澳大利亚尖峰电价的预测

黄含驰、杨悦然、周瀚旭、张钰

2019 年 6 月 7 日

【摘要】

在过去，澳洲电价是国家统一制定的。而自 20 世纪 90 年代，在政府放松管制和引入竞争性电力市场过程中，传统垄断电力部门的格局不断被重塑。开放电力市场在澳洲逐渐形成，并发展出以批发和零售市场为主的现货市场，之后的远期合约市场，及广泛参与、高度透明、具套期保值功能的期货市场。

在这样的市场中，零售商以不受监管的现货价格购买电力，以严格监管的价格向消费者出售。出现极端现货价格是零售商和公司的主要风险源，也是消费者所付电价的间接影响因素。若能做到对极端电价进行较准确的预测，对零售商灵活调整期货合同及事先应对极端事件、减少公司损失，有着重大意义。

传统的自回归时间序列模型，通过使用阈值来处理尖峰、Poisson 跳跃过程和多个重尾误差过程。现货价格的扩散模型引入了一个带恒定强度参数的 Poisson 跳跃分量或时变强度参数，其中跳跃过程的强度通常是季节和昼夜因素的线性组合。此外，亦有方法将价格波动分为跳跃和非跳转组件，然后探讨是否可以分析出两者的表现机制来改善波动率预测。

所有这些模型都将价格峰值视为有强度的无记忆过程，但这与现实存在显著差异。因此，文章采取了一种超高频数据建模时常用的 ACD 模型为基础，设计出改良版的 ACH 模型，成功实现极端电价的改进预测。

目录

1 背景

澳洲新闻网曾报道：随着电费持续上涨，许多家庭已面临高额债务，甚至不少未支付的燃气、电费就高达 1.8 万澳元 (7 万多元)！为了研究澳洲高额电价的形成原因，我们小组研读了一篇澳洲极端电价预测的论文，深入了解澳洲极端电价的形成机制和应对策略。

图 1.1: 全球各地区的电价

1.1 历史介绍

在 20 世纪 90 年代中期，新南威尔士州，昆士兰州，维多利亚州，南澳大利亚州和澳大利亚首都直辖区的区域电力市场合并成为澳大利亚国家电力市场（NEM）。NEM 作为一个集合市场运作，其中每个地区所有可用的电力供应被汇总，然后派遣发电机以便尽可能有效地满足需求。如果一个区域的当地需求超过当地供应，或邻近地区的电力输出价格足够便宜，则电力将在有输电设施的区域之间进口和输出。就供应方的构成而言，燃煤发电机和水力发电生产的边际生产成本较低，分别占 NEM 产能的 84%和 7.2%。燃气轮机和燃油电厂分别提供约 8.5%和 0.3%的市场，仅需约 20 分钟即可开始发电，但边际生产成本相对较高。

1.2 定价机制

批发电力交易作为现货市场进行，通过集中协调的调度过程即时匹配供需。现货价格的投标，发送和计算过程概述如下。在生产前一天下午 12:30 之前，发货人投标自己的供应曲线，包括第二天每半小时最多 10 个价格与对应数量，最低限价为-1,000 美元，最高限价为 10,000 美元兆瓦时 (MWh)。发货人可以在发货前发货前 5 分钟内，自由重新投标数量，但不能修改价格,。在收到所有发电机的投标后，汇总供应曲线并根据其出价调度发电机，以便尽可能便宜地满足需求。

2 ACD 模型

ACD(Autoregressive Conditional Duration) 自回归条件持续期模型：它对事件间的时间差进行建模，刻画了事件间会彼此影响的情况。因在股市等交易中良好的预测表现，ACD 获得了学术界认可。

2.1 持续期 (duration)

$$d_i = \psi_i \varepsilon_i, \varepsilon_i \propto \text{i.i.d}\ \ p(\varepsilon;\pi) \tag{2.1}$$

$$\psi_i = \omega + \sum_{j=1}^{p} \alpha_j d_{i-j} + \sum_{j=1}^{q} \beta_j \psi_{i-j} \tag{2.2}$$

其中，d_i 表示两次市场事件发生时间 τ_{t-1} 和 τ_t 之间的时间差，即持续期 (duration)。d_i 和 ψ_i 的关系为

$$E(d_t|I_{t-1}) = \psi_t \tag{2.3}$$

I_{t-1} 表示 τ_{t-1} 时刻的信息集。模型中的系数满足 $\alpha_j, \beta_j \le 0, \omega > 0$，并且

$$\sum_{j=1}^{\max(p,q)} (\alpha_j + \beta_j) \le 1 \tag{2.4}$$

$p(\varepsilon;\pi)$ 为 ε_i 的密度函数。

ε_t 通常被假设满足的分布形式有指数分布、Weibull 分布、Burr 分布等等。ε_t 服从指数分布的模型是 ACD 模型的基本形式。从分布角度可以讲 ACD 模型分为 WACD 模型（ε_t 服从 Weibull 分布），GGACD 模型（ε_t 服从广义 Gamma 分布），BACD 模型（ε_t 服从 Burr 分布）等。

2.2 条件密度函数

令 $N(t)$ 为 $(t_0, t]$ 区间发生的事件数。定义条件密度函数

$$\lambda(t|I_t) = \lim_{\Delta t \to 0^+} \frac{\text{Prob}(N(t+\Delta t) > N(T)|I_t)}{\Delta t} \tag{2.5}$$

再假设 ε_t 服从指数分布，则可证明

$$\lambda(t|I_t) = \frac{1}{\psi_{N(t)+1}} \tag{2.6}$$

3 ACH 模型

ACH(Autogressive Conditional Hazard) 自回归条件危害模型基于 ACD 模型加以改进。

3.1 定义条件危害

条件危害是指在 I_t 条件下给定区间的电价的极端电价事件发生的概率。

$$h_{t+1} = \text{Prob}(N(t+1) > N(t) | I_t) \tag{3.1}$$

Hamilton 和 Jorda(2002) 证明 ACH 模型作为 ACD 模型的离散版本，它的危害函数可以写成

$$h_{t+1} = \frac{1}{\psi_{N(t)+1}} \tag{3.2}$$

3.2 增加模型参数

将 $\psi_{N(t)+1}$ 改进为

$$\psi^{v}_{N(t)+1} = \omega^{*} + \sum_{j=1}^{p} \alpha_j d^{v}_{N(t)+1-j} + \sum_{j=1}^{q} \beta_j \psi^{v}_{N(t)+1-j} \tag{3.3}$$

通过不同参数的模型对预测结果的比较，选择最优参数，起到增加模型与现实贴合度的效果。

3.3 考虑外生变量

$$h_{t+1} = \frac{1}{\Lambda(e^{-\gamma' \mathbf{z}_{t+1}} + \psi_{N(t)+1})} \tag{3.4}$$

其中 Λ, γ 为参数，$\mathbf{z}$ 为外生变量，$\mathbf{z} = [1, x_1, x_2, x_3, x_4, x_5]'$，具体含义见下表。

变量	含义
x_1	负载
x_2	$t+1$ 时刻所在日最高气温与最近一年间平均气温的绝对偏差
x_3	$t+1$ 时刻所在日最低气温与最近一年间平均气温的绝对偏差
x_4	是否在工作日且温度属于 $[T_{min}, T_{max}]$
x_5	是否在非工作日

表 3.1: 外生变量对应含义

注 1: 这是 Logistic 回归的二分类问题常用的 Sigmoid 函数变体，输出映射在 $(0,1)$ 之间，易于求导，优化稳定。

注 2: 对于 x_i，$i = 1,2,3,4,5$ 需要进行归一化

如此，极端电价刻画模型建立完毕，由方程 3.3和 3.4组成，其中待求参数为

$$\theta = (\gamma, \alpha_1, \cdots, \alpha_p, \beta_1, \cdots, \beta_p, v) \tag{3.5}$$

参数需根据样本数据并最大化似然函数求解得出。

3.4 似然函数的构造

设随机变量

$$X(t)=\begin{cases}1 & \text{，当极端电价事件在 } t \text{ 区间发生}\\ 0 & \text{，当极端电价事件未在 } t \text{ 区间发生}\end{cases} \tag{3.6}$$

则 $X(t)$ 的条件概率密度满足

$$\text{Prob}(X_t = xt|I_{t-1};\theta) = h_t^{x_t}(1-h_t)^{1-x_t} \tag{3.7}$$

在含 T 个时间区间的样本中，似然函数为

$$\log L(\theta) = \sum_{t=1}^{T} x_t \log h_t + (1-x_t)\log(1-h_t) \tag{3.8}$$

通过 Hessel 矩阵求无条件极值来最大化似然函数，可以求得参数 θ

3.5 模型求解

通过对比似然函数，发现 $(p,q)=(1,1)$ 时，ACH 作用效果相对较佳，此时参数及似然函数结果如下

变量	参数	NSW	Qld	SA	Vic
常数	γ_0	-6.0563	-4.9352	-5.3748	-6.0496
		(0.1926)	(0.0328)	(0.0634)	(0.0533)
负载 $_t$	γ_1	2.5463	2.8329	1.8809	2.5013
		(0.1337)	(0.0392)	(0.0428)	(0.0129)
$T_{max,t}$	γ_2	0.1914	0.6075	-0.0210	0.3058
		(0.0225)	(0.0162)	(0.0093)	(0.0027)
$T_{min,t}$	γ_3	-0.0446	-0.0538	0.0898	-0.0375
		(0.0435)	(0.0178)	(0.0096)	(0.0057)
$u_{N(t)}$	α_1	0.0310	0.0141	0.0517	0.0298
		(0.0104)	(0.0007)	(0.0001)	(0.0007)
$\psi_N(t)$	β_1	0.9522	0.9474	0.9481	0.9553
		(0.0021)	(0.0002)	0.0000	(0.0001)
	υ	0.4888	1.3425	0.0216	0.6993
		(0.1672)	(0.0313)	(0.0004)	(0.0251)
对数概率		-7498.9	-8234.1	-8013.3	-5993.2

表 3.2: 最佳参数与对应似然函数值

4 模型预测与评估

4.1 预测

预测对象为澳大利亚四个地区在 2007/07/01 至 2007/09/30 每半小时电价。采用方程 3.3与3.4并使用表 3.5中的参数。采取提前半小时预测的方案，即根据第 t 时间段及之前的信息预测第 $t+1$ 时间段 (隔 0.5h) 的 h 值，最终得到总的似然函数。(注：由于 $t+1$ 时刻的负载难以预测，故代入去年同时段的负载。

4.2 评估

以 Logit Model 作为基准模型 (benchmark)

$$p_{t+1} = \frac{1}{1+e^{-\xi' z_{t+1}}} \tag{4.1}$$

绘制 h 分别在 ACH 和 Logit 模型下随时间的变动。

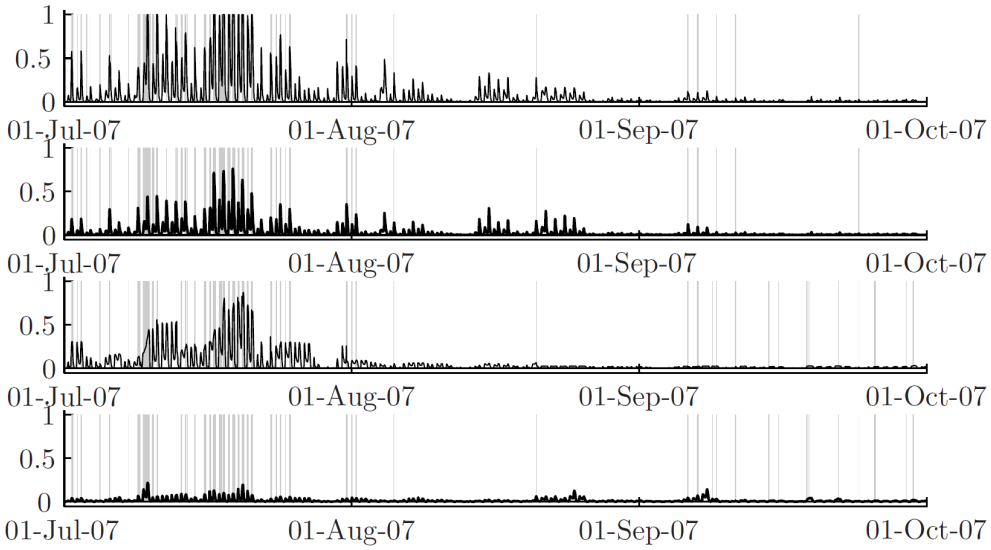

图 4.1: h 分别在 ACH 和 Logit 模型下随时间的变动

图为 ACH 预测与基准 Logit 预测，两者对紧随参数拟合月份后的三个月的电价情况进行预测。极端价格事件由灰色竖条表示。第一张图显示了新南威尔士州的 ACH 预测概率。第二张图显示了新南威尔士州的基准预测概率。第三张图显示了昆士兰州的 ACH 预测概率。第四张图显示了昆士兰州的基准预测概率。南澳大利亚和维多利亚州获得了类似的结果。可见，ACH 模型优于基准模型。

4.3 模型进一步修正：加入非对称度量

由图 4.2知，在原 ACH 模型的预测中，极端电价事件处 h 确实高于平常，但与 1 仍有不少差距。同时，似然函数大小与样本数目有关，故观察似然函数大小无法直接预估模型结果优劣。为了解决

上述两个问题，将目标函数设计为一种平均损失函数 LPSE(The log-probability score error)

$$\mathrm{LPSE} = -\frac{1}{t_1 - t_0 + 1}\sum_{t=t_0}^{t_1}(1 - x_t)\log(1 - h_t) + x_t \log h_t \tag{4.2}$$

并且加大对极端价格事件误判情况的惩罚力度，得到 LPSE 的非对称度量

$$\mathrm{Asym.} = \frac{1}{t_1 + t_0 + 1}\sum_{t=t_0}^{t_1}(x_t(1+\kappa) + (1 - x_t)(1-\kappa))|x_t - h_t| \tag{4.3}$$

实验时，若取 $k = 0.5$，则实验结果如下：

	NSW	Old	SA	Vic
使用 ACH 模型				
MAE	0.078	0.078	0.107	0.106
RMSE	0.191	0.193	0.3	0.234
LPSE	0.136	0.132	0.338	0.184
Asym.	0.073	0.08	0.149	0.108
使用 Logit 基准模型				
MAE	0.074	0.069	0.108	0.108
RMSE	0.216	0.235	0.307	0.303
LPSE	0.157	0.198	0.364	0.335
Asym.	0.092	0.092	0.153	0.151

表 4.1: 使用真实负载的误差

	NSW	Old	SA	Vic
使用 ACH 模型				
MAE	0.098	0.079	0.109	0.119
RMSE	0.205	0.196	0.287	0.233
LPSE	0.152	0.137	0.278	0.18
Asym.	0.08	0.083	0.145	0.115
使用 Logit 基准模型				
MAE	0.078	0.07	0.11	0.109
RMSE	0.216	0.232	0.305	0.301
LPSE	0.153	0.189	0.337	0.318
Asym.	0.094	0.092	0.153	0.151

表 4.2: 使用预期负载的误差

上表为预测极端电价事件的误差。第一个表使用的负载为真实负载，第二个表使用的是预期负载。在每个表中，前四行是 ACH 模型的误差值，后四行是 Logit 基准模型的误差值。

可见，无论外生变量代入计算的是实时负载还是去年用于预测的负载，非对称度量 LPSE 误差值在 ACH 模型及 Logit 模型中均小于原 LPSE，且 ACH 模型的非对称 LPSE 误差值在以上四个地区均小于 Logit 模型。因此，引入非对称度量的 ACH 模型确使预测得到改进。

4.4 模型验证

预测试验完成后，选择用于预测评估的三个月对应于电力市场中典期货合约的持续时间，设计一份简略的季度期货合约策略以修复极端电价时的经营损失。合约中对冲价格对实际现货价格平均值与分析师预测价格进行加权，模拟实验的结果如下表：

	NSW	Old	SA	Vic
	使用 ACH 模型			
利润率 (%)	21.47	8.02	0	24.22
	未使用 ACH 模型			
利润率 (%)	14.53	6.21	2.8	5.87

表 4.3: 利润率比较

除了在 SA 之外，这个简单模拟交易方案的结果 ACH 模型在极端电价事件预测中的实用性提供了有力佐证。

5 模型反思

5.1 优点

- **模拟现实**
  从思考如何刻画极端电价出发，根据现实改良已有模型。最终根据预测结果制定价格，并模拟交易，彰显非对称 ACH 模型的经济效益。
- **兼顾内外生变量**
  将内生变量与外生变量融于 sigmoid 函数的变种中，设计精密巧妙。

5.2 缺点

- **实时数据难获得**
  实时负载数据难以获得，且负载是外生变量中最难预测的。所以，简单地用去年数据带入，处理略显粗糙。

- **判断依据过于单一**
  因澳洲现货电价波动极大，单纯根据一个阈值来判定极端与否、对 ACH 模型仅引入一个常数 k 进行非对称度量，操作有些粗暴。

5.3 建议

不妨设定阶梯阈值来表示电价的不同极端程度，并在非对称度量中对不同阶梯的危害值赋予不同权重。

6 实验结果

以下是我们小组使用 Python 对部分模型代码的复现：

我们比较了 ACH(1,1) 模型与其相对于 Logit 基准模型在固定参数及对参数 α,β 网格搜索时的性能。(比较标准：相同参数下的对数似然函数值)

在具体操作时，我们对数据和模型做了以下简化：

1. 从 basics.mat 文件读取源数据，观察各外生变量的大致分布，在以下试验中据我们所观测分布随机生成一定范围的负载、温度数据 (我们将外生变量凝结成了两大类：负载和温度，所以外生变量的向量 $\mathbf{z}=[1,z_1,z_2]$
2. 我们将下图的 λ 设置为 1。这其实并不科学，但此处是我们模拟实验的一个瓶颈。我们没有理解作者是如何调整 λ 值的。如有机会，我们很乐意向原作者发 email 讨论这个小问题。

6.1 对方程 (3.4) 建立函数

```
#ACH(1,1),lambda=1
import numpy as np

def psi(v,a,b,dPrevious,psiPrevious):#a=alpha,b=beta
    return (a*dPrevious**v+b*psiPrevious**v)**(1/v)

def h(gamma0,gamma1,gamma2,z1,z2,psi):
    import math
    tmp=gamma0+gamma1*z1+gamma2*z2
    fenmu=math.exp(-tmp)+psi
    return 1.0/fenmu

def ACHlogfunc(gamma0,gamma1,gamma2,load,temp,v,a,b,delta,spikeCondition):
    psiList=[5]*1440
    hdata=h(gamma0,gamma1,gamma2,load[0],temp[0],psiList[0])
    hlist=[hdata]
    logvalue=(1-spikeCondition[0])*(1-hdata)+(spikeCondition[0])*(hdata)
    for i in range(1,len(spikeCondition)):
        psiList[i]=(a*(delta[i-1])**v+b*(psiList[i-1])**v)**(1/v)
```

```
        hdata=h(gamma0,gamma1,gamma2,load[i],temp[i],psiList[i])
        hlist.append(hdata)
        tmp=(1-spikeCondition[i])*np.log(1-hdata)+spikeCondition[i]*np.log(hdata)
        logvalue+=tmp
    return logvalue,hlist

def benchmarklogfunc(gamma0,gamma1,gamma2,load,temp,v,a,b,delta,spikeCondition):
    psiList=[1]*1440
    hdata=h(gamma0,gamma1,gamma2,load[0],temp[0],psiList[0])
    hlist=[hdata]
    logvalue=(1-spikeCondition[0])*(1-hdata)+(spikeCondition[0])*(hdata)
    for i in range(1,len(spikeCondition)):
        hdata=h(gamma0,gamma1,gamma2,load[i],temp[i],psiList[i])
        hlist.append(hdata)
        tmp=(1-spikeCondition[i])*np.log(1-hdata)+spikeCondition[i]*np.log(hdata)
        logvalue+=tmp
    return logvalue,hli
```

6.2 生成一个月数据用来模拟

生成一个月数据用来模拟 (30*48=1440),48 表示每天隔半个小时取一次样本，共取 48 次。生成方式：随机生成 $[0,10]$ 区间的整数，设定规则：$>=8$ 为极端事件发生 (spikeCondition=1)，否则未发生 (spikeCondition=0))，这样保证极端事件发生概率将较小。

```
import random

spikeCondition=[]
for i in range(1440):
    spikeCondition.append(int(random.randint(0,10)>=8))

#生成温度数据（观察basic.mat实际数据的大体分布后，决定如下模拟）
temp=[]
for i in range(1440):
    temp.append(random.randrange(10,40))

#生成负载数据（观察basic.mat实际数据的大体分布后，决定如下模拟）
load=[]
for i in range(1440):
    load.append(random.randrange(1000,10000))

#将负载和温度数据归一化
max_min_scaler = lambda x : (x-np.min(x))/(np.max(x)-np.min(x))
temp=max_min_scaler(temp)
load=max_min_scaler(load)
#根据以上数据求出极端事件发生时间间隔数列delta

#计算发生极端事件的时间点
spikeindex=[]
for i in range(1440):
    if spikeCondition[i]==1:
        spikeindex.append(i)
#计算时间间隔
tmp=[spikeindex[0]]
```

```
for i in range(len(spikeindex)-1):
    tmp.append(spikeindex[i+1]-spikeindex[i])
#构造时间间隔序列
delta=[0]*spikeindex[0]
for i in range(len(tmp)):
    if i!=(len(tmp)-1):
        for j in range(tmp[i+1]):
            delta.append(tmp[i])
    else:
        for j in range(tmp[i]):
            delta.append(tmp[i])
if len(delta)<1440:
    end=delta[-1]
    for i in range(1440-len(delta)):
        delta.append(end)
```

具体结果详见附录

6.3 对数似然函数

首先计算在给定参数下，ACH(1,1) 和 Logit 基准模型的对数似然函数值

```
ACHlogfunc(0.1,0.7,-0.2,load,temp,1.0,0.5,0.4,delta,spikeCondition)[0]
benchmarklogfunc(0.1,0.7,-0.2,load,temp,1.0,0.5,0.4,delta,spikeCondition)[0]
```

得到

$$\log L_{ACH}(\theta) = -904.9864927114387 \tag{6.1}$$

$$\log L_{Logit}(\theta) = -1150.1095604437858 \tag{6.2}$$

其次，在对 α,β 进行网格搜索时，ACH(1,1) 和 Logit 基准模型的对数似然函数值

```
for a in np.arange(0.1,0.9,0.1):
    for b in np.arange(0.1,1-a,0.1):
        print('a=',a,'b=',b)
        print(ACHlogfunc(0.1,0.7,-0.2,load,temp,1.0,a,b,delta,spikeCondition)[0],benchmarklogfunc
            (0.1,0.7,-0.2,load,temp,1.0,a,b,delta,spikeCondition)[0])
```

$\alpha\backslash\beta$	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
0.1	NA	NA	NA	NA	NA	NA	NA	-947
0.2	NA	NA	NA	NA	NA	-1012	-898	
0.3	NA	NA	NA	NA	-966	-895		
0.4	NA	NA	-1028	-952	-899			
0.5	NA	NA	-947	-905				
0.6	NA	NA	-911	-887				
0.7	NA	NA						
0.8	NA							

表 6.1: ACH 模型中不同 α 和 β 下的对数似然函数

由于不知如何对表达式中的 lambda 归一化 (论文中一笔带过了)，ACH(1,1) 的对数似然函数会时不时出现 nan 值，这也是我们实现模型时遇到的瓶颈。实验中，当 ACH(1,1) 的对数似然函数不为 NA 时，其对数似然函数值皆大于对应的 Logit 基准模型的对数似然函数值。

6.4 预测结果

抽取 1440 个样本中前 80 个进行绘图，比较 ACH 与 Logit 基准模型预测的 h 和真实结果的差异。

```
hOfACH=ACHlogfunc(0.1,0.7,-0.2,load,temp,1.0,0.5,0.4,delta,spikeCondition)[1]
hOfbench=benchmarklogfunc(0.1,0.7,-0.2,load,temp,1.0,0.5,0.4,delta,spikeCondition)[1]
x=list(range(80))
plt.scatter(x,spikeCondition[:80],color='green',label='real spike condition')
plt.scatter(x,hOfbench[:80],color='yellow',label='benchmark prediction')
plt.scatter(x,hOfACH[:80],color='red',label='ACHmodel prediction')
plt.legend()
plt.show()
```

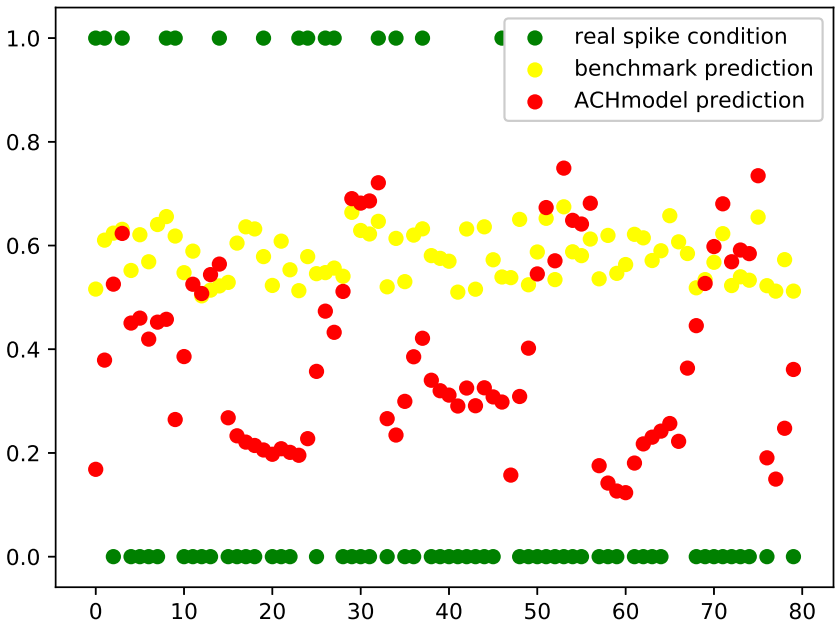

图 6.1: ACH 模型、Logit 模型与真实结果

6.5 结果分析

由于我们的实验将论文模型进行了简化，数据是依照分布按一定规律随机生成的（并非真实数据），且只考虑的一个月的情况，从而未考虑季节对负载、温度变化的影响。表达式中 λ 归一化问题也未解决，模型结果不甚理想也是意料之中。不过，从上图我们还是可以观察出一些迹象：Logit 基准模型预测的 h 值几乎都在一条水平线附近波动，并无区分性，其预测效果也就存疑。而 ACH 模型预测效果虽也欠佳，却在相对高点时和真实极端电价事件的发生恰好吻合，这一点比基准模型更具预测性能上的说服力。

7 参考文献

- <Forecasting Spikes in Electricity Prices> T M Christensen,A S Hurn,K A Lindsay, 2011

附录

spikeCondition

```
[0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0,
0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0,
0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0,
1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0,
0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0,
0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0,
0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,
1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1,
1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0,
0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1,
0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1,
0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0,
1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0,
0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0,
0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0,
1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0,
0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1,
1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0,
0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0,
1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1,
0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1,
0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0,
0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0,
1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1,
0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0,
1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1,
0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1,
0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1,
```

0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1,
0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1,
0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0,
1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1,
0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0,
1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0,
0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1,
0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0,
0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,
0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
1, 0, 0, 0, 0, 0, 0, 0, 0]

delta

[0, 0, 2, 2, 2, 3, 1, 1, 1, 1, 4, 1, 1, 1, 1, 4, 4, 4, 4, 4, 4, 6, 6, 6, 3, 1, 1,
1, 1, 4, 4, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 2, 1, 1, 1, 1, 1, 4, 4, 4, 4, 4, 1, 1,
1, 1, 1, 5, 1, 1, 1, 3, 3, 3, 3, 4, 4, 4, 3, 1, 1, 2, 2, 2, 2, 2, 2, 6, 6, 6, 6,
4, 4, 4, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 7, 7, 2, 2, 2, 2, 2, 2, 2, 2, 6, 1, 1,
1, 3, 1, 1, 2, 1, 1, 1, 2, 2, 2, 2, 4, 1, 1, 1, 3, 3, 3, 3, 4, 1, 1, 1, 1, 1, 1,
1, 4, 4, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 10, 10, 10, 3, 3,
3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 13, 13, 13, 13, 13, 13, 13, 13, 13, 9, 9, 9, 3,
3, 2, 2, 2, 2, 2, 2, 6, 6, 6, 6, 4, 1, 1, 2, 2, 2, 3, 3, 3, 3, 3, 5, 5, 5, 5, 5,
5, 5, 5, 5, 4, 4, 4, 4, 4, 4, 4, 7, 7, 2, 1, 1, 2, 2, 2, 2, 2, 2, 6, 1, 1, 1, 1,
1, 1, 1, 6, 1, 1, 1, 1, 4, 4, 4, 3, 1, 1, 1, 1, 1, 1, 6, 6, 6, 6, 6, 6, 6, 6, 6,
9, 9, 9, 9, 9, 5, 1, 1, 1, 3, 3, 3, 3, 3, 3, 3, 1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 3,
3, 3, 3, 3, 3, 3, 3, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 1, 1, 2, 2, 2, 2,
2, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 14, 14, 14, 14, 14, 14, 14, 14, 14,
9, 9, 9, 9, 9, 5, 5, 5, 5, 4, 4, 2, 1, 1, 1, 1, 4, 4, 4, 4, 4, 5, 5, 5, 3, 3, 2,
1, 1, 1, 1, 4, 1, 1, 1, 1, 4, 4, 4, 3, 3, 3, 3, 4, 4, 2, 2, 2, 3, 1, 1, 1, 1, 3,
3, 2, 2, 2, 2, 4, 4, 4, 3, 3, 2, 2, 2, 1, 1, 1, 1, 4, 4, 2, 2, 2, 2, 2, 2, 2, 7,
7, 2, 2, 2, 2, 2, 2, 6, 1, 1, 1, 1, 1, 1, 1, 1, 1, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9,
9, 9, 9, 9, 9, 15, 1, 1, 2, 2, 2, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 12, 12,
12, 3, 3, 2, 1, 1, 2, 1, 1, 1, 1, 4, 4, 4, 3, 3, 3, 3, 3, 3, 3, 7, 7, 2, 1, 1, 1,
1, 4, 1, 1, 1, 1, 1, 4, 4, 4, 4, 4, 4, 4, 4, 4, 9, 1, 1, 1, 1, 1, 1, 4, 4, 4, 4,
4, 5, 1, 1, 1, 1, 1, 1, 1, 1, 1, 9, 9, 2, 2, 2, 2, 2, 5, 5, 5, 5, 5, 5, 5, 5, 8,
8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 14, 14, 2, 2, 2, 2, 2, 2, 4, 1, 1, 1, 3,
3, 2, 1, 1, 2, 2, 2, 2, 2, 5, 5, 2, 1, 1, 1, 1, 1, 1, 1, 7, 7, 7, 7, 4, 4, 2, 2,

2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 18, 18, 2, 2, 2, 3, 3, 3, 3, 3, 3, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 10, 10, 10, 10, 10, 5, 5, 2, 2, 2, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 8, 8, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 13, 13, 2, 1, 1, 1, 1, 4, 4, 4, 4, 4, 5, 5, 5, 3, 3, 3, 3, 3, 3, 3, 3, 8, 1, 1, 1, 1, 3, 3, 2, 2, 2, 2, 4, 4, 4, 4, 4, 4, 4, 4, 4, 9, 9, 9, 9, 4, 4, 4, 4, 4, 5, 5, 5, 5, 4, 4, 4, 4, 4, 5, 5, 5, 3, 1, 1, 1, 1, 1, 1, 4, 4, 4, 4, 4, 4, 6, 6, 2, 2, 2, 2, 2, 2, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 12, 12, 12, 12, 12, 12, 12, 7, 7, 7, 7, 4, 4, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 13, 13, 13, 13, 13, 13, 13, 13, 8, 8, 2, 1, 1, 1, 1, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 13, 13, 13, 13, 4, 4, 4, 3, 3, 3, 3, 3, 3, 6, 1, 1, 2, 1, 1, 1, 3, 3, 2, 2, 2, 3, 3, 3, 3, 3, 2, 2, 2, 2, 2, 5, 5, 5, 3, 3, 2, 1, 1, 1, 1, 4, 4, 4, 3, 3, 3, 3, 3, 5, 1, 1, 1, 1, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 13, 1, 1, 1, 3, 3, 2, 2, 2, 3, 3, 3, 3, 4, 4, 4, 3, 3, 3, 3, 3, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 12, 12, 12, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 11, 1, 1, 1, 1, 1, 1, 1, 1, 1, 8, 8, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 10, 10, 2, 2, 2, 2, 2, 2, 2, 2, 2, 9, 9, 2, 2, 2, 2, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5, 5, 8, 8, 8, 8, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 13, 13, 13, 13, 13, 5, 5, 2, 2, 2, 1, 1, 1, 3, 1, 1, 1, 2, 1, 1, 1, 1, 3, 1, 1, 1, 1, 1, 1, 1, 7, 7, 7, 7, 7, 7, 7, 7, 8, 8, 2, 2, 2, 2, 2, 2, 6, 6, 6, 3, 1, 1, 1, 1, 1, 1, 1, 1, 7, 1, 1, 2, 1, 1, 1, 3, 3, 3, 3, 3, 3, 3, 3, 3, 6, 1, 1, 1, 1, 1, 1, 1, 1, 1, 9, 9, 2, 2, 2, 3, 3, 3, 3, 3, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 15, 1, 1, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 7, 7, 7, 7, 7, 7, 7, 7, 1, 1, 1, 1, 1, 1, 6, 6, 6, 3, 3, 3, 3, 3, 5, 5, 5, 5, 4, 4, 2, 2, 2, 3, 1, 1, 1, 1, 1, 1, 1, 7, 7, 2, 2, 2, 2, 2, 2, 6, 1, 1, 1, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 12, 1, 1, 2, 2, 2, 3, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 2, 1, 1, 1, 3, 1, 1, 1, 1, 1, 1, 1, 7, 7, 7, 7, 7, 5, 1, 1, 1, 2, 2, 2, 2, 4, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 3, 1, 1, 1, 1, 1, 1, 1, 5, 5, 2, 1, 1, 1, 1, 1, 5, 5, 5, 3, 1, 1, 1, 1, 1, 4, 4, 2, 1, 1, 1, 1, 1, 3, 1, 1, 1, 1, 1, 5, 5, 2, 1, 1, 1, 1, 1, 1, 1, 6, 1, 1, 1, 1, 3, 3, 3, 3, 3, 5, 5, 5, 5, 5, 5, 5, 5, 8, 1, 1, 2, 1, 1, 1, 1, 4, 4, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 10, 10, 10, 3, 3, 3, 3, 3, 3, 3, 3, 3]