基于 GPIB 的锁相放大器控制

江昊翰，钟添芸，王旭龙，杨云皓

求是科学班（计算机科学与技术）1801

摘　要：随着科学技术的发展，计算机的运用范围逐渐变得更为广泛。越来越多的物理测量仪器可以通过计算机程序控制，设定其测试步骤并且获取测量结果。而锁相放大器作为一种精密的测量仪器,也可以由计算机进行控制。本文提出了一种方法，即通过编写 C 程序经 GPIB 接口控制锁相放大器。这种方法大大简化了测量中调节锁相放大器旋钮的过程，节省了时间和人力。

关键词： 锁相放大器；GPIB；程序控制

中图法分类号： O4-29　　**文献标识码：** B

GPIB-based lock-in amplifier control

Hao-Han Jiang, Tian-Yun Zhong, Xu-Long Wang, Yun-Hao Yang

(1.*Dept. of Computer Science, Zhejiang University, 310012, China*)

Abstract: With the development of science and technology, the use of computers has gradually become wider. More and more physical measurement instruments can be controlled by computer programs, set their test procedures and obtain measurement results. As a precision measuring instrument, the lock-in amplifier can also be controlled by a computer. This paper presents a method to control the lock-in amplifier by writing a C program via the GPIB interface. This method greatly simplifies the process of adjusting the knob of the lock-in amplifier during measurement, saving time and labor.

Key words: Lock-in Amplifier; GPIB; Programming Control

1 问题的提出

1.1 锁相放大器基本原理

锁相放大器（也称为相位检测器）是一种可以从干扰极大的环境（信噪比可低至-60dB，甚至更低）中分离出特定载波频率信号的放大器。锁相放大器能够在极强噪声环境中提取信号幅值和相位信息，采用零差检测方法和低通滤波技术，测量相对于周期性参考信号的信号幅值和相位。锁相测量方法可提取以参考频率为中心的指定频带内的信号，有效滤除所有其他频率分量。

锁相放大器是根据正弦函数的正交性原理工作的，当一个频率为ω_1的正弦信号与另一个频率为ω_2的正弦信号相乘，然后对乘积进行积分。只有当两频率相同时，所得结果不为零，且此时结果与两信号幅值与相位差相关。通过这种处理方式，可以将信号中不同频率的波段过滤掉，再还原得到所需的信号幅值与相位。

在一般的交流电实验中，通常将输出端的信号作为锁相放大器中的参考信号，将待测元件两端电压接入锁相放大器中作为测量信号。这样可以很好地消除因外界干扰而产生的杂波信号，提高实验准确度。

1.2 问题的提出

在日常的实验中，我们感受到仪器参数调节与读数是一项极为麻烦的事情。特别是当数据处理并绘制图表时，想使点的横坐标均匀变化而横坐标并不与某仪器输入参数成线性关系时，调节仪器将是一件极度麻烦的事情，而且往往无法准确达到想要的效果。同时，每一个数据点都要进行调节是一件极耗费时间的事情，往往会因为各种限制使得测量数据点极为有限。

我们的专业是计算机科学与技术，我们希望能通过自己的专业来解决这个问题，于是就通过 C 语言编写了经 GPIB 接口调节锁相放大器的程序，以达到对锁相放大器进行快速，准确的调节目的，并且准确地收集所测量的数据以供分析。

2 SR830 驱动程序设计

2.1 探索与比较

尽管我们小组实际拿到 SR830 锁相放大器的 Interface 时实验时间已经所剩不多，我们对于 SR830 的 PC 驱动方式选择是经过了比较细致的探索的。

最开始的时候，我们先试用了 National Instrument 的 LabVIEW 软件，如下图：

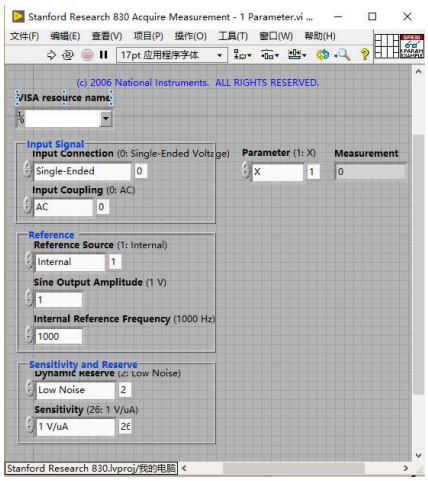


图 1 LabVIEW 测试例图

经过实际试用与资料查询，我们了解到这是一种图形可视化的硬件驱动程序开发环境。它实际上采用的是图形化编辑语言 G，最大的特点是能够直观的看出各模块层次之间的关系，同时它也是进行 NI 平台开发的核心语言：

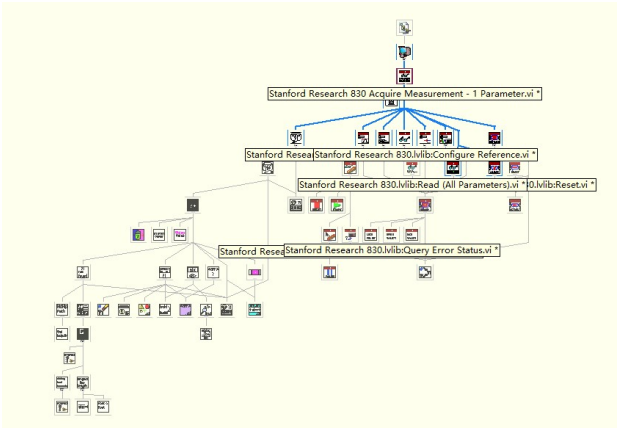


图 2 LabVIEW 模块层次例图

当然，这种开发方式的优点是相当多的，例如集成部署、信号分析、开发验证、数据管理等功能都可以通过 LabVIEW 与 NI 平台的其他工具完成。这一点得益于 LabVIEW 软件深厚的时间积累（1986 年 LabVIEW 就发行了 1.0 版本），但也正是它庞大的体系结构和功能容量，造成了它对于入门使用者最大的问题——学习成本不低。如果你是希望在实验仪器使用上有所建树或者需要复杂开发的话，的确可能需要使用 LabVIEW，但是如果只是为了某次简单实验的进行，它的确有些太过厚重。

在这之后，我们也尝试了直接使用 NI 提供的某个简单驱动，但是与之前体验相反的，已经完全 Release 封装后的小驱动程序留给我们的操作空间很小，也不能够实现比较自由的实验设计。

最终，我们转向了一个在开发难度与自由度上比较适中的仪器开发方法：使用 C 语言实现仪器控制。

2.2 C 语言 SR830 驱动程序

我们对于 C 语言程序的开发是建立在仪器厂商自带的 ni-4882.h 驱动头文件基础之上的。该头文件中的函数可以完成 PC 与锁相放大器的命令、读取、报错等工作。但是，由于开发时间较早并且属于 MIT 的 research 项目，开发范式陈旧，整体逻辑上比较晦涩难懂，不方便直接使用。

因此，我们对于 ni-4882.h 头文件进行了再度封装（SR830-Driver.h），提供了一些常用的函数，并给出了样例程序（详见第 3 节）。

比较基础的功能有：

- 错误收集函数- GpibError
- SR830 写入命令函数- SendMsg
- SR830 数据读取函数- ReadMsg
- SR830 初始化连接函数- InitSR830
- SR830 结束连接函数- ExitSR830

当然，详述代码的具体实现并不是本报告的重点，因此下面仅通过一个具体实现（从仪器获取当前的积分时间 TIME_CONSTANT）来体验这些函数的方便之处。

```
float GetTimeConstant(int Device)
{
    char tstr[101];
    float TIME_CONSTANT = 0.0;
    SendMsg(Device, "OFLT?"); /* Query TIME_CONSTANT */
    ReadMsg(Device, tstr); /* Read TIME_CONSTANT */

    int temp = atoi(tstr);
    float TCmap[20] = { 0.01f, 0.03f, 0.1f, 0.3f, 1.0f, 3.0f, 10.0f, 30.0f, 100.0f, 300.0f,  /* TC number to real TIME_CONSTANT map */
                        1000.0f, 3000.0f, 10000.0f, 30000.0f, 100000.0f, 300000.0f, 1000000.0f, 3000000.0f, 10000000.0f, 30000000.0f };
    TIME_CONSTANT = TCmap[temp]; /* Tranfer TC number to real TIME_CONSTANT */
    printf("Time Constant = %f\n", TIME_CONSTANT); // Output TIME_CONSTANT
    return TIME_CONSTANT;
```

图 3 SR830-Driver 函数实例 - GetTimeConstant

上述函数完成了查询当前仪器积分时间的功能。

具体步骤为：

1. 向仪器发出“OFLT?”查询命令
2. 从仪器读出积分周期代码
3. 再从积分周期代码表中查出对应的实际是分时间
4. 返回结果

可以注意到，使用了我们封装的库之后，实际上真正与仪器通信的“困难”部分在图中仅使用两行代码就完成了（图中红框部分）。因此，我们的 SR830-Driver 对于仪器智能使用开发的确是有成效的。

2.3 优势

使用上述 C 语言驱动来进行锁相放大实验的优势有：

- 开发难度较低。C 语言的使用是许多大学的计算机先修课程，因此拥有相当高的普及率。而我们封装的 SR830-Driver.h 头文件中的函数也仅仅需要入门级的 C 语言能力就可以正常使用了
- 程序自由度大。PC 机与锁相放大器的通信可以分为命令与读取，此二者都在我们的驱动程序有良好的实现。因此，使用者可以查阅官方命令手册，相当方便地构造出实际需要的复杂逻辑。
- 反馈体验好。由于驱动程序中加入了错误收集系统，因此当实验过程出现问题时，程序能够及时有效地向使用者发出反馈并终止程序。使用者便可以通过反馈的错误信息对于驱动程序或者实验步骤进行调整。

3 实验示例

3.1 实验简述

我们设计了一个基本交流电实验以体现程序控制的优越性：

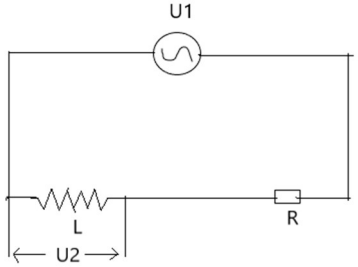


图 4 实验电路图

将电源输出 U1 作为参考信号，U2 作为测量信号，调节电源输出频率，测量不同频率下的 U2，并以此得到电感参数。

3.2 实验原理

$$U_1 = U_1 e^{i\omega t}$$

$$Z = j\omega L + R = \sqrt{R^2 + \omega^2 L^2} \cdot e^{i\varphi}$$

$$tan\varphi = \frac{\omega L}{R}$$

$$i = \frac{U_1}{Z} = \frac{U_1}{\sqrt{R^2 + \omega^2 L^2}} \cdot e^{-i\varphi}$$

$$U_2 = j\omega L \cdot i = \frac{\omega L \cdot U_1}{\sqrt{R^2 + \omega^2 L^2}} \cdot e^{i\left(\frac{\pi}{2} - \varphi\right)}$$

由最后一个式子平方。可以得到结果f^2于U_2^2成线性关系，我们只需要测量对应地值就可以得出待测地电感值。

我们使用锁相放大器完成这个实验。即通过调整电路地频率，测量电感两端地电压值来拟合曲线，计算出结果。

3.3 实验结果

数据测量结果见附录。

在这次实验过程当中，我们使用的模式是修改锁相放大器地频率之后等待一段时间，然后进行测量。为了模拟大量测试点地效果，我们取了 200 个测试点进行测量。

这个过程持续了半个小时，我们将测试地结果用统计软件进行统计，得到的相关性非常好，这证明了锁相放大器的精密；同时，得益于程序设计中可以随意设置间距的好处，整个结果图片中的点是均匀分布的，得到了良好的结果。

经过实验的测试，在两个测量数据不是单纯的线性关系时使用程序进行计算，可以合理地计算出每一个测试点的位置。进而得出结果较好且较为均匀的数据。而且 C 程序作为大学生课程中的必修课，接受过高等教育的人很容易掌握相关代码的设计方法，这样可以在实验过程中给予实验人员更多的自由度，设计出更多的测量模式。

而半个小时的测量过程中不需要人为操作仪器，可以大大节省人力的资源。

总的来说，这个实验验证了我们之前提到的各种优势，起到了作用。简化了测量中调节锁相放大器旋钮的过程，节省了时间和人力。

4 结语

经过本次的实验。我们将自身专业所需到的知识很好的应用在了物理学的相关实验中，做出了较有意义的结果。在此过程中，我们熟悉了锁相放大器这一仪器，并且学会使用 GPIB 接口进行通讯，同时也了解到了如何设计程序来自动控制锁相放大器工作，并且获取所需要的数据。

感谢郑老师在这次实验中提供的指导，给予我们各种各样的建议，将我们自身的专业优势发挥了出来。

参考文献:

［1］ Wikipedia. Lock-in amplifier [G/OL]. Wikipedia, 2019[2019]. https://en.wikipedia.org/wiki/Lock-in_amplifier.

［2］ Datasheets of SR830

［3］ Wikipedia. IEEE-488 [G/OL]. Wikipedia, 2017[2018] . en.wikipedia.org/wiki/IEEE-488

作者简介:

江昊翰(3180101995)，求是科学班（计算机科学与技术）1801。

钟添芸(3180103009)，求是科学班（计算机科学与技术）1801。

王旭龙(3180105260)，求是科学班（计算机科学与技术）1801。

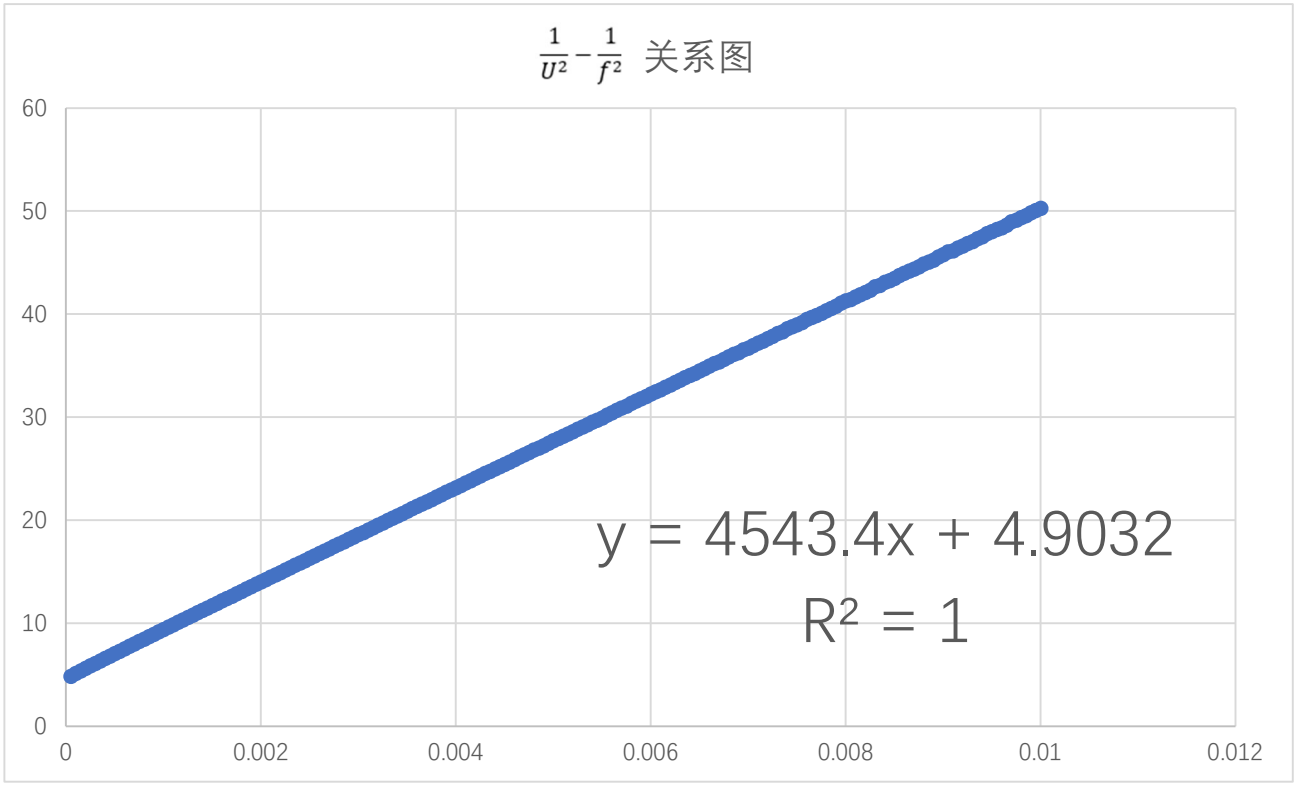
杨云皓(3180106047)，求是科学班（计算机科学与技术）1801。

附录

附录一 实验数据

f/Hz	U_2/V	f/Hz	U_2/V	f/Hz	U_2/V	f/Hz	U_2/V	f/Hz	U_2/V
141.4214	0.453615	22.0863	0.265017	15.71349	0.206849	12.85649	0.175538	11.14557	0.155396
100	0.441896	21.82179	0.26288	15.61738	0.20575	12.8037	0.174989	11.11112	0.154969
81.64966	0.431459	21.56656	0.260805	15.52302	0.204896	12.75154	0.174378	11.07698	0.154542
70.71068	0.421999	21.32007	0.258852	15.43034	0.203858	12.70002	0.173768	11.04316	0.154115
63.24556	0.413149	21.08185	0.256899	15.33931	0.202943	12.64912	0.173158	11.00965	0.153749
57.73503	0.404909	20.85144	0.255007	15.24986	0.201905	12.59882	0.172608	10.97643	0.153138
53.45225	0.39728	20.62843	0.253054	15.16197	0.201112	12.54912	0.171937	10.94352	0.152894
50	0.390017	20.41242	0.251223	15.07557	0.200135	12.50001	0.171388	10.9109	0.152345
47.14045	0.383181	20.20305	0.249452	14.99064	0.19922	12.45146	0.170899	10.87857	0.152101
44.72136	0.376711	20	0.247682	14.90713	0.198365	12.40348	0.170289	10.84653	0.151673
42.64015	0.370546	19.80295	0.245973	14.82499	0.197511	12.35605	0.169801	10.81477	0.151185
40.82483	0.364687	19.61162	0.244264	14.7442	0.196595	12.30916	0.16919	10.78328	0.150819
39.22323	0.359133	19.42572	0.242677	14.66472	0.19568	12.26279	0.16858	10.75207	0.150453
37.79645	0.353823	19.24501	0.24109	14.58651	0.194886	12.21695	0.168214	10.72113	0.150147
36.51484	0.348757	19.06925	0.239442	14.50953	0.194093	12.17162	0.167603	10.69046	0.149781
35.35534	0.343874	18.89823	0.237916	14.43376	0.193177	12.12679	0.166993	10.66004	0.149293
34.29972	0.339235	18.73172	0.23639	14.35917	0.192445	12.08245	0.166444	10.62989	0.148988
33.33334	0.33478	18.56954	0.234864	14.28572	0.191712	12.03859	0.166078	10.59999	0.148683
32.44429	0.330507	18.4115	0.233338	14.21339	0.190797	11.99521	0.165467	10.57034	0.148194
31.62278	0.326357	18.25742	0.231874	14.14214	0.189942	11.95229	0.165101	10.54093	0.147828
30.86067	0.322451	18.10715	0.230592	14.07196	0.189271	11.90983	0.164552	10.51177	0.147401
30.15114	0.318605	17.96054	0.229188	14.00281	0.188539	11.86782	0.164002	10.48286	0.147218
29.4884	0.314821	17.81742	0.227784	13.93467	0.187745	11.82626	0.163575	10.45417	0.146791
28.86752	0.311281	17.67768	0.226441	13.86751	0.187074	11.78512	0.163026	10.42573	0.146485
28.28427	0.307863	17.54117	0.22516	13.80132	0.18628	11.74441	0.162599	10.39751	0.146119
27.73501	0.304567	17.40777	0.223817	13.73606	0.185548	11.70412	0.161988	10.36952	0.145814
27.21656	0.301271	17.27737	0.222535	13.67172	0.184876	11.66424	0.161622	10.34176	0.145387
26.72612	0.29822	17.14986	0.221375	13.60828	0.184083	11.62477	0.161012	10.31422	0.145082
26.26129	0.29529	17.02514	0.220155	13.54572	0.183473	11.5857	0.160584	10.2869	0.144654
25.81989	0.292238	16.90309	0.218873	13.484	0.182801	11.54701	0.160218	10.25979	0.144349
25.40002	0.289431	16.78363	0.217652	13.42313	0.182008	11.50871	0.159791	10.2329	0.144044
25	0.286684	16.66667	0.216493	13.36307	0.181336	11.47079	0.159181	10.20621	0.1438
24.6183	0.283937	16.55212	0.215333	13.30381	0.180604	11.43325	0.158753	10.17974	0.143434
24.25356	0.281435	16.4399	0.214234	13.24533	0.179994	11.39606	0.158387	10.15347	0.142945
23.90457	0.278811	16.32994	0.213197	13.18762	0.179383	11.35924	0.15796	10.1274	0.142701
23.57022	0.276369	16.22215	0.212098	13.13065	0.178651	11.32278	0.157472	10.10153	0.142335
23.24953	0.273989	16.11647	0.211061	13.07442	0.178041	11.28666	0.157044	10.07586	0.142091
22.94157	0.271669	16.01282	0.209901	13.0189	0.17743	11.25089	0.156617	10.05039	0.141664
22.64554	0.269411	15.91115	0.208924	12.96408	0.176759	11.21545	0.156068	10.0251	0.141358
22.36068	0.267214	15.8114	0.207826	12.90995	0.176148	11.18035	0.155641	10.00001	0.141053

附录二 实验结果图像



附录三 程序获取地址

https://github.com/Kizuna-AII/SR830-Driver