基于最小二乘法的磷酸铁锂电池 SOC 估计

孙传余，肖林京，于志豪，李洪宇，朱绪力
（山东科技大学 机械电子工程学院，山东 青岛 266590）

摘　要：从工程应用角度出发，将磷酸铁锂电池状态分为静置态、工作态和维护态。静置态下采用开路电压法，查询开路电压-荷电状态（*OCV-SOC*）关系曲线，获取 SOC 初值；工作态下采用安时（Ah）积分法估计 SOC 的变化量；给出判断磷酸铁锂电池进入维护态的三个条件，并在维护态下，通过最小二乘算法重新拟合 *OCV-SOC* 关系曲线，以纠正环境温度、电流、内阻、循环次数等变化的影响，避免了复杂算法和模型分析，提高了 SOC 估计精度。通过实验绘制了基于最小二乘算法的 *OCV-SOC* 变化曲线以及 *SOC-T* 曲线，进行误差和数据分析，给出最小二乘算法的 C 语言矩阵源码。结果表明，该 SOC 估计法不受电池单体、数学模型和参数变化的影响，简单易行，判断可靠，可应用在各种锂电池管理系统中。

关键词：磷酸铁锂电池；*OCV-SOC* 曲线；安时积分法；最小二乘算法

中图分类号：TM910.1　　**文献标志码**：A　　**文章编号**：1672-3767(2013)02-0094-09

SOC Estimation of $LiFePO_4$ Battery Based on the Least Squares Algorithm

Sun Chuanyu, Xiao Linjing, Yu Zhihao, Li Hongyu, Zhu Xuli
(College of Mechanical and Electronic Engineering, Shandong University of Science and Technology, Qingdao, Shandong 266590, China)

Abstract: From the perspective of engineering application, the $LiFePO_4$ battery was divided into static state, working state and maintenance state. Under the static state, open circuit voltage (OCV) method was used to obtain the initial value of state of charge (SOC) through inquiring the *OCV-SOC* curves; under the working state, Ah integral method was used to estimate the variation of SOC; three conditions were given to judge whether the $LiFePO_4$ battery should go into the maintenance state or not, and under the maintenance state, the least squares algorithm was used to refit the *OCV-SOC* curves, so as to correct the influence of the environment temperature, current, resistance, cycle times etc., which can avoid the complex algorithm and model analysis, and also improve the SOC estimation precision. By experiment, the *OCV-SOC* relation curves based on the least squares algorithm and *SOC-T* relation curves were drawn, the error and data analysis made, and the C language source code of least squares matrix algorithm given. The results show that the SOC estimation of the three states, which is simple, feasible and reliable without being influenced by single battery, mathematics model and parameter variation, can be used in all kinds of lithium battery management system.

Key words: $LiFePO_4$ battery; *OCV-SOC* curves; Amper-hour integral method; least squares algorithm

根据《PNGV 电池试验手册》的标准电池模型，20 ℃时锂电池内部欧姆内阻和极化电阻在 1 mΩ 和 0.3 mΩ 附近波动[1]，对于磷酸铁锂电池车辆所用的 100 Ah 容量的动力电池而言，在 1 C 充放电情况下开路电压（open circuit voltage，OCV）和工作电压之间存在 0.1～0.3 V 的偏差，在荷电状态（state of charge，SOC）为

收稿日期：2012-10-15
基金项目：山东省科技发展计划项目（2012GSF11606）；青岛市科技计划项目（124131gx）
作者简介：孙传余（1982—），男，山东日照人，讲师，博士，主要从事嵌入式系统及锂电池管理方面的研究.
E-mail: suncy2011@yeah.net

10%时偏差可达 0.4 V[2]。由于磷酸铁锂电池的 *OCV-SOC* 曲线比较平坦，较小的电压变化就能引起显著的 SOC 改变，仅靠测量工作电压是不能可靠估计 SOC 的，而且由于电池内阻在充放电过程中所产生的电压降，往往会出现放电时显示电量已很低，但只要停止放电显示电量马上回升的现象（虽然停止放电会产生电荷集聚，有电压回升现象，但这一过程变化本身是缓慢的）。

在锂电池常用的五种等效电路模型中[3]，每种模型都可列写状态方程，通过 HPPC（hybrid pulse power characteristic，混合动力脉冲能力特性）测试过程可确定状态方程中各参数在不同 SOC 或 OCV 的值[1]，进而可通过工作电压和工作电流求得开路电压，再由 *OCV-SOC* 曲线完成 SOC 估计，但是 HPPC 测试过程在现场很难做到，且等效电路模型受各种不确定因素的影响较大。

神经网络[4]、卡尔曼滤波、模糊预测等算法均需要大量的数据样本，虽然有较高的预测精度，但运算量大，表达式复杂，对数据样本建立起的电池模型有依赖性，随着电池老化或环境改变，需重新采集各种情况下的数据样本，实现起来也比较困难。若依靠神经网络在线训练修正网络模型，依靠动态数据更新卡尔曼滤波处理方程及观测方程[5]，依靠历史数据重建模糊推理状态空间和预测空间[6]，则对处理器的性能要求较高。

磷酸铁锂电池 SOC 的估计一直是工程应用领域的一个难题，为了实现更准确的电量显示，将 OCV 法和安时（Ampere hour，Ah）积分法应用到工程实践中，在静置态时采用 OCV 法，在工作态时采用 Ah 积分法，然而，*OCV-SOC* 关系曲线受环境温度、放电电流、电池内阻以及循环次数的交互影响会发生变化，当电池满足进入维护态的条件时，需重新对 *OCV-SOC* 关系曲线进行标定，以提高 SOC 估计精度。

1 磷酸铁锂电池 SOC 估计

磷酸铁锂电池 SOC 的估计需满足假设条件：①磷酸铁锂电池充电环境和放电环境温度基本相同；②通过静置 1 h 后测得数据，可消除充电 *OCV-SOC* 曲线与放电 *OCV-SOC* 曲线的差异；③通过静置 1 h 后测得数据，可消除不同充电与放电倍率对测量 OCV 的影响。

1.1 静置态开路电压法

磷酸铁锂电池是一种环保高效的动力电池，用磷酸铁锂作为正极材料，石墨碳棒作为负极材料，电解质作为中间材料，其内部充电、放电化学反应方程式为：

$$\begin{aligned}&\mathrm{LiFePO_4}-x\mathrm{Li^+}-x\mathrm{e}\xrightarrow{\text{Charge}}x\mathrm{FePO_4}+(1-x)\mathrm{LiFePO_4}\\&\mathrm{FePO_4}+x\mathrm{e}+x\mathrm{Li^+}\xrightarrow{\text{Discharge}}x\mathrm{LiFePO_4}+(1-x)\mathrm{FePO_4}\end{aligned}\tag{1}$$

当反应达到平衡时，电池开路电压和电池容量之间有一定的对应关系，在静置态下通过测量开路电压，然后查询 *OCV-SOC* 关系曲线，即可获得 SOC 的值[7]。如图 1 所示，磷酸铁锂电池 *OCV-SOC* 曲线比较平坦，较小的 OCV 变化即可引起较大的 SOC 变化，查询精度低。为了提高 SOC 查询的准确性，可将曲线进行分段法处理，当 *SOC*≤10%时，采用线性处理，当 *SOC*>10%时，采用最小二乘法拟合。

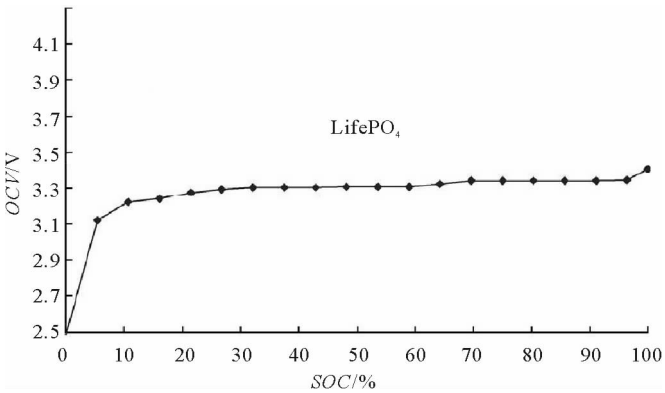

图 1 磷酸铁锂电池 *OCV-SOC* 曲线图

Fig. 1 *OCV-SOC* curves of $LiFePO_4$ battery

1.2 工作态 Ah 积分法

工作态就是磷酸铁锂电池正常充放电的状态，由于在工作状态下获取模型参数和开路电压都比较困难，而且现场环境复杂多变，负载可能不断变化，放电电流也不恒定，比较可靠的办法是采用 Ah 积分法，积分公式为

$$SOC_{K+1}=SOC_K+\eta\times i\times\Delta T/C_N; \tag{2}$$

$$\eta=U_L/U_{oc}。 \tag{3}$$

其中：η—电池效率系数；i—电池放电电流，A；ΔT—采用时间间隔，S；C_N—电池额定容量，A·S；U_L—工作电压，V；U_{oc}—工作前的开路电压，V。

新电池内阻较小，效率系数 η 近似等于 1，随着电池老化，系数 η 的计算存有误差，电流积分值 $i\times\Delta T$ 亦有误差，为避免误差的累积，Ah 积分法的初值定义如下：锂电池静置时间≥1 h 时，Ah 积分法的初值由 *OCV-SOC* 曲线查得；锂电池静置时间<1 h 时，Ah 积分法的初值等于上次 Ah 积分法的终值。

1.3 维护态最小二乘法

维护态就是电池的一种特殊充电状态，与正常充电操作完全相同，只是充电时间长，存在静置 1 h 过程和数据记录操作，可由电池管理系统自动识别并进入该状态。判断电池进入维护态的三个条件如下：

1）用 SOC_{K+1} 表示电池在工作态结束时的积分电池容量信息，若电池静置时间<1 h，则不进行判断，若电池静置时间≥1 h，测量 *OCV*，并查询 *OCV-SOC* 曲线，获得电池容量信息 SOC'_{K+1}，如果式(4)成立，则进入维护态[7]：

$$P_1=\frac{SOC_{K+1}-SOC'_{K+1}}{SOC_{K+1}}\geqslant 10\%。 \tag{4}$$

2）根据化学反应的热力学 Nernst 方程[8]，电池电动势（开路电压）受温度的影响；电池荷电状态 SOC 本身也受温度变化的影响[9]，可认为温度变化超过 10℃时也要进入维护态，即：

$$P_2=T-T'\geqslant 10℃ \tag{5}$$

其中：T—静置态下对应某一 *SOC* 时的温度，℃；T'—维护态下记录这一 *SOC* 时的温度，℃。

3）磷酸铁锂电池循环使用次数 $N\geqslant 100$ 时，需进入维护态，也可人工干预不定期进行维护。

由于现场多采用恒流恒压式充电机，因此维护态下的数据测量过程为：先将电池放电至 *SOC* 低于 10%，然后恒流充电，当出现电压突变时，停止充电并静置 1 h，记录 *OCV*、测量温度 T 并标定 *SOC* 为 10%，再继续恒流充电，*SOC* 状态每增加一定程度，就停止充电并静置 1h，记录此时的 *OCV* 和测量温度 T，如此重复，直至充电电流为零，记录最终 *OCV*、测量温度 T 和累积充入的总电量。

维护态下的数据处理过程为：①根据测量的开路电压和电池容量数据，采用折线法得到 $SOC\leqslant 10\%$ 时的线性方程，采用最小二乘法，拟合 $SOC>10\%$ 时的 *OCV-SOC* 变化曲线；②根据测得的电池容量和温度数据，分段线性处理，得到 *SOC-T* 关系曲线，进而可查询对应某一 *SOC* 值时的温度 T，但不能反过来根据温度 T 确定 *SOC* 值，因为这些折线段上 *SOC* 的变化主要由充放电电流造成的，而不是温度变化造成的，温度只是该状态下的一个微影响作用参数，只具有对应关系而无决定关系；③根据数据综合分析，给出电池健康状况 SOH（state of health）评价，判断电池故障并报警。

2 等精度最小二乘矩阵算法

2.1 最小二乘法计算过程

变量 y 代表磷酸铁锂电池 *SOC* 值，变量 x 代表磷酸铁锂电池的开路电压，建立一元非线性回归模型[10]：

$$y=b_0+b_1x+b_2x^2+\cdots+b_px^p。 \tag{6}$$

令 $x_1=x$，$x_2=x^2$，…，$x_p=x^p$，通过变量代换，得到多元回归数学模型[10]：

$$y=b_0+b_1x_1+b_2x_2+\cdots+b_px_p。 \tag{7}$$

由此列写开路电压矩阵 $\boldsymbol{X}$，荷电状态 SOC 矩阵 $\boldsymbol{Y}$，最小二乘估计系数矩阵 $\hat{\boldsymbol{B}}$：

$$\boldsymbol{X}=\begin{bmatrix}1 & x_{11} & x_{12} & \cdots & x_{1p}\\ 1 & x_{21} & x_{22} & \cdots & x_{2p}\\ \vdots & & & & \\ 1 & x_{n1} & x_{n2} & \cdots & x_{np}\end{bmatrix};\quad \boldsymbol{Y}=\begin{bmatrix}y_1\\ y_2\\ \vdots\\ y_n\end{bmatrix};\quad \hat{\boldsymbol{B}}=\begin{bmatrix}b_0\\ b_1\\ \vdots\\ b_p\end{bmatrix}。 \tag{8}$$

其中：x_{np}—第 n 次测量开路电压的 p 次幂计算值；y_n—第 n 次测量的 SOC 值。

根据测量结果残余误差平方和最小的原则，列出等精度测量正规方程[11]，若行列式 $|\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}|\neq 0$，可求得最小二乘法估计系数：

$$\hat{\boldsymbol{B}}=(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{Y}。\tag{9}$$

2.2 最小二乘算法精度分析

根据扩展贝塞尔公式，测量数据的标准差计算式为[11]：

$$\sigma=\sqrt{\frac{\sum_{i=1}^{N}(y_i-\hat{y}_i)^2}{N-t}}。\tag{10}$$

其中：y_i—磷酸铁锂电池第 i 个 SOC 实测值；$\hat{y}_i$—采用最小二乘法计算得到的第 i 个 SOC 估计值；N 为数据测量次数；t 为待估计变量的个数。

若定义不定乘数矩阵为

$$\boldsymbol{D}=\begin{bmatrix}d_{11} & d_{12} & \cdots & d_{1t}\\ d_{21} & d_{22} & \cdots & d_{2t}\\ & & \vdots & \\ d_{t1} & d_{t2} & \cdots & d_{tt}\end{bmatrix}=(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})^{-1},\tag{11}$$

结合式(10)和式(11)，可得估计量 $b_0,b_1,\cdots,b_p$ 的精度为

$$\left.\begin{aligned}\sigma_{b_0}&=\sigma\sqrt{d_{11}}\\ \sigma_{b_1}&=\sigma\sqrt{d_{22}}\\ &\vdots\\ \sigma_{b_p}&=\sigma\sqrt{d_{tt}}\end{aligned}\right\}。\tag{12}$$

2.3 最小二乘矩阵算法 C 程序

由于电池管理系统多采用 C 语言开发，实现上述算法就需要矩阵运算的 C 语言程序，目前在各类开放源代码的矩阵库中，Meschach 库可实现 C 代码的矩阵和向量运算，Cooperware Matrix 库可用于 C^{++} 代码编写，Blitz 库为 C^{++} 提供整数、浮点数和复数的 N 维数组运算，但是这些基于矩阵库的函数代码量太大，而电池管理系统 CPU 的内存区和程序空间非常有限，实现全功能代码移植是不可能的。

为此进行代码裁剪，去掉对象属性和类封装，只保留实数矩阵的加法、减法、乘法、转置和求逆运算，重新编写代码[12-13]，程序中使用指令 float * MatrixA = new float[N×N]创建 N 维方阵浮点数组，使用 MatrixA[i×N+j]或 *(MatrixA+i×N+j)访问数组中的第 i 行 j 列元素，最终生成 1 个头文件和 1 个源文件，其中 MyMatrix.h 头文件参见附录一，MyMatrix.C 源文件参见附录二。移植时只需将这 2 个文件添加到 C 或 C^{++} 工程中，并通过指令 #include "MyMatrix.h"包含头文件，然后就可以直接调用矩阵运算函数，从而完成最小二乘矩阵算法。

3 实验分析

3.1 实验条件

在性能完善的磷酸铁锂电池管理系统中，一般都能测量电池的电压、电流和温度，由于 SOC 的估计与系统的硬件无关，可通过图 2 所示实验电路进行数据采集。该测量电路由 PT100 温度变送器、HALL 电流传感器、USB5935 数据采集卡、监控计算机、恒流恒压充电机、可调功率负载、2 个 60Ah 串联磷酸铁锂电池及可控功率管等部分组成。当 PORT A 连接至 PORT B 时构成放电回路，当 PORT A 连接至 PORT C 时构成充电回路。

为了自动进行数据测量和分析，并实现锂电池的欠压保护、过压保护、过流保护、超温保护等功能，采用 VC^{++} 语言开发了一套基于 USB5935 采集卡的控制管理软件[14]，界面如图 3 所示，可同时对串联的两节锂

电池进行信息检测，可设置保护参数的数值，选择静置态、工作态和维护态下的数据采集方式，而采集到的数据则显示在左侧框中。

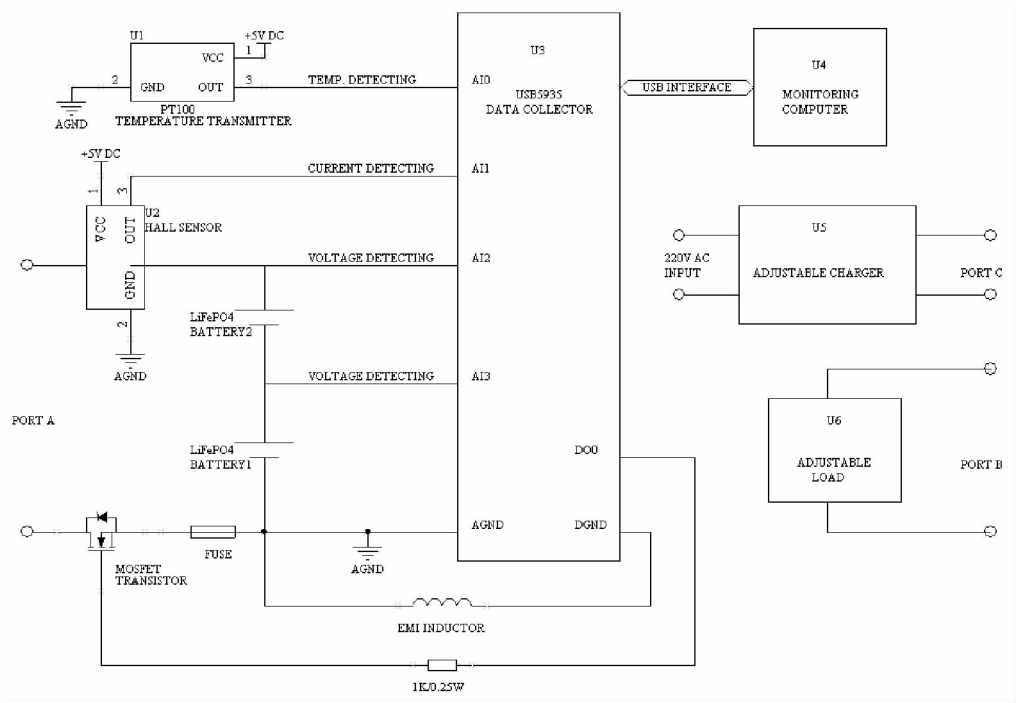

图 2　实验电路图

Fig. 2　Circuit diagram of experiment

图 3　控制管理软件界面图

Fig. 3　Software of control management

3.2 实验结果和分析

1)磷酸铁锂电池充电实验

在维护态下进行 2 节串联磷酸铁锂电池的充电实验,为了减少维护态充电的时间,*SOC* 每变化 9.48%,就静置 1 h 并记录下测量结果,在一次充电过程中,测量了 10 个数据点,记录数据如表 1 所示,通过最小二乘法处理后,得到式(6)中各个回归系数及其估计精度,如表 2 所示。

表 1 *OCV-SOC* 实验数据表

Tab. 1 *OCV-SOC* experiment data

编号	OCV_1/V	SOC_1/%	OCV_2/V	SOC_2/%	T/℃
1	2.7234	11.31	1.9800	<10.00	23
2	3.0798	20.79	2.9663	16.05	21
3	3.1494	30.27	3.1384	25.53	19
4	3.1677	39.75	3.1555	35.01	18
5	3.1934	49.23	3.1824	44.49	20
6	3.2104	58.71	3.2031	53.97	22
7	3.2263	68.19	3.2178	63.45	24
8	3.2397	77.67	3.2336	72.93	20
9	3.2458	87.15	3.2434	82.41	18
10	3.2483	96.63	3.2471	91.98	19

使用 Matlab 将回归方程绘成曲线[15],得如图 4 所示的 *OCV-SOC* 曲线,图中拟合的 2 条曲线并不完全重合,说明 2 节电池之间有一定差异,拟合曲线有一定曲度,提高了数据查询的精度。折线处理的 *SOC-T* 曲线如图 5 所示,可用来查询上次维护态下某一 *SOC* 对应的温度。

表 2 回归系数及估计精度表

Tab. 2 Regression coefficient and precision estimation

回归参数	电池 1 系数	估计精度	电池 2 系数	估计精度
b_0	25311	6.49	2495	0.71
b_1	−33652	8.53	−3738	1.04
b_2	16761	4.20	2071	0.57
b_3	−3707	0.92	−505	0.13
b_4	307	0.08	48	0.01

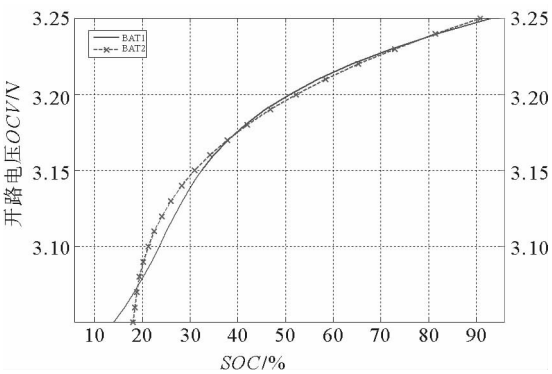

图 4 *OCV-SOC* 最小二乘法关系曲线图

Fig. 4 *OCV-SOC* Curves of the least squares algorithm

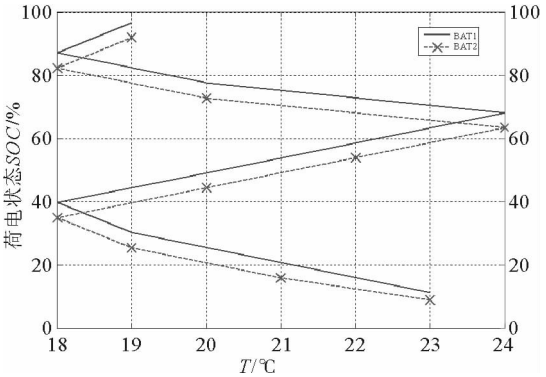

图 5 *SOC-T* 关系曲线图

Fig. 5 Curves of *SOC-T*

2)磷酸铁锂电池放电实验

采用恒定电流放电,每放电一段时间后静置 1 h,确保放电测量点和充电测量点是不同的,10 次测量的 *SOC* 值以及通过图 4 所示 *OCV-SOC* 曲线查得的 *SOC′* 值如表 3 所示,并按式(4)计算放电时的相对误差变化 *ERR*。可见,在一次放电过程中,2 节磷酸铁锂电池 SOC 估计的最大相对误差为 8.35%,式(4)中判断标准选为 10%是合理的。如果相对误差大于 10%,则说明由于环境温度、电流、内阻、循环次数等变化造成了图 4 曲线已不再准确,故需重新进入维护态再次拟合曲线。

表 3 *SOC* 相对误差数据表

Tab. 3 Relative error data of *SOC*

编号	SOC_1/%	SOC'_1/%	ERR_1	SOC_2/%	SOC'_2/%	ERR_2
1	99.00	91.03	0.0805	94.26	96.0	−0.0185
2	89.52	88.15	0.0153	84.78	85.59	−0.0096
3	80.04	84.17	−0.0516	75.30	79.02	−0.0493
4	70.56	72.17	−0.0228	65.82	67.24	−0.0215
5	61.08	58.37	0.0384	56.34	56.34	0.0000
6	51.60	50.54	0.0206	46.86	45.45	0.0302
7	42.12	40.14	0.0471	37.38	34.26	0.0835
8	32.64	33.21	−0.0174	27.90	29.77	−0.0672
9	23.16	24.72	−0.0674	18.42	17.67	0.0405
10	——	——	——	8.94	8.93	0.0008

3)磷酸铁锂电池温度变化实验

在静置状态下,每隔 3 h 测量一次电压数据,数据的变化主要由电池自放电和温度变化造成的,如表 4 所示。可见气温变化对开路电压有一定影响,相应的 SOC 变化也较明显,近似可以得到温度每变化 1 ℃对应 SOC 的变化约为 1%,故式(5)中的判断标准选为 10 ℃也是合理的。

表 4 *SOC-T* 实验数据表

Tab. 4 *SOC-T* experiment data

编号	OCV_1/V	SOC_1/%	OCV_2/V	SOC_2/%	T/℃
1	3.2263	68.94	3.2254	69.28	23
2	3.2239	66.95	3.2230	67.46	22
3	3.2227	65.98	3.2215	66.35	21
4	3.2214	64.96	3.2201	65.32	20
5	3.2202	64.00	3.2187	64.31	19
6	3.2190	63.00	3.2175	63.46	18
7	3.2178	62.22	3.2164	62.69	17
8	3.2166	61.34	3.2151	61.79	16

4 结论

为了提高磷酸铁锂电池 SOC 估计的精度,同时避免采用过于复杂的数学算法和模型分析,提出了一种 SOC 估计的实用方法,可通过电池管理系统自动识别并进入维护态,通过最小二乘法拟合 *OCV-SOC* 曲线,纠正环境温度、电流、内阻、循环次数等变化的影响。进一步的实验还表明,测量数据点数越多,基于最小方

差意义上的最小二乘算法精度越高，当测量点数达到 20 个时，磷酸铁锂电池 SOC 估计的最大相对误差小于 5%，但此时维护态时间太长，具体应用场合可根据 SOC 估计精度和维护时间要求，合理选择采集数据点数。该办法简单易行，可在确保精度的条件下，大大减少现场数据量，特别适用于矿井等恒温场所。

参考文献：

[1]杨阳，汤桃峰，秦大同，等. 电动汽车锂电池 PNGV 等效电路模型与 SOC 估算方法[J]. 系统仿真学报，2012，24(4)：938-942.
Yang Yang，Tang Taofeng，Qin Datong，et al. PNGV equivalent circuit model and SOC estimation algorithm of lithium batteries for electric vehicle[J]. Journal of System Simulation，2012，24(4)：938-942.

[2]刘吉良. 电动汽车用磷酸铁锂电池建模及剩余电量估计[D]. 秦皇岛：燕山大学，2012：15-20.

[3]贾玉健，解大，顾羽洁，等. 电动汽车电池等效电路模型的分类和特点[J]. 电力与能源，2011，32(6)：516-521.
Jia Yujian，Xie Da，Gu Yujie，et al. Classification and characteristics of equivalent circuit models for EV's battery[J]. Power & Energy，2011，32(6)：516-521.

[4]尹安东，张万兴，赵韩，等. 基于神经网络的磷酸铁锂电池 SOC 预测研究[J]. 电子测量与仪器学报，2011，25(5)：433-437.
Yin Andong，Zhang Wanxing，Zhao Han，et al. Research on estimation for SOC of $LiFePO_4$ Li-ion battery based on neural network[J]. Journal of Electronic Measurement and Instrument，2011，25(5)：433-437.

[5]高明煜，何志伟，徐杰. 基于采样点卡尔曼滤波的动力电池 SOC 估计[J]. 电工技术学报，2011，26(11)：161-167.
Gao Mingyu，He Zhiwei，Xu Jie. Sigma point kalman filter based SOC estimation for power supply battery[J]. Transactions of China Electrotechnical Society，2011，26(11)：161-167.

[6]张利，王为，陈泽坚，等. 新能源汽车 SOC 估算的模糊预测算法研究[J]. 电子测量与仪器学报，2011，25(4)：315-319.
Zhang Li，Wang Wei，Chen Zejian，et al. Research of fuzzy prediction algorithm in SOC estimation of new energy vehicles[J]. Journal of Electronic Measurement and Instrument，2011，25(4)：315-319.

[7]时玮，姜久春，李索宇，等. 磷酸铁锂电池 SOC 估算方法研究[J]. 电子测量与仪器学报，2010，24(8)：769-774.
Shi Wei，Jiang Jiuchun，Li Suoyu，et al. Research on SOC estimation for $LiFePO_4$ Li-ion batteries[J]. Journal of Electronic Measurement and Instrument，2010，24(8)：769-774.

[8]王斯成，陈子平，杨军，等. 蓄电池剩余容量(SOC)数学模型探讨和在线测试仪的开发[J]. 太阳能学报，2005，26(1)：6-13.
Wang Sicheng，Chen Ziping，Yang Jun，et al. SOC modeling for lead-acid battery and developments of SOC on-line tester[J]. Acta Energiae Solaris Sinica，2005，26(1)：6-13.

[9]李哲，韩雪冰，卢兰光，等. 动力型磷酸铁锂电池的温度特性[J]. 机械工程学报，2011，47(18)：115-120.
Li Zhe，Han Xuebing，Lu Languang，et al. Temperature characteristics of power $LiFePO_4$ batteries[J]. Journal of Mechanical Engineering，2011，47(18)：115-120.

[10]曾洁，卜凡涛. 基于多项式回归算法的锂电池 SOC 估测[J]. 大连交通大学学报，2011，32(4)：70-74.
Zeng Jie，Bu Fantao. Study of SOC estimation and measurement of Li-ion battery based on polynomial regression algorithm [J]. Journal of Dalian Jiaotong University，2011，32(4)：70-74.

[11]费业泰. 误差理论与数据处理[M]. 北京：机械工业出版社，2010：100-125.

[12]王永华，张浞，薄翠梅，等. 矩阵类的 C#语言实现及其工程应用[J]. 南京工业大学学报，2005，27(4)：92-95.
Wang Yonghua，Zhang Shi，Bo Cuimei，et al. Realization and application of matrix class by program language C sharp[J]. Journal of Nanjing University of Technology，2005，27(4)：92-95.

[13]黄保和，江弋. C 语言程序设计[M]. 北京：清华大学出版社，2006：135-172，188-206，227-259.

[14]Jeff Prosise. Programming windows with MFC[M]. 2nd ed. Pedmond，WA：Microsoft Press，2007：278-420，721-880.

[15]王正林，刘明. 精通 MATLAB7[M]. 北京：电子工业出版社，2006：23-52，253-300.

附录一(MyMatrix.h 头文件)

```
//申明 5 个矩阵运算函数
extern void Matrix_ADD(int N, float * MatrixA, float * MatrixB, float * MatrixR);
extern void Matrix_SUB(int N, float * MatrixA, float * MatrixB, float * MatrixR);
extern void Matrix_MULT(int N, float * MatrixA, float * MatrixB, float * MatrixR);
extern void Matrix_TRANSPOSE(int N, float * Matrix, float * MatrixR);
extern int Matrix_INVERSE(int N, float * Matrix,float * MatrixR);
```

附录二(MyMatrix.c 源文件)

```
#include "stdio.h"
#include "malloc.h"
#include "math.h"
//矩阵加法,N 为矩阵维数,结果在 MatrixR
void Matrix_ADD(int N, float *MatrixA, float *MatrixB,
float *MatrixR)
{
    for(int i=0;i<N;++i)
        for(int j=0;j<N;++j)
    MatrixR[i*N+j]=MatrixA[i*N+j]+MatrixB[i
*N+j];
}
//矩阵减法,N 为矩阵维数,结果在 MatrixR
void Matrix_SUB(int N, float *MatrixA, float *MatrixB,
float *MatrixR)
{
    for(int i=0;i<N;++i)
        for(int j=0;j<N;++j)
    MatrixR[i*N+j]=MatrixA[i*N+j]-MatrixB[i*
N+j];
}
//矩阵乘法,N 为矩阵维数,结果在 MatrixR
void Matrix_MULT(int N, float *MatrixA, float *Ma-
trixB, float *MatrixR)
{
    float Temp;
    for (int i=0;i<N;++i)
      for (int j=0;j<N;++j)
        {
            Temp=0.0;
            for (int k=0;k<N;++k)
        Temp += MatrixA[i*N+k]*MatrixB[k*N
+j];
            MatrixR[i*N+j] = Temp;
        }
}
//矩阵转置,输入在 Matrix,结果在 MatrixR
void Matrix_TRANSPOSE(int N, float *Matrix, float *
MatrixR)
{
    for (int i=0; i<N; ++i)
        for (int j=0 ; j<N; ++j)
          MatrixR[j*N+i] = Matrix[i*N+j];
}
//变量交换函数,供矩阵求逆使用
void VarSwap(float *TempA, float *TempB)
{
    float TempC;
    TempC = *TempA;
    *TempA = *TempB;
    *TempB = TempC;
}
//矩阵求逆,输入在 Matrix,结果在 MatrixR
//Gauss-Jordan 法求逆成功返回 1,无逆矩阵返回 0
int Matrix_INVERSE(int N, float *Matrix, float *Ma-
trixR)
{
  int *iBuffer, *jBuffer,i,j,k;
  iBuffer=(int *)malloc(N*sizeof(int));
  jBuffer=(int *)malloc(N*sizeof(int));
  for(i=0;i<N;++i)
      for(j=0;j<N;++j)
        MatrixR[i*N+j] = Matrix[i*N+j];
  float CTemp,ResultMax;
  for(k=0;k<N;k++)
  {
    ResultMax=0.0;
    for(i=k;i<N;i++)
    for(j=k;j<N;j++)
    {
      CTemp=fabs(*(MatrixR+i*N+j));
      if(CTemp>ResultMax)
      {   ResultMax=CTemp;
          iBuffer[k]=i;
              jBuffer[k]=j;
      }
    }
  if(ResultMax<0.000001)
  {
    free(iBuffer);
    free(jBuffer);
  return(0);
  }
  if((i=iBuffer[k])!=k)
    for(j=0;j<N;j++)
      VarSwap(MatrixR+k*N+j,MatrixR+i*N+j);
  if((j=jBuffer[k])!=k)
    for(i=0;i<N;i++)
      VarSwap(MatrixR+i*N+k,MatrixR+i*N+j);
  MatrixR[k*N+k]=1.0/MatrixR[k*N+k];
  for(j=0;j<N;j++)
        if(j!=k)
```

(下转第 108 页)