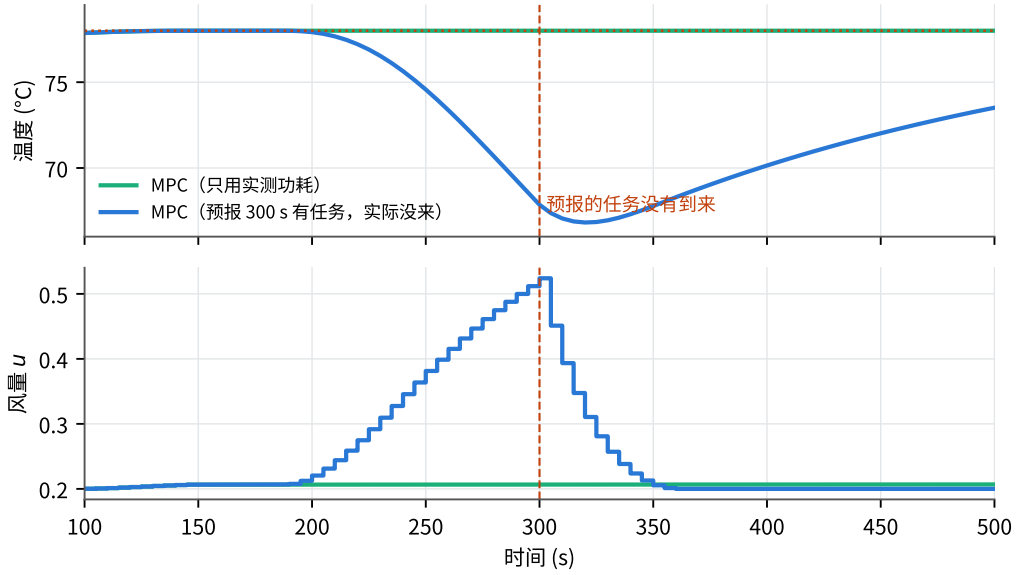


图 17　预报虚警。错误的预报使 MPC 从约 195 s 起提前加速，把温度预冷到 66.8°C；任务没有到来，这部分能耗白白花掉。

在 150 ～ 650 s 这段时间里，虚警使风扇能耗（按 $\int u^3 dt$ 计）从 4.4 增加到 11.0，多花了 6.6。作为对照，在第 14 节的场景中，正确的预报节省了 $143.4 - 131.4 = 12.0$。粗略地说，若预报正确的概率为 p，期望收益为 $12.0\,p - 6.6\,(1-p)$，大于零要求 $p > 0.35$。这个数字只针对本例，但它说明了一个实际的判断标准：**预报不必完美，只要对的时候赚的比错的时候亏的多**。

另一种错误是漏报：任务来了，预报却没说。这时 MPC 退化为只用实测功耗的 MPC，仍然安全，只是没有提前量。

15.5 把不确定性写进约束：鲁棒与随机 MPC

既然负载信息有误差，就可以把误差直接写进约束，有两种思路。

鲁棒 MPC：按最坏情况设计。 假设负载可能在 100 ～ 250 W 之间任意变化，要求约束在所有可能的情况下都满足。最直接的实现是把约束 $T_k \le 78$ 收紧为 $T_k \le 78 - \delta_k$，δ_k 按最坏情况算出。上面“留约 2 K 余量”的做法就是它最简单的形式。代价是保守：为了防备很少发生的最坏情况，平时一直多花电。

随机 MPC：允许极小的违反概率。 若负载预报不仅给出一个数，还给出误差的大小，就可以要求“温度不超限的概率至少为 99%”。设第 k 步预测温度的误差近似服从标准差为 σ_k 的正态

分布，这一要求等价于把约束收紧为

$$\hat{T}_k \le 78 - 2.33\,\sigma_k, \tag{41}$$

2.33 是正态分布 99% 分位点对应的倍数。越远的预测越不确定，σ_k 越大，约束收得越紧。收紧后的约束仍是一次不等式，第三部分的凸性不受影响。随机 MPC 比鲁棒 MPC 省电，代价是需要可靠的误差估计，并接受极小概率的违反。

> **小结**　负载预报决定 MPC 能不能“提前”，不决定 MPC 能不能用。有调度信息时，预报近乎免费；有统计规律时，需要积累数据，并权衡虚警的代价；完全随机时，MPC 依靠实测功耗和物理模型预测温度，依然比只看温度的反馈控制反应更快、更省电，安全则靠约 2 K 的余量或鲁棒约束来保证。

16　预测 N 步的前提：惯性

MPC 每次都要预测 N 步，这件事依赖系统的什么性质？答案是惯性，也就是热容。这一节用一个反例说明这一点，再说明惯性决定了预测要看多远，最后指出只有惯性还不够。

16.1　没有惯性会怎样

设想热容趋于零，温度瞬间达到平衡：

$$T = T_{\text{in}} + \frac{P}{G(u)}. \tag{42}$$

此时风扇的效果立刻显现，也立刻消失，没有任何东西需要预测。每一时刻只需解一个代数不等式 $T_{\text{in}} + P/G(u) \le 78$，得到恰好不超温的最小风量：

$$u_{\min}(P) = 0.5 \left(\frac{P}{3.5 \times 53} \right)^{1/0.7}. \tag{43}$$

$P = 100$ W 时 $u_{\min} = 0.21$，$P = 250$ W 时 $u_{\min} = 0.77$。MPC 退化为“按当前功耗查一张表”，$N = 1$ 就够了，提前动作也不可能，因为冷却效果存不下来。

16.2　惯性决定预测要看多远

有热容时，风扇的一次动作，效果会分摊到之后很长一段时间。图 18 让风扇在 0 ∼ 5 s 多开 0.2，然后恢复原值，比较三种热容下温度的反应。

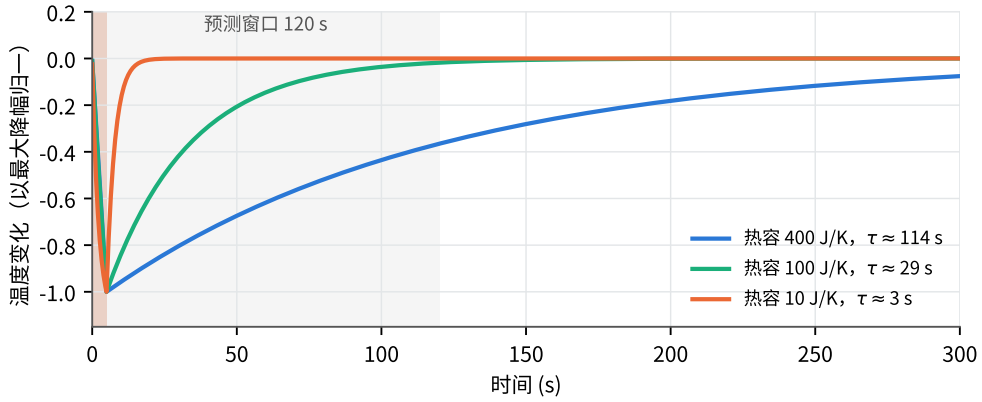


图 18　风扇在 0 ～ 5 s（浅橙色）多开 0.2 后，温度的变化（以最大降幅归一）。热容越大，这次动作的效果保留得越久；灰色区域为 120 s 的预测窗口。

本例的热容为 400 J/K，时间常数约 114 s，120 s 之后这次动作的效果还保留约 35%。热容为 10 J/K 时，效果在十几秒内就消失了。这条衰减曲线就是矩阵 Φ 的一列（图 11），也是“提前动作”能起作用的物理基础。

由此得到选择预测窗口的规则：$N \cdot t_s$ 应覆盖 1 到 2 个主时间常数。再往后，a^k 已经很小，当前动作对那么远的温度几乎没有影响，算了也无用；窗口太短，则看不到当前动作的完整后果。本例取 $N \cdot t_s = 24 \times 5 = 120$ s。

比热容更难对付的是纯延迟：风从风扇到传感器的传输时间、传感器读数本身的延迟。在这段时间里动作完全看不到效果，只看当前测量的反馈控制很难调好，而基于预测的方法可以把延迟直接写进模型。

16.3　只有惯性还不够

有惯性就一定需要 MPC 吗？不一定。考察第 11 节的算例：没有约束、模型线性、代价是二次的。改变当前温度 T_0 重新求解方程组 (34)，第一步的风量为

$$u_0^* = 0.326 + 0.040\,(T_0 - 72). \tag{44}$$

这是一个比例控制器：温度每高 1 K，风量多开 0.040。推导见附录 I。一般地，**没有约束的线性 MPC，等价于一个固定增益的反馈控制器**（有负载预报时再加一项固定的前馈）。这时每个周期在线求解，只是用绕远的方式算出了一个可以事先算好的线性公式。

约束改变了这一点。温度离限值较远时，约束不起作用，最优动作是一种线性关系；温度贴近限值时，约束起作用，最优动作变成另一种线性关系。最优控制律随工况切换，这是任何固定增益都做不到的。表 7 中 MPC 比 PI 多省的 7%，正是来自这一点。

16.4　MPC 的价值在哪里

把本节和上一节的结论合在一起，就能判断 MPC 在什么场合值得使用（表 9）。

表 9　系统特性与 MPC 的价值

系统特性	MPC 的价值
无惯性（瞬间平衡）	几乎没有，按当前功耗查表即可
有惯性，无约束，无负载信息	小，等价于一个调好的反馈控制器
有惯性，有约束，能测当前功耗	中等：贴着限值运行，负载一变立即反应（本例比 PI 省 7%）
有惯性，有约束，有可靠预报	较大：还能提前预冷（本例再省 6%）
有明显纯延迟	大：反馈控制本身就难以做好
多风扇、多温度点相互耦合	大：一个优化问题统一处理所有约束与耦合

单台服务器的负载多数是随机的，热容又提供了不错的缓冲，大致落在表中第三行：收益有限，但真实存在。到了机柜和集群层面，调度器掌握作业计划，多个风扇和温度点相互耦合，约束更多更紧，才进入后三行，MPC 的投入在这里最值得。

第五部分

拓展：把数据放进 MPC

前四部分的 MPC 里，有几样东西是工程师手写的：一步公式、未来负载、代价函数。自适应、深度学习、强化学习这些方法做的都是同一件事：把其中某一样从“手写”改成“从数据中学”。学哪一样，就对应哪一种方法。本部分沿着这条线逐一展开，每一步都用同一台服务器举例。

17 MPC 里哪些东西是手写的

17.1 把 MPC 写成一行

把前四部分的内容压缩成一个式子。每个采样时刻，MPC 求解

$$\min_{u_0,\dots,u_{N-1}} \underbrace{\sum_{k=0}^{N-1} \ell(u_k, T_k)}_{\text{窗口内的代价}} + \underbrace{V(T_N)}_{\text{窗口之后的代价}} \quad \text{满足} \quad T_{k+1} = f(T_k, u_k, P_k), \quad T_k \le 78. \tag{45}$$

其中有四样东西需要事先给定：

- **一步公式** f：前文由物理推导，含参数 $G_{\text{ref}} = 3.5$；
- **未来负载** P_k：前文要么已知，要么用当前测得的功耗代替；
- **阶段代价** ℓ：前文是 $w_f u^3 + w_d(\Delta u)^2$，表达“要什么”；
- **终端代价** V：窗口结束之后还要付出多少代价。前文一直没写它，相当于取 $V = 0$，即窗口之外的事情一概不管。

此外，每个周期求解这个优化问题本身，也可以看成一个“从状态到风量”的函数，称为**控制律**。

17.2 每一样都可以从数据中学

表 10 把这几样东西逐一列出：前文怎么写的，写不准会怎样，用什么方法从数据中学，在本部分哪一节讨论。

表 10　MPC 中手写的部分与对应的学习方法

手写的东西	前文怎么写	写不准的后果	从数据学的方法	章节
f 的参数	$G_{\text{ref}} = 3.5$ 出厂值	积灰、老化后温度长期偏高	在线最小二乘（自适应 MPC）	18
f 的形状	$(2u)^{0.7}$ 加牛顿冷却	热风回流等效应写不进公式	神经网络补全	19
未来负载 P_k	实测功耗保持不变	看不到将要发生的变化	负载预测模型	20
终端代价 V	$V = 0$	窗口太短时目光短浅	强化学习	21
控制律整体	每周期在线求解	弱算力芯片算不完	模仿学习加安全层	22

阶段代价 ℓ 不在表中。它表达的是业务目标，例如“省电”和“安静”之间怎样取舍，这由人决定，不是数据能替你回答的问题。数据能帮忙的，是算出 ℓ 里那些写不出公式的量，第 21 节末尾会讨论。

17.3　两个求最小值的问题

在往下读之前，有一点必须分清，否则很容易混淆：引入学习之后，系统里有两个求最小值的问题（图 19）。

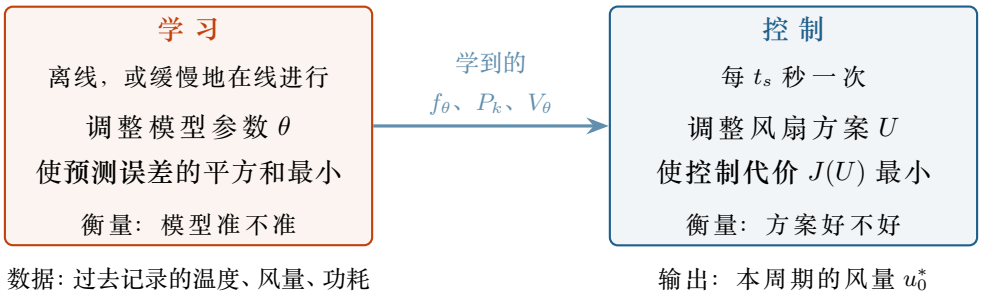


图 19　引入学习后的两个优化问题。左边决定模型长什么样，右边用这个模型决定风扇怎么转。两者的代价函数不同，变量也不同。

左边是学习：变量是模型参数 θ，代价是“模型预测与实测之差的平方和”，衡量模型准不准。右边是控制：变量是风扇方案 U，代价是第二部分定义的 $J(U)$，衡量方案好不好。学习的结果（一个更准的 f、一份负载预报、一个终端代价 V）交给控制去使用。所以说深度学习“用在代价函数上”，大多数时候说的其实是左边那个训练用的代价函数；控制用的代价函数，结构基本不变。

18　学参数：自适应 MPC

第一种情形最简单：一步公式的形状是对的，只是里面的数不准了。

18.1　模型为什么会过时

前文的模型参数是出厂时测定的，而设备会变。散热器鳍片积灰、风扇轴承老化，都会让同样风量下的散热能力 G_{ref} 下降；机房空调设定改变，会让进风温度 T_{in} 改变。每步重新线性化只能处理工况的变化，处理不了参数本身的漂移。

18.2　参数不准的后果

仿真一个具体情形：散热器积灰，真实的 G_{ref} 从 3.5 降到 2.98（下降 15%），而 MPC 的模型仍以为是 3.5。控制器使用实测功耗，不用预报，其余设置与第 14 节相同。

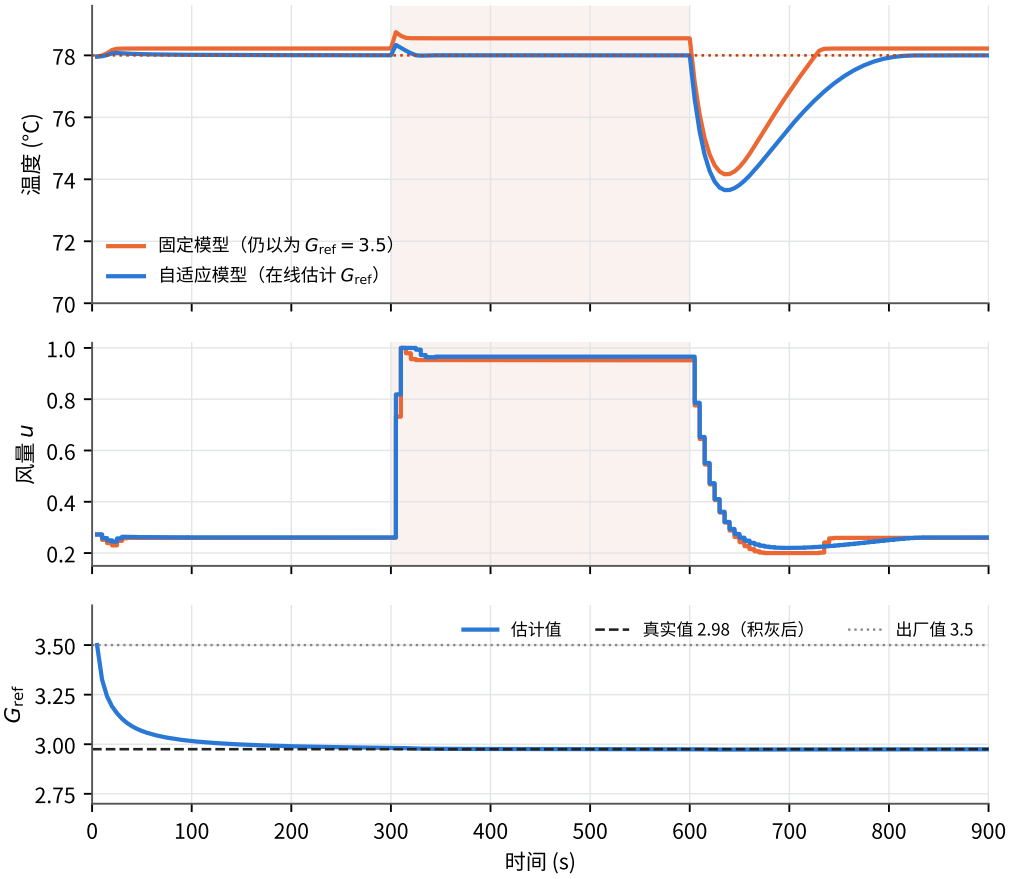


图 20　散热器积灰后（真实 $G_{\text{ref}}=2.98$），固定模型与自适应模型的 MPC。上：温度；中：风量；下：自适应 MPC 对 G_{ref} 的在线估计。

固定模型的 MPC 系统性地高估了风扇的作用，每次都以为“这么多风量刚好够”，结果温度长期停在限值之上：空闲时约 78.2°C，繁忙时约 78.6°C（图 20 上图）。每个周期重新测量能把偏差拉回一部分，但模型的偏差每一步都在重新制造误差，所以消除不了。

18.3 从数据估计参数

一步公式本身就提供了估计 G_{ref} 的方法。把式 (7) 移项：

$$\underbrace{P_k - \frac{C}{t_s}\bigl(T_{k+1} - T_k\bigr)}_{z_k} = G_{\text{ref}} \cdot \underbrace{(2u_k)^{0.7}\bigl(T_k - T_{\text{in}}\bigr)}_{\varphi_k}. \tag{46}$$

左边的 z_k 和右边的 φ_k 全部由测量值算出：功耗、前后两个时刻的温度、执行过的风量。每个采样周期得到一对 (φ_k, z_k)，它们应当落在一条过原点、斜率为 G_{ref} 的直线上。拟合这条直线的斜率，就得到 G_{ref} 的估计。

设备在缓慢变化，越旧的数据越不可信，所以给第 i 个数据乘以权重 λ^{k-i}（$0 < \lambda < 1$，称为遗忘因子）。令加权误差平方和对 G_{ref} 的导数为零，得到

$$\hat{G}_{\text{ref}} = \frac{\sum_{i \le k} \lambda^{k-i}\, \varphi_i\, z_i}{\sum_{i \le k} \lambda^{k-i}\, \varphi_i^2}. \tag{47}$$

这就是图 19 左边那个“学习”问题的最简单形式：只有一个参数，代价是预测误差的平方和，令导数为零可以直接解出。分子和分母都可以递推更新，每个周期只需两次乘加（附录 J）。仿真中取 $\lambda = 0.97$，相当于主要参考最近约 33 个周期（165 s）的数据。

图 20 下图显示，估计值从出厂值 3.5 出发，30 s 后为 3.13，100 s 后为 3.02，最终稳定在 2.97，与真实值 2.98 一致。模型准确之后，温度回到 78.0°C。自适应 MPC 的风扇能耗比固定模型高约 5%，这是积灰之后真正需要付出的冷却。

自适应改变的只是模型里的一个数。每个周期仍然在新的参数下重新线性化，仍然求解第三部分那个凸问题，计算上的可靠性全部保留。

18.4 注意事项

- **激励**：要估计的参数越多，越需要工况有变化。若风量和温度长期不变，多个参数会无法区分。只估计一个参数时，只要 $\varphi_k \neq 0$ 就没有问题。
- **限幅**：估计值必须限制在物理上合理的范围内（仿真中为 $1 \sim 6$ W/K），防止传感器故障把模型带偏。
- **遗忘因子**：λ 越小，跟踪越快，对噪声也越敏感。积灰以周、月计，实际中 λ 可以更接近 1。
- **更简单的替代**：不估计参数，而是估计一个附加的偏差量（扰动估计，也称无静差 MPC），由卡尔曼滤波在线更新。它实现简单，在工业中最常用，能消除稳态偏差，但不修正 b，模型对风扇作用强弱的判断仍然是错的。

自适应有一个前提：公式的形状是对的，只是数不准。如果形状本身就不对，调一个数是不够的。下一节就是这种情形。

19　学形状：用神经网络补全一步公式

19.1　物理公式漏掉的东西

机柜里常有热风回流：服务器排出的一部分热风绕回进风口，使实际的进风温度高于 25°C。风扇转得越慢，气流越弱，回流越严重。设回流比例为 $0.5(1-u)^2$，即风量 0.2 时有 32% 的排风回到进风口，风量 1 时没有回流。工程师手写的一步公式不知道这件事。

怎样发现公式漏掉了东西？看一步预测误差：

$$e_k = T_{k+1}^{\text{实测}} - f(T_k, u_k, P_k), \tag{48}$$

即“下一时刻实际测到的温度”减去“一步公式预测的温度”。这个量每个周期都能算出来，不需要额外的传感器。

图 21a 是 600 个周期的一步预测误差，横轴为当时的风量，颜色为当时的温度。误差不是杂乱的噪声，而是有清楚的规律：风量越低、温度越高，误差越大，最多达到每 5 秒 0.45 K；风量接近 1 时误差接近零。误差有规律，就说明公式少了某个依赖于风量和温度的项，而这个项可以从数据中学出来。

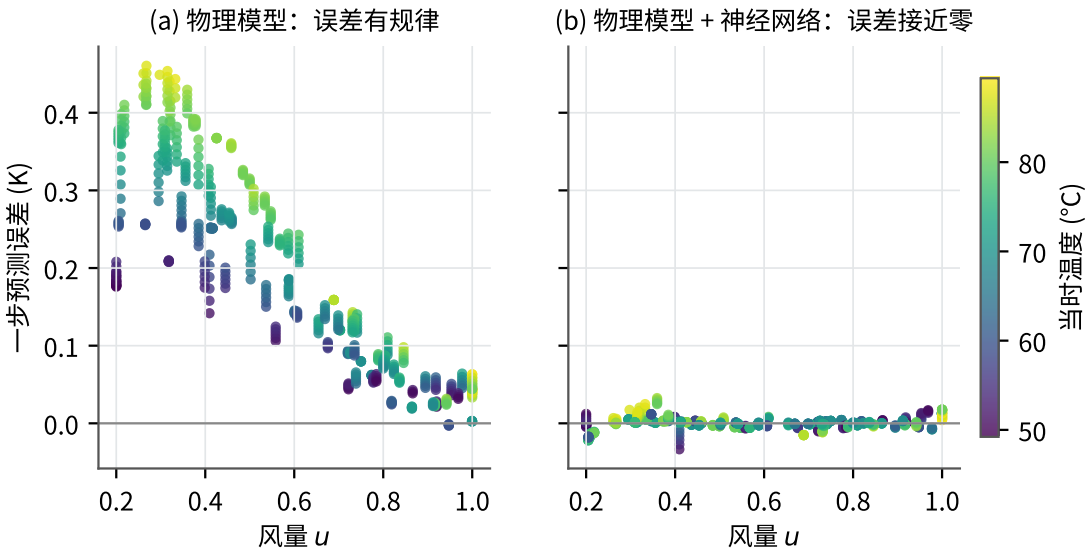


图 21　一步预测误差与风量的关系（未参与训练的 600 个周期）。(a) 只用物理公式，误差随风量和温度呈现系统性规律；(b) 物理公式加上神经网络修正项后，误差接近零。

19.2　为什么调一个参数不够

能不能用上一节的自适应来解决？试一下。图 23 下图是自适应估计的 G_{ref}：空闲时（风量约 0.3，回流严重）被拉低到 2.67，繁忙时（风量约 0.8，回流很轻）又被拉高到 3.39。同一台设备，“参数”却随工况来回变化。这说明误差的形状不能用“把 $G(u)$ 整体乘一个数”来表示。自适应只能追着工况跑，每次负载切换都要花一两分钟重新收敛。

需要学的不是一个数，而是一个函数：误差怎样随温度、风量、负载变化。

19.3　神经网络：一个可以调的函数

神经网络就是一类带有很多可调参数的函数，调整这些参数可以让它近似各种形状的曲线和曲面。这里用的网络只有一层隐藏层，写出来是

$$r_\theta(T,u,P)=w_0+\sum_{j=1}^{16} w_j\tanh\!\big(\alpha_{j1}\tilde T+\alpha_{j2}\tilde u+\alpha_{j3}\tilde P+\beta_j\big),\tag{49}$$

其中 $\tilde T,\tilde u,\tilde P$ 是把温度、风量、功耗缩放到同一量级后的值。tanh 是一条 S 形曲线，从 -1 平滑地过渡到 $+1$。括号里的 α 和 β 决定每条 S 曲线朝哪个方向、在哪里、多陡地过渡；外面的 w 决定每条曲线的高度。16 条 S 曲线叠加起来，可以拼出各种平滑的形状，道理类似于用许多小台阶拼出一条曲线（图 22）。全部参数 θ 一共 81 个数。

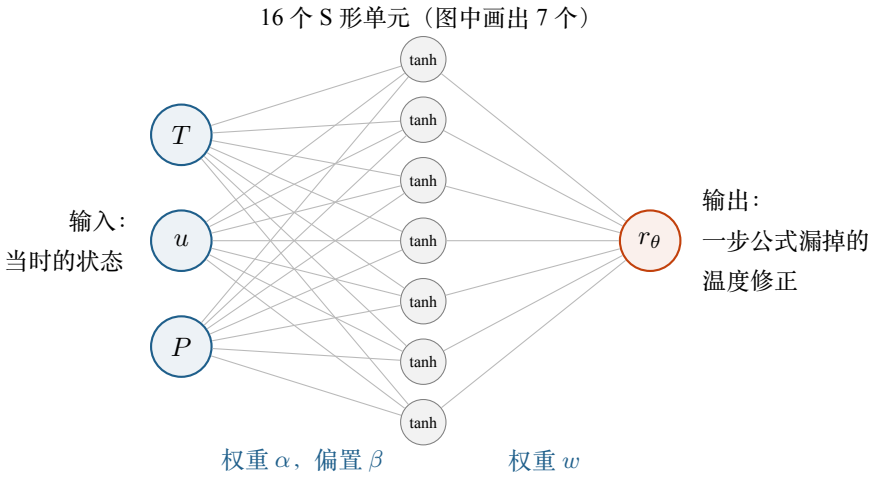


图 22　本节使用的神经网络。三个输入经过 16 个 S 形单元，加权求和得到一步公式的修正量。

19.4　训练：又一个求最小值的问题

用这个网络去补一步公式漏掉的部分：

$$T_{k+1}=\underbrace{f(T_k,u_k,P_k)}_{\text{物理公式}}+\underbrace{r_\theta(T_k,u_k,P_k)}_{\text{神经网络修正}}.\tag{50}$$

网络要学的目标，正是上面算出的一步预测误差 e_k。从运行记录中取 2400 个周期（约 3.3 小时）作为训练数据。采集时有意让风量和负载在允许范围内多变一些，使数据覆盖各种工况，选择 θ 使

$$L(\theta)=\sum_{k}\big(e_k-r_\theta(T_k,u_k,P_k)\big)^2\tag{51}$$

最小。这就是图 19 左边的学习问题。

求最小值的方法和第三部分一样：令 L 对每个参数的导数为零。不同的是，这里有 81 个参数，而且 tanh 是非线性的，导数为零的方程解不出来，只能迭代：每一步沿着让 L 下降最快的方向把参数挪一小步，即

$$\theta\ \leftarrow\ \theta-\eta\,\frac{\partial L}{\partial\theta},\tag{52}$$

称为梯度下降。导数 $\partial L/\partial\theta$ 用链式法则逐层算出，这个计算在深度学习里叫“反向传播”，本质上就是系统地使用链式法则（附录 K）。

第三部分说过，在线控制不能忍受迭代次数不确定、可能停在假谷底的问题。训练却可以忍受：它在离线进行，时间充裕，可以多试几个起点，挑最好的结果。这也是为什么学习放在左边、控制放在右边，各自用适合自己的求解方式。

训练完成后，在没有参与训练的 600 个周期上检验：只用物理公式时，一步预测误差的均方根为 0.233 K；加上神经网络修正后降到 0.0076 K，缩小到约 1/30（图 21b）。

训练得到的是什么。 这里要分清 θ 和 r_θ。θ 是式 (49) 中那 81 个数：$\alpha_{j1},\alpha_{j2},\alpha_{j3},\beta_j,w_j$ 和 w_0。r_θ 是由这 81 个数确定的函数。训练求的是 θ，也就是这 81 个数；数一旦定下来，式 (49) 就成了一个完全确定的公式，给它任意一组 (T,u,P)，都能算出修正量，也能对它求导。这和第 18 节的自适应是同一回事：自适应的 θ 只有一个数 G_{ref}，求出来代回一步公式；这里的 θ 有 81 个数，求出来代回式 (49)。

所以训练结束后，手里有的是一个新的、完全确定的一步公式：

$$f_{\text{新}}(T,u,P)=f(T,u,P)+r_\theta(T,u,P), \tag{53}$$

其中 f 是物理公式，r_θ 是参数已经定好的神经网络。

19.5 放进 MPC：照第三部分的步骤走

接下来把 $f_{\text{新}}$ 当作一步公式，按第三部分的步骤逐一执行。

预测。 把 $f_{\text{新}}$ 串起来，$T_1=f_{\text{新}}(T_0,u_0,P_0)$，$T_2=f_{\text{新}}(T_1,u_1,P_1)$，与第 6 节完全相同。

线性化。 在展开点 $(\bar T,\bar u,\bar P)$ 处做一阶泰勒展开，与第 9 节完全相同，只是求导的对象换成了 $f_{\text{新}}$。由于 $f_{\text{新}}$ 是两项之和，它的函数值和偏导数也都是两项之和：

$$c=f+r_\theta,\qquad a=\frac{\partial f}{\partial T}+\frac{\partial r_\theta}{\partial T},\qquad b=\frac{\partial f}{\partial u}+\frac{\partial r_\theta}{\partial u},\qquad e=\frac{\partial f}{\partial P}+\frac{\partial r_\theta}{\partial P}, \tag{54}$$

全部在展开点处取值。前一项就是式 (27)；后一项是神经网络的函数值和导数。神经网络的导数用链式法则对式 (49) 逐项求出（附录 K），例如

$$\frac{\partial r_\theta}{\partial u}=\sum_{j=1}^{16} w_j\bigl(1-\tanh^2(s_j)\bigr)\,\alpha_{j2}\cdot\frac{1}{\sigma_u},$$

其中 s_j 是第 j 个单元括号里的值，σ_u 是风量的缩放系数。

以空闲工况（$\bar T=78^\circ$C，$\bar u=0.25$，$\bar P=100$ W）为例：物理公式给出 $c=77.82$，神经网络在这一点算出修正量 $r_\theta=+0.40$，两者相加得 $c=78.23$；物理公式给出 $b=-4.00$，神经网络算出 $\partial r_\theta/\partial u=-0.16$，相加得 $b=-4.16$；a 则从 0.973 变为 0.980。

求解。 得到 a、b、c、e 之后，神经网络就不再出现了。预测温度仍是风扇方案的一次函数，代价函数仍是凸的，求解方法与第三部分完全相同。

所以神经网络进入 MPC 的方式，最终体现为每个周期线性化系数的几个数的变化。表 11 比较两个工况点的结果。

表 11　三种一步公式在两个工况点的线性化系数（温度 78°C，$t_s = 5$ s）

	空闲：$u = 0.25$，$P = 100$ W			繁忙：$u = 0.77$，$P = 250$ W		
	a	b	c	a	b	c
真实系统	0.981	−3.92	78.22	0.944	−3.41	78.07
只用物理公式	0.973	−4.00	77.82	0.941	−2.85	77.99
物理公式 + 神经网络	0.980	−4.16	78.23	0.945	−3.41	78.08

空闲时，物理公式算出的 c 比真实值低 0.4 K：它以为保持现有风量，5 秒后温度是 77.82°C，实际是 78.22°C。这 0.4 K 正是热风回流造成的，加上神经网络之后，偏差基本消失。神经网络对 b 的修正并不完美（空闲时为 -4.16，真实值为 -3.92），因为训练只要求修正量本身准确，并不直接要求它对风量的导数准确；MPC 每个周期都会重新测量，这点误差的影响不大。

19.6 结果

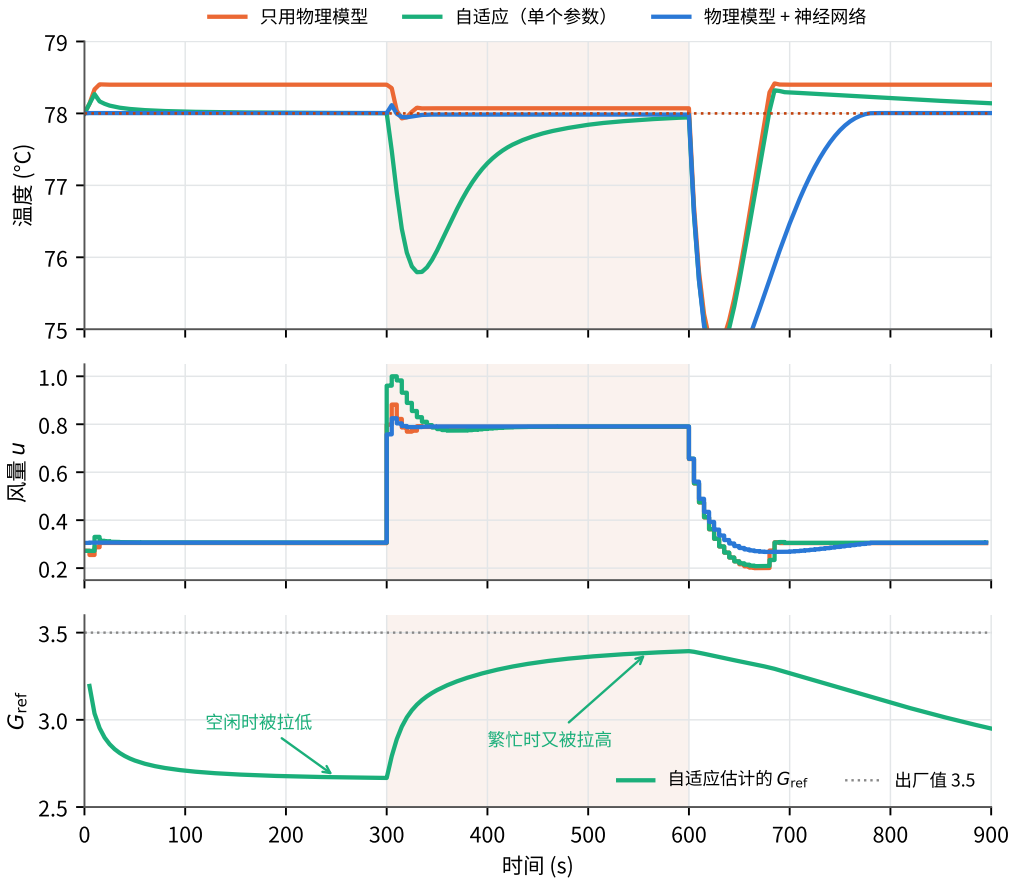


图 23　存在热风回流时三种 MPC 的比较。上：温度；中：风量；下：自适应 MPC 估计的 G_{ref}，随工况来回变化。

图 23 比较了三种 MPC，测试场景与第 14 节相同：

- **只用物理公式**：空闲时温度一直停在 78.4°C，持续越过限值。
- **自适应**：空闲时能把温度拉回 78.0°C，但负载一切换，估计值就要重新收敛：升载后它仍以为风扇很弱，过度冷却，温度一度降到 75.8°C；降载后又以为风扇很强，温度再次越过限值。风扇能耗比另外两种高约 5%。
- **物理公式加神经网络**：在所有工况下都把温度稳定在 78.0°C，风扇能耗与只用物理公式时相同。

单台服务器上，回流造成的偏差只有零点几开；到了机柜层面，多台服务器的排风互相影响，这类写不进公式的效应会大得多，学习的价值也更大。

19.7 为什么保留物理公式

也可以完全不要物理公式，让神经网络直接学 $T_{k+1} = f_\theta(T_k, u_k, P_k)$。实践中一般不这样做，原因有三：

- **外推**：神经网络只在见过的数据范围内可靠。训练数据里温度最高只到 89°C，网络对 95°C 时会发生什么一无所知；物理公式在任何温度下都成立。保留物理公式，网络只负责一个小的修正，即使在陌生工况下出错，错误也有限。修正量还可以限幅，例如不超过 ± 1 K。
- **数据量**：只学一个小修正，比从头学整个规律需要的数据少得多。
- **求解**：只要每步线性化，网络多复杂都不影响第三部分的凸性。若不线性化，直接把网络串起来做预测，第三部分的困难就全部回来了，只能改用迭代求解或采样式的方法，例如随机生成大量风扇方案、用网络并行模拟、挑代价最低的。

20 学负载：预测未来的 P_k

第四部分说明了负载信息的价值和虚警的代价。如果负载有统计规律，预测模型就从历史数据中把规律学出来。它同样是图 19 左边的学习问题，只是学的对象换成了负载：

- **输入**：过去十分钟的功耗、当前时刻、星期几、调度队列中等待的任务数、请求速率等；
- **输出**：未来 120 s 内每 5 s 的功耗，共 24 个数，正好是 MPC 需要的 $P_0, \dots, P_{23}$；
- **训练代价**：预测功耗与实际功耗之差的平方和，与式 (51) 同一形式。

模型可以从简单到复杂：最简单的是“今天此刻的负载约等于昨天此刻”；其次是线性回归、梯度提升树；最复杂的是 LSTM、Transformer 这类专门读取时间序列的神经网络。应当从简单的开始，只有当简单模型的误差明显妨碍控制时，才换复杂的。

负载预测进入 MPC 的位置只有一处：线性预测式中的负载项 $e\,(P_k - \bar{P})$，即式 (32) 之外多出的 $\Psi \boldsymbol{D}$。它只改变预测温度中的常数部分，不改变风量前的系数。因此不论预测模型多复杂，控制用的优化问题都还是原来那个凸问题。这是数据进入 MPC 风险最低的位置。

预测总有误差。处理办法在第四部分已经出现过：用虚警与漏报的代价判断预测值不值得用（第 15 节）；让预测模型同时给出误差的大小，交给随机 MPC 收紧约束（第 15.5 小节）。

21 学终端代价：强化学习的位置

21.1 窗口太短会怎样

MPC 只计算窗口内的代价，窗口之外的事一概不管。窗口足够长时这没有问题；但算力有限时，窗口可能很短，MPC 就会目光短浅。

用第 13 节的两步例子说明。原本 MPC 看两段（60 s），算出第一段应当提前开到 0.42。现在假设它只能看一段（30 s）：第一段末的温度约束 $T_1 \le 78$ 不做任何事也能满足，于是它选择 $v_0 = 0$，不提前。30 s 后负载到来，只能把第二段开到 0.63，总代价 0.175，比提前分摊的 0.101 高 73%。

21.2 终端代价：替窗口之外的事情记账

补救的办法是在窗口末尾加一个终端代价 $V(T_1)$：如果第一段结束时温度是 T_1，之后至少还要付出多少代价。

在这个例子里，V 可以手算。已知第二段负载为 250 W，第二段末的温度为 $T_2 = 88.37 + 0.857\,(T_1 - 77.06) - 24.8\,v_1$。要使 $T_2 \le 78$，第二段至少要多开

$$v_1 = \max\!\left(0,\ \frac{10.37 + 0.857\,(T_1 - 77.06)}{24.8}\right),$$

于是 $V(T_1) = v_1^2$。第一段结束时温度越低，V 越小：现在多冷却一点，之后就少付一点。

把它加到只看一段的 MPC 上，代价变为 $v_0^2 + V(T_1)$，其中 $T_1 = 77.06 - 24.8\,v_0$。图 24 画出了这个函数：最低点在 $v_0 = 0.207$，与看两段的 MPC 完全相同。有了正确的终端代价，只看一步的 MPC 也会提前动作。

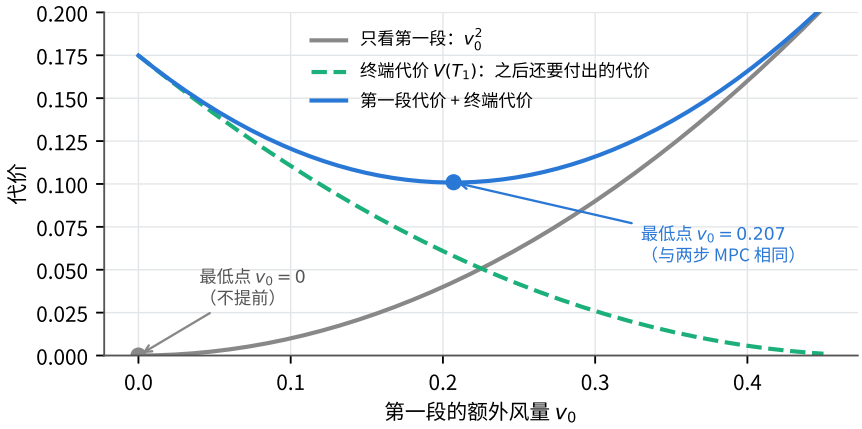


图 24　只看一段的 MPC。没有终端代价时（灰线），最低点在 $v_0 = 0$，不提前；加上终端代价后（蓝线），最低点移到 $v_0 = 0.207$，与看两段的结果相同。

21.3 终端代价从哪里来：强化学习

上面能手算，是因为我们知道第二段的负载，问题也只有两段。一般情况下，“从某个状态出发，之后至少还要付出多少代价”是一个未知的函数，称为**价值函数**。它满足一个很直观的关系：

$$V(\text{现在的状态}) = \min_{u}\Big[\ \text{这一步的代价}\ +\ V(\text{下一步的状态})\ \Big], \tag{55}$$

即“从现在起的最小总代价，等于这一步的代价加上从下一步起的最小总代价”，称为贝尔曼方程。

强化学习就是从经验中学出这个函数的方法。最基本的做法是：

1. 让系统运行（通常在仿真器里，以免损坏真实设备），记录每一步的状态、动作和代价；
2. 用一个神经网络 V_θ 表示价值函数；
3. 调整 θ，使记录下来的每一步都尽量满足式 (55)，即使

$$\sum_k \Big(V_\theta(\text{第}k\text{ 步状态}) - \big[\text{第}k\text{ 步代价} + V_\theta(\text{第}k+1\text{ 步状态})\big] \Big)^2$$

最小，并反复进行。

这又是图 19 左边的一个学习问题，训练代价是“贝尔曼方程两边之差的平方和”。学到的 V_θ 交给右边，作为 MPC 的终端代价。这里的状态可以不只是温度，还可以包括当前负载、时刻、调度队列等，凡是影响将来代价的量都可以放进去。

21.4 MPC 与强化学习是一条线的两端

从式 (45) 看，MPC 和强化学习的区别只在于怎样分配“显式计算”和“学到的估计”：

- **纯 MPC**：窗口内的每一步都用模型显式计算，$V = 0$；
- **纯强化学习**：几乎不显式计算（窗口只有一步甚至零步），所有将来的代价都交给学到的 V，或者直接学一个“状态到动作”的函数（称为策略）；
- **两者结合**：较短的窗口加上学到的终端代价。窗口内的约束仍然显式处理，窗口外的长远后果由 V 估计。

显式计算的部分需要模型，但约束处理得清清楚楚，每个决定都能解释；学到的部分不需要准确的模型，能看得很远，但需要大量数据或高保真的仿真器，也难以保证约束。单台服务器的风扇物理清楚、变量少、约束严格，适合以显式计算为主；数据中心的制冷系统（冷水机组、冷却塔、水泵、空调相互耦合）难以建模，更适合以学习为主。DeepMind 曾报道，用机器学习控制 Google 数据中心的制冷系统，最多降低了约 40% 的制冷能耗。

21.5 阶段代价里能学什么

剩下的是阶段代价 ℓ。它表达“要什么”，例如省电与安静之间的取舍，由人决定。但 ℓ 里常常含有写不出公式、需要预测的量，这些量可以学。

例如，若关心的不只是风扇的电，还有整机的电，就要把 CPU 的漏电功耗写进代价：

$$\ell = w_f\, u^3 + P_{\text{leak}}(T). \tag{56}$$

漏电随温度上升，温度太高时，漏电多花的电可能超过风扇省下的电，最优温度就不再贴着限值。$P_{\text{leak}}(T)$ 很难从原理写出，但可以在同样的计算任务下，测量不同温度时的整机功耗，从数据中拟合出来。只要拟合结果随温度单调上升且向上弯曲（例如指数形式），由于线性化

后 T 是风扇方案的一次函数，代价仍是凸的。这类直接优化真实经济指标的做法称为经济型 MPC。

类似地，风扇噪声与转速的关系、高温对硬件寿命的影响，都可以从数据中学出来，再放进代价。权重 w_f、w_d 也可以在仿真或实际运行中，按一天的总能耗、超温时长等最终指标反复试验来调整。数据帮你把代价算准，代价表达的目标仍由你决定。

22　学整个控制律：模仿学习与安全层

最后一种做法走得最远：不在线求解优化问题，直接学一个“状态到风量”的函数。

附录 I 说明，线性 MPC 的最优风量是当前温度与负载信息的分段线性函数。既然如此，就可以离线把 MPC 在大量不同状态下求解一遍，记录“状态到最优风量”的对应关系，再训练一个神经网络去模仿。在线时只需计算一次神经网络，速度极快，适合 BMC 这类算力很弱的芯片。这叫**模仿学习**，也称近似 MPC。显式 MPC 是它的精确版本：离线把每一段线性函数都算出来，在线查表。

学到的控制律只是近似，在训练数据没有覆盖的工况下可能给出危险的动作，例如把风扇降得过低。强化学习直接学到的策略也有同样的问题。工程上的标准做法是在控制律和风扇之间加一个**安全层**：设学到的控制律给出风量 u_{AI}，求解一个很小的优化问题

$$\min_{u} \left(u - u_{\mathrm{AI}}\right)^2 \qquad \text{满足：线性模型预测的未来几步温度} \le 78, \quad 0.2 \le u \le 1. \tag{57}$$

u_{AI} 本来就安全时，安全层原样放行；否则把它修正为离它最近的安全风量。这个问题正是第三部分的凸问题，求解快速可靠。BMC 的硬件过温保护作为最后一道防线，任何情况下都保留。

学习得到的部分负责做得好，安全层负责不出事，这是目前把学习方法放进控制回路时最普遍的分工。

23　从哪里开始：先看预测误差

这么多方法，对一台具体的服务器该用哪一种？答案取决于数据告诉你哪里不够好。最有用的诊断工具，就是第 19 节中用过的一步预测误差 e_k。

先搭起最基础的版本：物理公式、实测功耗、每步线性化的 MPC，在影子模式下运行（只计算、不下发，由原有的风扇控制器继续工作），记录每个周期的温度、风量、功耗、一步预测误差和求解时间。然后按表 12 对照。

表 12　根据运行记录选择方法

看到的现象	说明什么	对策
预测误差均值接近零，杂乱无规律	模型已经够准	不必再学模型；精力放在负载信息上
预测误差有一个缓慢漂移的偏移，与工况无关	参数在变：积灰、老化、进风温度	自适应或扰动估计（第 18 节）
预测误差随风量、温度、负载呈现固定规律（如图 21a）	公式的形状不对	物理公式加神经网络修正（第 19 节）
模型准确，但负载一变温度总是先冲高	缺少未来的信息	接入调度计划或负载预测（第 20 节）
算力只允许很短的窗口，控制表现短视	窗口之外没有记账	终端代价（第 21 节）
控制芯片上解不完优化问题	算力不足	模仿学习加安全层（第 22 节）

每加入一项，都用同样的指标在影子模式下比较：风扇能耗、越限时长、转速波动。只有指标确实改善，才把它切换到真正控制风扇。

本部分的方法看起来各不相同，做的却是同一件事：把式 (45) 中原本手写的某一样东西，换成从数据中学到的版本。学习负责把模型、负载和代价算准；MPC 的结构，也就是预测、约束、线性化后的凸优化和滚动执行，负责在每个周期可靠地做出决定。

附录 A　一阶线性常微分方程的解

当 u 和 P 不变时，$G = G(u)$ 是常数。令 $T_\infty = T_{\text{in}} + P/G$，则

$$P - G\,(T - T_{\text{in}}) = P - GT + GT_{\text{in}} = -G\Big(T - T_{\text{in}} - \frac{P}{G}\Big) = -G\,(T - T_\infty).$$

方程 (4) 变为 $C\,\dfrac{dT}{dt} = -G\,(T - T_\infty)$。令 $y = T - T_\infty$，因 T_∞ 为常数，$\dfrac{dy}{dt} = \dfrac{dT}{dt}$，于是

$$\frac{dy}{dt} = -\frac{G}{C}\,y = -\frac{y}{\tau}, \qquad \tau = \frac{C}{G}.$$

分离变量后积分：$\displaystyle\int \frac{dy}{y} = -\int \frac{dt}{\tau}$，得 $\ln|y| = -t/\tau + \text{常数}$，即 $y(t) = y(0)\,e^{-t/\tau}$。代回 T 即得式 (5)。$t = \tau$ 时 $e^{-1} \approx 0.37$，温度走完全程的 63%；$t = 3\tau$ 时走完约 95%。

附录 B　泰勒展开、偏导数与求导法则

一元泰勒展开。 光滑函数在点 x 附近可以展开为

$$f(x + d) = f(x) + f'(x)\,d + \tfrac{1}{2}f''(x)\,d^2 + \tfrac{1}{6}f'''(x)\,d^3 + \cdots$$

只保留前两项得到切线（一阶近似），保留前三项得到抛物线（二阶近似）。d 越小，略去的高次项越小，近似越准确。

偏导数。 对多元函数 $f(T, u, P)$，偏导数 $\partial f/\partial u$ 是把 T 和 P 视为常数、只对 u 求导的结果。例如对一步公式 (7)，

$$\frac{\partial f}{\partial T} = 1 - \frac{t_s}{C}G(u), \qquad \frac{\partial f}{\partial u} = -\frac{t_s}{C}G'(u)\,(T - T_{\text{in}}), \qquad \frac{\partial f}{\partial P} = \frac{t_s}{C}.$$

二阶偏导数 $\partial^2 f/\partial u_0 \partial u_1$ 是先对 u_0、再对 u_1 求偏导数的结果；对光滑函数，求导顺序不影响结果。

多元一阶泰勒展开（切平面）。 在点 $(\bar T, \bar u, \bar P)$ 附近，

$$f(T, u, P) \approx f(\bar T, \bar u, \bar P) + \frac{\partial f}{\partial T}(T - \bar T) + \frac{\partial f}{\partial u}(u - \bar u) + \frac{\partial f}{\partial P}(P - \bar P),$$

偏导数均在展开点处取值。它对每个输入都是一次的；两个输入时几何上是一个平面，在展开点处与原函数的曲面相切，所以称为切平面。

链式法则。 若 J 依赖 T，T 又依赖 u，则 $\dfrac{dJ}{du} = \dfrac{dJ}{dT}\cdot\dfrac{dT}{du}$。例如 $J = (T(u) - 70)^2$ 时，$J'(u) = 2\,(T(u) - 70)\,T'(u)$。多元时，若 T_2 依赖 T_1 和 u_1，而 T_1 依赖 u_0，则 $\dfrac{\partial T_2}{\partial u_0} = \dfrac{\partial T_2}{\partial T_1}\cdot\dfrac{\partial T_1}{\partial u_0}$。

乘积法则。 $(pq)' = p'q + pq'$。对 $J'(u) = 2\,(T - 70)\,T'$ 再求导，得 $J''(u) = 2\big[T'^2 + (T - 70)\,T''\big]$，这就是式 (17) 和式 (25) 的来源。

沿方向求导。 令 $g(s)=J(u_0+s\,v_0,\;u_1+s\,v_1)$。由链式法则 $g'(s)=J_0\,v_0+J_1\,v_1$；J_0 和 J_1 本身也是 u_0,u_1 的函数，再用一次链式法则：

$$g''(s)=\big(J_{00}\,v_0+J_{01}\,v_1\big)v_0+\big(J_{01}\,v_0+J_{11}\,v_1\big)v_1=J_{00}v_0^2+2J_{01}v_0v_1+J_{11}v_1^2,$$

即式 (24)。

附录 C　向量、矩阵与线性方程组

向量与矩阵。 把若干个数按顺序排成一列称为向量，例如 $(u_0,u_1,u_2)^{\top}$；把数排成若干行若干列称为矩阵。

矩阵乘以向量。 m 行 n 列的矩阵 M 乘以 n 维向量 $\boldsymbol{v}$，结果是 m 维向量，其第 i 个分量为 $(M\boldsymbol{v})_i=\sum_j M_{ij}\,v_j$，即矩阵第 i 行与向量对应相乘再求和。例如

$$\begin{bmatrix}-17.27 & 0\\ -12.74 & -17.27\end{bmatrix}\begin{bmatrix}u_0\\ u_1\end{bmatrix}=\begin{bmatrix}-17.27\,u_0\\ -12.74\,u_0-17.27\,u_1\end{bmatrix}.$$

一组一次式因此可以紧凑地写成一个矩阵等式，两种写法完全等价。

二元一次方程组。 方程组 $p\,x+q\,y=m$，$q\,x+s\,y=n$ 在 $ps-q^2\neq 0$ 时的解为

$$x=\frac{m\,s-q\,n}{ps-q^2},\qquad y=\frac{p\,n-q\,m}{ps-q^2}.$$

方程组 (34) 中 $p=961.2$，$q=440.0$，$s=636.7$，$m=419.4$，$n=297.1$，得 $u_0=0.326$，$u_1=0.241$。更多未知数时同样可用消元法求解，计算量是固定的。

附录 D　凸函数与凸集

一元凸函数。 若对任意两点 x,y 和任意 $\lambda\in[0,1]$ 都有

$$g\big(\lambda x+(1-\lambda)y\big)\le\lambda\,g(x)+(1-\lambda)\,g(y),$$

则称 g 为凸函数，几何意义是图像上任意两点之间的弦都在图像上方。对二阶可导的函数，这等价于处处 $g''\ge 0$。x^2、e^x 以及 $x>0$ 时的 x^3 都是凸函数；$\sqrt{x}$ 与 $x^{0.7}$ 向下弯曲，不是凸函数。

多元凸函数。 若对任意点 U 和任意方向 $\boldsymbol{v}$，沿直线切下来的一元函数 $g(s)=J(U+s\boldsymbol{v})$ 都是凸函数，则称 J 为凸函数。由附录 B，这等价于在每个点、对每个方向都有 $J_{00}v_0^2+2J_{01}v_0v_1+J_{11}v_1^2\ge 0$。

关键性质。 若凸函数 J 在 U^{*} 处所有偏导数都为零，则 U^{*} 是全局最小值点。

证明。任取方向 $\boldsymbol{v}$，令 $g(s)=J(U^{*}+s\boldsymbol{v})$。偏导数全为零意味着 $g'(0)=J_0v_0+J_1v_1=0$；凸性意味着 $g''(s)\ge 0$，从而 $g'(s)$ 不减。于是 $s>0$ 时 $g'(s)\ge 0$，g 在 $s\ge 0$ 上不减，$g(s)\ge g(0)$，

即 $J(U^* + s\boldsymbol{v}) \geq J(U^*)$。平面上任何一点都可以写成 $U^* + s\boldsymbol{v}$ 的形式，所以 U^* 处的函数值不大于任何其他点。□

运算规则。 凸函数之和仍是凸函数；一次函数的平方是凸函数；凸的递增函数（如 e^x）复合一次函数仍是凸函数。正是这些规则保证了线性化后的各种代价函数都是凸的。

凸集。 若集合中任意两点之间的线段仍在集合内，则称为凸集。一次不等式 $\alpha u_0 + \beta u_1 \leq \gamma$ 定义的区域（直线的一侧）是凸集；多个凸集的交集仍是凸集。在凸集上最小化凸函数，任何局部最小值都是全局最小值，最小值可能在区域内部，也可能在边界上。

附录 E 牛顿法

求根的角度。 要解 $h(u) = 0$，在当前猜测 u 处用切线近似 h：$h(u + d) \approx h(u) + h'(u)\, d$，令其为零得 $d = -h(u)/h'(u)$。求最小值时取 $h = J'$，就得到式 (16)。

抛物线的角度。 在当前点作二阶泰勒展开 $q(d) = J(u) + J'(u)\, d + \frac{1}{2} J''(u)\, d^2$。若 $J''(u) > 0$，抛物线开口向上，令 $q'(d) = 0$ 得最低点 $d = -J'(u)/J''(u)$，与求根角度的结果相同。若 $J''(u) < 0$，抛物线开口向下，这个 d 指向最高点，牛顿法会朝函数增大的方向走。

多元情形。 两个变量时，用切平面分别近似两个偏导数并令其为零，得到方程组 (22)。用矩阵记号写作 $H\boldsymbol{d} = -\nabla J$，其中 $\nabla J = (J_0, J_1)^\top$ 称为梯度，$H = \begin{bmatrix} J_{00} & J_{01} \\ J_{01} & J_{11} \end{bmatrix}$ 称为黑塞矩阵。

收敛性。 J 是二次函数时，J' 是一次函数，切线与它本身重合，一步即到达最小值。J 不是二次函数时，若起点离最小值足够近且附近是凸的，迭代收敛很快；起点较远时可能跳过头、离开允许范围，或在不凸处走向错误方向。实用的求解器通过限制步长保证每一步都使 J 下降，但迭代次数无法事先确定。

附录 F PI 控制

PI 控制根据温度误差 $\epsilon(t) = T(t) - T_{\text{set}}$ 计算风量：

$$u(t) = u_{\text{b}} + K_p\, \epsilon(t) + K_i \int_0^t \epsilon(s)\, ds,$$

其中 u_{b} 为基准风量，K_p 为比例增益，K_i 为积分增益。比例项对当前误差作出反应；积分项累积过去的误差，只要误差不为零就持续调整，从而消除稳态误差。风量达到上下限时需停止积分累积（抗积分饱和）。PI 控制只使用当前和过去的温度，不使用模型，也不使用任何关于未来的信息。

附录 G　二次规划与求解器

目标函数为二次函数、约束为一次不等式的优化问题称为二次规划（Quadratic Programming，QP），一般形式为

$$\min_{U} \; U^{\top} H U + 2\boldsymbol{f}^{\top} U \qquad \text{满足} \qquad A_c U \le \boldsymbol{b}_c,$$

其中 $U^{\top} H U = \sum_{i,j} u_i H_{ij} u_j$ 是二次项的紧凑写法。当 H 对所有非零方向都给出正的弯曲程度时，问题是凸的，有唯一解。没有约束时，令梯度为零得到一次方程组 $HU = -\boldsymbol{f}$，这正是方程组 (34) 的一般形式。有约束时可用有效集法、内点法或交替方向乘子法求解，常用的开源求解器有 OSQP、qpOASES 等。N 为 20 ∼ 50 时，求解时间通常在毫秒量级，迭代次数有可预期的上界。省电型代价含 u^3 项，严格说不是二次函数，但它在 $u > 0$ 时是凸的，可用一般的凸优化求解器直接求解，或在当前风量处做二阶展开后按 QP 求解。

附录 H　欧拉法的精度与采样周期

本例的比较。 风扇不变时有精确解 (5)。令 $t = t_s$，得到精确的一步公式 $T_{k+1} = T_\infty + (T_k - T_\infty)\, e^{-t_s/\tau}$。欧拉公式 (7) 可改写为 $T_{k+1} = T_\infty + (T_k - T_\infty)(1 - t_s/\tau)$。两者只差在每步对偏差所乘的系数：精确值 $e^{-t_s/\tau} = 1 - t_s/\tau + \tfrac{1}{2}(t_s/\tau)^2 - \cdots$，欧拉值 $1 - t_s/\tau$ 是它的前两项。t_s/τ 较小时两者几乎相同；$t_s/\tau > 1$ 时欧拉系数为负，计算结果来回振荡；$t_s/\tau > 2$ 时欧拉系数的绝对值大于 1，计算发散（图 25）。

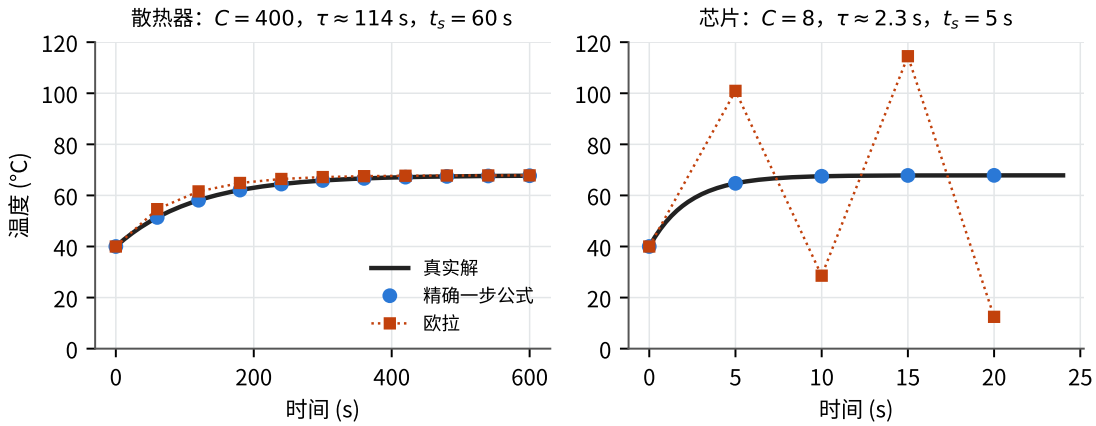


图 25　欧拉法与精确一步公式。左：$t_s/\tau \approx 0.53$，欧拉法略有偏差；右：若只考虑热容很小的芯片，$t_s/\tau \approx 2.2$，欧拉法发散。

一般情形。 上述比较依赖本例的指数解。一般的线性系统 $\dot{\boldsymbol{x}} = A\boldsymbol{x} + B\boldsymbol{u}$ 的精确离散化为 $A_d = e^{At_s}$（矩阵指数），欧拉近似为 $I + At_s$，对每个特征值 λ 比较的是 $e^{\lambda t_s}$ 与 $1 + \lambda t_s$，结论相同。非线性系统一般没有精确的一步解，只能用欧拉法或龙格–库塔法等数值积分。普遍适用的规则是：t_s 应远小于系统最小的时间常数，通常取其 1/10 到 1/20。t_s 过大，模型失准，对负载变化的反应变慢；t_s 过小，覆盖同样长的预测窗口需要更多步，计算量增大；t_s 也不

应小于传感器的更新周期。

附录 I　无约束线性 MPC 等价于固定增益的反馈

取固定的展开点 $(\bar T, \bar u, \bar P)$，线性预测可以写成

$$T_k = F_k + \sum_{j<k} \Phi_{kj}\, u_j ,$$

其中 Φ_{kj} 只由 a、b 决定，F_k 由常数项、$a^k(T_0-\bar T)$ 和负载项 $e\,(P_j-\bar P)$ 组成，是 T_0 和负载预报的一次函数。

以算例的代价 $J=\sum_k (T_k-70)^2+20\sum_k u_k^2$ 为例，对每个 u_j 求偏导数并令其为零，得到的方程左边是 $u_0,\dots,u_{N-1}$ 的一次式，系数只由 Φ 和权重决定，与 T_0 无关；右边是 F_k-70 的一次组合。所以方程组的解 U^* 是右边的一次函数，也就是 T_0 与负载预报的一次函数。其第一个分量可以写成

$$u_0^* = \kappa_0 + \kappa_T\,(T_0-\bar T) + \sum_j \kappa_j\,(P_j-\bar P),$$

系数 κ 都可以事先算好。第一项是常数，第二项是对温度的比例反馈，第三项是对负载预报的前馈。

用第 11 节的算例（$\bar T=72$，$\bar u=0.5$，负载不变）验证：T_0 取 70、72、74、76 时，解出的 u_0^* 依次为 0.246、0.326、0.406、0.486，每升高 2 K 增加 0.080，即

$$u_0^* = 0.326 + 0.040\,(T_0-72).$$

有约束时，若已知哪些约束起作用，就把它们当作等式加入方程组，解仍是 T_0 和预报的一次函数；起作用的约束不同，得到的一次函数也不同。所以有约束的线性 MPC 是一个分段线性的控制律，显式 MPC（第 22 节）正是把这些分段离线算好。

附录 J　带遗忘因子的最小二乘

数据满足 $z_i \approx \theta\,\varphi_i$，其中 $\theta = G_{\text{ref}}$ 为待估参数（式 (46)）。在时刻 k，选择 θ 使加权误差平方和最小：

$$S(\theta)=\sum_{i\le k}\lambda^{k-i}\big(z_i-\theta\,\varphi_i\big)^2 .$$

这是 θ 的二次函数（开口向上）。令导数为零：

$$S'(\theta)=-2\sum_{i\le k}\lambda^{k-i}\varphi_i\big(z_i-\theta\,\varphi_i\big)=0 \quad\Longrightarrow\quad \hat\theta=\frac{\sum_{i\le k}\lambda^{k-i}\varphi_i z_i}{\sum_{i\le k}\lambda^{k-i}\varphi_i^2}.$$

记分子为 α_k，分母为 β_k，则它们可以递推更新：

$$\beta_k=\lambda\beta_{k-1}+\varphi_k^2,\qquad \alpha_k=\lambda\alpha_{k-1}+\varphi_k z_k,\qquad \hat\theta_k=\alpha_k/\beta_k .$$

每个周期只需两次乘加和一次除法。λ 决定记忆长短：一个数据的权重经过 $\ln 0.5/\ln\lambda$ 步减半，$\lambda=0.97$ 时约为 23 步；有效记忆长度约为 $1/(1-\lambda)=33$ 步。初始时取 $\alpha_0=\theta_0\beta_0$，θ_0 为出厂值，β_0 表示对出厂值的信任程度。估计值每步限制在物理合理的范围内。

附录 K 神经网络与梯度下降

网络的计算。 式 (49) 的网络对输入 $\boldsymbol{x}=(\tilde T,\tilde u,\tilde P)$ 依次计算

$$s_j=\alpha_{j1}\tilde T+\alpha_{j2}\tilde u+\alpha_{j3}\tilde P+\beta_j,\qquad h_j=\tanh(s_j),\qquad r=w_0+\sum_{j=1}^{16}w_j h_j.$$

$\tanh(s)=\dfrac{e^{s}-e^{-s}}{e^{s}+e^{-s}}$，其导数为 $1-\tanh^2(s)$。输入先缩放为 $\tilde T=(T-\mu_T)/\sigma_T$ 等形式（减去训练数据的均值、除以标准差），使三个输入处在同一量级。

训练代价对参数的导数。 对一个数据点，误差平方为 $E=(e-r)^2$。由链式法则：

$$\frac{\partial E}{\partial w_j}=-2(e-r)\,h_j,\qquad \frac{\partial E}{\partial \alpha_{j1}}=-2(e-r)\,w_j\,(1-h_j^2)\,\tilde T,\qquad \frac{\partial E}{\partial \beta_j}=-2(e-r)\,w_j\,(1-h_j^2).$$

先算出输出处的误差 $-2(e-r)$，再一层层往输入方向乘上各层的导数，这就是“反向传播”。对所有数据点求和，得到 $\partial L/\partial\theta$。

梯度下降。 导数指向 L 增大最快的方向，沿反方向走一小步可以使 L 下降：$\theta\leftarrow\theta-\eta\,\partial L/\partial\theta$，$\eta$ 称为学习率。反复进行，直到 L 不再明显下降。实际中常用它的改进版本（如 Adam、拟牛顿法），本文的实验使用拟牛顿法（L-BFGS），思路与附录 E 的牛顿法相近。L 一般不是凸函数，可能停在不同的谷底，所以通常从几个不同的初始参数出发训练，选验证误差最小的结果。

网络对输入的导数。 线性化需要 $\partial r/\partial u$ 等导数，同样由链式法则得到：

$$\frac{\partial r}{\partial u}=\sum_{j=1}^{16}w_j\,(1-h_j^2)\,\alpha_{j2}\cdot\frac{1}{\sigma_u}.$$

深度学习框架中的“自动求导”会自动完成这类计算；也可以用有限差分 $\big[r(u+h)-r(u-h)\big]/(2h)$ 近似。

附录 L 仿真设置

- **被控对象**：方程 (4)，参数见表 1，以 0.5 s 步长数值积分，初始温度 78°C。
- **负载**：0 ~ 300 s 为 100 W，300 ~ 600 s 为 250 W，600 ~ 900 s 为 100 W。
- **控制周期**：所有控制器每 5 s 更新一次风量，风量限制在 $[0.2,\,1]$。
- **风扇曲线**：$u=0.2+0.8\,(T-60)/20$，并限幅。

- **PI**：$T_{\text{set}} = 76°\text{C}$，$K_p = 0.2$ /K，积分时间 150 s，基准风量 0.3，带抗积分饱和。参数取自网格搜索，选峰值温度不超过 78.5°C 时能耗最低的一组。
- **MPC**：$t_s = 5$ s，$N = 24$；代价为 $\sum u_k^3 + 20\sum(u_k - u_{k-1})^2$，约束 $T_k \le 78 + \varepsilon_k$，对松弛量 $\varepsilon_k \ge 0$ 施加大罚项 $1000\sum(\varepsilon_k + \varepsilon_k^2)$，以保证问题总有解；每个时刻在当前温度与上一次风量处重新线性化。“有预报”版本已知未来负载，“只用实测功耗”版本假设未来负载等于当前测得的功耗。
- **预报虚警**：被控对象负载始终为 100 W；300 s 之前，控制器收到“300 ∼ 600 s 为 250 W”的预报，300 s 之后改用实测功耗。
- **自适应**：被控对象 $G_{\text{ref}} = 0.85 \times 3.5 = 2.975$；MPC 只用实测功耗；自适应版本按附录 J 估计 G_{ref}，$\lambda = 0.97$，$\theta_0 = 3.5$，$\beta_0 = 2500$，估计值限制在 $[1, 6]$。
- **热风回流**：被控对象的实际进风温度为 $T_{\text{in}} + 0.5(1 - u)^2(T - T_{\text{in}})$，MPC 的物理公式不包含这一项。训练数据为 3000 个周期（$t_s = 5$ s），风量每 30 s 在 $[0.2, 1]$ 内随机取值，负载每 60 s 在 60、100、150、200、250 W 中随机取值；前 2400 个周期用于训练，后 600 个用于检验。网络为一层 16 个 tanh 单元，用 scikit-learn 的 MLPRegressor（L-BFGS，正则化系数 10^{-4}）训练。三种 MPC 都在每个周期用有限差分对各自的一步公式线性化，其余设置与上面的 MPC 相同。
- **能耗指标**：$\int u^3\,dt$。