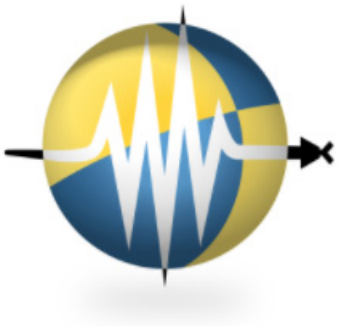

ObsPy

A Python Framework for Seismology

OBSPY 参考手册

基于 OBSPY v1.0.3

目　录

图目录

表目录

第一部分

OBSPY 教程

第 1 章　OBSPY 简介

1.1　OBSPY 是什么？

Seismic Analysis Code，简写为 OBSPY，是天然地震学领域使用最广泛的数据分析软件 包之一。

OBSPY 首先是一个软件，主要在命令行下操作，通过各种命令来处理时间序列数据，尤 其是地震波形数据，同时也提供了一个简单的图形界面，使得用户可以方便地查看波形并拾取震相。

OBSPY 同时还是一种数据格式，定义了以何种方式存储时间序列数据及其元数据。OBSPY 格式已经成为了地震学的标准数据格式之一，有很多工具可以实现 OBSPY 格式与其它地震 数据格式间的相互转换。

OBSPY 实现了地震数据处理过程中的常用操作，包括重采样、插值、自/互相关、震相拾 取、快速 Fourier 变换与反变换、谱估计、滤波、信号叠加等；同时为了满足数据批处理的 需求，OBSPY 设计了一个基础的编程语言，包含了变量、参数、条件判断、循环控制等特性。 这些都会在稍后的章节中详细介绍。

1.2　OBSPY 发展史

Lawrence Livermore 国家实验室 和 Los Alamos 国家实验室 是美国承担核武器设计工作的两个实验室。OBSPY 于 20 世纪 80 年代诞生于实验室的 Treaty Verification Program 小组里，该组由 W. C. Tapley 和 Joe Tull 共同领导。

起初，OBSPY 是用 Fortran 语言实现的，并将源代码分发给感兴趣的学者，允许用户进 行非商业性的地震数据处理，用户和开发者之间的合作协议要求用户提交 bug 修正和改进 以换取 OBSPY 的使用权。到了大概 1990 年，OBSPY 已经成为全球地震学家的数据处理标准软 件。

从 1992 年开始，OBSPY 的开发逐渐由 Livermore 接管，并开始通过分发协议严格限制 源代码的分发。与此同时，开发者认为 Fortran 是一种过于局限的编程语言，其阻碍了 OBSPY 特性的进一步开发，因而开发者使用 f2c[1] 转换工具将 OBSPY 的 Fortran 源码转换成了 C 源 码[2]。接下来，Livermore 以转换得到的 C 源码为基础，计划开发一个商业版的地震数据处

[1] Fortran77 语言到 C 语言的自动转换工具。

[2] 个人猜测，目前 OBSPY 源码的混乱和不易读正是由于这次自动转换导致的。

理产品，命名为 OBSPY2000。这个版本扩展了很多功能，其中一个功能是建立一个日志数据 库，记录一个波形从原始数据到最终产品之间的所有处理步骤。这样的设计允许用户随时提交数据处理的中间结果，也可随时回滚到之前的状态。

约 1998 年，IRIS 意识到，OBSPY 的核心用户群（主要是 IRIS 的成员）无法确保能够获 取 OBSPY 的源码。IRIS 开始和 Livermore 协商，希望将 OBSPY 的开发分成两条线：一个包含 数据库特性，供核监测机构使用；另一个不包含数据库特性，仅供学术机构使用。商业化的 努力主要集中在含数据库功能的版本上。

终于，在 2005 年，IRIS 与 Livermore 签订了合同，Livermore 提供给 IRIS 一个 OBSPY 协议，允许其在 IRIS 社区内部分享 OBSPY/OBSPY2000 的源代码，并提供有限的支持以促进 社区的发展。而学术圈对于商业版的 OBSPY 没有太大兴趣，因而 Livermore 逐渐撤出了对于 OBSPY2000 的支持。最终 IRIS 完全接手了 OBSPY 的开发和技术支持，成为了一个独立的新分 支，也就是本手册中介绍的 OBSPY，有时为了区分，也称之为 OBSPY/IRIS。

1.3 OBSPY 变体

OBSPY 的发展史还是很曲折的，这也导致 OBSPY 存在多个不同的变体。

Fortran OBSPY 即 OBSPY 的 Fortran 语言实现。最后一个分发版本发布于 2003 年，版 本号 10.6f。曾经以限制性的形式在 iaspei 软件库中分发。

OBSPY2000 从 Fortran 源码转换为 C 源码，并以 C 源码为基础继续维护。该版本加入 了数据库特性以及一些新的命令。目前该版本已不再分发。

OBSPY/IRIS 由 OBSPY2000 衍生的版本，不包含数据库特性 [1]，也就是本文档所介绍的 版本，在本文档中称为 OBSPY。现在由 IRIS 下的 OBSPY 开发小组负责维护，并由 IRIS 分发。

MacOBSPY 也称为 OBSPY/BRIS，仅可在 Mac OS 下使用。该变种由 10.6d Fortran 源码 衍生而来，后期与 10.6f 集成，其功能是 OBSPY/IRIS 功能的超集。相对于 OBSPY/IRIS 的最主要扩展在于宏语言功能的增强以及处理台阵数据的能力。其作者为 George Helffrich。

1.4 申请 OBSPY

在 OBSPY 发展史 中已经说到，OBSPY 协议仅允许 IRIS 将 OBSPY 源码包及二进制包分发 给地震学相关人员。所以想要从官方渠道获取 OBSPY 软件包，必须在 IRIS 网站上申请。

OBSPY 软件包申请地址：http://ds.iris.edu/ds/nodes/dmc/forms/Obspy/ 申请的过程中需要注意如下几点：

- 认真填写个人信息，否则可能会被拒绝

[1] 目前的 OBSPY/IRIS 中还可以看到一些与数据库特性相关的命令和选项，比如很多命令中的 commit、rollback、recalltrace 选项，这些选项的存在属于历史遗留问题，且已经基本不再维护，因而本文档中完全没有提 及。

- 电子邮箱最好填写学术邮箱，一般邮箱可能会被拒绝[2]
- 若无学术邮箱，则需要其他信息证明你是地震学相关人员

IRIS 提供了 OBSPY 最新版的源码包、Linux 64 位下的二进制包和 Mac OSX 64 位下的 二进制包。其中，二进制包可以在相应的平台下直接使用，源代码包则需要编译才能使用。具体申请那个包由用户的操作系统决定：

- Linux 64 位系统可以申请源码包或 Linux 64 位包
- Mac OSX 64 位系统可以申请源码包或 Mac 64 位包
- 其他系统，如 Linux 32 位、Mac OSX 32 位、Cygwin，申请源码包 提交申请之后，需要人工审核，若审核通过，则 IRIS 会通过邮件将软件包发送给你。

一般审核时间为两到三个工作日。由于审核周期稍长，建议同时申请 64 位二进制包和源码包。

需要注意，OBSPY 协议规定了用户没有分发 OBSPY 软件包的权利。所以，请勿将 OBSPY 软 件包在网络上公开。

1.5 在 Linux 下安装 OBSPY

Linux 下安装 OBSPY，可以直接使用官方提供的二进制包，也可以手动编译源码包。对于 大多数用户而言，建议安装二进制包。下面会分别介绍两种安装方法，要求读者了解 Linux 的一些基本概念和操作。

1.5.1 安装二进制包

安装依赖包

官方提供的二进制包中的可执行文件可以直接使用，在运行时需要用到几个动态链接 库。大部分 Linux 发行版下，都默认安装了这几个动态链接库。若不幸没有安装或不确定 有没有安装，可以通过如下命令安装所需的软件包。

对于 Ubuntu/Debian：

```
$ sudo apt-get update
$ sudo apt-get install libc6 libsm6 libice6 libxpm4 libx11-6
$ sudo apt-get install zlib1g libncurses5
```

对于 CentOS/Fedora/RHEL：

```
$ sudo yum install glibc libSM libICE libXpm libX11
$ sudo yum install zlib ncurses
```

安装二进制包

直接将官方提供的二进制包解压并移动到安装目录即可：

```
$ tar -xvf Obspy-101.6a-linux_x86_64.tar.gz  # 解压
$ sudo mv Obspy /usr/local                   # 安装
```

[2] 学术邮箱是指能证明你学术身份的邮箱，如 edu 结尾的邮箱。

1.5.2 编译源码

安装依赖包

编译源码时需要安装若干软件包。

对于 Ubuntu/Debian 系：

```
$ sudo apt-get update
$ sudo apt-get install build-essential
$ sudo apt-get install libncurses5-dev libsm-dev libice-dev
$ sudo apt-get install libxpm-dev libx11-dev zlib1g-dev
```

对于 CentOS/Fedora/RHEL 系：

```
$ sudo yum install gcc gcc-c++ make
$ sudo yum install glibc ncurses-devel libSM-devel libICE-devel
$ sudo yum install libXpm-devel libX11-devel zlib-devel
```

编译源码

将源码按如下命令解压、配置、编译、安装：

```
$ tar -xvf Obspy-101.6a_source.tar.gz
$ cd Obspy-101.6a
$ mkdir build
$ cd build
$ ../configure --prefix=/usr/local/Obspy
$ make
$ sudo make install
```

1.5.3 配置变量

向 ~/.bashrc[1] 中加入如下语句以配置环境变量和 OBSPY 全局变量：

```
export OBSPYHOME=/usr/local/Obspy
export OBSPYAUX=${OBSPYHOME}/aux
export PATH=${OBSPYHOME}/bin:${PATH}

export OBSPY_DISPLAY_COPYRIGHT=1
export OBSPY_PPK_LARGE_CROSSHAIRS=1
export OBSPY_USE_DATABASE=0
```

其中，

- OBSPYHOME 为 OBSPY 的安装目录
- OBSPYAUX 目录中包含了 OBSPY 运行所需的辅助文件
- PATH 为 Linux 系统环境变量

[1] 某些发行版需要修改 ~/.bash_profile

- OBSPY_DISPLAY_COPYRIGHT 用于控制是否在启动 OBSPY 时显示版本和版权信息，一般设 置为 1。在脚本中多次调用 OBSPY 时会重复显示版本和版权信息，干扰脚本的正常输 出，因而在脚本中一般将其值设置为 0。具体的设置方法可以参考 “脚本中调用 OBSPY” 中的相关内容
- OBSPY_PPK_LARGE_CROSSHAIRS 用于控制震相拾取过程中光标的大小，在 震相拾取时 会用到
- OBSPY_USE_DATABASE 用于控制是否允许将 OBSPY 格式转换为 GSE2.0 格式，一般用不 到该特性，故而设置其值为 0

修改完 ~/.bashrc 后，执行以下命令使配置的环境变量生效：

```
$ source ~/.bashrc
```

1.5.4　启动 OBSPY

终端键入小写的 Obspy[1]，显示如下则表示 OBSPY 安装成功：

```
$ Obspy
 SEISMIC ANALYSIS CODE [11/11/2013 (Version 101.6a)]
 Copyright 1995 Regents of the University of California

OBSPY>
```

1.6　在 Mac 下安装 OBSPY

本节介绍如何在 Mac OS X 10.10 Yosemite 安装 OBSPY。

Mac 下安装 OBSPY，可以直接使用官方提供的二进制包，也可以手动编译源码包。对于 大多数用户而言，建议安装二进制包。下面会分别介绍两种安装方法。

1.6.1　准备工作

首先要安装 Mac 下的命令行工具。在终端执行如下命令：

```
$ xcode-select --install
```

还需要安装 X11 相关工具，到 XQuartz 下载 dmg 安装包并安装即可。

1.6.2　安装二进制包

直接将官方的二进制包解压并移动到安装目录即可：

```
$ tar -xvf Obspy-101.6a-mac_x86_64.tar.gz
$ sudo mv Obspy /usr/local
```

[1]Ubuntu 的源里有一个名叫 Obspy 的软件，是用来显示登录账户的一些信息；CentOS 的源里也有一个名叫
Obspy 的软件，是 CSS 语法分析器的 Java 接口。所以一定不要试图用发行版自带的软件包管理器安装 Obspy!!!

1.6.3 编译源码

按照如下命令即可正确编译源码。需要注意的是，由于 OBSPY 默认使用的 editline 库在 Mac 下无法正常编译，因而执行 `configure` 时使用了 `--enable-readline` 选项使得 OBSPY 使用 readline 库而不是 editline 库。

```
$ tar -xvf Obspy-101.6a_source.tar.gz
$ cd Obspy-101.6a
$ mkdir build
$ cd build
$ ../configure --prefix=/usr/local/Obspy --enable-readline
$ make
$ sudo make install
```

1.6.4 配置变量

向 `~/.bash_profile` 中加入如下语句以配置环境变量和 OBSPY 全局变量：

```
1 export OBSPYHOME=/usr/local/Obspy
2 export OBSPYAUX=${OBSPYHOME}/aux
3 export PATH=${OBSPYHOME}/bin:${PATH}
4
5 export OBSPY_DISPLAY_COPYRIGHT=1
6 export OBSPY_PPK_LARGE_CROSSHAIRS=1
7 export OBSPY_USE_DATABASE=0
```

其中，

- `OBSPYHOME` 为 OBSPY 的安装目录
- `OBSPYAUX` 目录中包含了 OBSPY 运行所需的辅助文件
- `PATH` 为系统环境变量
- `OBSPY_DISPLAY_COPYRIGHT` 用于控制是否在启动 OBSPY 时显示版本和版权信息，一般设 置为 1。在脚本中多次调用 OBSPY 时会重复显示版本和版权信息，干扰脚本的正常输 出，因而在脚本中一般将其值设置为 0。具体的设置方法可以参考 “脚本中调用 OBSPY” 中的相关内容；
- `OBSPY_PPK_LARGE_CROSSHAIRS` 用于控制震相拾取过程中光标的大小，在 震相拾取 时会用到
- `OBSPY_USE_DATABASE` 用于控制是否允许将 OBSPY 格式转换为 GSE2.0 格式，一般用不 到该特性，故而设置其值为 0；

修改完 `~/.bash_profile` 后，执行以下命令使配置的环境变量生效：

```
$ source ~/.bash_profile
```

1.6.5 启动 OBSPY

终端键入小写的 Obspy，显示如下则表示 OBSPY 安装成功：

```
$ Obspy
 SEISMIC ANALYSIS CODE [11/11/2013 (Version 101.6a)]
 Copyright 1995 Regents of the University of California

OBSPY>
```

1.7　邮件组

邮件组是个好东西，有点我们熟悉的 QQ 群的味道。在加入了邮件组之后，如果你在使用 OBSPY 的过程中遇到问题，可以向这个邮件组的邮箱发送邮件，该组内的所有成员都会 收到你的邮件。如果某人知道答案，他或许就会给你回复。发现了 OBSPY 的 bug 也可以向这 里报告，开发者会尽快给你回复的。当然问问题之前要思考，可阅读《提问的智慧》[1]，提交 bug 的时候要详细指出 bug 是如何出现的，也可以给出代码或文件以使得开发者能够重现 该 bug。

邮件组邮箱：Obspy-help@iris.washington.edu 订阅地址：
http://www.iris.washington.edu/mailman/listinfo/Obspy-help

[1] 参考译文：http://doc.zengrong.net/smart-questions/cn.html；有 OBSPY 方面的问题，请勿向《提问 的智慧》的译者提问！

第 2 章　OBSPY 基础

2.1　如何学习 OBSPY？

学习 OBSPY 最好的方式是找一个有经验且有耐心的人，让他/她给你演示 OBSPY 是如何工作的。　如果没有这样一个人的话，那么你就需要打开终端从头开始自学

学习 OBSPY 的过程大致可以分成三个阶段，下面列出了每个阶段的具体要求。普通用户需要达 到 OBSPY 进阶才能满足日常数据处理的要求。

OBSPY 初阶

1. 掌握　OBSPY　中最常用的命令，包括但不限于　help、read、write、plot、quit、plotpk、listhdr、　chnhdr、rmean、rtrend、taper、bandpass、plot1、plot2、cut、fft、transfer；
2. 理解地震数据处理流程，参见 “OBSPY 数据处理” 一章；
3. 了解 OBSPY 文件格式，掌握常见的 OBSPY 头段变量，理解 OBSPY 中的时间概念；
4. OBSPY 相关工具：Obspylst；

OBSPY 进阶

1. 掌握 OBSPY 的大部分命令，至少要知道哪个命令可以实现什么功能；
2. 掌握如何绘制波形图，见第 5 章；
3. 了解 OBSPY 编程以及如何在脚本中调用 OBSPY，见第 6、7 章；
4. 学会在自己的程序中读写 OBSPY 文件，见第 8 、9 章；

OBSPY 高阶

1. 了解 OBSPY 软件包的内部结构；
2. 自己写程序实现 OBSPY I/O 库；
3. 阅读 OBSPY 源码，了解命令的技术细节；
4. 向 OBSPY 贡献代码；

2.2　如何阅读本文档？

本文档的内容分为两个部分：教程部分、命令部分和附录。

- 教程部分介绍了 OBSPY 的基础及进阶知识，并通过尽可能多的示例来演示如何操作和使用　OBSPY。初学者应该坐在计算机前，打开终端，键入 [1] 书中的示例，试着理解每一个步骤的原理　以及结果，并不断熟悉常用的 OBSPY 命令。
- 命令部分详细地列出了 OBSPY 中的每一个命令的语法、参数以及一些技术细节，并包含了大量　示例，适合作为参考，在需要的时候查阅。

[1] 严禁复制！不许偷懒！

- 附录部分包含了一些与 OBSPY 有关但又稍微有些偏离本文档的主线的内容。 在阅读教程的同时，应随时翻看相应命令的说明，在实践的过程中掌握基础命令的语法和用法。

这样基本就完成了 OBSPY 初阶的要求。

在读完教程部分之后，应浏览 OBSPY 的几乎所有命令，并挑选其中感兴趣的一些进行尝试。此 后，在平常的科研工作中经常使用 OBSPY，有了实践经验和对 OBSPY 的进一步认识之后，可以阅读文 档中的进阶内容，达到 OBSPY 进阶的要求。

最后，如果对 OBSPY 的内部机理感兴趣，可以阅读 OBSPY 的源码，重新实现一些 OBSPY 底层的功 能。

2.3 启动和退出

在终端键入 `Obspy` 以启动 OBSPY，显示如下版本号以及版权信息[1]：

```
$ obspy
 SEISMIC ANALYSIS CODE [11/11/2013 (Version 101.6a)]
 Copyright 1995 Regents of the University of California

OBSPY>
```

其中，“`OBSPY>`” 是 OBSPY 程序特有的命令提示符。

退出 OBSPY：

```
OBSPY
```

也可以使用 `done`、`exit` 命令退出 OBSPY，但不推荐。 一次完整的启动和退出称为一个 OBSPY 会话。

2.4 OBSPY 设计思想

OBSPY 的设计思想大概可以总结如下：

1. 每个信号[2] 被保存到单独的 OBSPY 格式数据文件中；
2. OBSPY 格式包含了描述数据特征的头段区和存储信号的数据区，参见 “OBSPY 文件格式” 一章；
3. 将单个或多个 OBSPY 文件从磁盘读入内存；
4. 通过各种命令对内存中的数据进行操作；
5. 操作完毕，将内存中的数据写入到磁盘，可以覆盖原 OBSPY 文件或写入新文件中。

读取 OBSPY 文件时的若干限制：

- OBSPY 一次性最多处理 1000 个 OBSPY 文件；想要修改这个上限，参考 “修改最大允许的文件数 目”一节；
- 单个文件名所允许的最大长度为 128 字符；

[1] Livermore 实验室由 University of California 于 1952 年创立，2007 年改由 University of California、Bechtel National、BWX Technologies、Washington Group International 共同组成的安全机构管理。

[2] 信号，或称之为 trace，即**单个**台站 **单个**仪器**单个**分量记录到的连续时间序列。

2.5　OBSPY 命令初探

2.5.1　OBSPY 命令长什么样？

一个完整的 OBSPY 命令一般由“命令 + 选项 + 参数” 构成，其中命令必须有，选项和参数可以 成对出现，也可以只出现其中一个。命令、选项以及参数之间用空格分开。如果要将多个命令写在一 行，要用分号隔开每个命令。例如：

```
OBSPY> funcgen random delta 0.1 npts 1000
OBSPY> rmean; rtrend; taper            // 一行内多个命令用分号隔开
OBSPY> write rand.OBSPY
```

其中，funcgen、write、rmean、rtrend 和 taper 是命令；random 是选项；0.1 是选项 delta 的参数，1000 是选项 npts 的参数；而 rand.OBSPY 则是一个无选项的参数 [1]。

Note: 官方文档的原文是“*command*”、“*keyword*”和“*option*”，本文档 *v2.0* 中译为“命令”、“关键字”和“参数”。个人感觉，无论是官方的用词还是 *v2.0* 版的译词都很容易让使用 *C* 语言和 *Linux* 的人困惑，因而 *v3.0* 中一律将其改为命令（*command*）、选项（*option*）和参数（*argument*）。

这里解释一下选项（*option*）和参数（*argument*）的区别。一个命令有哪些选项是由命令规定的，其控制了命令的一些特性，因而选项的作用是告诉命令“要改某个特性”。但是具体怎么改呢？这个就交给参数来控制了。命令或选项只规定了参数的类型（整型、浮点型、字符串、枚举型或者逻 辑型），用户需要根据自己的需求给定参数值。

2.5.2　大小写

OBSPY 的命令和选项都是不区分大小写的，这意味着你可以根据自己的喜好使用 funcgen 或者 FUNCGEN，OBSPY 在解释命令前都会将其转换为大写字母。

需要注意的是，由于 Linux 本身是区分大小写的，所以对于出现在参数中的文件名、目录名或者由引号包围的字符串来说，大小写是完全不同的。比如 rand.OBSPY 和 RAND.OBSPY 是两个完全不同 的参数。

2.5.3　命令简写

OBSPY 的大多数命令及选项都有简写形式。比如上面的命令简写形式如下：

```
OBSPY>fg r d 0.1 n 10
OBSPY> rmean; rtr;
OBSPY> w rand.OBSPY
```

命令和选项究竟可以简写成怎样的形式，是由 OBSPY 自身规定的。简写的好处在于，在不产生歧 义的前提下尽量减少用户的击键数；坏处在于，若对命令不是足够熟悉，简写后的命令变得很难读 和难理解。比如你一看就知道 delta 代表的是采样周期 [2]，而 d 却不那么直观，可能是 delta，也可 能是 demon。所以，在终端调用 OBSPY 时，可以多用简写以减少击键数，但在脚本中调用 OBSPY 时应 仅使用那些常用命令的简写，不要滥用，否则一段时间后你会看不懂自己写的脚本的。

2.5.4　查看命令语法

OBSPY 自带了英文的帮助文档，详细解释了每个命令的语法，可以通过 help 命令查看相应文档：

[1] 其实可以有很多选项，这里都省略了。

[2] 也称为采样时间，即两次数据采样的时间间隔，本文档将统一使用 “采样周期”。

```
OBSPY> help funcgen write // 命令的简写是 h fg w
```

也可以直接查看 `$OBSPYHOME/aux/help` 下的文档，或者查看本文档的命令部分。

2.5.5 参数默认值

为了让 OBSPY 易学易用，几乎所有命令参数都有一个“系统默认参数值”，这些“系统默认参数 值”都是经过精心挑选的，同时用户又可以随时修改参数值。这样的设计使得 OBSPY 易用同时又不失 灵活性。

下面以 C 语言为例做一些说明[1]，希望能够帮助理解 OBSPY 参数的一些特点。

在 C 语言中，函数有主函数和子函数之分，变量又有全局变量和局部变量之分。所有的变量都可以被初始化为适当的值。

任意一个子函数，都可以使用全局变量的值，即函数的执行可以被全局变量所控制，同时也可 以修改全局变量的值。这使得代码的管理和调试变得困难。实际写程序时一般会定义专门的子函数 来修改全局变量。

任意一个子函数，又都有自己的局部变量。这些局部变量在每次子函数被调用时都会被定义、 初始化、使用和赋值，一旦子函数调用结束，变量即被撤销。如果给这些变量加上 `static` 修饰符， 则这些局部变量变身为静态局部变量。

静态局部变量，会在程序刚开始的时候就完成初始化，也是唯一的一次初始化。静态局部变量 仅在定义它的子函数里可见，子函数可以任意修改静态局部变量的值，但是每次子函数调用结束时 变量不会被撤销，因此再次调用一个子函数时，静态局部变量的值可能已经被上一次的子函数调用 所修改。

OBSPY 中有与之相对应的一些概念。Obspy 就是一个主函数，每一个命令都是一个子函数。所以
OBSPY 命令可以分为 2 类：

操作执行类 对数据进行某些操作（受全局变量控制，同时又有自己的静态局部变量）

参数设定类 改变 OBSPY 的全局参数值（即 C 语言中专门用于修改全局变量的子函数）

在启动 OBSPY（主函数）的时候，所有的选项（C 语言中的全局变量和静态局部变量）都会被初始 化为指定的 “系统默认参数值”（全局变量和静态局部变量的唯一一次初始化）。

使用参数设定类命令的时候，其修改了 OBSPY 的全局参数，会影响接下来与之相关的所有其它 命令的执行效果。使用操作执行类命令的时候，在命令中设定参数，相当于修改静态局部变量的值，不仅会影响当前命令的执行，也会影响之后所有同名命令的执行。

当你在某个命令中为某个选项指定了一个参数值的时候，该参数值会成为该命令的该选项的“参数当前值”，该“参数当前值”即成为接下来所有该命令的该选项的“当前默认值”。

鉴于 OBSPY 的这样一个特性，在一次会话中，多次执行同一个命令时，一定需要注意选项的当前 值是多少，因为这可能会影响到后面的一系列结果，这个必须理解和牢记！

令

Note: 当你在一次会话中执行了很多个命令的时候，*OBSPY* 参数可能已经被弄得一片混乱，你可以使用 *inicm* 命令在不退出 *OBSPY* 的情况下重新初始化。

下面用例子解释一下：

```
OBSPY>
funcgen
OBSPY> plot
```

[1] 有些地方不是很准确。

```
OBSPY> plot
OBSPY> funcgen
OBSPY> plot
```

1. funcgen 的默认值为 funcgen impulse npts 100 delta 1.0 begin 0.
2. 第一个 funcgen 命令没有使用任何选项和参数，其直接使用系统默认值，生成一个脉冲数据，并保存到内存中。该数据的起始时间为 0，采样周期为 1.0，数据点数为 100
3. plot 命令会打开一个绘图窗口，并将内存中的数据绘制在窗口中
4. 第二个 funcgen 命令生成了一个 step 函数[1]，并设置其采样周期为 0.1，数据点数为 1000
5. 0.1 和 1000 分别成为 delta 和 npts 的“参数当前值”
6. 第三个 funcgen 命令生成了 boxcar 函数，从绘图结果可以看出 delta 的值为 0.1，npts 的值为 1000，即继承了上一次命令的参数值

2.6　文档约定

约定这个事情，说起来容易做起来难，遇到不符合约定的地方只能靠读者自己领悟了。

语法约定

1. 命令和选项使用大写字母，参数使用小写字母
2. 命令和选项均使用全称，简写形式可省略的部分用灰色表示
3. “[XXX]”表示中括号内的 XXX 为可选项
4. “A|B|C”表示可以在 A、B、C 中任选一项

示例如下：

```
BANDPASS [BUTTER|BESSEL|C1|C2] [CORNERS v1 v2] [NPOLES n] [PASSES n]
    [TRANBW v] [ATTEN v]
```

需要特别说明的是，命令语法中选项的简写形式是在保证不产生歧义下的前提下所允许的**最简 形式**。本例中，CORNERS 的最简形式为首字符 C ，用户也可以使用 CO、COR 等来表示 CORNERS。

示例约定

1. 命令、选项、参数均使用小写字母
2. 常见的命令和选项均使用简写表示
3. 含有提示符“OBSPY>”的行是用户键入的命令，无提示符的行是 OBSPY 输出行；
4. OBSPY 输出行中可能会删除一些不重要的信息；
5. 示例中加入注释以帮助用户理解，注释使用了 C 语言的行注释符号“//”；
6. 命令长度过长时会被拆分成多行，每一行的行尾会加上续行符“\”，但需要注意，OBSPY 中不 能使用续行符；
7. 示例中若出现“...”，表示省略了一堆对数据的处理流程；
8. 除非上下文说明，否则每个例子都运行在单独的 OBSPY 会话中，即每个命令都省略了启动 Obspy
   和退出 Obspy 的命令；
9. 除特别情况外，均省略 plot 命令，用户应该学会随时 plot 以查看当前内存中的波形结果；

示例如下：

[1] 注意：内存中的脉冲函数已经没了。

```
$ obspy                          // 该行省略
OBSPY> fg                        // 这是注释
OBSPY> p                         // 该行省略
OBSPY> lh oo = -
4.143000e+01                     // OBSPY 输出行
OBSPY> q
```

2.7 样本数据

想要学习 OBSPY，手头必须有 OBSPY 格式的数据，OBSPY 提供了两个命令可以用于生成 OBSPY 格式 数据，分别是 `funcgen` 和 `datagen`。

2.7.1 funcgen

`funcgen`（简写为 `fg`）表示 “function generator”，即该命令可以生成一些特定的函数，比如脉冲、阶跃、正弦等等，还可以生成一个地震波形样本：

```
OBSPY> fg impulse     // 生成脉冲函数
```

上面的命令生成了一个脉冲函数并存储在 OBSPY 的内存中，可以用命令 `plot`（写为 `p`）在图形界面 上查看这个函数的样子：

```
OBSPY> p
```

在学习 OBSPY 的过程中，`funcgen` 可以生成地震波形样本：

```
OBSPY> fg seismogram // 生成地震波形样本，简写为 fg seis
```

这个命令在 OBSPY 内存中产生了一个地震波形样本，同时删除了内存中刚才生成的脉冲信号，可以 使用 `plot` 命令查看地震波形。这个地震波形样本在以后的教程中经常用到。

2.7.2 datagen

`datagen`（简写为 `dg`）表示 “data generator”。顾名思义，就是用来生成数据的。 下面的示例在内存中生成了 CDV 台站记录到的一个近震的三分量波形数据[1]，并用 `plot1`（简写 `p1`）将三个波形画在一张图上：

```
OBSPY> dg sub local
OBSPY> p1
```

更多示例参考 `datagen` 命令的语法说明。

2.8 OBSPY 的读和写

OBSPY 的读命令是 `read`（简写为 `r`），写命令为 `write`（简写为 `w`）。读和写是紧密联系的，所以把 这两者放在一起讲。

注意：本节的所有示例都运行在同一个 OBSPY 会话中。

[1] 101.4 的软件包中没有自带波形数据，因而无法使用该命令。

要演示如何读 OBSPY 文件，首先得有一些 OBSPY 数据才行，利用上一节提到的 datagen 生成一 些数据。

```
$ ls                    // 空文件夹
$ ob                    // 启动一个 OBSPY
 SEISMIC ANALYSIS CODE [11/11/2013 (Version 101.6a)]
 Copyright 1995 Regents of the University of California

OBSPY> dg sub local cdv.n cdv.e cdv.z // 生成三个 OBSPY 数据
OBSPY> w cdv.n cdv.e cdv.z          // 将 OBSPY 数据写入磁盘
OBSPY> ls                           // OBSPY 中调用常见的系统
命令 cdv.e cdv.n cdv.z
                                    // 注意：这里并没有退出
```

有了数据之后，就可以练习如何去读了，在读数据之前，先说一说通配符的概念。

OBSPY 中，在指定文件名的时候，可以使用绝对路径，也可以使用相对路径。可以使用其全名，也可以使用通配符。OBSPY 的通配符与 Unix 定义的通配符一致，只包含如下三种：

1. “*” 匹配任意长度的字符串（包括零长度）；
2. “?” 匹配任意单个非空字符；
3. “[]” 匹配列表中的任意单一字符；
   - “[ABC]” 匹配单个字符 A 或 B 或 C
   - “[A,B,C]” 匹配单个字符 A 或 B 或 C
   - “[0-9]” 匹配任意一位数字
   - “[a-g]” 匹配从 a 到 g 范围内的任意单个字符 下面的例子展示了如何读

取 OBSPY 文件：

```
OBSPY> r cdv.n cdv.e cdv.z // 读入三个文件，分别指定其文件名
OBSPY> r cdv.?          // 问号可以匹配单个字符。
./cdv.e ...cdv.n ...cdv.z // 注意！这里文件按照字符排序的顺序读入
OBSPY> r cdv.[nez]      // 还可以这样读
./cdv.e ...cdv.n ...cdv.z
OBSPY> r *              // 也可以这样读
./cdv.e ...cdv.n ...cdv.z
```

需要注意的是，OBSPY 在执行读取命令时，会读入新的波形数据，并删除内存中原有的波形数据，所以经过上面四次 read 之后，内存中依然只有三个波形。

如果想要在读取新数据时将波形直接追加到内存中的波形数据集之后，而不替换内存中的原有 波形，则需要使用 read 的 more 选项[1]：

```
OBSPY> r ./cdv.n        // one file     3 -> 1
OBSPY> r                // one MORE file 1 -> 2
more ./cdv.e            // one MORE file 2 -> 3
```

将数据读入到内存中之后，对内存中的数据做一些处理，然后就需要将内存中的数据写回到磁 盘中：

[1] 在执行完上面的例子之后，内存中有三个 OBSPY 文件，所以本例在执行 read 命令时内存中的文件数由三个变成 1 个。

```
OBSPY> w test.n test.e test.z // 分别写入到三个新文件中
OBSPY> w over                 //  覆盖磁盘原文件
OBSPY> w append .new          // 在原文件名的基础上加上后缀".new"
cdv.e.new cdv.n.new cdv.z.new
OBSPY> ls
cdv.e cdv.e.new cdv.n cdv.n.new cdv.z cdv.z.new tesn.n test.e test.z
OBSPY> q                      // 退出本节的 OBSPY 会话
```

2.9 绘图

OBSPY 中有四个常用的绘图命令，分别是 `plot`、`plot1`、`plot2`、`plotpk`。这一节只介绍最基础的 `plot` 命令，其他的命令及更多的绘图功能将在 OBSPY 图像 中说明。

`plot` 命令会在单个图形窗口中显示单个波形：

```
OBSPY> r
cdv.[nez] OBSPY>
p
Waiting
Waiting
```

上面的示例中，首先将三个波形数据读入内存，然后使用 `plot` 命令绘图，此时焦点位于绘图窗口，且绘图窗口中只显示第一个波形，终端中出现 “Waiting” 字样；将焦点切换 [1] 回终端，敲击回车 键，绘图窗口中显示第二个波形，终端中出现第二个 “Waiting” 字样，焦点位于终端中；再次敲击回 车键，窗口中显示第三个波形，焦点位于终端，由于已经没有更多的波形需要显示，此时终端中显示 OBSPY 提示符。

如果内存中还有波形在 “Waiting”，而你想要退出 plot，不想要再继续查看后面的波形，可以在终端中键入 `kill`（简写为 `k`），即可直接退出 plot，如下例：

```
OBSPY> r
cdv.[nez] OBSPY>
p
Waitingk
```

[1] Linux 下的快捷键是 Alt+Tab。

第 3 章　OBSPY 文件格式

3.1　OBSPY 格式简介

一个地震波形数据包含了时间上连续的一系列数据点，数据点可以是等间隔或不等间隔采样。OBSPY 的数据格式要求一个文件中只包含一个地震波形数据，这样的定义更适合单个地震波形的处理。

每个 OBSPY 文件包含两个部分：一个头段区和一个数据区。头段区位于每个文件的起始处，其大小是固定的，用于描述数据的相关信息，比如数据点数、采样周期等等。

数据区紧跟在头段区之后，数据区又包含了一个或多个子数据区：

- 如果数据是时间序列，且是等间隔采样的，则只有一个子数据区，包含因变量（Y，也就是数据）的值，因变量（X）的信息可以直接从头段区中获得；
- 如果数据是时间序列，但是不等间隔采样的，则有两个子数据区，分别包含因变量（Y）和自变量（X）的值；
- 如果数据是谱数据而非时间序列，则有两个子数据区，分别包含振幅和相位或者实部和虚部；
- 如果数据是三维数据（XYZ），则包含 `nxsize*nysize` 个子数据区。最经常使用的数据是等间隔采样的数据，即文件中只有一个头段区和一个数据区。

3.2　两种数据形式

OBSPY 文件格式有两种形式：二进制型和字符型 [1]。字符型与二进制型是完全等价的，只是字符型是给人看的，二进制型是给机器读写的。从 C 程序的角度来看，两者的区别在于，写文件时前者使用 `fprintf` 后者使用 `fwrite`。

二进制型的 OBSPY 数据，占用更小的磁盘空间，读写速度更快，因而是最常用的 OBSPY 格式形式。当文件出现问题时，字符型数据便于查看文件内容。

通常，字符型的 OBSPY 文件以 `ASC`[2] 结尾，二进制型的 OBSPY 文件以后缀 `OBSPY` 结尾。但是，OBSPY 在读取文件时是不判断文件后缀的，所以文件后缀是什么并不重要，在某些情况下，会以 `BHE`、`BHN`、`BHZ` 这样的后缀结尾。在下面的示例中，很多 OBSPY 文件甚至没有后缀。

3.2.1　两种形式的互相转换

你是否想要一个字符型的 OBSPY 文件，用编辑器打开好好看看 OBSPY 数据究竟长什么样？OBSPY 自带的命令可以实现两种形式的转换 [3]。

[1] 原为 alphanumeric，译为文数字。

[2] ASCII 的简写。

[3] 也可以使用 OBSPY 的 convert 命令进行转换，不过此命令即将被淘汰，因而这里不会介绍。

```
OBSPY> fg
seis                    //  先生成一个二进制型 OBSPY 数据，以做
```

将二进制型转换成字符型：

```
OBSPY> r                // 读二进制型文件
OBSPY> w alpha seis.a   // 以字符型
```

将字符型转换成二进制型：

```
OBSPY> r alpha seis.a   // 读字符型文件
OBSPY> w obspy seis.b // 以二进制型写入，可以省略 obspy，写成 w seis.b
```

试试用你最喜欢的文本编辑器打开字符型的 `seis.a` 吧，其内容如下：

```
     0.01000000      -1.569280       1.520640      -12345.00      -12345.00
       9.459999       19.45000      -41.43000       10.46400      -12345.00
      -12345.00      -12345.00      -12345.00      -12345.00      -12345.00
      -12345.00      -12345.00      -12345.00      -12345.00      -12345.00
      -12345.00      -12345.00      -12345.00      -12345.00      -12345.00
      -12345.00      -12345.00      -12345.00      -12345.00      -12345.00
      -12345.00       48.00000      -120.0000      -12345.00      -12345.00
       48.00000      -125.0000      -12345.00       15.00000      -12345.00
      -12345.00      -12345.00      -12345.00      -12345.00      -12345.00
      -12345.00      -12345.00      -12345.00      -12345.00      -12345.00
       373.0627       88.14721       271.8528       3.357465      -12345.00
      -12345.00    -0.09854718       0.000000       0.000000      -12345.00
      -12345.00      -12345.00      -12345.00      -12345.00      -12345.00
      -12345.00      -12345.00      -12345.00      -12345.00      -12345.00
      1981        88        10        38        14
         0         6         0         0      1000
    -12345    -12345    -12345    -12345    -12345
         1        50         9    -12345    -12345
    -12345    -12345        42    -12345    -12345
    -12345    -12345    -12345    -12345    -12345
    -12345    -12345    -12345    -12345    -12345
         1         1         1         1         0
CDV      K8108838
-12345  -12345  -12345
-12345  -12345  -12345
-12345  -12345  -12345
-12345  -12345  -12345
-12345  -12345  -12345
-12345  -12345  -12345
-12345  -12345  -12345
    -0.09728001    -0.09728001   -0.09856002    -0.09856002   -0.09728001
    -0.09600000    -0.09472002   -0.09344001    -0.09344001   -0.09344001
```

```
33      -0.09344001   -0.09344001   -0.09472002   -0.09472002   -0.09344001
34      ......
```

第 1–30 行是头段区，31 之后的行是数据区。目前你可能还看不懂头段区的这些数字或者字符代表什么。没关系，在下一节会详细介绍 OBSPY 头段区，记得一定要一边看下一节的内容，一边对照 着这个例子，好好琢磨每一个头段的含义。

3.3　OBSPY 头段结构

OBSPY 头段区包含了一系列头段变量，从这些头段变量中可以了解到波形记录的很多信息，比如 台站经纬度、发震时刻、震相到时等等。表 3.1 列出了 OBSPY 头段区的全部头段变量。

表 3.1: OBSPY 头段变量列表

Byte	Type	Names				
0	F	delta	depmin	depmax	scale	odelta
20	F	b	e	o	a	internal
40	F	t0	t1	t2	t3	t4
60	F	t5	t6	t7	t8	t9
80	F	f	resp0	resp1	resp2	resp3
100	F	resp4	resp5	resp6	resp7	resp8
120	F	resp9	stla	stlo	stel	stdp
140	F	evla	evlo	evel	evdp	mag
160	F	user0	user1	user2	user3	user4
180	F	user5	user6	user7	user8	user9
200	F	dist	az	baz	gcarc	internal
220	F	internal	depmen	cmpaz	cmpinc	xminimum
240	F	xmaximum	yminimum	ymaximum	unused	unused
260	F	unused	unused	unused	unused	unused
280	N	nzyear	nzjday	nzhour	nzmin	nzsec
300	N	nzmsec	nvhdr	norid	nevid	npts
320	N	internal	nwfid	nxsize	nysize	unused
340	I	iftype	idep	iztype	unused	iinst
360	I	istreg	ievreg	ievtyp	iqual	isynth
380	I	imagtyp	imagsrc	unused	unused	unused
400	I	unused	unused	unused	unused	unused
420	L	leven	lpspol	lovrok	lcalda	unused
440	K	kstnm	kevnm*			
464	K	khole	ko	ka		
488	K	kt0	kt1	kt2		
512	K	kt3	kt4	kt5		
536	K	kt6	kt7	kt8		
560	K	kt9	kf	kuser0		
584	K	kuser1	kuser2	kcmpnm		
608	K	knetwk	kdatrd	kinst		

整个头段区，共有头段变量 133 个，占 632 个字节。头段区的前四个字节是第一个头段变量 delta，第 5–8 个字节是第二个头段变量 depmin，第 21–24 个字节是第 6 个头段变量 b，以此类推。

表的第一列给出了当前行的第一个头段变量在文件中的起始字节，第二列给出了当前行的头段变量的变量类型。

表 3.2 列出了 OBSPY 头段中的头段变量类型及其相关信息。第一列为头段变量类型代码，第二 类给出了其代表的头段变量类型，第三列指出 C 源码中该变量的是用什么类型定义的，第四列给出 了每个变量所占据的字节数，第五列给出了写字符型 OBSPY 文件时的输出格式，最后一列则给出该 类型的未定义值。

表 3.2: 变量类型说明

Code	Type	C Type	sizeof	printf	未定义值
F	浮点型	float	4	%15.7f	-12345.0
N	整型	int	4	%10d	-12345
I	枚举型	int	4	%10d	-12345
L	逻辑型	int	4	%10d	FALSE
K	字符型	char*	8	%-8.8s	"-12345␣"
A	辅助型				

说明：

- 所有头段变量的变量名均以变量类型码开头，比如 nvhdr 是 N 型变量，leven 是 L 型变量，但 F 型变量除外；
- 枚举型变量本质上是 int 型，只能在固定的几个值中取值
- 逻辑型变量可以取值 TRUE 和 FALSE，本质上分别是整型的 1 和 0
- 字符型变量长度为 8[1]，只有 kevnm 很特殊，其长度为 16；
- 变量名为 internal 表示该变量是 OBSPY 内部使用的头段变量，用户不可对其进行操作；
- 变量名为 unused 表示该变量尚未使用，为以后可能出现的新头段变量占位；
- 当某个头段变量未定义时，其包含未定义值；不同类型的头段变量有不同的未定义值；若一个整型头段变量的值为 -12345，则认为该变量未定义；实际使用时，可以直接用 undef 表示所有类型的头段变量的未定义值，OBSPY 会根据头段变量的类型自动将其转换成相应类型的未定 义值。
- 辅助型变量并不在 OBSPY 头段区中，而是从其它头段变量推导得到的；

3.4 OBSPY 头段变量

3.4.1 基本变量

nvhdr*

OBSPY 头段版本号。nvhdr[2] 是 OBSPY 中很重要但是不太常用的头段变量。目前版本号为 6，旧版 本的 OBSPY 文件（nvhdr<6）在读入时头段区会自动更新。

[1] C 语言中用 "\0" 作为字符串的结束标识符，因而源码中变量的实际长度为 9。

[2] 星号表示该头段变量在 OBSPY 中必须有定义值，下同。

nzyear, nzjday, nzhour, nzmin, nzsec, nzmsec

分别表示“年”、“一年的第几天”[3][4]、“时”、“分”、“秒”、“毫秒”[5]。这六个头段变量构成了 OBSPY 中唯一的绝对时刻，OBSPY 中的其它时刻都被转换为相对于该时刻的相对时间（单位为秒）。关于 OBSPY 中的绝对时间和相对时间的概念，参考“OBSPY 中的时间概念”一节。

根据这六个头段变量还可以推导出其它一些辅助型头段变量：

- `kzdate`：字符数字格式的参考日期，由 `nzyear` 和 `nzjday` 导出
- `kztime`：字符数字格式的参考时间，由 `nzhour`、`nzmin`、`nzsec`、`nzmsec` 导出

如下例所示：

```
OBSPY> fg seis
OBSPY> lh nzyear nzjday nzhour nzmin nzsec nzmsec

    nzyear = 1981
    nzjday = 88
    nzhour = 10
     nzmin = 38
     nzsec = 14
    nzmsec = 0
  OBSPY> lh kzdate
       kztime

    kzdate = MAR 29 (088), 1981
```

iztype

等效参考时刻。OBSPY 的参考时刻是可以任意指定的，但一般选取某个特定的时刻（比如文件起始时刻、发震时刻等等）作为参考时刻。其可以取如下枚举值[1]：

- `IUNKN`：未知
- `IB`：以文件开始时刻为参考时间
- `IDAY`：以参考日期当天的午夜作为参考时间
- `IO`：以事件发生时间为参考时间
- `IA`：以初动到时为参考时间
- `ITn`：以用户自定义的时间 `Tn` 为参考时间（n 可取 0–9）

若 `iztype=IO`，则表示数据以发震时刻作为参考时刻，此时头段变量 o 的值应为 0。

iftype*

OBSPY 文件类型，其决定了头段区之后有几个子数据区。可以取如下枚举值：

- `ITIME`：时间序列文件（即 Y 数据，一般的地震波形数据）
- `IRLIM`：频谱文件（实部-虚部格式）
- `IAMPH`：频谱文件（振幅-相位格式）
- `IXY`：一般的 X-Y 数据

[3] 使用 jday 而不是“month+day”可以少用一个头段变量。

[4] 1 月 1 日对应的 nzjday 是 1 而不是 0。

[5] $1\,\mathrm{s} = 1000\,\mathrm{ms}$

[1] 枚举型在 C 源码中使用 #define 宏来定义的，比如 #define IO 11，所有可取的枚举值都以字母 I 开头。

- IXYZ：一般的XYZ（3D）文件

idep

因变量（Y）类型，该头段变量可以不定义，其可以取如下枚举值：

- IUNKN：未知类型
- IDISP：位移量，单位为 nm
- IVEL：速度量，单位为 nm/s
- IVOLTS：速度量，单位为 V[1]
- IACC：加速度量：单位为 nm/s^2

3.4.2 数据相关变量

npts*

数据点数，其值决定了在数据区有多少个数据点。

delta*

等间隔数据的数据点采样周期（标称值）。

odelta

采样周期的实际值，若实际值与标称值不同则有值，一般来说都是未定义的。

b*, e*

文件的起始时间和结束时间（相对于参考时刻的秒数）。

leven*

若数据为等间隔则为 TRUE，否则为 FALSE。

depmin, depmax, depmen

因变量（Y）的最小值、最大值和均值。

在读入 OBSPY 文件以及对数据进行处理时，这三个头段变量的值会被自动计算并更新。示例如下：

```
$ obspy
OBSPY> fg
seis OBSPY> lh
depmax                          // 最大值
    depmax = 1.520640e+00       // 强行修改数据最大值
OBSPY> ch depmax 1000           // 这是错误的示范，不要这样做
                                // 查看 depmax，修改成功
OBSPY> lh depmax 1000
    depmax = 1.000000e+03       // 写到磁盘中
OBSPY> w seis.OBSPY
$ obspylst depmax f seis.OBSPY  // 调用 obspylst 查看磁盘文件中
seis.OBS        1000            //  可以看到磁盘中的文件 depmax=1000
PY
OBSPY> r ./seis.OBSPY           // 读入 OBSPY
OBSPY> lh depmax
    depmax = 1.520640e+00       // 此时 depmax 被自动计算并更
```

[1] 不解

scale

因变量比例因子，即真实物理场被乘以该比例因子而得到现有数据。

假设真实物理场的 Y 值大概在 10^{-20} 量级，由于数据量级太小处理起来可能不太方便。此时可以将数据乘以 10^{20} 变成合适的量级，并修改 `scale=1.0e20`，这样就可以知道自己对数据人为放大了多少倍。

101.5 之前的版本中，在使用 `transfer` 命令去仪器响应时，若 `scale` 的值有定义，则输出的数据会根据该值进行放大并修改 `scale`。在 101.5 及其之后的版本中，`scale` 被忽略。

xminimum, xmaximum, yminimum, ymaximum

仅用于 3D（XYZ）文件中，记录 X 和 Y 的最小/大值。

nxsize, nysize

仅用于 3D（XYZ）文件中，表示 X 和 Y 方向的数据点数。

iqual†

iqual[1] 标识数据质量，可取如下值：

- `IGOOD`：高质量数据
- `IGLCH`：数据中有毛刺（glitches）
- `IDROP`：数据有丢失（dropouts）
- `ILOWSN`：低信噪比数据
- `IOTHER`：其它

isynth†

合成数地震图标识。

- `IRLDTA`：真实数据

3.4.3　事件相关变量

kevnm

事件名，长度为 16 个字节。

evla, evlo, evel, evdp

分别代表事件的纬度（-90 到 90）、经度（-180 到 180）、高程（单位为 m，未使用）和深度（单位为 km，以前为 m）。

ievreg†

事件地理区域 [2]。

ievtyp

事件类型，这里仅列出部分常见的枚举值：

- `IUNKN`：未知事件
- `INUCL`：核事件
- `IEQ`：地震
- `IOTHER`：其它

[1] † 标识仅表示 OBSPY 程序内部未使用该头段变量，即变量有值或者无值、有何值，对于程序的运行不会产
生任何影响，但用户可以在自己的程序中自由使用这些头段变量。下同。

[2] Flinn-Engdahl　Regions:http://en.wikipedia.org/wiki/Flinn-Engdahl_regions

mag

事件震级。

imagsrc

震级信息来源，可以取如下枚举值：

- INEIC: http://earthquake.usgs.gov/earthquakes/search/
- IPDE: http://earthquake.usgs.gov/data/pde.php
- IISC: http://www.isc.ac.uk/iscbulletin/search/catalogue/
- IREB: 人工检查过的事件目录
- IUSGS: USGS
- IBRK: UC Berkeley
- ICALTECH: California Institute of Technology
- ILLNL: Lawrence Livermore National Laboratory
- IEVLOC: Event Location
- IJSOP: Joint Seismic Observation Program
- IUSER: The individual using OBSPY2000
- IUNKNOWN: 未知

imagtyp

震级类型，取如下枚举值：

- IMB: 体波震级
- IMS: 面波震级
- IML: 区域震级
- IMW: 矩震级
- IMD: 持续时间震级
- IMX: 用户自定义震级

gcarc, dist, az, baz

- gcarc: 全称 Great Circle Arc，即震中到台站的大圆弧的长度，单位为度；
- dist: 震中到台站的距离，单位为 km；
- az: 方位角，震中到台站的连线与地理北向的夹角；
- baz: 反方位角，台站到震中的连线与地理北向的夹角。

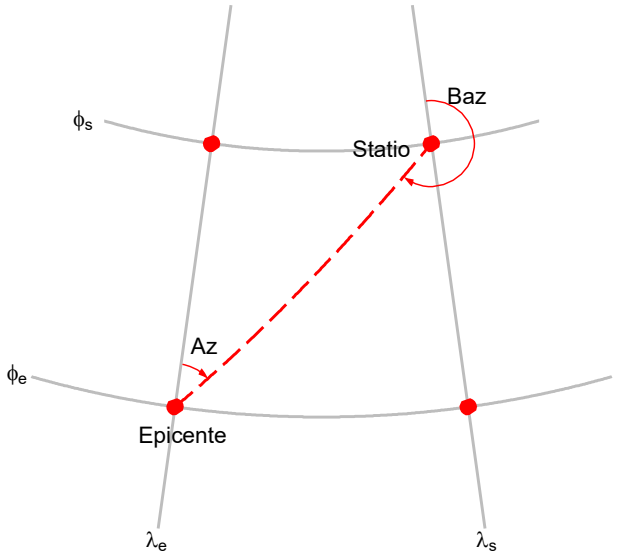

图 3.1: 震中距、方位角、反方位角示意图。 震中距、方位角和反方位

角的计算涉及到球面三角的知识，具体公式及其推导可以参考相关代码及书籍。此处列出部分仅供参考：

- http://www.eas.slu.edu/People/RBHerrmann/Courses/EASA462/
- http://www.seis.sc.edu/software/distaz/
- OBSPY 源码 `src/ucf/distaz.c`
- CPS330 源码 `VOLI/src/udelaz.c`

o, ko

o 为事件的发生时刻相对于参考时刻的秒数。ko 是绘图时时间变量 o 的标识符。

khole

若为核爆事件，则其为孔眼标识；若为其它事件，则为位置标识。

nevid, norid, nwfid

三者分别标识事件 ID、起始时间 ID 和波形 ID，仅用于 CSS 3.0 文件中。CSS 3.0 是 OBSPY 可 以处理的一种数据格式，应该是当初 OBSPY 商业化的产物，目前仍保留在 OBSPY 头段中。

3.4.4 台站相关变量

knetwk, kstnm

地震台网名和台站名。

istreg†

台站地理区域。

stla, stlo, stel, stdp

台站纬度（-90 到 90 度）、经度（-180 到 180 度）、高程（单位 m，目前未使用）、相对地表的深度（单位 m，目前未使用）。

cmpaz, cmpinc, kcmpnm, kstcmp

一个台站至少需要三个正交的通道/分量才能完整地记录地面运动物理量。`cmpaz` 和 `cmpinc` 指定了单个通道记录的方向矢量。

图 3.2 给出了 OBSPY 所使用的 NEU 坐标系，需要注意的是这是一个左手坐标系。图中蓝色箭头

为通道所记录的方向矢量，若地面运动与该方向一致，则为正，否则为负。其中，头段变量 cmpaz 表征通道的方位角，其定义为从 N 向开始顺时针旋转的角度，即图中的角度 ϕ；cmpinc 表征通道的入射角，定义为相对于 U 方向向下旋转的度数，即图中的角度 θ。

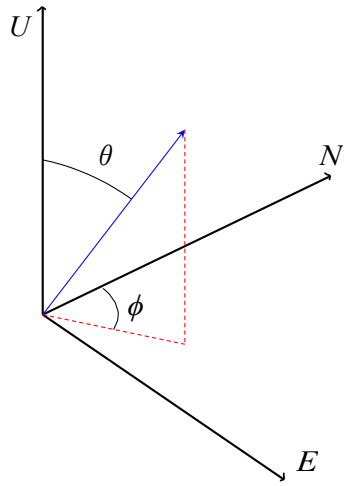

图 3.2: cmpaz 和 cpminc 示意图 根据定义，

地震仪标准通道的 cmpinc 和 cmpaz 值如下表：

表 3.3: 标准地震通道的 cmpaz 和 cpminc

方向	cmpaz	cmpinc
N	0	90
E	90	90
U	0	0

对于非标准方向的地震通道来说，很容易根据 cmpinc 和 cmpaz 的值，将其旋转到 NEU 坐标系或者 RTZ 坐标系，这些将在 “分量旋转” 一节中说到。

kcmpnm 用于存储分量名称。SEED 格式规定通道名的三个字符中的最后一个代表通道的分量方位，比如通道名 BHE 表示该通道为东西向。通常 kcmpnm 可以取为 E、N、Z。由于很多台站的水平分量并不严格是东西、南北方向，因而现在更倾向于用 1 和 2 代替 N 和 E。

kstcmp 为辅助型变量，表示台站分量，由 kstnm、cmpaz、cmpinc 推导得到。

lpspol

如图 3.2 所示，在左手坐标系下，若三通道都是正极性则为真，否则为假。

3.4.5　震相相关变量

a, f, tn

a 和 f 用于存储事件的初动时刻和结束时刻相对于参考时刻的秒数。

Tn（n=0–9）用于存储用户自定义的时刻相对于参考时刻的秒数，常用于存储震相到时。

ka, kf, ktn

a、f 以及 Tn 都有一个对应的以 k 开头的字符型头段变量，称之为时间标识。时间标识用于说明对应的时间头段变量中所包含时间的含义。

比如头段变量 a 中通常包含 P 波到时，则此时 ka 的值可以设置为 “P”；头段变量 t1 中包含了震相 PcP 的到时，则一般定义 kt1 为 “PcP”。

在绘图时，若时间头段变量中有值，则默认会在该时刻处绘制一条垂线，若相应的时间标记有 定义，则将时间标记的值显示在垂线附近。

Xmarker

震相相关的变量对可以构成一个辅助型变量。a 和 ka 可以构成 amarker，f 和 kf 可以构成 fmarker，o 和 ko 可以构成 omarker，tn 和 ktn 可以构成 tnmarker（n=0–9）。 这些辅助型变量可以在 listhdr 中使用。

3.4.6　仪器相关变量

kinst, iinst†, respn†

kinst 为记录仪器的通用名称，iinst 为记录仪器的类型，respn 为仪器相应参数。

3.4.7　其它变量

usern

usern（n=0–9）用于存储用户自定义的浮点型数值。

kusern

kusern（n=0–2）用于存储用户自定义的字符型值。

lovrok

若为 TRUE，则磁盘里的原始数据可被覆盖；若为 FALSE，则原始数据不可被覆盖。主要用于保护原始数据，一般来说很少用到，若是出于保护原始数据的目的，应优先考虑对原始数据做备份。

lcalda

全称为 Calculate Distance and Azimuth。若为 TRUE，则当事件和台站的坐标被写入或被修改时，头段变量 dist、gcarc、az、baz 将自动计算，否则不会被自动计算，OBSPY 头段中会存在信息的 不兼容。

kdatrd

数据被读入计算机的日期（一般很少使用）。

3.5　OBSPY 中的时间概念

3.5.1　基本思路

OBSPY 的头段区有很多与时间相关的头段变量，包括 nzyear、nzjday、nzhour、nzmin、nzsec、nzmsec、b、e、o、a、f、tn（n=0–9）。正确使用它们的前提是理解 OBSPY 中的时间概念。这一节将试 着说清楚这个问题。

首先，OBSPY 处理的是地震波形数据，OBSPY 格式里保存的是时间序列数据。先不管其它的一些台 站经纬度、事件经纬度信息，就数据而言，至少需要一系列数据值以及每个数据值所对应的时刻。

在本节接下来的内容中，将严格区分两个高中物理学过的概念：时刻和时间。简单地说，在时间 轴上，时刻是一个点，时间是一个线段。

一个简单的例子如下：

```
2014-02-26T20:45:00.000   0.10
```

```
2014-02-26T20:45:01.000      0.25
2014-02-26T20:45:02.000      0.33
2014-02-26T20:45:03.000      0.21
2014-02-26T20:45:04.000      0.35
2014-02-26T20:45:05.000      0.55
2014-02-26T20:45:06.000      0.78
2014-02-26T20:45:07.000      0.66
2014-02-26T20:45:08.000      0.42
2014-02-26T20:45:09.000      0.34
2014-02-26T20:45:10.000      0.25
```

其中第二列是数据点，每个数据点所对应的时刻放在第一列，格式为“yyyy-mm-ddThh:mm:ss.xxx”。数据点是以 1s 的等间隔进行采样的。

若把这堆时刻以及数据点直接写入文件中，将占据大量的磁盘空间，读写也很不方便。考虑将 某一个时刻定义为参考时刻，并把其它所有的时刻都用相对于该参考时刻的秒数来表示，这样可以 简化不少。

比如取“2014-02-26T20:45:00.000”为参考时刻，即

```
nzyear = 2014
nzjday = 57
nzhour = 20
nzmin  = 45
nzsec  = 00
nzmsec = 000
```

则上面的数据可以简化为

```
00.000     0.10
01.000     0.25
02.000     0.33
03.000     0.21
04.000     0.35
05.000     0.55
06.000     0.78
07.000     0.66
08.000     0.42
09.000     0.34
10.000     0.25
```

其中第二列是数据点，第一列是每个数据点对应的时刻相对于参考时刻的相对时间，下面简称其为 相对时间。

显然参考时刻的选取是任意的，若取“2014-02-26T20:45:05.000”为参考时刻，则上面的数据简化为

```
-05.000      0.10
-04.000      0.25
```

```
-03.000     0.33
-02.000     0.21
-01.000     0.35
 00.000     0.55
 01.000     0.78
 02.000     0.66
 03.000     0.42
 04.000     0.34
 05.000     0.25
```

一般来说，会选取一个比较特殊的时刻作为参考时刻，比如第一个数据点对应的时刻，或者地 震波形数据中的发震时刻。

下面还是回到以 “2014-02-26T20:45:00.000” 为参考时刻简化得到的结果。因为数据是等间距的，相对时间这一列完全可以进一步简化，比如用 “起始相对时间 + 采样间隔 + 数据点数” 或者 “起始相对时间 + 采样间隔 + 结束相对时间” 就完全可以表征第一列的相对时间。

OBSPY 选择了另外一种简化模式，“起始相对时间 + 采样间隔 + 数据点数 + 结束相对时间”，即 头段变量中的 “b+delta+npts+e”，这其实是存在信息冗余的，这就造就了头段变量 e 的一些特殊 性，后面会提到。

按照 OBSPY 的模式在对相对时间进行简化之后，整个数据可以表示为

```
nzyear = 2014
nzjday = 57
nzhour = 20
nzmin  = 45
nzsec  = 00
nzmsec = 000
b      = 0.0
e      = 10.0
delta  = 1.0
npts   = 11

0.10
0.25
0.33
0.21
0.35
0.55
0.78
0.66
0.42
0.34
0.25
```

似乎到这里就结束了。 地震学里的一个重要问题是拾取震相到时（时刻），所以还需要几个额外的头段变量来保存这

些震相到时（时刻），不过显然不需要真的把 “时刻” 保存到这些头段变量中，不然上面的一大堆就

真是废话了。OBSPY 将震相到时（时刻）相对于参考时刻的时间差（即相对时间）保存到头段变量 o、a、f、tn 中。

综上，OBSPY 中跟时间有关的概念有三个：

参考时刻 由头段变量 nzyear、nzjday、nzhour、nzmin、nzsec、nzmsec 决定

相对时间 即某个时刻相对于参考时刻的时间差（单位为秒），保存到头段变量 b、e、o、a、f、tn（n=0–9）

绝对时刻 = 参考时刻 + 相对时间

3.5.2 一些测试

下面以一个具体的数据为例，通过修改各种与时间相关的头段来试着去进一步理解 OBSPY 的时 间概念。

生成样例数据

```
OBSPY> fg
seis OBSPY>
lh iztype
    iztype = BEGIN TIME
OBSPY> ch iztype IUNKN
```

lh 是命令 listhdr 的简写，用于列出头段变量的值。ch 是 chnhdr 的简写，用于修改头段变量的值。这里额外多做了一个操作修改 iztype 的操作，这是由于这个数据稍稍有一点 bug。

iztype 指定了参考时刻的类型，其显示为 BEGIN TIME，实际上其枚举值是 IB，也就是说这个数据选取文件第一个数据点的时刻作为参考时刻，那么 b 的值应该为 0。而实际上这个数据的 b 值并不为 0，这其实是这个数据的一点小 bug。这也从另一个侧面说明 OBSPY 只有在修改与时间相关的 头段变量时才可能会检查到这个错误/警告，所以这里先将其修正为 IUNKN。

修改文件起始时间 b

```
OBSPY> r seis
OBSPY> lh kzdate kztime b delta npts e o a f

     kzdate = MAR 29 (088), 1981
     kztime = 10:38:14.000
          b = 9.459999e+00
      delta = 1.000000e-02
       npts = 1000
          e = 1.945000e+01
          o = -4.143000e+01
          a = 1.046400e+01
OBSPY> ch b
10 OBSPY> lh

     kzdate = MAR 29 (088), 1981
     kztime = 10:38:14.000
          b = 1.000000e+01
      delta = 1.000000e-02
       npts = 1000
          e = 1.999000e+01
          o = -4.143000e+01
```

```
        a = 1.046400e+01
```

修改 b 前后的变化仅在于 b 和 e 值的变化，而参考时刻以及其它相对时间并没有发生变化。

这意味着整段 OBSPY 数据中的任意一个数据点所对应的时刻[1] 都向后延迟了 0.54 秒！这样做很
危险，因为 b 和 e 的绝对时刻被修改了，而其它头段如 o、a、f、tn 的绝对时刻却没有变。

使用的时候必须非常小心：

- 如果 o、a、f、tn 都没有定义，那么修改 b 值可以用于校正仪器的时间零飘[2] 以及时区差异[3]。关于时区的校正，参考 “时区校正” 一节。
- 如果 o、a、f、tn 已经被定义，则修改 b 值会导致与震相相关的头段变量出现错误！[4]

修改文件结束时间 e

```
OBSPY> r ./seis
OBSPY> lh kzdate kztime b delta npts e o a f

     kzdate = MAR 29 (088), 1981
     kztime = 10:38:14.000
          b = 9.459999e+00
      delta = 1.000000e-02
       npts = 1000
          e = 1.945000e+01
          o = -4.143000e+01
          a = 1.046400e+01
OBSPY> ch
e 0 OBSPY>
lh

     kzdate = MAR 29 (088), 1981
     kztime = 10:38:14.000
          b = 9.459999e+00
      delta = 1.000000e-02
       npts = 1000
          e = 1.945000e+01
          o = -4.143000e+01
```

可以看到，修改前后所有变量均没有发生变化，即 e 的值是不可以随意改变的，根据上面的结果可知，e 的值是通过 b、delta、npts 的值动态计算的。这也与上一节说到的头段变量冗余问题相符合。不要试图修改 delta、npts，这不科学！

修改 o、a、f、tn

这几个头段变量完全是由用户自定义的，因而任何的定义、修改、取消定义都不会对数据的正 确性产生影响，因而这里不再测试。

---

[1] 好长的修饰语

[2] 零飘，即仪器中的时刻与标准时刻不同。

[3] 时区差异可以理解成另一种零飘。

[4] 如果只定义了 o 值，或者 a、f、tn 为理论震相到时而非计算机拾取或人工拾取的到时，修改 b 也是没有问题的。有些乱，不多说了。总之不要随便修改 b 的值。

修改参考时间

```
OBSPY> r ./seis
OBSPY> lh kzdate kztime b delta npts e o a f

     kzdate = MAR 29 (088), 1981
     kztime = 10:38:14.000
          b = 9.459999e+00
      delta = 1.000000e-02
       npts = 1000
          e = 1.945000e+01
          o = -4.143000e+01
          a = 1.046400e+01
OBSPY> ch nzsec
15 OBSPY> lh

     kzdate = MAR 29 (088), 1981
     kztime = 10:38:15.000
          b = 9.459999e+00
      delta = 1.000000e-02
       npts = 1000
          e = 1.945000e+01
          o = -4.143000e+01
          a = 1.046400e+01
```

试图修改参考时刻，整个 OBSPY 头段，除了参考时刻外其它时间变量都没有发生变化。根据“绝 对时刻 = 参考时刻 + 相对时间” 可知，这导致所有 OBSPY 数据点的绝对时刻发生了平移，这一点理 论上可以用于校正零飘或者时区，但是由于 OBSPY 不支持智能判断时间（比如不知道 1 时 80 分实际 上是 2 时 20 分），所以修改时区时需要获取参考时刻 6 个头段变量，加上时区的校正值，再写入到 参考时刻 6 个变量中，相对较为繁琐，因而若要校正时区，建议直接修改头段变量中的 b 值。

修改发震时刻

数据处理中一个常见的需求是修改发震时刻，这可以通过修改头段变量 o 来实现，但是经常需要将参考时刻设置为发震时刻。上面的测试表明，直接修改参考时刻是很危险的，所以 OBSPY 的 ch 命令提供了 allt 选项来实现这一功能，在“事件信息”一节中会具体解释。

3.5.3 总结

将 OBSPY 中的时间变量分为三类：

1. 参考时刻：即 nzyear、nzjday、nzhour、nzmin、nzsec、nzmsec；
2. 相对时间：即 o、a、f、tn；
3. 特殊的相对时间：即 b[1]； 第二类时间变量可以随意修改，即震相拾取。

第一、三类时间变量的修改会导致数据绝对时刻发生改变。一般通过修改第三类时间变量来校 正时间零漂和时区差异。在设置了发震时刻后，应使用 chnhdr 命令的 allt 选项修改第一、三类时 间变量。

[1] 由于 e 不可独立修改，所以不再考虑

第 4 章　OBSPY 数据处理

地震数据处理的流程大致为：数据组织 ➔ 数据预处理 ➔ 人机交互 ➔ 数据分析 ➔ 绘制图件。本章将介绍如何高效地组织数据，以及如何利用 OBSPY 进行数据预处理和人机交互。

本章的各节分别介绍了数据处理流程中的各个小步骤，各节的顺序尽量按照数据处理的流程排 序。但实际数据处理的流程是由具体的研究来决定的，因而本章的数据处理流程仅供参考，读者应 理解每个步骤的含义及作用，根据实际情况决定要采用哪些步骤以及各个步骤之间的先后顺序。

4.1　数据申请

申请地震波形数据，按照具体的需求，大致可以分为两大类，即事件波形数据和连续波形数据。 这两者的主要区别在于如何确定要申请的数据时间窗，前者需要震相到时信息，后者则不需要。

4.1.1　事件波形数据

1. **筛选事件**：从地震目录中找出某个特定事件的信息，或根据条件筛选出需要的事件，筛选条件包括：经纬度范围、深度范围、时间范围、震级范围等
2. **筛选台站**：台站位置、仪器类型（宽频段、长周期或短周期）、分量类型（三分量或 Z 分量）
3. **确定起始和结束时间**：根据发震时刻和震中距信息，计算震相理论到时，由此确定要申请的数据时间窗

事件波形数据又可以进一步分成两类：以事件为中心的事件波形数据（比如地震定位、震源机 制）和以台站为中心的事件波形数据（比如接收函数）。对于以事件为中心的事件波形数据，通常先 筛选地震事件，在筛选台站时，还可以加上震中距范围和方位角范围的约束；对于以台站为中心的 事件波形数据，通常先确定要使用的台站，在筛选地震事件时，还可以加上震中距和反方位角范围 的约束。

4.1.2　连续波形数据

连续波形数据中没有地震事件的信息，因而都是以台站为中心的，比如背景噪声相关。

1. **选择台站**：台站位置、仪器类型、分量类型
2. **确定时间范围**：选择合适的时间范围

当然也可以根据地震事件的信息，确定申请数据的时间范围，所以，某种程度上，事件波形数据属于 连续波形数据的一个子集。

4.2　数据格式转换

4.2.1　数据格式

通常，地震数据以 SEED 格式进行保存和传输。SEED[1] 即 Standard for the Exchange of Earthquake Data，其可以存储多台站多分量的波形数据以及台站元数据 [2]。SEED 格式本质上是一个压缩 格式，因而可以大大减少网络传输的数据量以及硬盘空间，同时又可以通过特定的软件将其中的波形数据解压成常见的地震数据格式，也可以将从台站元数据中提取出仪器响应信息。

除了 SEED 格式，还有 miniSEED 格式和 dataless SEED 格式。miniSEED 格式中仅包含波形数据，dataless SEED 格式中仅包含台站元数据。之所以要将 SEED 格式拆分成 miniSEED 和 dataless SEED，是因为若每个 SEED 文件中都包含台站元数据，会造成台站元数据的冗余，浪费网络资源及硬盘容量。

除了 SEED 格式之外，还有其他数据格式，比如为数据库设计的 CSS 3.0 格式，以及众多数据处理软件自定义的格式，如 OBSPY、AH、evt 等等。不同国家的台网也可能会自定义自己的数据格式，比如日本 Hi-net 台网的数据使用自己定义的 win32 格式。

4.2.2　格式转换

IRIS 提供了 rdseed 软件，用于提取 SEED 数据中的连续波形数据以及台站元数据，并可将连续波形数据保存为多种地震数据格式。

下面的命令可以从 SEED 数据中提取 OBSPY 格式的波形数据，以及台站的 RESP 仪器响应文 件：

```
$ rdseed -Rdf file.seed
```

下面的命令可以从 SEED 数据中提取 OBSPY 格式的波形数据，以及台站的 PZ 仪器响应文件：

```
$ rdseed -pdf file.seed
```

4.3　合并数据

相关命令：merge

相关脚本：Perl 脚本、Python 脚本

有些时候，从 SEED 数据中解压出来的同一台站同一分量的连续波形数据，会被分割成多个等长或不等长的文件，数据断开的可能原因是仪器在某些时刻存在问题导致连续数据出现间断，也可 能是出于其它考虑将数据进行切割。此时需要首先对数据进行合并。

假定解压出来的台网 NET、台站 STA、位置为 00 的 BHZ 分量的连续波形被分割成了多个文件，需要将多个文件合并成单个文件。

对于 v101.6 之前的版本，只能用如下命令进行合并：

```
OBSPY> r 2012.055.12.00.00.0000.NET.STA.00.BHZ.Q.OBSPY
OBSPY>                                              merge
2012.055.12.25.00.0000.NET.STA.00.BHZ.Q.OBSPY       OBSPY>
merge        2012.055.12.40.00.0000.NET.STA.00.BHZ.Q.OBSPY
OBSPY> ...
```

[1] SEED 格式的详细说明参考官方文档：www.fdsn.org/seed_manual/SEEDManual_V2.4.pdf。

[2] 台站元数据（Metadata）中包含了台站相关的全部信息，比如台站位置、分量信息、仪器响应等。

```
OBSPY> w NET.STA.00.BHZ
```

即先读取第一段数据，然后合并第二段数据，再合并第三段数据。对于多个数据段的合并，需要执行 多次合并命令，且合并时文件的顺序必须按照绝对时间的先后顺序。

OBSPY 从 v101.6 开始重写了 `merge` 命令，可以使用如下更简洁的形式：

```
OBSPY> r *.NET.STA.00.BHZ   // 读入所有需要合并的文件
OBSPY> merge                // 内存中的所有文件被合并为一个文件
OBSPY> w NET.STA.00.BHZ     //  写回到磁盘中
```

对于所有要合并的数据文件，OBSPY 会检测 `knetwk`、`kstnm`、`kcmpnm` 和 `delta` 是否完全匹配，并 智能判断每个文件的合并顺序。

实际合并的过程中，可能会出现数据间断或数据重叠的情况。若数据存在间断，可对其直接补 零或线性插值；若数据存在重叠，则可以比较重叠部分数据是否相同或对重叠的波形进行平均。

4.4　数据重命名

相关脚本：Perl 脚本、Python 脚本

用 rdseed 软件从 SEED 格式中解压得到的 OBSPY 数据，一般都具有固定格式的文件名。示例如
下：

```
2012.055.12.34.56.7777.YW.MAIO.01.BHE.Q.OBSPY
2012.055.12.34.50.6666.YW.MAIO.01.BHN.Q.OBSPY
2012.055.12.34.54.5555.YW.MAIO.01.BHZ.Q.OBSPY
```

这三个文件是 YW 台网 MAIO 台站的宽频地震仪记录的宽频带三分量（BHE、BHN、BHZ）波形数据。文件名中每一项的具体含义在“数据命名规则”中有介绍，这里不再重复。

默认的文件名比较长，在数据处理时可能会显得比较麻烦，一般都会根据实际需求进行适当的 简化。

是否要对数据文件做重命名，以及按照什么格式重命名，都是没有固定的标准的。通常需要用 户根据自己所做研究的实际情况来决定。

在某些情况下，需要将同一事件在所有台站的波形数据放在同一个文件夹下，并将文件名以事 件的发生日期/时间来命名。那么，OBSPY 文件名中的时间等信息就可以被省略掉。数据文件名简化 为：

```
YW.MAIO.01.BHE
YW.MAIO.01.BHN
YW.MAIO.01.BHZ
```

有时候，需要将不同事件在同一个台站的波形数据放在同一个文件夹下，并将文件名以台站名 来命名，此时数据文件名中可能需要保留事件的日期信息。数据文件名可以简化为：

```
YW.MAIO.01.20120224.BHE
YW.MAIO.01.20120224.BHN
YW.MAIO.01.20120224.BHZ
```

数据重命名这一步骤可以单独执行，也可以在执行其他操作的过程中顺便进行重命名（比如将 数据合并并写入磁盘的时候）。通常需要写脚本来完成重命名的操作。

4.5 时区校正

假设有一个数据文件，数据中的时间都是中国时间，即东八区时间，现想将数据修改至国际标 准时间，即要对数据做时区校正，将数据的绝对时间整体减少 8 个小时。前面说过，时区校正可以通过修改头段变量 b 的值来实现。

```
OBSPY> r nykl.z                          // 读入数据
OBSPY> lh b e kzdate kztime              // 查看头段信息

          b = 1.999622e+02               // b=200
          e = 1.600968e+03               // e=1600
     kzdate = SEP 10 (254), 1984
     kztime = 03:14:07.000
OBSPY> ch b (&1,b& - 8*3600)             //  取 b 值，减去 8 小时，再赋值
给 b OBSPY> lh

          b = -2.860004e+04              // 此时 b=200-8*3600=28600
          e = -2.719903e+04
     kzdate = SEP 10 (254), 1984         // 参考时间不变
     kztime = 03:14:07.000
OBSPY> ch allt (0 - &1,b&) iztype IB // 将参考时间设置为文件起始时间
OBSPY> lh b e kzdate kztime

          b = 0.000000e+00
          e = 1.401006e+03
     kzdate = SEP 09 (253), 1984         // 中国时间的 9 月 10 日 3 时
     kztime = 19:17:26.963               // => 国际标准时间的 9 月 9 日 19 时
```

从上面的例子中可以看出，头段变量 b 的值被减去了 8 个小时，而数据的参考时间并没有改变，因而数据整体向前移动了 8 个小时，即完成了时区校正。

需要注意的是，若数据中头段 o、a、f 或 tn 这些相对时间是有定义的，则这些相对时间都会由于 b 值的修改而出错，因而时区校正要尽早做。

4.6 事件信息

相关头段：evla、evlo、evdp、mag、o、nzyear、nzjday、nzhour、nzmin、nzsec、nzmsec 相关脚本：Perl 脚本、Python 脚本

一般来说，从 SEED 连续波形中解压得到的 OBSPY 数据中是没有事件信息的。这就需要用户从 地震目录中获取事件的发震时刻、经度、纬度、深度和震级信息，并将这些信息写入到 OBSPY 文件的 头段中。OBSPY 提供了用于可以修改头段变量的命令 chnhdr，以及将修改后的头段变量写到磁盘文 件的命令 writehdr[1]。

4.6.1 经纬度、深度与震级

想要修改事件的经纬度、深度和震级，操作如下：

```
OBSPY> r cdv.?
```

[1] 也可以使用 w over 将修改写回磁盘文件。关于 wh 和 w over 的区别，参考 wh 与 w over 一节。

```
OBSPY> ch evla 37.52 evlo -121.68 evdp 5.95 // 修改三个头段变量
OBSPY> ch mag 5.0                           // 修改一个头段变量
OBSPY> wh                                   // 将修改后的头段写入文件
```

4.6.2　发震时刻

通常，需要将发震时刻信息写入 OBSPY 头段，并设置 OBSPY 文件的参考时刻为发震时刻。这样设 置的好处在于，可以直观地从 X 轴坐标上读取震相走时。要实现这一操作，需要用到 chnhdr 的两 个特殊用法。

先看看如何修改一个 OBSPY 文件的发震时刻，假设发震时刻为 1987 年 06 月 22 日 11 时 10 分 10.363 秒：

```
OBSPY> r ./cdv.z
OBSPY> ch o gmt 1987 173 11 10 10 363 // 06 月 22 日是第
173 天 OBSPY> lh kzdate kztime o

    kzdate = JUN 22 (173), 1987
    kztime = 11:09:56.363
         o = 1.400000e+01     //  发震时刻相对于参考时刻的时间为 14
秒 OBSPY> ch allt -14 iztype IO        //  参考时间加 14 秒，其他时
间减 14 秒 OBSPY> lh kzdate kztime o

    kzdate = JUN 22 (173), 1987
    kztime = 11:10:10.363
         o = 0.000000e+00
OBSPY> wh                          // 写回磁盘
```

上面的例子中，首先从地震目录中获取了地震的发震时刻，然后计算发震日期是一年中的第几天， 本例中为第 173 天，再利用 “ch o gmt yyyy ddd hh mm sss xxx” 语法将发震时刻赋值给头段 变量 o，OBSPY 会自动将发震时刻转换为相对于参考时刻的相对时间。此时 OBSPY 文件的参考时刻为 “1987-06-22T11:09:56.363”，而 o 值对应的时刻为发震时刻 “1987-06-22T11:10:10.363”，所以
头段变量 o 的值为发震时刻相对于参考时刻的时间差，即 14s。 将发震时刻写入头段之后，还需要

将参考时刻修改为发震时刻，与此同时还要修改所有的相对

时间。ch allt xx.xx 的功能是将所有已定义的相对时间加上 xx.xx 秒，同时从参考时刻中减去 xx.xx 秒，此时参考时刻即为发震时刻，而 o 值为 0。

上面的做法需要执行 5 个命令才能实现，而且需要人工查看 o 的值，因而无法用于处理大量数据。下面就对这一例子做进一步简化，这其中需要使用 OBSPY 提供的 “在命令中引用头段变量的值” 的功能。具体的语法以及用法在 “OBSPY 编程” 一章中会介绍。

```
OBSPY> r cdv.z
OBSPY> ch o gmt 1987 173 11 10 10 363
OBSPY> ch allt (0 - &1,o&) iztype
IO OBSPY> wh
```

在这个例子中，(0 - &1,o&) 代替了上个例子中的 -14。简单介绍一下 (0 - &1,o&) 的含义，&1,o& 表示引用内存中第一个 OBSPY 文件的头段变量 o 的值（即 14），然后 (0 - 14) 得到的结果即为 -14。此处简化的优点在于，不需要使用 lh o 查看头段变量的值，完全可以实现自动化。

Note:　在 *OBSPY v101.6* 及之后的版本中，上例中的 (0 - &1,o&) 还可以写成 (0-&1,o&)、

(-&1,o&) 或(-&1,o)。而在 *OBSPY v101.5c* 及之前的版本中，只能使用(0 - &1,o&)，注意减号两边的空格。考虑到命令的通用性，建议使用上面示例中的写法。

上面的示例只适用于为一个 OBSPY 数据添加发震时刻的情况。如果要一次性为多个 OBSPY 数据 添加同样的发震时刻，最直观的想法是：

```
OBSPY> r *.OBSPY
OBSPY> ch o gmt 1987 173 11 10 10 363
OBSPY> ch allt (0 - &1,o&) iztype
IO OBSPY> wh
```

这样的做法是有很大风险的。因为内存中一次性读入了很多 OBSPY 数据，而在使用 ch allt 命令时，&1,o& 引用的是第一个 OBSPY 数据的 o 头段。第二个命令已经保证了内存中所有的数据的 o 都有相 同的绝对时刻（即发震时刻），只要所有数据的参考时刻是一致的，那么所有数据的头段变量 o 的值也必然是一样的。所以当且仅当内存中的所有数据的参考时刻完全一致时，上面的例子才是安全的。 实际处理数据时会遇到很复杂的情况，“所有数据的参考时刻完全一致” 这一假设不一定成立。

在上面的例子的基础上再加一个命令：

```
OBSPY> r
*.OBSPY                    // 同步所有数据的参考时间
OBSPY> ch o gmt 1987 173 11 10 10
363
OBSPY> ch allt (0 - &1,o&) iztype
```

synchronize 的作用是使内存中所有的数据拥有相同的参考时刻，在此命令的基础上，所有数据的头段变量 o 将拥有相同的值，所以直接引用第一个头段变量的 o 值就不再是一件危险的事情了。

4.7　台站和分量信息

相关头段：stla、stlo、cmpaz、cmpinc

一般来说，从 SEED 数据中解压的 OBSPY 数据中都包含了准确的台站信息和分量信息。若数据 中没有包含台站和分量信息，则需要从其它途径获取这些信息，并使用 chnhdr 将这些信息写入到 相应的数据头段中。

4.8　去毛刺

相关命令：rglitches 地震仪器偶尔会出现问题，导致连续地震数据流中出现尖锋或者数据丢失。

这些所谓的毛刺，

肉眼很容易识别，但是在使用程序自动处理数据时却很容易被误认为是地震信号，因而需要在数据分析之前将毛刺去除。rglitches 命令可以在某种程度上检测并去除地震信号中的毛刺。毛刺在模拟地震记录中很常见，现在的数字地震记录中则很少见到，因而实际上很少需要执行这一步操作。

rglitches 的效果可以从图 4.1 中直观地看到。

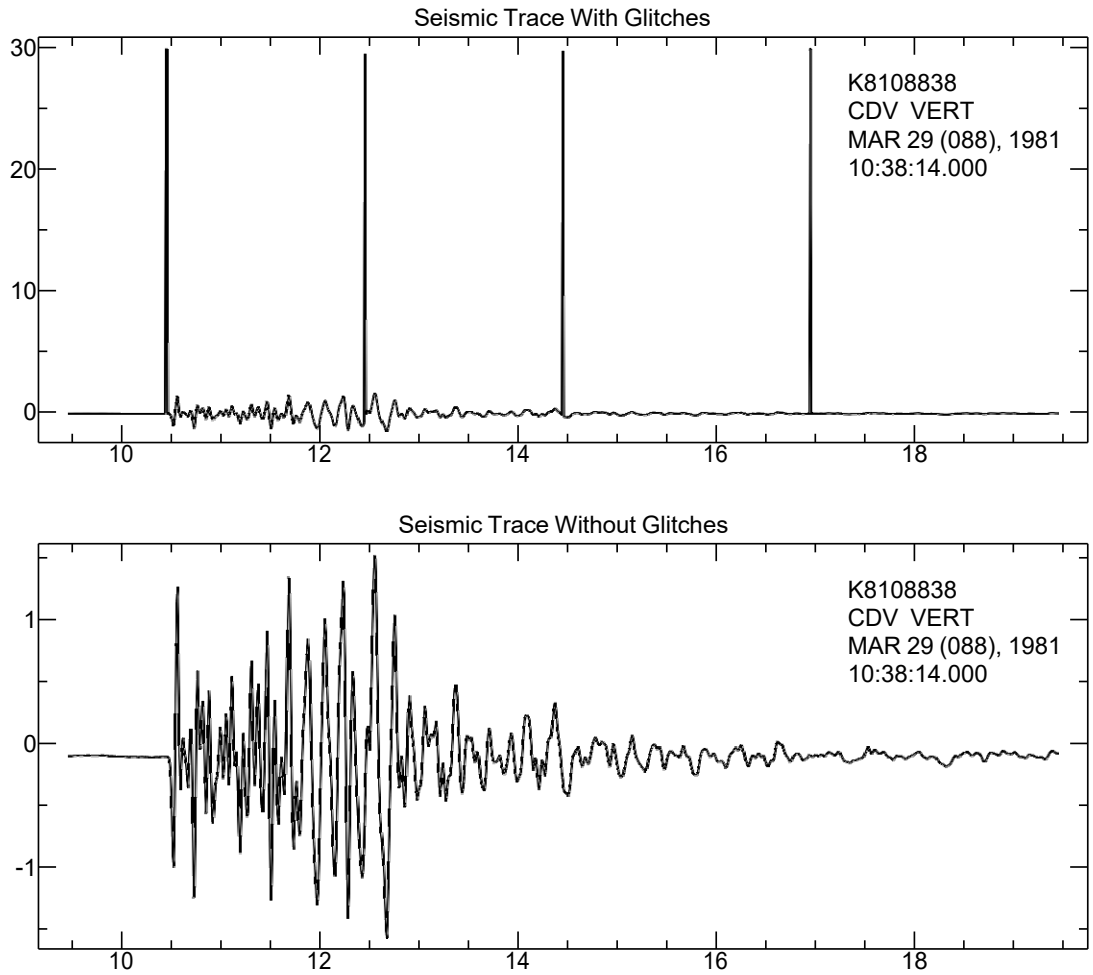

图 4.1: 地震波形去毛刺。上图为包含 glitches 的地震信号，下图为去除 glitches 后的地震信号。

4.9 去均值、去线性趋势和波形尖灭

相关命令：rmean、rtrend、taper 通常，波形数据总会存在一个非零的均值或者存在一个长

周期的线性趋势，这会影响到数据的
分析，必须在数据分析前去除。另一方面，在对数据进行谱域操作（如 FFT、滤波等）时，若数据的两端不为零，则会出现谱域假象，因而实际数据经常需要做尖灭处理，使得数据两端在短时间窗内 逐渐变成零值。

```
OBSPY> fg seis
OBSPY> rmean; rtr; taper
```

图 4.2 中，波形从上到下依次为原始波形、去均值、去线性趋势、和尖灭之后的波形。

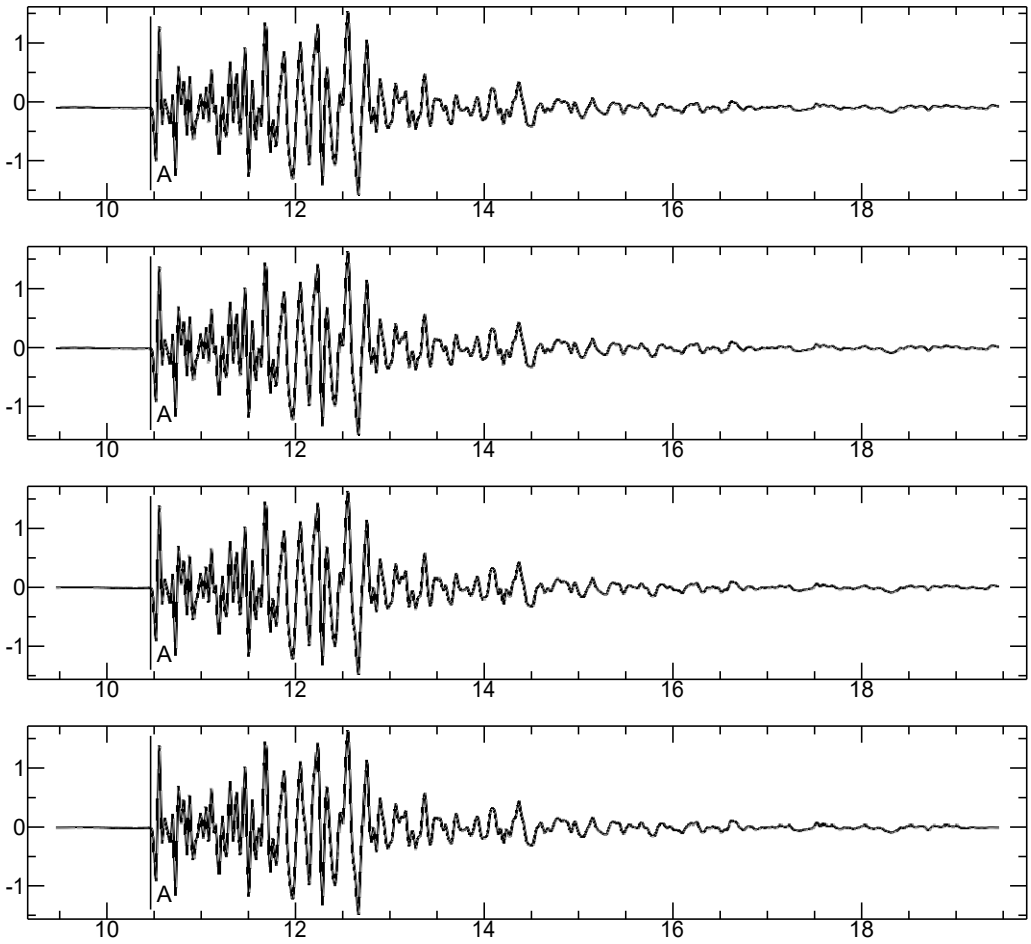

图 4.2: 去均值、去线性趋势和波形尖灭

4.10　去仪器响应

相关命令：`transfer` 相关脚本：

Perl 脚本、Python 脚本

OBSPY 中可以使用命令 `transfer` 实现去仪器响应，本节只列出日常数据处理过程中最常用的命 令。关于仪器响应的详细介绍，请参考附录“仪器响应”以及教科书中的相关内容。关于 `transfer` 命令的具体用法，参考该命令的语法说明及示例。

OBSPY 中常用的仪器响应文件有两种格式，即 RESP 文件和 OBSPY PZ 文件。本节会介绍 RESP
文件和 PZ 文件的几种用法，并对每种方法的优缺点进行比较。

4.10.1　RESP 文件

RESP 方法 1

使用 `evalresp` 选项但不指定 RESP 文件时，`transfer` 会对内存中的所有 OBSPY 数据进行循 环。对于内存中的每个 OBSPY 数据，从头段中提取台站分量信息，然后在当前目录下寻找并使用对应 的仪器响应文件 `RESP.NET.STA.LOC.CHN`。

```
OBSPY> r *.OBSPY
OBSPY> trans from evalresp to none freq 0.004 0.007 0.2 0.4
```

该方法的好处是，可以一次处理多个 OBSPY 数据，且无需指定仪器响应文件的文件名。

RESP 方法 2

可以使用 `evalresp fname` 选项为每个波形分别指定 RESP 文件：

```
OBSPY> r 2006.253.14.30.24.0000.TA.N11A..LHZ.Q.OBSPY
OBSPY> rmean; rtr; taper
OBSPY> trans from evalresp fname RESP.TA.N11A..LHZ to none \
                                    freq 0.004 0.007 0.2 0.4
```

该方法的优点在于，RESP 文件的文件名可以任意，使用起来更灵活。缺点在于，一次只能处理一个 OBSPY 数据，数据的批量处理需要写脚本实现。

RESP 方法 3

可以将所有台站的 RESP 文件都合并到同一个文件中（`cat RESP.*.*.*.* >> RESP.ALL`），并指定该总 RESP 文件为仪器响应文件，此时命令会从总 RESP 文件中自动寻找匹配的仪器响应。

```
OBSPY> r *.OBSPY
OBSPY> trans from evalresp fname RESP.ALL to none \
                                  freq 0.004 0.007 0.2 0.4
```

4.10.2　PZ 文件

PZ 方法 1

手动为每个波形指定 PZ 响应文件：

```
OBSPY> r OR075_LHZ.OBSPY
OBSPY> rmea; rtr; taper
OBSPY> trans from polezero subtype OBSPY_PZs_XC_OR075_LHZ to none \
                            freq 0.008 0.016 0.2 0.4
OBSPY> mul 1.0e9  // 用 PZ 文件 transfer to none 得到的位移数据的单位为 m
                  // 而 OBSPY 默认的单位为 nm，因而必须乘以
1.0e9  OBSPY> w OR075.z //  此时位移数据的单位为 nm
```

该方法的缺点是，一次只能处理一个波形数据，且需要用户编程指定 PZ 文件名。

PZ 方法 2

可以将所有台站的 PZ 文件合并到同一个文件中（`cat OBSPY_PZs_* >> OBSPY.PZs`），并指定该总
PZ 文件为仪器响应文件，此时命令会从总 PZ 文件中自动寻找匹配的仪器响应。

```
OBSPY> r *.OBSPY
OBSPY> trans from pol s OBSPY.PZs to none freq 0.008 0.016
0.2 0.4 OBSPY> mul 1.0e9
OBSPY> w over
```

该方法的优点是一次可以处理多个波形数据。

4.10.3　对比

从易用性来看，RESP 方法 1、RESP 方法 3 和 PZ 方法 2 都是比较易于使用的，只需要一个简单的命令，即可同时对所有波形数据进行处理。而 RESP 方法 2 和 PZ 方法 1，需要用户自己从数据文件的文件名或头段中提取信息，并指定对应的响应文件，这需要通过写少量的脚本来实现。

从执行效率来看，做了一个简单的测试，共 670 个波形数据，用不同的方法去仪器响应，所需时间如下：

RESP 方法 1 58 秒；
RESP 方法 2 43 秒；
RESP 方法 3 227 秒；
PZ 方法 1 8 秒；
PZ 方法 2 90 秒；

从中可以总结出执行效率的如下规律：

1. RESP2 和 PZ1 相比，RESP3 与 PZ2 相比，可知，PZ 文件的效率要高于 RESP 文件。这很容易理解，毕竟 RESP 文件中包含了更为完整的信息，计算量要更大一些；PZ 文件中仅包含了零极点信息和总增益信息，对于日常的使用来说，已经足够；
2. RESP1 和 RESP2 相比，区别在于：后者使用指定的文件，前者则需要从数据中提取信息、构建文件名并在当前目录下搜索，因而 RESP1 要比 RESP2 慢一些。
3. RESP3 和 PZ2 方法，都是把多个响应函数放在同一个响应文件中，对于每个波形都需要对响应文件做遍历以找到匹配的响应函数，因而是所有方法中速度最慢的。

总结下来：

- 想要写起来简单，用 RESP 方法 1；
- 想要执行快，可以用 PZ 方法 1；

4.11　数据截窗

相关命令：`cut` 数据申请时一般会选择尽可能长的时间窗，而实际进行数据处理和分析时可能只需要其中的一

小段数据，这就需要对数据进行时间窗截取。

OBSPY 中的 `cut` 命令可以实现数据截取，需要注意的是，该命令是 “参数设定类” 命令，即需要 先执行 `cut` 命令再执行 `read` 命令。

4.11.1　pdw

使用 `cut` 命令对数据进行截取时需要定义数据时间窗。除了截取数据之外，其他一些命令也会需要定义时间窗，比如 `rms`、`mtw`、`xlim` 等，这些命令都使用同样的方式定义时间窗，在 OBSPY 中称 为 pdw，即 partial data window。

pdw 定义了一个开始时间和一个结束时间，其格式为 `ref offset ref offset`。其中 `ref` 为参考时刻，可以取 `Z`、`B`、`E`、`O`、`A`、`F`、`N` 和 `Tn`(n=0–9)，而 `offset` 为相对于参考时刻的时间偏移量。

参考时刻 `ref` 可以取如下值：

- `B`：磁盘文件起始值
- `E`：磁盘文件结束值
- `O`：事件开始时间
- `A`：初动到时
- `F`：信号结束时间
- `Tn`：用户自定义时间标记，n=0–9
- `Z`：参考时刻
- `N`：将 `offset` 解释为数据点数而非时间偏移量，其仅可以用于结束值

若开始或结束的 `offset` 省略则认为其值为 0。若开始 `ref` 省略则认为其为 Z；若结束 `ref` 省略则认为其值与开始 `ref` 相同。

下面的例子中展示了一些常见的 pdw 及其含义：

```
B E              // 文件开始到文件结束，即与 cut off 相同
B 0 30           // 文件开始的 30 秒
A -10 30         // 初动前 10 秒到初动后 30 秒
B N 2048         // 从 T0 前 10 秒起的 1000 个点
T0 -10 N 1000    // 相对磁盘文件 0 点的 30.2 到 48
30.2 48
```

4.11.2　cut

cut 命令是“参数设定类”命令，因而需要先 cut 再 read：

```
OBSPY> cut t0 -5 5    // 截取 t0 前后各 5 秒，共计 10 秒的数据
```

4.12　分量旋转

相关命令：rotate 相关头段：

cmpinc、cmpaz 相关脚本：Perl 脚本、Python 脚本

三个正交的地震传感器即可完全记录地面运动矢量。因此可以将三个正交的分量任意旋转到其 它三个正交的方向上。

出于仪器安装的考虑，地震仪三分量一般都是 N、E、U 向的。而地震学里，由于 SH 波与 P-SV 波的解偶，更常处理的是 R、T、Z 向的三分量数据。因而地震信号的旋转是有必要的。

OBSPY 提供了 rotate 命令，用于旋转任意两个相互正交的分量。旋转前，OBSPY 会检查两个分量 的 cmpinc 和 cmpaz，以确定两个分量是正交的。

```
OBSPY> dg sub teleseis ntkl.[enz]
/opt/obspy/aux/datagen/teleseis/ntkl.e  ...ntkl.n  ...nt
kl.z OBSPY> w ntkl.e ntkl.n ntkl.z
OBSPY>
r ./ntkl.n ./ntkl.e
OBSPY> lh cmpinc cmpaz
  FILE: ./ntkl.n - 1
 --------------
     cmpinc = 9.000000e+01
      cmpaz = 0.000000e+00
  FILE: ./ntkl.e - 2
 --------------
     cmpinc = 9.000000e+01
      cmpaz = 9.000000e+01
OBSPY> rotate to gcp          // 旋转到大圆路径
OBSPY> lh cmpinc
  cmpaz
  FILE: ./ntkl.n - 1
 --------------
     cmpinc = 9.000000e+01
      cmpaz = 2.440466e+01
```

```
     cmpinc = 9.000000e+01
      cmpaz = 1.144047e+02
OBSPY>  w  ntkl.r  ntkl.t        // 保存为 R 分量和 T 分量
```

图 4.3 中，左图中从上至下为 N、E、Z 分量，右图中从上至下为 R、T、Z 分量。旋转到 R、T 分量后，可以很容易地识别出 Rayleigh 和 Love 波。

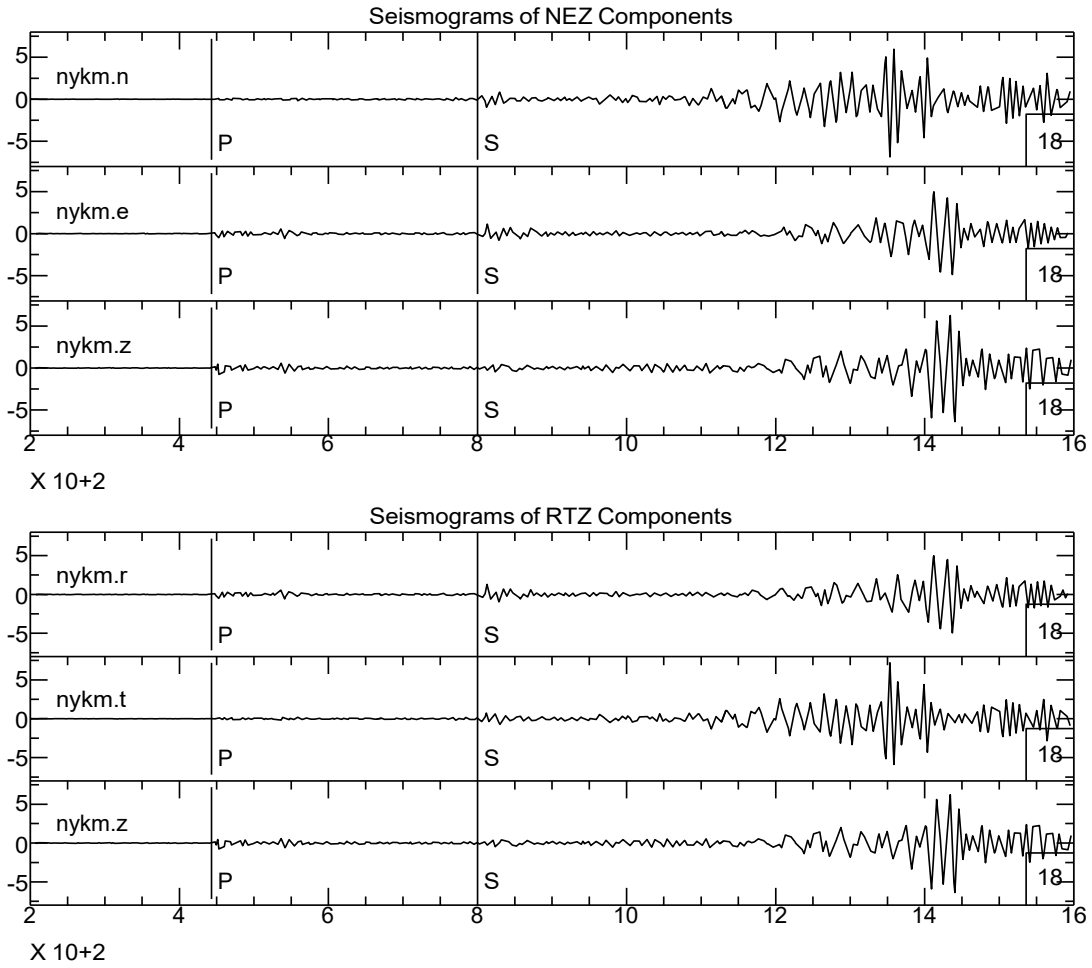

图 4.3: 将 N、E 分量旋转到 R、T 分量。

4.13　数据重采样

相关命令：decimate、interpolate

如下情形，需要对数据进行重采样：

- 不同仪器的采样周期可能不同，需要将所有的数据重采样到相同的采样周期；
- 数据的采样周期很小，导致数据量很大，而实际研究中不需要如此小的采样周期，因而可以对 数据做减采样以减小数据量；
- 数据的采样周期过大，实际研究中需要更小的采样周期，此时需对数据做插值重采样；
- 数据为不等间隔数据，需要插值成为等间隔数据；

4.13.1　decimate

decimate　专门用于解决上面所说的第二种情形，即等间隔数据的减采样。在减采样过程中，根

据 Nyquist 采样定理，可能会出现混叠现象，而 decimate 对数据自动做了低通滤波，以避免混叠现象的产生。

下面的示例中，将一个等间隔数据减采样 10 倍：

```
OBSPY> fg seis
OBSPY> lh delta npts

    delta = 1.000000e-02     //采样间隔 delta=0.01
     npts = 1000
OBSPY> decimate 5; decimate 2 // 减采样 10 倍
OBSPY> lh delta npts

    delta = 9.999999e-02     //采样间隔 delta=0.1，忽略浮点数误差
     npts = 100
```

4.13.2 interpolate

与 decimate 相比，interpolate 命令功能更加强大，其可以对等间隔或不等间隔数据进行增采样或减采样。

比如增采样，即插值：

```
OBSPY> fg seis
OBSPY> lh delta npts

    delta = 1.000000e-02
    npts = 1000
OBSPY> interp delta 0.005    // 增采样 2 倍
OBSPY> lh

    delta = 5.000000e-03
    npts = 1999
```

对于减采样，interpolate 与 decimate 的功能略有重复，但 interpolate 在减采样时不会对数据进行低通滤波，因而使用 interpolate 进行减采样时可能会出现混叠现象，故而需要手动进行低通滤波。

下面的示例将数据减采样到 20 Hz。根据 Nyquist 采样定理，为了保证不产生混叠现象，应首先对数据做 10 Hz 的低通滤波。

```
OBSPY> fg seis
OBSPY> lh npts delta

    npts = 1000
    delta = 1.000000e-02
OBSPY> lp c 10
OBSPY> interpolate delta 0.05
WARNING potential for aliasing. new delta: 0.0500 data delta: 0.0100
   // 在做了 lowpass 之后，此处的警告可忽略
```

4.14　滤波

相关命令：bandpass、lowpass、highpass、bandrej　几乎所有的数据分析都需要将数据限制在一定的频率范围内，这就需要对数据做各种不同方式的滤波。关于滤波的细节，可以参考滤波命令的相关说明，以及信号处理相关书籍。至于滤波的范围，则依赖于具体的研究对象。

滤波过程中，有几个可以调节的参数：截止频率、阶数、通道数等等。不同的参数对波形造成的变形有多大区别？不同的参数得到的滤波器的频率响应又是怎样的？其实很容易就可以知道。

下面的代码生成了一个脉冲函数，然后对脉冲函数用某个参数组合做滤波，得到滤波后的波形，然后对波形做 FFT，得到波形的振幅谱和相位谱。根据定义可知，得到的波形为滤波器的时间响应，得到的振幅谱和相位谱为滤波的频率响应：

```
OBSPY> fg impulse delta 0.01 npts 1000 // 生成脉冲函数, delta 和 npts
可调
OBSPY> bp c 0.2 2 n 2 p 1          // 要查看的滤波参数
OBSPY> beginframe
OBSPY> xvport 0.1 0.9; yvport 0.7
0.9 OBSPY> fileid off
OBSPY> qdp off
OBSPY> title 'Time Domain Response'
OBSPY> p                           // 绘制时间域响应
OBSPY> fft
OBSPY> xvport .1 .45;
OBSPY> psp am loglog               // 振幅响应
OBSPY> xvport .55 .9
OBSPY> title 'Phase
Response' OBSPY> psp ph            // 相位响应
linlin
```

上面的代码涉及到了一些尚未介绍的命令，在后面会具体介绍。生成的效果图如下：

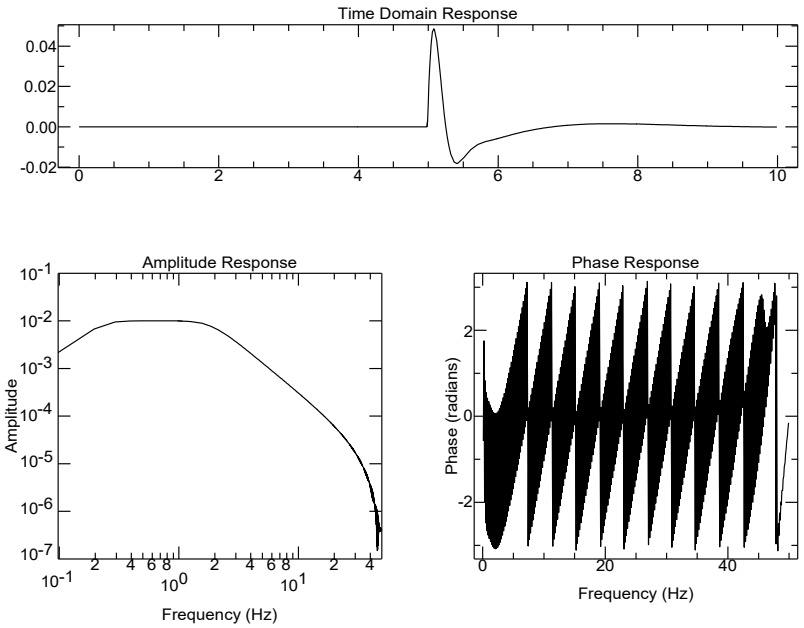

图 4.4: 滤波器的时间响应和频率响应

对脉冲波形做 0.5Hz 到 5Hz 的带通滤波，下图中给出了不同的阶数和通道数对波形的影响：

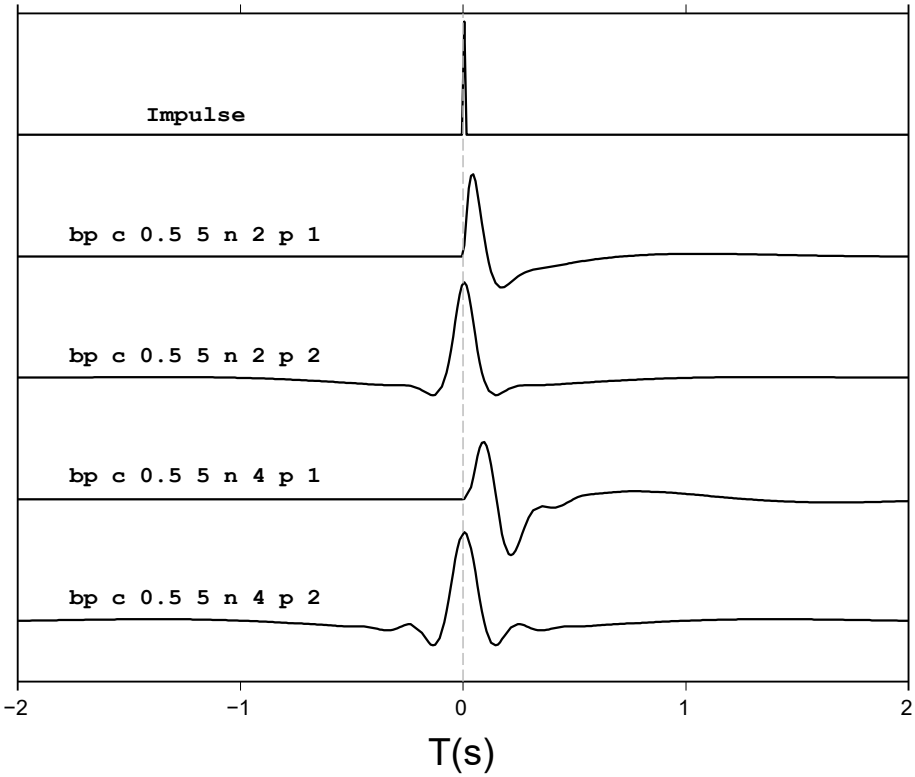

图 4.5: 不同参数的带通滤波效果

图 4.5 中 Impulse 为原始脉冲波形，下面四条波形是分别取不同的 n 值和 p 值的结果。

p 取 1 时，对波形做一次带通滤波，由于滤波器存在相位延迟，因而导致波形的峰值出现了时间延迟，因而会影响到震相的最大峰值的拾取，但对震相的初至却没有影响。

p 取 2 时，对波形做正反两次带通滤波，此时不存在相位延迟，因而不会影响到最大峰值的拾取，但震相的初至则存在时间上的提前。

4.15　震相理论到时

震相理论到时的计算是地震学的基础，具体的原理涉及到一大堆公式，感兴趣的读者可以阅读 相关文献。

日常工作中经常需要识别震相、将波形按理论到时排序或迭加、计算震相到时残差等等，这些 都需要用到震相理论到时。将震相理论到时写入到 OBSPY 文件的头段中，可以使得后期处理更加简 单。OBSPY 头段中可以用于保存震相到时信息的头段包括：a、f 以及 tn（n=0−9），ka、kf 和 ktn 则 用于保存相应的震相名信息。

给 OBSPY 文件标记理论到时有三种方法，下面一一介绍。

4.15.1　手动标记

最直观的办法是根据 OBSPY 文件中的震源深度和震中距信息用某些程序计算出理论到时，然后 用 chnhdr 命令手动将到时信息写入到 OBSPY 头段中。

```
OBSPY> dg sub teleseis nykl.z // 以 nykl.z 为例
OBSPY> lh evdp gcarc          // 查看震源深度和震中距
    evdp = 0.000000e+00
```

```
    gcarc = 3.841450e+01
//  利用某程序计算得到 ak135 模型下，P 波走时为 443.14 秒，S 波走时为 799.05 秒
//  若 OBSPY 文件的参考时间为发震时刻，则
OBSPY> ch t0 443.14 t1 799.05 kt0 P kt1 S
OBSPY> lh t0 kt0 t1
     kt1 t0 =
     4.431400e+02
    kt0 = P
     t1 = 7.990500e+02
```

该方法很直接也很基础，其缺点也很明显，那就是麻烦。如果参考时刻不是发震时刻的话，则更复杂 一些。

4.15.2　traveltime 命令

`traveltime` 是 OBSPY 提供的一个命令，用于计算 iasp91 或者 ak135 地球模型下的震相理论走 时，并自动将震相到时信息保存到 OBSPY 头段变量中。

```
OBSPY> dg sub teleseis nykl.z
OBSPY> traveltime model iasp91 picks 3 phase
P S traveltime: depth: 0.000000 km
OBSPY> lh t3 kt3 t4 kt4
        t3 = 4.430530e+02
       kt3 = P
        t4 = 7.999642e+02
       kt4 = S
```

该命令会将震相 P、S 的理论到时依次写入头段变量 `t3`、`t4` 中，并在相应的 K 头段中写入震相名信息。该方法的优点在于是 OBSPY 内置命令，使用起来也很简单，缺点在于只支持两种地球速度模型。

4.15.3　taup_setObspy 命令

`taup_setObspy` 是 taup 软件提供的用于计算震相理论到时的独立于 OBSPY 的外部命令。

```
$ taup_setObspy -mod prem -ph P-6,S-7 -evdpkm *.OBSPY
```

该命令会根据 OBSPY 文件中的震中距和震源深度信息，计算震相在一维地球模型下的理论走时，将 震相走时写入到头段 Tn 中，并向 `KTn` 中写入震相名称，向 `USERn` 中写入震相的射线参数（单位为 s/radian）。上面的示例中，使用了 PREM 地球模型，将 P 和 S 的走时信息写入到头段 `T6` 和 `T7` 中。

该方法的优点在于支持多种地球标准模型、自定义模型以及自定义震相。

4.16　波形排序

相关命令：`sort`

在画图时，波形数据在图像上的顺序与 OBSPY 数据读入内存时的顺序是一样的，有时候会需要 对波形更加灵活的排序，比如在手动拾取震相的时候，会希望将所有波形数据按照震中距从小到大 的顺序进行排列。OBSPY 提供了 `sort` 命令使得波形可以按照某个头段变量的值进行排序。

将波形数据按照方位角升序排列：

```
OBSPY> r *.OBSPY
OBSPY> sort az
```

```
OBSPY> ppk
```

将波形数据按照震中距降序排列：

```
OBSPY> r *.OBSPY
OBSPY> sort gcarc
descend OBSPY> ppk p
```

4.17　质量控制

质量控制就是标记/删除信噪比低或不合适的波形。 用户可以自己在程序中判断数据的好坏以进行质量控制，但这样做很困难，因为实际情况中，会遇到各种奇形怪状的“坏”波形，很难用统一的程序将这些“坏”波形挑选出来，所以更多时候需要人工的参与。 一种常规的做法如下：

```
OBSPY> r *.OBSPY          // 读入全部的 OBSPY 数据
                          // plotpk，每次绘制 5 个波
// 若波形质量很差，则用 t9 标记
OBSPY> wh
```

解释一下以上做法，首先读入所有的 OBSPY 数据，然后利用 `plotpk`，每次绘制 n 个波形，如果 是 3 分量数据，n 一般取 3。若波形质量很好，则不理会；若波形质量很差，则在波形的任意时刻标 记 `t9` 的值（具体如何标记可以参考下一节的内容），然后使用 `wh` 将标记的 `t9` 保存到头段中，再退 出 OBSPY。

完成上面的步骤之后，所有“坏”波形的 `t9` 都被标记，一般来说都是一个正值，而所有“好”波形的 `t9` 则都处于未定义状态，其值为 -12345.0。

鉴于此，可以通过如下命令删除“坏”波形：

```
$ Obspylst t9 f *.OBSPY | awk '$2>0 {print "rm",$1}' | sh
```

令

Note: 一定要在理解该命令的含义的前提下才可使用，否则可能会造成数据的丢失！

当然，也可以用如下命令将“坏”数据移动到专门的目录中：

```
$ mkdir BAD
$ Obspylst t9 f *.OBSPY | awk '$2>0 {print "mv", $1, "BAD/"}' | sh
```

令

Note: *awk* 命令中，目录名 *BAD* 后最好加上斜杠。若不加斜杠，且忘记新建目录 *BAD*，则所有应该放在目录 *BAD* 中的文件都会被重命名为 *BAD*，进而导致文件丢失。

4.18　震相拾取

相关命令：`plotpk`

震相拾取，或者说标定到时，是 OBSPY 的一种常用功能。

4.18.1　ppk 模式的进入与退出

要进行震相拾取，首先要进入“ppk 模式”。

读取波形数据后，在终端中键入 `plotpk`（简写为 `ppk`），就会出现一个绘图窗口。若之前未曾打开过绘图窗口，则此时焦点位于 ppk 打开的绘图窗口中；若之前曾经打开过绘图窗口，则需要鼠标点击一下绘图窗口以使得焦点位于绘图窗口而不是终端中。此时，OBSPY 就进入了“ppk 模式”，终端 中光标所在行没有 OBSPY 提示符“OBSPY>”。

```
OBSPY> fg seis
OBSPY> ppk //   焦点位于绘图窗口中，进入 ppk 模式
          // 光标所在行没有提示符"OBSPY> "
```

学会如何进入 ppk 模式后，还要学会退出 ppk 模式。首先，确保焦点位于绘图窗口而不是终端，然后将光标移动到绘图窗口中，按下“q”键即可退出 ppk 模式。此时，终端中光标所在行会重新出现 OBSPY 提示符“OBSPY>”。

今

Note: 只有当使用了 *ppk* 命令，焦点位于当前绘图窗口，且鼠标位于当前绘图窗口内才称为 *ppk* 模式。在 *ppk* 模式下，所有的键盘输入都会被解释为"*ppk* 命令"，但不会在终端中显示出来。若使用 *ppk* 命令后，不慎使焦点位于终端内，即脱离了 *ppk* 模式，此时所有的键盘输入都会出现在终端中，但不会被 *OBSPY* 解释，当退出 *ppk* 模式时，*OBSPY* 才会依次解释终端中的命令。

4.18.2　ppk 模式下拾取震相

下面介绍如何在 ppk 模式中拾取震相。先进入 ppk 模式，此时焦点位于绘图窗口，并保证鼠标位于绘图区（即四个边框）的内部，移动鼠标到要标记到时的地方，依次按下 `t`、`0`，在要标记的到时 处会出现一条竖线，旁边有标识 `T0` ，此时已经将要标记的到时（即竖线对应的 X 轴位置）保存到头段变量 `T0` 中。再按下 `q` 以退出 ppk 模式，最后在终端键入 `wh` 将内存中的头段变量写回到磁盘文件中。

除了可以键入 `t` 和 `0` 之外，0 还可以用 1 到 9 的任意数字替换，分别表示将要标记的到时保存到 `T0` 到 `T9` 中。

```
OBSPY> fg
seis OBSPY>
ppk
//   键入"t"和"0"标记到时，然后按"q"退出 ppk 模式
OBSPY> lh t0
     t0 = 1.255385e+01
```

今

Note: 在键入"*t*" 时，鼠标不仅要在绘图窗口内，还要在绘图区（即四个边框）的内部，否则会得到"*Bad cursor position. Please retry.*" 的错误提示。

今

Note: *OBSPY* 全局变量 `OBSPY_PPK_USE_CROSSHAIRS` 可以控制 *ppk* 模式下鼠标在绘图窗口内的
形态。若其值为 *0*，则鼠标会以十字线的形式出现，即"＋"；当其值为 *1* 时，会在十字线的基础上加上水平线和垂直线。通常建议设置其值为 *1*，使得拾取到时时更精确。该全局变量的设置方式参考 *在 Linux 下安装 OBSPY* 一节。

4.18.3　qdp off

OBSPY 在默认情况下会打开快速绘图选项，即 `qdp on`。关于 `qdp`，可以参考“图像外观”一节以 及命令 `qdp` 的说明。

在拾取震相时，若打开了快速绘图选项，则由于数据没有完全绘制而导致震相的可识别度降低，也导致波形拾取精度降低。为了提高拾取精度，通常会在进入ppk模式前关闭快速绘图选项，即使用 `qdp off` 命令：

```
OBSPY> dg sub reg
OBSPY> qdp     // 打开快速绘图选项（默认值）
on
OBSPY> ppk     // 关闭快速绘图选项
OBSPY> qdp     // 注意观察与之前的区别
```

每次启动 OBSPY 后进入 ppk 模式前，都要手动执行 `qdp off` 以关闭快速绘图选项，这样相对比较麻 烦，可以使用 “OBSPY 初始化” 一节中介绍的方法使得每次 OBSPY 启动时自动关闭快速绘图选项。

4.18.4　放大与缩小

有时数据时间较长，难以精确标定到时，此时需要将图幅放大，以显示整个波形的一小部分。

首先需要将光标移动到绘图区域中的某位置，键入 “x”，再移动至另一位置，再次键入 “x”。这样，两次键入确定了一个时间窗。这时，绘图窗口中将只显示该时间窗内的波形，也就实现了图幅的放大。可不断重复此步骤，进行多次放大。

OBSPY 101.5 之后的版本有更方便的方式：在绘图窗口中某位置按下鼠标左键，并拖动至另一位 置再松开鼠标左键，则两个位置之间的时间窗内的波形会被放大。

图幅的缩小通过键入 “o” 来实现，“o” 最多可以回退 5 次绘图历史。

4.18.5　同时标记三分量

通常，震相在同一台站的三分量数据上具有相同的到时，因而将同一台站的三分量数据画在一 张图上，一方面可以综合三分量的波形信息以更准确地识别震相，另一方面，一次标定三分量的震 相到时可以减少工作量并保证震相在三分量上的到时相同。使用命令 “`ppk p 3 a m`” 进入 ppk 模 式即可每次只显示并同时标记三个波形数据。

通常在拾取震相时会一次性读入多个台站的波形数据，而 “`ppk p 3 a m`” 一次只能显示三个波形数据，可以在 ppk 模式下不断键入 “n” 以依次显示接下来的三个波形，也可以键入 “b” 以显示前三个波形。当不断键入 “n” 直到所有波形数据都显示完毕的时候，会自动退出 ppk 模式。

```
<<<<<<< HEAD
OBSPY> dg sub tele *  // 生成多个台站的三分量数据
OBSPY> ppk p 3 a m
//  键入"t0"标记 ntkl 台站的三分量到时
// 键入"n"以绘制接下来的三个数据
//  键入"t0"标记 nykl 台站的三分量到时
// 键入"n"以绘制接下来的三个数据
// 键入"b"以绘制之前的三个数据
//  键入"t0"重新标记 nykl 台站的三分量到时
// 键入"n"以绘制接下来的三个数据
// 键入"t0"标记 onkl 台站的三分量到时
// 键入"n"以绘制接下来的三个数据
//  键入"t0"标记 sdkl 台站的三分量到时
// 键入"n"自动退出 ppk 模式
OBSPY>
wh
```

在使用“ppk p 3 a m”选项同时标记三分量时需要注意：

- 三分量数据的参考时刻必须相同；若参考时刻不相同，则标记的结果是错误的
- 该命令每次会按照顺序显示内存中的三个波形数据，当且仅当每次显示的三个波形数据恰好是同一台站的三分量数据时，该命令才能用作同时标记同一台站的三个分量

要使得每次显示的恰好是同一台站的三分量波形数据，则要求同一台站的三个分量在内存中分别位于第 n、$n+1$ 和 $n+2$ 位，其中 n 为正整数。通常情况下，一次性读入全部数据的时候，都可以满足这一要求。但也有一些例外：

- 数据文件名比较奇葩，导致读入时同一台站的三分量数据不是紧挨着读入的，可以使用“ls *.OBSPY”命令检查文件的读入顺序；
- 某个台站丢失了一个分量的数据，导致后面的所有台站都出现问题；

4.18.6 ppk 命令

除了上面介绍的若干 ppk 命令之外，还有很多其他 ppk 命令。表 4.1 列出了 ppk 模式下的所有命令，其中常用的命令包括“b”、“l”、“n”、“o”、“q”、“t”和“x”。

表 4.1: ppk 模式命令一览表

命令	含义	说明
a	定义事件初至 a	1,7
b	如果有，则显示上一张绘图	
c	计算事件的初至和结束	1,4,7
d	设置震相方向为 DOWN	
e	设置震相起始为 EMERGENT（急始）	
f	定义事件结束 f	1,2,3,7
g	以 HYPO 格式将拾取显示到终端	4
h	将拾取写成 HYPO 格式	3,4
i	设置震相起始为 IMPULSIVE	
j	设置噪声水平	2,6,8
k	即 kill，退出 ppk 模式	
l	显示光标当前位置	2,4
m	计算最大振幅波形	2,3,5
n	显示下一绘图	
o	显示前一个绘图窗，最多可以保存 5 个绘图窗	
p	定义 P 波到时	1,2,3,7
q	即 quit，退出 ppk 模式	
s	定义 S 波到时	1,2,3,7
t	用户自定义到时 tn，输入 t 之后需要输入 0 到 9 中的任一数	1,2,7
u	设置震相方向为 UP	
v	定义一个 Wood-Anderson 波形	2,5
w	定义一个通用波形	2,5
x	使用一个新的 x 轴时间窗，简单说就是放大。	
z	设置参考水平	2,6,8
\	删除当前全部拾取的定义。当一个文件中包含多个事件时有用。	
+	设置震相方向为 PLUS	

接下页…

接上页

命令	含义	说明
-	设置震相方向为 MINUS	
␣	设置震相方向为 NEUTRAL	
n	设置震相质量为 n，n 取 0-4	

注意：ppk 模式的命令几乎都是由单个字符组成的，比如退出 “q”，唯一的例外是命令 “t”，由字符 “t” 和 0–9 的整数构成。

不同的命令效果可能不同，有些会在绘图窗口显示信息，有些会将信息写入头段 i 变量，下面对表 4.1 中的说明进行一个说明：

1 会将信息写入头段变量
2 写入字符型震相拾取文件（若已打开）
3 写入 HYPO 格式震相拾取文件（若已打开）
4 在绘图窗口中显示信息
5 窗口显示包含波形的矩形
6 在指定的水平处放置水平光标
7 绘图窗口显示含有到时标识的垂直线
8 绘图窗口显示含有标识的水平线

4.19　数据分析

数据分析是整个科学研究的核心。OBSPY　　软件只是负责完成前期的预处理工作，真正核心的数据　分析工作通常需要用户自己写程序完成。但无论怎样都牵涉到如何在程序中读写 OBSPY 文件的问题，　这个问题放在 “使用 OBSPY 函数库” 和 “OBSPY I/O” 这两章中具体说明。

保护环境，从阅读电子文档开始！

第 5 章　OBSPY 图像

5.1　图形设备

OBSPY 支持两种图形设备，分别是 xwindows 和 sgf，默认的图形设备是 xwindows。可以使用 `begindevices` 和 `enddevices` 命令开启/关闭指定的图形设备；同时也可以使用 `setdevice` 命令 设定默认的图形设备。

5.1.1　xwindows

xwindows 即 X Window System，也称为 X11 或 X，是一种以位图方式显示的软件窗口系统。几乎所有的现代操作系统都能支持与使用 X，Linux 下知名的桌面环境 GNOME 和 KDE 也都是以 X 窗口系统为基础建构成的。

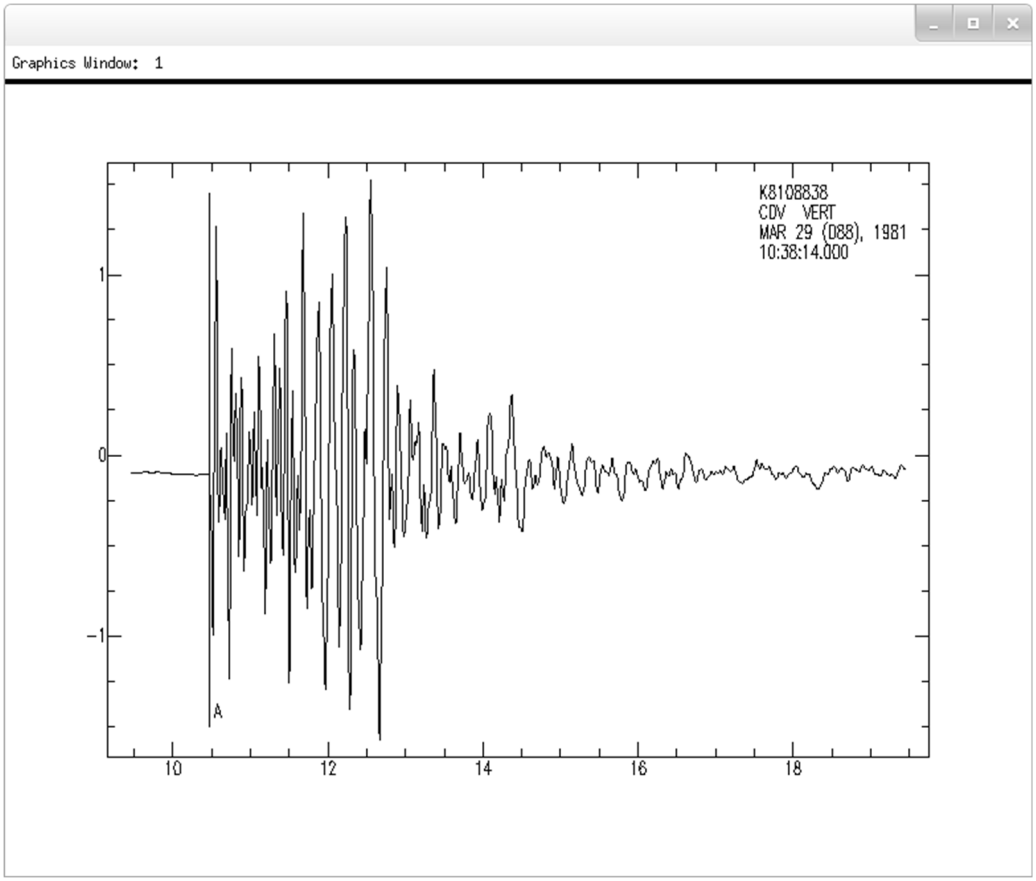

图 5.1: OBSPY 绘图窗口

图 5.1 展示了 OBSPY 中的 xwindows 图形设备的外观，它是 OBSPY 默认的图形设备。同很多其它 软件界面类似，xwindows 窗口在左上角显示图标，右上角显示 “最小化”、“最大化”、“还原” 和 “关

闭”　　按钮。窗口的中间部分为真正的绘图区，本文档的其余插图将只给出绘图区的图像而不再包含窗口部分。

左上角的 “Graphics Window: 1” 指明了当前绘图窗口的编号为 “1”，OBSPY 最多支持同时打开 10 个 X 窗口，编号为 1–10。默认情况下只启动并使用 1 号 X 窗口。`beginwindow` 命令用于启动指 定编号的 X 窗口；`window` 命令还可以设置每个 X 窗口的长宽比以及 X 窗口相对于屏幕的位置。

5.1.2　sgf

SGF，全称 OBSPY Graphic File，即 OBSPY 图形文件，是 OBSPY 自定义的一种文件格式，其包含了 绘制一个图件所需要的全部信息，可以通过　sgftops　等工具转换到其它图形设备或图形文件格式。

若启用了 SGF 图形设备，每次绘制的图件将分别保存到单独的 sgf 文件中。默认情况下，sgf 图形文件的文件名格式为 `fnnn.sgf`，其中 “nnn” 为图件编号，起始编号为 001，每生成一个图件该编号递增。`sgf` 命令可以控制 SGF 图形设备的选项，比如文件名前缀（默认为 `f`）、起始编号（默认从 `001` 开始）、保存目录、文件尺寸等。

5.2　OBSPY 绘图流程

严格地说，OBSPY 绘图的流程应该是：启动图像设备 → 绘图 → 关闭图像设备。如下所示：

```
OBSPY> r cdv.[nez]
OBSPY> begindevices xwindows // 启动图像设备 xwindows，简写为
OBSPY> p                     // 绘图
Waiting
Waiting
OBSPY> enddevices            // 关闭图像设备 xwindows，简写为 ed
xwindows
```

上面的步骤稍显繁琐，OBSPY 将这一流程进行了简化。在第一次执行绘图命令前，OBSPY 偷偷启动 了默认的图像设备 xwindows，接下来的绘图工作都在该窗口中完成。当用户退出 OBSPY 时，OBSPY 会 自动关闭图像设备。

也许你已经发现，即使 plot 结束或者中途退出 plot，绘图窗口依然没有被关闭。即便点击窗口的 “关闭” 按钮，窗口依然无法关闭。若想要关闭绘图窗口，应如上例那样使用 `ed x` 命令。

5.3　绘图命令

OBSPY 提供了许多与绘图有关的命令，包括控制图像外观的参数控制类命令以及执行绘图功能 的操作执行类命令。这一节将介绍常用的几个操作执行类命令。

5.3.1　plot

`plot`　命令会绘制内存块中的所有波形数据，但每次只显示一个波形，然后等待用户输入再决定是否显示下一个波形。该命令的具体用法在第 2.9 节已经详细介绍。

5.3.2　plot1

`plot1`　命令会绘制内存块中的所有波形数据，在一个窗口中一次显示多个波形，这些波形共用 一个 X 轴（时间轴），但拥有单独的 Y 轴。

```
OBSPY> dg sub local
cdv.[enz] cdv.e  cdv.n  cdv.z
```

```
OBSPY> p1
```

执行 plot1 命令后，焦点位于图形窗口，显示如图 5.2。

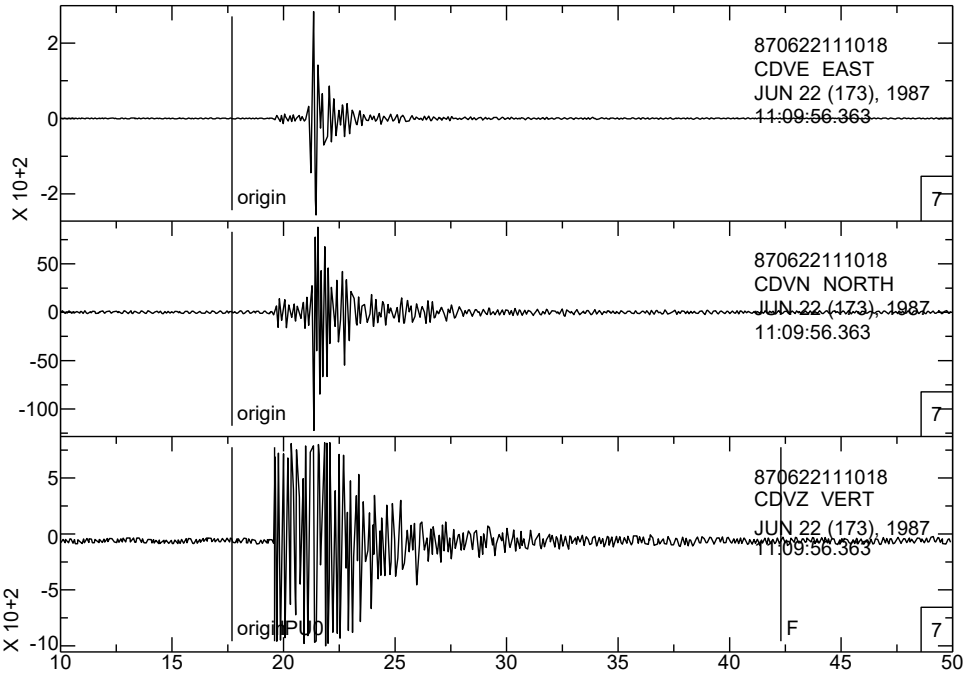

图 5.2: plot1 绘图效果

当一次性读入多个波形数据时，若直接使用 plot1 绘图，会一次性显示全部波形，导致窗口内波形太密，反而什么都看不清。plot1 提供了“perplot n”选项以指定窗口内一次最多显示多少个波形，余下的波形则处于等待状态。在查看波形的时候，经常需要将每个台站的三分量波形记录放 在一起看，此时设置选项perplot的参数值为3即可。

```
OBSPY> dg sub local cdv.[enz] cvl.[enz] cvy.[enz] // 生成 9 个地震波形
cdv.e cdv.n cdv.z cvl.e cvl.n cvl.z cvy.e cvy.n cvy.z
OBSPY> p1 p 3     //   p 是选项 perplot 的简写，3 代表每次显示 3 个
波形  Waiting
Waiting
OBSPY
```

默认情况下，所有的波形数据会按照绝对时间（absolute）对齐，若波形数据具有不同的开始 时间，则波形数据之间会出现相对错动；也可以使所有的波形数据相对于（relative）各自的开始 时间绘图，此时 X 轴的起始坐标为 0。

5.3.3 plot2

plot2 会一次性将内存块中的所有波形绘制在一个窗口内，所有的波形共用 X 轴，因而绘图时也可以使用绝对模式或相对模式。与 plot1 不同的是，所有的波形还同时共用 Y 轴，因而波形会相互覆盖。

plot2 适合绘制多个波形的对比图，常用于数据处理前后波形对比或真实波形与合成波形间的对比。

```
OBSPY> fg seis                  // 生成数据
OBSPY> rmean; rtrend; taper     // 预处理
OBSPY> w seis.0                 // 写入滤波前文件
```

```
OBSPY> bp c 0.05 10 n 4 p 2  // 滤波
OBSPY> w seis.1              // 写入滤波后文件
OBSPY> r ./seis.[01]         // 读入两个文件
./seis.0 ...seis.1
OBSPY> color red inc list red blue // 对两个数据分别设置红色和蓝色
OBSPY> p2                    // 绘图
```

图 5.3 中红线为滤波前波形，蓝线为滤波后波形，二者共用 X 轴和 Y 轴，从这样的波形对比图中，可以很明显得看到滤波对于波形的影响。

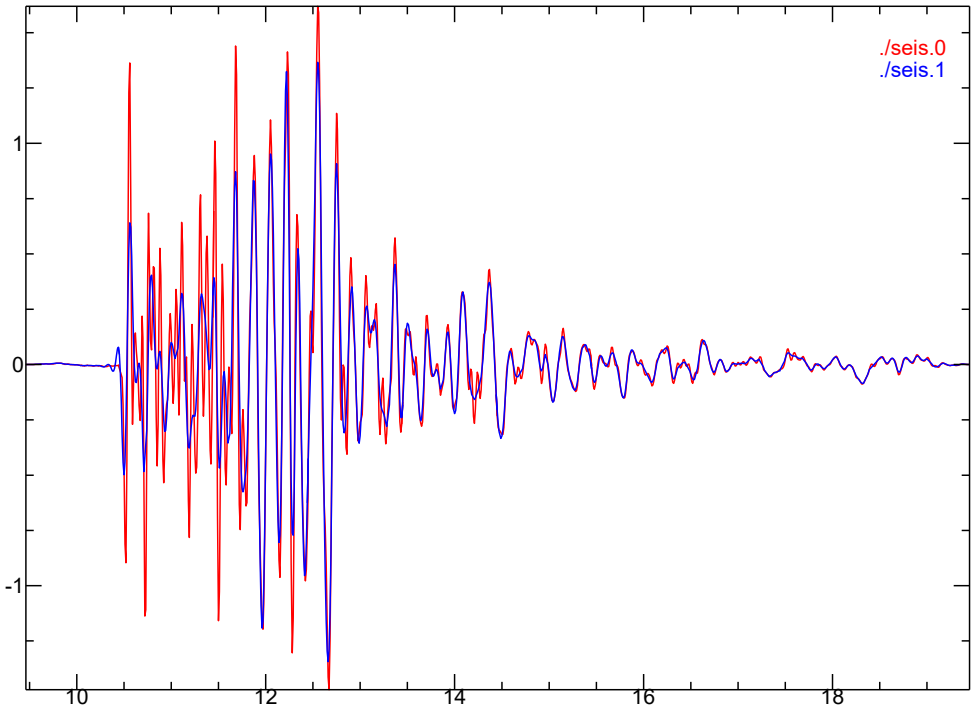

图 5.3: plot2 绘图效果。红色为滤波前波形，蓝色为滤波后波形。

5.3.4 plotpk

`plotpk` 是 OBSPY 中最常用的命令之一。其可以在窗口中显示指定个数的波形，所有波形共用 X 轴，但拥有单独的 Y 轴。该命令主要用于震相拾取，在 “震相拾取” 一节有详细介绍。

5.3.5 plotpm

`plotpm` 可以利用成对的波形数据，提取出任一时间段内两个波形数据的振幅信息，绘制在 “振幅-振幅” 图中。若一对波形数据恰好是同一台站两个互相垂直的分量，则 “振幅-振幅” 图即为 “质点运动图”。从 “质点运动图” 中，可以提取出震相的一些重要信息。

下面的例子利用垂直和径向分量的波形数据绘制 Rayleigh 面波的质点运动轨迹：

```
OBSPY> dg sub tele           // Z 分
nykl.z
OBSPY> dg sub tele nykl.e nykl.n // E、N 分量
OBSPY> rotate to gcp         // 旋转至大圆路径
OBSPY> w nykl.r nykl.t       // R、T 分量
OBSPY> r nykl.z nykl.r       // 读入 Z 和 R 分量
OBSPY> xlabel 'Radial
component' OBSPY> ylabel
```

```
OBSPY> title 'Particle-motion plot for partial Rayleigh wave'
OBSPY> xlim 1300 1340                    // 仅绘制 Rayleigh 面波的部分时间
窗 OBSPY> ppm                            // 绘制质点运动图
```

鉴于 `plotpm` 命令绘图的效果很糟糕，就不再贴效果图了，读者可以根据上面的命令自行绘制。

5.3.6 plotsp

`plotsp` 命令用于绘制不同格式的谱文件，可以绘制 “振幅 + 相位” 或者 “实部 + 虚部”，同时可以任意指定 X、Y 轴为线性轴或对数轴。

下面的命令对波形数据进行 FFT 得到谱文件，并使用 `plotsp` 命令绘制其振幅谱：

```
OBSPY> fg
seis OBSPY>
fft
```

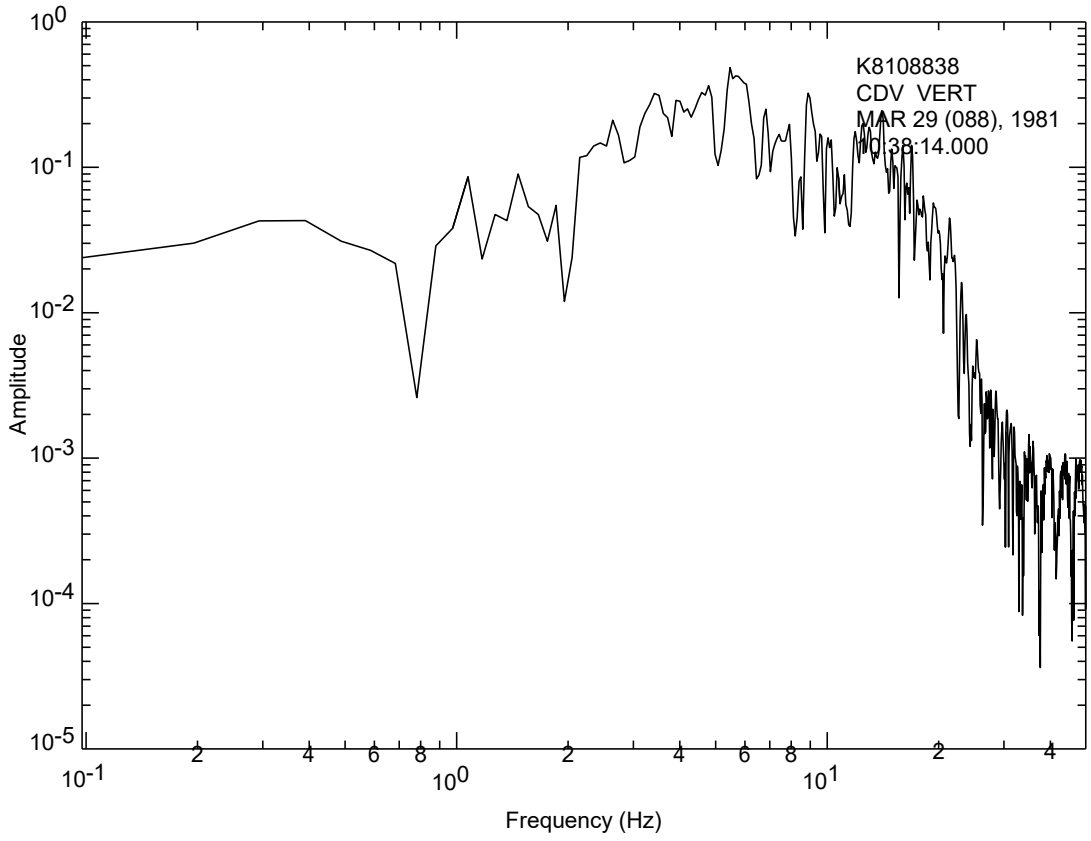

图 5.4: plotsp 绘制振幅谱

5.4 图像外观

5.4.1 图像元素

对于 OBSPY 而言，最基本的显示元素是将所有数据点用线连起来所构成的地震图。除此之外，OBSPY 的绘图命令还会在图像的四个边绘制坐标轴以及刻度，为图像添加标题、轴标签等。

图 5.5 展示了一个完整的 OBSPY 图像所包含的所有元素。

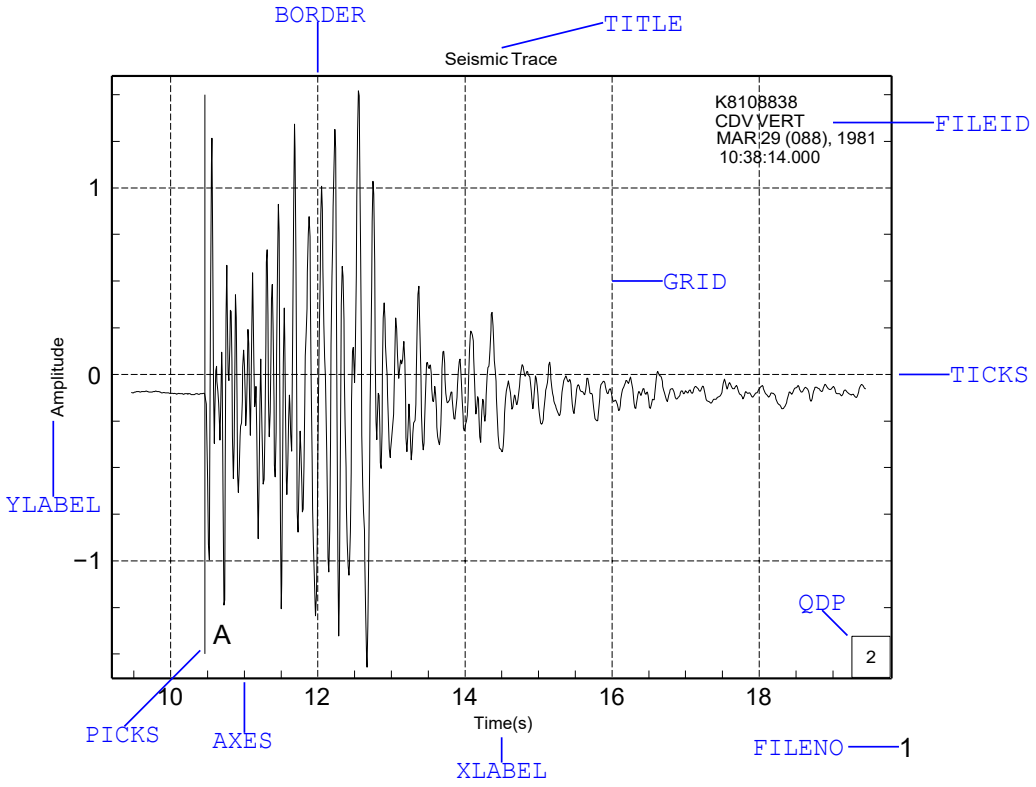

图 5.5: 绘图外观及其相关命令。图中蓝色部分为对绘图外观的说明。

图 5.5 可以用如下命令绘制得到：

```
OBSPY> fg                    // 生成数据
seis                         // 打开 QDP 选项（默认值即为
OBSPY> qdp                   开）
OBSPY> title 'Seismic Trace' // 设置标题
OBSPY> xlabel "Time(s)" // 设置 x 轴标
签 OBSPY> ylabel "Amplitude" // 设置 y
轴标签 OBSPY> filenumber on        //
显示文件号
OBSPY> axes only left bottom // left 和 bottom 显示
axes OBSPY> ticks only right   //   right 显示 ticks
OBSPY> border on             // top 显示 border
```

图像中显示的元素包括：

标签

标签大致可以分为三种：标题、轴标签和通用标签。

TITLE 图像的标题。`title` 命令可控制标题文本、位置和尺寸

XLABEL、**YLABEL** 轴标签。`xlabel` 和 `ylabel` 命令可指定 X 和 Y 轴标签文本、位置和尺寸。

PLABEL 通用标签。`plabel` 可指定通用标签的文本、位置和尺寸。 标签文本需要用单引号或双引号包围，文本尺寸选项 `size` 可以选择 `tiny`、`small`、`medium` 或 `large`，文本位置选项 `location` 则可以取 `top`、`bottom`、`left` 或 `right`。

可以通过 `plabel` 命令定义最多三个通用标签。通用标签与轴标签类似，其更通用之处在于可 以任意指定其位置。每个标签可以用 `position x y a` 来指定其位置，其中 x、y 为标签位置相对于

窗口尺寸的比例，a　表示标签相对于水平方向顺时针旋转的角度；也可以用　`below`　设置新标签位于上一标签的下方。

标记

图像中包含了如下标记：

FILEID　文件 ID。`fileid` 用于控制文件 ID 的内容、位置及其格式。

FILENO　文件号。`filenumber` 控制文件号显示与否。

PICKS　到时标记。`picks` 用于控制是否显示到时标记以及显示效果。

QDP　QDP 因子。`qdp` 用于控制 qdp 因子的大小。

QDP，全称为 “quick and dirty plot”。在开发 OBSPY 的那个年代，计算机的性能一般，若在绘图 时绘制全部数据点，则绘图过程会耗费大量时间。因而 OBSPY 采用了 “qdp” 的方式：每隔若干个数 据点绘制一个数据点 [1]。图中右下角的 “2” 即表示每两个点中绘制一个点。目前计算机的性能已经 足够强大，因而一般使用 `qdp off` 命令关于该选项。

框架

每张图都有一个框架，每个框架有 TOP、BOTTOM、LEFT 和 RIGHT 四条边。

OBSPY 中，每条边都可以用四种不同的形式表示：

- 不绘制；
- border：仅一条直线，即图 5.5 中 TOP 边；
- ticks：直线 + 刻度 [2]，即图中 RIGHT 边；
- axes：直线 + 刻度 + 标注 [3]，即图中 LEFT 边和 BOTTOM 边；

从上面的定义可以看到，四种形式的边存在包含与被包含的关系，因而在设定边时，有如下规则：

1. 用 `axes` 控制在哪些边使用 “axes”；
2. 只有不使用 “axes” 的边才可以用 `ticks` 命令控制是否使用 “ticks”；
3. 只有不使用 “axes” 和 “ticks” 的边才可以使用 `border` 命令控制是否使用 “border”；
4. 不使用 “axes”、“ticks” 和 “borders” 的边则不绘制。

除了边之外，还可以使用 `grid` 命令控制网格的显示以及网格的线型，或使用 `xgrid`、`ygrid` 分别控制横、纵方向网格的显示和属性。

5.4.2　图像控制

坐标轴

OBSPY 使用了优秀的默认算法，根据要绘制的数据范围选择合适的刻度间隔和标注。若对于默认 的结果不满意，可以使用 OBSPY 提供的命令分别对 X、Y 坐标轴进行调整，下面仅列出与 X 轴相关 的命令。

xlim　控制绘图的 X 轴范围

xdiv　控制 X 轴刻度间隔

xfudge　设定 fudge 因子，根据数据极值扩展 X 轴范围

[1] 本质上就是绘图时的一次　“减采样”，但是没有做抗混淆处理。

[2] 刻度专指每条边上的短线。

[3] 标注专指每条边上的数字。

坐标系

绘制时间序列一般使用线性坐标系，OBSPY 也提供了一系列命令以指定 X、Y 轴为线性坐标轴或 对数坐标轴。这些命令包括：linlin、linlog、loglin、loglog、xlin、xlog、ylin、ylog。

对于对数坐标轴，还有一些命令可以控制其外观，比如 xfull、loglab、floor。

5.4.3 线条属性

线条的属性包括线型（line）、线宽（width）、颜色（color）和符号（symbol）。 下面的命令展示了如何修改线条的属性。

```
OBSPY> fg seis
OBSPY> line 3      // 线型为
3 OBSPY> width 2  // 线宽为
2 OBSPY> color red      //
红色 OBSPY> p
```

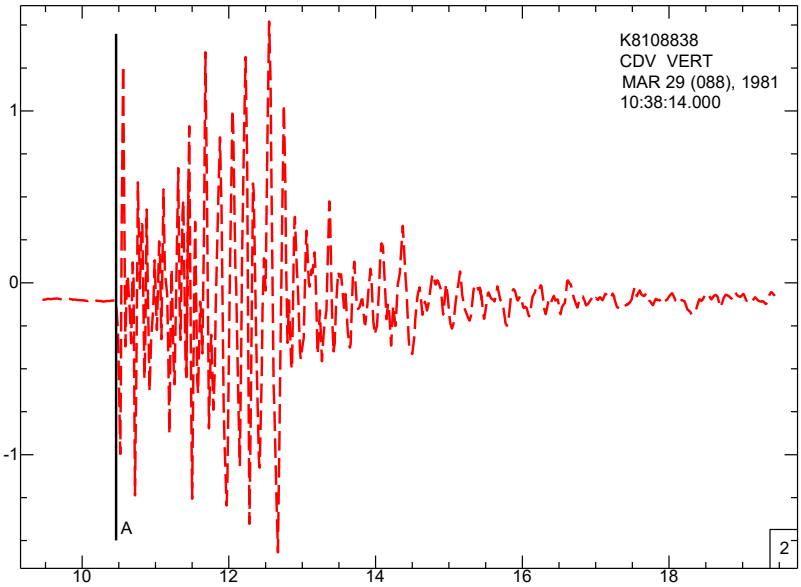

图 5.6: 线条属性

在绘制多个波形数据时，可以设置线条的属性按照某个列表递增。下面的命令一次绘制四个波 形文件，使每个数据的线型和颜色都按照默认列表递增。

```
OBSPY> dg sub teleseis ntkl.z nykl.z onkl.z
sdkl.z OBSPY> line incre
OBSPY> color black
incre OBSPY> p
```

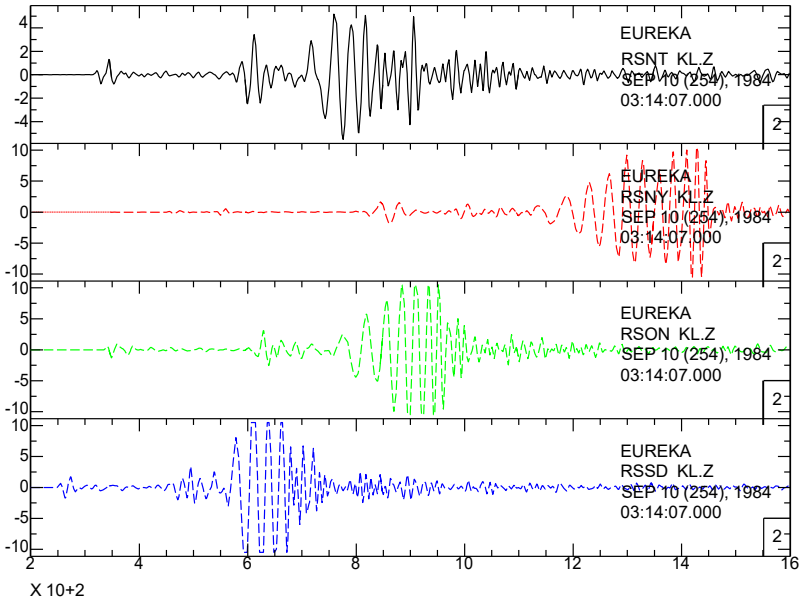

图 5.7: 线条属性递增

`line` 命令不仅可以设置线条的线型，同时可以对波形数据进行颜色填充：

```
OBSPY> fg
seis OBSPY>
qdp off
OBSPY> rmean; rtr; taper
OBSPY> line 0 fill
```

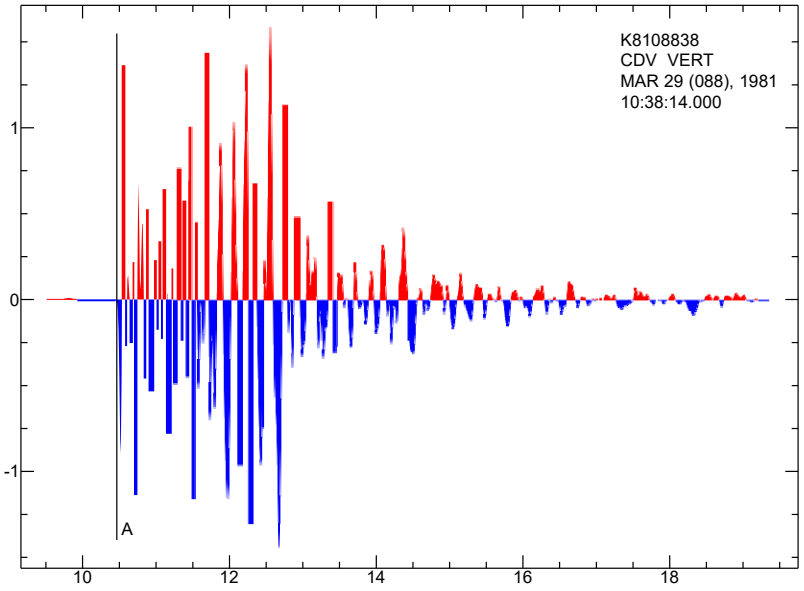

图 5.8: 颜色填充图

5.5 等值线图

OBSPY 中 `spectrogram` 等命令可以生成 IXYZ 数据（即 3D 数据），这种数据需要用等值线图来 展示。`contour` 命令用于等值线，`zcolors`、`zlabels`、`zlevels`、`zlines`、`zticks` 分别用于控制等值 线的颜色、标签、间距、线型以及刻度。

下面的例子中，读入了 XYZ 文件 contourdata，从头段中找出 Z 数据的范围。选择等值线范围为 700 km 到 1150 km，增量为 25 km。

选择包括四种线型的线型表，其中第一个为实线。这个列表将每四条等值线重复一次。然后给 等值线图起了个名字，最后绘制出来:

```
OBSPY> r ./contourdata
OBSPY> lh iftype depmin depmax

       IFTYPE = GENERAL XYZ (3-D) FILE
       DEPMIN = 6.977119e+02
       DEPMAX = 1.154419e+03
OBSPY> zlevels range 700 1150 increment 25
OBSPY> zlines list 1 2 3 4
OBSPY> title 'Katmai topography from survey data [inc = 25
km]' OBSPY> contour
```

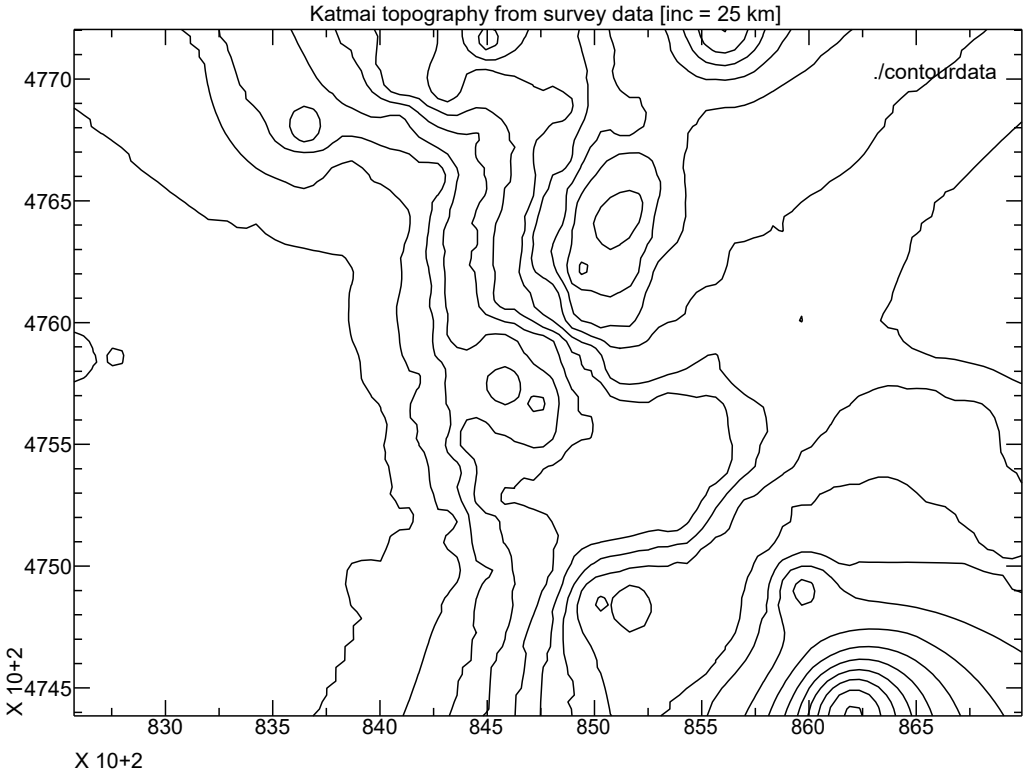

图 5.9: contour 绘制等值线 I

下面的例子中，使用同样的文件，但是显示选项不同。每四条等值线有一个整数标签。每条等值 线之间都有一个指向向下的箭头。所有等值线为实线型：

```
OBSPY> r ./contourdata
OBSPY> zlevels range 700 1150 increment
25 OBSPY> zlabels on list int off off
off
OBSPY> zticks on direction
down OBSPY> zlines list 1
OBSPY> title 'Katmai topography from survey data [labels and
```

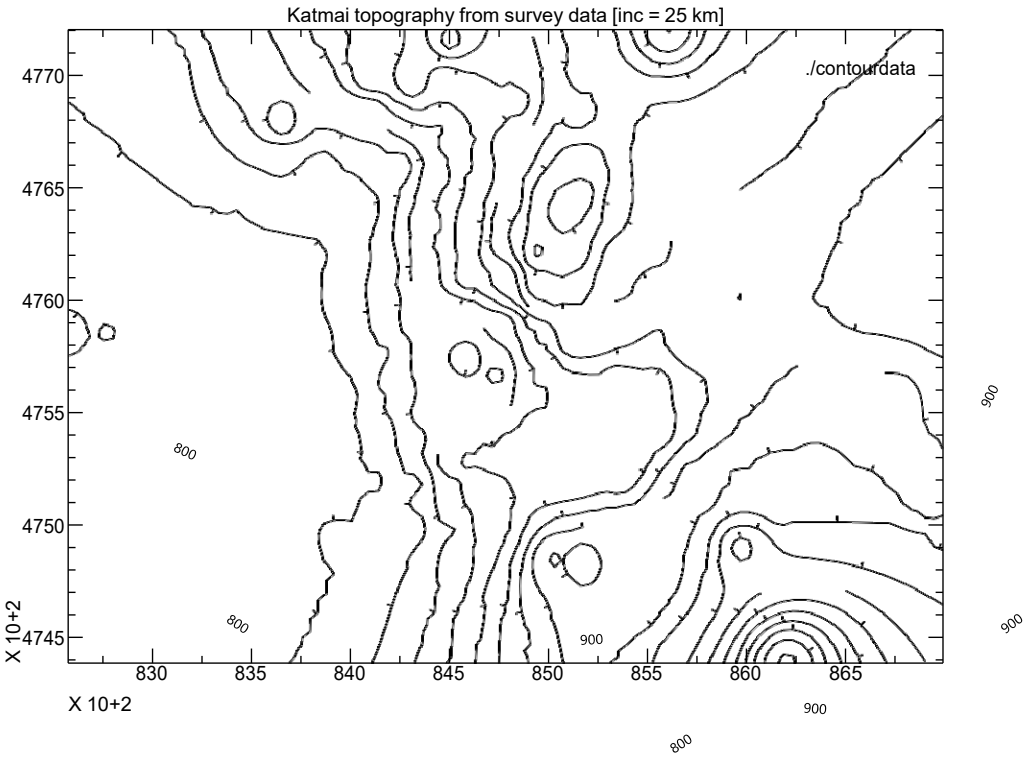

图 5.10: contour 绘制等值线图 II

5.6　组合图

前面介绍的绘图命令五花八门，但无论是 `plot`、`plot1` 或是 `plot2`，同一个窗口内绘制的所有波形总是共用同一个 X 轴。实际绘图时，经常需要在一张图中绘制多个不同 X 轴的图，即组合图。

OBSPY 提供了绘制组合图的功能，这其中牵涉到一些新的概念，其中之一是 `frame`。一般而言，在执行绘图命令时会首先对整个窗口进行擦除。比如，先执行 `plot` 命令，窗口中会显示出相应的波形，然后执行 `plot1` 命令，首先会将窗口中的已有图像全部擦除，再绘制相应波形。

在 frame 中，每次执行绘图命令时，不会擦除窗口中的已有图像，从而实现了将多个命令的绘图效果同时显示在一个窗口中。使用 `beginframe` 打开 frame 时，首先会擦除整个窗口，进入 “组合图模式”；当组合图绘制完成时，需要使用 `endframe` 命令关闭 frame。

除了 frame 之外，在绘制组合图时还需要了解与窗口有关的几个概念，如图 5.11：

- window：图形窗口。对于 xwindows 图形设备，window 如图 5.1 所示，其默认长宽比为 11.0/8.5=1.294；对于 sgf 图形设备，可以认为 window 的大小即为 A4 纸张的大小。
- viewspace：window 内可以用于绘图的部分；
- viewport：执行单个绘图命令时，图像的显示区域；

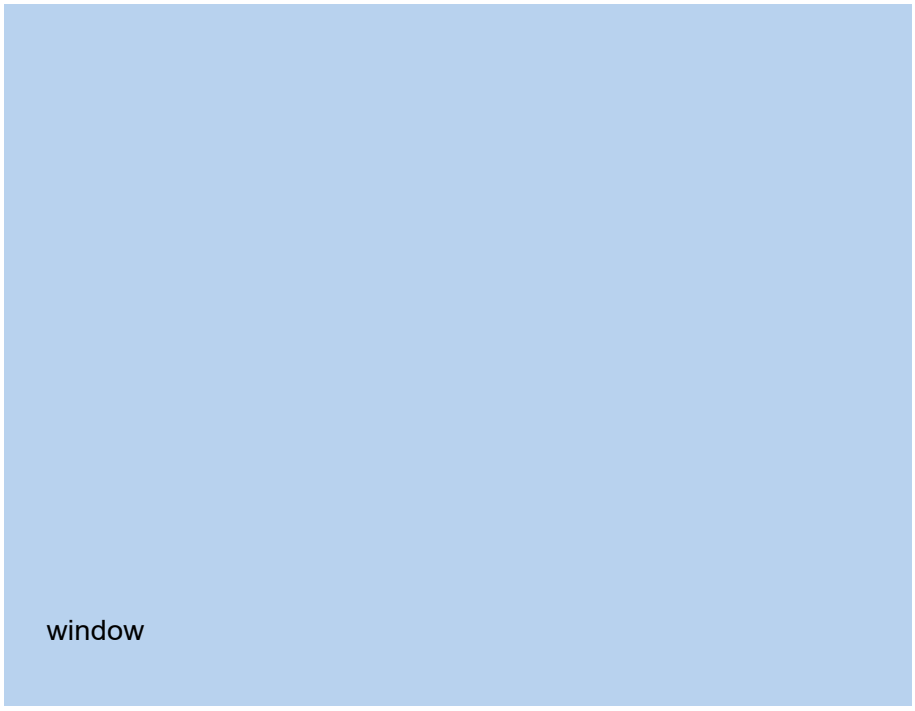

图 5.11: window、viewspace 和 viewport

图 5.11 中给出了 window、viewspace、viewport 的相互关系。可以使用 `window` 命令设定窗口相对于整个屏幕的位置以及X、Y方向的范围；`vspace` 用于设定整个绘图区的比例；`xvport` 和 `yvport` 则分别定义了单个绘图命令所能使用的X、Y方向的范围。

一个典型的组合图的绘制如下所示：

```
OBSPY> fg seis                    // 生成数据
OBSPY> beginframe                 // 打开 frame，开始绘制组合图
OBSPY> xvport 0.1 0.9             // 设定第一个绘图命令的
OBSPY> yvport 0.7 0.9             viewport
OBSPY> title 'Seismic
Trace' OBSPY> fileid off          // 设定标题
OBSPY> qdp                        // 不显示文件 id
off OBSPY> p
OBSPY> fft wmean
OBSPY> xvport .1 .45              // FFT
OBSPY> yvport .15 .55
OBSPY> title 'Amplitude Response (linlog)'
OBSPY> ylim 1e-5 1                // Y 轴范围
OBSPY> psp am linlog              // 绘制振幅谱
OBSPY> xvport .55 .9              // 设定第三个绘图命令的
viewport OBSPY> title 'Amplitude Response (loglog)'
OBSPY> xlim 1 60
OBSPY> psp am loglog              // 绘制振幅谱
OBSPY> endframe                   // 关闭 frame
```

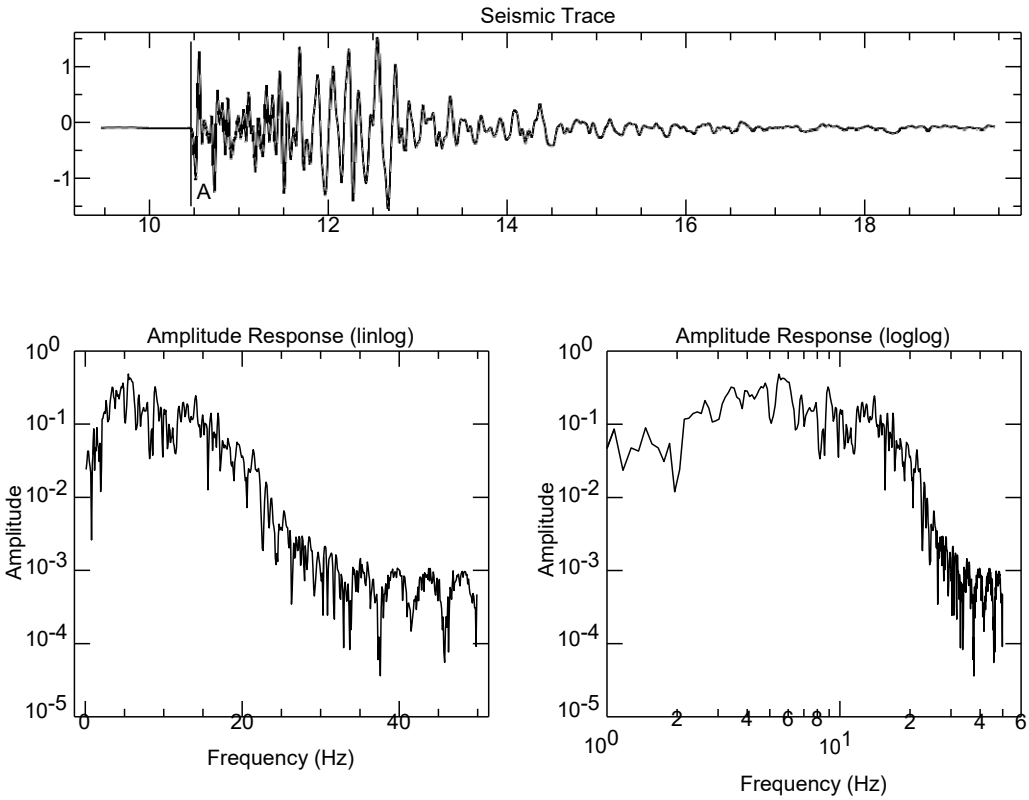

图 5.12: 绘制组合图

5.7　图像保存

5.7.1　xwindows

xwindows 是 OBSPY 中最常用绘图设备，对于震相拾取等交互式操作更是必不可缺。

```
OBSPY> fg seis
OBSPY> bd x     // begindevice xwinows，可省略
OBSPY> p        // 绘图
OBSPY> ed x     // enddevice xwindows，可省略
OBSPY> q
```

对于 xwindows，最简单的保存图像的方式是截图，常用的工具包括 gnome 下的 `screenshot` 或者 ImageMagick 的 `import` 命令。

5.7.2　sgf

SGF 图形设备会将图像信息保存到 SGF 文件中。其使用方式为：“启用 sgf 图形设备”→ “绘图到 sgf”→“关闭 sgf 设备，退出 OBSPY”→“将 sgf 文件转换为其它格式”。

```
OBSPY> fg
seis                    // 启动 sgf 设备，不可省
OBSPY> bd
sgf OBSPY>              // 关闭 sgf 设备，可省
p                       略
OBSPY> ed
sgf OBSPY>
```

生成的 sgf 文件可以通过 `sgftops` 等命令转换为其它图像格式，在 “sgftops” 中会介绍，也可以使用 `sgftox` 直接将 sgf 文件显示在绘图窗口中。

5.7.3　PS 和 PDF

自 101.5 之后，OBSPY 加入了 `saveimg` 命令，可以将当前 xwindow 或 sgf 图形设备中的图像保 存到其它 ps 或 pdf 格式的图像文件中，以获得更高质量的绘图效果。[1]

```
OBSPY> fg seis
OBSPY> p                    // 首先在 xwindows 上绘图
OBSPY> saveimg foo.ps       //  将 xwindows 上的图像保存到 foo.ps 中
save file foo.ps [PS]
OBSPY> q
```

5.7.4　psObspy

psObspy 是 Prof. Lupei Zhu 写的用于绘制 OBSPY 文件的 C 程序。该程序利用了 GMT 的 PS 绘 图库，直接读取 OBSPY 文件并绘制到 PS 文件中。得益于 GMT 的 PS 库的灵活性，利用 psObspy 可以 绘制出超高质量的复杂图像。具体参见 “psObspy” 一节。

5.7.5　小结

- 在 xwindows 上绘图简单省事，直接截图的效果较差，仅可用于非正式的演示；
- sgf 转换为其它图像格式稍显麻烦，但适合在脚本中批量做图；
- saveimg 生成图像文件质量相对较高，可以满足大多数需求；
- psObspy 功能强大，在一般绘图以及复杂图像时非常有用，适合在发表文章时使用；

5.8　图像格式转换

OBSPY 中的图像可以保存为 SGF、PS 和 PDF 格式，有些时候会需要将其转换为其他图像格式，比如 JPG 或 PNG。

`convert` 是 Linux 下的一个功能强大的图像格式转换工具。如果你的系统里没有这个命令，你可以通过安装 `ImageMagick` 来获取该命令。

`convert` 命令的选项众多，这里只说其中几个常用的选项：

- `-trim`：切边，即将图像周围多余的白边切除；
- `-density` *width*x*height*：设置图像精度，一般情况下，设置 width 为 300 即可，height 可以不指定。
- `-rotate` *degree*：图像旋转的角度；

下面给出一个简单的例子：

```
convert -trim -density 300x300 -rotate 90 image.ps image.png
```

[1] 该命令也支持输出为 png 和 xpm 格式，但 png 和 xpm 为位图图像格式，精度不够，且依赖于其它函数库，因而不推荐使用。

第 6 章　OBSPY 编程

一个基本的编程语言需要包含哪些特性呢？变量、参数、函数、条件判断、循环控制等等。 这一章介绍 OBSPY 所设计的一个基本的编程语言，在官方文档中直接称其为 OBSPY 宏。OBSPY 与
OBSPY 宏的关系在某种程度上更像是 matlab 与 matlab 脚本之间的关系。最简单的，将一系列要一起执行的 OBSPY 命令放在一个文件中即构成了 OBSPY 宏文件。 本文档中使用了稍有不同的说法，并将这一章命令为 “OBSPY 编程”。其包含了三个主要的部分：

- 变量：由于 OBSPY 的特殊性，又分为头段变量和一般变量；
- 内置函数：基本的数学和字符串函数；
- OBSPY 宏：参数、条件判断、循环控制等；

在官方文档中，变量和内置函数都是 OBSPY 宏的一部分，本文档将其从 OBSPY 宏中提取出来是出 于如下几个方面的考虑：

1. 变量和内置函数既可以在 OBSPY 命令中使用，也可以在 OBSPY 宏中使用；而其它特性如参数、 条件判断、循环控制等几乎只能在 OBSPY 宏中使用；
2. OBSPY 设计的编程语言功能简单，不够友好；非常建议使用 Bash、Perl 或 Python 这些更成熟 的脚本语言来替代 OBSPY 的编程功能。宏参数、条件判断、循环控制等特性都可以被脚本的相 应功能完全取代，而由于 OBSPY 设计的特殊性，诸如变量和内置函数等特性在某些情况下不能 完全取代。

所以，建议的做法是读完本章的内容，掌握如何引用头段变量、如何使用黑板变量以及内联函数，简单了解 OBSPY 宏的特性，选择 Perl 或者 Python 脚本语言[1] 进行数据批量处理，尽量避免使用 黑板变量和内联函数。就目前的个人经验而言，在脚本语言中，OBSPY 的引用头段变量功能必不可少， 黑板变量和内联函数这两个功能或多或少都可以被取代。

另外，尤其需要注意的是，OBSPY 自 101.6 起彻底重写了 OBSPY 宏的语法分析器，因而导致 101.6
以后的 OBSPY 宏与之前的 OBSPY 宏有很大不同，本文档只讨论 101.6 重写的 OBSPY 宏语法。

6.1　引用头段变量值

前面已经介绍了 OBSPY 中的很多头段变量，也知道如何使用 `listhdr` 查看头段变量的值，`lh` 命 令的输出对于人来说很直观，但是对于机器来说却很不友好。有些时候需要直接使用头段变量的值，这就需要一些特殊的技巧。

最常见的情况是第 39 页给出的例子。在使用 “`ch o gmt`” 指定发震时刻后，需要获取头段变量 `o` 的值，对该值取负值，并用于 “`ch allt`” 中。

本例中，需要先获取头段变量 `o` 的值，再将其值用于其它命令中，准确的说这叫变量值的引用。

[1] Bash 在很多地方还是不如 Perl 或 Python 方便，不推荐。

在 OBSPY 命令中引用头段变量的值有两种方式，分别是“&fname,header&”和“&fno,header&”[2]。

fname 和 fno 都唯一指向了内存中的某个波形数据，其中 fname 表示文件名，fno 表示文件号（即内存中的第几个文件，索引值从1开始），header 则为头段变量名。 下例展示了如何通过两种方式引用头段变量的值：

```
OBSPY> fg seis
OBSPY> w
seis.OBSPY                    // 注意"./"
OBSPY>                        // 查看三个头段变量的值

     kevnm = K8108838
         o = -4.143000e+01
      stla = 4.800000e+01
OBSPY> echo on processed      // 打开回显，显示处理信息
OBSPY> ch kuser0 &1,kevnm&    // 通过文件号引用头段变量
kevnm
 ==> ch kuser0 K8108838       // 实际执行的效果
OBSPY> ch user0 &./seis.OBSPY,o& // 利用文件名，引用头段
变量 o
 ==> ch user0 -41.43
OBSPY> ch user1 &seis.OBSPY,stla& // 文件名少了
"./" ERROR 1363: Illegal data file list name:
seis.OBSPY OBSPY> lh kuser0 user0 user1
```

在通过文件名指定波形数据时要注意：OBSPY 记录的是文件的全路径。一般情况下，使用文件号 会更方便些。

6.2　黑板变量

既然是 OBSPY 编程，就必然少不了变量，OBSPY 中的变量称之为黑板变量。

黑板变量是 OBSPY 中用于临时储存和取回信息而设计的。黑板变量不需要声明即可直接使用，可 以用 setbb 和 evaluate 命令给黑板变量赋值，用 getbb 获取黑板变量的值。也可以用 writebbf 将黑板变量保存在磁盘文件中，然后使用 readbbf 命令重新将这些变量读入 OBSPY 中。

引用黑板变量的值的方式为：“%bbvname%”，其中 bbvname 为黑板变量的变量名。

```
OBSPY> echo on
processed
OBSPY> fg
seis OBSPY>                   // 黑板变量 low=2.45
p                             // 黑板变量 high=4.94
OBSPY> setbb low 2.45         // 引用黑板变量 low 和 high 的值作为滤波的频
OBSPY> setbb high             带
4.94 OBSPY> bp
c %low% %high%                // 查看黑板变量的值
```

[2] 实际上，OBSPY 官方文档给出的引用方式中没有末尾的 & 符号，仅当一些特殊的情况下才使用，这样容易使得整个语法混乱不堪，所以这里采用了另外一种引用方式。所有示例均已通过测试。

```
low = 2.45
high = 4.94
```

下例展示了如何将黑板变量写入磁盘文件，等需要时再从磁盘文件中获取：

```
$ obspy
OBSPY> setbb var1 10      // 整型
OBSPY> setbb var2 "text"  // 字
符串 OBSPY> setbb var3 0.2      //
浮点型
OBSPY> wbbf bbf.file      // 写入到文件
OBSPY> q
$ ls
bbf.file
$ obspy
OBSPY>
readbbf ./bbf.file
OBSPY> getbb
 NUMERROR = 0
 OBSPYERROR =
 'FALSE' OBSPYNFILES
 = 0
 VAR1 = 10
 VAR2 = 'text'
 VAR3 = 0.2
OBSPY> getbb var2
```

6.3　内联函数

内联函数是 OBSPY 实现的一些函数，其可以在 OBSPY 命令中使用。在执行命令时，内联函数会首 先被调用，内联函数的结果将替代命令中的内联函数的位置。

OBSPY 提供了如下几类内联函数：

- 算术运算符；
- 常规算术运算函数；
- 字符串操作函数；
- 其他函数；

所有的内联函数的共同形式是：“(func)”，其中 func 为内联函数名。某种程度上，内联函数与前面说到头段变量（“& &”）和黑板变量（“% %”）类似，可以认为是通过“( )”引用了内联函数的结果或值。

内联函数支持嵌套，目前最多可以嵌套 10 层。

6.3.1　算术运算符

算术运算符即常规的加减乘除运算符，但又有不同，其一般形式如下：

```
( number operator number )
```

所有的操作数都被认为是实型的，所有的算术运算都按照双精度浮点型进行运算；

OBSPY 支持的操作符是包括：“ + - * / ** ”。 看几个简单的例子:

```
OBSPY> echo on
OBSPY> setbb var1                    // 忘记加括号了！"4+7"被当成了字符串
 4+7 setbb var1 4+7
OBSPY> setbb var2
 (4+7) setbb var2
 (4+7)                               // 4+7=11
 ==> setbb var2 11                   // 优先级正确
OBSPY> setbb var3
 (4+7/3) setbb var3
 (4+7/3)                             // 括号改变优先级
 ==> setbb var3 6.33333              // 可以看作是内联函数的嵌套
OBSPY> setbb var4
OBSPY> setbb var1 ( ( 4 + 7 ) / 3 ) // 支持
 空格
 setbb var1 ( ( 4 + 7 ) / 3 )
```

6.3.2 常规算术运算函数

OBSPY 提供了 20 个常规算术运算函数，其基本形式为“(func arg1 arg2 ...)”。具体函数如 表 6.1 所示。

演示如下：

```
OBSPY> echo on processed
OBSPY> setbb var1 (add 1 3 4)  // 1+3+4
 ==> setbb var1 8
OBSPY> setbb var2 (subtract 1 3 4) // 1-3-4
 ==> setbb var2 -6
OBSPY> setbb var3 (multiply 1 3 4) // 1*3*4
 ==> setbb var3 12
OBSPY> setbb var4 (divide 1 3 4) // 1/3/4
 ==> setbb var4 0.0833333
OBSPY> setbb var5 (absolute -5.1) // abs(-5.1)
 ==> setbb var5 5.1
OBSPY> setbb var6 (power 5)     // 10^5
 ==> setbb var6 100000
OBSPY> setbb var7 (alog10 10000) //
log10(10000)
OBSPY> setbb var8 (alog          // ln(10000)
10000)
 ==> setbb var8 9.21034          // e^5
OBSPY> setbb var9 (exp
5)                               // sqrt(9)
 ==> setbb var9 148.413
OBSPY> setbb var10 (sqrt         // PI
9)
 OBSPY> setbb var12 (sine (pi/6)) //
 sin(30)
```

表 6.1: 常规算数运算函数

命令	语法	功能
add	(add v1 v2 ... vn)	v1+v2+...+vn
subtract	(subtract v1 v2 ... vn)	v1-v2-...-vn
multiply	(multiply v1 v2 ... vn)	v1*v2*...*vn
divide	(divide v1 v2 ... vn)	v1/v2/.../vn
absolute	(absolute v)	取绝对值
abs	(abs v)	取绝对值
power	(power v)	取 10 的 v 次方
alog10	(alog10 v)	以 10 为底取 v 的对数
alog	(alog v)	取 v 的自然对数
exp	(exp v)	取 e 的 v 次方
sqrt	(sqrt v)	求 v 的平方根
pi	(pi)	返回 pi 值
sine	(sine v)	正弦 (v 为弧度，下同)
cosine	(cosine v)	余弦
tangent	(tangent v)	正切
arcsine	(arcsine v)	反正弦
arccosine	(arccosine v)	反余弦
arctangent	(arctangent v)	反正切
integer	(integer v)	取整
maximum	(maximum v1 v2 ... vn)	求最大值
minimum	(minimum v1 v2 ... vn)	求最小值

```
OBSPY> setbb var13 ((arcsine 0.5)*180/(pi))
 ==> setbb var13 30
OBSPY> setbb var14 (integer 3.11)
 ==> setbb var14 3
OBSPY> setbb var15 (max 3.11 -1.5 5) // maximum 简写为 max
 ==> setbb var15 5
OBSPY> setbb var16 (min 3.11 -1.5 5) // minimum 简写为 min
 ==> setbb var16 -1.5
```

为了对一组数据做归一化，首先要找到所有数据中的绝对最大值，如下：

```
OBSPY> r file1 file2 file3
file4 OBSPY> echo on processed
OBSPY> setbb vmax (max &1,depmax& &2,depmax& &3,depmax& &4,depmax&)
 ==> setbb vmax 1.87324
OBSPY> setbb vmin (min &1,depmin& &2,depmin& &3,depmin& &4,depmin&)
 ==> setbb vmin -2.123371
OBSPY> div ( max (abs %vmax%) (abs %vmin%) ) // 嵌套
 ==> div 2.123371
```

此例可以通过多重嵌套的方式在单个命令中完成，但上面的写法可读性更强。

6.3.3 字符串操作函数

OBSPY 提供了若干个函数用于字符串的处理，如表 6.2 所示：

表 6.2: 字符串操作函数

命令	语法（简写形式）	功能
change	(cha s1 s2 s3)	在 s3 中用 s1 代替 s2
substring	(substring n1 n2 s)	取 s 中第 n1 到第 n2 个字符
delete	(del s1 s2)	从 s2 中删去 s1
concatenate	(conc s1 s2 ... sn)	将多个字符串拼接起来
before	(bef s1 s2)	得到 s2 中位于 s1 前的部分字符串
after	(aft s1 s2)	得到 s2 中位于 s1 后的部分字符串
reply	(rep s1)	发送信息 s1 到终端并得到回应

下面的例子展示了部分函数的用法：

```
OBSPY> echo on processed
OBSPY> setbb var1 (cha short long "this is short")
 ==> setbb var1 this is long
OBSPY> set var2 (del def abcdefghi)
 ==> set var2 abcghi
OBSPY> set var4 (before de abcdefg)
 ==> set var4 abc
OBSPY> set var4 (after de abcdefg)
 ==> set var4 fg
OBSPY> fg seis
OBSPY> setbb month (substring 1 3 &1,kzdate&)
 ==> setbb month MAR
OBSPY> setbb val "1234567890"
OBSPY> message (substring 1 5 %val%)
 ==> message 12345
 12345
```

下面的例子展示 concatenate 函数的用法以及如何灵活定义标题：

```
OBSPY> fg seis
OBSPY> echo on processed
OBSPY> setbb var (conc Seismogram of &1,kevnm& &1,kstnm&)
 ==> setbb var SeismogramofK8108838CDV              // 没有空格
OBSPY> setbb var (conc "Seismogram of " &1,kevnm& " " &1,kstnm&)
 ==> setbb var Seismogram of K8108838 CDV           // 含空格
OBSPY> getbb var
 var = 'Seismogram of K8108838 CDV'
OBSPY> title (conc "Seismogram of " &1,kevnm& " " &1,kstnm&)
 ==> title Seismogram of K8108838 CDV               // 错误标题！
OBSPY> title '(conc "Seismogram of " &1,kevnm& " " &1,kstnm&)'
 ==> title "(conc "Seismogram of " K8108838 " " CDV)" // 错误标题！
OBSPY> title "Seismogram of &1,kevnm& &1,kstnm&"
 ==> title "Seismogram of K8108838 CDV"             // 正确标题！
```

下面的例子使用 reply 函数实现了交互：

```
OBSPY> fg seis
OBSPY> echo on
processed OBSPY>
rmean; rtr; taper
OBSPY> setbb low (reply "Enter low freqency limit for bandpass:
") Enter low freqency limit for bandpass: 2.1 // 用户输入 2.1
 ==> setbb low 2.1
OBSPY> setbb high (reply "Enter low freqency limit for bandpass:
") Enter low freqency limit for bandpass: 6.5 // 用户输入 6.5
 ==> setbb high 6.5
OBSPY> bp
```

下面的例子中 reply 函数包含了一个默认值值：

```
OBSPY> setbb bbday (reply "Enter the day of the week:
[Monday]") Enter the day of the week: [Monday]Tuesday // 用户输
入 Tuesday OBSPY> getbb bbday
 bbday = 'Tuesday'
OBSPY> setbb bbday (reply "Enter the day of the week:
[Monday]") Enter the day of the week: [Monday]          // 用户无
输入
OBSPY> getbb
```

当 reply 函数执行时，引号中的字符串将出现在屏幕上，提示用户输入。如果用户输入，OBSPY 会将 输入的字符串作为返回值，如果用户只是敲击回车键，OBSPY 则会使用该默认值“MONDAY”。

6.3.4 其他函数

这类函数目前只有一个：gettime，其语法为“(gettime max|min [value])”。

gettime 函数用于返回数据中首先出现大于或小于 value 的时间相对于文件参考时刻的相对时间；若没有指定 value，max 会返回文件中第一个最大值的相对时间，min 会返回文件中第一个最小值的相对时间。

对于所有的文件有一个最大振幅，要找到这些文件中第一个文件中第一次大于该值所对应的时 间偏移量：

```
OBSPY> fg seis
OBSPY> echo on processed
OBSPY> setbb maxtime (gettime max)
 ==> setbb maxtime 12.55
OBSPY> setbb mintime (gettime min)
 ==> setbb mintime 12.67
```

为了找到第一个大于或等于 1.0 的数据点的时间偏移，可以使用如下命令：

```
OBSPY> fg seis
OBSPY> echo on processed
OBSPY> setbb valuetime ( gettime max 1.0 )
 ==> setbb valuatime 10.55
```

6.4 OBSPY 宏

6.4.1 简单的例子

假如你有一些重复的工作需要完成，那么 OBSPY 宏显然可以帮你节省不少时间。例如，要经常读 取三个文件 ABC、DEF 和 XYZ，每个文件分别乘以不同的值，做 Fourier 变换，然后将频谱的振幅 部分绘制到 SGF 文件中，这样的一系列命令可以写入到 OBSPY 宏文件中：

```
** This certainly is a simple little macro.
r ABC DEF XYZ
mul 4 8 9
fft
bg sgf
psp am
```

假设上面的代码保存到文件 `mystuff` 中，且该文件位于当前目录中，可以通过下面的命令执行该宏文件：

```
OBSPY> macro mystuff
```

终端中并不会显示正在执行的宏文件中的命令，可以使用 echo 命令来设置在终端显示哪些东西。另外，若某行的第一列为星号则该行为注释行，OBSPY 不会去执行注释行。

6.4.2 宏搜索路径

当你执行一个宏文件而又没有给出宏文件的绝对路径时，OBSPY 会按照下面的路径顺序搜索宏 文件：

1. 在当前目录搜索；
2. 在 `setmacro` 命令设置的搜索目录中搜索；
3. 在 OBSPY 的全局宏目录（`${OBSPYHOME}/aux/macros`）中搜索； 所有人都可以使用全局宏目录中的宏文件，可以使用 `installmacro` 命令将自己的宏文件安装到这个目录中。你也可以通过绝对/相对路径指定搜索路径。

6.4.3 宏参数

如果想要每次读取不同的文件或者乘以不同的值那么必须每次都修改该文件，让宏文件在执行 之前允许用户输入参数可以大大增加宏文件的灵活性。

OBSPY 宏参数的格式为：“`$n$`”，其中 n 从 1 开始。 下面将对先前的宏文件进行修改以使其可以接收文件名作为参数：

```
r
$1$ $2$ $3$
mul 4 8 9
fft
bg sgf
```

“`$1$`”、“`$2$`”和“`$3$`”分别表示宏文件接收到的第一、二、三个参数，用下面的命令执行这个宏文件：

```
OBSPY> macro mystuff ABC DEF XYZ
```

可以用下面的命令再次执行这个宏文件，但读取不同的文件：

```
OBSPY> macro mystuff AAA BBB CCC
```

6.4.4　关键字驱动参数

关键字驱动参数允许用户按照任意顺序输入参数，这也使得宏文件的内容变得简单易懂。 当参数的数目以及宏文件的大小不断增大的时候这就变得更加重要了。下面将再一次修改这个例子以使其可以接受文件列表以及乘数的列表：

```
$keys$ files values
r  $files$
mul
$values$ fft
bg sgf
psp am
```

`$keys$` 表明 “files” 和 “values” 是关键字。可以按照下面的输入来执行这个宏文件：

```
OBSPY> macro mystuff files ABC DEF XYZ values 4 8 9
```

因为参数的顺序不再重要，所以你可以像下面这样输入：

```
OBSPY> macro mystuff values 4 8 9 files ABC DEF XYZ
```

这个宏文件并不限于读取三个文件，它对于文件的数目没有限制，只要文件数与值数目相匹配就好。

6.4.5　宏参数缺省值

有些时候会遇到这样的情况，宏文件的有些参数在多次执行的过程中经常但并不总是拥有相同 的值。为这些参数提供缺省值可以减少输入那些相同值的次数同时又保有宏参数本身的灵活性。如 下例所示：

```
$keys$ files values
$default$ values 4 8 9
r  $files$
mul
$values$ fft
bg sgf
psp am
```

`$default$` 指定了宏参数 `values` 的缺省值，若在执行宏文件时不输入 values 的参数值那么这些参数将使用缺省值：

```
OBSPY> macro mystuff files ABC DEF XYZ
```

如果想要使用不同的值，可以像下面这样输入：

```
OBSPY> macro mystuff values 10 12 3 files ABC DEF XYZ
```

6.4.6　参数请求

若执行宏文件时没有输入参数而这些参数又没有缺省值，OBSPY 会在终端中提示你输入相应的 参数值。在上面的例子中，如果你忘记输入参数则会出现下面的情况：

```
OBSPY> macro
mystuff                    // 用户输入 ABC DEF
```

注意到 OBSPY 并不会提示输入参数 `values` 的值，因为它们已经有了缺省值。OBSPY 并非在一开始就 提示输入参数，其等到需要计算参数值却发现没有缺省值或者输入值时才会提示需要输入该参数。

6.4.7　联接

头段变量、黑板变量、宏参数以及字符串可以直接联接在一起。

```
$keys$  station
fg  seis
echo on
setbb  sta  $station$.z
setbb tmp ABC
setbb tmp1 XYZ%tmp%
setbb tmp2 (&1,o&)
setbb fname $station$%tmp%%tmp1%%tmp2%.OBSPY
```

执行效果如下：

```
OBSPY> m stuff station
 STA setbb  sta
 $station$.z
 ==> setbb sta STA.z
 setbb tmp ABC
 setbb tmp1 XYZ%tmp%
 ==> setbb tmp1 XYZABC
 setbb  tmp2  @(&1,o&@)
 ==> setbb tmp2 (-41.43)
 setbb fname $station$%tmp%%tmp1%%tmp2%.OBSPY
```

6.4.8　条件判断

条件判断在任何一个编程语言中都是必不可少的，OBSPY 宏的条件判断语句与 Fortran77 类似，但不完全相同，要注意区分。

OBSPY 宏的条件判断格式如下：

```
 IF expr
       commands
 ELSEIF  expr
       commands
 ELSE
       commands
 ENDIF
```

逻辑表达式 expr 具有如下形式：

```
   token 关系运算符 token
```

其中 `token` 可以是一个常数、宏参数、黑板变量或头段变量，关系运算符则是 `GT`、`GE`、`LE`、`LT`、`EQ`、`NE` 中的一个。上面的逻辑表达式在计算之前 `token` 会被转换为浮点型数。

条件判断语句目前最多支持 10 次嵌套，且 elseif、else 是可选的，elseif 的次数没有限制。下面给出一个例子：

```
r
$1$ ma
rkptp
if &1,user0& ge 2.45
    fft
    psp am
else
    message "Peak to peak for $1 below threshold."
```

在这个例子中，一个文件被读入内存，markptp 测出其最大峰峰值，并保存到头段变量 user0 中，若该值大于某一确定值，则对其做Fourier变换并绘制振幅图，否则输出信息到终端。

6.4.9　循环控制

循环特性允许在一个宏文件中重复执行一系列命令。通过固定循环次数、遍历元素列表或者设 定条件来执行一系列命令，也可以随时中断一次循环。循环的最大嵌套次数为 10 次。其语法可以有多种形式：

```
DO variable = start, stop [,increment]
    commands
ENDDO
```

```
DO variable FROM start TO stop [BY increment]
    commands
ENDDO
```

```
DO variable LIST entrylist
    commands
ENDDO
```

```
DO variable WILD [DIR name] entrylist
    commands
ENDDO
```

```
WHILE expr
    commands
ENDDO
```

其中大写字符串均为关键字，不可更改：

- variable 是循环变量名，在变量名前后加上 “$” 即可在 do 循环中引用该变量；
- start、stop、increment 循环变量的初值、终值、增值，start、stop 必须为整型数，increment 缺省值为1
- entrylist 是 do 循环执行时变量可以取的所有值的集合，值之间以空格分开，其可以为整型、浮点型或字符型。DO WILD 中 entrylist 由字符串和通配符构成，循环执行前，这个列表将根据通配符扩展为一系列文件名。

下面给出一些 DO 循环的例子：

该宏文件对数据使用了 dif 以进行预白化处理，进行 Fourier 变换，然后使用 divomega 命令去除预白化的影响，有时需要在做变换之前多次预白化，那么就可以这样写：

```
$keys$ file nprew
$default$ nprew 1
r  $file
do j = 1 ,
    $nprew$ dif
enddo
fft amph
do j = 1 ,
    $nprew$ divom
    ega
```

下面这个例子，用相同的数据绘制 5 个不同的两秒时间窗的质点运动矢量图：

```
r abc.r abc.t
setbb time1 0
do time2 from 2 to 10 by 2
    xlim %time1% $time2$
    title 'Particle motion from %time1% to $time2$'
    plotpm
    setbb time1
$time2$ enddo
```

在下面的例子中，一个宏文件调用另一个名为 preview 的宏文件，通过 do 循环以达到多次调用 preview 的目的：

```
do station list abc def xyz
    do component list z n e
        macro preview
    $station$.$component$ enddo
enddo
```

在下面的示例展示了如何处理目录 mydir 中所有以 .Z 结束的文件：

```
do file wild dir mydir *.Z
    macro preview $file$
enddo
```

最后一个例子有三个参数，第一个是文件名，第二个是一个常数，第三个是一个阀值。宏文件读 取了一个数据文件，然后每个数据点乘以一个常数直到其超过某一阀值：

```
r $1$
while &1,depmax& gt
    $3$ mul $2$
enddo
```

另一个与 break 有关的宏文件：

```
r $1$
```

```
while 1 gt 0
    div $2
    if &1,depmax& gt
        $3$ break
    endif
enddo
```

这个 while 循环是一个无限循环，它只能通过 break 来中断。

6.4.10　嵌套与递归

OBSPY 宏提供嵌套功能，不支持递归，但是 OBSPY 并不会去检查宏的调用是否保证不是递归，因 而需要用户去保证宏文件不要直接或间接调用自己。

6.4.11　中断宏

有些时候需要临时中断宏文件的执行，用户自己从终端输入一些命令，然后继续执行宏文件。这 个可以利用 OBSPY 的 pause 和 resume 特性做到。当 OBSPY 在宏文件中遇到 $TERMINAL$ 时会临时停 止执行宏文件，更改提示符为宏名，然后提示从终端输入命令，然后当 OBSPY 在终端中看到 $RESUME$ 时则会停止从终端读取命令继续从宏文件读取。如果你不想再继续执行宏文件中的命令，可以在终 端输入 $KILL$，OBSPY 将关闭宏文件，回到上一层。在一个宏文件中可以有多个 $TERMINAL$ 中断。

6.4.12　调用外部程序

你可以在 OBSPY 宏内部执行其他程序，可以向程序传递参数。如果程序是交互式的你也可以将 输入行发送给它，语法如下：

```
$RUN$ program message
inputlines
ENDRUN
```

宏参数、黑板变量、头段变量、内联函数均可使用，在程序执行之前它们会被计算，当程序执行结束，OBSPY 宏会在 ENDRUN 之后继续执行。

6.4.13　转义字符

字符“$”和“%”在 OBSPY 中具有特殊的含义，有时在字符串中需要使用这些特殊字符，但 OBSPY 会将其解释成一个变量，此时就需要使用转义字符，OBSPY 中的转义符为“@”，可以被转义的特殊符 号包括：

- $ 宏参数标识符
- % 黑板变量标识符
- & 头段变量标识符
- @ 转义字符本身
- () 内联函数起始符

保护环境，从阅读电子文档开始！

第 7 章　脚本中调用 OBSPY

7.1　脚本语言

日常工作中有大量的数据需要处理，因而需要将数据处理尽可能地自动化以加快数据处理速度 并减少人力支出。这就需要脚本来实现批处理。

前一章介绍的 OBSPY 宏，从某种意义上来说，就是一种脚本语言，而 OBSPY 就是这种脚本语言的 解释器。不推荐使用 OBSPY 宏的原因在于，OBSPY 宏提供的功能过于简单，无法满足日常需求。正如上 一章开头所说，读者只需要了解变量即可，不需要去学习 OBSPY 宏。

脚本语言有很多，地震学领域经常使用的脚本语言包含 Bash、Perl 和 Python。不推荐使用 Bash，推荐使用 Perl 或 Python 中的任意一个。

7.2　Bash 中调用 OBSPY

7.2.1　简介

OBSPY 宏的功能相对比较单一，难以满足日常数据处理的需求，可以在 Bash 脚本中直接调用
OBSPY，这样可以利用 Bash 脚本的更多特性。 下面的例子展示了如

何在 Bash 脚本中调用 OBSPY：

```
#!/bin/bash
OBSPY_DISPLAY_COPYRIGHT=0

Obspy << EOF
fg seis
lh evla evlo
q               # 必须!
EOF
```

OBSPY 在启动是默认会显示版本信息，当用脚本多次调用 OBSPY 时，版本信息也会显示多次，可 以通过设置变量`OBSPY_DISPLAY_COPYRIGHT=0`的方式隐藏版本信息。

脚本中从“`Obspy << EOF`”开始到“`EOF`”的全部内容，都会被 Bash 传递给 OBSPY，OBSPY 会逐一解 释并执行每行命令。

7.2.2　头段变量和黑板变量

想要在 Bash 脚本中引用头段变量，需要借助于 OBSPY 宏的语法。

```
#!/bin/bash
OBSPY_DISPLAY_COPYRIGHT=0
```

```
3
4  Obspy << EOF
5  fg seis
6  ch kuser0 &1,kevnm&
7  setbb tmp ABC
8  ch kuser1 %tmp%
9  lh kuser0 kuser1
10 quit
11 EOF
```

7.2.3 内联函数

bash 可以完成基本的数学运算，但是所有的运算只支持整型数据，浮点型运算或者其它更高级 的数学运算需要借助 bc 或者 awk 来完成。Bash 中的变量以“$”作为标识符，Bash 会首先做变量替换再将替换后的命令传递给 OBSPY。

```
1  #!/bin/bash
2  OBSPY_DISPLAY_COPYRIGHT=0
3
4  declare -i var1 var2 var3 var4
5  var1=(1+2)*3
6  var2=10/4
7  var3=10/4
8  echo $var1 $var2 $var3
9
10 Obspy << EOF
11 echo on
12 fg seis
13 bp c $var2 $var1
14 q
15 EOF
```

本例中的变量“$var1”和“$var2”会首先被 OBSPY 解释成为 1 和 2，因而 OBSPY 实际接收到的 命令是“bp c 1 2 ”。

借助于 awk、sed 等工具，也可以实现部分字符串处理函数：

```
1  #!/bin/bash
2  OBSPY_DISPLAY_COPYRIGHT=0
3  str1=`echo "this is long" | sed 's/long/short/'`      # 替换
4  str2=`echo "abcdefghi" | sed 's/def//'`               # 删除
5
6  Obspy << EOF
7  fg seis
8  title "$str1"
9  p
10 saveimg string.ps
11 q
```

```
EOF
```

7.2.4　条件判断和循环控制

Bash 具有更灵活的条件判断和循环控制功能，但由于 Bash 自身的限制，这些特性仅能在 OBSPY
外部使用，因而下例中需要多次调用 OBSPY，在某些情况下会相当耗时。

```
#!/bin/sh
OBSPY_DISPLAY_COPYRIGHT=0

for file in *.OBSPY; do
    Obspy <<EOF
      read $file
      rmean
      rtrend
      lp co 1.0 p 2 n 4
      write  ${file}.filtered
      quit
EOF
done
```

7.2.5　文件重命名

Bash 下可以借助于 `awk` 来实现文件重命名。下面的例子中，首先用点号对文件名做分割，`$0` 表示原始文件名，`$7` 表示用逗号分割后的第 7 段字符，即台网名，其他同理。最后将 `awk` 的输出传给 `sh` 去执行。

```
ls *.OBSPY | awk -F. '{printf "mv %s %s.%s.%s.%s\n", $0, $7, $8, $9, $10}' |
sh
```

7.3　Perl 中调用 OBSPY

7.3.1　简单示例

下面的脚本展示了如何在 Perl 中调用 OBSPY。

```
#!/usr/bin/env perl
use strict;
use warnings;
$ENV{OBSPY_DISPLAY_COPYRIGHT}=0;

open(OBSPY, "| Obspy ") or die "Error opening Obspy\n";
print OBSPY "fg seismo \n";
print OBSPY "lh evla kstnm \n";
print OBSPY "q \n";
close(OBSPY);
```

Perl 中调用 OBSPY 本质上是使用 `open(OBSPY, "| Obspy")` 语句定义了一个名为 `OBSPY` 指向 `Obspy` 的 句柄，然后通过 `print OBSPY` 语句将要执行的 OBSPY 命令传递给 OBSPY。

7.3.2 数据转换

首先要将 SEED 格式的数据转换成 OBSPY 格式。

- 假定每个目录下有且只有一个 SEED 数据
- rdseed 一次只能处理一个 SEED 数据
- rdseed 的 -pdf 选项会提取出 OBSPY 波形数据和 PZ 格式的仪器响应文件

```
#!/usr/bin/env perl
use strict;
use warnings;

@ARGV == 1 or die "Usage: perl $0 dirname\n";
my ($dir) = @ARGV;

chdir $dir;   # cd 进数据所在目录，以避免处理路径

my @seed = glob "*.seed";
# rdseed 一次只能处理一个 SEED 文件
die "One and only one SEED file is needed!\n" if @seed != 1;
system "rdseed -pdf @seed";

chdir "..";
```

7.3.3 文件合并

SEED 文件的波形数据可能会因为多种原因而出现间断，导致同一个通道会解压出来多个 OBSPY 文件，因而需要将属于同一个通道的 OBSPY 数据合并起来。

- 此处使用了新版 merge 命令的语法，要求 OBSPY 版本大于 v101.6
- merge 命令还有更多选项可以控制数据合并的细节，见命令的语法介绍
- 合并后的数据，以最早的数据段的文件名命名
- 多余的数据段均被删除，以保证每个通道只有一个 OBSPY 文件
- 由于脚本运行速度比 OBSPY 运行速度快，因而应先退出 OBSPY 再删除多余的数据段

```
#!/usr/bin/env perl
use strict;
use warnings;
$ENV{OBSPY_DISPLAY_COPYRIGHT}=0;

@ARGV == 1 or die "Usage: perl $0 dirname\n";
my ($dir) = @ARGV;

chdir $dir;

# 利用 hash 的 key 的不可重复性构建集合：
#   hash 的 key 定义为 NET.STA.LOC.CHN
#   hash 的 value 是文件名与 key 匹配的 OBSPY 文件数目
my %sets;
```

```
foreach (glob "*.OBSPY") {
    # 将文件名用'.' 分割，并取其中的第 7 到 10 个元素
    my ($net, $sta, $loc, $chn) = (split /\./)[6..9];
    $sets{"$net.$sta.$loc.$chn"}++;
}

# 在循环体外部调用 OBSPY，若没有数据需要 merge，则此次调用是做无用工
# 若将 OBSPY 调用放在循环体内，则可能会多次调用 OBSPY，而影响运行效率
open(OBSPY, "|Obspy") or die "Error in opening Obspy\n";
print OBSPY "wild echo off \n";
my @to_del;
while (my ($key, $value) = each %sets) {
    # 检查每个 key 所对应的 value，只有 value 大于 1 时才需要 merge
    next if $value == 1;

    print STDERR "merge $key: $value traces\n";
    my @traces = sort glob "*.$key.?.OBSPY";
    # 对 glob 的返回值做 sort，以保证第一个元素是最早的数据段

    # 在 OBSPY 中使用通配符而不是使用 @traces 以避免命令行过长的问题
    # merge 不支持通配符
    print OBSPY "r *.$key.?.OBSPY \n";  # OBSPY v101.6 or later only
    print OBSPY "merge \n";
    print OBSPY "w $traces[0] \n";      # 以最早数据段的文件名保存

    # 将要删除的数据段文件名保存到数组中，待退出 OBSPY 后再删除
    push @to_del, splice(@traces, 1);
}
print OBSPY "q \n";
close(OBSPY);
unlink(@to_del);

chdir "..";
```

7.3.4　文件重命名

从 SEED 解压出的 OBSPY 文件名较长，因而对其重命名以简化。

- SEED 解压出的默认文件名格式为 `yyyy.ddd.hh.mm.ss.ffff.NN.SSSSS.LL.CCC.Q.OBSPY`
- 重命名后的文件名为 `NET.STA.LOC.CHN.OBSPY`

```
#!/usr/bin/env perl
use strict;
use warnings;

@ARGV == 1 or die "Usage: perl $0 dirname\n";
my ($dir) = @ARGV;
```

```
7
8  chdir $dir;
9
10 foreach my $file (glob "*.OBSPY") {
11     my ($net, $sta, $loc, $chn) = (split /\./, $file)[6..9];
12     rename $file, "$net.$sta.$loc.$chn.OBSPY";
13 }
14
15 chdir "..";
```

7.3.5　添加事件信息

若 SEED 中不包含事件信息，则解压得到的 OBSPY 文件中也不会包含事件信息。因而需要用户 手动添加事件的发震时刻、经纬度、深度和震级信息。

- 输入参数包括：目录名、发震时刻、经度、纬度、深度、震级
- 发震时刻的格式为 `yyyy-mm-ddThh:mm:ss.xxx`，其中 `T` 用于分隔日期和时间

```
1  #!/usr/bin/env perl
2  use strict;
3  use warnings;
4  use POSIX qw(strftime);
5
6  @ARGV == 6 or die
7      "Usage: perl $0 dirname yyyy-mm-ddThh:mm:ss.xxx evlo evla evdp mag\n";
8  my ($dir, $origin, $evlo, $evla, $evdp, $mag) = @ARGV;
9
10 # 对发震时刻做解析
11 my ($date, $time) = split "T", $origin;
12 my ($year, $month, $day) = split "-", $date;
13 my ($hour, $minute, $second) = split ":", $time;
14 # 秒和毫秒均为整数
15 my ($sec, $msec) = split /\./, $second;
16 $msec = int(($second - $sec) * 1000 + 0.5);
17
18 # 计算发震日期是一年中的第几天
19 my $jday = strftime("%j", $second, $minute, $hour, $day, $month-1,
   ↪  $year-1900);
20
21 chdir $dir;
22 open(OBSPY, "| Obspy") or die "Error in opening OBSPY\n";
23 print OBSPY "wild echo off \n";
24 print OBSPY "r *.OBSPY \n";
25 print OBSPY "synchronize \n"; # 同步所有文件的参考时刻
26 print OBSPY "ch o gmt $year $jday $hour $minute $sec $msec \n";
27 print OBSPY "ch allt (0 - &1,o&) iztype IO \n";
28 print OBSPY "ch evlo $evlo evla $evlo evdp $evdp mag $mag \n";
```

```
print OBSPY "wh \n";
print OBSPY "q \n";
close(OBSPY);

chdir "..";
```

7.3.6　去仪器响应

使用 PZ 文件去仪器响应。若数据的时间跨度太长，在该时间跨度内可能仪器响应会发生变化，因而会存在一个通道有多个 PZ 文件的情况。目前该脚本在遇到这种情况时会自动退出。

```
#!/usr/bin/env perl
use strict;
use warnings;
$ENV{OBSPY_DISPLAY_COPYRIGHT} = 0;

@ARGV == 1 or die "Usage: perl $0 dirname\n";
my ($dir) = @ARGV;

chdir $dir;

open(OBSPY, "| Obspy") or die "Error in opening Obspy\n";
foreach my $Obspyfile (glob "*.OBSPY") {
    my ($net, $sta, $loc, $chn) = split /\./, $Obspyfile;

    # 找到对应的 PZ 文件
    # PZ 文件名为: OBSPY_PZs_NET_STA_CHN_LOC_BeginTime_EndTime
    my @pz = glob "OBSPY_PZs_${net}_${sta}_${chn}_${loc}_*_*";
    die "PZ file error for $Obspyfile \n" if @pz != 1;

    print OBSPY "r $Obspyfile \n";
    print OBSPY "rmean; rtr; taper \n";
    print OBSPY "trans from pol s $pz[0] to none freq 0.01 0.02 5 10\n";
    print OBSPY "mul 1.0e9 \n";
    print OBSPY "w over\n";
}
print OBSPY "q\n";
close(OBSPY);

chdir "..";
```

7.3.7　分量旋转

将成对的水平分量旋转到大圆路径。

- 检查三分量是否缺失
- 检查 `delta` 是否相等
- 取三分量中的最大 `b` 和最小 `e` 值作为数据窗口，此操作要求三分量的 `kzdate` 和 `kztime` 必

须相同，这一点在添加事件信息时使用 `synchronize` 已经实现

- 未检查分量的 `cmpinc` 和 `cmpaz` 是否符合要求。若不符合要求，OBSPY 会报错且不会旋转，写 到磁盘中的 R 和 T 分量实际上是原数据，暂不知如何处理。

```perl
#!/usr/bin/env perl
use strict;
use warnings;
use List::Util qw(min max);

@ARGV == 1 or die "Usage: perl $0 dirname\n";
my ($dir) = @ARGV;

chdir $dir;

# 利用 hash 的 key 的不可重复性构建集合:
#   hash 的 key 定义为 NET.STA.LOC.CH (分量信息不是 BHZ 而是 BH)
#   hash 的 value 是文件名与 key 匹配的 OBSPY 文件数目，正常情况下是 3 的整数倍
my %sets;
foreach (glob "*.OBSPY") {
    my ($net, $sta, $loc, $chn) = split /\./;
    my $chn2 = substr $chn, 0, 2;
    $sets{"$net.$sta.$loc.$chn2"}++;
}

open(OBSPY, "|Obspy") or die "Error in opening Obspy\n";
# 对所有的 key 做循环
foreach my $key (keys %sets) {
    my ($E, $N, $Z, $R, $T, $Z0);

    # 检查 Z 分量是否存在
    $Z = "${key}Z.OBSPY";
    if (!-e $Z) {   # 若不存在，则删除该台站的所有数据
        warn "Vertical component missing!\n";
        unlink glob "$key?.OBSPY";
        next;
    }

    # 检查水平分量是否存在
    if (-e "${key}E.OBSPY" and -e "${key}N.OBSPY") {    # E 和 N 分量
        $E = "${key}E.OBSPY";
        $N = "${key}N.OBSPY";
    } elsif (-e "${key}1.OBSPY" and -e "${key}2.OBSPY") {      # 1 和 2 分量
        $E = "${key}1.OBSPY";
        $N = "${key}2.OBSPY";
    } else {    # 水平分量缺失
```

```
42          warn "Horizontal components missing!\n";
43          unlink glob "$key?.OBSPY";
44          next;
45      }
46
47      # 假定 kzdate 和 kztime 相同
48      # 检查 B, E, DELTA
49      my (undef, $Zb, $Ze, $Zdelta) = split " ", `Obspylst b e delta f $Z`;
50      my (undef, $Eb, $Ee, $Edelta) = split " ", `Obspylst b e delta f $E`;
51      my (undef, $Nb, $Ne, $Ndelta) = split " ", `Obspylst b e delta f $N`;
52      die "$key: delta not equal\n" if $Zdelta != $Edelta or $Zdelta !=
    ↪  $Ndelta;
53
54      # 获取三分量里的最大 B 和最小 E 值作为数据窗
55      my $begin = max($Zb, $Eb, $Nb);
56      my $end = min($Ze, $Ee, $Ne);
57
58      #输出文件名为 NET.STA.LOC.[RTZ]
59      my $prefix = substr $key, 0, length($key)-2;
60      $R = $prefix."R";
61      $T = $prefix."T";
62      $Z0 = $prefix."Z";
63
64      print OBSPY "cut $begin $end \n";
65      print OBSPY "r $E $N \n";
66      print OBSPY "rotate to gcp \n";
67      print OBSPY "w $R $T \n";
68      print OBSPY "r $Z \n";
69      print OBSPY "w $Z0 \n";
70  }
71  print OBSPY "q\n";
72  close(OBSPY);
73  unlink glob "*.OBSPY";
74
75  chdir "..";
```

7.3.8　数据重采样

通常有两种情况下需要对数据进行重采样：

- 原始数据的采样率过高，而实际研究中不需要如此高的采样率，此时，对数据做减采样可以大大减少数据量；
- 原始数据中，不同台站的采样率不同，可能会影响到后期的数据处理，因而需要让所有数据使用统一的采样率；

下面的 Perl 脚本中使用 `interpolate` 命令将所有数据重采样到相同的采样周期。用户可以在命令行中直接指定要使用的重采样后的采样周期，若命令行中的采样周期指定为　　0，则以大多数数据所

使用的采样周期作为重采样后的采样周期。

```perl
#!/usr/bin/env perl
use strict;
use warnings;

@ARGV == 2 or die "Usage: perl $0 dirname [delta | 0]\n";
# 若第二个参数为 0，则取数据中出现次数最多的周期为采样周期

my ($dir, $delta) = @ARGV;

chdir $dir;

if ($delta == 0) {
    # 假定所有数据已旋转到 RTZ 坐标系，文件名格式为 NET.STA.LOC.[RTZ]
    # hash 的键为出现了的采样周期，其值为出现的个数
    my %group;
    foreach (glob "*.*.*.[RTZ]") {
        my (undef, $delta0) = split /\s+/, `Obspylst delta f $_`;
        $group{$delta0}++;
    }
    # 将 hash 按 value 排序，找到最大 value 所对应的 key
    my @keys = sort {$group{$b} <=> $group{$a}} keys %group;
    ($delta) = @keys;
}

# 重采样
open(OBSPY, "|Obspy") or die "Error in opening Obspy\n";
foreach (glob "*.*.*.[RTZ]") {
    my (undef, $delta0) = split /\s+/, `Obspylst delta f $_`;
    next if $delta == $delta0;   # 不需要重采样

    print OBSPY "r $_ \n";
    #用 interpolate 实现减采样或增采样
    #若是减采样，则需要对数据做低通滤波以防止出现混淆效应
    # 低通滤波时或许需要加上 p 2 以避免滤波引起的相移
    printf OBSPY "lp c %f\n", 0.5/$delta if $delta > $delta0;
    print OBSPY "interpolate delta $delta \n";

    print OBSPY "w over\n";
}
print OBSPY "q\n";
close(OBSPY);

chdir "..";
```

7.4　Python 中调用 OBSPY

7.4.1　简单示例

下面的脚本展示了如何在 Python 中调用 OBSPY。

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import os
import subprocess

os.putenv("OBSPY_DISPLAY_COPYRIGHT", '0')

p = subprocess.Popen(['Obspy'], stdin=subprocess.PIPE)
s = "fg seismo \n"
s += "lh evla kstnm \n"
s += "q \n"
p.communicate(s.encode())
```

Python 中使用 `subprocess` 模块的 `Popen` 方法调用 OBSPY，通过 `p.communicate()` 将命令 `s.encode()` 传递给 OBSPY。

7.4.2　数据转换

首先要将 SEED 格式的数据转换成 OBSPY 格式。

- 假定每个目录下有且只有一个 SEED 数据
- `rdseed` 一次只能处理一个 SEED 数据
- `rdseed` 的 `-pdf` 选项会提取出 OBSPY 波形数据和 PZ 格式的仪器响应文件

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import os
import sys
import glob
import subprocess

os.putenv("OBSPY_DISPLAY_COPYRIGHT", '0')

if len(sys.argv) != 2:
    sys.exit("Usage: python %s dirname" % sys.argv[0])

dir = sys.argv[1]

os.chdir(dir)   # cd 进数据所在目录，以避免处理路径

# 假定当前目录只有一个 SEED
```

```
seed = glob.glob("*.seed")
if len(seed) != 1:
    sys.exit("One and only one SEED file is needed!")

subprocess.call(['rdseed', '-pdf', seed[0]])

os.chdir("..")
```

7.4.3 文件合并

SEED 文件的波形数据可能会因为多种原因而出现间断，导致同一个通道会解压出来多个OBSPY文件，因而需要将属于同一个通道的 OBSPY 数据合并起来。

- 此处使用了新版merge命令的语法，要求OBSPY版本大于v101.6
- merge 命令还有更多选项可以控制数据合并的细节，见命令的语法介绍
- 合并后的数据，以最早的数据段的文件名命名
- 多余的数据段均被删除，以保证每个通道只有一个 OBSPY 文件
- 由于脚本运行速度比 OBSPY 运行速度快，因而应先退出 OBSPY 再删除多余的数据段

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import os
import sys
import glob
import subprocess

os.putenv("OBSPY_DISPLAY_COPYRIGHT", '0')

if len(sys.argv) != 2:
    sys.exit("python %s dirname" % sys.argv[0])

dir = sys.argv[1]

os.chdir(dir)

# 利用 dict 的 key 的不可重复性构建集合:
#     dict 的 key 定义为 NET.STA.LOC.CHN
#     dict 的 value 是文件名与 key 匹配的 OBSPY 文件数目
sets = {}
for fname in glob.glob("*.OBSPY"):
    net, sta, loc, chn = fname.split('.')[6:10]
    key = '.'.join([net, sta, loc, chn])
    if key not in sets:
        sets[key] = 1
    else:
        sets[key] += 1
```

```
29
30 p = subprocess.Popen(['Obspy'], stdin=subprocess.PIPE)
31 s = "wild echo off \n"
32 to_del = []
33 for key, value in sets.items():
34     # 仅当 value 大于 1 时才需要 merge
35     if value == 1:
36         continue
37
38     print("merge %s: %d traces" % (key, value))
39     # Python 的 glob 返回值是乱序的，因而必须 sort
40     traces = sorted(glob.glob('.'.join(['*', key, '?', 'OBSPY'])))
41
42     # 在 OBSPY 中使用通配符而不是使用 @traces 以避免命令行过长的问题
43     # merge 不支持通配符
44     s += "r *.%s.?.OBSPY \n" % key   # OBSPY v101.6 or later
45     s += "merge \n"
46     s += "w %s \n" % traces[0]       # 以最早数据段的文件名保存
47
48     to_del.extend(traces[1:])
49
50 s += "q \n"
51 p.communicate(s.encode())
52
53 # 删除多余的数据段
54 for file in to_del:
55     os.unlink(file)
56
57 os.chdir("..")
```

7.4.4　文件重命名

从 SEED 解压出的 OBSPY 文件名较长，因而对其重命名以简化。

- SEED 解压出的默认文件名格式为 `yyyy.ddd.hh.mm.ss.ffff.NN.SSSSS.LL.CCC.Q.OBSPY`
- 重命名后的文件名为 `NET.STA.LOC.CHN.OBSPY`

```
1 #!/usr/bin/env python
2 # -*- coding: utf-8 -*-
3
4 import os
5 import sys
6 import glob
7
8 os.putenv("OBSPY_DISPLAY_COPYRIGHT", '0')
9
10 if len(sys.argv) != 2:
```

```
    sys.exit("Usage: python %s dirname\n" % sys.argv[0])

dir = sys.argv[1]

os.chdir(dir)
for fname in glob.glob("*.OBSPY"):
    net, sta, loc, chn = fname.split('.')[6:10]
    os.rename(fname, "%s.%s.%s.%s.OBSPY" % (net, sta, loc, chn))

os.chdir("..")
```

7.4.5　添加事件信息

若 SEED 中不包含事件信息，则解压得到的 OBSPY 文件中也不会包含事件信息。因而需要用户 手动添加事件的发震时刻、经纬度、深度和震级信息。

- 输入参数包括：目录名、发震时刻、经度、纬度、深度、震级
- 发震时刻的格式为 `yyyy-mm-ddThh:mm:ss.xxx`，其中 `T` 用于分隔日期和时间

```
#!/usr/bin/env python
# -*- coding: utf8 -*-

import os
import sys
import datetime
import subprocess

os.putenv("OBSPY_DISPLAY_COPYRIGHT", '0')

if len(sys.argv) != 7:
    sys.exit("Usage: python %s dirname yyyy-mm-ddThh:mm:ss.xxx evlo evla
    ↪  evdp mag" % sys.argv[0])

dir, origin, evlo, evla, evdp, mag = sys.argv[1:]

o = datetime.datetime.strptime(origin,'%Y-%m-%dT%H:%M:%S.%f')
# 计算发震日期是一年中的第几天
jday = o.strftime("%j")
# 提取毫秒值
msec = int(o.microsecond / 1000 + 0.5)

os.chdir(dir)

p = subprocess.Popen(['Obspy'], stdin=subprocess.PIPE)
s = "wild echo off \n";
s += "r *.OBSPY \n"
s += "synchronize \n"
```

```
s += "ch o gmt %s %s %s %s %s %d \n" % (o.year, jday, o.hour, o.minute,
    ↪  o.second, msec)
s += "ch allt (0 - &1,o&) iztype IO \n"
s += "ch evlo %s evla %s evdp %s mag %s \n" % (evlo, evla, evdp, mag)
s += "wh \n"
s += "q \n"
p.communicate(s.encode())

os.chdir("..")
```

7.4.6　去仪器响应

使用 PZ 文件去仪器响应。若数据的时间跨度太长，在该时间跨度内可能仪器响应会发生变化，因而会存在一个通道有多个 PZ 文件的情况。目前该脚本在遇到这种情况时会自动退出。

```
#!/usr/bin/env python
# -*- coding: utf8 -*-
import os
import sys
import glob
import subprocess

os.putenv("OBSPY_DISPLAY_COPYRIGHT", "0")

if len(sys.argv) != 2:
    sys.exit("Usage: python %s dirname\n" % sys.argv[0])

dir = sys.argv[1]

os.chdir(dir)

p = subprocess.Popen(['Obspy'], stdin=subprocess.PIPE)

s = ""
for Obspyfile in glob.glob("*.OBSPY"):
    net, sta, loc, chn = Obspyfile.split('.')[0:4]
    pz = glob.glob("OBSPY_PZs_%s_%s_%s_%s_*_*" % (net, sta, chn, loc))
    # 暂不考虑多个 PZ 文件的情况
    if len(pz) != 1:
        sys.exit("PZ file error for %s" % Obspyfile)

    s += "r %s \n" % Obspyfile
    s += "rmean; rtr; taper \n"
    s += "trans from pol s %s to none freq 0.01 0.02 5 10\n" % pz[0]
    s += "mul 1.0e9 \n"
    s += "w over \n"
```

```
32
33 s += "q \n"
34 p.communicate(s.encode())
35
36 os.chdir("..")
```

7.4.7 分量旋转

将成对的水平分量旋转到大圆路径。

- 检查三分量是否缺失
- 检查 delta 是否相等
- 取三分量中的最大 b 和最小 e 值作为数据窗口，此操作要求三分量的 kzdate 和 kztime 必须相同，这一点在添加事件信息时使用 synchronize 已经实现
- 未检查分量的 cmpinc 和 cmpaz 是否符合要求。若不符合要求，OBSPY 会报错且不会旋转，写 到磁盘中的 R 和 T 分量实际上是原数据，暂不知如何处理。

```
1 #!/usr/bin/env python
2 # -*- coding: utf8 -*-
3
4 import os
5 import sys
6 import glob
7 import subprocess
8
9 os.putenv("OBSPY_DISPLAY_COPYRIGHT", '0')
10
11 if len(sys.argv) != 2:
12     sys.exit("Usage: python %s dirname" % sys.argv[0])
13
14 dir = sys.argv[1]
15 os.chdir(dir)
16
17 # 构建 NET.STA.LOC.CH 的集合
18 sets = set()
19 for fname in glob.glob("*.OBSPY"):
20     net, sta, loc, chn = fname.split('.')[0:4]
21     key = '.'.join([net, sta, loc, chn[0:2]])
22     sets.add(key)
23
24 p = subprocess.Popen(['Obspy'], stdin=subprocess.PIPE)
25 s = ""
26 for key in sets:
27     # 检测 Z 分量是否存在
28     Z = key + "Z.OBSPY"
29     if not os.path.exists(Z):
30         print("%s: Vertical component missing!\n" % key)
```

```
31          for fname in glob.glob(key + "?.OBSPY"):
32              os.unlink(fname)
33          continue
34
35      # 检测水平分量是否存在
36      if os.path.exists(key + "E.OBSPY") and os.path.exists(key + "N.OBSPY"):
37          E = key + "E.OBSPY"
38          N = key + "N.OBSPY"
39      elif os.path.exists(key + "1.OBSPY") and os.path.exists(key + "2.OBSPY"):
40          E = key + "1.OBSPY"
41          N = key + "2.OBSPY"
42      else:
43          print("%s: Horizontal components missings!\n" % key)
44          for fname in glob.glob(key + "?.OBSPY"):
45              os.unlink(fname)
46          continue
47
48      # 检查 B, E, DELTA
49      Zb, Ze, Zdelta = subprocess.check_output(['Obspylst', 'b', 'e', 'delta',
    ↪  'f',  Z]).decode().split()[1:]
50      Eb, Ee, Edelta = subprocess.check_output(['Obspylst', 'b', 'e', 'delta',
    ↪  'f',  E]).decode().split()[1:]
51      Nb, Ne, Ndelta = subprocess.check_output(['Obspylst', 'b', 'e', 'delta',
    ↪  'f',  N]).decode().split()[1:]
52
53      # 获取三分量的最大 B 和最小 E 值作为数据窗
54      begin = max(float(Zb), float(Eb), float(Nb))
55      end = min(float(Ze), float(Ee), float(Ne))
56
57      #输出文件名为 NET.STA.LOC.[RTZ]
58      prefix = key[:-2]
59      R, T, Z0 = prefix + '.R', prefix + '.T', prefix + '.Z'
60
61      s += "cut %f %f \n" % (begin, end)
62      s += "r %s %s \n" % (E, N)
63      s += "rotate to gcp \n"
64      s += "w %s %s \n" % (R, T)
65      s += "r %s \n" % Z
66      s += "w %s \n" % Z0
67  s += "q \n"
68  p.communicate(s.encode())
69
70  # 删除原文件
71  for fname in glob.glob("*.OBSPY"):
72      os.unlink(fname)
```

```
73
74 os.chdir("..")
```

7.4.8 数据重采样

通常有两种情况下需要对数据进行重采样：

- 原始数据的采样率过高，而实际研究中不需要如此高的采样率，此时，对数据做减采样可以大 大减少数据量；
- 原始数据中，不同台站的采样率不同，可能会影响到后期的数据处理，因而需要让所有数据使 用统一的采样率；

下面的 Python 脚本中使用 `interpolate` 命令将所有数据重采样到相同的采样周期。用户可以在命令行中直接指定要使用的重采样后的采样周期，若命令行中的采样周期指定为 0，则以大多数数 据所使用的采样周期作为重采样后的采样周期。

```
1 #!/usr/bin/env python
2 # -*- coding: utf-8 -*-
3
4 import os
5 import sys
6 import glob
7 import subprocess
8
9 os.putenv("OBSPY_DISPLAY_COPYRIGHT", "0")
10
11 if len(sys.argv) != 3:
12     sys.exit("Usage: python %s dirname [delta | 0]" % sys.argv[0])
13 # 若第二个参数为 0，则取数据中出现次数最多的周期为采样周期
14
15 dir, delta = sys.argv[1], float(sys.argv[2])
16
17 os.chdir(dir)
18
19 if delta == 0:
20     # 假定所有数据已旋转到 RTZ 坐标系，文件名格式为 NET.STA.LOC.[RTZ]
21     # hash 的键为出现了的采样周期，其值为出现的个数
22     groups = {}
23     for fname in glob.glob("*.*.*.[RTZ]"):
24         delta0 = subprocess.check_output(['Obspylst', 'delta', 'f',
    ↪  fname]).decode().split()[1]
25         if delta0 not in groups:
26             groups[delta0] = 1
27         else:
28             groups[delta0] += 1
29     delta = float(max(groups, key=groups.get))
30
31 p = subprocess.Popen(['Obspy'], stdin=subprocess.PIPE)
```

```
s = ""
for fname in glob.glob("*.*.*.[RTZ]"):
    delta0 = float(subprocess.check_output(['Obspylst', 'delta', 'f',
    ↪ fname]).decode().split()[1])
    if delta == delta0:  # 不需要重采样
        continue

    s += "r %s \n" % fname
    #用 interpolate 实现减采样或增采样
    #若是减采样，则需要对数据做低通滤波以防止出现混淆效应
    # 低通滤波时或许需要加上 p 2 以避免滤波引起的相移
    if delta > delta0:
        s += "lp c %f \n" % 0.5/delta
    s += "interpolate delta %f \n" % delta
    s += "w over \n"

s += "q \n"
p.communicate(s.encode())

os.chdir("..")
```

保护环境，从阅读电子文档开始！

第 8 章　使用 OBSPY 函数库

8.1　OBSPY 库简介

OBSPY 提供了两个函数库：libObspyio.a 和 libObspy.a，用户可以在自己的 C 或 Fortran 程序中 直接使用函数库中的子函数。这些库文件位于 ${OBSPYHOME}/lib 中。

8.1.1　libObspyio 库

此库文件包含的子函数可用于读写 OBSPY 数据文件、OBSPY 头段变量、黑板变量文件。这些子函 数可以在用户的 C 或 Fortran 程序中直接使用。

libObspyio.a 中可用的子函数包括：

表 8.1: libObspyio 子函数

子函数	说明
rObspy1	读取等间隔文件
rObspy2	读取不等间隔文件和谱文件
wObspy1	写入等间隔文件
wObspy2	写入不等间隔文件
wObspy0	可以写等间隔文件或不等间隔文件
getfhv	获取浮点型头段变量值
setfhv	设置浮点型头段变量值
getihv	获取枚举型头段变量值
setihv	设置枚举型头段变量值
getkhv	获取字符串头段变量值
setkhv	设置字符串头段变量值
getlhv	获取逻辑型头段变量值
setlhv	设置逻辑型头段变量值
getnhv	获取整型头段变量值
setnhv	设置整型头段变量值
readbbf	读取一个黑板变量文件
writebbf	写一个黑板变量文件
getbbv	获取一个黑板变量的值
setbbv	给一个黑板变量赋值
distaz	计算地球上任意两点间的震中距、方位角和反方位角

对于 C 源码，用如下命令编译

```
$ gcc -c source.c -I/usr/local/Obspy/include
$ gcc -o prog source.o -lm -L/usr/local/Obspy/lib -lObspyio
```

也可以利用OBSPY提供的`Obspy-config`命令简化此编译命令：

```
$ gcc -c source.c `Obspy-config -c`
$ gcc -o prog source.o -lm `Obspy-config -l Obspyio`
```

对于 Fortran77 源码，用如下命令编译

```
$ gfortran -c source.f
$ gfortran -o prog source.o -L/usr/local/Obspy/lib/ -lObspyio
```

也可以利用OBSPY提供的`Obspy-config`命令简化此编译命令：

```
$ gfortran -c souce.f
$ gfortran -o prog source.o `Obspy-config -l Obspyio`
```

8.1.2 libObspy.a 库

这个库是从 101.2 版本才引入的，其是 `libObspyio.a` 的超集，包含了几个数据处理常用的子函数。

`libObspy.a` 包含如下子函数：

- `xapiir` 无限脉冲响应滤波器；
- `firtrn` 有限脉冲滤波器，Hilbert 变换；
- `crscor` 互相关；
- `next2` 返回比输入值大的最小的 2 的幂次；
- `envelope` 计算包络函数; 对

于 C 源码，用如下命令编译

```
$ gcc -c source.c -I/usr/local/Obspy/include
$ gcc -o prog source.o -lm -L/usr/local/Obspy/lib -lObspy
```

也可以利用OBSPY提供的`Obspy-config`命令简化此编译命令：

```
$ gcc -c source.c `Obspy-config -c`
$ gcc -o prog source.o -lm `Obspy-config -l Obspy`
```

对于 Fortran77 源码，用如下命令编译

```
$ gfortran -c source.f
$ gfortran -o prog source.o -L/usr/local/Obspy/lib/ -lObspy
```

也可以利用OBSPY提供的`Obspy-config`命令简化此编译命令：

```
$ gfortran -c souce.f
$ gfortran -o prog source.o `Obspy-config -l Obspy`
```

8.2　调用 libObspyio 库

8.2.1　rObspy1

子函数 rObspy1 用于读取等采样间隔的 OBSPY 数据。 函数定义如下：

```
void
rObspy1(        char        *kname,      // 要读入的文件名
        float   *yarray,     // 数据被保存到 yarray 数组中
        int     *nlen,       // 数据长度
        float   *beg,        // 数据开始时间，即头段变量 b
        float   *del,        //数据采样周期，即头段变量 delta
        int     *max_,       // 数组 yarray 的最大长度，若 nlen>max_ 则截断
        int     *nerr,       // 错误标记，0 代表成功，非零代表失败
        int      kname_s     // 数组 kname 的长度
)
```

相关示例代码为 rObspy1c.c [1] 和 rObspy1f.f。

8.2.2　rObspy2

子函数 rObspy2 用于读取非等间隔采样的。

```
void
rObspy2(        char        *kname,      // 要读入的文件名
        float   *yarray,     // 因变量数组
        int     *nlen,       // 数据长度
        float   *xarray,     // 自变量数组
        int     *max_,       // 数组最大长度
        int     *nerr,       // 错误标记
        int      kname_s     // 数组 kname 的长度
)
```

相关示例代码为 rObspy2c.c 和 rObspy2f.f。

8.2.3　wObspy1

子函数 wObspy1 用于写等间隔 OBSPY 文件。

```
void
wObspy1(      char          *kname,      // 要写入的文件名
        float *yarray,      // 要写入文件的数组
        int   *nlen,        // 数组长度
        float *beg,         // 数据起始时刻
        float *del,         // 数据采样周期
        int   *nerr,        // 错误标记
        int    kname_s      // 文件名长度
)
```

[1] 代码位于 Obspy/doc/examples，下同。

相关示例代码为 wObspy1c.c 和 wObspy1f.f。

8.2.4 wObspy2

写非等间隔 OBSPY 文件。

```
void
wObspy2(      char          *kname,// 文件名
        float *yarray,      //因变量数组
        int   *nlen,        //数组长度
        float *xarray,      //自变量数组
        int   *nerr,        //错误码
        int    kname_s      //文件名长度
)
```

相关示例代码为 wObspy2c.c 和 wObspy2f.f。

8.2.5 wObspy0

子函数 wObspy0 相对来说更加通用也更复杂，利用该函数可以创建包含更多头段的 OBSPY 文件。

```
void
wObspy0(      char          *kname,// 文件名
        float *xarray,      //自变量数组
        float *yarray,      //因变量数组
        int   *nerr,        //错误码
        int    kname_s      //文件名长度
)
```

要使用子函数 wObspy0，首先要调用子函数 newhdr 创建一个完全未定义的头段区，并利用其 它头段变量相关子函数设置头段变量的值，并由 wObspy0 写入到文件中。必须要赋值的头段变量为 delta、b、e、npts、iftype。

相关示例代码为 wObspync.c 、wObspynf.f。n=3、4、5。

8.2.6 getfhv

获取浮点型头段变量的值。

```
void
getfhv(char   *kname,       //头段变量名
        float *fvalue,      //浮点型头段变量的值
        int   *nerr,        //错误码
        int    kname_s      //变量名长度
)
```

相关示例代码为 gethvc.c 和 gethvf.f。 至于如何获取和设置其它类型的头段变量，方法类似，不再多说。

8.2.7 readbbf

读取一个黑板变量文件。

```
void
readbbf( char    *kname,      // 要读取的文件
         int     *nerr,       // 错误码
         int     kname_s      // 文件名长度
)
```

writebbf 与之类似，不再列出。

8.2.8 getbbv

获取一个黑板变量的值。

```
void
getbbv( char     *kname,      // 黑板变量名
        char     *kvalue,     // 黑板变量的值
        int      *nerr,       // 错误码
        int      kname_s,     // 变量名长度
        int      kvalue_s     // 变量值的长度
)
```

setbbv 的函数定义与其类似，不再列出。

8.2.9 distaz

计算地球上任意两点之间的震中距、方位角和反方位角。

```
void
distaz( double   evla,      // 事件纬度
        double   evlo,      // 事件经度
        float    *stla,     // 台站纬度
        float    *stlo,     // 台站经度
        int      nsta,      // 台站数目
        float    *dist,     // 震中距 (km)
        float    *az,       // 方位角
        float    *baz,      // 反方位角
        float    *deg,      // 震中距 (度)
        int      *nerr      // 错误码
)
```

示例代码如下：

```
#include <stdio.h>
#include <stdlib.h>

void distaz(double, double, float*, float*, int,
        float*, float*, float*, float*, int*);

int main() {
    float evla, evlo, stla, stlo;
    float dist, az, baz, gcarc;
```

```
    int nsta, nerr;

    evla = 20;
    evlo = 35;
    stla = 59;
    stlo = 45;
    nsta = 1;

    distaz(evla, evlo, &stla, &stlo, nsta, &dist, &az, &baz, &gcarc, &nerr);
    fprintf(stderr, "%f %f %f %f\n", az, baz, gcarc, dist);

    return 0;
}
```

本例中，台站数目 `nsta=1` 。实际上该函数可以计算任意数目的台站到事件的震中距、方位角信息，若台站数目 `nsta` 不为 1，则 `stla`、`stlo`、`dist`、`az`、`baz`、`deg` 均可用数组或指针表示。

8.3 调用 libObspy 库

8.3.1 next2

OBSPY 在做 FFT 时要保证数据点数为 2^n 个，对于不足 2^n 个点的数据需要补零至 2^n 次方个 点。

next2 函数定义为：

```
int next2(int num)   // 输入为 num，返回值为大于 num 的最小 2 次幂
```

8.3.2 xapiir

`xapiir` 用于设计 IIR 滤波器，并对数据进行滤波。这个子函数底层调用了 `design` 和 `apply` 两个子函数。

```
void
xapiir(float    *data,      //  待滤波的数据，滤波后的数据保存在该数组中
       int       nsamps,    //数据点数
       char     *aproto,    //滤波器类型
                            //  - 'BU'  :   butterworth
                            //  - 'BE'  :   bessel
                            //  - 'C1'  :   chebyshev type I
                            //  - 'C2'  :   chebyshev type II
       double    trbndw,    // chebyshev 滤波器的过渡带宽度
       double    a,         // chebyshev 滤波器的衰减因子
       int       iord,      //滤波器阶数
       char     *type,      // 滤波类型，'LP','HP','BP','BR'
       double    flo,       //低频截断频率
       double    fhi,       //高频截断频率
       double    ts,        //采样周期
```

```
16          int         passes      // 通道数，1 或 2
17 )
```

相关示例代码为 filterc.c 和 filterf.f 。

8.3.3 firtrn

```
1  void
2  firtrn( char     *ftype,      // 类型，取'HILBERT' 或'DERIVATIVE'
3          float    *x,          // 输入数据
4          int       n,          // 数据点数，至少含 201 个点
5          float    *buffer,     //临时数组，长度至少 4297
6          float    *y           // 输出数组
7  )
```

参考示例:

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #define NPTS 1000
4  float *hilbert(float *in, int npts);
5  int main(){
6      float data[NPTS];
7      float *hdata;
8      int i;
9      //准备输入数据
10     for (i=0; i<NPTS; i++)   data[i] = i;
11     //进行 Hilbert 变换，hdata 为 Hilbert 变换的结果
12     hdata = hilbert(data, NPTS);
13     for (i = 0; i < NPTS; i++)
14         printf("%f\n", hdata[i]);
15
16     free(hdata);
17 }
18
19 // 这里定义了 hilbert 函数，是对 firtrn 函数的一个封装
20 float *hilbert(float *in, int npts) {
21     float *buffer;
22     float *out;
23
24     buffer = (float *)malloc(sizeof(float)*50000);
25     out = (float *)malloc(sizeof(float)*npts);
26     firtrn("HILBERT", in, npts, buffer, out);
27
28     free(buffer);
29     return out;
30 }
```

8.3.4 envelope

该子函数用于计算数据的包络函数，其底层调用了 firtrn 函数。

```
void
envelope(int    n,      //数据点数
         float *in,     //输入数据
         float *out     //输出数据
)
```

相关示例代码为envelopec.c和envelopef.f。

8.3.5 crscor

该子函数用于计算两个数据的互相关，此互相关在频率域中完成，相对时间域互相关而言效率 更高。

```
void
crscor( float   *data1,     // 数据 1
        float   *data2,     // 数据 2
        int      nsamps,    // 数据点数
        int      nwin,      // 相关窗数目
        int      wlen,      // 窗内数据点数，最大值为 2048
        char    *type,      // 窗类型，可以取'HAM','HAN','C','R','T'
        float   *c,         // 输出数据，长度为 2*wlen-1
        int     *nfft,      // 相关序列的数据点数
        char    *err,       // 错误消息
        int      err_s      // 错误消息长度
)
```

示例代码为 convolvec.c、convolvef.f、correlatec.c 和 correlatef.f。

第 9 章　OBSPY I/O

OBSPY 提供的命令可以帮助用户实现地震数据的预处理，但无法实现所有的数据分析功能。日常 科研中会需要自己写一套算法对数据进行处理，以得到想要的结果。这就需要能够在自己的程序中 读写 OBSPY 文件，即 OBSPY I/O。

根据 OBSPY 格式的定义，OBSPY 格式文件分为固定长度的头段区和非固定长度的数据区，通常 的做法是先读入头段区，然后从中提取出数据点数等信息，然后再据此读入数据区，由此即可实现 OBSPY 数据的读写。

这一章将介绍如何在 C、Fortran、Matlab、Python 中读写 OBSPY 文件。

9.1　C 程序中的 OBSPY I/O

OBSPY 自带的函数库中提供了一系列用于读写 OBSPY 文件的子函数，具体细节可以参考“调用 libObspyio 库”一节，这些子函数可以直接在 C 程序中调用。但这些子函数用起来不太方便，比如：

- 函数参数太多太复杂，有些参数基本不会用到，但还是需要定义；
- 读文件时无法只读取部分文件，即没有截窗的功能；
- 要获取某个头段变量的值，必须单独调用相应的子函数；

在了解了 OBSPY 文件的具体格式后，可以很容易的写一套函数来实现 OBSPY 文件的读写。

Prof. Lupei Zhu 实现了一套相对比较易用的 OBSPY I/O 函数库，可以在 C 或 Fortran 程序中 直接调用，姑且称之为 `Obspyio`。

`Obspyio` 函数库与 Prof. Lupei Zhu 的其他程序一起发布。你可以从 Prof. Lupei Zhu 的主页下 载 `fk` [1] 软件包，并从中提取出源文件 `Obspy.h` 和 `Obspyio.c`。

`Obspyio` 简单易用，但也存在一些潜在的 Bug 及缺陷。seisman 在 `Obspyio` 的基础上重写了 OBSPY I/O 函数库以及 OBSPY 相关工具，项目地址为 https://github.com/seisman/Obspy_tools。该项目 目前属于玩具性质，仅供有兴趣的读者参考。

9.1.1　Obspyio 函数接口

`Obspy.h` 中 OBSPY 格式的头段区被定义为 `OBSPYHEAD` 类型的结构体，每一个头段变量都是结构体的 成员。`Obspyio.c` 定义了一系列用于读写 OBSPY 文件的函数，表 9.1 中列出了 `Obspyio` 提供的函数接口。

[1] http://www.eas.slu.edu/People/LZhu/downloads/fk3.2.tar

表 9.1: Obspyio 函数列表

函数名	功能
read_Obspyhead	仅读取 OBSPY 文件的头段部分
read_Obspy	读取整个 OBSPY 文件
read_Obspy2	读取 OBSPY 文件中一部分
write_Obspy	将数据写到 OBSPY 文件中
Obspyhdr	初始化 OBSPY 头段

调用 OBSPY I/O 接口的程序，可以通过如下方式编译：

```
$ gcc -c Obspyio.c
$ gcc -c prog.c
$ gcc Obspyio.o prog.o -o prog -lm
```

写成 Makfile 会更简单一些：

```
all: prog clean

prog: Obspyio.o prog.o
	$(CC) -o $@ $^ -lm

clean:
	-rm *.o
```

9.1.2 读写 OBSPY 文件

下面的示例展示了如何在 C 程序中读取一个 OBSPY 二进制文件，经过简单的数据处理后，最后 写回到原文件中：

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "Obspy.h"

int main (int argc, char *argv[]) {
    char    file[80];
    OBSPYHEAD hd;
    float   *data;
    int     i;

    strcpy(file,  "seis.OBSPY");

    // 读入数据
    data = read_Obspy(file, &hd);
    fprintf(stderr, "npts=%d delta=%f \n", hd.npts, hd.delta);

```

```
18      // 其它数据处理
19      for (i=0; i<hd.npts; i++) {
20          data[i] *= 0.1;
21      }
22
23      // 将结果写回到原文件中
24      write_Obspy(file, hd,
25
26      free(data);
27      return 0;
28  }
```

read_Obspy 函数将 OBSPY 文件的头段区保存到结构体 OBSPYHEAD hd 中，可以通过 hd.npts、 hd.delta 这样的方式引用 OBSPY 头段变量的值。OBSPY 文件的数据区保存到指针/数组 float *data 中，read_Obspy 会根据头段中的数据点数为指针 data 分配内存空间并读入数据，在用完之后要记得 用 free(data) 释放已分配的内存，以避免内存泄露。

9.1.3　仅读取 OBSPY 头段区

有时候只需要 OBSPY 文件头段区的一些信息，若读取整个文件就有些浪费了，read_Obspyhead 仅 读取 OBSPY 头段区而不读取数据区。

```
1  #include <stdio.h>
2  #include "Obspy.h"
3
4  int main() {
5      OBSPYHEAD hd;
6
7      read_Obspyhead("seis.OBSPY",  &hd);
8      fprintf(stderr, "%d %f %f %f\n", hd.npts, hd.delta, hd.depmax,
   ↪  hd.depmin);
9
10     return 0;
11 }
```

9.1.4　读 OBSPY 数据的一段

有些时候，数据可能很长，但用户只需要其中的一小段。为了读取一小段数据而把整个文件都 读入进去实在太浪费了。OBSPY 中的 cut 命令可以实现数据的截取，read_Obspy2 则是实现了类似的 功能。下面的例子截取了数据中以 T0 为参考的 -0.5 到 2.5 秒，即相当于 cut t0 -0.5 2.5。

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include "Obspy.h"
5
6  int main (int argc, char *argv[]) {
7      char fin[80], fout[80];
8      OBSPYHEAD hd;
```

```
    float   *data;
    int     tmark;
    float   t1, t2;
    int     i;

    strcpy(fin,  "seis.OBSPY");

    tmark   =   0;
    t1      =   -0.5;
    t2      =   2.5;

    data = read_Obspy2(fin, &hd, tmark, t1, t2);
    fprintf(stderr, "npts=%d delta=%f\n", hd.npts, hd.delta);

    for (i=0; i<hd.npts; i++) {
        data[i] += 0.1;
    }

    strcpy(fout,  "seis.OBSPY.cut");
    write_Obspy(fout, hd, data);

    free(data);
    return 0;
}
```

read_Obspy2 与 read_Obspy 相比，多了三个用于定义时间窗的参数，其中 tmark 表示参考时间标 记，可以取值为：

-5 以 B 作为时间标记

-3 以 O 作为时间标记

-2 以 A 作为时间标记

0–9 以 T0-T9 中的某个为时间标记

这里只读入了数据的一小部分数据，但子程序中对头段中的 b、e、npts 等做了修改，因而返回的头 段与数据是相互兼容的。

Note: *tmark* 的有效取值为什么是*-5*、*-3*、*-2*、*0-9*?

在头段区中，若以 *T0* 作为参考位，*T3* 是 *T0* 后的第 *3* 个变量，*T7* 是 *T0* 后的第 *7* 个变量，*B* 是 *T0* 前的第 *5* 个变量，*O* 是 *T0* 前的第 *3* 个变量。

Obspyio.c 的代码中使用了 tref = *((float *)hd + 10 + tmark) 来获取某个时间头段变量 的值。首先将结构体指针 *hd* 强制转换成 *float* 型，加上 *10* 使得指针指向结构体的成员 *T0* 所 在的位置，然后 *tmark* 取不同的值，则将指针相应地前移或后移相应的浮点变量，最后取指针所在 位置的浮点变量值。由此即可很简单地获取某个时间标记头段中的值。这种用法在自己的程序中也 会经常用到。

9.1.5　从零创建 OBSPY 文件

在做合成数据时，经常需要从无到有完全创建一个 OBSPY 文件。这相对于一般的“读 → 处理 → 写”而言要更复杂一些，因为必须首先构建一个基本的头段区。下面的例子展示了如何用 Obspyhdr 构建一个最基本的头段区，并填充其他一些头段，最后将创建的头段及数据写入到文件中。

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "Obspy.h"

int main (int argc, char *argv[]) {
    char    fout[80];
    OBSPYHEAD hd;
    float   *data;

    float   delta;
    int     npts;
    float   b;

    int     i;

    delta = 0.01;       // 采样周期
    npts  = 1000;       // 数据点数
    b     = 10;         // 文件开始时间
    hd = Obspyhdr(delta, npts, b);  // 构建基本头段
    hd.dist     = 10;   // 给其它头段变量赋值
    hd.cmpaz    = 0.0;
    hd.cmpinc   = 0.0;

    strcpy(fout, "seis.syn");
    // 生成合成数据
    data = (float *)malloc(sizeof(float)*npts);
    for (i=0; i<npts; i++) {
        data[i] = i;
    }

    // 写入到文件中
    write_Obspy(fout, hd, data);
    free(data);

    return 0;
}
```

9.2 Fortran 程序中的 OBSPY I/O

王亮和 seisman 用 Fortran 90 写了一个读写 OBSPY 文件的模块，以开源许可 Apache 2.0 在 github 上发布。项目地址为：https://github.com/PeterPanwl/Obspyio_Fortran。项目中提供了名为 Obspyio 的模块，该模块中包含了 6 个子程序，所有子程序的实现都写在源文件 Obspyio.f90 中。项目中包含了详细的使用说明及演示示例，此处仅简单列出每个子程序的功能：

表 9.2: Obspyio 模块中的子程序

子程序	功能
Obspyio_readhead	仅读取 OBSPY 文件的头段部分
Obspyio_readObspy	读取整个 OBSPY 文件
Obspyio_writeObspy	将数据写到 OBSPY 文件中
Obspyio_readObspy_cut	读取 OBSPY 文件中一部分
Obspyio_nullhead	获得一个未定义状态的 OBSPY 头段
Obspyio_newhead	初始化 OBSPY 头段

9.3 Matlab 脚本中的 OBSPY I/O

OBSPY 官方自带了几个可以读写 OBSPY 文件的 Matlab 脚本，位于 ${OBSPYHOME}/utils 目录下。

9.3.1 readObspy

readObspy.m 中定义了函数 readObspy，可以用于读取 OBSPY 文件，其参数为要读取的 OBSPY 文件名。文件名可以是：

1. 单个文件名
2. 多个文件名组成的字符串数组
3. 含通配符的字符串或字符串数组
4. 空字符串，表示读取当前目录下的全部 OBSPY 文件

readObspy 函数有三种用法。第一种用法：

```
% 读取 OBSPY 文件，并保存到结构体 S 中
>> S = readObspy('seis.OBSPY');
% 通过 S.varname 形式引用结构体成员的值
>> S.NPTS
ans = 1000
>> S.DELTA
ans = 0.0100
% 数据保存到 S.DATA1 数组中
>> S.DATA1(10)
ans = -0.0934
```

第二种用法：

```
% 读取 OBSPY 文件，时间和波形振幅分别保存到数组 X 和 Y 中
>> [X, Y] = readObspy('seis.OBSPY');
```

```
% 若发震时刻 O 未定义，则 X 数组中保存相对于 B 值的时间
% 若发震时刻 O 有定义，则 X 数组中保存相对于 O 值的时间
>> X(1)
ans = 50.8900
```

第三种用法，有三个返回值：

```
% 用于读取不等间隔数据或 OBSPY 谱数据
>> [X, Y1, Y2] = readObspy('seis.OBSPY');
% 用于读取 XYZ 类型的 OBSPY 文件
>> [X, Y, Z] = readObspy("seis.OBSPY.xyz");
```

9.3.2 getObspydata

在用 readObspy 读取 OBSPY 文件时，可以直接读入到结构体 S 中，也可以读入到多个数组中。读 入到数组中便于 matlab 处理，但是却丢失了 OBSPY 头段中的很多信息；读入到结构体 S 中，有时却 需要用数组做处理，这就可以使用 getObspydata 函数，可以从结构体 S 中提取出自变量和因变量数 组：

```
% 从文件中读入结构体 S
>> S = readObspy('seis.OBSPY');
% 从结构体 S 中提取自变量和因变量
>> [X, Y] = getObspydata(S);
```

9.3.3 writeObspy

writeObspy.m 中定义了函数 writeObspy，用于写 OBSPY 格式的文件，其输入是结构体 S，返回值是 状态码：

```
% 读入 OBSPY 数据
>> S = readObspy('seis.OBSPY');
% 做一些数据处理
% ...
% 修改结构体 S 中的文件名
>> S.FILENAME = 'new.OBSPY';
% 写入到新文件中
>> status = writeObspy(S);
```

9.3.4 其他

除了 OBSPY 官方提供的几个 Matlab 脚本之外，还有其他人也提供了一些 Matlab 脚本：

- http://geophysics.eas.gatech.edu/people/zpeng/Teaching/MatOBSPY.tar.gz
- http://web.utah.edu/thorne/software.html

9.4 Python 脚本中的 OBSPY I/O

ObsPy [1] 是用 Python 写的一个专用用于地震学数据处理的模块，这一节介绍如何利用 ObsPy 模块读写 OBSPY 文件。

[1] http://www.obspy.org

9.4.1 安装 obspy

obspy 可以通过 Python 自带的模块管理工具 pip 来安装，目前 obspy 的最新版本为 0.10.2：

```
$ pip install obspy
```

9.4.2 读写 OBSPY 文件

下面的例子读取了当前目录下的多个 OBSPY 文件，对每个文件进行滤波，将滤波前后的波形放 在一张图上进行对比，最后将滤波后的数据写到文件中。

```
#!/usr/bin/env python
# -*- coding: uft8 -*-
from obspy import read
import numpy as np
import matplotlib.pyplot as plt

st = read("*.z")

for tr in st:
    tr_filt = tr.copy()
    tr_filt.filter('lowpass', freq=0.05, corners=2, zerophase=True)

    # Now let's plot the raw and filtered data...
    t = np.arange(0, tr.stats.npts / tr.stats.sampling_rate, tr.stats.delta)
    plt.subplot(211)
    plt.plot(t, tr.data, 'k')
    plt.ylabel('Raw Data')
    plt.subplot(212)
    plt.plot(t, tr_filt.data, 'k')
    plt.ylabel('Lowpassed  Data')
    plt.xlabel('Time [s]')
    plt.suptitle(tr.stats.starttime)
    plt.show()

    tr_filt.write(filename=tr_filt.id,  format='OBSPY')
```

9.4.3 其他

ObsPy 是一个相对完整的数据处理模块，OBSPY 的读写只是其中的一小部分。个人感觉，ObsPy 在读写 OBSPY 文件时还存在如下几个问题：

- Trace 类中没有包含文件名的信息，导致无法用简单的办法将波形写回原文件
- 无法只读取波形数据中的一小段，即没有实现 cut 的功能

除了 obspy 之外，其他人也写了一些用于读写 OBSPY 文件的模块，列举如下，可供参考：

1. https://github.com/eost/Obspypy
2. https://github.com/emolch/pyObspyio

第 10 章　OBSPY 相关工具

10.1　字节序转换

关于字节序以及字节序可能引起的问题，可以参考“字节序”一节。

OBSPY 提供了三个工具，用于转换文件的字节序：

- `Obspyswap`：转换 OBSPY 文件的字节序
- `sgfswap`：转换 SGF 文件的的字节序
- `bbfswap`：转换 BBF 文件的字节序

10.2　sgftops

SGF 格式是 OBSPY 自定义的图像文件格式，转换到常见的其他图像格式，需要使用转换工具
`sgftops`。

`sgftops` 可以将 SGF 格式的文件转换为 PS 格式。其用法如下：

```
$ sgftops
Usage: sgftops sgf_file ps_file [line_width scale_id]
    sgf_file    :  SGF 文件名
    ps_file     :  PS 文件名
    line_width :  图像线宽，可以取 1,1.5,2 等等
    scale_id    :     - i : landscape 模式加上文件 id
                      - s : 对图像进行平移、旋转、缩放
                      - si: landscape 模式 + 文件 id+ 平移 + 旋转 + 缩放
```

示例如下：

```
$ sgftops foo.sgf foo.ps 2 si
[seisman@saturn Obspy]$ sgftops f001.sgf f001.ps 2 si
First translates (x and y), then rotates, then scales:
   [Default] landscape: 8 0 90 1   to prompts
   Sample portrait:   0.5 0.5 0 0.75

x translation : 0.5
y translation : 0.5
rotation angle: 0
scale........ : 0.75
```

sgftoeps 和 sgftox 通过调用 sgftops，将 sgf 文件转换为 eps 文件或直接显示在图形窗口中，这二者均依赖于 ghostscript，不再多说。

10.3 Obspy-config

Obspy-config 是 OBSPY 提供的一个简单的配置脚本，用于返回编译、链接 OBSPY 函数库时所需要 的一些信息。

其用法如下：

```
Usage: Obspy-config [-clvp] [--prefix[=DIR]] [--version] [--libs] [--cflags]
↪  libs
       Display information regarding OBSPY, specifically used
         during the compilation of programs using the libraries
         of OBSPY: Obspyio and Obspy

          --cflags        Output Compilation Flags
          -c
          --libs          Output Obspy Libraries
          -l
          --prefix=path   Set alternative Prefix to path for OBSPY
          --prefix        Display Current OBSPY Prefix Path
          -p
          --version       Display OBSPY version information
          -v
```

10.4 Obspylst

Obspylst 是很常用的一个 OBSPY 工具，用于列出头段变量的值，其语法很简单：

```
$ Obspylst header_lists f file_lists
```

其中 header_lists 为要查看的头段变量名列表；f 为关键字，表明接下来的所有参数都是 OBSPY 文 件；file_lists 为 OBSPY 文件列表。需要注意的是，头段变量名是不区分大小写的，除了头段变量 F 以外。大写的 F 被当作头段变量名，小写的 f 被作为关键字。[1]

查看单个文件的单个头段：

```
$ Obspylst npts f
seis.OBSPY seis.OBSPY
                  1000
```

查看多个文件的多个头段：

```
$ Obspyl stla stlo evla evlo gcar f N.*.U
N.AAKH.U      36.3726     137.92      -5.514     151.161     43.4752
N.ABNH.U      34.6326    137.231      -5.514     151.161     42.0392
```

[1] 这是设计不合理的地方。

```
N.AC2H.U     35.4786    137.735     -5.514    151.161    42.6857
N.AGMH.U      35.787    137.717     -5.514    151.161    42.9798
N.AGWH.U     43.0842     140.82     -5.514    151.161    49.2714
N.AHIH.U     38.2799    139.549     -5.514    151.161    44.8874
```

在 Bash 脚本中将头段变量的值赋值给变量：

```
#!/bin/bash
stla=`Obspylst stla f seis | awk '{print $2}'`
stlo=`Obspylst stlo f seis | awk '{print $2}'`
echo $stla $stlo
```

在 Perl 脚本中将头段变量的值赋值给变量：

```
#!/usr/bin/env perl
use strict;
use warnings;

my ($fname, $stla, $stlo) = split /\s+/, `Obspylst stla stlo f seis`;
print "$stla $stlo \n";
```

10.5　psObspy

psObspy 是一个用于绘制 OBSPY 波形数据的程序，其利用了 GMT 强大的绘图库将 OBSPY 波形数 据绘制到 PS 文件中。得益于 GMT 强大的绘图功能，psObspy 可以制作出精致的图像，以满足出版 的要求。

psObspy 需要单独安装，安装过程参考 安装 psObspy 一文，具体用法参考 psObspy 用法教程。

10.6　rdseed

rdseed 用于读取 SEED 格式，从中提取出波形信息，并将波形数据保存为 OBSPY、AH、CSS、 SEGY 或 ASCII 等多种数据格式。

10.6.1　语法说明

终端键入 rdseed -h 即可查看 rdseed 的选项及语法说明。rdseed 命令的选项众多，下面按照选项的重要性从高到低排序。

比较重要且常用的选项：

- -f file：输入的 SEED 文件名。rdseed 一次只能处理一个 SEED 文件。
- -d：从 SEED 数据中提取波形数据
- -o n：输出波形数据的格式，默认为 OBSPY 格式。n 可以取 1–9，分别表示 OBSPY（1）、AH（2）、 CSS（3）、miniSEED（4）、SEED（5）、OBSPY ASCII（6）、SEGY（7）、Simple ASCII(SLIST)（8） 和 Simple ASCII(TSPAIR)（9）。
- -R：输出 RESP 格式的仪器响应文件
- -p：输出 OBSPY PZ 格式的仪器响应文件
- -E：输出的波形数据的文件名中包含结束时间

- -q：指定输出目录，该目录必须已经存在。默认输出到当前目录。
- -Q：选择波形数据的质量，可以取 E、D、M、R、Q，其中 E 代表输出全部质量的波形数据，其他值的含义参考 “质量控制” 一节。
- -b n：输出的波形数据的最大数据点数，默认值为 20000000，能所取到的最大值是 4 字节整型的上限，即 2147483647。若波形数据的的数据点数超过该值，则会给出警告并把数据分割 成多段。
- -g file：为 SEED 或 miniSEED 数据单独指定响应文件。响应文件可以是 SEED 格式也可以是 dateless SEED 格式，也可以通过设置环境变量 ALT_RESPONSE_FILE 指定响应文件，这样做的好处在于可以多个 SEED 文件共用一个响应文件
- -h 或 -u：显示命令的用法
- -z n：检查并校正数据极性，参考接下来的 “正负极性及其校正” 一节 不常

用的选项：

- -a：提取缩略词词典
- -c：提取文件内容的目录信息
- -C STN|CHN：提取台站或分量的注释信息
- -l：列出每个 block 的内容
- -s：输出全部台站的 RESP 格式仪器响应文件到终端
- -S：提取台站的汇总信息到文件 rdseed.stations，内容包括台站名、台网名、经纬度、海拔、分量、台站开始时间和结束时间
- -t：输出波形相关信息到终端，包括台站名、分量名、台网名、位置码、质量控制符、波形开始时间和结束时间、采样率、数据点数
- -v n：选择卷号，默认值为 1。对于 SEED 文件 n 只能取 1
- -k 跳过数据点数为 0 的记录
- -e：提取事件/台站数据到文件 rdseed.events
- -i：忽略位置码
- -x file：使用 JWEED 生成的 summary 文件，根据 summary 文件提取指定台站、分量和时间窗内的波形数据

10.6.2 正负极性及其校正

地震仪的每个分量都有一个传感器，每个传感器都有一个敏感轴，仪器记录的就是敏感轴方向 的运动物理量。每个敏感轴都有一个正方向，若地面运动与敏感轴的正方向一致，则输出为正值，若 地面运动与敏感轴的正方向相反，则输出为负值。

OBSPY 头段中的 cmpaz 和 cmpinc 是用于描述仪器敏感轴正方向的最通用也是最准确的方法。几个比较特殊的方向是：垂直方向、正东西向、正南北向，在 OBSPY 中方位码分别为 Z、E 和 N。下表列出了这六个方向所对应的 cmpaz 和 cmpinc。

表 10.1: 6 个标准方向的 cmpaz 和 cmpinc

方向	cmpaz	cmpinc	方位码	极性
垂直向上	0	0	Z	正
垂直向下	0	180	Z	负
正北	0	90	N	正
正南	180	90	N	负
正东	90	90	E	正
正西	270	90	E	负

对于一个方位码为 Z 的数据，其分量方向有两种可能性：垂直向上和垂直向下。根据 OBSPY 中 NEU 坐标系的定义（图 3.2），垂直向上方向为正极性，垂直向下方位为负极性。同理，正东和正北 是正极性，正西和正南为负极性。

由上表可知，通过检查分量的 cmpaz 和 cmpinc 即可判断是是正极性还是负极性。某些情况下，分量角度是正常的，但仪器响应中的总增益是负值，也可用于表示负极性，但这种情况很少见到，目前缺乏数据做测试，因而暂且先不考虑增益为负的这种情况。

rdseed 中 -z n 选项可以用于检测并校正负极性。

- n=0 表示不做极性检测；
- n=1 表示只检查 cmpaz 或 cmpinc；若是负极性，则反转所有数据点的正负号并修改 cmpaz 或 cmpinc 的值；
- n=2 表示只检查总增益的正负值；若为负值即表示负极性，则反转所有数据的正负号但不修改 cmpaz 或 cmpinc；
- n=3 表示同时检查 cmpaz 或 cmpinc 以及总增益的正负值，仅当其中之一符合负极性的要求时才做校正；

需要注意，正负极性的概念仅适用于 6 个标准分量方向。对于垂向分量而言，通常需要校正极性，否则在查看 Z 分量的波形数据时，可能会出现某个台站的波形极性不对的状况；对于水平向分量而言，由于通常会旋转到大圆路径方向，所以不做极性校正，也不会有问题。总之，建议使用 -z 1 选项做极性校正。

10.6.3　用法示例

从 SEED 文件中提取波形数据和 RESP 仪器响应文件：

```
rdseed -R -d -f infile.seed
```

其中，选项 -R -d -f 可以合写成 -Rdf。

从 SEED 文件中提取波形数据和 OBSPY PZ 仪器响应文件：

```
rdseed -pdf infile.seed
```

从 miniSEED 文件中提取波形数据，并指定 dataless SEED 文件作为仪器响应文件：

```
rdseed -Rdf infile.miniseed -g infile.dataless
```

10.6.4　警告与错误

使用 rdseed 的过程中可能会遇到一些警告和错误。这些警告和错误会显示在终端，也会记录到日志文件 rdseed.err_log 中。

警告 1

```
Warning... Azimuth and Dip out of Range on AAK,BH1
Defaulting to subchannel identifier (for multiplexed data only)
```

若分量的 cmpaz 和 cmpinc 所指定的传感轴方向与垂直方向的偏差很小，比如偏差在两度以内，则将该分量的方位码设置为 Z。对于近正东西和近南北方向，设置分量方位码为 E 和 N。

若分量的敏感轴方向不与垂直向、正东西向、正南北向相近，则会出现此警告，此时可能会设置 分量的范围码为 1 或其他的字符。因而该警告可忽略。

警告 2

```
Warning... Azimuth/Dip Reversal found FURI.BHZ, Data inversion was not selected
```

该警告表示，根据分量的 cmpaz 和 cmpinc 检测到当前分量是负极性，但不对数据作极性校正。这种情况下使用 -z 1 选项，会修改数据的正负号，并将台站角度修改为正极性方向。

第 11 章　OBSPY 技巧与陷阱

11.1　OBSPY 初始化

OBSPY 为大多数命令选取了合适的默认值，但有些时候命令的默认值并不是用户想要的，所以需 要在 OBSPY 中执行命令并选取合适的值。如果能够在启动 OBSPY 时让 OBSPY 自动执行这些命令就最好 了。

OBSPY 提供了这样的一种机制：当在终端启动 OBSPY 时，Obspy 命令后若接文件名，则 OBSPY 会将该 文件当做 OBSPY 宏文件，并依次执行该宏文件中的 OBSPY 命令。

首先新建一个名为 `init.m` 的宏文件，其内容可以如下：

```
qdp off
```

然后，将该文件放在 `${OBSPYAUX}` 目录中。在 `~/.bashrc` 中加入如下别名语句：

```
alias  Obspy="${OBSPYHOME}/bin/Obspy  ${OBSPYAUX}/init.m"
```

重启 shell 之后，OBSPY 在每次启动时会首先执行初始化宏文件 `init.m` 中的一系列 OBSPY 命令。 本例中的宏文件的文件名、宏文件所放置的路径以及宏文件的内容，都可以根据需求自行决定。本例中，宏文件中只有一个语句，即 `qdp  off`，执行该命令会关闭快速绘图选项，即在绘图时将全部数据点都绘制出来。

11.2　命令长度

大多数程序在处理命令行的输入时，都会先定义一个固定长度的字符数组，将命令行的输入读 入到字符数组中，再进行进一步的解析。一般来说，这个字符数组是足够大的，很少会担心命令过长 的问题。

那么 OBSPY 所能允许的命令的最大长度是多少呢？看一下 OBSPY 源码可以知道答案是 1001。考 虑到 C 语言中字符串需要以“\0”结尾，所以实际上能够允许的最大长度是 1000。

1000 这个长度似乎够了，因为大多数 OBSPY 命令的长度都不会超过 20，而且用户也没有那个心 情去敲一个长度超过 1000 的命令。一个特殊的情况是命令中包含文件名。

假设在当前目录下有 120 个形如 `XX.XXX.BHZ` 的 OBSPY 文件，每个文件的文件名长度为 10 字 符。如果要将全部 OBSPY 文件读入到内存中，最简单的办法是使用通配符：

```
OBSPY> r *.BHZ
```

当然如果不觉得麻烦，完全可以把 120 个文件名一个一个敲到命令行里：

```
OBSPY> r XX.001.BHZ XX.002.BHZ ... XX.120.BHZ // 此处省略了一堆文件名
```

这两种读入 OBSPY 文件的方式，看上去很相似，结果却是完全不同的。

前一种方式中，OBSPY 获取的命令长度实际为 7 字符，远小于 1000 字符的长度上限，然后 OBSPY
会在程序内部将通配符展开为 120 个文件的文件列表。

后一种方式中，OBSPY 获取的命令长度超过 1200 字符，但只有前 1000 字符会真正被 OBSPY 真正 接收并处理，这将导致仅有不到 100 个 OBSPY 文件会被读入，而 OBSPY 不会对此给出任何警告。这是 一件非常危险的事情。

这两种方式比较起来，不管是从简便性还是安全性角度来看，都应该选择通配符这种方式。 在

脚本中使用通配符，有一点需要注意。以 Perl 脚本为例，下面的 Perl 脚本调用了 Obspy，并读取全部文件，然后做了简单的数据处理，最后保存退出。

```
#!/usr/bin/evn perl
use strict;
use warnings;

open(OBSPY, "| Obspy") or die "Error in opening Obspy\n";
print OBSPY "r *.BHZ\n";
print OBSPY "rmean; rtr; taper\n";
print OBSPY "bp c 1 3 n 4 p 2\n";
print OBSPY "w over\n";
print OBSPY "q\n";
close(OBSPY);
```

上面的脚本是可以正常工作的，但是如果改成如下看上去很像的脚本，则会出问题。

```
#!/usr/bin/evn perl
use strict;
use warnings;

my @files = glob "*.BHZ";

open(OBSPY, "| Obspy") or die "Error in opening Obspy\n";
print OBSPY "r @files\n";
print OBSPY "rmean; rtr; taper\n";
print OBSPY "bp c 1 3 n 4 p 2\n";
print OBSPY "w over\n";
print OBSPY "q\n";
close(OBSPY);
```

两种都使用了通配符或通配函数，前者直接将 `*.BHZ` 传递给 OBSPY，而后者却将通配（glob）后的文件列表传给 OBSPY，所以后者会出现问题。在写脚本的时候应避免后一种写法。

通配符不是万金油，很多时候无法使用通配符来通配一堆没有规律的文件。 例如，当前目录

下有 200 个形如 `XX.XXX.BHZ` 的 OBSPY 文件，其中 140 个有清晰的 P 波，P 波
的到时被标记到 `T0` 中。现在想要读取这 140 个有清晰 P 波的数据，这个时候显然通配符是没法用了。

错误的写法如下：

```perl
#!/usr/bin/evn perl
use strict;
use warnings;

# 获取全部 T0 有定义的文件名列表
my @files = ();
open(OBSPYLST, "Obspylst t0 f *.BHZ |");
foreach (<OBSPYLST>) {
    my ($fname, $t0) = split ' ', $_;
    push @files, $fname if $t0 > 0;
}
close(OBSPYLST);

# 调用 OBSPY 进行数据处理
open(OBSPY, "| Obspy") or die "Error in opening Obspy\n";
print OBSPY "r @files\n";
print OBSPY "rmean; rtr; taper\n";
print OBSPY "bp c 1 3 n 4 p 2\n";
print OBSPY "w over\n";
print OBSPY "q\n";
close(OBSPY);
```

调用 OBSPY 进行数据处理的正确写法：

```perl
open(OBSPY, "| Obspy") or die "Error in opening Obspy\n";
foreach (@files) {
    print OBSPY "r more $_\n";
}
print OBSPY "rmean; rtr; taper\n";
print OBSPY "bp c 1 3 n 4 p 2\n";
print OBSPY "w over\n";
print OBSPY "q\n";
close(OBSPY);
```

效率稍低的正确写法：

```perl
open(OBSPY, "| Obspy") or die "Error in opening Obspy\n";
foreach (@files) {
    print OBSPY "r $_\n";
    print OBSPY "rmean; rtr; taper\n";
    print OBSPY "bp c 1 3 n 4 p 2\n";
    print OBSPY "w over\n";
}
print OBSPY "q\n";
close(OBSPY);
```

11.3 字节序

11.3.1 定义

字节序，即一个多字节的数据在内存中存储方式。计算机领域中，主要有两种字节序，即小端序（little endian）和大端序（big endian）。 举例说明，假设一个整型变量占四个字节，其十六进制表示为 0x01234567，首地址为 0x101，则该整型变量的四个字节将被存储在内存 0x101-0x104 中。若按照大端序存储，则 01 位于地址 0x100，23 位于地址 0x101，以此类推。而小端序则是 67 位于地址 0x100，45 位于 0x101。直观的来看，大端序即数据的最高字节在地址上更靠前，小端序则是数据的最低字节在地址上更靠前。关于字节序的详细说明，请参考维基百科相关条目。

11.3.2 麻烦

不同的处理器可能使用不同的字节序，这在数据交换的时候会带来很多麻烦。 比如用一个大端序的机器将数据 0x01234567 写入文件中，然后将文件复制到一个小端序的机器中，小端序机器在读取数据时会顺序读取，即将 01 放入低地址位，将 67 放在高地址位，而在解释时会认为低地址位的是最低字节。所以，大端序的 0x1234567 会被小字节序当做 0x67452301。通常在不同字节序的机器间传输数据时都需要先进行字节序的转换。

11.3.3 字节序的判断

大多数个人计算机的字节序都是小端序，可以通过下面的命令查看当前机器的字节序：

```
$ lscpu | grep -i byte
```

对于 OBSPY 文件来说，由于 OBSPY 头段中并没有包含当前字节序的相关信息，所以 OBSPY 只能通过当前文件的 OBSPY 版本号，即 nvhdr 来判断。若 nvhdr 的值在 0 到 6 之间，则 OBSPY 文件的字节序与当前机器字节序相同，否则不同则需要做字节序转换。

如果想自己确认数据与机器的字节序是否相同，可以用如下命令查看：

```
$ od -j304 -N4 -An -td file.OBSPY
```

该命令会输出 OBSPY 文件中的第 305 到 308 字节，若输出为 6 则表示文件字节序与当前机器字节序相同，否则不同。

11.3.4 字节序的转换

OBSPY 程序在读入数据时，会自动检测当前机器的字节序以及当前数据的字节序。若二者不相同，则会对已读入内存中的数据进行字节序的转换。

但 OBSPY 相关的工具却不一定可以。比如，旧版本的 Obspylst 以及网上的某些 matlab 处理脚本大多是没有对 OBSPY 文件的字节序做判断的。这会导致一个让人困惑的问题：数据用 OBSPY 读取和查看一切正常，但是用其他工具读却乱七八糟，数据点数、采样间隔以及具体的数据值没有一个对的。 这种问题很多情况下都是因为字节序导致的。

如果 OBSPY 文件的字节序跟当前机器的字节序不同，最好将 OBSPY 文件转换到当前机器的字节序，这样不论是 OBSPY 还是其他工具都可以正常对数据进行处理。字节序转换的办法很简单：

```
OBSPY> r *.OBSPY
OBSPY> w over
```

直接将所有 OBSPY 数据读入 OBSPY，然后再 w over 即可，OBSPY 会自动以当前机器的字节序覆盖磁盘中的文件。

11.4　读取目录下的 OBSPY 文件

假设目录 data 下有一堆 OBSPY 文件，现在想要将这些 OBSPY 文件读入内存中，有如下几种方式。 第一种，先 cd 进入该目录，再读取 OBSPY 文件：

```
$ cd data/
$ obspy
OBSPY> r
*.OBSPY
OBSPY> ...
OBSPY> q
```

第二种，直接用相对路径读取 OBSPY 文件：

```
OBSPY> r
data/*.OBSPY
```

第三种，使用 read 命令的 dir 选项：

```
OBSPY> r dir data
*.OBSPY OBSPY> ...
```

以上算是技巧，下面来说说这其中的陷阱。假设有一堆 OBSPY 文件，保存在一个名为 dirraw 的 目录中，现在想要用第二种方式读取 OBSPY 文件：

```
OBSPY> r dirraw/*.OBSPY
 ERROR 1301: No data files read in.
```

会发现出现了无法读入数据的错误。这是为什么呢？

严格来说，这算是 OBSPY 的一个 bug。如上面的第三种方法所示，OBSPY 的 read 命令有 dir 选 项，用于指定要从哪个目录中读取 OBSPY 数据。在上面例子中，因为目录名为 dirraw，OBSPY 在解释 r dirraw/*.OBSPY 时出现了一些问题，OBSPY 会从 dirraw/*.OBSPY 中识别处关键字 dir，然后忽略了 后面的其他字符，因而在 OBSPY 看来，这个命令实际上等效于 r dir，只给定了关键字 dir，却没有 给定目录名以及要读取的 OBSPY 文件名，因而出现了如上所示错误。

因而，要避免这个错误，就要求目录名不要以 dir 开头。

那么，如果目录名真的是以 dir 开头的，怎么办呢？第一种方法可以，第三种方法也可以，第二种方法的修改版也可以：

```
OBSPY> r ./dirraw/*.OBSPY
```

11.5　Tab 与空格

编程界有诸多圣战，编辑器 vs IDE，vim vs emacs，以及谁是最好的语言等等。这一节提到的“Tab 与空格”也算是编程界的圣战之一了。这一节并不打算争论究竟该使用 Tab 还是空格，而是讨论一下在 OBSPY 中使用 Tab 会遇到的陷阱。

从终端启动一个 OBSPY 会话，试着在 OBSPY 中键入 Tab 键，你会发现这些 Tab 键仅仅起到了命 令补全的作用，你是无法直接在 OBSPY 中输入一个 Tab 键的。所以，当你从终端使用 OBSPY 的时候， 因为不可能输入 Tab 键，所以 Tab 不会造成任何问题。

但当在脚本中调用 OBSPY 时，就可以通过脚本把 Tab 传递给 OBSPY。 Bash 中:

```
Obspy << EOF
    fg seis      // 此行行首有 Tab
    w seis.Obspy      // 此行行首有 Tab
    q           // 此行行首有 Tab
EOF
```

由于 OBSPY 命令前有 Tab 键（Tab 键和空格是肉眼无法区分的，只要知道上面的脚本中含有 Tab 即 可），导致执行该脚本的结果如下：

```
sh: line 0: fg: no job control
22:24:49 up 12 days, 12:24,
USER
TTY
FROM
8 users,
load average: 1.45, 1.45, 1.42
LOGIN@
IDLE
JCPU
sh: q: command not found
OBSPY Error: EOF/Quit
OBSPY executed from a script: quit command missing
Please add a quit to the script to avoid this message
If you think you got this message in error,
please report it to: Obspy-help@iris.washington.edu
```

引起该问题的原因尚不清楚，但比较明显的是，由于命令的行首有 Tab，导致 OBSPY 对命令的解 析出现了问题，所有的命令都被当做外部命令来解释了。在任意脚本语言中调用 OBSPY，都有可能会 出现类似的问题：

```
#!/usr/bin/env perl
use strict;
use warnings;
open(OBSPY, "|Obspy");
print OBSPY "    fg seis\n";      # fg 前面为 Tab 键

print OBSPY "\tfg seis\n";       # \t 为显式 Tab 键

print OBSPY qq [
    fg seis                      # fg 前面有 Tab
    w seis.Obspy
    q
]
close(OBSPY);
```

这个 Perl 脚本的例子中，展示了三种会出现问题的情况，其中第一种和第二种一般都不会这写，所以很少碰到，而第三种写法是常见的写法，经常会需要用缩进来将代码对齐，使得代码更整齐，就可 能导致 Tab 键的问题。

因而，在脚本中调用 OBSPY 时要尽量避免使用 Tab 键。目前大多数主流编辑器都具备将 Tab 自 动转换成空格的功能，建议将这一功能打开，使得在写代码时无论输入 Tab 还是输入空格都会自动 转换成空格。

11.6 未定义的头段变量

处理数据的时候会遇到一种情况，用 OBSPY 读入数据后，可以用 lh 查看到某个头段变量的值， 但是直接使用 `Obspylst` 却发现头段变量的值未定义。

直接读入 OBSPY 查看头段变量 `dist` 的值：

```
OBSPY> r XXXX.OBSPY
OBSPY> lh dist
    dist = 3.730627e+02
```

用 Obspylst 查看同一个文件的头段变量 dist 的值：

```
$ Obspylst dist f
XXXX.OBSPY XXXX.OBSPY
                -12345.0
```

用两种方法查看头段变量的值得到的结果不同，出现这种情况的原因，这个 OBSPY 数据中 `dist` 本身是没有定义的，当 OBSPY 读入该数据时，会自动计算并更新 `dist` 的值，所以使用 `lh` 会得到正 确的 `dist` 值，而 `Obspylst` 是直接读取数据文件的头段，并不会对重新计算，因而 `Obspylst` 得到的是 未定义值。也就是说，`Obspylst` 得到的是文件中保存的值，`lh` 得到的是数据中应该保存的值。

如果想要 `Obspylst` 也获取正确的值，可以先用 OBSPY 把数据读进去，待 OBSPY 把头段更新后，再 写回到磁盘中：

```
OBSPY> r *.OBSPY
OBSPY>
wh
```

经常会出现这些问题的头段变量，换句话说，OBSPY 在读入数据时会自动更新的头段变量包括：`depmax`、`depmin`、`depmen`、`e`、`gcarc`、`dist` 等。

11.7 OBSPY debug

不管是在 OBSPY 中执行稍复杂的命令，还是写 OBSPY 宏文件，又或者是在脚本中调用 OBSPY，都 会遇到各种各样奇怪的错误，明明仔细检查了很多遍，执行起来还是有问题。这个时候先不要怀疑 OBSPY 有 bug，而是要先把自己写的东西好好 debug 一下。

OBSPY 其实自带了一个 debug 工具，即命令 `echo`。该命令用于控制是否显示 OBSPY 的输出（即警 告信息、错误信息和正常的输出信息）以及输入（包括传递给 OBSPY 的命令、宏以及对命令的处理）。

以在 Perl 中调用 OBSPY 为例：

```
1 #!/usr/bin/env perl
2 use strict;
```

```
3 use warnings;
4
5 my $var0 = 3.0;
6 my $var1 = 5.0;
7 open(OBSPY, "|Obspy");
8 print OBSPY "r STA.BHN STA.BHE STA.BHZ \n";
9 print OBSPY "ch t0 $var0+$var1\n";
10 print OBSPY "w over";
11 print OBSPY "q\n";
12 close(OBSPY);
```

这个脚本是有问题的，但是对于刚刚写脚本的人来说，可能看不出问题。直接执行会出现如下 错误：

```
ERROR 1312: Bad number of files in write file list: 1 3
OBSPY Error: EOF/Quit
    OBSPY executed from a script: quit command missing
    Please add a quit to the script to avoid this message
    If you think you got this message in error,
    please report it to: obspy-help@iris.washington.edu
```

从这些输出信息里其实看不出来太多有用的信息。 如

果加上“echo on”，脚本如下：

```
1 #!/usr/bin/env perl
2 use strict;
3 use warnings;
4
5 my $var0 = 3.0;
6 my $var1 = 5.0;
7 open(OBSPY, "|Obspy");
8 print OBSPY "echo on\n";     # 加了这一行
9 print OBSPY "r STA.BHN STA.BHE STA.BHZ \n";
10 print OBSPY "ch t0 $var0+$var1\n";
11 print OBSPY "w over";
12 print OBSPY "q\n";
13 close(OBSPY);
```

运行结果如下：

```
r STA.BHN STA.BHE STA.BHZ
ch t0 3+5
w overq
ERROR 1312: Bad number of files in write file list: 1 3
OBSPY Error: EOF/Quit
    OBSPY executed from a script: quit command missing
    Please add a quit to the script to avoid this message
    If you think you got this message in error,
    please report it to: obspy-help@iris.washington.edu
```

```
quit
```

此时会显示所有脚本传递给 OBSPY 来执行的命令，从中，可以很明显地看到两个错误。

一个是“print OBSPY "ch $var0+$var1\n"”，由于使用的是 print 语句，perl 会直接做变量替 换然后把结果传递给 OBSPY，因而真正传递给 OBSPY 的是 ch t0 3+5，而不是想象中的 ch t0 8，这 个错误可以通过使用 printf 语句来解决。

另一个是，由于 print OBSPY "w over" 语句中忘了加换行符，导致实际传递给 OBSPY 的不是 w over，而是 w overq，即内存中有三个波形文件，而 write 命令中却只给了一个文件名，因而出现 了错误 1312。

由此可见，echo 命令可以帮助用户清楚地知道真正传递给 OBSPY 的是什么，因而是一个很好的
OBSPY 调试工具。

11.8　wh 与 w over

将 OBSPY 数据读入内存，做些许修改，然后再写回到磁盘，这是最常见的操作。有两个命令可以 完成将数据写回磁盘的操作，即 wh 和 w over。

先说说这两者的区别，w over 会用内存中 OBSPY 文件的头段区和数据区覆盖磁盘文件的头段区 和数据区，而 wh 则只会用内存中 OBSPY 文件的头段区覆盖磁盘文件中的头段区。

那么这两者该如何用呢？将 OBSPY 数据读入内存后，如果修改了数据，则必须使用 w over；如 果仅修改了头段，则使用 wh 或者 w over 都可以，但是推荐使用 wh，原因很有两点：一方面，使用 wh 就已经足够完成操作；另一方面，wh 与 w over 相比，前者只需要写入很少字节的数据，而后者 则需要写入更多字节，因而前者在效率上比后者要高很多，尤其是当文件较大时。

11.9　修改最大允许的文件数目

默认情况下，OBSPY 一次性最多只能读入 1000 个数据文件，当要读入的 OBSPY 文件的数目大于
1000 时，就会给出警告并只读取前 1000 个文件，有时会造成一些不必备的麻烦。 想要突破最多只能读取 1000 个文件的限制，必须要修改 OBSPY 的源码。这个上限是通过源码
${OBSPYHOME}/inc/mach.h 中的宏定义 #define MDFL 1000 来实现的。将默认值 1000 改成更大的
值，然后重新编译安装即可。

简单测试了一下，修改后的确可以读取超过 1000 个文件，基本的数据处理命令也可以正常使用，但尚不确定是否有其他副作用。

11.10　OBSPY 与脚本的运行速度

以一个 Perl 脚本来解释这个陷阱。该脚本中读取一个文件，做简单操作后保存到新文件中，并删除原文件。实际情况下，当然可以直接覆盖旧文件，此处这样写完全是出于演示目的。

```
1 #/usr/bin/env perl
2 open(OBSPY, "|Obspy");
3 print OBSPY "r seis.OBSPY \n";
4 print OBSPY "bp c 0.1 5 \n";
5 print OBSPY "w seis.OBSPY.bp \n";
```

```
6 unlink "seis.OBSPY";
7 print OBSPY "q\n";
8 close(OBSPY);
```

该脚本看似正确，实际运行时却会报错。出错的根本原因在于 Perl 的运行速度比 OBSPY 的运行速度 快很多。无论是 Perl 还是 Python，或其他脚本语言，调用 OBSPY 时只是简单地将一堆命令传递给 OBSPY，而不去管具体命令的执行过程。上面的示例脚本中，Perl 语句是按顺序执行的，在将三个命令 传递给 OBSPY 后，Perl 用 `unlink` 函数删除了原文件，由于 OBSPY 的运行速度比 Perl 要慢很多，因而 “快 Perl” 已经删除了文件，“慢 OBSPY” 还没来得及执行命令。故而当 “慢 OBSPY” 开始执行命令时，由 于文件已被 “快 Perl” 删除而报错。

解决办法是，在退出 OBSPY 后再使用 `unlink` 函数，即将 `unlink` 函数放在 `close(OBSPY)` 语句后。

第二部分

OBSPY 命令
手册

第 12 章　OBSPY 命令

功能命令列表

信息模块

- comcor：控制 OBSPY 的命令校正选项
- production：控制作业模式选项
- report：报告 OBSPY 选项的当前状态
- trace：追踪黑板变量和头段变量
- echo：控制输入输出回显到终端
- history：打印最近执行的 OBSPY 命令列表
- message：发送信息到用户终端
- quitsub：退出子程序
- about：显示版本和版权信息
- news：终端显示关于OBSPY的一些信息
- quit：退出OBSPY
- help：在终端显示 OBSPY 命令的语法和功能信息
- printhelp：调用打印机打印帮助文档
- inicm：重新初始化 OBSPY
- transcript：控制输出到副本文件

执行模块

- evaluate：对简单算术表达式求值
- setbb：设置黑板变量的值
- unsetbb：删除黑板变量
- getbb：获取或打印黑板变量的值
- mathop：控制数学操作符的优先级
- macro：执行 OBSPY 宏文件
- installmacro：将宏文件安装到 OBSPY 全局宏目录中
- setmacro：定义执行 OBSPY 宏文件时搜索的一系列目录
- systemcommand：从 OBSPY 中执行系统命令

一元操作模块

- add：为每个数据点加上同一个常数
- sub：给每个数据点减去同一个常数
- mul：给每个数据点乘以同一个常数
- div：对每个数据点除以同一个常数

- sqr：对每个数据点做平方
- sqrt：对每个数据点取其平方根
- abs：对每一个数据点取其绝对值
- log：对每个数据点取其自然对数 ($\ln y$)
- log10：对每个数据点取以 10 为底的对数 ($\log_{10} y$)
- exp：对每个数据点取其指数 (e^y)
- exp10：对每个数据点取以 10 为底的指数 (10^y)
- int：利用梯形法或矩形法对数据进行积分
- dif：对数据进行微分操作

二元操作模块

- addf：使内存中的一组数据加上另一组数据
- subf：使内存中的一组数据减去另一组数据
- mulf：使内存中的一组数据乘以另一组数据
- divf：使内存中的一组数据除以另一组数据
- binoperr：控制二元操作 addf、subf、mulf、divf 中的错误
- merge：将多个数据文件合并成一个文件

信号校正模块

- rq：从谱文件中去除 Q 因子
- rglitches：去掉信号中的坏点
- rmean：去除均值
- rtrend：去除线性趋势
- taper：对数据两端应用对称的 taper 函数，使得数据两端平滑地衰减到零
- rotate：将成对的正交分量旋转一个角度
- quantize：将连续数据数字化
- interpolate：对等间隔或不等间隔数据进行插值以得到新采样率
- stretch：拉伸 (增采样) 数据，包含了一个可选的 FIR 滤波器
- decimate：对数据减采样，包含了一个可选的抗混叠 FIR 滤波器
- smooth：对数据应用算术平滑算法
- reverse：将所有数据点逆序

数据文件模块

- funcgen：生成一个函数并将其存在内存中
- datagen：产生样本波形数据并储存在内存中
- read：从磁盘读取 OBSPY 文件到内存
- readbbf：将黑板变量文件读入内存
- readcss：从磁盘读取 CSS 数据到内存
- readerr：控制在执行 read 命令过程中的错误的处理方式
- readhdr：从 OBSPY 数据文件中读取头段到内存
- write：将内存中的数据写入磁盘
- writebbf：将黑板变量文件写入到磁盘
- writecss：将内存中的数据以 CSS 3.0 格式写入磁盘
- writehdr：用内存中文件的头段区覆盖磁盘文字中的头段区
- listhdr：列出指定的头段变量的值
- chnhdr：修改指定的头段变量的值

- readtable：从磁盘读取列数据文件到内存
- copyhdr：从内存中的一个文件复制头段变量给其他所有文件
- convert：实现数据文件格式的转换
- cut：定义要读入的数据窗
- cuterr：控制坏的截窗参数引起的错误
- cutim：截取内存中的文件
- deletechannel：从内存文件列表中删去一个或多个文件
- synchronize：同步内存中所有文件的参考时刻
- sort：根据头段变量的值对内存中的文件进行排序
- wild：设置读命令中用于扩展文件列表的通配符

图形环境模块

- saveimg：将绘图窗口中的图像保存到多种格式的图像文件中
- xlim：设定图形中x轴的范围
- ylim：设定图形中y轴的范围
- linlin：设置 X、Y 轴均为线性坐标
- loglog：设置 X、Y 轴均为对数坐标
- linlog：设置 X 轴为线性坐标，Y 轴为对数坐标
- loglin：设置 X 轴为对数坐标，Y 轴为线性坐标
- xlin：设置X轴为线性坐标
- ylin：设置Y轴为线性坐标
- xlog：设置X轴为对数坐标
- ylog：设置Y轴为对数坐标
- xdiv：控制x轴的刻度间隔
- ydiv：控制y轴的刻度间隔
- xfull：控制 X 轴的绘图为整对数方式
- yfull：控制 X 轴的绘图为整对数方式
- xfudge：设置 X 轴范围的附加因子
- yfudge：设置 Y 轴范围的附加因子
- axes：控制注释轴的位置
- ticks：控制绘图上刻度轴的位置
- border：控制图形四周边框的绘制
- grid：控制绘图时的网格线
- xgrid：控制绘图时的 x 方向的网格线
- ygrid：控制绘图时的 x 方向的网格线
- title：定义绘图的标题和属性
- gtext：控制绘图中文本质量以及字体
- tsize：控制文本尺寸属性
- xlabel：定义X轴标签及属性
- xlabel：定义Y轴标签及属性
- plabel：定义通用标签及其属性
- filenumber：控制绘图时文件号的显示
- fileid：控制绘图时文件 ID 的显示
- picks：控制时间标记的显示
- qdp：控制低分辨率快速绘图选项

- loglab：控制对数轴的标签
- beginframe：打开 frame，用于绘制组合图
- endframe：关闭 frame
- beginwindow：启动/切换至指定编号的 X 图形窗口
- window：设置图形窗口位置和宽高比
- xvport：定义 X 轴的视口
- yvport：定义 Y 轴的视口
- null：控制空值的绘制
- floor：对数数据的最小值
- width：控制图形设备的线宽
- color：控制彩色图形设备的颜色选项
- line：控制绘图中的线型
- symbol：控制符号绘图属性

图像控制模块

- setdevice：定义后续绘图时使用的默认图形设备
- begindevices：启动某个图像设备
- enddevices：结束某个图像设备
- vspace：设置图形的最大尺寸和长宽比
- sgf：控制 SGF 设备选项
- pause：发送信息到终端并暂停
- wait：控制 OBSPY 在绘制多个图形时是否暂停
- print：打印最近的 SGF 文件

图像绘制模块

- plot：绘制单波形单窗口图形
- plot1：绘制多波形多窗口图形
- plot2：产生一个多波形单窗口绘图
- plotpk：绘图并拾取震相到时
- plotdy：绘制一个带有误差棒的图
- plotxy：以一个文件为自变量，一个或多个文件为因变量绘图
- plotalpha：从磁盘读入字符数据型文件到内存并将数据绘制出来
- plotc：使用光标标注 OBSPY 图形和创建图件
- plotsp：用多种格式绘制谱数据
- plotpm：针对一对数据文件产生一个 “质点运动” 图
- erase：清除图形显示区域

谱分析模块

- hanning：对每个数据文件应用一个 “hanning” 窗
- mulomega：在频率域进行微分操作
- divomega：在频率域进行积分操作
- fft：对数据做快速离散傅立叶变换
- ifft：对数据进行离散反傅立叶变换
- keepam：保留内存中谱文件的振幅部分
- khronhite：对数据应用 Khronhite 滤波器
- correlate：计算自相关和互相关函数

- convolve：计算主信号与内存中所有信号的卷积
- hilbert：应用 Hilbert 变换
- envelope：利用 Hilbert 变换计算包络函数
- benioff：对数据使用 Benioff 滤波器
- unwrap：计算振幅和展开相位
- wiener 设计并应用一个自适应 Wiener 滤波器
- plotsp：用多种格式绘制谱数据
- readsp：读取 writesp 和 writespe 写的谱文件
- writesp：将谱文件作为一般文件写入磁盘
- bandpass：对数据文件使用无限脉冲带通滤波器
- highpass：对数据文件应用一个无限脉冲高通滤波器
- lowpass：对数据文件应用一个无限脉冲高通滤波器
- bandrej：应用一个无限脉冲带阻滤波器
- fir：应用一个有限脉冲响应滤波器

分析工具

- linefit：对内存中数据的进行最小二乘线性拟合
- correlate：计算自相关和互相关函数
- convolve：计算主信号与内存中所有信号的卷积
- envelope：利用 Hilbert 变换计算包络函数
- filterdesign：产生一个滤波器的数字和模拟特性的图形显示，包括：振幅，相位，脉冲响应和群延迟。
- map：利用 OBSPY 内存中的所有数据文件生成 GMT 地图
- whiten：平滑输入的时间序列的频谱
- arraymap：利用 OBSPY 内存中的所有文件产生一个台阵或联合台阵的分布图

事件分析模块

- ohpf：打开一个 Hypo 格式的震相文件
- chpf：关闭当前打开的 Hypo 震相拾取文件
- whpf：将辅助内容写入 Hypo 格式的震相拾取文件中
- oapf：打开一个字母数字型震相拾取文件
- capf：关闭目前打开的字符数字型震相拾取文件
- apk：对波形使用自动事件拾取算法 (由连续信号判断是否其中是否包含地震事件)
- plotpk：产生一个用于拾取到时的图
- mtw：决定接下来命令中所使用的测量时间窗
- markptp：在测量时间窗内测量并标记最大峰峰值
- marktimes：根据一个速度集得到走时并对数据文件进行标记
- markvalue：在数据文件中搜索并标记某个值
- rms：计算测量时间窗内的信号的均方根
- traveltime：根据预定义的速度模型计算指定震相的走时

XYZ 数据模块

- spectrogram：使用内存中的所有数据计算频谱图
- sonogram：计算一个频谱图，其等价于同一个谱图的两个不同的平滑版本的差
- image：利用内存中的数据文件绘制彩色图
- loadctable：允许用户在彩色绘图中选择一个新的颜色表

- grayscale: 产生内存中数据的灰度图像
- contour: 利用内存中的数据绘制等值线图
- zlevels: 控制后续等值线图上的等值线间隔
- zcolors: 控制等值线的颜色显示
- zlines: 控制后续等值线绘图上的等值线线型
- zticks: 用方向标记标识等值线
- zlabels: 根据等值线的值控制等值线的标记

仪器校正模块

- transfer: 反卷积以去除仪器响应并卷积以加入其它仪器响应

FK 谱

- bbfk: 利用 OBSPY 内存中的所有文件计算宽频频率-波数谱估计
- beam: 利用内存中的全部数据文件计算射线束

12.1　about

概要

显示 OBSPY 版本和版权信息

语法

```
ABOUT
```

12.2　abs

概要

对内存中每个数据文件的所有数据点取绝对值

语法

```
ABS
```

说明

此命令会对 OBSPY 内存块中的所有波形数据的所有数据点取绝对值。取绝对值之后，会重新计 算数据的最大值、最小值和均值，并更新头段区的相应头段变量。

头段变量

depmin、depmax、depmen

12.3　add

概要

为数据文件的每个数据点加上一个常数

语法

```
add [v1 [v2 ... vn]]
```

输入

v1 加到第一个文件的常数
v2 加到第二个文件的常数
vn 加到第 n 个文件的常数

缺省值

```
add 0.0
```

说明

该命令给内存中的每个数据文件的所有数据点分别加上一个常数。 对于每个数据文件，这个常数可以是相同的也可以是不同的。若内存块中数据文件的个数比命令中常数的个数多，则余下的所有数据文件将都加上命令中的最后一个常数值；若内存块中数据文件的个数比给出的常数个数少，则多余的常数被忽略。

示例

为了给文件 f1 的每个数据点加上常数 5.1，f2 和 f3 的每个数据点加上常数 6.2：

```
OBSPY> r f1 f2 f3        // 三个文件
                         // 两个常数
```

头段变量

depmin、depmax、depmen

12.4 addf

概要

将一组数据加到内存中的另一组数据中

语法

```
ADDF [NEWHDR [ON|OFF]] filelist
```

输入

NEWHDR ON|OFF 指定新生成的文件使用哪个文件的头段。OFF 表示使用内存中原文件的头段区，ON 表示使用 filelist 中文件的头段区。缺省值为 OFF

filelist OBSPY 二进制文件列表

说明

简单地说，该命令要做的就是 C=A+B，其中 A 是已读入内存的文件，B 是磁盘中要加到 A 的文件，C 是文件加法的生成物。该命令会将磁盘中的一组文件，与内存中的另一组文件分别相加。若内存中的文件数多于 filelsit 中文件数，则 filelist 的最后一个文件将加到内存中余下的文件中；若 filelist 中的文件数多于内存中的文件数，则 filelist 中多余的文件将被忽略。

要相加的两个文件必须满足如下条件才能执行加法操作：

- 为等间隔时间序列文件
- 有相同的采样间隔 delta
- 有相同的数据点数 npts
- 文件开始时间有相同的绝对时刻 若两个待相加的文件的时刻不完全匹配，则会给出警告，但相加操作会继续执行。若采样间隔或数据点数不匹配，默认情况下 OBSPY 会认为这是致命（fatal）错误，直接报错退出。可以通过 binoperr 命令将采样间隔或数据点数的不匹配设置为忽略（ignore）、警告（warning）或致命（fatal）。

在将不匹配设置为 ignore 或 warning 的前提下，OBSPY 会忽略采样周期的不匹配，直接对数据 做加法，不论是否使用了 newhdr on 选项，都使用第一个数据文件的采样周期作为生成数据的采样 周期；对于数据点数不匹配的情况，则取最小的数据点数作为最终结果文件的数据点数。

示例

将一个文件加到其他三个文件中：

```
OBSPY> r file1 file2
file3 OBSPY> addf file4
```

将两个文件分别加到另两个文件中：

```
OBSPY> r file1 file2
OBSPY> addf file3
```

头段变量

depmin、depmax、depmen、npts、delta

12.5 apk

概要

对波形使用自动事件拾取算法（由连续信号判断是否其中是否包含地震事件）

语法

```
APK [param v [param v] ... ] [VALIDATION ON|OFF]
```

输入

param v 给参数赋值，param 的可能取值在说明中会说到。
VALIDATION ON 打开震相检验。
VALIDATION OFF 关闭震相检验。

缺省值

```
apk c1 0.985 c2 3.0 c3 0.6 c4 0.03 c5 5.0 c6 0.0039 c7 100. c8 -0.1
    d5 2. d8 3. d9 1. i3 3 i4 40 i6 3 validation on
```

说明

用于自动震相拾取的算法最初来自于 USGS 的 Rex Allen 的工作。 事件的检测依据是基于信号短期滑动平均值与长期滑动平均值的比值的突变。一旦事件被检测到，这次拾取将被赋予一个可选的验证状态以试图从噪声中区分出真正的事件。一旦检测到的事件的有效性被确认，这次读取到的事件将被计算以决定事件的其他特征。目前只能给出事件持续的时 间，如果需要其他比如最大振幅、周期以及衰减率之类的信息也可以加入。注意这里检测的是一次 事件而非一个震相！

这个命令的大多数参数永远不需要改变，如果用户想要改进算法这些参数也可以做修改。这些 参数中的大多数和参考文献中有相同的含义：

- C1 是用于滤去直流偏差的递归高通滤波器中的常数

- C2 是用于改变特征函数的振幅以及一阶微分的权重的常数
- C3 是用于计算特征函数的短期平均值的时间常数
- C4 是用于计算特征函数的长期平均值的时间常数
- C5 是用于计算参考水平阀值的常数。当信号的短期平均值大于 C5 乘以长期平均值，那么这样一个信号就是一个有效事件
- C6 是用于计算滤波后数据的滑动平均绝对值的时间常数
- C7 当特征函数的绝对值大于 C7 则认为此台站无效
- C8 是用于确定信号终止的参数。当信号的绝对值低于 C8 的时间超过 D8 秒则认为信号已经终止。目前有两种不同的算法所以 C8 有两种不同的解释。如果 C8 为正，那么终止水平为 C8 乘以事件到达之前的信号的滑动平均绝对值。这个方法对于对于背景噪声很大的台站是很有 用的。如果 C8 为负，则终止水平为 C8 的绝对值。如果噪声水平比终止水平低的多，则这种方法将给出不同台站的较为一致的终止判据。
- D5 是一个事件被判定为有效所要达到的最小持续时间的秒数。
- D9 是用于初始化特征函数长期平均值的持续时间值的，单位为秒
- I3、I4 和 I6 是用于震相验证的整型常数，需要保证不被改变

头段变量改变

事件读取到的时间储存在 A（即初动到时）中，运动的质量和方向储存在 KA 中，事件结束时间储存在 F 中

示例

```
OBSPY> fg seis              // 利用这个数据做个例子
OBSPY> lh a                 // 这个数据本身是标有 A 的，即初动到
     a = 1.046400e+01
OBSPY> apk v on             // apk，且打开震相验证
 CDV IPD0 81 329103824.49 // 台站名，KA 的值，后面两个不知道是什么
OBSPY> lh a ka f
      a = 1.049000e+01      // 新标记的初动到时，可以 p 看一下效果
     ka = IPD0              // 初动信息，I 表示急始，P 表示初动 P 波，
                            //  D 表示初动向下，0 不清楚
OBSPY> apk v off            // apk，关闭震相验证
 CDV -123 81 329103824.49 53897
OBSPY> lh a ka f
     a = 1.049000e+01       // 初始震相
     f = 5.389773e+06       // 事件结束，这里的 f 好像有问题？
```

12.6 arraymap

概要

利用 OBSPY 内存中的所有文件产生一个台阵或联合台阵的分布图

语法

```
ARRAYMAP [ARRAY|COARRAY]
```

输入

ARRAY 根据头段变量中的偏移 X、Y 值绘制台站分布
COARRAY　根据各台站之间的相对坐标绘制台站分布图

缺省值

```
arraymap array
```

头段数据

下面的两个头段变量必须使用 OBSPY 宏文件 `wrxyz` 或者与之功能相似的其他函数提前设定，所 有的偏移是相对于某个参考点的千米数。

- `USER7`：向东的偏移（X）
- `USER8`：向北的偏移（Y）

说明

不是很清楚这个命令的作用是什么，对于每个数据来说，需要用宏文件 `wrxyz` [1] 定义头段变量 `user7` 和 `user8`，然后才能利用该命令绘制出 arraymap，从命令的名字来理解，应该是绘制某个台站的台站分布图，理论上只需要台站的真实位置即可。不知这个究竟在什么场合要使用。

限制

在 `bbfk` 中允许的最多台站数

12.7　axes

概要

控制注释轴的位置

语法

```
AXES [ON|OFF|ONLY] [ALL] [TOP] [BOTTOM] [RIGHT] [LEFT]
```

输入

ON 显示列表中指定的注释轴，其他不变
OFF　不显示列表中指定的注释轴，其他不变
ONLY　只显示列表中指定的注释轴，其他的不显示
ALL　指定所有四个注释轴
TOP 绘图上部的 X 注释轴
BOTTOM 绘图下部的 X 注释轴
RIGHT 绘图右侧的 Y 注释轴
LEFT 绘图左侧的 Y 注释轴

缺省值

```
axes only bottom left
```

即只有下边和左边使用注释轴

[1] 位于 $OBSPYAUX/macros 目录中

说明

坐标轴可以绘制在一张图四边的任意一或多个边，有很多命令可以控制坐标轴长什么样。坐标 轴的注释间隔用 xdiv 命令设定（即隔多长显示一个数字），刻度标记的间距可以用 ticks 命令单独控制。

only 表示仅在后面列表中指定的边上使用注释轴，而 on 和 off 则表示仅对列表中的边打开或关闭注释轴，对其他不在列表中的边不起作用。

要获得自己想要的效果，使用 on 或者 off 时你必须要知道当前已经显示的轴有哪些，哪些是你想要打开或关闭的。这是一个有点容易弄错的问题，不如只使用 only 加上想要显示的轴更加简单一点。

示例

```
OBSPY> fg seis
OBSPY> p          //  看看 OBSPY 的默认设置，左边和底部有注释
OBSPY> axes on t // 打开顶部注释，左边和底部注释依然保留
OBSPY> p          // 看到的结果是只有顶部注释，没有左边和底部注释，
                  //  这里和说明中强调的不一样，应该是程序的 bug，
                  // 将 on 认为是 only 的简写了
OBSPY> axes on a // 打开所有注释轴
OBSPY> axes off b // 仅关闭底部注释轴(off 选项和说明是一致的)
OBSPY> axes only b // 仅显示底部注释轴
```

12.8 bandpass

概要

对数据文件使用无限脉冲带通滤波器

语法

```
BANDPASS [BUTTER|BESSEL|C1|C2] [CORNERS v1 v2] [NPOLES n] [PASSES n]
    [TRANBW v] [ATTEN v]
```

输入

BUTTER 应用 Butterworth 滤波器
BESSEL 应用 Bessel 滤波器
C1 应用 Chebyshev I 型滤波器
C2 应用 Chebyshev II 型滤波器
CORNERS v1 v2 设定拐角频率分别为 v1 和 v2，即频率通带为 v1-v2
NPOLES n 设置极数为 n，取值范围：1–10
PASSES n 设置通道数为 n，取值范围：1–2
TRANBW v 设置 Chebyshev 转换带宽为 v
ATTEN v 设置 Chebyshev 衰减因子为 v

缺省值

```
bandpass butter corner 0.1 0.4 npoles 2 passes 1 tranbw 0.3 atten 30
```

说明

OBSPY 提供了一系列无限脉冲滤波器（IIR）可对数据进行滤波，包括 Butterworth、Bessel、Chebyshev I 型和 Chebyshev II 型滤波器。这些递归数字滤波器都基于传统的模拟滤波器，将模拟滤波器经过双线性变换（一种可以保留模拟滤波器稳定性的数学变换）即可得到数字滤波器。

多数情况下 Butterworth 滤波器都是个不错的选择，因为 Butterworth 滤波器从通带到阻带的转变相对比较尖锐，且群延迟响应相对平缓，因而 Butterworth 滤波是 OBSPY 中默认的滤波器类型， 其 3 dB 点位于指定的截止频率处。

Bessel　滤波器适用于在不使用双通滤波的情况下要求线性相位的情形，其振幅响应不够理想，OBSPY 中的 Bessel 滤波器经过归一化以保证其 3 dB 点也位于指定的截止频率处。

两个 Chebyshev 滤波器用于需要通带与阻带间具有较为尖锐的转变的情况。尽管他们有较好的振幅响应，但是它们的群延迟响应是 OBSPY 所有滤波器中最差的。

使用这些滤波器时需要小心。首先，所有的递归滤波器都具有非线性相位响应，会导致滤波后 波形的频散，即波形出现畸变。若要求滤波后的波形保留原始相位，则要求滤波器具有零相位。对同 一个数据做正向和反向两次滤波，即相当于对数据做了一次零相位滤波，对应于命令中双通道的情 况。双通道滤波器的振幅响应，等于单通道滤波器的振幅响应的平方。与此同时，零相位滤波器具有 非因果的脉冲响应，会导致滤波后的信号在尖锐的跳起之前附加一个虚假的前驱信号。因而双通滤 波后的数据是不能用于拾取震相的跳起到时的，此时，双通滤波不是一个好选择。

其次，当滤波器的通带宽度比数据的 Nyquist 频率小很多时，滤波器会出现数值不稳定，当滤波器的级数增加时问题更加严重。对于需要极窄通带的情形，更可靠的方式是先对数据进行减采样， 然后对采样后的数据应用一个通带稍宽一些的滤波器，再将滤波后的数据插值回原始数据的采样率。当滤波通带的带宽降到 Nyquist 频率的百分之几以下时很有必要使用这种策略。

一般来说，随着极数的增加，滤波器从通带到阻带的过渡会更加尖锐。但是，使用极数过大是要 付出代价的。随着极数的增加，滤波器的群延迟也会变宽，进而导致滤波后的波形频散更加严重。因 而要仔细考虑是否真的需要超过 3 个或 4 个极点的滤波器。

Butterworth 和 Bessel 滤波器的设计特别简单，只需要指定截止频率和极数即可。Chebyshev 滤波器设计起来更复杂一点，除了截止频率和极点数目之外，还需要提供转换带宽以及阻带衰减因 子。转换带宽是滤波器通带和阻带之间的区域的宽度。由于双线性变换频率轴的非线性弯曲，递归 数字滤波器的转换带宽可能会比设计时指定的要小。在 OBSPY 里，模拟滤波器的截止频率在双线性 变换之后要做补偿以保证其满足设计要求。阻带边界同样也是不准确的。因此，若要精确指定通带 边界，在选定截止频率后必须对其进行补偿。阻带衰减因子是通带增益与阻带增益的比值。

示例

应用一个四极 Butterworth 滤波器，拐角频率为 2 Hz 和 5 Hz：

```
OBSPY> bp n 4 c 2 5
```

在此之后如果要应用一个二极双通具有相同频率的　Bessel：

```
OBSPY> bp n 2 be p 2
```

头段变量改变

depmin、depmax、depmen

12.9 bandrej

概要

应用一个无限脉冲带阻滤波器

语法

```
BANDREJ [BUTTER|BESSEL|C1|C2] [CORNERS v1 v2] [NPOLES n] [PASSES n]
    [TRANBW v] [ATTEN v]
```

输入

BUTTER 应用一个 Butterworth 滤波器
BESSEL 应用一个 Bessel 滤波器
C1 应用一个 Chebyshev I 型滤波器
C2 应用一个 Chebyshev II 滤波器
CORNERS v1 v2 设定拐角频率分别为 v1 和 v2，即频率通带为 v1–v2
NPOLES n 设置极数为 N，范围：1–10
PASSES n 设通道数为 N，范围：1–2
TRANBW v 设置 Chebyshev 转换带宽为 v
ATTEN v 设置 Chebyshev 衰减因子为 v

缺省值

```
bandrej butter corner 0.1 0.4 npoles 2 passes 1 tranbw 0.3 atten 30
```

说明

参见命令 bandpass 的说明

示例

应用一个四极 Butterworth 滤波器，拐角频率为 2 Hz 和 5 Hz：

```
OBSPY> br n 4 c 2 5
```

在此之后如果要应用一个二极双通具有相同频率的 Bessel：

```
OBSPY> br n 2 be p 2
```

头段变量改变

depmin、depmax、depmen

12.10 bbfk

概要

利用 OBSPY 内存中的所有文件计算宽频频率波数谱估计

语法

```
BBFK [FILTER] [NORMALIZE] [EPS v] [MLM|PDS] [EXP n] [WAWVENUMBER v]
    [SIZE m n] [LEVELS n] [DB] [TITLE text] [WRITE [ON|OFF fname]
    [SSQ n] [PRINT pname]]
```

输入

FILTER 使用最近一次 `filterdesign` 命令设计的带通滤波器

NORMALIZE 用 Capo 方法归一化协方差矩阵，如果各信号道的振幅差别比较大，这是一个好方法

EPS v 调整协方差矩阵的分量值，矩阵对角线的项是（1.0+EPS）的整数倍

MLM 在高分辨率估计中使用最大似然法

PDS 不采用最大似然法的功率谱密度

EXP n 波数谱增加的幂次

WAVENUMBER v 从中采样谱估计的波数目

SIZE m n 极坐标中等值线的尺寸：m 是方位角方向上的采样点数；n 是在波数方向上的采样点数。m、n 必须为偶数，而且其乘积最大限为 40000

LEVELS n 等值线间隔数

DB 以分贝为单位的对数坐标图形

TITLE text 图形标题

WRITE ON|OFF fname 是否计算二维等值线数据并写入磁盘（xyz 类型的 OBSPY 文件）。fname 是要写入的文件名或路径名。如果没有指定文件名，则默认为 BBFK

SSQ n 二维图的尺寸（取沿着正方形每个边的采样数据点），最大允许值为 200

缺省值

```
bbfk eps .01 pds exp 1 wvenumber 1.0 size 90 32 levels 11
    write off ssq 100
```

头段数据

分情况决定头段的信息：

- 若参考台站设置在 kuser1 中并且其对于所有文件是相同的，则所有文件的 user7 和 user8 都需要设置为偏移量
- 若所有文件台站纬度 stla 以及台站经度 stlo 都设置了，则偏移量通过这些经纬度计算，以第一个文件作为参考台站
- 若所有文件的 user7 和 user8 都设置了，则它们直接作为偏移量
- 若所有文件的事件纬度 evla 以及事件经度 evlo 都设置了，则他们用于计算偏移量，使用第一个台站作为参考台站

输出

polar 输出立即被绘制出（不保留），square 输出会写入到硬盘。FK 的峰值、反方位角以及波数将分别写入黑板变量 BBFK_AMP、BBFK_BAZIM 以及 BBFK_WVNBR。

错误消息

- 尺寸 m 或者 n 不是一个偶数
- 偏移量X、Y、Z未设置在头段变量user7、user8、user9中
- 未找到 filterdesign 得到的系数数据，或者滤波器类型不是 BP

限制

- 台站最多允许有 100 个
- 极性等值线的最大尺寸是 $m \times n = 40000$
- 二维等值线输出的最大尺寸是 $i = 200$

12.11 beam

概要

利用内存中的全部数据文件计算射线束

语法

```
BEAM [BEARING v] [VELOCITY v] [REFERENCE ON|OFF| lat lon [el]]
    [OFFSET REF|USER|STATION|EVENT|CASCADE] [EC anginc survel]
    [CENTER x y z] [WRITE fname]
```

输入

BEARING v 方位，由北算起的度数

VELOCITY v 速度，单位为公里每秒

REFERENCE lat lon el 参考点，打开 REFERENCE 选项并定义参考点，这样其他文件的偏移量以此而定。lat、lon、el 分别代表纬度、经度、深度（下为正）

REFERENCE ON|OFF 开或关 REFERENCE 选项

OFFSET REF 偏移量是相对于 REFERENCE 选项设置的参考点的。这要求开启 REFERENCE 选项

OFFSET USER 偏移量直接从 USER7、USER8 以及 USER9 中获取，分别代表纬度、经度以及海拔。这就要求所有文件的 USER7 和 USER8 必须定义。如果设置了 EC 选项，则 OFFSET USER 要求 USER9 必须被设置

OFFSET STATION 偏移量相对于第一个台站的位置，这要求所有文件的 STLA、STLO 必须定义

OFFSET EVENT 偏移量相对于第一个事件的位置，这要求所有文件的 EVLA、EVLO 必须定义

OFFSET CASCADE OBSPY 将会按照前面给出的顺序考虑决定偏移量的方法，并检查必要 的数据是否具备。它将使用第一个满足要求的方法

EC 高程校正。anginc：入射角，从 z 轴算起，单位为度（震源距离越远，入射角越小）；survel：表面介质速度（km/s）。

CENTER 用于计算射线束的中心台站。X 为距参考台站的东向偏移；Y 为距参考台站的北向偏移；Z 为距参考台站的向上偏移，其单位为 m；

WRITE fname 将射线束写入磁盘

缺省值

```
beam b 90 v 9.0 ec 33 6.0 c 0. 0. 0. w BEAM
```

说明

beam 不覆盖 OBSPY 内存中的数据，因而当变换方位和速度时这一操作可以重复执行。射线结果 写入到磁盘文件中，并且每次可以写到不同的文件。这个设计考虑到了用户的需求，即比较多次使 用这一命令的不同结果，以寻找最佳射线束的方位和速度。

头段数据

参见 bbfk 命令。

错误消息

CENTER 参数缺失偏移量，EC 参数需要正值

12.12 begindevices

概要

启动某个图像设备

语法

```
BEGINDEVICES  SGF|XWINDOWS
```

输入

SGF OBSPY 图像文件设备
XWINDOWS X Window 窗口显示系统

说明

该命令用于启动一种图像设备，之后的所有绘图都会被传送到该设备中，直到再次执行 begindevices 启动其它图像设备或 enddevices 结束该图像设备为止。

OBSPY 默认使用 xwindows 图像设备。具体用法参考 “图像保存” 一节的示例。

12.13 beginframe

概要

打开 frame，用于绘制组合图

语法

```
BEGINFRAME [PRINT [pname]]
```

输入

PRINT pname 当使用 PRINT 选项时，OBSPY 会把图形打印到名为 pname 的打印机，若 pname 未指定则打印到默认的打印机

说明

一般情况下，在每次使用绘图命令时，OBSPY 会对绘图设备执行刷新操作，以清除上一个绘图命 令绘制的图像，然后再显示本次命令绘制的图像，这样可以保证多次绘图命令绘制的图像不会重叠 在一起。

beginframe 命令会关闭绘图设备的自动刷新功能，直到 endframe 命令恢复自动刷新功能为止。在这两个命令中间执行的所有绘图命令所产生的图像将会叠加在一起，形成组合图。

通过这两个命令，并结合 xvport 和 yvport 定义每次绘图的 viewport，可以很容易地绘制出复杂的组合图。

关于如何绘制组合图以及这几个命令的使用，可以参考 组合图 一节。

12.14 beginwindow

概要

启动/切换至指定编号的 X 图形窗口

语法

```
BEGINWINDOW n
```

输入

n 要启用的绘图窗口的编号，目前 n 的取值为 1 到 10

缺省值

```
beginwindow 1
```

说明

现在的图形终端或工作站大多支持多窗口，即启动多个窗口，并在每个窗口中显示相同或不同 的图像。

window 命令可以控制每个 X 绘图窗口的位置和形状，而 beginwindow 则用于启用该绘图窗口，使得接下来所有的绘图命令的绘图效果都显示在该绘图窗口中，直到再次使用 beginwindow 命令切换到另一窗口。若你所选择的绘图窗口没有打开，则 beginwindow 会首先创建这个窗口。

需要注意的是，window 命令只在绘图窗口被创建之前起作用，即 window 命令是一个参数设定类命令。在多数系统上，均允许通过鼠标拖曳的方式动态改变这些窗口的大小。一般情况下，在动态 改变窗口大小或比例之后，当前窗口的绘图会自动重画以适应新窗口。

需要注意的是，这个命令没有与之对应的 endwindow。

12.15 benioff

概要

对数据使用 Benioff 滤波器

语法

```
BENIOFF
```

说明

1960 年左右，美国空军 VELA 计划中使用了一些可变磁阻短周期地震仪，这些仪器以加州理工的 Hugo Benioff 教授命名。其自然频率为 1 Hz，与一个电流计耦合在一起（自然频率为 5 Hz）。耦合因子的标称值为 0.01，其响应在 1 Hz 到 5 Hz 频段内近乎为平的。

该命令生成的滤波器是 Benioff 短周期地震仪的数字等效，用于将宽频带地震数据模拟成短周期系统。

该滤波器的响应函数如下图所示：

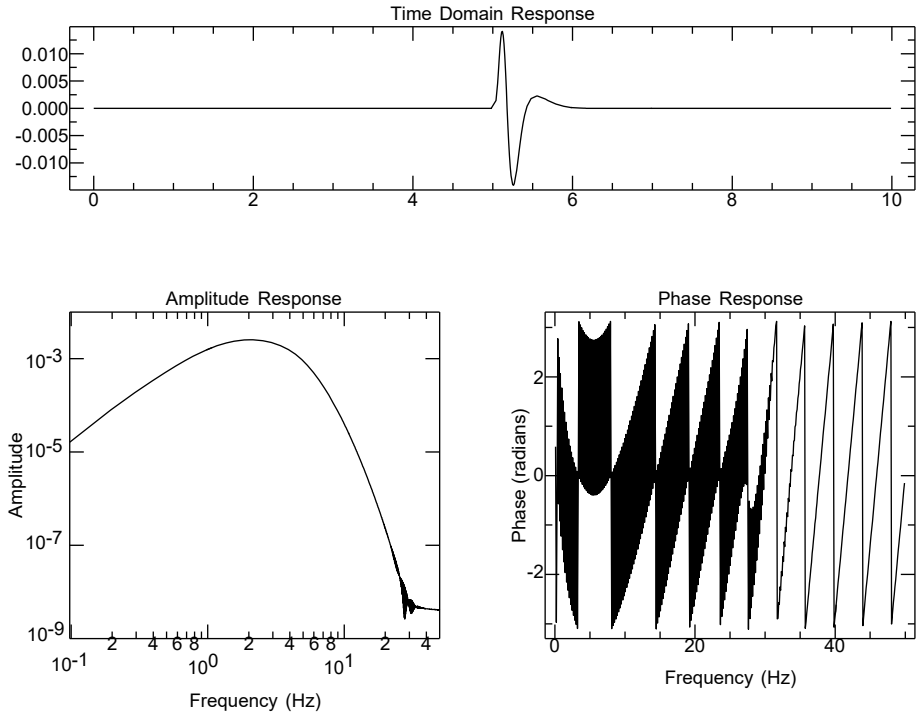

图 12.1: Benioff 滤波器的响应函数

头段变量

depmin、depmax、depmen

12.16 binoperr

概要

控制二元文件操作 addf、subf、mulf、divf 中的错误

语法

```
BINOPERR [NPTS FATAL|WARNING|IGNORE] [DELTA FATAL|WARNING|IGNORE]
```

该令可以简写为 boec

输入

NPTS　修改数据点数不相等的错误条件
DELTA　修改采样周期不相等的错误条件
FATAL　设置错误条件为“致命”

WARNING 设置错误条件为“警告”
IGNORE 设置错误条件为“忽略”

缺省值

```
binoperr npts fatal delta fatal
```

说明

对文件执行二元操作（addf、divf 等）时，OBSPY 会检测两个文件的数据点数和采样周期是否匹配。该命令可以控制 OBSPY 在遇到不匹配时该如何处理：

- 若设置错误条件为 fatal，则 OBSPY 在遇到错误时将停止执行当前命令，忽略当前行的其它其余命令，输出错误信息到终端并将控制权交还给用户
- 若设置错误条件为 warning，则遇到错误时会发送一个警告消息，程序内部尽可能纠正错误并继续执行
- 若设置错误条件为 ignore，则 OBSPY 会自动纠正错误并继续执行而不输出任何警告消息 在设置错误条件为 warning 或 igonre 的情况下，若要操作的两个数据文件的数据点数不匹配，OBSPY会使用数据点最少的那个文件的数据点数作为最终结果文件的数据点数，以保证正常操作；若要操作的两个文件采样周期不匹配，则无论是否使用了 newhdr on 选项，OBSPY都会使用第一个数据文件的采样周期作为结果文件的采样周期。

示例

假定 file1 有 1000 个数据点，file2 有 950 个数据点：

```
OBSPY> binoperr npts
fatal OBSPY> read file1
OBSPY> addf file2
ERROR: Header field mismatch: NPTS file1 file2
```

上例中由于数据点数不匹配导致文件加法未执行，假设你输入：

```
OBSPY> binoperr npts
warning OBSPY> addf file2
WARNING: Header field mismatch: NPTS file1 file2
```

则仅对文件的前 950 个数据点执行加法操作。

12.17 border

概要

控制图形四周边框的绘制

语法

```
BORDER [ON|OFF]
```

输入

ON 打开边界绘图
OFF 关闭边界绘图

缺省值

```
border off
```

说明

参考 “图像外观” 一节。

12.18　capf

概要

关闭目前打开的字符数字型震相拾取文件

语法

```
CAPF
```

12.19　chnhdr

概要

修改指定的头段变量的值

语法

```
CHNHDR [FILE n1 n2 ...] field v [field v ...] [ALLT v]
```

输入

FILE n1 n2　只修改内存中的指定文件的头段变量，n 为内存中文件的文件号
field v　OBSPY 头段变量名及其值 [1]
ALLT v　将所有已定义的时间相关头段变量的值加 v 秒，同时将参考时刻减去 v 秒

说明

关于值 v 的说明：

- 头段变量的类型和值的类型必须匹配；
- 对于有内部空格的字符串要用单引号括起来；
- 逻辑型头段变量的取值为 TRUE 或 FALSE，YES 或 NO 也可以接受
- 对于相对时间头段变量（B、E、O、A、F、Tn），v 可以是相对参考时刻的时间偏移量（浮点型），也可以使用绝对时刻的形式 GMT v1 v2 v3 v4 v5 v6，其中 v1、v2、v3、v4、v5、v6 是 GMT 年、一年的第一天、时、分、秒、毫秒。如果 v1 是两位整数，OBSPY 假定其为当前世纪，除非那 个时间是未来时间，那种情况下 OBSPY 假定是上个世纪，最好还是用 4 位整数表示年。
- 对于任意类型的头段变量，均可以设置其值为 undef，使头段变量未定义 该命令允许你修改指定的一个或多个文件的头段变量值。在未指定文件号的情况下，则对内

存中的所有文件进行操作。要将内存中修改后的头段覆盖磁盘文件的头段，需要使用 write 或 writehdr 命令，OBSPY 会对新值做有效性检查，不过你可以使用 listhdr 自己检查。

[1] 为了保证数据内部一致性，以下头段变量的值不可用该命令修改：nvhdr、npts、nwfid、norid 和 nevid

头段中用 6 个变量定义了参考时刻，这是 OBSPY 中唯一的绝对时刻，其它时刻都被转换成相对 于参考时刻的相对时间。可以使用 ALLT v 修改参考时刻以及相对时间。参考时间被减去了 v 秒，相 对时间被加上了 v 秒，这保证了数据的绝对时刻不发生改变。为了方便，你可以通过输入绝对时刻 而非相对时间来改变时间偏移变量的值。绝对时刻首先被转换为相对时间，然后再存入头段中。

示例

为了定义内存中所有文件的事件经纬度、事件名：

```
OBSPY> ch evla 34.3 evlo -
118.5 OBSPY> ch kevnm 'LA
```

为了定义第二、四个文件的事件经纬度、事件名：

```
OBSPY> ch file 2 4 EVLA 34.3 EVLO -118.5
OBSPY> ch file 2 4 KEVNM 'LA goes under'
```

设定初动到时为无定义状态：

```
OBSPY> ch a undef
```

假设你知道事件的 GMT 起始时间，你想要快速改变头段中所有的时间变量，使得发震时刻是 o 即参考时间为发震时刻，并且所有的相对时间根据这个时间去纠正相对值。 首先用 GMT 选项设置事件起始时间：

```
OBSPY> ch o GMT 1982 123 13 37 10 103
```

现在使用 listhdr 检查发震时刻 o 相对于当前参考时间的描述：

```
OBSPY> lh o
 o = 123.103
```

现在使用 ALLT 选项从所有的偏移时间中减去这个值，并加到参考时间上，同时需要改变描述参考时间类型的字段：

```
OBSPY> ch allt -123.103 iztype iO
```

注意这里的负号意味着从偏移时间中减去这个值。 更方便的做法是直接引用头段变量的值：

```
OBSPY> ch allt (0 - &1,o&) iztype IO
```

12.20 chpf

概要

关闭当前打开的 HYPO 震相拾取文件

语法

```
CHPF
```

说明

自动附加指令 card “10” 到被关闭的文件的结尾。

12.21 color

概要

控制彩色图形设备的颜色选项

语法

```
COLOR [ON|OFF|color] [INCREMENT [ON|OFF]] [SKELETON color]
    [BACKGROUND color] [LIST STANDARD|colorlist]
```

color是下面中的一个：

```
WHITE|RED|GREEN|YELLOW|BLUE|MAGENTA|CYAN|BLACK
```

这里有些参数在缩写的情况下可能会有歧义，请谨慎使用，而且 LIST 选项必须放在命令的最后

输入

ON 打开颜色选项单数不改变其他选项
OFF 关闭颜色选项
color 打开颜色选项并将数据设置为颜色 color
INCREMENT ON 每个数据文件绘出后，根据 colorlist 的顺序改变颜色
INCREMENT OFF 不改变数据颜色
SKELETON color 按照标准颜色名或颜色号修改边框颜色
BACKGROUND color 修改背景色为 color [1]
LIST colorlist 改变颜色列表，将数据颜色设置为列表中第一个颜色，并打开颜色开关
LIST STANDARD 将颜色列表设为标准列表，将数据颜色设置为列表中第一个颜色，并打开颜色开关

缺省值

```
color black increment off skeleton black background white
    list standard
```

说明

该命令控制设备的颜色属性，数据颜色是用于绘制这个数据文件的颜色。当一个数据文件绘制完毕后，数据颜色可以根据颜色列表自动改变。skeleton 颜色是用于绘制注释轴、标题、网格、框架的颜色。背景色是空框架在未绘制任何图形之前的颜色。

多数情况下你会选择标准颜色名，比如 red，这是与图形设备无关的。然而有时候你可能想选择一个非标准颜色，比如 aquamarine，这个可以将颜色表装入图形设备来实现。

这个表将特定的颜色、亮度、对比度等与一个数字联系起来，然后你就可以通过设定对应的整数值选择 aquamarine 作为你的绘图的一个部分的颜色，这个需要点工作量，可是如果你喜欢，这就值得。

[1] 白色背景与黑色线条对比强烈，可以考虑设置背景色为 cyan

如果你正在同一张图上绘制多个数据文件，通过 INCRMENT 选项可以使得不同数据有不同的颜色。标准颜色表顺序如下：

```
RED, GREEN, BLUE, YELLOW, CYAN, MAGENTA, BLACK
```

示例

为了使数据颜色从红色开始不断变换：

```
OBSPY> color red increment
```

为了设置数据颜色为红色，背景白色，蓝色边框：

```
OBSPY> color red background white skeleton blue
```

为了设置一个数据颜色不断变换，颜色列表为 red、white、blue，背景色为 aquamarine：

```
OBSPY> color red increment backgroud 47 list red white blue
```

上面的例子假设 aquamarine 是颜色表的 47 号。

12.22 comcor

概要

控制 OBSPY 的命令校正选项

语法

```
COMCOR [ON|OFF]
```

缺省值

```
comcor off
```

输入

ON 打开命令校正选项；
OFF 关闭命令校正选项；

缺省值

```
comcor off
```

说明

OBSPY 会检查你输入的每个命令的格式和内容。当 OBSPY 发现错误时，会给你发送一个错误消息 并告诉错误是原因及其位置。若开启了命令校正选项，OBSPY 将允许你修正这个命令然后 OBSPY 自动 重新执行它。若关闭校正，OBSPY 只是打印错误消息，将控制权返回给你。

12.23 contour

概要

利用内存中的数据绘制等值线图

语法

```
CONTOUR [ASPECT ON|OFF]
```

输入

ASPECT ON　打开视图比开关。当这个开关打开时，等值线图的视口将会调整保持数据中 Y 与 X 的比值

ASPECT OFF 关闭视图比开关，这时将使用整个视口

缺省值

```
contour aspect off
```

说明

这个命令用于绘制二维数组数据的等值线图，包括 `spectrogram` 命令的输出。这个文件操作的 OBSPY 文件必须 “XYZ” 类型的（OBSPY 头段中 `IFTYPE` 为 `IXYZ`）。有些命令可以控制数据显示的方式：

- `zlevels` 控制等值线的数目以及间隔
- `zlines` 控制等值线的线型
- `zlabels` 控制等值线标签
- `zticks` 控制方向标记
- `zcolors` 控制等值线颜色

根据 `contour` 选项的不同，有两种不同的绘制等值线算法。一种快速扫描方法用于既不选择实线型也没有时标和标识的情况。另一种慢一点的方法，在绘图之前要组合全部的线段。你可以使用 快速扫描方法粗看你的数据，然后选择其他选项绘制最终图形。

示例

参见 “等值线图” 一节。

头段变量改变

要求：iftype（为 “IXYZ”）、nxsize、nysize

使用：xminimum、xmaximum、yminimum、ymaximum

12.24　convert

概要

实现数据文件格式的转换

语法

```
CONVERT [FROM] [format] infile [TO [format] outfile]|[OVER [format]]
```

其中 `format` 可以为 `OBSPY|ALPHA`

输入

infile 输入文件名
outfile 输出文件名
OVER　覆盖输入文件

OBSPY OBSPY 格式二进制文件
ALPHA OBSPY 字母数字型文件

缺省值

```
convert from obspy infile over obspy
```

说明

该命令将单个文件从一种格式转换为另一种格式。该命令已经逐渐被 `read` 和 `write` 命令所取代，`convert` 命令已经不再需要，保留该命令只是为了兼容性考虑。

12.25 convolve

概要

计算主信号与内存中所有信号的卷积

语法

```
CONVOLVE [MASTER name|n] [NUMBER n] [LENGTH ON|OFF|v]
    [TYPE  RECTANGLE|HAMMING|HANNING|COSINE|TRIANGLE]
```

输入

MASTER name|n 通过文件名或文件号指定某文件为主文件，内存中的所有文件将与主文件进行卷积
NUMBER n 设置卷积窗的数目
LENGTH ON 打开/关闭固定窗长选项开关
LENGTH v 打开固定窗长选项开关，并设置窗长度为 v 秒
TYPE RECTANGLE 对每个窗应用一个矩形函数，这等价于不对窗加上函数
TYPE HAMMING|HANNING|COSINE|TRIANGlE 对每个窗应用 XX 函数

缺省值

```
convolve master 1 number 1 length off type rectangle
```

说明

该命令允许用户指定一个主信号，并将主信号与自己及其它信号做卷积。如果内存中有 N 个 OBSPY 文件，则输出文件为 N 个文件与主文件卷积的结果。卷积公式如下，

$$CV(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau$$

需要注意的是，实际代码中该卷积在频率域完成，且没有进行归一化。内存中所有的信号需要有相同的 `delta`。

该算法假定所有的时间序列都是因果的，因此如果你想要将信号卷积上一个 boxcar 函数（非因果低通滤波器，可用于平滑合成波形中的尖峰），输出信号将出现半个 boxcar 宽度的时移。 **头段**

变量

depmin、depmax、depmen

12.26　copyhdr

概要

从内存中的一个文件复制头段变量给其他所有文件

语法

```
COPYHDR [FROM name|n] hdrlist
```

输入

FROM name　从内存中文件名为 `name` 的文件中复制头段变量
FROM n　从内存中第 n 个文件中复制头段变量
hdrlist　要复制的头段变量列表

缺省值

```
copyhdr from 1
```

说明

该命令允许你从内存中的一个文件复制任意头段变量值到内存中其他所有文件，让数据有相同 的头段使得数据易于处理。

示例

假设你使用 `ppk` 命令在文件 FILE1 中标记了多个时间，并将其储存到头段变量 `T3` 和 `T4` 中。为了将这些时间标记复制到到 FILE2 和 FILE3 中：

```
OBSPY> r FILE1
OBSPY> ppk
OBSPY> ... use cursor to mark times T3 and
T4. OBSPY> r more FILE2 FILE3
OBSPY> copyhdr from 1 T3 T4
```

假设你读取了很多文件，想要复制文件 ABC 中的头段变量 `evla` 和 `evlo` 到其他所有文件中去，这时使用文件名而非数字会更简单：

```
OBSPY> copyhdr from abc stla stlo
```

12.27　correlate

概要

计算自相关和互相关函数

语法

```
CORRELATE [MASTER name|n] [NUMBER n] [LENGTH ON|OFF|v] [NORMALIZED]
    [TYPE    RECTANGLE|HAMMING|HANNING|COSINE|TRIANGLE]
```

输入

MASTER name|n 通过文件名或文件号指定主文件，所有文件将与此文件做相关
NUMBER n 设置相关窗的个数
LENGTH ON|OFF 打开/关闭固定窗长选项开关
LENGTH v 打开固定窗长选项开关，并将窗长度设置为 v 秒
NORMALIZED 对相关结果进行归一化
TYPE RECTANGLE 给每个窗应用矩形函数，其等价于不对窗加函数
TYPE HAMMING|HANNING 对每个窗应用 Hamming/Hanning 函数
TYPE COSINE 对每个窗的前后 10% 的数据点应用余弦函数
TYPE TRIANGLE 对每个窗应用三角函数

缺省值

```
correlate master 1 number 1 length off type rectangle
```

说明

该命令允许用户指定内存中的某个信号为主信号，并将主信号与内存中的所有信号（包括主信 号自身）进行相关。主信号与自身计算得到自相关函数，与其他信号计算得到互相关函数。

两个信号之间的互相关函数定义如下：

$$Cor(t) = \int_{-\infty}^{\infty} f^*(\tau)g(t+\tau)d\tau$$

互相关函数的计算可以在时间域或频率域完成，该命令在频率域计算信号间的相关函数。 该命令的窗特性允许你计算对多个窗计算平均相关函数，其中窗的数目以及窗函数均可以指定。当窗特性被打开时，会将信号划分为 n 个固定长度的窗，计算每个窗的互相关函数，然后将所有的互相关函数做平均、截取到与原信号相同的数据长度，并替换内存中的原始数据文件。 当窗长度（LENGTH 选项）以及窗数目（NUMBER 选项）超过文件中的数据点数时，会自动计算窗之间的重叠。缺省情况下，此窗特性是关闭的。

使用归一化选项，会对相关函数做归一化，最终得到的结果位于 -1 到 1 之间，由此可以得到常用的互相关系数。

示例

以内存中第三个文件为主文件计算互相关函数：

```
OBSPY> r file1 file2
file3 OBSPY> cor master
```

也可以通过文件名来指定主文件。

假设有两个数据文件，每个包含 1000 个噪声数据。将数据划分为无重叠的 10 个窗，每个窗包含 100 个数据点，且对窗应用 hanning 函数，并计算 10 个窗的平均相关函数：

```
OBSPY> r file1 file2
OBSPY> cor type hanning number 10
```

为了使窗之间有 20% 的混叠，可以设置窗长度为 120 个数据点。假设数据采样周期为 0.025（即每秒 40 个采样点），则窗长为 3 秒：

```
OBSPY> r file1 file2
OBSPY> cor type hanning number 10 length 3.0
```

下面的例子计算了两个数据之间的归一化互相关函数，并从中提取出了互相关系数：

```
OBSPY> r file1
file2                                                    // 归一化互相关
OBSPY> setbb cc (max &2,depmax (abs &2,depmin)) // 取互相关函数的极
值
```

头段变量

depmin、depmax、depmen

12.28　cut

概要

定义要读入的数据窗

语法

```
CUT [ON|OFF|pdw|SIGNAL]
```

输入

pdw 打开截窗选项并修改 pdw
ON 打开截窗选项但不改变 pdw
OFF 关闭截窗选项
SIGNAL 等效于设置 pdw 为 A -1 F 1，即 a 前一秒到 f 后一秒的数据窗

缺省值

```
cut off
```

说明

cut 命令仅仅设置了要读取的时间窗选项，并不对内存中的数据进行截取。因而，若要该命令 起作用，需要在 cut 命令设置时间窗后使用 read 命令。与此相反，cutim 命令会在命令执行时直接对内存中的数据进行截取。

若截窗选项为关，则读取整个文件；若截窗选项为开，则只读取由 pdw 定义的部分。 如果你想对一组有不同参考时刻的文件使用同样的时间窗，必须在执行 cut 前先使用 synchronize 命令使所有文件具有相同的参考时刻。synchronize 命令修改了文件的头段使得所有文件具有相同的参考时刻，并调整所有相对时间。因而，你需要先读取所有文件，执行 synchronize 命令，使用 writehdr 将修改后的头段写入到磁盘文件中，然后再执行 cut 命令，并读取数据，这样才能得到正确的结果。

示例

下面的宏文件展示了 cut 命令的一些常见用法。

```
fg seismo
wrie
seismo.obspy
echo on
* no cutting
lh b e a kztime
read
seismo.obspy
* begin to end---same as not cutting.
cut B E
read
lh b e a kztime
read
seismo.obspy
* First 3 secs of the file
cut B 0 3
read
lh b e a kztime
read
seismo.obspy
* First 100 points of the file.
cut B N 100
read
lh b e a delta kztime
read seismo.obspy
* From 0.5 secs before to 3 secs after first arrival
cut A -0.5 3
read
lh b e a kztime
read
seismo.obspy
* From 19 to 15 secs relative to zero (DIFFERENT FROM CUTIM).
cut 10 15
read
lh b e a kztime
read
seismo.obspy
* First 3 secs of the file and next 3 sec
cut b 0 3
read
write tmp.1
read
seismo.obspy
```

当要截取的窗超过了文件的时间范围时，可以使用 `cuterr` 命令的 `FILLZ` 选项，在文件的开始或结尾处补 0，再读入内存。

```
OBSPY> r
N11A.lhz
```

```
    npts = 3101

OBSPY> cuterr fillz; cut b n
4096 OBSPY> r
OBSPY> lh npts
    npts = 4096
```

限制

目前不支持非等间隔文件或谱文件的截断。该命令对 ASCII 格式的 OBSPY 文件无效。

12.29　cuterr

概要

控制坏的截窗参数引起的错误

语法

```
CUTERR  FATAL|USEBE|FILLZ
```

输入

FATAL 将截窗错误设置为致命
USEBE　将坏的起始和结束截窗参数设置为文件开始和文件结束
FILLZ　在文件开始时间之前或文件结束时间之后填补适当数目的 0 以弥补坏的截窗参数

缺省值

对于 SSS 子程序默认值为 FILLZ，其他的默认值为 USEBE

说明

这个命令控制由于坏的截窗参数引起的错误条件。可以将这些错误定义为致命错误。若裁剪参 数的起始值或结束值在文件头段中未定义，则可以选择为 USEBE。若定义了要截取的时间窗但是其 截窗起始值小于文件起始值或者截窗参数结束值大于文件结束值，则可以分别用文件起始结束值代 替截窗参数，或者也可以使用 FILLZ 在文件前后补适当的 0。

示例

假设文件 FILE1 起始时间为 B=25 s，初动到时 A=40 s，采样率为 0.01 s。

```
OBSPY> cut a
20 e OBSPY>
```

截窗起始值为 20s，产生了一个错误条件。在 USEBE 模式下，截窗起始值将替换为 25s(即 B)。在 FILLZ 模式下，在数据之前将插入 500 个零值（5 秒钟，每秒 100 个点），截窗起始值保持为 20s。

12.30　cutim

概要

截取内存中的文件

语法

```
CUTIM pdw [pdw ... ]
```

输入

pdw 要截取的时间窗。参考 pdw

缺省值

如果起始或结束 offset 省略则认为其为 0，如果起始参考值省略则认为其为 z，如果结束参考值省略则认为其值与起始参考值相同。

说明

cut 命令设置截窗选项，仅对即将读取的文件进行截窗，而对内存中的数据没有效果。cutim 则在这个命令给出的时候对内存中的数据进行截窗操作。

用户可以用 read 读入文件，然后用 cutim 对内存中的文件直接进行截窗。cutim 也允许使用多个截取区间，用户可以读三个文件到内存，然后使用有 4 个截取区间的 cutim 命令，最终内存中将得到 12 个文件。

示例

下面的宏文件展示了 cutim 命令的常见用法：

```
fg seismo
echo on
* no cutting
lh b e a kztime
* begin to end---same as not cutting.
cutim B E
lh b e a kztime
fg seismo
* First 3 secs of the file.
cutim B 0 3
lh b e a kztime
fg seismo
* From 0.5 secs before to 3 secs after first arrival
cutim A -0.5 3
lh b e a kztime
fg seismo
* From 0.5 to 5 secs relative to disk file start.
cutim 0.5 5
lh b e a kztime
fg seismo
* First 3 secs of the file and next 3 sec
cutim b 0 3 b 3 6
lh b e a kztime
p1
```

限制

目前不支持截取非等间隔数据或谱文件

BUG

该命令似乎有问题，执行该命令时，似乎会删除内存中的全部波形数据，请勿使用

12.31 datagen

概要

产生样本波形数据并储存在内存中

语法

```
DATAGEN [MORE] [SUB LOCAL|REGIONAL|TELESEIS] [filelist]
```

输入

MORE 将新生成的样本数据放在内存中旧文件后。若省略此项，则新数据将替代内存中的旧数据

SUB LOCAL|REGIONAL|TELESEIS 要生成的数据的子类型，每个子类型对应不同的样本数据

filelist 样本数据文件列表。每个子类型可选的文件列表在后面给出

说明

OBSPY 提供了一些样本地震数据以供用户学习时使用，该命令将读取一个或多个样本地震数据 到内存中。事实上，该命令与 `read` 命令类似，只是该命令是从特殊的数据目录（`$OBSPYAUX/datagen`） 中读取文件。

该命令提供了三种子类型，分别是 `local`、`regional` 和 `teleseis`，分别对应近震、区域地震和远震。不同的子类型，其所包含的数据文件也不同。

LOCAL

该 local 事件发生在加州的 Livermore Valley，是一个很小的无感地震（ML=1.6），其被 Livermore Local Seismic Network (LLSN) 所记录。

LLSN 拥有一系列垂直分量和三分量台站。该数据集中包含了 9 个三分量台站的数据。数据时长 40 秒，每秒 100 个采样点。台站信息、事件信息、p 波及尾波到时都包含在头段中，这些文件包括：

```
cal.z,  cal.n,  cal.e
cao.z, cao.n, cao.e
cda.z, cda.n, cda.e
cdv.z, cdv.n, cdv.e
cmn.z, cmn.n, cmn.e
cps.z, cps.n, cps.e
cva.z, cva.n, cva.e
cvl.z,  cvl.n,  cvl.e
cvy.z, cvy.n, cvy.e
```

REGIONAL

该区域地震发生在 Nevada，被 Digital Seismic Network (DSS) 所记录。DSS 包含了美国西部的四个宽频带三分量台站。数据包含了从发震前 5 秒开始的为 300 秒地震数据，每秒含 40 个采样点，文件名为：

```
elk.z,  elk.n,  elk.e
lac.z,  lac.n,  lac.e
knb.z, knb.n, knb.e
mnv.z, mnv.n, mnv.e
```

TELESEIS

该远震事件于 1984 年 9 月 10 日发生在加州北海岸 Eureka 附近，其为中等偏大的地震（ML 6.6、MB 6.1、MS 6.7），多地有感。该数据集中包含了 Regional Seismic Test Network (RSTN) 的 5 个台站的中等周期和长周期数据（其中 cpk 台站的数据无法获取，sdk 台站的长周期数据被截断）。 这个数据集的数据时长 1600 秒，长周期数据每秒 1 个采样点，中等周期数据每秒 4 个采样点。文 件包括：

```
ntkl.z, ntkl.n, ntkl.e, ntkm.z, ntkm.n, ntkm.e
nykl.z, nykl.n, nykl.e, nykm.z, nykm.n, nykm.e
onkl.z, onkl.n, onkl.e, onkm.z, onkm.n, onkm.e
sdkl.z, sdkl.n, sdkl.e, sdkm.z, sdkm.n, sdkm.e
```

示例

下面的示例展示了一些基本的用法：

```
OBSPY> dg sub l cal.z // 单个近震 Z 分量数据
OBSPY> dg sub r *.z // 区域地震多台 Z 分量数
据 OBSPY> dg sub t sdkl.? // 远震的单台三分
```

生成一堆波形数据，并保存数据到磁盘中：

```
OBSPY> dg sub l cdv.e cdv.n
cdv.z OBSPY> w cdv.e cdv.n
```

在写文件时，需要手动指定文件名列表，当文件很多时，就会变得很麻烦。可以利用 `write` 命令的语法简化这一命令：

```
OBSPY> dg sub local *
OBSPY> w delete /opt/obspy/aux/datagen/local/
```

`delete` 选项的作用是从原始的文件名中删去 `/opt/Obspy/aux/datagen/local/`，只留下文件名。

12.32 decimate

概要

对数据做减采样

语法

```
DECIMATE [n] [FILTER ON|OFF]
```

输入

n 设置减采样因子为 n，即每 n 个点中取一个点，n 取值范围为 2 到 7
FILTER ON|OFF 打开/关闭抗混叠 FIR 滤波器

缺省值

```
decimate 2 filter on
```

说明

此命令用于对内存中的数据进行减采样，减采样因子 n 表示从每 n 个数据点中取一个点，因而经过减采样之后的数据点数近似为 *npts*/*n* 个。减采样因子的允许取值为 2 到 7，为了得到更大的减采样因子，可以多次执行该命令。

根据采样定理：

> 如果信号是带限的，并且采样频率大于信号带宽的2倍，那么，原来的连续信号可以从 采样样本中完全重建出来。

若不满足此采样条件，采样后信号的频率就会重叠，即高于采样频率一半的频率成分将被重建成低 于采样频率一半的信号。这种频谱重叠导致的失真称为混淆效应。

该命令提供了一个可选的 FIR 滤波器对数据进行低通滤波，以避免减采样过程中可能出现的混淆效应。可用的的 FIR 滤波器的具体参数位于 `$OBSPYHOME/aux/fir/decn` 中，这些滤波器是经过 精心设计的，保留了相位信息。使用 FIR 滤波器有时会在数据的两端产生瞬时跳变，因而减采样的 结果需要在图形界面下人工审核。只有当高频响应的准确度不重要的时候（比如绘图时），才可以关 闭 FIR 滤波器。

示例

对数据减采样 42 倍：

```
OBSPY> r file1
OBSPY> decimate 7 // 减采样因子为 7 时 FIR 滤波器偶尔不稳定，慎用！
OBSPY> decimate 6
```

头段变量

npts、delta、e、depmin、depmax、depmen

12.33　deletechannel

概要

从内存中的文件列表中删除一个或多个文件

语法

```
DELETECHANNEL ALL
```

或

```
DELETECHANNEL fname|fno|range [fname|fno|range ...]
```

输入

ALL 删除内存中全部文件
fname 要删除文件的文件名
filenumber 要删除文件的文件号。第一个文件的文件号是 1

range 要删除文件的文件号范围，范围用破折号分开，如 11-20

示例

```
dc 3 5                          // 删除第 3、5 个文件
dc SO01.sz SO02.sz              // 删除这些名字的文件
dc 11-20                        // 删除第 11 至 20 个文
dc 3 5 11-20 SO01.sz SO02.sz // 删除上面的全部文件
```

12.34 dif

概要

对数据进行微分

语法

```
DIF [TWO|THREE|FIVE]
```

输入

TWO 使用两点差分算子
THREE 使用三点差分算子
FIVE 使用五点差分算子

缺省值

```
dif two
```

说明

该命令通过对数据应用差分算法实现数据的微分操作，要求数据必须是等间隔采样的时间序列 文件。微分会使位移量变成速度量，速度量变成加速度量，因而微分的同时会修改头段变量 idep 的 值。

两点差分算法：

$$Out(j) = \frac{Data(j+1) - Data(j)}{\Delta}$$

两点差分算子不是中心差分算法。此算法的最后一个输出值是未定义的，OBSPY 的处理方式是：令数据点数减 1，文件起始时间 B 增加半个采样周期。

三点（中心两点）差分算法：

$$Out(j) = \frac{1}{2}\frac{Data(j+1) - Data(j-1)}{\Delta}$$

此算法输出值的第一个和最后一个是未定义的，OBSPY 将数据点数减去 2，并将文件起始时间 B 增加 一个采样周期。

五点（中心四点）差分算法：

$$Out(j) = \frac{2}{3}\frac{Data(j+1) - Data(j-1)}{\Delta} - \frac{1}{12}\frac{Data(j+2) - Data(j-2)}{\Delta}$$

此算法输出值的首尾各两个点是未定义的，OBSPY 使用三点差分算符计算第二个和倒数第二个点的值，并将数据点数减 2，将 B 增加一个采样周期。

头段变量改变

npts、b、e、depmin、depmax、depmen、idep

12.35　div

概要

数据文件的每个数据点除以同一个常数

语法

```
DIV [v1 [v2 ... vn] ]
```

输入

v1 第一个文件要除以的常数
v2 第二个文件要除以的常数
vn 第 n 个文件要除以的常数

缺省值

```
div 1.0
```

说明

参见 add 的相关说明。

头段变量改变

depmin、depmax、depmen

12.36　divf

概要

使内存中的一组数据除以另一组数据

语法

```
DIVF [NEWHDR [ON|OFF]] filelist
```

输入

NEWHDR ON|OFF 指定新生成的文件使用哪个文件的头段。OFF 表示使用内存中原文件的头段区，ON 表示使用 filelist 中文件的头段区。缺省值为 OFF
filelist OBSPY 二进制文件列表

说明

参见 addf 命令的相关说明。

头段变量

depmin、depmax、depmen、npts、delta

12.37 divomega

概要

在频率域进行积分操作

语法

```
DIVOMEGA
```

说明

根据傅里叶变换的微分性质：

$$\mathrm{F}[f'(x)] = i\omega \mathrm{F}[f(x)]$$

其中 $\omega = 2\pi f$，即函数积分在频率域可以用简单的除法来表示。 该命令仅可对谱文件进行操作，谱文件可以是振幅-相位型或实部-虚部型。对于正常的谱数据来说还是很方便的，但不适用于谱跨越几个量级的数据。比如，假设你使用 `dif` 命令对数据进行预白化，然后对数据进行 Fourier 变换，用此命令在频率域积分可以去除时间域微分的效应。

若为振幅-相位型：

$$\mathrm{F}[f(x)] = \frac{\mathrm{F}[f'(x)]}{i\omega} = \frac{A(\omega)e^{\theta(\omega)}}{i\omega} = \frac{A(\omega)}{\omega}e^{\theta(\omega)-\pi/2}$$

若为实部-虚部型：

$$\mathrm{F}[f(x)] = \frac{\mathrm{F}[f'(x)]}{i\omega} = \frac{a(\omega)+ib(\omega)}{i\omega} = \frac{b(\omega)}{\omega} - i\frac{a(\omega)}{\omega}$$

在零频部分，直接设置其值为 0 比避免除以 0 的问题。

示例

```
OBSPY> read file1          // 微分预白化
OBSPY> dif                 // FFT
OBSPY> fft                 // 积分
```

头段变量

depmin、depmax、depmen

12.38 echo

概要

控制输入输出回显到终端

语法

```
ECHO  ON|OFF  ERRORS|WARNINGS|OUTPUT|COMMANDS|MACROS|PROCESS
```

输入

ON|OFF　打开/关闭列出的项的回显选项
ERRORS　命令执行过程中生成的错误信息
WARNINGS　命令执行过程中生成的警告信息
OUTPUT　命令执行过程中生成的输出信息
COMMANDS 终端键入的原始命令
MACROS　宏文件中出现的原始命令
PROCESSED　经过处理后的终端命令或宏文件命令，包括宏参数、黑板变量、头段变量、内联函数的计算和代入

缺省值

```
echo on errors warnings output off commands macros processed
```

说明

该命令控制 OBSPY 输入输出流中哪一类要被回显到终端或屏幕。 输出分为三大类：错误消息、警告消息、输出消息；输入也分为三大类：终端键入的命令、宏文件中执行的命令以及处理后的命令。处理后的命令指所有的宏参数、黑板变量、头段变量、内联函数首先被计算，并代入到命令中而形成的命令。你可以分别控制这些类的回显。 当在终端键入命令时，操作系统一般会显示用户键入的每个字符，因此该命令在交互式会话中没有太大作用。设置显示宏命令以及处理后的命令在调试宏文件时会很有用。

12.39　enddevices

概要

结束某个图像设备

语法

```
ENDDEVICES [ALL|SGF|XWINDOWS]
```

输入

ALL 关闭所有图像设备
SGF OBSPY 图形文件设备
XWINDOWS X Window 图像窗口系统

说明

参见命令 begindevices 的说明。

12.40　endframe

概要

关闭 frame

语法

```
ENDFRAME
```

说明

参见 beginframe 命令的相关说明。

12.41 envelope

概要

利用 Hilbert 变换计算包络函数

语法

```
ENVELOPE
```

说明

该命令用于计算内存中数据的包络函数。

原始信号为 $s(t)$，对其做 Hilbert 变换得到 $H(t)$，将这两个信号合并起来构成复信号

$$C(t) = s(t) + i * H(t)$$

复信号不仅可以用 “实部-虚部” 形式表示，也可以用 “振幅-相位” 形式表示：

$$C(t) = A(t)e^{j\Phi(t)}$$

其中 $A(t)$ 即为包络函数，其可以进一步表示为

$$A(t) = \sqrt{s(t)^2 + H(t)^2}$$

和 hilbert 一样，数据点数不得少于 201，且超长周期的数据需要在处理之前进行减采样。

头段变量

depmin、depmax、depmen

12.42 erase

概要

清除图形显示区域

语法

```
ERASE
```

说明

只有 OBSPY 知道你在使用的图形设备的情况下这个命令才可以工作。而且这只有在你已经进行 了一些绘图操作之后才可以使用。这个命令对于没有清除屏幕键的 ADM 终端很有必要。特别是你 想在发送大量文本之前清除屏幕时，这个命令在命令文件中非常有用。

12.43　evaluate

概要

对简单算术表达式求值

语法

```
EVALUATE [TO TERM|name] [v] op v [op v ...]
```

其中 op 可以取下面中的一个：

```
ADD|SUBTRACT|MULTIPLY|DIVIDE|POWER|SQRT|EXP|ALOG|ALOG10|
SIN|ASIN|COS|ACOS|TAN|ATAN|EQ|NE|LE|GE|LT|GT
```

输入

TO TERM　结果写入终端
TO name　结果写入黑板变量 name
v　浮点数或整数。OBSPY 中所有的运算都是浮点运算，整数会首先转换为浮点型
op　算术或逻辑操作符

其他形式

- + 代替 ADD
- - 代替 SUBTRACT
- * 代替 MULTIPLY
- / 代替 DIVIDE
- ** 代替 POWER

缺省值

```
evaluate to term 1. * 1.
```

说明

这个命令允许你对算术或逻辑表达式求值。算术表达式可以是包含多个操作符的复合表达式，在这种情况下表达式由左向右计算，不支持嵌套功能。逻辑表达式只能包含一个操作符。计算结果可以写入用户终端或者指定的黑板变量。你可以通过 getbb 命令使用该黑板变量的值。

示例

一个简单的例子：

```
OBSPY> eval
 2*3 6
OBSPY> eval tan
45 1.61978
```

下面将一个以度为单位的角度转换为弧度并计算其正切值：

```
OBSPY> eval
 45*pi/180
 0.785398
OBSPY> eval tan
```

下面将计算的结果保存到黑板变量：

```
OBSPY> evaluate to temp1
45*pi/180 OBSPY> evaluate
tan %temp1%
```

12.44 exp

概要

对每个数据点取其指数（e^y ）

语法

```
EXP
```

头段变量

depmin、depmax、depmen

12.45 exp10

概要

对每个数据点取以 10 为底的指数（10^y）

语法

```
EXP10
```

头段变量

depmin、depmax、depmen

12.46 fft

概要

对数据做快速离散傅立叶变换

语法

```
FFT [WOMEAN|WMEAN] [RLIM|AMPH]
```

输入

WOMEAN 变换前先去除均值
WMEAN 变换前不去除均值
RLIM 输出为实部-虚部格式
AMPH 输出为振幅-相位格式

缺省值

```
fft wmean amph
```

说明

该命令对数据进行离散傅立叶变换。为了使用快速傅立叶变换算法，在进行变换之前，需要对 数据文件进行补零以保证数据点数为 2 的整数次幂，比如 1000 个点的时间序列文件会被补零至 1024 个点，且头段变量 npts 也会被相应修改。

进行离散傅立叶变换之后，头段变量 b 为谱文件的起始频率，其值为 0；delta 为谱文件的采样频率，取值为 1/(delta*npts)；e 为谱文件的结束频率。b、e、npts 和 delta 的原值被保存到 sb、se、nsnpts 和 sdelta，这些值在做反傅立叶变换时会用到。

傅里叶变换得到的谱数据可以是振幅-相位格式或实部-虚部格式。头段变量 iftype 会告诉你谱文件是哪种格式。

由于实序列的离散傅立叶变换的结果具有“共轭对称性”，因而在使用 plotsp 绘制谱文件时只显示一半的数据点数。

示例

```
OBSPY> fg seis
OBSPY> lh b e delta npts
          iftype

          b = 9.459999e+00
          e = 1.945000e+01
      delta = 1.000000e-02
       npts = 1000
     iftype = TIME SERIES FILE
OBSPY> fft
 DC level after DFT is -0.98547
OBSPY> lh b e delta npts
iftype                              // b 值为

          b = 0.000000e+00          //  delta=1/(1024*0.01)
          e = 5.000000e+01          // 1000 -> 1024
     iftype = SPECTRAL FILE-AMPL/PHASE
OBSPY> lh sb sdelta nsnpts          // 保留原值

         sb = 9.459999e+00
     sdelta = 1.000000e-02
     nsnpts = 1000
```

头段变量

b、e、delta、npts、sb、se、nsnpts、sdelta

限制

离散傅立叶变换所允许的最大数据点数为 $2^{24} = 16777216$ 个。

12.47 fileid

概要

控制绘图时文件 ID 的显示

语法

```
FILEID [ON|OFF] [TYPE DEFAULT|NAME|LIST hdrlist]
    [LOCATION UR|UL|LR|LL] [FORMAT EQUALS|COLONS|NONAMES]
```

输入

ON 显示文件 id，不改变文件 id 类型或位置
OFF 不显示文件 id
TYPE DEFAULT 设置文件 id 为默认类型
TYPE NAME 使用文件名作为文件 id
TYPE LIST hdrlist 定义在文件 id 中显示的头段列表
LOCATION UR|UL|LR|LL 文件 id 的显示位置，分别表示右上角、左上角、右下角、左下角
FORMAT EQUALS 格式为 `variable=value`
FORMAT COLON 格式为 `variable:value`
FORMAT NONAMES 格式只包含头段值

缺省值

```
fileid on type default location ur format nonames
```

说明

文件 ID 用于标识绘图的内容。默认的文件 ID 包括事件名、台站名、分量、参考日期及时间。如果需要也可以使用文件名代替默认的文件 id，或者根据头段变量定义一个特殊的文件 ID，这个 ID 最多可以由 10 个 OBSPY 头段变量构成。文件 ID 的位置以及格式也可以修改。

示例

将文件名放在左上角：

```
OBSPY> fileid location ul type name
```

定义一个特殊的文件 id，包含台站分量、经纬度：

```
OBSPY> fileid type list kstcmp stla stlo
```

文件 id 为头段名后加一个冒号：

```
OBSPY> fileid format colon
```

12.48 filenumber

概要

控制绘图时文件号的显示

语法

```
FILENUMBER [ON|OFF]
```

缺省值

```
filenumber off
```

说明

当该选项为开时，绘图时会在右下角显示文件号，当需要文件号的信息时可以通过此文件号唯一识别指定的波形。

12.49 filterdesign

概要

产生一个滤波器的数字和模拟特性的图形显示，包括：振幅、相位、脉冲响应和群延迟。

语法

```
FILTERDESIGN [FILE [prefix]] [filteroptions] [delta]
```

输入

FILE prefix 生成的三个 OBSPY 文件的前缀
filteroptions 与 OBSPY 中其他的滤波命令相同，包括滤波器类型
delta 数据的采样间隔。

缺省值

`delta` 缺省值为 0.025 s，其他参数无缺省值

说明

`filterdesign` 命令调用了函数 `xapiir`（一个递归数字滤波器包）。`xapiir` 通过模拟滤波器原型的双线性变换实现标准递归数字滤波器的设计。这些原型滤波器由零点和极点给定，然后使用模拟谱变换，变换到高通、带通和带阻滤波器。

`filterdesign` 用实线显示数字滤波器响应，用虚线显示模拟滤波器响应。在彩色显示器上，数字曲线是蓝色的而模拟曲线是琥珀色的。

生成的三个 OBSPY 文件分别为 `prefix.spec`、`prefix.grd`、`prefix.imp`。其中 `prefix.spec` 为该命令产生的振幅相位信息，为振幅-相位格式谱文件。`prefix.grd` 为该命令产生的群延迟信息，是时间序列文件。需要注意的是，尽管这个文件是时间序列文件，但是实际上群延迟是频率的函数。用户要记住，虽然绘图时横轴单位是秒，实际的单位却是 Hz。`prefix.imp` 是时间序列文件，包含脉冲响应信息。

在这三个 OBSPY 文件中，用户自定义头段变量 `USERn`、`KUSERn` 设置如下：

- user0：表示 pass code。1 代表 LP；2 代表 HP；3 代表 BP；4 代表 BR；
- user1：type code。1 代表 BU，2 代表 BE，3 代表 C1，4 代表 C2；
- user2：number of poles
- user3：number of passes
- user4：tranbw

- user5: attenuation
- user6: delta
- user7: first corner
- user8: second corner if present, or -12345 if not
- kuser0: pass (lowpass, highpass, bandpass, or bandrej)
- kuser1: type (Butter, Bessel, C1, or C2)

示例

下面的例子展示了如何使用 `filterdesign` 命令产生一个高通，拐角频率为 2 Hz，六极、双通滤波器的数字和模拟响应曲线，数据采样间隔为 0.025 s：

```
OBSPY> fd hp c 2 n 6 p 2 delta .025
```

12.50 fir

概要

应用一个有限脉冲响应滤波器

语法

```
FIR [REC|FFT] file
```

输入

FFT 通过 FFT 变换方法应用 FIR 滤波器
REC 递归应用 FIR 滤波器
file 包含 FIR 滤波器的文件名

缺省值

```
fir fft fir
```

说明

该命令中使用的滤波器必须首先用 DFIR 交互式滤波器设计。该滤波器通过变换方法应用，除非你要求使用递归方法或者数据点数对于变换方法来说太大。这些滤波器都没有相位失真但在脉冲 信号前会产生前驱波。

头段变量

depmin、depmax、depmen

限制

变换方法的最大数据点数是 4096。

12.51 floor

概要

对数数据的最小值

语法

```
FLOOR [ON|OFF|v]
```

输入

ON 打开 floor 选项开关但不改变其值
OFF 关闭 floor 选项开关
v 打开 floor 选项开关并改变阈值

缺省值

```
floor 1.0e-10
```

说明

当坐标轴取对数坐标时，对于 X 和 Y 轴，当其值小于 `floor` 设置的值时，则在绘图前将这些值改为 `floor` 设置的值。floor 使用一个小的正值，对于非正数的对数运算是不允许的。

12.52 funcgen

概要

生成一个函数并将其存在内存中

语法

```
FUNCGEN [type] [DELTA v] [NPTS n] [BEGIN v]
```

其中`type`是下面中的一个：

```
IMPULSE | STEP | BOXCAR | TRIANGLE | SINE [v1 v2] | LINE [v1 v2] |
QUADRATIC [v1 v2 v3] | CUBIC [v1 v2 v3 v4] | SEISMOGRAM |
RANDOM [v1 v2] | IMPSTRIN [n1 n2 ... nN]
```

输入

IMPULSE 位于时间序列中点的脉冲函数
IMPSTRIN n1 n2 ... nN 在指定的一系列数据点处产生脉冲函数
STEP 阶跃函数。数据的前半段为0，后半段为1
BOXCAR 矩形函数。数据的前、后三分之一值为 0，中间三分之一值为 1
TRIANGLE 三角函数。数据的第一个四分之一值为 0，第二个四分之一的值从 0 线性增加到 1，第三个四分之一的值从 1 线性减少到 0，最后四分之一值为 0
SINE v1 v2 正弦函数。`v1` 表示频率，单位为 Hz；`v2` 是以度为单位的相位角。正弦函数的振幅为 1，注意在相位参数中有一个 2π 因子：$F = 1.0\sin(2\pi(v_1 t + v_2))$
LINE v1 v2 线性函数。斜率为`v1`，截距为`v2`，即 $v_1 t + v_2$
QUADRATIC v1 v2 v3 二次函数 $v_1 t^2 + v_2 t + v_3$
CUBIC v1 v2 v3 v4 三次函数 $v_1 t^3 + v_2 t^2 + v_3 t + v_4$
SEISMOGRAM 地震样本数据。此样本数据有 1000 个数据点。`DELTA`、`NPTS` 和 `BEGIN` 选项对该样本数据无效

RANDOM v1 v2 生成随机序列（高斯白噪声）。v1 是要生成的随机序列文件的数目，v2 是用于产生第一个随机数的“种子”，该种子值保存在 USER0 中，因而如果需要你可以在稍后生成一个完全相同的随机序列

DELTA v 设置采样周期为 v，储存在头段 delta 中

NPTS n 设置函数的数据点数为 n，储存在头段 npts 中

BEGIN v 设置起始时间为 v，储存在头段 b 中

缺省值

```
funcgen impulse npts 100 delta 1.0 begin 0.
```

对于正弦函数频率和相位缺省值分别为0.05和0。一次、二次、三次函数的系数都是1。随机序列数为 1，种子是 12357。

说明

执行这个命令等效于读取单个文件（RANDOM 选项会生成多个文件）到内存中，文件名即为函数名。内存中原有的数据会被该命令生成的函数所替换。

12.53 getbb

概要

获取或打印黑板变量的值

语法

```
GETBB [TO TERMINAL|filename] [NAMES ON|OFF] [NEWLINE ON|OFF]
    ALL|variable [variable ...]
```

输入

TO TERMINAL 打印值到终端

TO filename 将值追加到文件 filename 后

NAMES ON 输出格式为“黑板变量名 = 黑板变量值”

NAMES OFF 只打印黑板变量值

NEWLINE ON 打印每个黑板变量后换行

NEWLINE OFF 打印黑板变量后不换行

ALL 打印当前定义的全部黑板变量

variable 打印列表指定的黑板变量

缺省值

```
getbb to terminal names on newline on all
```

说明

该命令用于获取或打印黑板变量的值。可以控制打印哪些黑板变量以及具体的打印格式。可以 将黑板变量打印到终端或者文本文件中。可以使用这些选项对一系列数据文件进行测量，将结果保 存到文本文件中，然后用 readtable 命令将这个文件读回 OBSPY，绘图或者进行更多的分析。

示例

假设你已经设置了一些黑板变量：

```
OBSPY> setbb c1 2.45 c2 4.94
```

稍后可以这样打印他们的值：

```
OBSPY> getbb c1
 c2 c1 = 2.45
 c2 = 4.94
```

想要在一行内只打印其值：

```
OBSPY> getbb names off newline off c1
 c2 2.45 4.94
```

假设你有一个宏文件叫 GETXY，其可以对单个文件进行某些分析操作，并将结果储存在两个头段变量中 X 和 Y 中。你想要对当前目录中所有垂直分量进行操作，保存每对 X 和 Y 的值，然后绘图。下面的宏文件的第一个参数是用于储存这些结果的文本文件：

```
DO FILE WILD *Z
  READ FILE
  MACRO GETXY
  GETBB TO 1 NAMES OFF NEWLINE OFF X Y
ENDDO
GETBB TO TERMINAL
READALPHA CONTENT P 1
PLOT
```

最终这个文本文件将包含成对的 X-Y 数据点，每行一个，对应一个垂直分量的数据文件。为了关闭文本文件并清空缓存区，最后将输出重定向到终端的 `getbb` 命令是必要的。

12.54 grayscale

概要

产生内存中数据的灰度图像 [1]

语法

```
GRAYSCALE [VIDEOTYPE NORMAL|REVERSED] [SCALE v] [ZOOM n]
    [XCROP n1 n2|ON|OFF] [YCROP n1 n2|ON|OFF]
```

输入

VIDEO NORMAL 设置 video 类型为 `normal`。在 Normal 模式中，最小值附近的数据位黑色，最大值附近的数据为白色

VIDEO REVERSED 设置 video 类型为 `reversed`。在 Reversed 模式中，最小值附近的数据位白色，最大值附近的数据为黑色

[1] 这个命令使用了未在 OBSPY 中发布的命令，要使用这个命令你必须安装 Utah Raster Toolkit。

SCALE v 改变数据的比例因子为 v，The data is scaled by raising it to the vth power。小于 1 的值将平滑图像、降低峰和谷，大于 1 的值将伸展整个数据

ZOOM n Image is increased to n times its normal size by pixel replication.

XCROP n1 n2 Turn x cropping option on and change cropping limits to n1 and n2. The limits are in terms of the image size.

XCROP ON Turn x cropping option on and use previously specified cropping limits.

XCROP OFF Turn x cropping option off. All of the data in the x direction is displayed.

YCROP n1 n2 Turn y cropping option on and change cropping limits to n1 and n2. The limits are in terms of the image size.

YCROP ON Turn y cropping option on and use previous specified cropping limits.

YCROP OFF Turn y cropping option off. All of the data in the y direction is displayed.

缺省值

```
grayscale videotype normal scale 1.0 zoom 1 xcrop off ycrop off
```

说明

这个命令可以用于绘制 spectrogram 命令输出的灰度图，用这个命令显示的 OBSPY 数据须是“xyz” 文件。

注意：OBSPY 启动了一个脚本来运行图像操作和显示程序，然后再显示 OBSPY 的提示符。对于大 型图像或较慢的机器，这中间会有个明显的延迟。

限制

最大只能显示 512*1000

头段变量改变

需要：iftype、nxsize、nysize

错误消息

OBSPY> getsun: Command not found.（需要 Utah Raster Toolkit 提供一些工具程序）

12.55 grid

概要

控制绘图时的网格线

语法

```
GRID [ON|OFF|SOLID|DOTTED]
```

输入

SOLID 设置网格线为实线
DOTTED 设置网格线为虚线
ON 绘制网格，但不改变网格类型
OFF 不绘制网格

缺省值

```
grid off
```

说明

该命令控制 X 和 Y 轴的网格线的绘制。可以使用 xgrid 和 ygrid 分别控制单个坐标轴的网格类型。

12.56 gtext

概要

控制绘图中文本质量以及字体

语法

```
GTEXT [SOFTWARE|HARDWARE] [FONT n] [SIZE size] [SYSTEM system]
    [NAME name]
```

输入

SOFTWARE 绘图中使用软件文本
HARDWARE 绘图中使用硬件文本
FONT n 设置软件文本字体为 n，n 取值为 1 到 8
SIZE size 改变缺省文本大小，可以取 TINY、SMALL、MEDIUM、LARGE，这些缺省文本尺寸的具体大小可以参考 tsize 命令
SYSTEM system 修改字体子系统，可以取值为 SOFTWARE、CORE、XFT
NAME name 修改 CORE 或 XFT 子系统的默认字体名，可以取 Helvetica、Times-Roman、Courier、ZapfDingbats

缺省值

```
gtext software font 1 size small
```

说明

软件文本使用了图形库的文本显示功能，将每个字符以线段的形式保存起来，因而可以任意缩放或旋转至任意角度。使用软件文本在不同图形设备上可以产生相同的结果，但是其速度会慢于硬件文本。目前有 8 种可用的软件字体：

1. simplex block
2. simplex italics
3. duplex block
4. duplex italics
5. complex block
6. complex italics
7. triplex block
8. riplex italics

硬件文本使用图形设备自身的文本显示功能，因而文本在不同的设备上尺寸可能不同，所以使用硬件文本会导致在不同的图形设备上看到不同的图。如果一个设备有超过一个硬件文本尺寸，那

么最接近预期值的那个尺寸将被使用。其最主要优点在于速度较快，因而当速度比质量重要时可以使用。

示例

选择 triplex 软件字体：

```
SCA> gtext software font 6
```

12.57 hanning

概要

对数据文件加 hanning 窗

语法

```
HANNING
```

说明

hanning 窗是一种对数据点的递归平滑算法。对于每个非端点数据点（$j \in [2, N-1]$）而言，有

$$Y(j) = 0.25Y(j-1) + 0.50Y(j) + 0.25Y(j+1)$$

更新 $Y(2)$ 时使用了 $Y(1)$、$Y(2)$、$Y(3)$ 的旧值，而更新 $Y(3)$ 时则使用了 $Y(2)$ 的新值以及 $Y(3)$、$Y(4)$ 的旧值，这也是其称为**递归**平滑算法的原因。

两个端点的数据，需要做特殊处理，令 $Y(1)$ 等于 $Y(2)$ 的新值，$Y(N)$ 等于 $Y(N-1)$ 的新值。

头段变量

depmin、depmax、depmen

12.58 help

概要

在终端显示 OBSPY 命令的语法和功能信息

语法

```
HELP [item ... ]
```

输入

item 命令（全称或简写）、模块、子程序等等。若 item 为空，则显示 OBSPY 的帮助文档的介绍

说明

OBSPY 的官方帮助文档位于 `$OBSPYHOME/aux/help` 目录中，该命令实际上是从该目录中读取相应 的文件并输出到终端中。item 列表中每一项会按照顺序依次显示在终端中，若输出超过一屏，可以 使用 PgUp、PgDn、Enter、空格、方向键等实现翻页。直接输入 `q` 则退出当前 item 的文档并显示 下一 item 的文档。

示例

```
OBSPY> h              // 获得帮助文档包的介绍
OBSPY> h r cut bd     // 一次显示多个命令的文档
```

12.59　highpass

概要

对数据文件应用一个无限脉冲高通滤波器

语法

```
HIGHPASS [BUTTER|BESSEL|C1|C2] [CORNERS v1 v2] [NPOLES n] [PASSES n]
    [TRANBW v] [ATTEN v]
```

输入

BUTTER 应用一个 Butterworth 滤波器
BESSEL 应用一个 Bessel 滤波器
C1 应用一个 Chebyshev I 型滤波器
C2 应用一个 Chebyshev II 滤波器
CORNERS v1 v2 设定拐角频率分别为 v1 和 v2，即频率通带为 v1–v2
NPOLES n 设置极数为 N，范围：1–10
PASSES n 设通道数为 N，范围：1–2
TRANBW v 设置 Chebyshev 转换带宽为 v
ATTEN v 设置 Chebyshev 衰减因子为 v

缺省值

```
highpass butter corner 0.2 npoles 2 passes 1 tranbw 0.3 atten 30
```

说明

参见 bandpass 的相关说明。

示例

应用一个四极 Butterworth，拐角频率为 2 Hz：

```
OBSPY> hp n 4 c 2
```

在此之后如果要应用一个二极双通具有相同频率的 Bessel：

```
OBSPY> hp n 2 be p 2
```

头段变量

depmin、depmax、depmen

12.60 hilbert

概要

应用 Hilbert 变换

语法

```
HILBERT
```

说明

一个实序列 $f(t)$ 的 Hilbert 变换定义为

$$H(f(t)) = f(t) * (\frac{1}{\pi t}) = \frac{1}{\pi}\int_{-\infty}^{+\infty}\frac{f(t)}{t-\tau}d\tau$$

其中星号表示卷积。

由 $\frac{1}{\pi t}$ 的 Fourier 变换为 $-isgn(\omega) = -e^{i\pi/2}sgn(\omega)$，因而对一个信号做 Hilbert 变换可以理解为先将信号做 Fourier 变换，然后乘以 $-e^{i\pi/2}sgn(\omega)$，再做反 Fourier 变换。即 Hilbert 变换的一个重要性质是对信号产生 $\frac{\pi}{2}$ 的相移。[1]

该命令通过将原始信号与一个 201 点 FIR 滤波器进行时间域卷积以实现 Hilbert 变换。此 FIR 滤波器是通过对理想 Hilbert 变换的脉冲响应加 Hanning 窗获得的。在频率域，每个频率处的振幅响应为 1，相位为 90 度。Hilbert 变换后的结果将替代内存中的原始信号。

需要注意的是，此操作在零频和 Nyquist 频率附近的小区域内是不精确的。若要对很低频率的数据进行 Hilbert 变换（比如长周期面波），则需要先对信号进行减采样。由于该变换是在时间域完成的，所以计算时在原地使用重叠储存算法，其对于文件长度没有限制。

理论上，Hilbert 变换可以从振幅谱的对数中计算最小延迟相位，本质上 Hilbert 变换是一个非带限的低通滤波器，因而本命令中的 Hilbert 变换无法用于计算最小延迟相位。

OBSPY 提供了 Hilbert 变换的函数库，可以直接在 C 或 Fortran 程序中调用，详情参考“调用 libObspy 库”一节。

头段变量

depmin、depmax、depmen

12.61 history

概要

打印最近执行的 OBSPY 命令列表

语法

```
HISTORY
```

说明

该命令可以打印最近执行的命令历史，也可用于重复执行命令历史中某个特定的命令：

- !! 重复上一命令

[1] 本段内容不够严谨，正负号可能有误，其中的细节也被省略，因而仅供参考。

- `!n` 重复第 n 行的命令
- `!-n` 重复倒数第 n 个命令
- `!str` 重复最近的以字符串 `str` 开头的命令

示例

打印命令历史列表：

```
OBSPY> history
```

重复命令 1：

```
OBSPY> !1
```

重复最后一条命令：

```
OBSPY> !!
```

重复倒数第二个命令：

```
OBSPY> !-2
```

重复以 `ps` 开头的命令：

```
OBSPY> !ps
```

12.62　ifft

概要

对数据进行离散反傅立叶变换

语法

```
IFFT
```

说明

数据文件必须是之前利用 `fft` 命令生成的谱文件，可以是实部-虚部格式或振幅-相位格式。 该命令会从 `sb、sdelta、nsnpts` 中读取原始数据在时间域的起始时间、采样周期和数据点数。频率域的起始频率、采样频率、采样点数将被保存到 `sb、sdelta、nsnpts` 中。

头段变量

b、delta、npts、sb、sdelta、nsnpts

限制

目前 `ifft` 所允许的最大数据点数为 65536。

12.63　image

概要

利用内存中的数据文件绘制彩色图

语法

```
IMAGE [COLOR|GREY] [BINARY|FULL] [PRINT [pname]]
```

输入

COLOR|GREY 绘制彩图或者灰度图

BINARY|FULL 绘图时所有正值是一个颜色，所有负值是另一种颜色，或者根据数据值不同变换颜色

缺省值

```
image color full
```

说明

该命令允许用户用 OBSPY 三维数据绘制彩图或灰度图。

三维数据可以用 spectrogram、sonogram 或 bbfk 命令产生，也可以自己生成 OBSPY 格式的三 维数据。可以使用 xlim 和 ylim 以控制要显示的绘图效果，也可以使用其他命令对数据做振幅上 的操作。

示例

以 OBSPY v101.5c 自带的 contourdata 为例：

```
OBSPY> r
contourdata
```

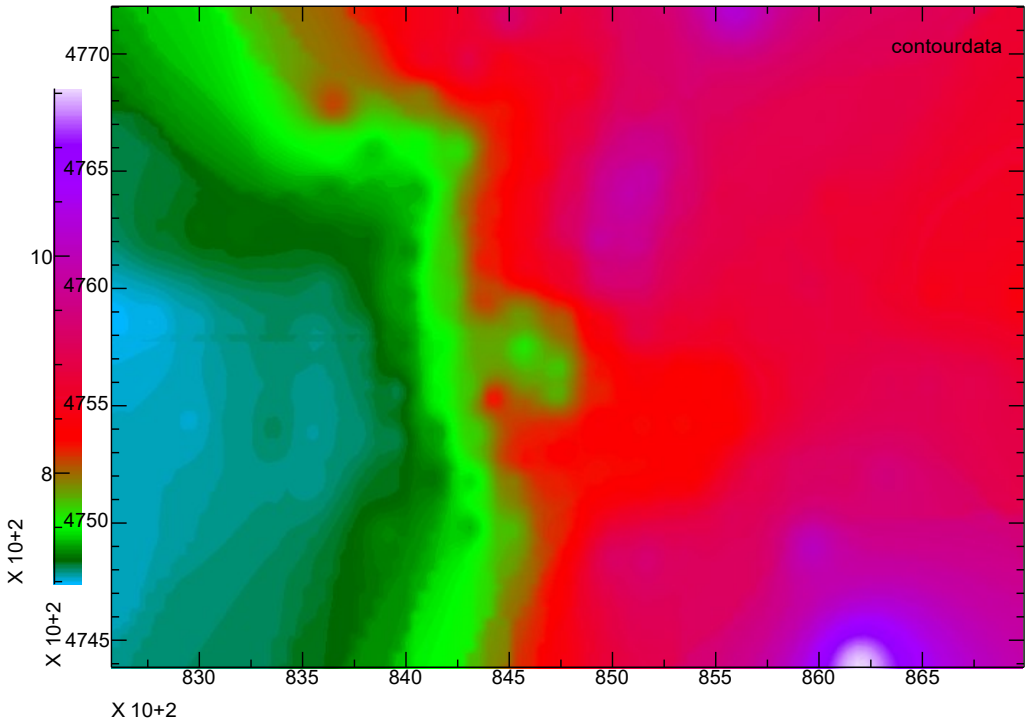

图 12.2: image 示意图

头段变量

需要：iftype（设为 “IXYZ”）、nxsize、nysize

使用：xminimum、xmaximum、yminimum、ymaximum

12.64 inicm

概要

重新初始化 OBSPY

语法

```
INICM
```

说明

此命令可以在任意时刻将 OBSPY 初始化到其刚启动的状态。所有活动的图像设备被终止，全局 变量初始化到其初始值，内存中数据丢失。

12.65 installmacro

概要

将宏文件安装到 OBSPY 全局宏目录中

语法

```
INSTALLMACRO name [name ...]
```

输入

name OBSPY 宏文件名

说明

该命令允许用户将自己写的宏文件安装到 OBSPY 全局宏目录（`${OBSPYAUX}/macros`）中，使得所 有人都可以使用。

12.66 int

概要

利用梯形法或矩形法对数据进行积分

语法

```
INT [TRAPEZEIDAL|RECTANGULAR]
```

缺省值

```
int trapezoidal
```

说明

该命令使用梯形法或矩形法对数据进行数值积分，可以处理等间隔数据或非等间隔数据。除积 分之外，还会根据具体情况修改因变量类型 `idep`，并重新计算 `depmax`、`depmin`、`depmen`。

对于函数$f(x)$其积分用梯形法表示为

$$y_n = y_{n-1} + \frac{1}{2}(x_{n+1} - x_n)(f(x_n) + f(x_{n+1})), \quad n \in [1, npts-1]$$

用矩形法表示为：

$$y_n = y_{n-1} + (x_n - x_{n-1})f(x_n), \quad n \in [1, npts]$$

二者均有边界条件$y_0 = 0$。

对于等间隔数据，若使用梯形积分，数据点数 npts 将减 1，文件的头段变量 b 将增加半个采样周期[1]。

头段变量

depmin、depmax、depmen、idep、npts、b、e

12.67 interpolate

概要

对等间隔或不等间隔数据进行插值以得到新采样率

语法

```
INTERPOLATE [DELTA v|NPTS v] [BEGIN v]
```

输入

DELTA v 设置新采样率为 v。数据的时间跨度（E-B）保持不变，npts 变化，E 由于需要与 b 的间距为 delta 的整数倍，所以可能会有微调

NPTS n 强制设置插值后文件的数据点数为 n。时间宽度不变，delta 发生变化。

BEGIN v 在 v 处开始插值，该值将作为插值文件的起始时间。BEGIN 可以和 DELTA 或 NPTS 选项一起使用。

说明

该命令使用 Wiggins 的 weighted average-sloped 插值方法将不等间隔数据转换为等间隔数据，以及对等间隔数据插值得到新的采样率。不像三次样条插值，在输入样本数据点间不会存在极值。 如果要降低采样率，即减采样，由于该命令没有抗混叠滤波器，所以最好使用 decimate 命令。

DELTA 选项和 NPTS 选项只能同时使用一个，若二者同时使用，则命令中的后者起作用。

BEGIN 选项用于控制输入数据的插值起点，也可以通过 cut 命令设置 b 和 e 再进行插值操作。

示例

假定 filea 是等间隔数据，采样间隔为 0.025s，为了将转换到采样间隔为 0.02s：

```
OBSPY> r filea
OBSPY> interp delta 0.02
```

由于新 delta 小于原数据 delta，可能会出现混叠现象，所以会输出警告信息。

假定 fileb 数据点数为 3101，想要保持其时间跨度，并采样至 4096 个点：

[1] 若使用矩形积分，理论上 npts 和 b 也应有所修改，但实际代码中却未对此多做处理，暂不确定是否是 Bug。

```
OBSPY> r fileb
OBSPY> interp npts 4096
```

假设 filec 是不等间隔数据，为了将其转换为采样率为 0.01s 的等间隔数据：

```
OBSPY> read filec
OBSPY>  interpolate  delta  0.01
```

头段变量

delta、npts、e、b、leven

12.68　keepam

概要

保留内存中谱文件的振幅部分

语法

```
KEEPAM
```

说明

该命令会保留谱文件的振幅部分，丢弃相位部分，这样做可以将谱文件的振幅部分转换为等间 隔的数据，可以用 OBSPY 的其他命令直接操作。

谱文件可以是振幅-相位格式或实部-虚部格式。若文件以实部-虚部格式存在，则首先将其转换 为振幅-相位格式再丢弃相位部分。

对于实序列，其振幅谱是对称的，因而最终生成的振幅文件仅含 npts/2+1 个数据点，数据的文件类型定义为 `IXY` ，可以跟一般的时间序列区分开。

12.69　khronhite

概要

对数据应用 Khronhite 滤波器

语法

```
KHRONHITE [v]
```

输入

v 截断频率，单位 Hz

缺省值

```
khronhite  2.0
```

说明

此低通滤波器是两个级联的四阶 Butterworth 低通模拟滤波器的数字近似。其拐角频率为 0.1Hz 以增强区域地震的基模瑞利波（Rg）的振幅信号。

头段变量

depmin、depmax、depmen

12.70 line

概要

控制绘图中的线型

语法

```
LINE [ON|OFF|SOLID|DOTTED|n] [FILL ON|OFF|pos_color/neg_color]
    [INCREMENT [ON|OFF]] [LIST STANDARD|nlist]
```

输入

ON 打开线型选项，不改变线型
OFF 关闭线型选项
SOLID 改变线型为实线型，并打开线型开关
DOTTED 改变线型为虚线型，并打开线型开关
n 设置线型为 n 并绘制线条。若 n 取值为 0 则表示不绘制该线条
FILL ON|OFF 打开/关闭颜色填充
FILL pos_color/neg_color 对每个数据的正值和负值部分涂色
INCREMENT ON 每个数据被绘出之后，按照线型表中的次序改变为另一个线型
INCREMENT OFF 不改变线型
LIST nlist 改变线型列表
LIST STANDARD 设置为标准线型列表

缺省值

```
line solid increment off list standard
```

说明

这个命令控制绘图时的线型，图形的框架（轴、标题等等）通常使用实线。网格的线型用 grid 命令控制。并非所有的图形设备都有除实线型之外的其他线型的，在那些设备上显然这个命令没有什么效果。而对于不同的设备线型号 n 也可能是不同的。

还有其他命令可以控制数据显示的其他方面。symbol 命令用于将每个数据点的值用一个符号显示在图上。color 命令控制彩色图形设备的颜色选择。所有的这些属性都是独立的。如果你想的话你可以选择在每个数据点上选择带符号的蓝色虚线。线型为 0 代表关闭画线选项。在 LIST 选项和 symbol 命令中可以利用线型为 0 的特性，在同一张图上将某些数据用线显示某些数据用符号显示。

示例

选择依次变化的线型，从线型 1 开始：

```
OBSPY> line 1 increment
```

改变线型表使之包含线型 3、5 和 1：

```
OBSPY> line list 3 5 1
```

使用 `plot2` 在同一个图形上绘制三个文件，第一个使用实线无符号，第二个没有线条，用三角符号，第三个无线条，用十字符号：

```
OBSPY> read file1 file2 file3
OBSPY> line list 1 0 0
increment
OBSPY> symbol list 0 3 7
```

将地震图的正值部分涂上红色，负值部分涂上蓝色，如果线型为 0，则涂色区域用黑色描边：

```
OBSPY> fg seis
OBSPY> line 0 fill
red/blue OBSPY> p
```

12.71 linefit

概要

对内存中数据的进行最小二乘线性拟合

语法

```
LINEFIT
```

说明

此命令的底层实现与 `rmean` 命令是相同的。 对数据使用最小二乘拟合得到一条直线，并将拟合结果写到黑板变量中：

- SLOPE：直线的斜率
- YINT：Y轴截距
- SDSLOPE：斜率的标准差
- SDYINT：截距的标准差
- SDDATA：数据的标准差
- CORRCOEF：数据和模型间的相关系数

示例

```
OBSPY> fg
seis                    // 线性拟合
 Slope and standard deviation are: 0.00023042 0.0035114
 Intercept and standard deviation are: -0.10165 0.048355
 Data standard deviation is: 0.32054
```

```
 Data correlation coefficient is: 0.0020772
OBSPY> getbb          // 查看黑板变量
 CORRCOEF     = 0.00207718
 NUMERROR = 0
 OBSPYERROR =
 'FALSE' OBSPYNFILES
 = 1
 SDDATA      = 0.32054
 SDSLOPE      = 0.00351136
 SDYINT      = 0.0483548
 SLOPE      = 0.000230417
 YINT      = -0.10165
OBSPY> getbb SLOPE   //   查看单个头段变量时出错，猜测是
```

12.72 linlin

概要

设置 X、Y 轴均为线性坐标

语法

```
LINLIN
```

12.73 linlog

概要

设置 X 轴为线性坐标，Y 轴为对数坐标

语法

```
LINLOG
```

12.74 listhdr

概要

列出指定的头段变量的值

语法

```
LISTHDR [DEFAULT|PICKS|SPECIAL] [FILES ALL|NONE|list]
    [COLUMNS 1|2] [INCLUSIVE ON|OFF] [hdrlist]
```

输入

DEFAULT　使用默认的头段变量列表，即列出所有已定义的头段变量
PICKS　使用 picks 头段列表，即列出与到时拾取相关的头段变量
SPECIAL　使用用户自定义的特殊头段变量列表
FILES ALL　列出内存中所有文件的头段
FILES NONE　不列出头段，为将来的命令设置默认值
FILES list　列出部分文件的头段，`list` 是要列出的文件的文件号
COLUMNS 1|2　输出格式为每行一/两列
INCLUSIVE ON|OFF　`ON` 表示列出未定义的头段变量的值，`OFF` 则不列出
hdrlist　指定头段变量列表

缺省值

```
listhdr default files all columns 1 inclusive off
```

说明

该命令可以列出指定的头段变量的值，用户可以使用默认列表 `DEFAULT` 列出全部头段变量，或 `PICKS` 列出与到时拾取相关的头段变量，包括 `B`、`E`、`O`、`A`、`Tn`、`KZTIME`、`KZDATE`。用户可以自定义一个特殊列表并通过 `SPECIAL` 选项再次使用该列表。

该命令的输出包含头段变量名、一个等于号以及头段变量的当前值。未定义的头段变量的值用 `undefined` 表示。

示例

获取 `picks` 列表，输出为两列显示：

```
OBSPY> lh picks column 2
```

获得第三、四个文件的默认头段列表：

```
OBSPY> lh files 3 4
```

列出文件开始和结束时间：

```
OBSPY> lh b e
```

定义一个包含台站参数的特殊列表：

```
OBSPY> lh kstnm stla stlo stel stdp
```

稍后再次使用上面的特殊列表：

```
OBSPY> lh special
```

为后面的 `lh` 命令设置输出为两列：

```
OBSPY> lh columns 2 files none
```

12.75　load

概要

导入外部命令（在 OBSPY 的 linux 版本中外部命令的导入需要额外的工作）

语法

```
LOAD comname [ABBREV abbrevname]
```

输入

comname 从一个共享目标文件中载入的一个外部函数名
ABBREV abbrevname 命令的简写

说明

该命令允许用户载入由 OBSPY 外部命令接口写的命令。该命令必须是一个储存在动态库文件 .so 文件中。OBSPY 将检查所有环境变量 OBSPYSOLIST 中的动态库文件，这个环境变量需要包含一个或多个动态库文件，每个库文件以空格分割。到这些动态库文件的路径必须在环境变量 LD_LIBRARY_PATH 中指定。如果 OBSPYSOLIST 未设置，则 OBSPY 将使用由 LD_LIBRARY_PATH 中指定的路径在动态库文件 libObspy.so 中寻找。一个叫做 libcom.so 的库随着 OBSPY 发布。

示例

设置你的环境变量使得 OBSPY 在当前目录从库文件 libbar.so 中查找一个称为 foo 的命令，并为 foo 设置别名为 myfft：

```
% export OBSPYSOLIST = "libcom.so libbar.so"
# Add the current directory to the search path.
% export LD_LIBRARY_PATH {$LD_LIBRARY_PATH}:.
% obspy
OBSPY> load foo abbrev myfft* load the
command OBSPY> read file1.z file2.z
file3.z
OBSPY> myfft real-imag* invoke command with its arguments,
* commands must parse their own args.
```

如何创建一个包含你的命令的共享目标库：

```
Solaris::
cc -o libxxx.so -G extern.c foo.c bar.c
SGI::
cc -g -o libxxx.so -shared foo.c bar.c
LINUX: (gcc)::
gcc -o libxxx.so -shared extern.c foo.c bar.c obspy.a
```

其中 Obspy.a 可以从你得到 Obspy 的地方获得

OBSPY 发布的外部命令

在 OBSPY 的发布版中有一个外部命令，叫做 FLIPXY。FLIPXY 将一个或多个 X-Y 数据文件作为输入，并调换 X 和 Y。这个命令在 ${OBSPYAUX}/external 中的 libcom.so 中，同时还有 FLIPXY 的源代码作为参考。为了导入 FLIPXY，libcom.so 必须包含在 $OBSPYSOLIST 中。

12.76 loadctable

概要

允许用户在彩色绘图中选择一个新的颜色表

语法

```
LOADCTABLE n|[DIR CURRENT|name] [filelist]
```

输入

n 标准 OBSPY 颜色表对于的号，n 可以取 1–17

DIR CURRENT 从当前目录载入颜色表，当前目录指的是你启动 OBSPY 的目录

DIR name 从目录 name 中载入颜色表，这是一个相对或绝对目录名

filelist 颜色表文件列表

说明

该命令允许用户选择一个新的颜色表或者通过指定颜色表的文件以及路径提供他们自己的颜　色表。如果未使用 DIR 选项，OBSPY 将首先在 $OBSPYAUX 中寻找颜色表，然后在用户的工作目录中寻　找.

在 $OBSPYAUX/ctables 下有 21 个文件，其中一个为 gmt.cpt，用于在 OBSPY 中绘制与 GMT 有　关的彩色图形，另外 20 个文件为三列数据，应该就是颜色表了，前面说的 n 取 1 到 17 可能有些过　时了。具体谁对应谁还不清楚，just try them！

12.77　log

概要

对每个数据点取其自然对数（ln　y）

语法

```
LOG
```

头段变量

depmin、depmax、depmen

12.78　log10

概要

对每个数据点取以 10 为底对数（$\log_{10} y$）

语法

```
LOG10
```

头段变量

depmin、depmax、depmen

12.79 loglab

概要

控制对数轴的标签

语法

```
LOGLAB [ON|OFF]
```

输入

ON 打开对数标签开关
OFF 关闭对数标签开关

缺省值

```
loglab on
```

说明

对于对数轴而言，标签一般位于轴上每个 10 的整数倍处，如果这个选项被打开并且标签之间有足够的空间的话则在 10 的整数倍的中间加入第二标签。

12.80 loglin

概要

设置 X 轴为对数坐标，Y 轴为线性坐标

语法

```
LOGLIN
```

12.81 loglog

概要

设置 X、Y 轴均为对数坐标

语法

```
LOGLOG
```

12.82 lowpass

概要

对数据文件应用一个无限脉冲低通滤波器

语法

```
LOWPASS [BUTTER|BESSEL|C1|C2] [CORNERS v1 v2] [NPOLES n] [PASSES n]
    [TRANBW v] [ATTEN v]
```

输入

BUTTER 应用一个 Butterworth 滤波器
BESSEL 应用一个 Bessel 滤波器
C1 应用一个 Chebyshev I 型滤波器
C2 应用一个 Chebyshev II 滤波器
CORNERS v1 v2 设定拐角频率分别为 v1 和 v2，即频率通带为 v1–v2
NPOLES n 设置极数为 N，范围：1–10
PASSES n 设通道数为 N，范围：1–2
TRANBW v 设置 Chebyshev 转换带宽为 v
ATTEN v 设置 Chebyshev 衰减因子为 v

缺省值

```
lowpass butter corner 0.2 npoles 2 passes 1 tranbw 0.3 atten 30
```

说明

参见 bandpass 命令的相关说明。

示例

应用一个四极 Butterworth，拐角频率为 2 Hz：

```
OBSPY> lp n 4 c 2
```

在此之后如果要应用一个二极双通具有相同频率的 Bessel：

```
OBSPY> lp n 2 be p 2
```

头段变量改变

depmin、depmax、depmen

12.83 macro

概要

执行 OBSPY 宏文件

语法

```
MACRO name [arguments]
```

输入

name 要执行的 OBSPY 宏文件名
arguments 宏文件参数

说明

参考 OBSPY 宏 一节。

12.84 map

概要

利用 OBSPY 内存中的所有数据文件生成一个包含台站/事件符号、地形以及台站名的 GMT 地 图，也可以在命令行上指定一个事件文件。每个地震事件符号可以根据震级、残差等确定其大小。这个命令会产生一个 PS 文件，并将该文件在屏幕上显示，同时产生一个绘制该图的 shell 脚本。

语法

```
MAP  [MERCATOR|EQUIDISTANT|AZIMUTHAL_EQUIDISTANT|ROBINSON]
     [WEST minlon] [EAST maxlon] [NORTH maxlat] [SOUTH minlat]
     [MAGNITUDE|RESIDUAL|RMEAN_RESIDUAL]  [EVEVNTFILE  filename]
     [TOPOGRAPHY] [STANAMES] [MAPSCALE on|off] [PLOTSTATIONS on|off]
     [PLOTEVENTS on|off] [PLOTLEGEND on|off] [LEGENDXY x y]
     [FILE output-file]
```

输入

OBSPY 中可以使用的投影方式包括：

- MERCATOR：投影方式为 Mercator 投影
- EQUIDISTANT：投影方式为等间距圆柱投影，经纬度为线性
- ROBINSON：投影方式为 Robinson 投影，适用于世界地图
- LAMBERT：适用于东西范围较大的区域
- UTM：通用横向 Mercator（尚未实现） 下面的选项允许用户指定地图的区域，其默认使用台站以及事件经纬度的最小最大值（如果真

是如此，这样的缺省值并不合适，因为那样意味着某些台站或事件将位于地图的边界处，但是实际 上地图范围给的还是不错的）：

- WEST：地图的最小经度
- EAST：地图的最大经度
- NORTH：地图的最大纬度
- SOUTH：地图的最小纬度
- AUTOLIMITS：自动决定地图的区域 [缺省值]

下面的选项允许用户向地图中添加位置和注释：

- STANames on|off：在地图上绘制台站名 [默认为 off]
- MAPSCALE on|off：在地图上绘制地图比例尺 [默认为 off]
- PLOTSTATIONS on|off：绘制地震图给出的全部台站 [默认为 on]
- PLOTEVENTS on|off：绘制 eventfile 和/或地震图给出的全部事件 [默认为 on]

下面的选项允许用户根据不同的值给出不同地震事件符号的大小。默认值是所有符号大小一样：

- MAGnitude：user0 定义地震震级，user0 越大，则事件符号越大
- REsidual：user0 定义残差。根据 user0 的绝对值定义事件符号的大小。正值为 + 负值为 -
- RMean_residual：与 residual 相同，除了将所有残差去除均值之外

- PLTLEGEND on|off：绘制地震震级以及残差的图例 [默认为 on]
- LEGENDXY x y：绘制图例的绝对位置，默认为 [1,1]。位置是相对于页面的左下角，其单位为 inch。这是一个与地震震级和残差有关的图例。
- EVENTFILE：指定一个自由格式的 ASCII 文本文件，其包含了额外的事件数据，文件的每一行包含单个事件的数据。每行的头两列必须包含纬度和经度（单位为度）。第三列可以包含符 号大小信息（比如震级、深度、走时残差等）。
- TOPOgraphy on|off：设置 TOPO 为开允许用户向地图中添加地形和海洋深度。该命令读取 GMT 中 grdraster.info 的第一个地形文件，当然地形文件中必须要有该区域的数据。地形彩色图使用 $OBSPYAUX/ctables/gmt.cpt。网格文件被写入当前目录
- FILE：默认的输出文件名为 gmt.ps，你可以通过 FILE 选项指定文件名 可以

用 OBSPY 的 title 命令指定地图标题。

缺省值

```
map mercator topo off stan off file gmt.ps plotstations on
    plotevents on
```

示例

利用 OBSPY 提供的一些数据作为例子：

```
OBSPY> dg sub regional *.z
OBSPY> title "Station Location
Map" OBSPY> map stan on
Using Default Postscript Viewer
       gs -sDEVICE=x11 -q -dNOPROMPT -dTTYPAUSE
```

绘制出的地图如图 12.3 所示，整个地图的边界控制的还算不错，还算比较美观，三角形代表台站位置，圆形代表地震位置，大小也控制的不错。生成这个图的同时，还有一个可以用于生成该地图的 shell 脚本。

默认情况下，该命令会自动使用 gs 预览生成的 PS 文件。如果想用其他 PS 阅读器预览，可以通过修改环境变量 OBSPYPSVIEWER 来实现，比如 export OBSPYPSVIERER=evince。

头段数据

台站纬度（stla）以及经度（stlo）必须在头段中被定义。如果事件纬度（evla）以及经度（evlo）被定义则其会被包含在地图中。如果这个命令在执行 bbfk 之后执行，map 将沿着反方位角方向绘制大圆弧路径。这个版本的 map 是基于 4.0 版本的 Generic Mapping Tools，要执行这个命令，你需要将 GMT4.0 安装在你的机器上并保证可执行文件位于路径中。

每个 map 命令的结果将写入当前目录下一个称为 gmt.csh 的脚本中。用户可以修改这个文件以利用更多 OBSPY 未利用的选项。默认单位是 inch，当然可以在脚本中修改。

在使用 pscoast 绘制海岸线时，OBSPY 采用了 -Dl 选项，其中 l 代表低精度的海岸线数据。用 户可以在脚本中修改使用更高精度的海岸线数据。

12.85 markptp

概要

测量并标记信号在测量时间窗内的最大峰峰值

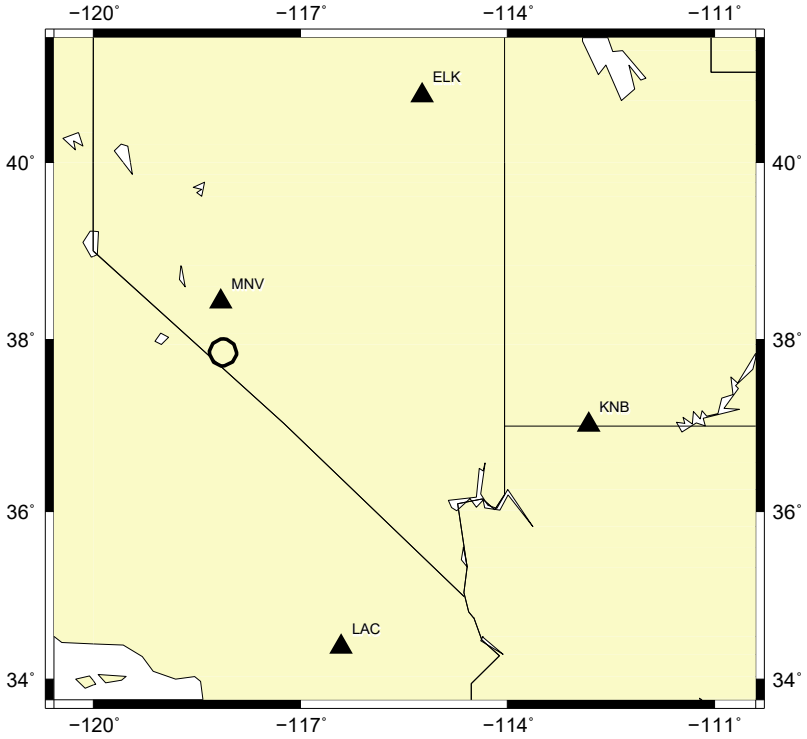

图 12.3: map 绘制地震、台站分布图

语法

```
MARKPTP [LENGTH v] [TO marker]
```

输入

LENGTH v　设置滑动窗的长度为 v 秒

TO marker　指定某个时间标记头段用于保存最小值的所对应的时刻；最大值所对应的时刻保存在下一个时间标记头段中。时间标记头段 marker 可以取 Tn（n=0−9）。

缺省值

```
markptp length 5.0 to t0
```

说明

该命令会计算信号在测量时间窗内的最大峰峰值。所谓最大峰峰值，即最大振幅与最小振幅的振幅差。测量结果中，最小值（波谷）所对应的时刻会写到 TO marker 中所指定的时间标记头段中，最大值（波峰）所对应的时刻会写到相应的下一个时间标记头段中。最大峰峰值保存到 user0 中，kuser0 中的值为 PTPAMP。如果使用 oapf 打开了字符数字型震相拾取文件，则该命令的结果也会写入到文件中。

默认情况下，测量时间窗为整个信号，可以使用 mtw 命令设置新的测量时间窗。同时，在测量时还需要设置滑动时间窗（sliding time window）的长度。滑动窗的工作原理是，首先将长度为 v 的滑动窗置于测量时间窗的起始位置，搜索该滑动窗内的最大峰峰值，然后将长度为 v 的滑动窗向右移动一个数据点，并搜索该滑动窗内的最大峰峰值，以此类推，直到滑动窗的右边界与测量时间窗 的右边界重合为止，此时将有多个最大峰峰值，最后返回所有最大峰峰值中最大的一个。

对于滑动时间窗（sliding time window）的长度，若 stw 的长度大于 mtw 的长度，则 stw=mtw；若 stw 的长度小于等于零，则 stw=mtw/2。

示例

设置测量时间窗为头段 T4 和 T5 之间，并使用默认的滑动时间窗长和时间标记：

```
OBSPY> mtw t4
t5 OBSPY>
markptp
```

设置测量时间窗为初动之后的 30s，滑动时间窗为 3s，起始时间标记为 T7：

```
OBSPY> mtw a 0 30
OBSPY> markptp l 3. to
t7 OBSPY> lh t7 t8 user0
```

头段变量

Tn、KTn、usero、kusero

源码

src/smm/xmarkptp.c、src/smm/ptp.c

12.86　marktimes

概要

根据一个速度集得到走时并对数据文件进行标记

语法

```
MARKTIMES [TO marker] [DISTANCE HEADER|v] [ORIGIN HEADER|v|GMT time]
    [VELOCITIES v ... ]
```

输入

TO marker 定义头段中用于储存结果的第一个时间标记。对下一个的速度使用下一个时间

marker T0|T1|T2|T3|T4|T5|T6|T7|T8|T9

DISTANCE HEADER 使用头段中的 dist 代表距离用于走时计算

DISTANCE v 使用 v 作为走时计算中的距离

ORIGIN HEADER 使用头段中的参考时间（O）用于走时计算

ORIGIN v 使用相对参考时间偏移 v 秒用于走时计算

ORIGIN GMT time 使用 GMT 时间作为参考时间

time GMT 时间包含六个整数：年、儒略日、时、分、秒、毫秒

VELOCITIES v ... 设置用于走时计算的速度集，最多可以输入 10 个速度

缺省值

```
marktimes velocities 2. 3. 4. 5. 6. distance
    header origin header to t0
```

说明

这个命令在头段中标记震相的到时，给定事件的发生时间，震中距以及速度即可计算到时。下 面的方程用于简单估计走时：

$$time(j) = origin + \frac{distance}{velocity(j)}$$

结果被写入指定的时间标记中

示例

使用默认的速度集，强制距离为 340 km，第一个时间标记为 T4：

```
OBSPY> marktimes distance 340. to t4
```

选择一个不同的速度集：

```
OBSPY> markt v 3.5 4.0 4.5 5.0 5.5
```

设置新的参考时间并将结果保存在 T2 中：

```
OBSPY> markt origin gmt 1984 231 12 43 17 237 to t2
```

头段变量改变

Tn、KTn

12.87 markvalue

概要

在数据文件中搜索并标记某个值

语法

```
MARKVALUE [GE|LE v] [TO marker]
```

输入

GE v 搜索并标记第一个大于或等于 v 的数据点
LE v 搜索并标记第一个小于或等于 v 的数据点
TO marker 用于保存数据点的时刻的时间标记头段，marker 可以取 Tn（n=0–9）

缺省值

```
markvalue ge 1 to t0
```

说明

该命令会在信号的测量时间窗内搜索第一个满足条件（大于或等于/小于或等于）的数据点，并 将该数据点所对应的时刻记录下来。默认情况下，测量时间窗为整个信号，可以使用 mtw 命令设置新的测量时间窗。

示例

搜索文件中第一个值大于 3.4 的点并将结果保存在头段 T7 中：

```
OBSPY> markvalue ge 3.4 to t7
```

设定测量时间窗为 T4 后的 10 秒，并搜索第一个小于-3 的值：

```
OBSPY> mtw t4 0 10
OBSPY> markvalue le -3 to t5
```

头段变量改变

Tn、KTn

12.88　mathop

概要

控制数学操作符的优先级

语法

```
MATHOP NORMAL|MATH|FORTRAN|NONE|OLD
```

输入

NORMAL　使用正常的数学操作符优先级
MATH　与 NORMAL 相同
FORTRAN　与 NORMAL 相同
NONE　不使用操作符优先级
OLD　与 NONE 相同

缺省值

```
mathop NORMAL
```

说明

该命令控制数学操作符的优先级。正常情况下，乘法和除法的优先级要比加法和减法高，指数 运算拥有最高的优先级。

101.6 之前的版本中，OBSPY 在进行代数运算时没有考虑操作符的优先级，整个表达式按照从左 到右的顺序依次进行计算。

OBSPY 在 101.6 之后的版本中，默认使用正常的操作符优先级。对于一些在老版本 OBSPY 下写的 脚本或宏来说，可能依赖于旧的优先级顺序。可以考虑修改脚本以适应新版本的优先级或者直接设 定适应 OLD 优先级。

示例

正常的操作符优先级：

```
OBSPY> mathop
normal OBSPY> eval
1+2*3
```

```
OBSPY> eval
 1+(2*3) 7
```

旧的操作符优先级：

```
OBSPY>   mathop
old OBSPY> eval
1+2*3 9
OBSPY> eval
 1+(2*3) 7
```

12.89 merge

概要

将多个数据文件合并成一个文件

语法

```
MERGE [VERBOSE] [GAP ZERO|INTERP] [OVERLAP COMPARE|AVERAGE] [filelist]
```

输入

VERBOSE 合并数据时在终端显示合并的细节

GAP ZERO|INTERP 如何处理数据间断。ZERO 表示将数据间断处补零值；INTERP 表示对数据间断处进行线性插值

OVERLAP COMPARE|AVERAGE 如何处理数据重叠。COMPARE 表示对重叠的时间段内的数据进行比较，若不匹配则退出；AVERAGE 表示对重叠时间段内的数据进行平均

filelist OBSPY 二进制数据文件列表

说明

该命令用于将一系列时间上相邻的数据文件首尾相连合并成一个文件。在合并数据时，会先检查要合并的文件是否有相同的 kstnm、kentwk、kcmpnm、delta。

在 OBSPY v101.6 中完全重新了该命令，因而该命令的功能和语法与之前的版本有所不同。

在 v101.6 之前的版本中，merge 命令的用法与 addf 命令的用法相似。该命令会将内存中的文件依次分别与 filelist 中的文件进行合并，且内存中的文件的开始时刻必须早于 filelist 中的文件的开始时刻。因而在将多个数据文件合并成一个文件时，只能先读取第一段数据，然后 merge 第二段数据，再 merge 第三段数据，依次不断循环执行。

旧版本的merge 命令用起来相对复杂。需要多次执行 merge 命令，且需要用户自己判断波形文件的先后顺序。

v101.6 之后的新版 merge 命令，会将内存中的全部数据文件以及 filelist 中的全部文件合并成单个文件。若内存中无数据，则只合并 filelist 中的数据文件；若 filelist 为空，则只合并内存中的数据文件。且新版的 merge 命令可以合并任意数目和任意顺序的数据文件。

若要合并的数据段之间存在间断，可以通过补零或线性插值的方式弥补间断；若数据段之间存在重叠，可对重叠的部分进行比较，判断重叠的区域内振幅是否匹配，或者直接进行振幅的平均。

示例

下面看一个数据合并的示例：

```
OBSPY> read file1
file2 OBSPY> merge
```

在 v101.6 之前的版本中，本示例的运行结果是：file3 与 file1 合并成文件 file1，file4 与 file2 合并成文件 file2，此时内存中有两个文件 file1 和 file2。而在 v101.6 及其之后的版本中，这个例子的结果是，文件 file1、file2、file3、file4 合并成文件 file1，此时内存中只有一个文件 file1。这样的修改使得不同版本的语法不兼容，但实际使用时新版的 `merge` 命令要更方便也更符合人的正常思维。

下面介绍几个新版本的 `merge` 命令的用法示例。 多个文件合并成单个文件的一种方法：

```
OBSPY> r file1                    // 读取一个文件
OBSPY> merge file2 file3 file4 // merge 其余文
件 OBSPY> w over
```

另一种合并办法：

```
OBSPY> r file1 file2 file3 file4
OBSPY> merge                  // 合并内存中的所有文件
OBSPY> w over                 //  合并后的文件写入到 file1 中
```

再一种合并方法：

```
                              // 内存中无数据
OBSPY> merge file1 file2 file3 file4 // 合并 filelist 中的全部文件
OBSPY> w over                 //  保存到 file1 中
```

头段变量改变

npts、depmin、depmax、depmen、e

BUGS

- v101.6a 中 `filelist` 暂不支持通配符，在下一版本中将修复此 bug

12.90 message

概要

发送信息到用户终端

语法

```
MESSAGE text
```

输入

text 要发送到终端的信息文本。若文本中有空格则必须用引号括起来

说明

此命令用于在OBSPY宏文件执行时发送状态或其他信息给用户，在交互式模式下一般没有用（除 非你想自己跟自己聊天，如示例所示）。

示例

发送无空格的信息：

```
OBSPY> message
 finished finished
```

发送带有空格的信息，需要加上单引号或双引号：

```
OBSPY> message 'Job has
 finished.' Job  has  finished.
```

12.91 mtw

概要

设置某些测量命令中的测量时间窗（measurement time window）

语法

```
MTW [ON|OFF|pdw]
```

输入

ON 打开测量时间窗选项但不改变窗的值
OFF 关闭测量时间窗选项，测量命令将对整个文件进行操作
pdw 打开测量时间窗并设置其新值，参考 pdw 一节

缺省值

```
mtw off
```

说明

OBSPY 中与测量有关的命令有 markptp 和 markvalue。当不设置 mtw 时，测量命令会对整个文 件进行操作；当设置了 mtw 时，测量命令仅 mtw 时间窗内的数据执行测量命令。

12.92 mul

概要

数据文件的每个数据点乘以同一个常数

语法

```
MUL [v1 [v2 ... vn]]
```

输入

v1 乘到第一个文件的常数
v2 乘到第二个文件的常数
vn 乘到第 n 个文件的常数

缺省值

```
mul 1.0
```

说明

参见 add 的相关说明。

头段变量改变

depmin、depmax、depmen

12.93 mulf

概要

使内存中的一组数据乘以另一组数据

语法

```
MULF [NEWHDR [ON|OFF]] filelist
```

输入

NEWHDR ON|OFF 指定新生成的文件使用哪个文件的头段。OFF 表示使用内存中原文件的头段区，ON 表示使用 filelist 中文件的头段区。缺省值为 OFF
filelist OBSPY 二进制文件列表

说明

参见 addf 命令的相关说明。

头段变量

depmin、depmax、depmen、npts、delta

12.94 mulomega

概要

在频率域进行微分操作

语法

```
MULOMEGA
```

说明

根据傅里叶变换的微分性质：

$$\mathrm{F}[f^{'}(x)] = i\omega \mathrm{F}[f(x)]$$

其中 $\omega = 2\pi f$，即函数微分在频率域可以用简单的乘法来表示。 该命令仅可对谱文件进行操作，谱文件可以是振幅-相位型或实部-虚部型。 若为振幅-相位型：

$$\mathrm{F}[f^{'}(x)] = i\omega \mathrm{F}[f(x)] = i\omega(A(\omega)e^{\theta(\omega)}) = (A(\omega)\omega)e^{\theta(\omega)+\pi/2}$$

若为实部-虚部型：

$$\mathrm{F}[f^{'}(x)] = i\omega \mathrm{F}[f(x)] = i\omega(a(\omega) + ib(\omega)) = -b(\omega)\omega + ia(\omega)\omega$$

头段变量

depmin、depmax、depmen

12.95 news

概要

终端显示关于 OBSPY 的一些信息

语法

```
NEWS
```

说明

其实质就是将文件 `${OBSPYHOME}/aux/news` 的内容显示到终端。

12.96 null

概要

控制空值的绘制

语法

```
NULL [ON|OFF|value]
```

输入

value 将数据中的空值替换为 `value`
ON 打开绘图时的空值选项
OFF 关闭绘图时的空值选项

缺省值

```
null off
```

说明

有些情况下，数据中存在间断，此时没有有用的值，称为空值。在多数情况下将空值设置为一个 预定的值，一般来说是 0.0、-1.0、-99。通常用户不希望这些值在绘图时显示。null 命令允许用户定 义空值并在绘图时不连接这些点。

示例

设置空值为-1.0 并打开空值选项：

```
OBSPY> null on -1.0
```

12.97 oapf

概要

打开一个字母数字型震相拾取文件

语法

```
OAPF [STANDARD|NAME] [file]
```

输入

STANDARD 在写震相拾取文件时使用标准文件 id。标准文件 id 包含了头段中事件名、台站名、分量方位角以及入射角。

NAME 使用 OBSPY 文件名代替标准文件 id

file 要打开的字符数字型震相拾取文件，如果该文件已经存在，则其将会被打开，新的震相拾取会加到文件的底部

缺省值

```
oapf standard apf
```

说明

震相拾取文件可以作为自动拾取（apk）以及人工拾取（plotpk）命令产生的简单数据库。每个 震相拾取被写在文件的一行上。这个文件的每个常规行包含一个文件 id、一个震相拾取 id、震相拾取的时间、拾取的幅度以及一些格式信息。这些行为 80 个字符长。文件 id 是一些标准的头段信息或者文件名。拾取到的时间是 GMT 时间或者时间偏移。这依赖于产生这些拾取的命令如 apk 或 ppk 中指定的选项。这将导致 4 个不同的格式，在第 79 列上以不同的字符标识这些行，如那些波形和峰峰值的读取数据，在 80 列以后不附加字段。一行的最大长度为 200。下面会展示不同格式的行。

12.98 ohpf

概要

打开一个 HYPO 格式的震相文件

语法

```
OHPF [file]
```

输入

file 要打开的文件名。如果文件已经存在，则打开并将新的震相加到文件底部

缺省值

```
OHPF HPF
```

说明

OBSPY 产生的 HYPO 震相拾取文件可以用于程序 HYPO71 以及其他类似事件定位程序的输入。由 apk 和 plotpk 得到的震相拾取信息被写入这个打开的文件中，这个文件可以使用 chpf 关闭。打开一个新的 HYPO 文件会自动关闭前一个已经打开的文件。打开一个已经存在的 HYPO 文件的同时也会自动删除文件的最后一行，这一行原本有一个指令标记 10 作为 HYPO 文件的结束标志符，删除最后一行意味着可以在其后添加新的拾取。终止 OBSPY 也会自动关闭任何已经打开的拾 取文件，事件分割符能够用 whpf 命令写入震相拾取文件。

12.99 pause

概要

发送信息到终端并暂停，直到终端输出换行符

语法

```
PAUSE [MESSAGE text] [PERIOD ON|OFF|v]
```

输入

MESSAGE text 暂停前发送到终端的文本，若文本包含空格则用引号括起来

PERIOD v 打开 period 选项并设置暂停的时间长度为 v 秒

PERIOD ON 打开 period 选项但不改变暂停的长度，若改选项为开，OBSPY 会暂停一段时间然后自动恢复执行

PERIOD OFF 关闭 period 选项开关。当该选项关闭时，OBSPY 将暂停直到终端输入回车键

缺省值

```
pause message 'Pausing' period off
```

说明

该命令使你能够暂停 OBSPY 宏文件的执行。当该命令被执行时，OBSPY 发送信息到终端、暂停，然 后等待用户在终端输入回车或者暂停时间结束。当你想要研究一个特定命令的输出时可以让宏文件 暂停执行再继续。

12.100　picks

概要

控制时间标记的显示

语法

```
PICKS [ON|OFF] [pick VERTICAL|HORIZONTAL|CROSS] [WIDTH v] [HEIGHT v]
```

输入

ON|OFF　打开/关闭时间标记的显示
pick OBSPY 中与时间标记有关的头段变量名，可以取 O、A、F、Tn（n=0–9）
VERTICAL　在时间标记处绘制垂直线，时间标记 ID 位于线的右下方
HORIZONTAL　在最接近时间标记的数据点处绘制水平线，时间标记 ID 位于线上或线下
CROSS　在时间标记处绘制垂直线，在最近的数据点处绘制水平线
WIDTH v　时间标记的宽度为 v
HEIGHT v 修改时间标记的高度为 v。高度和宽度仅对 HORIZONTAL 和 CROSS 使用，其数值为占图形的比例

缺省值

```
pick on width 0.1 height 0.1
```

所有的拾取标记类型都是 VERTICAL

说明

这个命令控制 OBSPY 绘图上时间标记的显示。这些时间标记标识了如震相到时、事件发生时刻 等。当打开显示选项时，每一个定义了的时间标记都会在绘图上相应时刻处绘制一条线，并且在其 旁有一个时间标记 ID。时间标记 ID 是一个 8 字符长的头段变量。头段变量中 KA、KF、KO 以及 KTn 分别是 A、F、O 和 Tn 的时间标记 ID。如果时间标记 id 未定义，那么标记的名字就是就头段变量本 身。每一个时间标记可以被显示为一条垂直线、一条水平线或一个交叉线。

示例

以交叉符号显示时间标记 T4、T5 和 T6，并改变交叉符号的高度和宽度：

```
OBSPY> picks t4 c t5 c t6 c w 0.3 h 0.1
```

12.101　plabel

概要

定义通用标签及其属性

语法

```
PLABEL [n] [ON|OFF|text] [SIZE TINY|SMALL|MEDIUM|LARGE]
    [BELOW|POSITION x y [a]]
```

输入

n 设置第 n 个通用标签的属性，若省略，则在前一通用标签号上加 1
text 设置标签的内容，并打开通用标签选项
ON 打开通用标签选项，但不改变标签文本
OFF 关闭绘图标签选项
SIZE TINY|SMALL|MEDIUM|LARGE 修改通用标签的文本尺寸。TINY、SMALL、MEDIUM、LARGE 分别表示一行 132、100、80、50 个字符
BELOW 将此标签放在前一标签的下面
POSITION x y a 定义该标签的位置。其中 x 的取值为 0 到 1，y 的取值为 0 到最大视口（一般为 0.75），a 是标签相对于水平方向顺时针旋转的角度

缺省值

默认字体大小为 small，标签 1 的位置为 0.15 0.2 0.。默认其他标签的位置为上一个标签之下。

说明

该命令允许你为接下来的绘图命令定义通用的绘图标签。你可以定义每个标签的位置及文本尺 寸。文本质量以及字体可以用 gtext 命令设定，也可以使用 title、xlabel、ylabel 生成图形的标 题以及轴标签。

示例

为绘图定义一系列标签：

```
// 三行标签
OBSPY> dg sub local cdv.z
OBSPY> plabel 'Sample seismogram'
p .12 .5 OBSPY> plabel 'from earthquake'
OBSPY> plabel 'in Livermore Valley, CA'
// 放在左下角的标签
OBSPY> plabel 5 'LLNL station: CDV' S T
P .12 .12 OBSPY> p
```

BUGS

- text 必须用引号括起来（v101.6a）
- text 的最后一个字符会被忽略（v101.6a）
- text 不能以 on 或 off 开头，会被误解释为选项 on 或 off，前者导致直接出现段错误，后者导致不显示标签（v101.6a）

12.102 plot

概要

绘制单波形单窗口图形

语法

```
PLOT
```

说明

每个数据文件绘制在单独窗口中，图形的大小由 xvport 和 yvport 决定。每个图形的 Y 轴范围由数据的极值决定，也可以用 ylim 手动限制 Y 轴的范围。X 轴的范围由 xlim 命令控制。用户可以用 fileid 控制每张图的文件 ID，也可以用 picks 控制时间标记的显示。

OBSPY 会在每张图之间暂停给你机会检查每张图，其将会在终端输出 Waiting 并等待你的响应。你可以输入回车以查看下一张图，关键字 go 或 g 可以不暂停地绘制余下的所有文件，或者关键字 kill 或 k 可以中止绘制余下的文件。

12.103 plot1

概要

绘制多波形多窗口图形

语法

```
PLOT1 [ABSOLUTE|RELATIVE] [PERPLOT ON|OFF|n]
```

输入

ABSOLUTE|RELATIVE 绝对/相对绘图模式
PERPLOT n 每张图上绘制 n 个文件
PERPLOT ON 每张图绘制 n 个文件，使用 n 的旧值
PERPLOT OFF 所有文件绘制在一张图上

缺省值

```
plot1 absolute perplot off
```

说明

plot1 用于一次性绘制多个波形，多个波形共用同一个 X 轴，但各自拥有一个单独的 Y 轴。绘图的总尺寸由当前视口决定（xvport 和 yvport）。每一个子图的大小由视口大小以及要绘制的波形数目决定。绘图的 X 轴范围可以是固定的（xlim）也可以是与数据长度成比例的。每个子图的 Y 轴范围由文件极值决定或者可以通过 ylim 命令自己设置。

多个波形共用 X 轴时，有 absolute 和 relative 两种绘图模式。在 absolute 模式下，所有波形将按照其绝对时刻对齐，即 X 轴表示相对于某个固定参考时刻的秒数；在 relative 模式下，所有数据将按照文件开始时间对齐，X 轴的范围为 0 到最大时间差（所有文件中结束时间和开始时间的最大时间差，即最大 e-b），每个波形从 X 轴的零点开始绘制，该零点所对应的真实时刻，会在图中以 OFFSET: xxx 的形式标出。

示例

下面的例子是由 LLNL DSS 的 4 个台站 Elko、Kanab、Landers 和 Mina 记录到的美国西部的一个地震。参考时间为事件发生时刻：

```
OBSPY> cut -5 200
OBSPY> read *v
 elk.v knb.v lac.v mnv.v
OBSPY> fileid location ul type list kstcmp
OBSPY> title 'regional earthquake: &1,kztime& &1,kzdate&'
```

```
OBSPY> qdp 2000
OBSPY> plot1
```

12.104 plot2

概要

产生一个多波形单窗口绘图

语法

```
PLOT2  [ABSOLUTE|RELATIVE]
```

输入

ABSOLUTE|RELATIVE 绝对/相对绘图模式

缺省值

```
p2 absolute
```

说明

所有波形数据将绘制在一个绘图窗口中，所有波形共用 X 轴和 Y 轴。xlim 和 ylim 用于控制 X 轴和 Y 轴的范围，若未指定轴范围，则根据所有文件的极值确定轴范围。由于所有波形重叠在一起，所以通常需要用 line、color、symbol、width 为不同波形设置不同的属性，图例的位置由 fileid 控制。

多个波形共用 X 轴时，有 absolute 和 relative 两种绘图模式。在 absolute 模式下，所有波形将按照其绝对时刻对齐，即 X 轴表示相对于某个固定参考时刻的秒数；在 relative 模式下，所有数据将按照文件开始时间对齐，X 轴的范围为 0 到最大时间差（所有文件中结束时间和开始时间的最大时间差，即最大 e-b），每个波形从 X 轴的零点开始绘制。

与 plot 和 plot1 不同，plot2 可以直接用于绘制谱文件。实部-虚部格式的谱数据被绘制为实部-频率。振幅-相位数据被绘制为振幅-频率图，虚部和相位信息被忽略。谱文件总是用 relative 模式绘图。在频率域 b、e 和 delta 分别被设置为 0、Nqquist 频率以及频率间隔 df。头段值 depmin 和 depmax 未改变。与 plotsp 类似，若 xlim off，绘图将从 df=delta 处开始，而非 0。若 xlim 或 ylim 在数据变换到频率域之前被改变了，最好在使用 plot2 绘图之前输入 xlim off 和 ylim off。

若内存中同时存在时间序列文件和谱文件，且绘图时没有选择 relative 模式，则时间序列将以绝对模式绘图，谱文件将以相对模式绘图绘图。相对模式意味着相对于第一个文件，因而内存中 文件的顺序将影响绘图之间的关系。

示例

```
OBSPY> read mnv.z.am knb.z.am
elk.z.am OBSPY> xlim 0.04 0.16
OBSPY> ylim 0.0001 0.006
OBSPY> linlog
OBSPY> symbol 2 increment
OBSPY> title 'rayleigh wave amplitude spectra for nessel'
```

```
OBSPY> xlabel 'frequency
(Hz)' OBSPY> plot2
OBSPY> fft
OBSPY> xlim off ylim off
OBSPY> line increment list 1
3 OBSPY> plot2 print
```

12.105 plotalpha

概要

从磁盘读入字符数据型文件到内存并将数据绘制出来

语法

```
PLOTALPHA [MORE] [DIR CURRENT|name] [FREE|FORMAT text] [CONTENT text]
    [PRINT printer] [filelist]
```

输入

MORE 将新读入的文件加到内存中老文件之后。如果没有这个选项，新文件将代替内存中的老文件。参见 read 命令

DIR CURRENT 从当前目录读取并绘制所有文件

DIR name 从文件夹 name 中读取并绘制所有文件，其可以是相对或绝对路径

FREE 以自由格式（以空格分隔数据各字段）读取并绘制 filelist 中的数据文件

FORMAT text 以固定格式读取并绘制 filelist 中的数据文件，格式声明位于 text 中

CONTENT text 定义 filelist 中数据每个字段的含义。text 的含义参见 readtable 命令中的

filelist 字符数字型文件列表，其可以包含简单文件名、绝对/相对路径、通配符。

缺省值

```
plotalpha free content y. dir current
```

说明

参考 readtable 命令的相关说明。该命令与 readtable 之后再 plot 不同，因为它允许你在每个数据点上绘制标签。

示例

读取并绘制一个自由格式的 X-Y 数据，且其第一个字段是标签：

```
OBSPY> plotalpha content lp filea
```

12.106 plotc

概要

使用光标标注 OBSPY 图形和创建图件

语法

```
PLOTC [REPLAY|CREATE] [FILE|MACRO filename] [BORDER ON|OFF]
```

输入

REPLAY 重新显示或绘制一个已经存在的文件或宏，关于文件和宏的区别见下
CREATE 创建一个新的文件或宏
FILE filename 重新显示或创建一个文件。如果省略文件名则使用上一个文件
MACRO filename 重新显示或创建一个宏
BORDER ON|OFF 打开/关闭图形的边界

缺省值

```
plotc create file out border on
```

说明

这个命令让你可以注释一个 OBSPY 绘图或者创建一个用于会议或报告的图件。简单的说就是一 个简单的“画图”软件。你需要一个具有光标的图形设备。你可以通过在终端屏幕上放置一个目标
（比如圆、方）或者文本创建一个图件。光标的位置决定了目标绘制的位置，敲入的字符决定了要绘 制哪个目标。目标包括圆、矩形、多边形、线、箭头、弧，还有多种放置文本的方法。

这个命令绘制出的图可以直接截图利用，绘制过程中用到的所有命令将作为输出文件输出。这 个命令有两种输出文件格式：简单文件或宏文件。两者都是字符数字型文件，可以直接用文本编辑 器修改。它们包含了 plotc 命令中光标响应的历史以及位置。宏文件一旦被创建，可以用于更多的 绘图。其比例和旋转角度均可以修改。简单的 plotc 文件名以 .PCF 为后缀，宏文件名则以 .PCM 为 后缀。这使得你可以区分你的目录下的这些文件。

当你创建一个新的文件或宏时，OBSPY 在屏幕上绘制一个矩形，代表你可以绘图的区域，你可以 将光标移动到该区域内任何你想放置的位置，并输入代表你想绘制的目标或你想要光标执行的动作 所对应的字符。

有两种光标选项类型：动作类和参数设置类。动作类选项将做一些事情（绘制一个矩形，放一些文本），他们如何执行这个操作部分基于当前参数设置选项的值（例如多边形的边数，文本大小等）。 这个区别与 OBSPY 自己的操作命令和参数设定命令相同。下面会列出动作和参数设定选项。

当你重新显示一个文件或者宏时，图形在终端屏幕上将会重新绘制，光标也将打开。你可以像 你当初创建这个文件一样向其中加入目标。当你完成创建图件之后你可以将其发送到不同的图形设 备，使用 begindevices 命令临时关闭终端屏幕打开其他图形设备（比如 sgf），然后重新显示这个 文件。

为了注释一个 OBSPY 绘图，要执行 vspace 命令设置正确的横纵比，然后执行 beginframe 命 令关闭自动刷新，执行需要的 OBSPY 绘图命令，执行 plotc 命令（创建或者重新显示），然后执行 endframe 命令恢复自动刷新。

示例

下面的例子展示了如何使用 plotc 命令给一个 OBSPY 标准绘图添加注释：

```
OBSPY> fg impulse npts 1024                    //生成文件
OBSPY> lp c2 n 7 c 0.2 t 0.25 a                //低通滤波
10 OBSPY> fft
 DC level after DFT is 1
OBSPY> axes only l b                           //左和下坐标轴设置
```

```
OBSPY> ticks only l b
OBSPY>  border
off      OBSPY>
fileid         off
OBSPY> qdp off                          //修改图形尺寸
OBSPY> vspace 0.75                      //开始绘图
OBSPY> beginframe                       //绘图
OBSPY> psp am                           //开始在图上做注释
linlin
OBSPY> plotc create file
```

plotsp 用于绘制滤波响应曲线以及两个轴，plotc 用于交互式地添加注释。vspace 命令限制了图形中纵横比为 3:4 的区域为绘图区域。这个对于之后将输出发送到具有纵横比 3:4 的 SGF 设备来说很有必要。在这之后你将有一个叫做 BANDPASS.PCF 的文件，其中包很了这个图形的注释信息。

为了将注释写入 SGF 文件：

```
OBSPY> begindevices                 // 打开 sgf 设
sgf
OBSPY>
beginframe                          // 重新绘制上一注释图
OBSPY> plotsp
```

这样一个包含注释绘图的 SGF 文件就建立了。

注意

1. 只有当设置正方形视窗（vspace 1.0）时绘制的圆形和扇形才是正确的，否则只能产生一个椭圆，其纵横比等于视窗的纵横比。
2. 除文本之外的所有操作码都按比例适应图形窗口。

文本尺寸并不是当前标度的。当你生成一个图像并想要将文本放在一个矩形或圆中时会产生一个问 题。在这种情况下，图形窗口必须与输出页具有相同的尺寸，以避免图形的偏差。这可以通过使用 window 命令设置窗的水平 X 尺寸为 0.75，垂直 Y 尺寸为 0.69。例如：WINDOW 1 X 0.05 0.80 Y 0.05 0.74。这个命令必须在窗口被创建之前执行。（即在 beginwindow 或 begindevices 之前）

关于 PLOTC 命令表的说明

- CURSOR 表示当前光标位置
- ORIGIN 一般为上次光标的位置
- G 选项强制 ORIGIN 固定
- O 选项再次允许 ORIGIN 移动
- Q 选项不自动拷贝至文件，但是可以通过文本编辑器直接加入

如果 OBSPY 在 replay 模式没有在文件中看到 Q 选项，则其在显示文件内容之后回到光标模式，这使 得你可以在文件结束之后继续增加更多的选项。如果 OBSPY 在文件中看到 Q 选项，则显示其内容并 退出。文件中以星号开头的行为注释行。plotc 还有一些更复杂的选项，但是运行起来好像有点问 题，有兴趣的可以试试 help plotctable。

表 12.1: plotc 命令表

字符	含义
A	绘制一条到 ORIGIN 到 CURSOR 的箭头
B	在绘图区周围绘制边界的 tick 标记
C	绘制一个圆心在 ORIGIN，且经过 CURSOR 的圆
D	从 replay 文件中删除最后一个动作选项
G	设置 ORIGIN，并将其全局化
L	绘制一条从 ORIGIN 到 CURSOR 的线
M	在 CURSOR 处插入一个宏文件（输入宏文件名，比例因子和旋转角。若没有指定，则使用上一次的值，默认是 OUT，1.0，0）
O	设置 ORIGIN 为 CURSOR
N	绘制一个中心在 ORIGIN，一个顶点位于 CURSOR 的 n 边形
Q	退出 PLOTC
R	绘制对脚位于 ORIGIN 和 CURSOR 的长方形
S	绘制一个圆心位于 ORIGIN 的扇形（用光标的移动来指定扇形的角度，键入 S 来绘制一个小于 180 度的扇形，或者键入 C 绘制它的补集）
T	在 CURSOR 处放置一行文本，文本以回车键结束
U	在 CURSOR 处放置多行文本，文本以空白行结束

12.107 plotdy

概要

绘制一个带有误差棒的图

语法

```
PLOTDY [ASPECT ON|OFF] [PRINT pname] name|number [name|number]
```

输入

ASPECT ON|OFF ON 表示保持 3/4 的纵横比。OFF 允许纵横比随着窗口维度的变换而变换

name 数据文件列表中数据名

number 数据文件列表中的数据号

说明

这个命令允许你绘制一个带有误差棒的数据集。你选择的第一个数据文件（通过名字或文件号 指定）将沿着 y 轴绘制第二个数据文件是 dy 值，如果选择了第三个数据文件则其为正的 dy 值

示例

假定你有一个等间距的 ASCII 文件，其包含了两列数据。第一列是 y 值，第二列是 dy 值，你可以像下面那样读入 OBSPY 并用数据绘制误差棒：

```
OBSPY> readtable content yy
myfile OBSPY> plotdy 1 2
```

12.108　plotpk

概要

绘图并拾取震相到时

语法

```
PLOTPK [PERPLOT ON|OFF|n] [BELL ON|OFF] [ABSOLUTE|RELATIVE]
    [REFERENCE ON|OFF|v] [MARKALL ON|OFF] [SAVELOC ON|OFF]
```

输入

PERPLOT n 一张图绘制 n 个文件
PERPLOT ON 一张图绘制 n 个文件，使用 n 的旧值
PERPLOT OFF 一张图上绘制所有文件
BELL ON|OFF 绘图区内击键时是否响铃
ABSOLUTE|RELATIVE 绝对/相对绘图模式
REFERENCE ON|OFF 是否显示参考线
REFERENCE v 显示参考线，并设置参考值为 v
MARKALL ON 一次标记一张图上所有文件的到时
MARKALL OFF 只标记光标位置所对应的震相到时
SAVELOCS ON|OFF 是否将 L 命令（表 4.1）拾取的位置保存到黑板变量中

缺省值

```
plotpk perplot off absolute reference off markall off savelocs off
```

说明

执行该命令后，即进入 ppk 模式，可用于手动拾取震相。在 ppk 模式下，键盘的输入将被解释为 ppk 命令（表 4.1）。详情请参考 震相拾取 一节。

PERPLOT 选项用于控制每张图上显示的文件数目，在标记三分量数据时通常设置为 “p 3”，使得一次显示一个台站的三分量数据。

在一张图上绘制多个波形时，所有波形共用 X 轴，此时存在 absolute 和 relative 两种绘图模式。在 absolute 模式下，所有波形将按照其绝对时刻对齐，即 X 轴表示对于**某个固定参考时刻**的秒数；在 relative 模式下，所有波形将按照自己的参考时刻对齐，即 X 轴表示相对于**各自参考时刻**的秒数。

在标记震相到时时，默认只会标记光标位置所对应的波形文件。MARKALL 选项可以一次标记当前绘图窗口内的所有波形，该选项会获取光标位置所对应的横坐标值，并将其标记到当前绘图窗口内所有波形的头段变量中。该选项常用于一次性拾取单个台站三分量的震相到时，需要注意的是，若三分量数据的参考时刻不同，使用 MARKALL 标记的到时则是错的。

若使用 SAVELOCS 选项，则会将 L 命令所拾取的当前光标位置会保存到黑板变量中：

- NLOCS：每次执行 ppk 命令时，该变量初始化为 0，每使用 L 命令拾取一次光标位置，其值加 1
- XLOCn：保存第 n 次光标拾取的位置的 X 值；若参考时刻 kzdate 和 kztime 有定义，则 XLOCn 保存绝对时刻的值，否则保存时间偏移的秒数
- YLOCn：保存第 n 次光标拾取的位置的 Y 值

BUGS

- BELL 选项在某些平台上不可用
- REFERENCE 选项无效
- 按照说明文档，绘图时默认使用 absolute 模式，但似乎代码存在 Bug。某些情况下默认使用 relative 模式，但在查看若干头段变量之后再绘图，则会使用 absolute 模式。示例代码如下：

```
OBSPY> dg sub local
cdv.z OBSPY> w 1.OBSPY
OBSPY> ch nzsec
50 OBSPY> w
2.OBSPY
OBSPY> r *.OBSPY
OBSPY> ppk          //   第一次绘图为 relative 模式
OBSPY> lh kztime  // 查看头段变量的值
```

12.109 plotpm

概要

针对一对数据文件产生一个 “质点运动” 图

语法

```
PLOTPM
```

说明

在一个质点运动图中，一个等间距文件相对于另一个等间距文件绘图。对于没有自变量的值，第一个文件的因变量的值将沿着 y 轴绘制，第二个文件的因变量值沿着 x 轴绘制。对于一对地震图来说，这种图展示了一个质点在两个地震图所确定的平面内随时间的运动。它将产生一个方形图，每个轴的范围为因变量的极大极小值。注释轴沿着底部和左边绘制。轴标签以及标题可以通过 xlabel、ylabel 和 title 命令设置。如果未设置 x、y 标签，则台站名和方位角将用作轴标签。xlim 可以用于控制文件的哪些部分需要被绘制。

示例

创建一个两个地震图的质点运动图：

```
OBSPY> read xyz.t xyz.r
OBSPY> xlabel 'radial component'
OBSPY> ylabel 'transverse component'
OBSPY> title 'particle-motion plot for station
xyz' OBSPY> plotpm
```

如果你想要值绘制每个文件在初动附近的一部分，你可以使用 xlim 命令：

```
OBSPY> xlim a -0.2
2.0 OBSPY> PLOTPM
```

也可以使用 plotpk 在之前的绘图窗口中设置新的区域，然后绘制命令如下：

```
OBSPY>
beginwindow 2
OBSPY> plotpk
 ... mark the portion you want using X and S
 ... terminate PLOTPK with a Q
OBSPY> beginwindow 1
```

12.110　plotsp

概要

用多种格式绘制谱数据

语法

```
PLOTSP [ASIS|RLIM|AMPH|RL|IM|AM|PH] [LINLIN|LINLOG|LOGLIN|LOGLOG]
```

输入

ASIS　按照谱文件当前格式绘制分量
RLIM　绘制实部和虚部分量
AMPH　绘制振幅和相位分量
RL 只绘制实部分量
IM　只绘制虚部分量
AM 只绘制振幅分量
PH 只绘制相位分量
LINLIN|LINLOG|LOGLIN|LOGLOG　设置 x-y 轴为线型还是对数型，与单独的 `linlin` 等命令区分开

缺省值

```
plotsp asis loglog
```

说明

OBSPY　数据文件可能包含时间序列文件或谱文件，`IFTYPE`　决定文件所有哪种类型。多数绘图命 令只能对时间序列文件起作用，这个命令则可以绘制谱文件。

你可以使用这个命令绘制一或两个分量。每一个分量绘制在一张图上。你也可以设置横纵坐标 为线型或对数型，其仅对该命令有效.

示例

获得一个谱文件振幅的对数-线性的绘图：

```
OBSPY> read
file1 OBSPY>
fft
```

12.111 plotxy

概要

以一个文件为自变量，一个或多个文件为因变量绘图

语法

```
PLOTXY name|number name|number [name|number ...]
```

输入

name 数据文件表中的一个文件名

number 数据文件表中的一个文件号

说明

你选择的第一个文件（通过文件名或文件号）为自变量，沿着X轴绘制。余下的数据为因变量，沿着Y轴绘制。所有的图形环境命令比如 `title`、`line` 和 `symbol` 都可以使用以控制绘图的属性。这个命令用于绘制由 `readtable` 命令读入的多列数据。在多情况下，其可以看作像用绘图命令一样作出的图表。

示例

假设你有一个 ASCII 文件，其包含 4 列数。你想要将其读入 OBSPY 并绘图。下面的命令读入这 个文件，将其储存为 OBSPY 内部 4 个分开的文件，打开线型增量开关，然后以第二列为自变量，其他 列为因变量绘图：

```
OBSPY> readalpha content ynnn
myfile OBSPY> line increment on
OBSPY> plotxy 2 1 3 4
```

12.112 print

概要

打印最近的 SGF 文件

语法

```
PRINT [printer]
```

说明

这个命令会打印最近生成的 SGF 文件，因而在该命令前至少要产生一个 SFG 文件。该命令会将 SGF 文件发送至打印机 printer，如果没有指定则使用系统默认打印机。如果自从登录之后 SGF 设备一直保持关闭，那么 print 命令就不会工作。可以使用 `begindevices` 打开 SGF 设备，使用 SGF 命令设置 SGF 设备的属性。

12.113　printhelp

概要

调用打印机打印帮助文档

语法

```
PRINTHELP [item ...]
```

输入

item　命令、模块、子程序、特性的全称或简称

说明

该命令与 help 命令的用法相同，其调用系统命令 lpr 将文档传递给默认打印机。若 lpr 未设置默认打印机，则报错。

12.114　production

概要

控制生产模式的开关

语法

```
PRODUCTION  ON|OFF
```

输入

ON|OFF　打开/关闭生产模式

缺省值

```
prod off
```

说明

当生产模式处于打开状态时，遇到致命错误将立刻终止 OBSPY 的运行。在生产模式处于关闭状 态时，遇到致命错误后将控制权返回到终端，可以继续执行其他命令。

12.115　qdp

概要

控制低分辨率快速绘图选项

语法

```
QDP [ON|OFF|n] [TERM ON|OFF|n] [SGF ON|OFF|n]
```

输入

ON|OFF 打开/关闭终端和 SGF 设备的 QDP 选项
n 打开终端和 SGF 设备的 QDP 选项，并设定要绘制的点数为 n
TERM ON|OFF 打开/关闭终端 qdp 绘图选项
TERM n 打开终端的 QDP 选项，并设定要绘制的数据点数为 n
SGF ON|OFF 打开/关闭 SGF 设备的 qdp 绘图选项
SGF n 打开 SGF 设备的 QDP 选项，并设定绘制的数据点数为 n

缺省值

```
qdp term 5000 sgf 5000
```

说明

当文件中的数据点数很多的时候，绘制波形要花费很长的时间。“quick and dirty plot” 选项绘制数据文件的部分数据点的方式来加速绘图。

当打开 QDP 选项时，OBSPY 用文件的数据点数除以 QDP 中的指定的数据点，由此计算每个子 区间所包含的数据点数。文件越大，每个子区间中数据点就越多，然后计算并绘制每个子区间内最小和最大数据点，同时在绘图的右下角的矩形框中显示减采样因子。实际显示的数据点数可能与该 值所表示的数据点数有所偏差。

以目前计算机的性能而言，大型文件的绘制基本都是瞬间完成的，所以一般都设置关闭此选项。

示例

假设文件 FILE1 有 20000 个数据点，文件 FILE2 有 40000 个数据点，如果你输入：

```
obspy> r file1
file2 obspy> p
```

那么两张图都将包含 5000 个点。对第一个文件每 4 个点取一个点用于绘图，第二个文件每 8 个点取一个点绘图。

如果想要绘制全部数据点，则需要关闭 QDP 选项：

```
OBSPY> qdp
off OBSPY> p
```

12.116 quantize

概要

将连续数据数字化 [1]

语法

```
QUANTIZE [GAINS n ...] [LEVEL v] [MANTISSA n]
```

[1] 无法理解这个命令，请阅读官方文档

输入

GAINS n ... 设置允许的增益表，必须单调递减的，增益的最大数目为 8
LEVEL v 设置最低增益的量化水平，即 Least Significant Bit，单位为伏特
MANTISSA n 设置 Bits，用其尾数表示。

缺省值

```
quantize gains 128 32 8 1 level 0.00001 mantissa 14
```

说明

此命令演示了与 Oppenheim 和 Schafer(1975, Fig. 9.1) 描述的 “rounding” 量化算法类似的一个量化算法。

算法中的位数通过其尾数表示，则若尾数 mantissa 为 n，则表示其所能允许的范围是 $\pm\frac{2^n-1}{2}$。

量化水平 level 即 least significant bit 的值，表征了系统所能识别的最小的电压变化，缺省值为 10 微伏。

这个量化函数描述的信号误差是量化水平的一半。在频率域中，这个误差或量化噪声为：

$$Err = \frac{1}{12}(Delta * Level^2)$$

其中 Delta 是采样周期。量化噪声的单位是 counts * counts/Hz，相当于功率谱密度。量化噪声的均方根为 $\frac{1}{6}Level^2$。然而，仅当信号的均方根水平远大于量化噪声的均方根值时，才是量化噪声的一个精确的近似。

头段变量

depmin、depmax、depmen

12.117 quit

概要

退出 OBSPY

语法

```
QUIT
```

说明

该命令用于正常退出 OBSPY。在退出 OBSPY 之前，做了如下清理工作：

- 关闭图形设备
- 关闭 pick 文件
- 清空已定义的黑白变量
- 将命令历史写到命令历史文件中
- 释放内存

12.118 quitsub

概要

退出子程序

语法

```
QUITSUB
```

说明

该命令用于终止当前活动的子程序，返回到 OBSPY 主程序，内存中的文件保留。

目前 OBSPY 提供了两个子程序：

- Spectral Estimation Subproess 谱估计子程序（SPE）
- Signal Stacking Subprocess 信号迭加子程序（SSS）

12.119 read

概要

从磁盘读取 OBSPY 文件到内存

语法

```
READ [MORE] [DIR CURRENT|name] [XDR|ALPHA|SEGY]
    [SCALE ON|OFF] [filelist]
```

所有的选项必须位于 filelist 之前。

输入

MORE 将读入的新文件添加到内存中老文件之后。若选项此忽略，则读入的新数据将替代内 存中的老数据

DIR CURRENT 从“当前目录”读取文件列表中的文件。“当前目录”为启动 OBSPY 的目录

DIR name 从目录 name 中读取文件列表中的文件，可以为绝对路径或相对路径 [1]

XDR 读取 XDR 格式的文件。此格式用于实现不同构架的二进制数据的转换

ALPHA 输入文件是 OBSPY 的字符数字型文件，该选项与 XDR 选项不兼容

SEGY 读取 IRIS/PASSCAL 定义的 SEGY 格式文件。该格式允许一个文件包含一个波形

SCALE 只能和 SEGY 选项搭配使用，该选项默认是关闭的。当 SCALE 选项为 OFF 时，OBSPY 直接从 SEGY 文件中读取数据值；当 SCALE 为 ON 时，OBSPY 将每个数据值乘以以文件中给 定的 SCALE 因子。若 SCALE 为 OFF，则这个文件中的 SCALE 值将储存在 OBSPY 头段 SCALE 中；若 SCALE 为 ON，OBSPY 的 SCALE 头段将被设置为 1.0。

filelist 文件列表。可以是简单的文件名，也可以包含相对或绝对路径，也可以使用通配符

缺省值

```
read dir current
```

[1] 关于 dir 选项，有一个很大的陷阱，详见“读取目录下的 OBSPY 文件”。

说明

该命令将 OBSPY 文件从磁盘读入到内存中，默认状态下会读取每个磁盘文件中的全部数据点。

cut 命令可以用于指定读取文件的一部分数据。在 2000 年之后产生的 OBSPY 文件会被假定年份为四位数字。年份为两个数字的文件被假定为 20 世纪，会被加上 1900。

在使用 read 命令时，正常情况下内存中的老数据会被新读取的数据所替代。若使用 more 选项，则新数据将被读入内存并放在老数据的后面。在如下三种情况下 more 选项可能会有用：

- 文件列表太长无法在一行中键入
- 在长文件列表中某个文件名拼错而没有读入，可以使用 more 选项再次读入
- 一个文件被读入，作了些处理，然后与原始数据比较

示例

read 命令的简单示例位于 OBSPY 的读和写 一节。 如果你想要对一个数据进行高通滤波，并与原始数据进行对比：

```
OBSPY> r f01
OBSPY> hp c 1.3
n 6 OBSPY> r
more f01 OBSPY>
```

假设 OBSPY 的启动目录位于 /me/data，你想要处理其子目录 event1 和 event2 下的文件。

```
OBSPY> read dir event1 f01 f02
```

读取了目录 /me/data/event1 下的文件。

```
OBSPY> read f03 g03
```

相同目录下的文件被读入。

```
OBSPY> read dir event2 *
```

/me/data/event2 下的全部文件被读入。

```
OBSPY> read dir current f03 g03
```

目录 /me/data 下的文件被读入。

头段变量

e、depmin、depmax、depmen、b

12.120 readbbf

概要

将黑板变量文件读入内存

语法

```
READBBF [file]
```

输入

file 黑板变量文件名，可以是简单文件名或相对/绝对路径

缺省值

```
readbbf bbf
```

说明

该命令使你能够读取一个黑板变量文件。该文件必须是先前通过 writebbf 命令写入磁盘的文件。该特性让你能够将某个 OBSPY 会话的信息保存起来并用在另一次 OBSPY 会话中。也可以在自己的 程序中调用 OBSPY 函数库以读取黑板变量文件中的黑板变量，这使得你可以在 OBSPY 和自己的程序 之间互相传送信息。

12.121 readcss

概要

从磁盘读取 CSS 格式的文件到内存

语法

```
READCSS [BINARY|ASCII] [MAXMEM v] [MORE] [TRUST ON|OFF]
    [VERBOSE ON|OFF] [SHIFT ON|OFF] [SCALE ON|OFF]
    [MAGNITUDE MB|MS|ML|DEF] [DIR name] wfdisclist [filelist]
    [cssoptions]
```

其中 cssoptions 用于进一步从 wfdisc 文件中筛选满足条件的数据文件，cssoptions 可以取：

```
    [STATION station] [CHANNEL channel] [BANDWIDTH bandcode]
    [ORIENTATION  orientation-code]
```

输入

ASCII 读取 ASCII 形式的 CSS 文件（默认值）

BINARY 读取二进制 CSS 文件，阅读 writecss 以了解更多信息

MAXMEM 设定读取大量数据时所能使用的最大内存占物理内存的百分比。当使用的内存达到设定的上限时，即使已经读取了其他数据库表，也不会再读取更多的波形数据。MAXMEN 的默认值是0.3。

MORE 将读入的波形数据放在内存中的原有波形之后，若不使用该选项，则新读入的波形数 据会覆盖内存中的原有波形数据，详情参考 read 命令。

VERBOSE ON|OFF 如果 VERBOSE 是 ON，OBSPY 会显示正在读取的波形数据的扩展信息，并打印出 CSS 数据库表的概要信息以及数据格式转换的进度信息。

SHIFT ON|OFF 若 SHIFT 是 ON，则发震时刻将被设置为 0，其他相关时间头段变量也会做相应修改。与震中距相关的一些头段变量也会受影响。默认值为 SHIFT ON。

SCALE ON|OFF SCALE 选项的默认值是 OFF。在 wfdisc 文件中，有一个字段为校准因子 CALIB。当 SCALE 选项是 OFF 时，OBSPY 直接从 .w 文件读取数字信号数据，此时数据的单位 是 counts，并将 CALIB 的值保存到 OBSPY 头段变量 SCALE 中。当 SCALE 选项是 ON 时，OBSPY 会给读取的数据乘以 CALIB 值，并设置 OBSPY 的头段变量 SCALE 的值为1.0。设置 SCALE ON， 将数据乘以 CALIB 值，在某种程度上可以认为是对数据去除了仪器响应，但该方法很粗糙，

完整地去除仪器响应应使用 transfer 命令。仅当 transfer 命令所需的仪器响应信息无法 获取时，才建议使用 SCALE ON。

MAGNITUDE 指定要将哪一种震级放在 OBSPY 的头段变量 mag 中。Mb 是体波震级，Ms 是 面波震级，ML 是地方震震级。默认值是 DEF，其算法为：若 Ms 存在且大于或等于 6.6，则最 优先用 Ms。否则，如果 Mb 存在，用 Mb。如果 Mb 不存在，而 Ms 存在，用 Ms。其他情况用 ML。

DIR name wfdisc 文件所在的路径

wfdiscfiles wfdisc 文件列表

filelist 若不指定 filelist，则 wfdisc 文件所包含的所有波形数据都会被读入内存；若指定了 filelist，则只有 filelist 中指定的波形数据才会被读取内存。需要注意，filelist 所指定的波形文件名必须位于之前指定的 wfdisc 文件中。

STATION station station 是一个 6 个或更少字符构成的字符串。wfdisc 文件中台站名 kstnm 与 station 匹配的行会被选中并读取。station 中可以包含通配符 * 和 ? 。

CHANNEL channel channel 是一个 8 个或更少字符构成的字符串。wfdisc 文件中通道 名与 channel 匹配的行会被选中并读取。channel 中可以包含通配符 * 和 ? 。

BANDWIDTH bandcode 单字符编码。常见的取值为 E、S、H、B、M、L、V、U、R 等。bandcode 的具体含义参考附录中表 C.1。channel 字段中第一个字符与 bandcode 匹配的行会被选择并读取。bandcode 中使用通配符 * 会匹配所有 bandcode。

ORIENTATION orientation-code orientation-code 通常可以取 “Z N E”（表示竖直、 北和东）、“1 2 3”（表示正交但非标准的三个方向）。channel 字段中最后一个字符与 orientatio-code 相匹配的行会被选中并读取。orientation-code 使用通配符 * 会匹配所有 orientation-code。

默认值

```
readcss ascii maxmem 0.3 verbose off station * band * chan * orient
```

说明

CSS 是一种数据库架构，该命令可以读取 CSS 3.0 或 CSS 2.8 中的文件。

每个 CSS 数据库包含了若干个数据库表表，每个数据库表包含若干个记录。对于 CSS 3.0 而 言，该命令支持读取如下数据库表：wfdisc、wftag、origin、arrival、assoc、sitechan、site、affiliation、origerr、origin、event、sensor、instrument、gregion、stassoc 和 remark Obspydata。对于 CSS 2.8 而言，该命令只支持表 wfdisc、arrival 和 origin。

关于 CSS 格式的详细介绍，请参考：

- https://anf.ucsd.edu/pdf/css30.pdf
- http://prod.sandia.gov/techlib/access-control.cgi/2002/023055.pdf
- ftp://ftp.pmel.noaa.gov/newport/lau/tphase/data/css_wfdisc.pdf

在 CSS 数据库的众多表中，最常用的是与波形相关的 wfdisc 表以及波形数据 .w 文件。wfdisc 表中每行代表一个波形记录，共 19 列，每列代表了波形记录的不同信息。详情参考上面列出的格式说明文档。

readcss 命令的 BINARY 选项，可以用于读取 writecss 命令生成的二进制 CSS 格式。在 BINARY 模式下，cssoptions 选项没有作用，即 wfdisc 文件中包含的全部波形数据都会被读取。

12.122 readerr

概要

控制在执行 read 命令过程中的错误的处理方式

语法

```
READERR [BADFILE FATAL|WARNING|IGNORE] [NOFILES FATAL|WARNING|IGNORE]
        [MEMORY SAVE|DELETE]
```

输入

BADFILE 当文件不可读或不存在时出现的错误
NOFILES 文件列表中没有文件可读时出现的错误
FATAL 设置错误条件为 fatal，发送错误消息并停止执行命令
WARNING 发送警告消息，但继续执行命令
IGNORE 忽略错误，继续执行命令
MEMORY 如果无文件可读则对内存中原有的数据进行处理
DELETE 内存中的原数据将被删除
SAVE 内存中的原数据将保留在内存中

缺省值

```
readerr badfile warning nofiles fatal memory delete
```

说明

当你试着使用 read 命令将数据文件读入内存时可能会发生错误。文件可能不存在或虽然存在但不可读。当 OBSPY 遇到这些 badfiles 时，一般会发送警告消息，然后试着读取文件列表中的其余文 件。如果你想要OBSPY在遇到坏文件时停止读取文件可以设置BADFILE为FATAL。如果你不想看到 警告信息，可以设置 BADFILE 为 INGORE。如果文件列表中的文件均不可读，OBSPY 将发送错误信息 并停止处理，如果你想要 OBSPY 发送警告信息或完全忽略这个问题，设置 NOFILES 为 INGORE。当然， OBSPY 内存中先前的文件也可以从内存中删除或者保留在内存中。

12.123 readhdr

概要

从 OBSPY 数据文件中读取头段到内存

语法

```
READHDR [MORE] [DIR CURRENT|name] [filelist]
```

输入

MORE 将新数据头段放在内存中老文件头段之后。若忽略，则新数据文件的头段将代替内存 中原文件的头段
DIR CURRENT 从当前目录读取文件。这里的当前目录是指启动 OBSPY 的目录
DIR name 从目录 name 中读取文件，目录名可以是绝对路径或相对路径

filelist 文件名列表。其可以是简单文件名也可以使用通配符，路径名可以是相对路径或绝对路径

说明

这个命令将一系列 OBSPY 文件的头段读入内存，你可以列出头段内容（listhdr）、改变头段值（chnhdr）、将头段写回磁盘（writehdr）。当你只需要文件的头段的时候，只读取头段要比读取整个文件到内存快很多。

12.124 readsp

概要

读取 writesp 和 writespe 写的谱文件

语法

```
READSP [AMPH|RLIM|SPE] [filelist]
```

输入

RLIM 读入实部和虚部分量
AMPH 读入振幅和相位分量
SPE 读取谱估计子程序文件，这个数据被从功率转换为振幅，相位分量设置为 0
filelist OBSPY 二进制数据文件列表

缺省值

```
READSP  AMPH
```

说明

writesp 命令将每个谱数据分量作为一个单独的文件写入磁盘，你可以分别处理每个分量。这个命令让你能从两个分量重建谱数据，参见 writesp。SPE 选项允许你读取并转换由 writespe 写出的谱文件格式。这也使你可以使用 mulomega 和 divomega 命令。

12.125 readtable

概要

从磁盘读取列数据文件到内存

语法

```
READTABLE [MORE] [DIR CURRENT|name] [FREE|FORMAT tex]
    [CONTENT text] [HEADER number] [filelist]
```

所有的选项必须位于 filelist 之前。最后两个选项可以放在每个文件的第一行。

输入

MORE 将新文件追加到内存中老文件之后。若忽略该选项，则新数据将替代内存中的老数据
DIR CURRENT 从当前目录读取所有简单文件名。当前目录是你启动 OBSPY 的目录

DIR name 从目录 name 中读取全部简单文件，其可以为绝对/相对路径
FREE 用自由格式读取文件列表中的数据（以空格分隔）
FORMAT text 以固定格式读取文件列表中的数据。该选项目前不可用
CONTENT text 定义数据内容。text 的具体格式见说明及示例
HEADER 文件中要跳过的几个头段行
filelist 列数据文件

缺省值

```
readtable free content y. dir current
```

说明

该命令可以读取字符型列数据。最简单的用法就是读取一个Y数据，也可以通过修改content的内容读入 X-Y 数据或更复杂的数据。因而该命令可以用于直接读取其他程序输出的复杂格式数据。也可以用这个方法读入多个 Y 数据集，但只允许一个 X 数据集。

读入数据时会计算基本的头段变量，包括 npts、b、e、delta、leven、depmin、depmax 和 depmin。若只有一个Y数据集，内存中的数据文件名将和磁盘文件名相同；若有多个Y数据集，则在文件名之加上一个两位数字。

字符数字型数据文件的每一行都将以自由格式或声明的格式读入，每行最多 160 个字符。content 选项用于决定对于数据每行的每个输入该如果处理。在 content text 中的每个字符分别代表了不同的数据元素，这些字符的顺序与数据中每行的输入所代表的含义相对应。content 字段允许的字符如下：

- Y：下一个输入属于Y（因变量）数据集
- X：下一个输入属于X（自变量）数据集
- N：下一个输入属于数据集
- P：下一对输入使用 X-Y 数据集
- R：下一对输入使用 Y-X 数据集
- I：忽略这个输入

还有一个重复计数器可以跟在上面的任何字符之后。这个重复计数器是一个 1 位或 2 位整数，其代表重复前面那个字符多少次，“.”是一个无穷次重复的计数器，其只能出现在 content 的 text 的最后，意味着最后一个字符可以表示接下来的所有输入列。

示例

为了读取一个或多个自由格式的 X-Y 数据对：

```
OBSPY> readtable content p. filea
```

你不能在文件行之间打断一个 X-Y 数据对。假设你有一个包含了格式化数据的文件，在每行的中间有一个 X-Y 数据对。每行的其它数据都没有用。假设每行 Y 数据在 X 数据之前，一旦正确的格式声明给出了，就可以用下面的命令：

```
OBSPY> readtable content r format \(24x,f12.3,14x,f10.2\) fileb
```

注意：在左括号和右括号两边的“\”是 OBSPY 的转义字符，这很重要，因为 OBSPY 使用括号作为内联 函数。由于没有重复计数器，因而只有一个 Y-X 数据对被从文件的每行读入。

假设你有一个文件 FILEC，其每行包括一个 X 值和 7 个不同数据集的 Y 值，其为 (8F10.2) 格式。为了在内存中创建 7 个不同的数据集，可以使用下面的命令：

```
OBSPY> readtable content xn . format \(8f10.2\) filec
```

这将在内存中产生 7 个不同的数据文件，其名称分别为 FILEC01、FILEC02 等等。 现在假设你不想读入第 5 个 Y 数据集，可以执行下面的命令：

```
OBSPY> readtable content xn6 format \(5f10.20x,2f10.2\) filec
```

另一个可以少敲键盘但是稍微低效一点的命令如下：

```
OBSPY> readtable content xn4in2 format \(8f10.2\) filec
```

头段变量改变

b、e、delta、leven、depmin、depmax、depmen

12.126 report

概要

报告 OBSPY 中的各参数的当前状态

语法

```
REPORT APF|COLOR|CUT|DEVICES|FILEID|GTEXT|HPF|LINE|MEMORY|MTW|PICKS|
    SYMBOL|TITLE|XLABEL|XLIM|YLABEL|YLIM
```

输入

APF 字符数字型震相拾取文件的文件名
COLOR 当前颜色属性。只有当某个图像设备被激活时，其值才有意义
CUT 当前数据截窗的状态
DEVICES 当前系统上可用的图形设备
FILEID 当前文件 ID 显示属性
GTEXT 当前图形文本属性
HPF HYPO 震相拾取文件名
LINE 当前线型属性
MEMORY 内存管理器的可用内存块
MTW 当前的测量时间窗状态
PICKS 当前时间拾取显示属性
SYMBOL 当前符号绘制属性
TITLE 当前绘制标题属性
XLABEL 当前 x 轴标签属性
XLIM 当前 x 轴范围
YLABEL 当前 y 轴标签属性
YLIM 当前 y 轴范围

说明

该命令会报告 OBSPY 的某些选项的当前值，并将其值打印到终端。

示例

为了获取当前颜色属性的列表：

```
OBSPY> report color
 COLOR option is ON
 DATA color is YELLOW
 INCREMENT data color is OFF
 SKELETON color is BLUE
BACKGROUND color is NORMAL
```

为了获取 HYPO 文件名：

```
OBSPY> report apf hpf
 Alphanumeric pick file is MYPICKFILE
 HYPO pick file is HYPOPICKFILE
```

12.127 reverse

概要

将所有数据点逆序

语法

```
REVERSE
```

12.128 rglitches

概要

去掉信号中的坏点

语法

```
RGLITCHES [THRESHOLD v] [TYPE LINEAR|ZERO] [WINDOW ON|OFF|pdw]
    [METHOD ABSOLUTE|POWER|RUNAVG]
```

输入

WINDOW ON|OFF|pdw 指定需要做校正的数据段。缺省值为 OFF，即校正整个数据文件；pdw 指定了时间窗，表示仅对该时间窗内的数据进行检测和校正；ON 表示只校正上一次 pdw 定义的时间窗内的数据；

THRESHOLD v 设置阈值水平为 v。当某个特定的指标大于该阈值时即认为是坏点，并做校正

METHOD ABSOLUTE 若数据点的绝对值大于或等值阈值 v，则做校正

METHOD POWER 用向后差分法计算信号的能量，若能量超过阈值 v 则做校正

METHOD RUNAVG 将一个长为 SWINLEN 的窗从整个数据的末尾以一个数据点的增量移动到数据的开头，并计算每个滑动窗内的平均值和标准差。若某个点与当前窗的均值的差的绝对值大于标准差 THRESH 的 2 倍，且大于 MINAMP，则认为其是坏点，其值将用当前窗的均值代替。此方法不受 WINDOW 选项的影响，总是适用于整段波形数据。

对于 RUNAVG 方法，另有三个与之相关的选项：

SWINLEN v　设置滑动平均窗的长度为 v
THRESH2 v　设置坏点的阀值
MINAMP v　设置坏点的最小幅度

缺省值

```
rglitches threshold 1.0e+10 type linear window off method absolute
    swinlen 0.5 thresh2 5.0 minamp 50
```

说明

此命令可以用于由于平滑数据采集系统故障或测试[1]而产生的坏点。该命令会检查每一个数据点是否超过指定的阈值，然后将这些坏点置零或者线性插值。使用 RUNAVG 方法，甚至可以去掉那些比整个数据的最大值要小的坏点。

头段变量

depmin、depmax、depmen

12.129　rmean

概要

去除均值

语法

```
RMEAN
```

头段变量

depmen、depmin、depmax

12.130　rms

概要

计算测量时间窗的信号的均方根

语法

```
RMS [NOISE ON|OFF|pdw] [TO USERn]
```

输入

NOISE ON　打开噪声归一化选项
NOISE OFF　关闭噪声归一化选项
NOISE pdw　打开噪声归一化选项并设置噪声的测量时间窗 pdw
TO USERn　将测量结果存储到头段变量 USERn 中

[1] 有些台网会在每天的指定时间生成若干个人工脉冲，以检测数据采集系统是否正常运行。

缺省值

```
rms noise off to user0
```

说明

该命令用于计算当前测量时间窗（由 mtw 定义）内的数据的均方根，并将测量结果写入头段变量USERn中。

均方根的定义为：

$$RMS = \sqrt{\frac{1}{N}\sum_{i=1}^{n} y_i^2}$$

NOISE 选项定义了噪声的测量时间窗，用于计算噪声的均方根，以对信号的均方根做修正。若 使用该选项，则会首先计算 mtw 定义的时间窗内的数据点的平方和的均值，然后再计算 NOISE pdw 定义的时间窗内数据点的平方和的均值，最后将信号的平方和均值减去噪声的平方和均值，并取其 平方根作为测量结果。

示例

为了计算两个头段T1和T2间的数据的未修正的均方根，并将结果保存在头段USER4中：

```
OBSPY> mtw t1 t2
OBSPY> rms to
```

将T3前5秒作为噪声窗，计算修正后的均方根：

```
OBSPY> mtw t1 t2
OBSPY> rms noise t3 -5.0 0.0
```

头段变量

USERn

12.131 rotate

概要

将成对的正交分量旋转一个角度

语法

```
ROTATE [TO GCP|TO v|THROUGH v] [NORMAL|REVERSED]
```

输入

TO GCP 旋转到大圆弧路径（“great circle path”）。两个分量必须都是水平分量且头段中必须定义台站和事件的经纬度

TO v 旋转一定角度使得第一个分量的方位角为 v 度。两个分量必须都是水平分量

THROUGH v 顺时针旋转 v 度。其中一个分量可以是垂直分量

NORMAL|REVERSED 输出分量为正/负极性

缺省值

```
rotate to gcp normal
```

说明

此命令可以对多对分量旋转一定的角度，内存中的每两个文件为一对分量。每对分量必须拥有 相同的台站名、事件名、采样率和数据点数，且头段变量 cmpaz 和 cmpinc 必须定义，程序会检查两 个分量是否正交（允许0.02度的偏差）。

THROUGH 选项表示将一对正交分量旋转一定的角度。这对正交分量可以均为水平分量（cmpinc=90）或包含一个垂直分量（cmpinc=0）。其中，水平面内的旋转是相对于北向顺时针的角度；垂直面内的 旋转是相对于垂直向上方向的角度。

TO 选项表示将一对正交分量旋转**到**一定的角度（方位角），这对正交分量必须都是水平分量（cmpinc=90）。对于 TO v 而言，表示将一对分量旋转一定角度，使得旋转后的第一个分量沿着方位角 v 的方向；对 TO GCP 而言，首先根据台站和事件经纬度计算出反方位角，并将分量旋转一定角度，使得旋转后的第一个分量沿着反方位角加/减 180 度的方向，此时第一个水平分量由事件位置指向台站位置，即地震学中的径向（Radial）分量，一般方位码用 R 表示；第二个水平分量垂直于 R 分量，即地震学的切向（Tangential）分量，一般方位码用 T 表示。

NORMAL 和 REVERSED 用于指定输出分量的极性，仅用于一对水平分量的旋转中。在 rotate 命令中，就一对水平分量而言，若第二个分量比第一个分量超前 90 度（可以理解为方位角大 90 度）则称为正极性；若第二个分量比第一个分量落后 90 度则称为负极性。对于一对输入分量而言，无论是 N 分量在前还是 E 分量在前均可，该命令会自动判断一对输入分量是正极性还是负极性，并对旋转公式进行调整，NORMAL 和 REVERSED 仅用于控制一对输出分量的极性。

示例

将一对水平分量旋转 30 度：

```
OBSPY> dg sub tele                    //  内存中的顺序是 E 分量先于 N 分
ntkl.[ne]


  FILE:  /opt/obspy/aux/datagen/teleseis/ntkl.e  -
  1
 ----------------------------------------


     cmpinc = 9.000000e+01
      cmpaz = 9.000000e+01


  FILE:  /opt/obspy/aux/datagen/teleseis/ntkl.n  -


     cmpinc = 9.000000e+01
      cmpaz = 0.000000e+00
OBSPY> rot through 30                 // 顺时针旋转 30
OBSPY> lh


  FILE:  /opt/obspy/aux/datagen/teleseis/ntkl.e  -
  1
 ----------------------------------------


     cmpinc = 9.000000e+01
```

```
 FILE:  /opt/obspy/aux/datagen/teleseis/ntkl.n  -  2
----------------------------------------

    cmpinc = 9.000000e+01
     cmpaz = 3.000000e+01
```

旋转两对水平分量到大圆弧路径：

```
OBSPY> read abc.n abc.e def.n
def.e OBSPY> rotate to gcp
OBSPY> w abc.r abc.t def.r def.t
```

上面的例子中若头段变量 baz 为 33 度，则径向分量指向 213 度，切向分量指向 303 度，如果设置反极性，切向分量指向 123 度。

头段变量

cmpaz、cmpinc

12.132 rq

概要

从谱文件中去除 Q 因子

语法

```
RQ [Q v] [R v] [C v]
```

输入

Q v 设置质量因子为 v
R v 设置距离为 v，单位为 km
C v 设置群速度为 v，单位 km/s

缺省值

```
rq q 1. r 0. c 1.
```

说明

该命令用于从波形数据中去除 Q 衰减效应，用于校正振幅谱的方程如下：

$$AMP_{corrected}(f) = AMP_{uncorrected}(f) * e^{\frac{\pi R f}{QC}}$$

其中 f 为频率，单位为 Hz，R 为距离，单位 km，C 是群速度，单位为 km/s。Q 是一个无量纲衰减因子。

头段变量

depmin、depmax、depmen

限制

实际上各参数应是频率的函数，目前限制为常数值。

12.133　rtrend

概要

去除线性趋势

语法

```
RTREND  [QUIET|VERBOSE]
```

输入

QUIET 不显示线性拟合信息
VERBOSE 终端显示线性拟合信息

缺省值

```
rtrend  quiet
```

说明

该命令利用最小二乘方法将数据拟合成一条直线，然后从数据中减去该直线所表征的线性趋势。数据可以是不等间隔的。

若有 n 个数据 (x_i, y_i)，利用最小二乘法拟合直线 $y = ax + b$。其中斜率为

$$a = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{n\sum x_i^2 - (\sum x_i)^2}$$

Y 轴截距为

$$b = \frac{\sum x_i^2 \sum y_i - \sum x_i \sum x_i y_i}{n\sum x_i^2 - (\sum x_i)^2}$$

内存中最后一个文件的线性拟合参数将会写入到如下黑板变量中：

- RTR_SLP 直线斜率
- RTR_YINT 直线的 Y 截距
- RTR_SDSLP 斜率的标准差
- RTR_SDYINT Y 截距的标准差
- RTR_SDDTA 数据的标准差
- RTR_CORRCF 数据和拟合结果的相关系数

头段变量改变

depmin、depmax、depmin

12.134　saveimg

概要

将绘图窗口中的图像保存到多种格式的图像文件中

语法

```
SAVEIMG  filename.format
```

输入

filename 要保存的图像的文件名

format 图像文件格式，支持 PS、PDF、PNG 和 XPM

说明

该命令将当前绘图窗口中的图像保存到图像文件中，可用的格式包 PS、PDF、PNG 和 XPM，命令会根据图像文件的扩展名自动识别文件格式。

saveimg 相对于 SGF 文件的好处在于，SGF 文件中的字母和数字是由线段组成的，而 saveimg 产生的 ps 或 pdf 图像采用 Postscript 的特性直接产生字体。对大多数情况，低精度的 png 或 xpm 文件也能满足要求。出于可移植性的考虑，OBSPY 默认是不支持 PNG 格式的。

png 和 xpm 将拥有当前窗口的横纵比，pdf 或 ps 文件拥有固定的横纵比 X/Y=11/8.5=1.2941，对这些绘图，如果显示窗口设置为 1.2941 会看起来比较好。

与 SGF 文件类似，saveimg 生成的 PS 文件是没有 BoundingBox 的。对于 sgf 文件，脚本 sgftoeps.csh 可以生成一个有 BoundingBox 的 eps 文件。对于 saveimg 生成的 PS 文件，目前还没有相应的脚本。

为了使用 saveimg 保存一个绘图，图像必须是可见的，即通过 plot、plot1 等命令绘制出来。saveimg 在子程序 SSS 中无法工作，但如果输入 quitsub 退出子程序，此时图像窗口未关闭，saveimg 此时可用于保存该图像。另外如果使用了 beginframe 命令在一个窗口中绘制多张图，必须等到执行 endframe 命令之后方可使用 saveimg 命令。

示例

将图像保存为 PDF 文件：

```
OBSPY> read
PAS.CI.BHZ.obspy OBSPY>
p1
```

将谱图用多种格式保存：

```
OBSPY> fg seismo
OBSPY>
spectrogram
OBSPY> save
spectrogram.ps OBSPY>
```

12.135 setbb

概要

设置黑板变量的值

语法

```
SETBB variable [APPEND] value [variable [APPEND] value ...]
```

输入

variable 黑板变量名，可以是一个新变量或一个已经有值的变量，变量名最长 32 字符
value 黑板变量的新值，若包含空格则必须用引号括起来
APPEND 将值加到变量的旧值之后，若无该选项，则新值将代替旧值

说明

setbb 命令可以给黑板变量赋值，这些值可以通过 getbb 命令获取，或在命令中直接引用。可以使用 evaluate 对黑板变量做基本算术操作，并将结果保存在新的黑板变量中，也可以通过 unsetbb 命令删除一个黑板变量。

示例

同时设置多个黑板变量，并在稍后使用使用这些黑板变量：

```
OBSPY> setbb c1 2.45 c2 4.94
OBSPY> bandpass corners %c1% %c2%
```

黑板变量的值中包含空格：

```
                                // 含空格的值需要用引号括起来
OBSPY> setbb mytitle 'sample filter
response' OBSPY> getbb mytitle // 检查变量值
是否正确 MYTITLE = Sample filter response
OBSPY> title '%MYTITLE%' // 引用时需要再次用引用括起来
```

12.136 setdevice

概要

定义后续绘图时使用的默认图形设备

语法

```
SETDEVICE name
```

输入

name 图形设备名

说明

OBSPY 的默认图形设备为 X11 图形窗口，你可以使用该命令修改默认的图形设备，也可以使用 begindevices 命令随时用新的图形设备取代默认图形设备。

BUGS

使用该命令似乎会直接段错误退出。

12.137 setmacro

概要

定义 OBSPY 宏文件的搜索路径

语法

```
SETMACRO [MORE] directory [directory ...]
```

输入

directory 放置 OBSPY 宏文件的目录，可以是相对路径或绝对路径
MORE 将 directory 加到已有的路径列表之后

说明

该命令让你能够定义一系列执行宏文件时搜索路径，最多可以定义 100 个。

当 setmacro 使用 more 选项时，指定的路径会追加到已存在的路径列表的后面；若没有使用 more 选项，则已存在的列表将被新列表取代。

当执行 macro 命令时，OBSPY 会先搜索当前目录，若没有找到则搜索 setmacro 指定的目录，若 依然没有找到则在全局宏目录中寻找。

12.138 sgf

概要

控制 SGF 设备选项

语法

```
SGF [PREFIX text] [NUMBER n] [DIRECTORY CURRENT|pathname]
    [SIZE NORMAL|FIXED v|SCALED v] [OVERWRITE ON|OFF]
```

输入

PREFIX text 设置 SGF 的文件名前缀为 text（最多 24 字符长）
NUMBER n 设置下一个 SGF 的编号为 n。若 n 为 0，则 OBSPY 搜索 SGF 文件目录下的 SGF 文件最大编号，并将其值加 1
DIRECTORY CURRENT 将 SGF 文件放在当前目录
DIRECTORY pathname 将 SGF 放在指定目录下
SIZE NORMAL 产生一个常规大小绘图，常规图形有一个 10*7.5 英寸的 viewspace（最大绘图区域）。viewport（viewspace 中除轴和标签之外的、绘图区）的默认值为 8*5 英寸
SIZE FIXED v 产生一个在 X 方向 viewport 为 v 英寸长的图形
SIZE SCALED v 产生一个视口在 X 方向上为 v 乘以 X 坐标极限值的图形
OVERWRITE ON|OFF 当打开时，文件号不递增，每个新文件将擦除先前的文件

缺省值

```
sgf prefix f number 1 directory current size normal
```

说明

该命令控制 SGF 文件的命名规则和后续 SGF 文件的图形尺寸。SGF 文件名由 4 部分组成，分别为：

- pathname 目录路径名，默认为当前目录
- prefix 前缀，默认值为 f

- `number` 三位数的 frame 编号，默认值为 `001`
- `.sgf` 用于表示 OBSPY 图形文件的后缀

因而 SGF 的第一个文件名为 `f001.sgf`。每次新建一个 frame 时，frame 编号加 1。可以强制 frame 编号从某个给定的值开始。如果你为准备一个报告需要工作几天时间，而同时又希望所有图形都按 一个统一的顺序排列，那么令 frame 编号不从 1 开始便很有用了。

有多个选项可以控制绘图的尺寸，常规的绘图为 10*7.5 英寸的 veiwport 范围，使用默认的 viewport 的结果是产生一个近似为 8*5 英寸的图形区。你可强制 viewport 的 X 方向为固定长度或将 viewport 的 X 方向与整个坐标范围成比例。尺寸信息会被写入 SGF 文件中，然后由其他转换程序负责将 SGF 转换成合适大小的图片。

示例

设置 SGF 文件的保存目录为非当前目录，并重置 frame 编号：

```
OBSPY> sgf dir /mydir/sgfstore frame 0
```

设置 viewport 的 X 方向长度为 3 英寸：

```
OBSPY> sgf size fixed 3.0
```

创建一个大小相当于海报的图形：

```
OBSPY> sgf size fixed 30.0
```

设置 vierport 的 X 方向的尺寸与地震数据的时间长度成比例：

```
OBSPY> sgf size scaled 0.1 // 10s 的数据长度为 1 英寸
```

本例中，持续 60 秒的数据图形将有 6 英寸长，而持续 600 秒的数据将有 60 英寸长，过长的图形需要特殊的后期处理。

12.139　smooth

概要

对数据应用算术平滑算法

语法

```
SMOOTH [MEAN|MEDIAN] [HALFWIDTH n]
```

输入

MEAN|MEDIAN 应用均值/中值平滑算法
HALFWIDTH n　设置滑动窗的半宽为 n 个数据点

缺省值

```
smooth mean halfwidth 1
```

说明

此命令对数据应用算术平滑算法，可以选择均值平滑或中值平滑算法。 滑动窗的窗长通过指定其半宽度 n 来定义，对于第 n 到第 npts-n+1 个的每个数据点而言，应用一个窗长为$2n+1$个数据点平滑窗。对于两端的$2(n-1)$个数据点而言，做特殊处理。

头段变量

depmin、depmax、depmen

12.140 sonogram

概要

计算一个频谱图，其等价于同一个谱图的两个不同的平滑版本的差

语法

```
SONOGRAM [WINDOW v] [SLICE v] [ORDER n] [CBAR ON|OFF]
    [YMIN v] [YMAX v] [FMIN v] [FMAX v] [BINARY|FULL]
    [METHOD PDS|MEM|MLM] [COLOR|GRAY] [PRINT pname]
```

输入

WINDOW v 设置滑动数据窗的长度为 v 秒，窗长决定了 FFT 的尺寸
SLICE v 设置数据滑动间隔为 v 秒，对每个滑动间隔将产生一个频谱图线
ORDER n 指定用于计算谱估计的自相关中的点数
CBAR ON|OFF 打开/关闭参考颜色条
BINARY|FULL 产生一个双色或彩色图像
YMIN v 指定绘图的最小频率
YMAX v 指定绘图最大频率
FMIN v 指定每次滑动频谱图被平滑的最小带宽
FMAX v 指定每次滑动频谱图被平滑的最大带宽
METHOD PDS|MEM|MLM 指定使用的谱估计方法，PDS 代表功率密度谱估计，MLM 代表最大似然，MEM 代表最大熵谱估计。
COLOR|GRAY 指定是彩色图还是灰度图

缺省值

```
sonogram window 2 slice 1 method mem order 100 ymin 0 ymax
    fnyquist fmin 2.0 fmax 6.0 full color
```

说明

sonogram 命令计算了一个谱图，其等效于同一谱图两种不同平滑版本的差。依赖于平滑参数 fmin 和 fmax 的选择，结果谱图可以加强小幅度谱特性，而其在传统谱图中很难观测到。这个在矿井爆炸的地震信号中寻找高频谱时很有用。

限制

图形在频率方向的尺寸为 512

问题

目前在头段检查以确定他们有相同的分量且在时间上连续方面还有些错误。

头段变量

需要：delta 修改：npts、delta、b、e、iftype、depmin、depmax、
depmen
创建：nxsize、xminimum、xmaximum、nysize、yminimum、ymaximum

12.141 sort

概要

根据头段变量的值对内存中的文件进行排序

语法

```
SORT header [ASCEND|DESCEND] [header [ASCEND|DESCEND]...]
```

输入

header 依据该头段变量的值进行文件排序
ASCEND 升序排列
DESCEND 降序排列

说明

根据给出的头段值对内存中的文件进行排序。头段变量在命令行中出现的越早，这个变量字段 就具有越高的优先权，后面的变量字段用于解决无法第一个变量字段相同的情况。最多可以输入 5 个头段变量。

每个头段变量都可以跟着 ASCEND 或 DESCEND 来表明特定字段的排序方式。若未指定，则默认为升序排列。如果使用 sort 命令但未指定任何头段值，它将根据上一次执行 sort 命令时的头段去排序，如果第一次调用 sort 但没有给出头段，则会报错。

缺省值

默认所有的字段都是升序排列

12.142 spectrogram

概要

使用内存中的所有数据计算频谱图

语法

```
SPECTROGRAM [WINDOW v] [SLICE v] [ORDER n] [CBAR ON|OFF]
    [SQRT|NLOG|LOG10|NOSCALING]
    [YMIN v] [YMAX v] [METHOD PDS|MEM|MLM] [COLOR|GRAY]
    [PRINT pname]
```

输入

WINDOW v 设置滑动数据窗的长度为 v 秒，决定了 **FFT** 的尺寸 **SLICE v** 设置数据滑动间隔为 v 秒，对每个滑动间隔将产生一个频谱图线
ORDER n 指定用于计算谱估计的自相关中的点数
CBAR ON|OFF 打开/关闭参考颜色条
SQRT|NLOG|LOG10|NOSCALING 指定振幅的自然对数、以 10 为底的对数或平方根
YMIN v 指定绘图的最小频率
YMAX v 指定绘图最大频率
METHOD PDS|MEM|MLM 指定使用的谱估计方法，PDS 代表功率密度谱估计，MLM 代表最大似然，MEM 代表最大熵谱估计。
COLOR|GRAY 指定是彩色图还是灰度图

缺省值

```
spectrogram window 2 slice 1 method mem order 100 noscaling
    ymin 0 ymax fnyquist color
```

说明

频谱图是通过计算连续并可能重叠的数据时间窗的功率谱并将谱沿着时间轴绘制产生的。谱是由一个使用 MLM 或 MEM 或 PDS 得到的被截断的自相关函数。一般会选择高精度的 MLM 和 MEM 方法，因为他们提高了精度且不会产生由于不同频率之间能量泄漏导致的人工干扰。这些技术的介绍可以参考 Kanasewich(1981) 以及 Lacoss(1971)。被截断的互相关函数的长度由 ORDER 参数决定。为了保持和 SPE 子程序的一致性，设置了 PDS 的默认 order 为 200，MEM 和 MLM 的 order 默认为 100。在 OBSPY 中每个数据窗的长度由 window 参数决定。沿着频谱图时间轴的谱之间的间隔由 slice 参 数决定。这两个参数的不同决定了临近时间窗的重叠量，如下图所示:

```
Time ->
0  1  2  3  4  5  6  7  8  9 10 11
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|--|
|  ^  | window 1, First time will be at the center of this window.
      |  ^  | window 2
            |  ^  | window 3

|.....| Slice: 临近时间窗的开始时间的差
```

频谱图时间轴的起始和结束点依赖于要被分析的时间序列的长度以及 window、slice 参数。频谱图的起始时间是时间序列起始时间迟半个窗对应的时间，因为它被定义为第一个窗的中间时刻。OBSPY 不会对数据的前面补 0。

限制

图形在频率方向的尺寸为 512

问题

目前在头段检查以确定他们有相同的分量且在时间上连续方面还有些错误。

头段变量改变

需要：delta 改变：npts、delta、b、e、iftype、depmin、depmax、
depmen

创建：nxsize、xminimum、xmaximum、nysize、yminimum、ymaximum

12.143　sqr

概要

对每个数据点做平方

语法

```
SQR
```

头段变量

depmin、depmax、depmen

12.144　sqrt

概要

对每个数据点取其平方根

语法

```
SQRT
```

头段变量

depmin、depmax、depmen

12.145　stretch

概要

拉伸（增采样）数据，包含了一个可选的 FIR 滤波器

语法

```
STRETCH n [FILTER ON|OFF]
```

输入

n 设置增采样因子，取值为 2 到 7
FILTER ON|OFF 打开/关闭插值 FIR 滤波器选项

缺省值

```
stretch 2 filter on
```

说明

此命令对数据进行拉伸，即增采样。当关闭滤波器选项时，仅仅在原始数据点之间插入适当数 目的零值。若使用了 FIR 滤波器，则通过对数据进行滤波可以创建一个与原始波形相似但是采样周期更小的文件。需要注意的是，滤波器对频谱成分是有影响的。

头段变量

npts、delta、e、depmin、depmax、depmen

12.146 sub

概要

数据文件的每个数据点减去同一个常数

语法

```
SUB [v1 [v2 ... vn]]
```

输入

v1 从第一个文件中减去的常数
v2 从第二个文件中减去的常数
vn 从第 n 个文件中减去的常数

缺省值

```
sub 0.0
```

说明

参见 add 的相关说明。

头段变量改变

depmin、depmax、depmen

12.147 subf

概要

使内存中的一组数据减去另一组数据

语法

```
SUBF [NEWHDR [ON|OFF]] filelist
```

输入

NEWHDR ON|OFF 指定新生成的文件使用哪个文件的头段。OFF 表示使用内存中原文件的头段区，ON 表示使用 filelist 中文件的头段区。缺省值为 OFF
filelist OBSPY 二进制文件列表

说明

参见 addf 命令的相关说明。

头段变量

depmin、depmax、depmen、npts、delta

12.148　symbol

概要

控制符号绘图属性

语法

```
SYMBOL [ON|OFF|n] [SIZE v] [SPACING v] [INCREMENT ON|OFF]
    [LIST STANDARD|nlist]
```

输入

ON 打开符号绘图开关，不改变符号号

OFF 关闭符号绘图开关

n 打开符号绘图开关。将符号号设置为 n。目前有 16 个不同的符号，符号 0 意味着关闭符号开关

SIZE v 设置符号尺寸为 v。值为 0.01 意味着占据整个绘图尺寸的 1%

SPACING v 设置符号间隔为 v。这是绘图时符号间的最小间隔；如果你想每个数据点都有符号，则设其值为 0，对注释行使用 0.2 到 0.4

INCREMENT ON 对每个数据文件操作完成后按符号列表递增为下一个符号

INCREMENT OFF 关闭上述 INCREMENT 选项

LIST nlist 改变符号表的内容。输入符号号码表。设置表中的第一个符号码，并打开符号绘图开关

LIST STANDARD 改变到标准符号表。设置表中的第一个符号表，并打开符号绘制选项。

缺省值

```
symbol off size 0.01 spacing 0. increment off list standard
```

说明

这些符号属性独立于由 line 命令定义的画线属性。打开画线选项，它们也可以用于注释在相同图形上的不同的线。关闭画线选项，则可以绘制散点图。如果你要将几个数据文件画在同一张图 上，也许需要使用不同的符号。这是可以使用 INCREMENT 选项。当这个选项打开时，每次绘制数据 文件，都从符号表中将原来的符号码增加 1，缺省符号表包含符号表从 2 到 16 的符号，你也可以 使用 LIST 选项改变这个表的次序和内容。如果你在绘制一系列重叠绘图，并需要经相同符号用在 相同次序的每个图形上，这样做很有用。符号码为 0 相当于关闭符号绘制选项。这个选项用于 LIST 选项和 LINE 选项，以在一张图上用线表示一些数据，用符号表示另外一些数据。

示例

为了创建一个散点分布图，关闭画线选项，选择适当的符号，然后绘图：

```
OBSPY>   line
off     OBSPY>
symbol        5
```

为了用符号 7、4、6、8 注释四条实线，间隔用 0.3，用 plot2 绘图：

```
OBSPY> line solid
OBSPY> sym spacing .3 increment list 7 4
6 8 OBSPY>  r  file1  file2  file3  file4
```

```
OBSPY> plot2
```

使用 plot2 在相同图形上绘制三个文件，第一个文件图形使用实线无符号；第二个没有线，为三角符号；第三个没有线，带有交叉符号：

```
OBSPY> read file1 file2 file3
OBSPY> line list 1 0 0
increment
OBSPY> symbol list 0 3 7
```

12.149 synchronize

概要

同步内存中所有文件的参考时刻

语法

```
SYNCHRONIZE [ROUND ON|OFF] [BEGIN ON|OFF]
```

输入

ROUND ON 打开 ROUND 选项。若打开该选项，则会对每个文件的开始时间做微调以使得开始时间是采样间隔的整数倍

ROUND OFF 关闭 ROUND 选项

BEGIN ON 设置每个文件的开始时间为 0

BEGIN OFF 保持参考时间的绝对时刻不变

缺省值

```
synchronize round off begin off
```

说明

该命令用于同步内存中所有文件的参考时刻。通过检查所有文件的参考时间和文件起始时间（B），找到所有文件中最晚的文件起始时刻，并取该时刻作为内存中所有文件共同的参考时刻，最后 再计算每个文件中所有相对时间相对于新参考时刻的值。

当数据文件具有不同的参考时刻时，对数据使用 cut 或 xlim 命令就会有些麻烦，因而可以使用该命令使得所有数据的参考时刻一致。

如果使用了 BEGIN 选项，则会将所有文件的 kzdate 和 kztime 设置为同样的值，并将所有文件的开始时间（B）设置为零，这样做会使得数据丢失绝对时间信息。

示例

假定你读取两个不同参考时间的文件到内存：

```
OBSPY> read file1 file2
OBSPY> listhdr b kztime kzdate

 FILE: FILE1
 -
```

```
  B = 0.
  KZTIME = 10:38:14.000
  KZDATE = MAR 29 (088), 1981

  FILE:  FILE2
  -
  B = 10.00
  KZTIME = 10:40:10.000
  KZDATE = MAR 29 (088), 1981
```

这些文件有相同的参考日期，不同的参考时刻以及不同的开始时间偏移量。可以执行 `synchronize` 同步参考时刻：

```
OBSPY>
synchronize
OBSPY> listhdr

  FILE:  FILE1
  -
  B = -126.00
  KZTIME = 10:40:20.000
  KZDATE = MAR 29 (088), 1981

  FILE:  FILE2
  -
  B = 0.
  KZTIME = 10:40:20.000
```

现在内存中的所有文件有相同的参考时间，如果头段中有任何已定义的时间标记，它们的值也会调整以保证其绝对时刻是不变的。

12.150 systemcommand

概要

在 OBSPY 内部执行系统命令

语法

```
SYSTEMCOMMAND command [options]
```

输入

command 系统命令名
options 命令需要的选项

说明

在 OBSPY 中是可以执行大部分系统命令的，比如常见的 `ls`、`cp` 等。但某些命令无法直接在 OBSPY 中执行，比如用于查看 PS 文件的 `gs` 命令会首先被 OBSPY 解释为 `grayscale` 的简写，故而在 OBSPY 中无法直接调用 `gs` 命令。

另一个经常使用但无法直接调用的命令是 rm。为了避免在读入数据时将命令 r *.OBSPY 误敲成 rm *.OBSPY 而导致数据文件被误删除，故而在 OBSPY 中禁止直接调用 rm 命令。

systemcommand 的作用就是为了能够在 OBSPY 内部间接调用这些无法被 OBSPY 直接调用的系统 命令。

示例

调用系统命令 rm 删除某些 OBSPY 文件：

```
OBSPY> rm              // 无法直接调用 rm 命
 ERROR 1106: Not a valid OBSPY command.
OBSPY> sc rm junks     // 通过 sc 间接调用 rm 命
```

12.151 taper

概要

对数据两端应用对称的 taper 函数，使得数据两端平滑地衰减到零

语法

```
TAPER [TYPE HANNING|HAMMING|COSINE] [WIDTH v]
```

输入

TYPE HANNING | HAMMIN | COSINE 应用 Hanning、Hamming、余弦衰减窗
WIDTH v 设置衰减窗的宽度占数据点数的比值为 v，v 取值在 0.0 和 0.5 之间

缺省值

```
taper type hanning width 0.05
```

说明

taper 函数是在 0 和 1 之间取值的单调函数，若将其对称地施加于数据的首尾两端，则可实现数据的“尖灭”。

taper 函数共计 $npts * v$ 个点，第一个点值为 0，最后一个点的值为 1，将此函数的每个点依次于数据的第 1 至 $npts * v$ 个点相乘，使得数据数据的首端从 0 开始光滑地增加到其原始值。数据的末端完全类似，此时数据由其原始值不断光滑地减小到 0。

taper 命令的通用形式为

$$Data(j) = Data(j) * (F_0 - F_1 \cos(\omega(j-1)))$$

此公式应用于数据的首端，另一个完全对称的数据用于数据的尾端。

表 12.2 定义了不同的衰减函数的参数，其中 N 为衰减窗的宽度，即 $npts * v$。

图 12.4 给出了不同 taper 衰减函数的曲线图，图中可以看出，hamming 窗实际上并没有完全实现尖灭。

头段变量

depmin、depmax、depmen

表 12.2: taper 衰减函数参数一览

类型	ω	F_0	F_1
HANNING	$\frac{\pi}{N}$	0.50	0.50
HAMMING	$\frac{\pi}{N}$	0.54	0.46
COSINE	$\frac{\pi}{2N}$	1.00	1.00

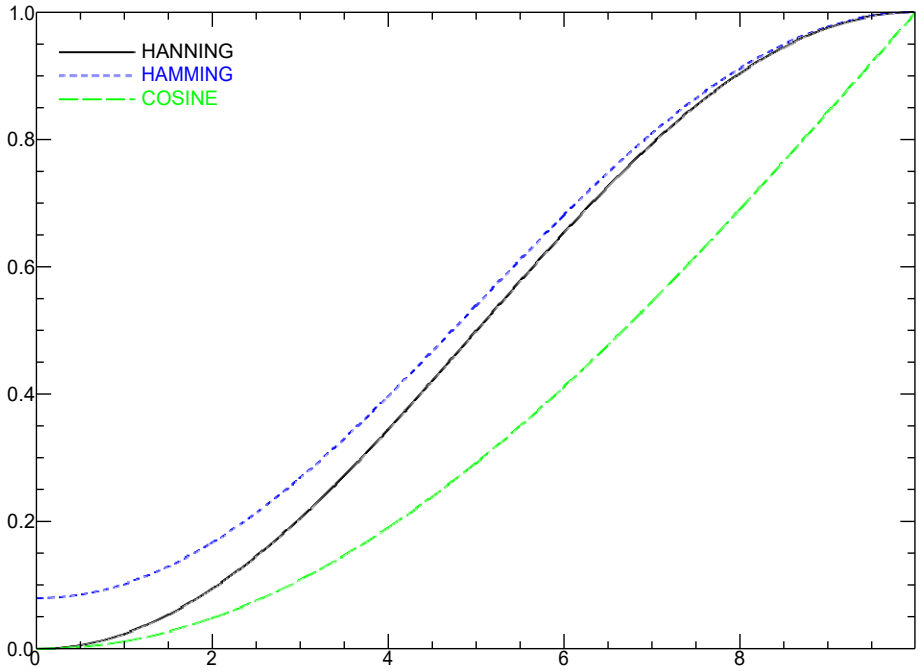

图 12.4: taper 衰减函数曲线

12.152 `ticks`

概要

控制绘图上刻度轴的位置

语法

```
TICKS [ON|OFF|ONLY] [ALL] [TOP] [BOTTOM] [RIGHT] [LEFT]
```

输入

ON 在指定的边上显示刻度，其他不变
OFF 在指定的边上不显示刻度，其他不变
ONLY 仅在指定的边上显示刻度，其他关闭
ALL 所有四条边
TOP X 轴的上边
BOTTOM X 轴的下边
RIGHT Y 轴的右边
LEFT Y 轴的左边

缺省值

```
ticks on all
```

说明

刻度轴可以画图形四边的一边或几边上，刻度间隔由 xdiv 命令控制。

示例

显示上部刻度轴，其他不变：

```
OBSPY> ticks on top
```

关闭所有刻度轴：

```
OBSPY> ticks off all
```

只显示底部刻度轴：

```
OBSPY> ticks only bottom
```

12.153 title

概要

定义绘图的标题和属性

语法

```
TITLE [ON|OFF|text] [LOCATION TOP|BOTTOM|RIGHT|LEFT]
    [SIZE TINY|SMALL|MEDIUM|LARGE]
```

输入

text 显示标题，并设置标题内容，若 text 包空格，则需要用引号括起来
ON 显示标题，但不改变标题内容
OFF 不显示标题
LOCATION 设置标题的位置，可以取 TOP、BOTTOM、RIGHT、LEFT
SIZE 设置标题的尺寸，可以取 TINY、SMALL、MEDIUM、LARGE

缺省值

```
title off location top size small
```

说明

若打开该选项，则在每个图形上都显示标题，标题的尺寸、位置及内容均可改变，文本质量和字 体可以通过 gtext 命令设置。

12.154 trace

概要

追踪黑板变量和头段变量

语法

```
TRACE [ON|OFF] name [name ...]
```

输入

ON 打开变量追踪选项

OFF 关闭变量追踪选项

name 要追踪的黑板变量或/和头段变量名。对于头段变量，其格式为 `filename,hdrname`，其中 `filename` 是要追踪的 OBSPY 文件名或文件号，`hdrname` 是 OBSPY 头段变量名

缺省值

```
trace on
```

说明

该命令用于在 OBSPY 执行过程中追踪 OBSPY 黑板变量或/和头段变量的值，主要用于调试大型或 复杂的宏文件。当变量追踪选项被打开时，将显示变量的当前值。若变量追踪选项处于打开状态，则 每次执行命令时将对变量值进行检查，若变量的值发生改变则将其新值打印到终端。当变量追踪选项被关闭时，也会显示变量的当前值。

示例

追踪黑板变量 TEMP1 和文件 MYFILE 的头段变量 `T0`：

```
OBSPY> trace on temp1
  myfile,t0 TRACE (on) TEMP1 =
  1.45623
```

在执行命令时，OBSPY 会检查变量值是否发生改变。若发生改变则将相关信息显示出来。假设在 完成一些计算之后改变了 TEMP1，并定义了 `T0` 的值，则 OBSPY 将显示如下信息：

```
  TRACE (mod) TEMP1 = 2.34293
  TRACE (mod) MYFILE,T0 = 10.3451
```

稍后的处理中 TEMP1 可能再次改变：

```
  TRACE (mod) TEMP1 = 1.93242
```

当跟踪选项被关闭时，OBSPY 最后一次显示变量当前值：

```
OBSPY> trace off temp1 myfile,t0
  TRACE (off) TEMP1 = 1.93242
  TRACE (off) MYFILE,T0 = 10.3451
```

12.155 transcript

概要

控制输出到副本文件

语法

```
TRANSCRIPT [OPEN|CREATE|CLOSE|CHANGE|WRITE|HISTORY] [FILE filename]
    [CONTENTS    ALL|ERRORS|WARNINGS|OUTPUT|COMMANDS|MACROS|PROCESSED]
    [MESSAGE text]
```

输入

OPEN 打开副本文件，并在已存在的文件底部添加副本

CREATE 创建一个新的副本文件

CLOSE 关闭一个已经打开的副本文件

CHANGE 改变一个已经打开的副本文件的内容

WRITE 写信息到一个副本文件，不改变其状态或内容

HISTORY FILE filename 将命令行历史保存到文件

FILE filename 定义副本文件的名字

MESSAGE text 向副本文件中写入文本。这个信息可以用于指定正在进行的进程或指定正在处理的不同事件，在两次执行这个命令的过程中这个信息不保存

CONTENTS ALL 定义副本文件的内容为全部输入输出的信息

CONTENTS list 定义副本文件的内容，即包含在文件中的输入输出的类型表

其中 `list` 可以取：

- `ERRORS`：执行命令期间产生的错误消息
- `WARNINGS`：执行命令期间产生的警告消息
- `OUTPUT`：执行命令期间的输出消息
- `COMMANDS`：终端输出的原始命令
- `MACROS`：宏文件中出现的原始命令
- `PROCESSED`：经终端或宏处理之后的命令

缺省值

```
transcript open file transcript contents all
```

说明

副本文件用于记录 OBSPY 执行的结果。其可以是一个完整或部分副本，可以包含一次或多次执 行的结果。你可以同时拥有 5 个活动的副本文件，每个文件用于追踪不同的方面。其中的一个用途 是记录终端输入的命令然后用于一个宏文件中，如下例所示。

示例

为了创建一个新的副本文件，文件名为 mytran，包含了除已处理命令之外的其他全部类型：

```
OBSPY> transcript create file mytran cont err warn out com macros
```

如果之后不想把宏命令送入这个文件，你可以使用 `CHANGE` 选项：

```
OBSPY> transcript change file mytran contents e w o c
```

为了定义一个名为myrecord的副本文件，其记录了终端输入的命令：

```
OBSPY> transcript create file myrecord contents commands
```

以后，经过适当的编辑，这个文件可以用作宏命令文件，去自动执行相同的一组命令。在最后的例子中假设你需要彻夜处理许多事件。你可以设置每个事件一个副本文件（用不同的副本文件名） 去记录处理的结果。另外你可以将处理所有事件得到的任何错误消息保存到一个副本文件中：

```
OBSPY> transcript open file errortran contents
errors OBSPY> transcript write message 'processing
```

这些命令将放在处理每个事件的宏文件中，它假设事件名作为第一个参数带入宏。使用打开选 项，运行信息和出错信息将添加到文件的后面，第二天检查一下这个出错信息副本文件，就可以快 速查阅在处理期间是否出现了错误。

为了将保存一个命令行副本，以记录 OBSPY 当前和将来要运行的命令：

```
SCA> transcript history file .obspyhist
```

这就在当前目录创建并写入了一个副本文件 “~/.Obspyhist”。任何储存在那里的文件将被载入命令历史中。如果这个命令位于你的启动初始化宏文件中，则每次你运行 OBSPY 时将在当前目录产生一 个单独的命令行历史。在一个新执行的 OBSPY 中，上下键将浏览完整的命令历史，你可以修改以前输 入的命令并再次执行它，如果你没有在 OBSPY 内或初始化宏文件中输入这个命令，则命令行历史将 自动保存到 ~/.Obspy_history。

12.156 transfer

概要

反卷积以去除仪器响应，如果需要，还可以卷积其他仪器响应

语法

```
TRANSFER [FROM type [options]] [TO type [options]]
    [FREQLIMITS f1 f2 f3 f4] [PREWHITENING ON|OFF|n]
```

输入

FROM type 要去除的仪器响应类型
TO type 要加入的仪器响应类型
FREQLIMITS f1 f2 f3 f4 压制大于 f4 以及低于 f1 的频段
PREWHITENING ON|OFF|n 预白化处理

缺省值

```
trans from none to none
```

说明

FROM and TO

transfer 命令的作用是将波形数据从一种仪器响应转换成另一种仪器响应。FROM type 指定要从波形数据中去除的仪器响应，TO type 指定了要加入到波形数据中的仪器响应。去仪器响应即反卷积，加仪器响应即卷积，二者分别通过谱域的除法和乘法完成。

type 为仪器类型，可以是 OBSPY 预定义的标准仪器类型（见附录中表 B.1），还可以是如下几种 特殊的 “仪器类型”：

none 表示“位移”
vel 表示“速度”
acc 表示“加速度”
evalresp 表示使用 RESP 仪器响应文件
polezero 表示使用 OBSPY PZ 仪器响应文件
fap 表示使用 fap 仪器响应文件

tranfser 命令的默认值是“transfer from none to none”，即默认的输入和输出“仪器类型”都是位移。因而当不指定 FROM type 或 TO type 时，OBSPY 会假定仪器类型为 NONE。

- 若输出的仪器类型为 NONE，即表示从波形中去除仪器响应得到位移，此时 OBSPY 头段中的 IDEP 设置为 IDISP，单位为 nm，若输出类型为 VEL 或 ACC，同理；
- 若输出的仪器类型不是 NONE、VEL 或 ACC，则内存中的波形会卷积上 TO type 所指定的仪器响应；
- 若不指定 FROM 选项，则假定原始波形数据是位移，且不会去除仪器响应；通常用于给理论地震图添加仪器响应；

freqlimits

freqlimits 用于在去除仪器响应时对波形的低频和高频部分进行压制。当 TO type 为 NONE、VEL 或 ACC 时，必须使用该选项，且必须认真选择合适的参数。

所有地震仪在零频率处都具有零响应，在只进行反卷积时，需要在频率域做谱的除法，此时可 能会除以一个很小的值进而导致低频处有很大的谱振幅；在高频处，信噪比通常很低，因而有必要 对其响应进行压制。

freqlimits 会在去仪器响应时对频谱做低通和高通尖灭，以实现对高频和低频的压制。四个频率参数应满足 f1<f2<f3<f4，即尖灭函数在零到 f1 之间为 0，f1 到 f2 之间从 0 渐渐变成 1，在 f2 和 f3 之间保持为 1，在 f3 到 f4 之间从 1 渐渐变成 0，大于 f4 的频段值为 0。过渡带内分别为余弦波的四分之一周期。如下图所示：

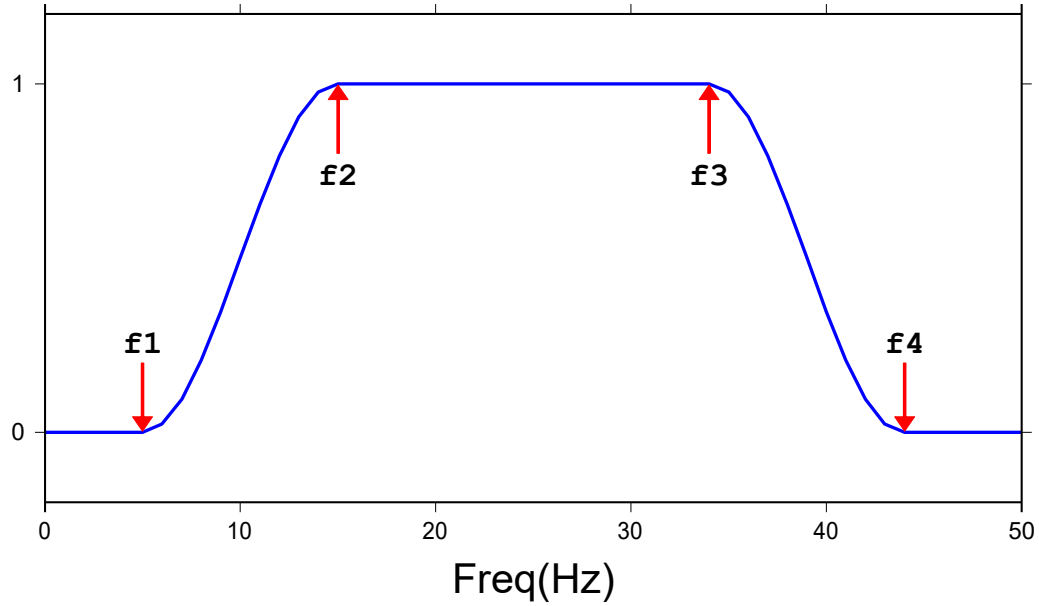

图 12.5: Freqlimits 尖灭函数 四个频率参数除了要满

足 f1<f2<f3<f4 外，还应注意如下几条原则：

- f4 应小于 Nyquist 采样率。比如若数据的采样周期为 0.01 s，则 Nyquist 采样率为 50 Hz，因而 f4 应小于 50 Hz

- f3 不能与 f4 太接近
- f2 与 f3 之间应尽可能宽，然后再根据具体需求进行滤波
- f1 和 f2 不能太接近；
- f1 的选取由具体需求决定，可以尝试不同的值并查看去仪器响应之后的效果来决定 若想要一个低通滤波器但在低频处不滤波，可以设置 f1=-2 和 f2=-1；若想要一个高通滤波器但在高频处不滤波，可以设置 f3 等于 Nyquist 频率，f4 为 Nyquist 频率的两倍。

需要注意，该滤波器是零相位、非因果滤波器，因而，若数据点数不为 2 的指数幂次，会导致在频段 (f1,f4) 之外振幅不完全为 0。若想要数据点数为 2 的幂次方，可以参考 OBSPY 中的 cut 命令。

prewhitening

prewhitening 用于控制数据的预白化。预白化可以将输入时间序列在变换到频率域之前，进行谱的平化。这会减小谱值的动态范围，并提高数据在高频的计算精度。参见 whiten 命令。打开预白化选项，会在谱操作之前在频率域进行谱白化，并在谱操作后在时间域做谱白化的补偿，也可以 设置预白化选项的阶数。默认情况下，预白化选项是关闭的，阶数为 n=6。

示例

内置仪器类型

OBSPY 中内置了一堆预定义的仪器类型，可以在命令中直接使用。

从数据中去除 LLL 宽频带仪器响应。并卷积上 SRO 仪器响应，且对频带做尖灭及预白化：

```
OBSPY> read abc.z
OBSPY> rmean; rtr; taper
OBSPY> trans from III to sro freq .02 .05 1. 2. prew 2
```

当前的仪器类型为 RSTN 的子类型 nykm.z，为了去除该仪器响应并卷积上 DSS 仪器响应：

```
OBSPY> read nykm.z
OBSPY> rmean; rtr; taper
OBSPY> trans from rstn subtype nykm.z to dss prew off
```

将电磁仪器响应转换成位移：

```
OBSPY> r XYZ.Z
OBSPY> trans from elmag freep 15. mag 750. to none
```

从波形中去除 WWSP 的仪器响应，得到位移波形：

```
OBSPY> read xyz.z
OBSPY> rmean; rtr; taper
OBSPY> trans from WWSP to none freq 0.05 0.01 5 10
                  // 也可使用 to vel 或 to acc 得到速度或加速度
```

向合成的位移地震图中加入 WWSP 仪器响应：

```
OBSPY> r syn.z
OBSPY> trans from none to WWSP // 简写为 trans to WWSP
```

evalresp 类型

evalresp 类型并不代表真正意义上的仪器类型，而是表示从 RESP 仪器响应文件中读取仪器响应信息。在使用 evalresp 选项时，transfer 依次从当前内存中的 OBSPY 波形数据中提取出各自 的头段信息，包括：kstnm、kcmpnm、kzdate、kztime、knetwk 和 locid，然后会在当前目录下寻找 文件名为 “RESP.<NET>.<STA>.<LOCID>.<CHN>” 的 RESP 文件（比如 “RESP.IU.COLA..BHZ”），
并检测 RESP 文件中给出的台站信息是否与数据中的台站信息匹配 [1]。

```
OBSPY> r 2006.253.14.30.24.0000.TA.N11A..LHZ.Q.OBSPY
OBSPY> rtr; rtr; taper
OBSPY> trans from evalresp to none freq 0.004 0.007 0.2 0.4
```

该命令会首先从头段中提取台站信息，然后自动在当前目录下寻找文件 RESP.TA.N11A..LHZ，一旦文件中的台站信息与数据中的台站信息匹配，则使用该响应函数。

OBSPY 数据中的头段信息可以用一些选项来覆盖：

```
STATION, CHANNEL, NETWORK, DATE, TIME, LOCID, FNAME
```

每个选项都必须有一个合适的值。若 DATE 在 OBSPY 头段中未设定且在选项中未指定，则使用当前系 统日期，TIME 同理；若 NETWORK 未指定，则默认使用任意台网名；若 LOCID 或 KHOLE 未指定，则默 认使用任意 LOCID。

假设台网 IU 的所有台站都具有完全相同的仪器响应函数，而此时你只有 COLA 台站的 RESP 文件 RESP.IU.COLA..BHZ。为了给所有台站去除仪器响应，一种办法是对 IU 台网的每一个台站复制一份 RESP.IU.COLA..BHZ，重命名，并修改 RESP 文件中的台站信息。显然，这样很麻烦，利用上面的选项可以大大简化这一过程：

```
OBSPY> r *.IU.*.BHZ
OBSPY> rmean; rtr; taper
OBSPY> trans from evalresp STATION COLA to none freq 0.01 0.02 5 10
```

使用 STATION 选项覆盖了波形数据中的台站名，此时，对每一个波形数据，transfer 命令都会去使用 RESP.IU.COLA..BHZ[1]。

下面的命令会将三分量数据去仪器响应，并卷积上 BHZ 分量的仪器响应：

```
OBSPY> r *.IU.COLA.00.BH?
OBSPY> rmean; rtr; taper
OBSPY> trans from evalresp to evalresp CHANNEL BHZ
```

操作完成后，BHZ 分量相当于没有进行操作，BH1 和 BH2 则去除了原本的仪器响应并卷积上 BHZ 的仪器响应。

为了显示 IU 台网 COL 台站 BHZ 通道，1992 年 01 月 02 日 16:42:05 的仪器响应：

```
OBSPY> fg impulse npts 16384 delta .05 begin 0.
OBSPY> trans to evalresp sta COL cha BHZ net IU \
                    date 1992/2 time 16:42:05
OBSPY> fft
OBSPY> psp
```

[1] 即，要求 RESP 文件名以及 RESP 文件中的台站信息都与数据头段中的台站信息匹配

[1] 这里假定所有台站的 LOCID 都是未定义的

如果你的 RESP 文件名与 OBSPY 的标准格式不同，可以使用 FNAME 选项强制指定要使用的 RESP 文件：

```
OBSPY> r 2006.253.14.30.24.0000.TA.N11A..LHZ.Q.OBSPY
OBSPY> rmean; rtr; taper
OBSPY> trans from evalresp fname /tmp/Resp/RESP.TA.N11A..LHZ to none \
                        freq 0.004 0.007 0.2 0.4
```

transfer 命令默认会使用 RESP.TA.N11A..LHZ 作为响应文件，此处使用 FNAME 选项强制指定使用 /tmp/RESP/RESP.TA.N11A..LHZ。需要注意的是，即便是使用 FNAME 强制指定了 RESP 文件，该命令还是会检测台站信息是否匹配。

由于一个 RESP 文件中可以包含多个响应函数，因而可以将所有仪器响应文件合并到一个总的 RESP 文件中：

```
OBSPY> r *.OBSPY
OBSPY> rmean; rtr; taper
OBSPY> transfer from evalresp fname RESP.ALL to none freq 0.1 0.2 5 10
```

这个例子中，RESP.ALL 包含了所有数据的响应函数，transfer 命令会读取 RESP.ALL 文件的内容，对于每一个波形数据，会从波形数据中提取出台站信息，并与 RESP.ALL 中的众多响应函数进行匹配，若匹配成功，则使用该响应函数。

polezero 类型

polezero 类型并不代表真正意义上的仪器类型，而是表示从 OBSPY 零极点文件中读取仪器响应函数。

polezero 类型会从数据波形中提取台站信息，但不会根据台站信息去寻找默认的 PZ 文件，用户必须使用 subtype 来指定要使用的 PZ 文件。若 PZ 文件有注释行，则注释行中的台站信息必须与波形中的台站信息匹配，才能正确执行；若 PZ 文件中无注释行，则不进行台站信息匹配的检测，直接执行。

```
OBSPY> r *IU.COLA.BHZ
OBSPY> rmean; rtr; taper
OBSPY> trans from polezero subtype OBSPY_PZs.IU.COLA.BHZ to WWSP
```

一个 PZ 文件中可以包含多台站、多通道、多时间段的响应函数。可以将所有数据的 PZ 文件合并得到总的 PZ 文件。下面的例子中读入全部波形数据，并利用总 PZ 文件进行去仪器响应：

```
OBSPY> r              // 读入全部数据
OBSPY> rmean; rtr; taper
OBSPY> trans from polezero s event.pz to none freq 0.05 0.1 10.0
15.0 OBSPY> mul 1.0e9    // 需要乘以 1.0e9 !!!!!
OBSPY> w over
```

需要格外注意，在用 PZ 文件去仪器响应得到位移物理量时，得到的数据的单位是 m，而 OBSPY 中默认的单位是 nm，因而需要将数据乘以 1.0e9 将数据的单位转换成 nm。对于转换得到速度或 加速度，同理。

fap 选项

fap 选项表明使用 FAP 文件作为响应函数。

假设有 fapfile 文件 fap.n11a.lhz_0.006-0.2，其名字表示频率段为 0.006 Hz 到 0.2 Hz，要从波形 2006.253.14.30.24.0000.TA.N11A..LHZ.Q.OBSPY 中移除该仪器响应：

```
OBSPY> r 2006.253.14.30.24.0000.TA.N11A..LHZ.Q.OBSPY
OBSPY>
rtr
OBSPY>
taper
OBSPY> trans from fap s fap.n11a.lhz_0.006-0.2 to none \
```

12.157 traveltime

概要

根据预定义的速度模型计算指定震相的走时

语法

```
TRAVELTIME [MODEL model] [PICKS n] [PHASE phaselist]
    [VERBOSE|QUIET] [M|KM]
```

输入

MODEL 预定义的速度模型，可以取 iasp91 或 ak135，缺省值为 为 iasp91

PICKS number 的取值为 0 到 9，表明将第一个震相的到时存储到对应的头段变量 Tn 中，其余震相依次写入到后面的头段变量 Tn 中；若未指定 PICKS 选项，则只计算震相到时但不写入头段中

PHASE 要计算/标记的震相列表，若使用了 PICKS n 选项，则震相到时信息将被写入头段 Tn 和 KTn 中

VERBOSE|QUIET 若使用 VERBOSE，则会在终端输出震相走时及其相对于文件参考时刻的秒数；若使用 QUIET，则不在屏幕上显示震相走时信息

M|KM 头段变量 evdp 的单位为 m 或者 km

缺省值

```
traveltime MODEL iasp91 KM PHASE P S Pn Pg Sn Sg
```

说明

该命令使用 iaspei-tau 程序计算标准速度模型下的震相理论走时，要求内存中的 OBSPY 数据文 件中必须包含事件位置、台站位置以及发震时刻。震相名区分大小写，可使用的震相名参考 iaspei- tau 的相关文档。

若使用了 PICKS n 选项，则会将震相列表中第一个震相的到时存储在头段变量 Tn 中，余下的震相到时依次存储到其余的 Tn 中。对于每个震相，终端会输出两个时间，前者时震相到时相对于参考时刻的秒数，后者是震相相对于发震时刻的走时。

由于历史原因，事件深度 evdp 以前的单位为 m，而在新版的 OBSPY 中，默认的 evdp 单位为 km，为了能够兼容以前的 evdp 为 m 的波形数据，新版的 OBSPY 中引入了 km 和 m 选项以指定深度 单位。

示例

区域事件，使用默认震相列表：

```
OBSPY> fg
seismo OBSPY>
traveltime
traveltime: depth: 15.000000
traveltime: error finding phase P
traveltime: error finding phase S
traveltime: setting phase Pn at 10.464321 s [ t = 51.894321 s ]
traveltime: setting phase Pg at 22.904724 s [ t = 64.334724 s ]
traveltime: setting phase Sn at 50.047722 s [ t = 91.477722 s ]
```

由于震中距比较小，没有 P 和 S 震相，所以会出现 `error finding phase P` 的错误，该错误可忽略。以 Pn 震相为例，震相走时为 51.89s，相对于参考时刻的秒数为 10.46s。

上面的示例，只是计算了震相走时，不会写入到头段变量中。要将震相走时存储到头段变量中， 需要使用 `PICKS n` 选项：

```
OBSPY> fg seismo
OBSPY> traveltime picks 0 phase Pn Pg Sn
Sg traveltime: depth: 15.000000
OBSPY> lh T0MARKER T1MARKER T2MARKER
T3MARKER T0MARKER = 10.464      (Pn)
T1MARKER = 22.905       (Pg)
T2MARKER = 50.048       (Sn)
T3MARKER = 66.414       (Sg)
OBSPY> write seismo-picks.z
```

可以看到，`T0` 到 `T3` 分别保存了震相 Pn、Pg、Sn、Sg 震相的到时相对于文件参考时刻的秒数。`KT0` 到 `KT3` 中则分别存储了相应的震相名。此处，尽管没有使用 `VERBOSE` 选项，深度信息还是会被打印出来，这是为了提醒用户注意 `evdp` 的单位，可以通过 `QUIET` 选项设置不显示深度信息。

`rdseed v5.0` 生成的波形数据，震源深度 `evdp` 的单位为 m：

```
OBSPY>  r  2008.052.14.16.03.0000.XC.OR075.00.LHZ.M.OBSPY
OBSPY> lh evdp
evdp = 6.700000e+03
OBSPY> traveltime M picks 0
traveltime: depth: 6.700000 km
OBSPY> lh t0marker t1marker t2marker
t3marker t0marker = 61.48       (Pn)
t1marker = 76.413       (Pg)
t2marker = 109.66       (Sn)
t3marker = 132.11       (Sg)
OBSPY> ch evdp (0.001 * &1,evdp&) // 将 evdp 的单位改成
km OBSPY> setbb station &1,KSTNM&
OBSPY> write %station%.z
```

12.158 tsize

概要

控制文本尺寸属性

语法

```
TSIZE [TINY|SMALL|MEDIUM|LARGE v ] [RATIO v] [OLD|NEW]
```

输入

TINY|SMALL|MEDIUM|LARGE v 设定标准文本尺寸的值为 v
RATIO v 设定文本的宽高比为 v
OLD 将所有文本尺寸值设置为旧值。旧值即 OBSPY 9 之前的版本中的文本尺寸值
NEW 设定所有文本尺寸值为OBSPY初始化时的缺省值

缺省值

```
tsize ratio 1.0 new
```

说明

大多数的文本注释命令（title、xlabel、fileid 等）允许你改变要显示的文本的尺寸。

OBSPY 提供了四个标准尺寸：TINY、SMALL、MEDIUM 和 LARGE。每一个标准尺寸都有一个初始值，如下表所示：

表 12.3: OBSPY 标准文本尺寸

NAME	A	B	C	D	E
TINY	0.015	66	50	68	110
SMALL	0.020	50	37	66	82
MEDIUM	0.030	33	25	44	55
LARGE	0.040	25	18	33	41

上面每列的定义如下：

- A 字符高度相对整个视窗的高度的比值
- B 全视窗下文本的行数
- C 正常视窗下文本的行数。正常视窗是指 x 为 0.0 到 1.0，y 为 0.0 到 0.75
- D 正常视窗中，每行的最小字符数
- E 正常视窗中每行字符的平均数

tsize 命令允许你重新定义这四个标准尺寸的大小以及宽高比。

从 OBSPY 9 开始，系统的默认文本尺寸发生了变化，新的尺寸集覆盖了更宽的尺寸范围，在多数 设备上看上去更好。你可以使用 OLD 选项将文本尺寸修改为之前版本的文本尺寸。

示例

为了改变 MEDIUM 的定义，并使用它创建一个特别尺寸的标题：

```
OBSPY> tsize medium 0.35
OBSPY> title 'rayleigh wave spectra' size medium
```

```
OBSPY> plot2
```

为了重置文本尺寸到其默认值：

```
OBSPY> tsize new
```

12.159　unsetbb

概要

删除黑板变量

语法

```
UNSETBB ALL|variable ...
```

输入

ALL　删除已定义的全部黑板变量
variable 删除黑板变量 `variable`

示例

一次删除多个黑板变量：

```
OBSPY> unsetbb c1 c2 x
```

删除所有黑板变量：

```
OBSPY> unsetbb all
```

12.160　unwrap

概要

计算振幅和展开相位

语法

```
UNWRAP [FILL ON|OFF|n] [INTTHR v] [PVTHR v]
```

输入

FILL ON|OFF　打开/关闭补零选项
FILL n 打开补零选项并设置填充值为 `n`
INTTHR v　改变积分阈值常量为 `v`
PVTHR v　改变主值阈值常量为 `v`

缺省值

```
unwrap fill off intthr 1.5 pvthr 0.5
```

说明

该命令将内存中的时间序列数据转换为谱数据，包括振幅谱和展开后的相位谱，该命令仅对有 相位光滑变化的数据起作用。在数据转换之前先对数据补零使得数据点数为 2^n，也可以使用 FILL 选项指定补其他的值。

用 fft 计算出的相位谱，相位限制在 $-\pi$ 与 π 之间，因而相位谱看上去比较杂乱。unwrap 将相位谱做展开，使得相位谱更加连续变化。下图中分别展示了原始相位谱（上）和展开后的相位谱（下）。

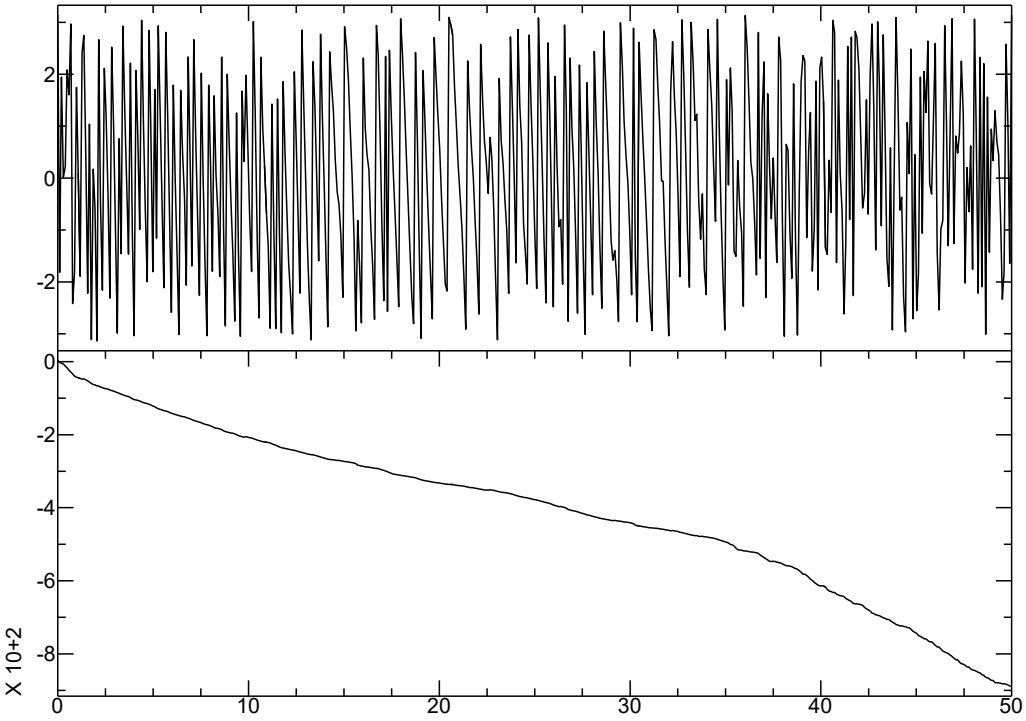

图 12.6: 展开相位 相位展开算法中使用了两种方法来估计

每个频率处的展开相位。 一种是通过快速傅氏变换做相位偏导的数值积分。若要得到一个一致的估计，则可将梯形积

分的步长在每个频率上对分。可以使用 INTTHR 选项控制这个验算的阈值，此值单位为弧度。减小 INTTHR 将改进相位计算结果，若该值太小，会导致解的发散。 算法中使用的第二个方法是先用反正切函数计算相位的主值。展开相位的计算方法是相位主值

加上 2π 的整数倍，直到相位的突变小于给定的阀值为止。可以使用 PVTHR 选项控制这个验算的阀值。与上一个算法类似，减少这个阀值将改进相位估算的结果，但也增加了无解的可能性。 这两个阀值的初值通常经验地取为：

$$\pi/4 < PVTHR < INTTHR < 2\pi$$

头段变量

b、e 和 delta 分别改变为变换的起始频率、结束频率和采样频率。原始的 b、e 和 delta 被保存在为 sb、se、sdelta，当进行反变换时将值带回。

限制

目前可以转换的数据最大长度为 4096。

12.161　vspace

概要

设置图形的最大尺寸和长宽比

语法

```
VSPACE [FULL|v]
```

输入

FULL　使用整个 viewspace，这是最大窗口尺寸
v　设置 viewspace 的纵横比为 v，具有这个纵横比的最大区域即为 viewspace

缺省值

```
vspace full
```

说明

viewspace 是屏幕上可以用于绘图的部分。viewspace 的形状和尺寸在不同图形设备之间有很大的变化。

1. 尽管在尺寸上有很大不同，许多图形终端都具有 0.75 的纵横比
2. SGF 文件的纵横比为 0.75，其大约是标准的 8.5*11 英寸纸张的纵横比
3. 由 XWindows 或 SunWindows 图形设备建立的窗口可以有你想要的任意纵横比　默认情况下绘图会占据整个 viewspace。该命令可以控制 viewspace 的纵横比，从而使你能够控制图形的形状。如果确定了一个纵横比，则　viewspace　就是设备上具有这个纵横比的最大区域。

当你使用 `plotc` 命令在交互设备上建立一张图，并且最终要将它发送到 SGF 设备上，这个命令特别有用。在绘制任何图形之前，必须设置纵横比为 0.75。这将保证图形在 SGF 文件上与在交互设备上相同。如果你要建立一个独立于图形设备的正方形　viewspace，则可以简单地设置纵横比为 1.0。

12.162　wait

概要

控制 OBSPY 在输出大量文本或绘制多个图形时是否暂停

语法

```
WAIT [TEXT ON|OFF] [PLOTS ON|OFF|EVERY]
```

输入

TEXT ON|OFF　打开/关闭文本等待选项
PLOTS ON|OFF　打开/关闭绘图等待选项
PLOTS EVERY　每个图形之间均等待

缺省值

```
wait text on plots on
```

说明

TEXT 选项用于控制当OBSPY输出到终端的内容超过一屏时，是等待用户输入，还是继续输出。

PLOTS 选项用于控制绘制多张图时是否要暂停。在读取多个数据文件并 plot 命令绘图时，每 个文件将产生一个 frame，正常情况下 OBSPY 将在每张图后暂停并发送信息 “Waiting” 到终端，用户键入回车即可看到下一张图，或输入 “Go” 使 OBSPY 不暂停地绘制剩下的图形，或键入 “Kill” 终止绘制这组文件。OBSPY 绘制完最后一张图后不再暂停，因为 OBSPY 的输入提示符已经实现了暂停功能。 当这个选项关闭时，OBSPY 在不同的绘图间不暂停。

12.163 whiten

概要

对输入的时间序列的频谱进行平滑

语法

```
WHITEN n
```

输入

n 阶数（极点数目）。阶数越大，结果数据就越平滑。高阶可以更好地清除一些数据，但是也可能会导致对数据处理过多而丢掉一些重要的数据。默认值为 6。

缺省值

```
whiten 6
```

说明

该命令对数据中加入白噪声，以平滑输入时间序列的频谱。在谱相关命令（比如子程序 SPE 中的命令、transfer、spectrogram）之前执行，可以减少频谱值的动态范围，提高了对地震数据高频操作的精度。

whiten 可以在 SPE 子程序内部调用，也可以在 OBSPY 主程序中调用。SPE 中的 whiten 和主程 序中的 whiten ，阶数是相互独立的，即在主程序中修改 whiten 的阶数，不会影响 SPE 中 whiten 的阶数。与此同时，主程序中的 whiten 命令与 transfer 命令的 prewhiten 选项共用一个阶数； SPE 中 whiten 命令与 SPE 中的 cor 命令的 prewhiten 选项共用一个阶数。

12.164 whpf

概要

将辅助内容写入 HYPO 格式的震相拾取文件中

语法

```
WHPF IC n m
```

输入

IC n m 在第 18 和 19 列插入带有两个整数的 n 和 m 的指令卡。n 的允许值为 0、1、5，m 的允许值为 0、1、9。

说明

“指令卡” 用于分开在 HYPO 文件中的不同事件，参见 HYPO71 手册。关闭一个已经打开的 HYPO 震相拾取文件（chpf）或者退出 OBSPY 时，将自动添加 “10” 指令卡到震相读取文件中。

12.165　width

概要

控制线宽

语法

```
WIDTH [ON|OFF|width] [SKELETON width] [INCREMENT ON|OFF]
    [LIST STANDARD|widthlist]
```

其中 width 只能取整数

输入

WIDTH width 设置数据的线宽为 width

WIDTH ON 打开 WIDTH 选项但是不改变当前线宽值

WIDTH OFF 关闭 WIDTH 选项

SKELETON width 设置图形边框宽度为 width

LIST STANDARD 设置为标准线宽列表，设置数据宽度为标准列表中的第一个宽度，并打开 WIDTH 选项

LIST widthlist 改变宽度列表的内容。输入宽度列表。设置数据宽度为列表中的第一个宽度，并打开 WIDTH 选项

INCREMENT ON 按照 widthlist 表中的次序，依次改变一个宽度值

INCREMENT OFF 关闭线宽递增功能

缺省值

```
width off skeleton 1 increment off list standard
```

说明

width 指定了绘制数据时的线条宽度。SKELETON 指定了坐标轴的宽度，其就仅修改坐标轴的宽度，网格、文本、标签和框架号总是用 1 号细线表示。

若将 WIDTH 设置为递增，则每次绘图之后，宽度都会按照宽度表中的顺序自动修改。 如果在同一张绘图中同时绘制几个数据文件，也许需要对每个文件使用不同的宽度。此时可使用 INCREMENT 选项。在这个选项打开时，每次绘制一个数据文件后，都按照宽度表中的次序自动地变成另一个宽度。宽度值和次序在标准宽度表中为:

```
1, 2, 3, 4, 5, 6, 7, 8, 9, 10
```

你可以使用 LIST 选项改变这个表的次序或内容。这个命令常用于重叠绘图（参见 plot2），此时你可能需要每张图上的数据宽度都按相同的顺序排列。

示例

选择自动变换的数据宽度起始值为 1：

```
OBSPY> width 1 increment
```

边框宽度起始值为 2，并按 1、3、5 的增量变化：

```
OBSPY> width skeleton 2 increment list 1 3 5
```

12.166 wiener

概要

设计并应用一个自适应 Wiener 滤波器

语法

```
WIENER [WINDOW pdw] [NCOEFF n] [MU OFF|ON|v] [EPSILON OFF|ON|e]
```

输入

WINDOW pdw 设置滤波器设计窗口为 pdw
NCOEFF n 设置滤波器系数为 n 个
MU off|on|v 设置自适应步长参数
MU Off 设置自适应步长参数为 0
MU ON 设置自适应步长为 1.95/Rho(0)，其中 Rho(0) 是 pdw 中延迟为 0 时的自相关系数
MU v 设置自适应步长为 v
EPSILON ON|OFF|e 设置岭回归（ridge regression）参数为。若为 OFF，则 OBSPY 将依次设置 epsilon 值为 0.0、10^{-5}、10^{-4}、10^{-3}、10^{-2}，直到滤波器稳定为止。若 epsilon=0 不稳定，则 OBSPY 会给出警告信息，若所有值均不稳定，则 OBSPY 会给出错误信息。

缺省值

```
wiener window b 0 10 ncoeff 30 mu off epsilon off
```

说明

对文件的 pdw 内的数据估计自相关函数，并利用 Yule-Walker 方法设计预测误差滤波器，然后将滤波器应用到整个信号，即信号被误差序列替换。该滤波器可以用作预白化或用作瞬时信号的检 测预处理器。若 mu 指定为非 0 值，则滤波器为时域自适应的，大值 mu 可能会导致不稳定。

岭回归（ridge regression ）参数通过给自相关矩阵的对角线元素加上 epsilon 以达到稳定 wiener 滤波器的目的。

示例

下面的命令将应用一个非自适应滤波器，将第一个十秒指定为数据窗：

```
OBSPY> wiener window b 0 10 mu 0.
```

下面命令将应用带 40 个系数的滤波器，指定设计窗为从文件开始到第一个到时前 1 秒：

```
OBSPY> wiener ncoeff 40 window b a -1
```

头段变量

depmin、depmax、depmen

12.167　wild

概要

设置读命令中用于扩展文件列表的通配符

语法

```
WILD [ECHO ON|OFF] [SINGLE char] [MULTIPLE char] [CONCATENATION chars]
```

输入

ECHO ON 打开扩展文件表回显选项；当该选项打开时，会回显被通配符展开的文件列表
ECHO OFF 关闭扩展文件表回显开关
SINGLE char 修改用于匹配单个字符的通配符
MULTIPLE char 修改用于匹配多个字符的通配符
CONCATENATION chars 修改用于将联接字符串括起来的字符

缺省值

选项	UNIX	VAX	PRIME
echo	on	on	on
single	?	?	+
multiple	*	*	'
concatenation	[,]	(,)	[,]

说明

很多现代操作系统都提供了通配符特性，也可以称为文件扩展。它是一个可以让你使用简短文 件名以及简单的简写形式去指定一组文件的表示符号。OBSPY 在 read、readtable 以及 readhdr 命令中使用通配符及一些扩展名，使用这些表示符号，你可以很容易地访问一组文件：

- 所有以字母abc开头的文件
- 所有以z结尾的文件
- 所有文件名中严格包含三个字母的文件 通配符代号有三个元素。对于不同的系统三个元素会有不同的缺省符。你可以使用这个命令改

变通配符。多重匹配字符（*）用于匹配字符串中任意字符串，包括空字符串。单个匹配符（?）用于匹配任意单个字符。连接符号（[　和　]）用于包围由逗号分隔的要匹配的字符串。在这个字符串中，可以包含单通配符或多通配符。

OBSPY 使用通配符完成文件名的扩展，通常有几个步骤：

1. 如果标识目录部分存在的话将将其去掉，否则使用当前目录
2. 做系统调用，以得到目录中所有文件的列表
3. 　　如果在标识中是一个连接表，就用其他字符形成连接表中每个字符的新的标识，然后匹配它们到文件表中。如果没有连接表标识，则可简单匹配标识到文件表
4. 去掉形成扩展文件表的所有重复的匹配

5. 如果需要，回显扩展文件表
6. 试着将扩展文件表读入内存 每个操作系统都使用一些不同的步骤在一个目录中存取文件。上面第一步的系统调用反映了这

些不同。例如，在 UNIX 中以字母顺序显示文件名，但在 PRIME 或 VAX 上就不是这样。在 PRIME 目录中文件次序是随意的。这些不同反映在扩展文件表的文件次序上。你可以用各种不同的通配符 和连接表进行实验，以确定扩展文件表中的文件次序是否重要。

下例将帮助你理解怎样使用这些通配符元素，一个有用的特征是 OBSPY 保存包含在连接表上的 字符串，当你输入一个空表，则前面的表将被重复使用，这可以节省许多输入的操作。

示例

假定当前目录中包含如下次序的文件：

```
ABC DEF STA01E STA01N STA01Z STA02E STA02N STA02Z STA03Z
```

同样假定扩展文件设置回显，下面显示怎样使用各种通配符去将上面文件表的一部分读入内存：

```
OBSPY> READ S*
 STA01E STA01N STA01Z STA02E STA02N STA02Z STA03Z
OBSPY> READ *Z
 STA01Z STA02Z STA03Z
OBSPY> READ ???
 ABC DEF
OBSPY> READ
STA01[Z,N,E] STA01Z
STA01N  STA01E OBSPY>
READ *[Z,N,E]
 STA01Z STA02Z STA03Z STA01N STA02N STA01E STA02E
OBSPY>  READ  *1[Z,N,E]  *2[ ]
```

限制

在一个标识中只可以有一个连接串

12.168 window

概要

设置图形窗口位置和宽高比

语法

```
WINDOW n [XSIZE xwmin xwmax] [YSIZE ywmin ywmax]
    [ASPECT [value|ON|OFF]]
```

输入

n 要设置属性的图形窗口号，n 取值 1 到 10

XSIZE xwmin xwmax 设置图形窗口的水平位置。其中 xwmin 和 xwmax 分别是窗口左/右边界位置，其可以取值为 0.0 到 1.0。

YSIEZ ywmin ywmax 设置图形窗口的垂直位置。其中 ywmin 和 ywmax 分别是窗口左/右边界位置，其可以取值为 0.0 到 1.0。

ASPECT value|ON|OFF 设置宽纵比为 value。若打开 ASPECT 选项，则自动计算 xwmax，使得 xsize 与 ysize 的比值为 value，若 value 则未设定，则使用系统默认值。若打开了 ASPECT 选项，但是却没有指定 xsize 选项，则 APSECT 选项被关闭，并且使用默认的 xwmin 和 xwmax 值。

缺省值

下面列出前 5 个绘图窗口位置的缺省值：

表 12.4: OBSPY 标准窗口

n	xwmin	xwmax	ywmin	ywmax
1	0.05	0.65	0.45	0.95
2	0.07	0.67	0.43	0.93
3	0.09	0.69	0.41	0.91
4	0.11	0.71	0.39	0.89
5	0.13	0.73	0.37	0.87

缺省情况下 ASPECT 选项是打开的，其值为 11.0/8.5=1.294，因而 xwmax 默认不使用。

说明

OBSPY 使用的 X11 图形系统支持多窗口绘图。beginwindow 命令使得你可以选择接下来的绘图 命令要绘制在哪个图形窗口中。如果想要修改窗口的属性，则必须在使用 beginwindow 命令前使用 window 命令。

该命令可以控制每个 X 图形窗口出现时相对于屏幕左下角的位置以及窗口的宽高比。屏幕左下角的坐标为 (0,0)，右上角的坐标为 (1,1)。

默认情况下，使用编号为 1 的图形窗口。其水平方向的位置为 0.05 到 0.65，垂直方向的位置为 0.45 到 0.95，即窗口位于屏幕的左上角。图形窗口随着编号的增加不断右下角移动。

若关闭 ASPECT 选项，则图形窗口的宽高比由屏幕的宽高比决定。对于 4:3 的屏幕，默认宽高比为 1.6:1；对于 16:10 的屏幕，默认宽高比为 1.9:1。SGF 文件的宽高比为 4:3。

示例

设定图形窗口 1 的水平位置，垂直位置不变：

```
OBSPY> window 1 x 0.25 0.85
OBSPY> beginwindow 1
```

在这种情况下，显式指定了 XSIZE，因而 ASPECT 被自动设置为 OFF。

```
OBSPY> window 1 aspect 1.33 x 0.25 0.85
OBSPY> beginwindow 1
```

该命令与上面的命令相同，虽然设置了 aspect 的值，但由于指定了 XSIZE，因而 XSIZE 具有更高的优先级。

```
OBSPY> window 1 x 0.25 0.85 aspect 1.33
OBSPY> beginwindow 1
```

由于 APSECT 位于 XSIZE 后面，因而 ASPECT 的优先级高于 XSIZE 的优先级，该命令会忽略 xwmax，并固定宽高比为1.33。

12.169 write

概要

将内存中的数据写入磁盘

语法

```
WRITE [OBSPY|ALPHA|XDR] [DIR OFF|CURRENT|name] [KSTCMP]
    [OVER|APPEND text|PREPEND text|DELETE text|CHANGE text1 text2]
    filelist
```

输入

无参数 使用以前的数据格式和文件列表

OBSPY 以 OBSPY 二进制文件格式写入磁盘 **ALPHA** 写 OBSPY 字符数字型写入磁盘

XDR 用 OBSPY 二进制 XDR 格式写文件。这个格式用于实现不同构架的二进制数据的转换

DIR OFF 关闭目录选项，即写入当前目录

DIR CURRENT 打开目录选项并设置写目录为当前目录

DIR name 打开目录选项并设置写目录为 name。将所有的文件写入目录 name 中，其可以是相对路径或绝对路径

KSTCMP 使用 KSTNM 和 KCMPNM 头段变量为内存中每个数据文件定义一个文件名。生成的文件名将检查是否唯一，如果不唯一，则在文件名后加序号以避免文件名冲突

OVER 使用当前读文件列表作为写文件列表，用内存中的文件覆盖磁盘中的文件

APPEND text 在当前读文件列表的文件名后附加字符串 text 以创建写文件列表

PREPEND text 在当前读文件列表的文件名前附加字符串 text 以创建写文件列表

DELETE text 在当前读文件列表的文件名中删除第一次出现的 text 以创建写文件表

CHANGE text1 text2 将当前读文件表中每个文件名中第一次出现的 text1 修改为 text2 来创建写文件表

filelist 将写文件列表设置为 filelist，这个列表可以包含文件名、相对/ 绝对路径，不可以包含通配符

缺省值

```
write obspy
```

说明

该命令允许你在数据处理的过程中将结果写入磁盘。写磁盘文件时可以选择几种数据格式，内 存中的每个文件都将完整地写入到磁盘中。

多数情况下，你会选择使用 OBSPY 数据文件格式，这是 OBSPY 软件的标准输入输出格式，用于快 速读写，其包含了以一个头段区和一个数据区。具体可以参考 OBSPY 文件格式一章。

你可以直接指定写文件名，也可以通过修改内存中的当前文件名间接地指定它们。OVER 选项把写文件表设置为读文件表。它用于覆盖包含当前内存的数据的读入的最后一组磁盘文件。APPEND、PREPEND、DELETE、CHANGE 选项通过以所需要的方式修改每个读文件名的方式建立一个写文件表，

这在宏命令中非常有用，在宏命令中你通常需要自动处理大量数据文件，并保持输出文件风格的一 致。当使用这四个选项中的任意一个时，命令执行时会在终端输出文件列表，使得你可以看到实际 写入磁盘的文件名。

示例

该命令的简单示例可以参考 OBSPY 的读和写一节。 对一组数据文件进行滤波，然后将结果存入一组新数据文件：

```
OBSPY> read d1 d2 d3
OBSPY> lowpass butter
npoles 4 OBSPY> write f1 f2
```

也可以使用 CHANGE 选项完成这一操作：

```
OBSPY> read d1 d2 d3
OBSPY> lowpass butter
npoles 4 OBSPY> write
```

若想要用滤波后的数据替换磁盘中的原始数据，则上例的第三行要变成：

```
OBSPY> write over
```

BUGS

- 使用dir off和dir current选项会直接报错，因而关键字off和current会被当作普通目录名，而由于目录不存在因而无法写入

12.170 writebbf

概要

将黑板变量文件写入到磁盘

语法

```
WRITEBBF [file]
```

输入

file 黑板变量文件的文件名

缺省值

```
writebbf bbf
```

说明

该命令让你能够将当前会话的所有黑板变量写入到磁盘文件中，稍后可以使用 readbbf 命令将黑板变量文件重新读入 OBSPY，该特性允许你保存某次 OBSPY 会话的信息，并用于另一次 OBSPY 会 话中。你也可以在自己的程序中调用 OBSPY 函数库以访问黑板变量文件中的信息。

12.171 writecss

概要

将内存中的数据以 CSS 3.0 格式写入磁盘

语法

```
WRITECSS [BINARY|ASCII] [DIR ON|OFF|CURRENT|name] name
```

输入

ASCII 以标准 ASCII 格式写入磁盘

BINARY 以 CSS 3.0 二进制文件输出

DIR ON 打开目录选项，但不改变目录名

DIR OFF 关闭目录选项，即写入当前目录

DIR CURRENT 打开目录选项并设置写目录为当前目录

DIR name 打开目录选项并设置写目录为 name 。将所有文件写入目录 name 中，其可以是相对路径或绝对路径

name 以文件名 name 写入磁盘。只能指定一个名字其不能包含通配符。对于 ASCII 型输出，以 name 为基础，在其后加上各个 CSS 表所对应的后缀（比如：name.wfdisc、name.origin）。对于 BINARY 输出，则 name 是输出文件名

缺省值

```
wirtecss ascii dir off
```

说明

该命令允许你在数据处理过程中将数据以 CSS 3.0 格式保存到磁盘中。在 ASCII 模式下（默认模式），会写入一个或多个 ASCII 文件到磁盘。输出的具体文件依赖于数据来源，但输出可以是下面列出的 CSS 表中的任意一个或多个：

```
wfdisc, wftag, origin, arrival, assoc, sitechan, site, affiliation,
origerr, origin, event, sensor, instrument, gregion, stassoc, remark Obspydata
```

在 BINARY 模式下，所有的 CSS 表以及波形数据都会以二进制格式写入到同一个文件中。

12.172 writehdr

概要

用内存中文件的头段区覆盖磁盘文字中的头段区

语法

```
WRITEHDR
```

说明

write 命令的 over 选项可以用内存中头段区和数据区覆盖磁盘文件中的头段区和数据区。该命令用内存中头段区覆盖磁盘文件中的头段区，数据区不会被覆盖。如果使用了 cut 命令，读取数据时将仅读入部分数据，内存中的头段区将会做相应修改以反映 cut 命令的效果，但是磁盘中的数据并没有被修改，因而此时不能使用 writehdr 命令。对被 cut 的数据使用 writehdr 命令将可能导致磁盘中的数据产生类似于平移或截断的效果。

12.173 writesp

概要

将谱文件作为一般文件写入磁盘

语法

```
WRITESP [ASIS|RLIM|AMPH|RL|IM|AM|PH] [OVER|filelist]
```

输入

ASIS 按照谱文件当前格式写入
RLIM 写入实部和虚部分量
AMPH 写入振幅和相位分量
RL 只写入实部分量
IM 只写入虚部分量
AM 只写入振幅分量
PH 只写入相位分量
filelist OBSPY 二进制数据文件列表，这个列表可以包含简单文件名和绝对/相对路径名

缺省值

```
writesp asis
```

说明

OBSPY数据文件可以为时间序列文件或谱文件。头段中的 IFTYPE 用于区分这两种格式。当你读取一个时间序列到内存，对其做快速 Fourier 变换，然后将数据写回磁盘，此时的文件即为谱文件。

某些操作只能对时间序列文件进行，而某些操作只能对谱文件进行。比如，你无法对一个谱文件应用 taper 命令或者将两个谱文件乘起来。这是 OBSPY 的保护机制。

然而有时你需要对谱文件做这些操作，为了越过 OBSPY 的保护机制，你可以使用这个命令将谱文件像时间序列数据一样写入磁盘。每一个分量都将作为一个单独文件写入磁盘。然后你可以将这些文件读入 OBSPY 并进行任何你想要的操作。因为 OBSPY 认为其为时间序列文件。一旦这些计算完成了，你可以将修改之后的数据通过 write 命令写回磁盘。如果你想要读回这个谱文件，可以使用 readsp 命令。

为了帮助你跟踪磁盘上的数据，OBSPY 将在你给出的文件名后加一个后缀以标识储存在文件的谱分量。后缀分别为 .RL、.IM、.AM 和 .PH 分别对应不同的分量。

示例

假设你想要对 FILE1 的谱文件振幅进行一些操作：

```
OBSPY> read
file1 OBSPY> fft
amph
```

OBSPY 将输出两个文件 FILE1.AM 和 FILE1.PH，现在可以对振幅文件进行操作：

```
OBSPY> read file1.am
OBSPY> ...perform
operations. OBSPY> write
```

现在磁盘中的文件为修改后的谱文件，如果你想要重建 OBSPY 谱数据并进行反变换：

```
OBSPY> readsp
file1 OBSPY> ifft
OBSPY> write file2
```

头段变量改变

磁盘文件中的 b、e、delta 将包含频率的起始值、结束值和增值，单位为 Hz

12.174 xdiv

概要

控制 x 轴的刻度间隔

语法

```
XDIV [NICE|INCREMENT v|NUMBER n] [POWER ON|OFF]
```

输入

NICE 由 OBSPY 自动选择合适的刻度间隔
INCREMENT v 设置刻度间隔增量为 v
NUMBER n 设置刻度的总数目数为 n
POWER ON 打开幂指数选项，OBSPY 以幂指数形式给出刻度值
POWER OFF 关闭幂选项

缺省值

```
xdiv nice power on
```

说明

该命令控制 X 轴的刻度间隔。多数时候默认的 nice 间隔即可满足需求。OBSPY 的 nice 刻度间 隔是根据坐标轴的最小最大值、坐标轴的长度以及当前坐标轴文本尺寸来决定的。

也可以使用 INCREMENT 选项强制刻度间隔为一个定值，或者使用 NUMBER 选项设置刻度间隔的数目。

12.175　xfudge

概要

设置 X 轴范围的附加因子

语法

```
XFUDGE [ON|OFF|v]
```

输入

v 设置附加因子为 v
ON 打开附加选项，但不改变附加因子
OFF 关闭附加选项

缺省值

```
xfudge 0.03
```

说明

当坐标轴的范围设置为数据的时间极值时，xmin=b，xmax=e。实际绘图时会将 xmin 调小一点，将 xmax 调大一点，使得绘图时波形的两端与边框之间留有一些空隙。附加因子定义了这个空隙相对于时间极值的比例。

实际绘图时，会根据附加因子计算新的坐标轴范围：

```
xdiff = xfudge * ( b - e )
xmin = b - xdiff
xmax = e + xdiff
```

其中 b 和 e 是数据的时间范围，xfudge 是附加因子，xmin 和 xmax 是计算得到的 X 坐标轴范围。该算法对对数坐标也得类似的效果。

12.176　xfull

概要

控制 X 轴的坐标范围的极值为 10 的倍数

语法

```
XFULL [ON|OFF]
```

输入

ON|OFF 打开/关闭整对数绘图选项

缺省值

```
xfull on
```

说明

该命令仅适用于使用对数坐标且坐标范围不固定（xlim off）的情况。当此选项打开时，实际的坐标轴范围将被设置为略大于数据范围的 10 的倍数。当关闭这个选项时，将使用实际数据范围。

12.177 xgrid

概要

控制绘图时的 x 方向的网格线

语法

```
XGRID ON|OFF|SOLID|DOTTED
```

输入

SOLID 设置网格线为实线
DOTTED 设置网格线为虚线
ON 绘制网格，但不改变网格类型
OFF 不绘制网格

缺省值

```
xgrid off
```

说明

这个命令控制 X 坐标轴网格的绘制。

12.178 xlabel

概要

定义 X 轴标签及属性

语法

```
XLABEL [ON|OFF|text] [LOCATION TOP|BOTTOM|RIGHT|LEFT]
    [SIZE TINY|SMALL|MEDIUM|LARGE]
```

输入

text 打开 X 轴标签选项，设置标签为 text。若文本含空格，需要用引号括起来
ON 打开 X 轴标签选项，但不改变标签文本
OFF 关闭 X 轴标签选项
LOCATION 设定 X 轴标签的位置。可以取 TOP、BOTTOM、RIGHT、LEFT
SIZE 改变绘图标签的文本尺寸
TINY 每行 132 个字符
SMALL 每行 100 个字符
MEDIUM 每行 80 字符
LARGE 大尺寸，每行 50 字符

缺省值

```
xlabel off location bottom size small
```

说明

若打开 X 轴标签选项，则绘图时会在图上显示 X 轴标签。标签的尺寸和位置以及文本均可以改变。文本质量以及字体可以使用 `gtext` 命令设置。

12.179 xlim

概要

设定图形中 X 轴的范围

语法

```
XLIM [ON|OFF|pdw|SIGNAL]
```

OBSPYTitle 输入

pdw 打开 X 轴范围选项并设置范围为新的 “partial data window”，参考 pdw
SIGNAL 等同于输入 `A -1 F +1`，即初至前 1 秒到事件结束的后 1 秒
ON 打开 x 轴范围选项，但不改变 X 轴范围值
OFF 关闭 x 轴范围选项，即根据数据的自变量范围决定 X 轴范围

缺省值

```
xlim off
```

说明

当此选项关闭时，会根据数据自变量的范围决定绘图时 X 轴的范围。当此选项打开时，限定 X 轴的范围，可以通过此种方式 “放大” 当前内存中数据的图形。

12.180 xlin

概要

设置 X 轴为线性坐标

语法

```
XLIN
```

12.181 xlog

概要

设置 X 轴为对数坐标

语法

```
XLOG
```

12.182 xvport

概要

定义 X 轴的视口

语法

```
XVPORT xvmin xvmax
```

输入

xvmin X 轴视口的最小值，范围为 0.0 到 xvmax
xvmax X 轴视口的最大值，范围为 xmin 到 1.0

缺省值

```
xvport 0.1 0.9
```

说明

视口（viewport）是视窗（viewspace）的一部分。在 OBSPY 中，定义视口时使用的是相对坐标系，即坐标系在 X 和 Y 方向都是 0 到 1 的范围。视窗左下角的坐标为 (0.0, 0.0)，右上角的坐标为 (1.0, 1.0)。该坐标系是与输出设备无关的，因而可以很方便地指定一个图形的位置。

xvport 和 yvport 定义了视口相对于视窗的位置，后续的命令将在定义的视口中绘图。默认值 xvport 0.1 0.9 在 X 方向上使用了视窗的 80%，在图形的左右两边留下一些空间绘制坐标轴、标签和标题。

当与 beginframe 和 endframe 命令一起使用时，可以让你能够在一个视窗内绘制若干不同的图形，构成复杂的组合图。

示例

参见 组合图 一节。

12.183 ydiv

概要

控制 Y 轴的刻度间隔

语法

```
YDIV [NICE|INCREMENT v|NUMBER n] [POWER ON|OFF]
```

说明

该命令的选项、默认值和说明，与 xdiv 相同。

12.184　yfudge

概要

设置 Y 轴范围的附加因子

语法

```
YFUDGE [ON|OFF|v]
```

说明

该命令的选项、默认值及说明，与 xfudge 类似。

12.185　yfull

概要

控制 Y 轴的坐标范围的极值是 10 的倍数

语法

```
YFULL [ON|OFF]
```

输入

ON|OFF　打开/关闭整对数绘图选项

缺省值

```
yfull  on
```

说明

该命令仅适用于使用对数坐标且坐标范围不固定（ylim　off）的情况。当此选项打开时，实际的坐标轴范围将被设置为略大于数据范围的　10　的倍数。当关闭这个选项时，将使用实际数据范围。

12.186　ygrid

概要

控制绘图时的 y 方向的网格线

语法

```
YGRID  ON|OFF|SOLID|DOTTED
```

输入

SOLID　设置网格线为实线
DOTTED 设置网格线为虚线
ON　绘制网格，但不改变网格类型
OFF　不绘制网格

缺省值

```
ygrid off
```

说明

这个命令控制 Y 坐标轴网格的绘制。

12.187 ylabel

概要

定义Y轴标签及属性

语法

```
YLABEL [ON|OFF|text] [LOCATION TOP|BOTTOM|RIGHT|LEFT]
    [SIZE TINY|SMALL|MEDIUM|LARGE]
```

说明

该命令的输入、默认值、说明，与 xlabel 相同。

12.188 ylim

概要

设定图形Y轴的范围

语法

```
YLIM [ON|OFF|ALL|min max|PM v]
```

输入

ALL 根据内存中所有文件的最大和最小值限定 Y 轴的范围
min max 设定 Y 轴的范围为 min 到 max 之间
PM v 设置 Y 轴的范围为 -v 到 +v 之间
ON 打开Y轴范围选项，但不改变范围值
OFF 关闭 Y 轴范围选项

缺省值

```
ylim off
```

说明

当内存中有多个数据文件需要绘制时，若关闭此选项，每个数据文件在绘图时都会根据自己的 数据因变量的范围决定绘图时 Y 轴的范围。当此选项打开时，将限制所有绘图的 Y 轴的范围。ALL 选项会找到内存中所有数据文件的最大值和最小值作为所有绘图的Y轴的范围。

可以为内存中不同的文件设定不同的Y轴范围，只要在命令中多次使用这些选项即可。

示例

file1 的 Y 轴范围为 0.0 到 30.0，file2 的 y 轴范围为内存中所有文件的最大、最小值，file3 的 y 轴范围将限定为文件自身的最大、最小值。如果文件多于三个，则其余的所有文件都限定为文件自 身的最大、最小值。

```
OBSPY> ylim 0.0 30.0 all
off OBSPY> r file1 file2
file3 OBSPY> p
```

12.189 ylin

概要

设置Y轴为线性坐标

语法

```
YLIN
```

12.190 ylog

概要

设置Y轴为对数坐标

语法

```
YLOG
```

12.191 yvport

概要

定义Y轴的视口

语法

```
YVPORT yvmin yvmax
```

输入

yvmin Y 轴视口的最小值，范围为 `0.0` 到 `yvmax`
yvmax Y 轴视口的最大值，范围为 `yvmin` 到 `1.0`

缺省值

```
yvport 0.1 0.9
```

说明

参考 `xvport` 命令的说明。

12.192 zcolors

概要

控制等值线的颜色显示

语法

```
ZCOLORS [ON|OFF] LIST c1 c2 ... cn
```

输入

ON 打开等值线颜色显示开关

OFF 关闭等值线颜色显示开关

LIST c1 c2 . cn 设置等值线要使用的颜色列表，每一个颜色对应一条等值线，如果等值线数目多于这个列表长度，则整个列表不断重复

cn OBSPY 当前颜色表的颜色名

缺省值

```
zcolors off list red green blue
```

示例

参考 “等值线图” 中的相关示例。

12.193 zlabels

概要

根据等值线的值控制等值线的标记

语法

```
ZLABELS [ON|OFF] [SPACING v1 [v2 [v3]]] [SIZE v]
    [ANGLE v] [LIST c1 c2 ... cn]
```

`LIST` 选项只能放在这个命令的最后

输入

ON|OFF 打开/关闭等值线标签选项开关

SPACING v1 v2 v3 设置相邻标签名的最小、适中和最大间隔（视口坐标系）分别为 `v1`、`v2` 和 `v3`。如果第二、三个值省略则使用前面一个值

SIZE v 设置标签的尺寸（高度）为 `v`

ANGLE v 设置标签文本最大角度为 `v`（自水平方向起算的角度，单位为度）

LIST c1 c2 . cn 设置使用的等值线标签的列表。在这个表上的每个输入用于相应的等值线，如果等值线数目大于这个表的长度，则重复使用整个等值线表

cn 可以取 `ON|OFF|INT|FLOATn|EXPn|text`

ON 在相应的等值线上放置标签，使用 Fortran 自由格式，用等值线值形成标签名

OFF 在相应的等值线上不放置标签名

INT 在相应的等值线上放置整数标签名

FLOATn 在相应的等值线上放置小数点后面 n 位的浮点数作为标签名。如果 n 被忽略则使用先前值

EXPn 在相应的等值线上放置小数点后面 n 位数的指数幂形式标签名，如果 n 忽略则使用先前值

text 使用文本标注相应的等值线

缺省值

```
zlabels off spacing 0.1 0.2 0.3 size 0.0075 angle 45.0 list on
```

示例

参考 “等值线图” 中的相关示例。

12.194 zlevels

概要

控制后续等值线图上的等值线间隔

语法

```
ZLEVELS [SCALE] [RANGE v1 v2] [INCREMENT v] [NUMBER n]
    [LIST v1 v2 ... vn]
```

输入

SCALE 根据数据自动确定等值线的标尺范围

RANGE v1 v2 用户设置等值线的范围为 v1 到 2。可以使用 SCALE 选项，也可以使用 RANGE 选项，但不可同时使用二者

INCREMENT v 设置等值线之间的增量为 v

NUMBER n 设置等值线的条数为 n，INCREMENT 和 NUMBER 选项只能二选一

LIST v1 v2 .. vn 设置一系列等值线上的值为 v1、v2 等等，如果使用这个选项，则其他选项均被忽略

缺省值

```
zlevels scale number 20
```

示例

参考 “等值线图” 中的相关示例。

限制

等值线的最多数目为 40

12.195 zlines

概要

控制后续等值线绘图上的等值线线型

语法

```
ZLINES [ON|OFF] [LIST n1 n2 ... nn] [REGIONS v1 v2 ... vn]
```

输入

ON|OFF 打开等值线显示选项

LIST n1 n2 .. nn 设置要使用的线型表，这个表上的每个输入用于相应的等值线。如果等值线的数目大于这个表中给出的线型的数目，则使用整个线型表

REGIONS v1 v2 .. vn 设置等值线范围表。这个表的长度应小于线型表的长度，小于范围值的等值线使用线型表中相应的线型。超过最后一个范围值的等值线采用线型表中最后一个 线型的值

缺省值

```
zlines on list 1
```

示例

循环四种不同线型，建立等值线：

```
OBSPY> zlines list 1 2 3 4
```

设置虚线表示低于 0.0 等值线，实线表示高于 0.0 的等值线：

```
OBSPY> zlines list 2 1 regions 0.0
```

12.196 zticks

概要

用方向标记标识等值线

语法

```
ZTICKS [ON|OFF] [Spacing v] [LENGTH v] [DIRECTION DOWN|UP]
    [LIST c1 c2 ... cn]
```

输入

ON|OFF 打开/关闭等值线方向标记

SPACING v 在每条线段上设置项链标识之间的间隔为 v（视口坐标系）

LENGTH v 设置每个标识的长度为 v（视口坐标系）

DIRECTION DOWN|UP 标识在 z 值减小/增加的方向上

LIST c1 c2 . cn 设置要使用的等值线标识表。在这个表上的每个输入都用于相应的等值线。如果等值线数多于这个列表的长度，则重复使用整个标识表。ON 意味着标识画在等值线上，OFF 意味着标识不画在等值线上

缺省值

```
zticks off spacing 0.1 length 0.005 direction down list on
```

示例

参考 “等值线图” 中的相关示例。

保护环境，从阅读电子文档开始！

第 13 章　SSS

13.1　信号迭加子程序

Signal Stack Subprocess，是 OBSPY 提供的一个用于信号迭加的子程序。

在 OBSPY 中键入“sss”即可进入该子程序；在子程序中键入 quitsub 即可退出子程序并回到主 程序；也可键入 quit 直接从子程序中退出 OBSPY。

在对多个信号进行迭加时，每个信号都有各自的属性，比如静延迟、震中距、权重因子、数据极 性，也可以根据 normal moveout 或折射波速度模型计算动延迟。

该子程序具有如下特点：

- 延迟属性可以在迭加过程中自动递增；
- 文件可以很容易地从迭加文件列表中增添；
- 迭加时间窗也可以很容易调整；
- 若文件在迭加时间窗内不含数据，则将其置零值；
- 迭加文件列表可以单独绘制，也可以绘制迭加后的结果；
- 每次迭加结果都可以保存到磁盘上；
- 支持绘制记录剖面图；

在 SSS 子程序中，你可以执行一系列 SSS 专属的命令，以及部分 OBSPY 主程序中的命令。下面 仅列出 SSS 专属的命令：

- addstack 向迭加文件列表中加入新文件
- changestack 修改当前迭加文件列表中的文件属性
- deletestack 从迭加文件列表中删除一个或多个文件
- deltacheck 修改采样率检测选项
- distanceaxis 定义剖面图中距离轴的参数
- distancewindow 控制接下来的剖面图的距离窗属性
- globalstack 设置全局迭加属性
- incrementstack 迭加文件列表中文件的增量属性
- liststack 列出当前迭加文件列表中文件的属性
- plotrecordsection 用迭加文件列表中的文件绘制剖面图
- plotstack 绘制迭加文件列表中的文件
- sumstack 对迭加文件列表中的文件进行迭加
- timeaxis 控制接下来剖面图的时间轴属性
- timewindow 设置迭加的时间窗
- traveltime 根据预定义的模型计算走时
- velocitymodel 用于计算动延迟的迭加速度模型参数
- velocityroset 控制剖面图中速度 roset 的放置

- writestack 将迭加结果写入磁盘
- zerostack 重新初始化信号迭加

13.2 addstack

概要

向迭加文件列表中加入新文件

语法

```
ADDSTACK filename [WEIGHT v] [DISTANCE v]
    [BEGINTIME v] [ENDTIME v]
    [DELAY v [SECONDS|POINTS]]
    [INCREMENT v [SECONDS|POINTS]]
    [NORMAL|REVERSE]
```

输入

filename 要加入迭加文件列表中的文件

WEIGHT v 当前文件的权重因子。v 的取值范围为 0 到 1，在迭加之前会首先对文件的每个值乘以该权重因子再做迭加。

DISTANCE v 该文件所对应的震中距，单位为 km。用于计算动态时间延迟

BEGINTIME v 事件开始的时间

ENDTIME 事件结束时间

DELAY v SECONDS|POINTS 该文件的静态时间延迟，单位为秒或数据点数

INCREMENT v SECONDS|POINTS 该文件的静态时间延迟增量，单位为秒或数据点数。在每次执行 incrementstack 命令时，静态时间延迟会增加一个常数。

NORMAL|REVERSED 文件拥有正/负极性

缺省值

DELAY 和 INCREMENT 选型的缺省单位为 SECONDS

说明

每个迭加列表中的文件，都有 7 个属性，分别为：

1. 权重因子；
2. 台站到震中的距离；
3. 事件开始时间；
4. 事件结束时间；
5. 静态时间延迟；
6. 静态时间延迟增量；
7. 数据极性；

可以用 globalstack 命令为这些迭加属性设置全局值。当一个文件通过 addstack 命令加入到迭加文件列表中时，若未指定属性值则使用全局属性值。changestack 可以用于在文件已经加入迭加文件列表后修改迭加属性值。

示例

```
OBSPY/SSS> gs delay 1.0 inc
0.03 OBSPY/SSS> as filea delay
2.0
OBSPY/SSS> as fileb delay 3.0 inc 0.01
rev OBSPY/SSS> as filec
```

第一个命令修改了时间延迟和时间延迟增量的全局属性值，其他全局属性则使用缺省值。

- filea：除时间延迟外其他属性均与全局属性相同；
- fileb：除时间延迟、时间延迟增量、信号极性外，其余属性均与全局属性相同；
- filec：所有属性与全局熟悉相同；
- filed：除权重因子外，其他所有属性与全局属性相同；　接下来

对信号进行迭加：

```
OBSPY/SSS> sumstack
```

该命令会讲迭加文件列表中的四个文件 filea、fileb、filec 和 filed 进行迭加，时间延迟分别为 2.0、3.0、1.0 和 1.0。文件 filec 的极性反转。文件 filed 在迭加时的权重是其他文件权重的一半。

```
OBSPY/SSS> incrementstack
OBSPY/SSS> changestack filec
normal OBSPY/SSS> sumstack
```

此次迭加，各个文件使用 2.03、3.01、1.03 和 1.03 的延迟。文件 filec 现在为正极性。

```
OBSPY/SSS> deletestack
filed OBSPY/SSS>
incrementstack OBSPY/SSS>
```

第三次迭加讲只对文件 filea、fileb、filec 进行，时间延迟分别为 2.06、3.02、1.06。

限制

迭加文件列表中文件数目的最大限制与 OBSPY 所能读取的文件数目一致，即最多 1000 个。

13.3 changestack

概要

修改当前迭加文件列表中的文件属性

语法

```
CHANGESTACK filename|filenumber [WEIGHT v] [DISTANCE v]
    [BEGINTIME v] [ENDTIME v] [DELAY v SECONDS|POINTS]
    [INCREMENT v SECONDS|POINTS] [NORMAL|REVERSED]
```

输入

filename 迭加文件列表中的文件名
filenumber 迭加文件列表中的文件号

WEIGHT v 当前文件的权重因子。v 的取值范围为 0 到 1，在迭加之前会首先对文件的每个值乘以该权重因子再做迭加。

DISTANCE v 该文件所对应的震中距，单位为 km。用于计算动态时间延迟

BEGINTIME v 事件开始的时间

ENDTIME 事件结束时间

DELAY v SECONDS|POINTS 该文件的静态时间延迟，单位为秒或数据点数

INCREMENT v SECONDS|POINTS 该文件的静态时间延迟增量，单位为秒或数据点数。在每次执行 incrementstack 命令时，静态时间延迟会增加一个常数。

NORMAL|REVERSED 文件拥有正/负极性

说明

该命令允许你修改修改迭加文件列表中任意文件的任意属性。详情参考 addstack 命令。

13.4 deletestack

概要

从迭加文件列表中删除一个或多个文件

语法

```
DELETESTACK filename|filenumber
```

输入

filename 迭加文件列表中的文件名

filenumber 迭加文件列表中的文件号

13.5 deltacheck

概要

修改采样率检测选项

语法

```
DELTACHECK ON|OFF|ROUNDOFF|v
```

输入

ON 打开采样率检测选项

OFF 关闭采样率检测选项

ROUNDOFF 打开采样率检测选项，并强制采样率符合当前机器的 roundoff 因子

v 打开采样率检测选项，并强制采样率允许存在 v 的偏差

缺省值

```
deltacheck roundoff
```

说明

该命令用于修改采样率检测选项的行为。若该选项关闭，则不检测迭加文件列表中的文件是否 具有相同的采样率。若该选项设置为开，则文件的采样率必须在一个能够容忍的范围之内，否则则 被会报错。容错范围可以设置为 roundoff 因子或某个特定的值。

迭加文件列表中的所有文件的采样率之间的差值的绝对值不能超过容错范围。

13.6 distanceaxis

概要

定义剖面图的距离轴参数

语法

```
DISTANCEAXIS FIXED v | SCALED v
```

输入

FIXED v 固定距离轴的长度为 v 厘米
SCALED v 设定距离轴的长度为总距离范围除以 v，v 的单位为 cm/km。

缺省值

```
distanceaxis fixed 35
```

示例

若剖面图的距离范围为150km 到300km，则如下命令设置距离轴长度为75cm：

```
OBSPY> distanceaxis scaled 2.0
```

13.7 distancewindow

概要

控制接下来的剖面图的距离窗属性

语法

```
DISTANCEWINDOW [USEDATE|WIDTHi v|FIXED v1 v2]
    [UNITS KILOMETERS|DEGREES]
```

输入

USEDATA 使用迭加文件列表中文件的距离属性的最大最小值
WIDTH v 使用迭加文件列表中文件的距离属性的最小值，但强制其宽度为 v，即最大值为最小值加上 v
FIXED v1 v2 固定距离的最小、最大值分别为 v1 和 v2
UNITS KILOMETERS 设置距离窗的单位为 km [1]
UNITS DEGREES 设置距离窗的单位为度

[1] 该选项尚未实现

缺省值

```
distancewindow usedata units kilometers
```

13.8 globalstack

概要

设置全局迭加属性

语法

```
GLOBALSTACK [WEIGHT v] [DISTANCE v] [DELAY v [SECONDS|POINTS]]
    [INCREMENT v [SECONDS|POINTS] [NORMAL|REVERSED]
```

输入

WEIGHT v 全局权重因子，取值为 0 至 1；
DISTANCE v 全局震中距，单位为 km；
DELAY v SECONDS|POINTS 全局静时间延迟，单位为 s 或数据点数；
INCREMENT v SECONDS|POINTS 全局静时间延迟的增量，单位为 s 或数据点数；
NORMAL|REVERSED 正/负极性；

说明

该命令用于定义全局迭加属性，这些全局迭加属性用于迭加文件列表中的每个文件。可以使用 addstack 命令为某个文件单独设定迭加属性。

13.9 incrementstack

概要

迭加文件列表中的增量属性

语法

```
INCREMENTSTACK
```

缺省值

缺省值为 0

说明

可以设定增量的属性包括静态时间延迟、视速度和速度模型中的截距时间。若属性增量为 0.0，则属性值不改变。

可以为视速度或速度模型截距时间设置增量，其他属性值则自动计算以保持在特定点的零时间 延迟。

示例

```
OBSPY/SSS>     addstack
filea          OBSPY/SSS>
addstack              fileb
OBSPY/SSS>     addstack
filec          OBSPY/SSS>
addstack filed
OBSPY/SSS> velocitymodel 1 refr vapp 7.9 vappi 0.1 \
                        tovm calc dist 320. tvm 45.
OBSPY/SSS> sumstack
OBSPY/SSS> writestack
stack1 OBSPY/SSS>
incrementstack OBSPY/SSS>
sumstack
OBSPY/SSS> writestack
```

上面的命令会产生三个迭加文件，即 stack1、stack2、stack3。迭加时使用折射波速度模型，视速度 VAPP 分别为 7.9、8.0、8.1。速度模型截距时间 TOVM 自动计算以保证在 320 km、45 s 处具有零时间延迟。

13.10 liststack

概要

列出迭加文件列表中的文件属性

语法

```
LISTSTACK [NARROW | WIDE]
```

输入

NARROW 使用“窄”报告格式。每个文件的信息输出为两行
WIDE 使用“宽”报告格式。每个文件的信息用 120 字符宽的一行表示

缺省值

```
liststack narrow
```

13.11 plotrecordsection

概要

用迭加文件列表中的文件绘制剖面图

语法

```
PLOTRECORDSECTION [LABLES ON|OFF|headerfield]
    [ORIGIN DEFAULT|REVERSED] [REFERENCELINE ON|OFF] [SIZE v]
    [WEIGHT ON|OFF] [POLARITY ON|OFF]
```

```
[CURSOR ON|OFF] [REDUCED ON|OFF|PHASE phasename|VELOCITY velocity]
[ASPECT ON|OFF] [ORIENT PORTRAIT|LANDSCAPE]
[TTIME ON|OFF|DEFAULT|TEXT]
[XLABEL ON|OFF|DEFAULT|TEXT] [YLABEL ON|OFF|DEFAULT|TEXT]
```

输入

LABELS ON|OFF 打开/关闭标签选项。若打开，则每个文件都用头段变量进行标签

LABELS headerfield 打开标签选项，病设置头段变量名

ORIGIN DEFAULT|REVERSED 在 Portrait 模式中，距离沿着 Y 轴，默认情况下距离原点位于左上角。在 landscape 模式下，距离沿着 X 轴，默认情况下原点位于左下角。

REFERENCELINE ON|OFF 开启/关闭参考线选项。若打开，则每个文件在距离属性值对应的地方绘制一条垂直虚线

SIZE v ?

WEIGHT ON|OFF 打开/关闭权重选项

POLARITY ON|OFF 打开/关闭极性选项

CURSOR ON|OFF

REDUCED ON|OFF|VELOCITY vel|PHASE phase reduced 走时曲线。可以指定 reduce 速度或者一个参考震相

ORIENT PORTRAIT|LANDSCAPE portrait 模式中，水平轴为时间，纵轴为震中距；landscape 模式下，水平轴为震中距，垂直轴为时间

TTIME ON|OFF|DEFAULT|TEXT 绘制走时曲线。需要首先用 `traveltime` 命令计算走时曲线

XLABEL ON|OFF|DEFAULT|TEXT 打开/关闭/设置 X 轴标签

YLABEL ON|OFF|DEFAULT|TEXT 打开/关闭/设置 Y 轴标签

缺省值

```
plotrecordsection labels filename origin default referenceline on
    size 0.1 weight on polarity on orient portrait reduced off
    cursor off ttime off
```

说明

该命令将利用迭加文件列表中绘制剖面图。在 portrait 模式下，X 轴为时间，Y 轴为震中距，在 landscape 模式下则交换 XY 轴。每个文件的零振幅将会画在距离轴上对应的震中距处。 为了能够正确绘图，迭加列表中的所有文件必须定义震中距属性，该属性可以来自于文件头段，也可以在 `globalstack`、`addstack`、`changestack` 等命令的 DISTANCE 选项中定义。

`distancewindow` 和 `timewindow` 命令可以控制要显示的数据窗。`distanceaxis` 和 `timeaxis` 命令控制横纵轴的尺寸。`velocitymodel` 定义了速度模型，用于计算动态延迟。`velocityroset` 命令用于控制速度 rosette 的显示效果。

光标模式

在光标模式下，有两个额外的功能：缩放和决定视速度。 缩放功能需要用户指定要显示的区域。

用户首先将光标放在当前图形区域的一个角落，键入 `c1`，再将光标移动到对角的另一个角落，键入 `c2`。两次键入确定了唯一的矩形区域，也确定了要绘制的区域的时间范围和距离范围，此时，会自动重新绘制缩放后的剖面图，用户可以键入 `o` 命令重新绘制原始图形。缩放功能最多可以递归 5 次。

视速度确定功能需要用于移动光标，并分别键入 v1 和 v2 以标记点，OBSPY 会自动计算视速度， 显示在输出设备上并保持到黑板变量 vapp 中。可以多次设置 v2，但只有最后一次的值会保存到黑 板变量中。

除了 c1、c2、v1、v2 之外，光标模式下还有一个命令，即 q，用于退出光标模式。

13.12 plotstack

概要

绘制迭加文件列表中的文件

语法

```
PLOTSTACK [SUM ON|OFF] [PERPLOT ON|OFF|n] [WEIGHT ON|OFF]
    [POLARITY ON|OFF]
```

输入

SUM ON|OFF 若打开该选项，则首先绘制迭加后的波形再绘制迭加文件列表中的文件；若关闭该选项，则只回执迭加文件列表中的文件

PERPLOT ON|OFF 若打开该选择，则每次只绘制固定数目的文件；若关闭该选项，则一次绘制迭加列表中的全部文件

PERPLOT n 打开 PERPLOT 选项，并设置每次绘制 n 个文件

WEIGHT ON|OFF 打开/关闭文件权重选项

POLARITY ON|OFF 打开/关闭文件极性选项

缺省值

```
plotstack sum on perplot off weight on polarity on
```

说明

该命令绘制迭加文件列表中的文件，所有的文件首先根据静/动延迟进行时移，该命令可以控 制绘制文件时是否考虑权重因子和极性。

该命令的用法与 plot1 类似，在每个子图的左上角会显示文件名以及其他非默认的属性值。

13.13 sumstack

概要

对迭加文件列表中的文件进行迭加

语法

```
SUMSTACK [NORMALIZATION ON|OFF]
```

输入

NORMALIZATION ON|OFF 打开/关闭归一化选项。若该选项打开，则对于迭加结果中的每个数据点除以所有文件的权重因子的和。

缺省值

该命令用于将迭加文件列表中的文件进行迭加。在该命令执行之前必须通过 timewindow 命令设置迭加时间窗。每个数据会根据其静/动时间延迟做相应的时移。对于不迭加时间窗内的数据直 接按零值处理。每个文件可以给定权重以及极性。

在迭加之后，会自动生成迭加结果的绘图。迭加结果可以通过 writestack 命令保存到磁盘中。

13.14 timeaxis

概要

控制剖面图的时间轴属性

语法

```
TIMEAXIS FIXED v | SCALED v
```

输入

FIXED v 固定时间轴的长度为 v cm

SCALED v 设定时间轴的长度为总时间窗长的 v 倍，v 的单位为 cm/s

缺省值

```
timeaxis fixed 23.0
```

示例

如果你在做多个不同时间窗长的剖面图，并希望剖面图中每秒对应 0.5 厘米长：

```
OBSPY> timeaxis scaled 0.5
```

13.15 timewindow

概要

设置迭加的时间窗范围

语法

```
TIMEWINDOW v1 v2
```

输入

v1 v2 读入数据时所使用的时间窗范围

缺省值

无缺省值，在迭加之前必须指定时间窗范围。

说明

该命令用于设置迭加时间窗，该设置会影响 sumstack、plotstack、plotrecordsection 等命令的执行效果，迭加时间窗必须在使用这些命令之前定义。

如果某个文件的数据落在迭加时间窗外，则对迭加时间窗内的数据补零值。

13.16 traveltime

概要

根据预定义的速度模型计算指定震相的走时，请参考 traveltime 命令中的相关说明。

13.17 velocitymodel

概要

设置计算动延迟时所使用的迭加速度模型参数

语法

```
VELOCITYMODEL n [ON|OFF] [REFRACTEDWAVE|NORMALMOVEOUT]
    [FLIP] [VAPP v|CALCULATE] [T0VM v|CALCULATE]
    [DVM v1 [v2]] [TVM v1 [v2]] [VAPPI v] [T0VMI v]
```

输入

n 设置速度模型号，取值为 1 或 2
ON|OFF 打开/关闭速度模型选项。若打开则使用速度模型，否则忽略
REFRACTEDWAVE 打开速度模型选项，并修改为折射波模型
NORMALMOVEOUT 打开速度模型选项，并修改为Normal moveout 模型
FLIP 交换两个速度模型的属性
VAPP v 设置视速度为 v
VAPP CALCULATE OBSPY 自动计算视速度
TOVM v 设置时间轴截距为 v
TOVM CALCULATE OBSPY 自动计算截距
DVM v1 v2 定义一/二个参考距离
TVM v1 v2 定义一/二个参考时间
VAPPI v 设置视速度增量为 v。每次 incrementstack 命令执行时视速度增加 v
TOVMI v 设置时间轴截距的增量为 v。每次 incrementstack 命令执行时视速度增加 v

缺省值

```
velocitymodel 1 off
velocitymodel 2 off
```

说明

第一个速度模型用于计算某个特定震相的动态台站延迟。在信号迭加 (sumstack)、绘图叠加图 (plotstack)、绘制剖面图 (plotrecordsection) 时会使用该模型。第二个速度模型用于在绘图剖面图时显示相对于第二震相的延迟。这两个模型的参数可以很容易的进行交换。

可以使用两种速度模型，即折射波速度模型：

$$T_{delay} = TVM(1) - \frac{TOVM + DIST}{VAPP}$$

以及 normal moveout 速度模型：

$$T_{delay} = TVM(1) - \sqrt{TOVM^2 + (\frac{DIST}{VAPP})^2}$$

这些速度模型延迟可以通过多种方式得到：

- 直接输入 VAPP、TOVM、TVM(1)
- 输入 DVM(1)、TVM(1) 以及 VAPP 或 TOVM，OBSPY 自动计算所需的变量以保证在距离 DVM(1) 处时间延迟为零
- 输入 DVM(1)、TVM(1)、DVM(2) 和 TVM(2)。OBSPY 将计算 VAPP 和 TOVM，以保证在距离 DVM(1) 处的时间延迟为零

示例

设置第一个迭加速度模型为折射波模型，视速度为 6.5 km/s，让 OBSPY 自动计算 TOVM 以使得 200 km 处的时间延迟为零：

```
velocitymodel 1 refractedwave vapp 6.5 tovm calculate dvm 200 tvm 35
```

13.18 velocityroset

概要

控制剖面图中速度 roset 的放置

语法

```
VELOCITYROSET [ON|OFF] [LOCATION UL|UR|LL|LR]
```

输入

ON|OFF 打开/关闭速度 roset 绘制选项

LOCATION UL|UR|LL|LR 修改速度 roset 的放置位置。分别对应左上、右上、左下、右下角。

缺省值

```
velocityroset off location ll
```

13.19 writestack

概要

将迭加结果写入磁盘

语法

```
WRITESTACK [filename]
```

输入

filename 要写入的磁盘文件名

缺省值

```
writestack sum
```

13.20　zerostack

概要

初始化信号迭加子程序

语法

```
ZEROSTACK
```

说明

该命令会删除迭加文件列表中的全部项，并将全局迭加选项设置回默认值。

保护环境，从阅读电子文档开始！

第 14 章　SPE

14.1　谱估计子程序

14.1.1　简介

SPE，全称为 Spectrum Estimation Subprocess，在 OBSPY 中键入 `spe` 命令即可进入谱估计子 程序。该子程序主要用于处理稳态随机过程，包含了如下三种谱估计方法：

PDS 能量密度谱
MLM 最大似然方法
MEM 最大熵方法

这三种方法都是间接法，因为它们都用了采样相关函数而不是数据本身来估计谱内容。

14.1.2　SPE 命令

SPE 子程序中包含了一些专门的命令，同时也可以使用 OBSPY 的部分命令。这里只列出 SPE 专 属的命令。

- `cor` 计算互相关函数
- `mem` 用最大熵方法计算谱估计
- `mlm` 用最大似然法计算谱估计
- `pds` 用能量密度谱方法计算谱估计
- `plotcor` 绘制相关函数
- `plotpe` 绘制 RMS 预测误差函数
- `plotspe` 绘制谱估计
- `readcor` 读取相关函数
- `writecor` 将相关函数以 OBSPY 文件格式写入磁盘
- `writespe` 将谱估计以 OBSPY 文件格式写入磁盘

14.1.3　理论

SPE 主要用于分析稳态随机过程。它实现了三种不同的间接的谱估计方法。它们之所以称为是间接的是由于它们不直接从数据出发去做谱估计，而是从由数据求出的样本相关函数出发去做频谱 估计。选择间接方法完全是一种偏爱，因为直接的频谱估计技术也是可以用的。相关函数本身也是 一个有用的函数，在进行频谱估计的过程中你会了解这一点。SPE 的谱估计类型为频率域中的功率 密度谱，其频谱被定义在一定的频率范围内，于是在一些频带中随机过程的功率即为这个频带的功 率谱密度的积分。

14.1.4　用户控制

SPE 可以使用户控制频谱估计过程中的一些细节。对那些有频谱估计的经验的人来说，这是很理想的。对那些不想过细地研究有关理论的用户也提供了便于使用的缺省值。在测定相关函数时用

户可以对数据窗口的类型、尺寸和使用的窗口数进行选择。一般地讲，这些参数控制了谱分析的分辨率，以及最后的谱估计中的方差。另外，数据的预白化可以指定为测定相关函数的过程的一部分， 预白化对减缓严重的窗口“混淆”现象是很有用的，“窗口混淆”有可能发生在具有大动态范围的谱 估计过程中。发生在预白化时的频谱失真在最后的结果中进行补偿。在这个过程中，数据的预白化 使用了低阶的预测误差滤波器。

14.1.5 算法

用户可以有三种谱估计算法的选择：功率密度谱、最大似然法和最大熵法。

PDS 法相当简单，样本相关函数乘以相关窗，然后对结果进行 FFT 以获得频谱估计结果。用户还可以对窗的类型和尺寸进行选择。

MLM 法生成一个频谱，这种频谱是一个经过平滑处理的功率密度谱的参量估计。用户可以选择参量的个数。

MEM 估计是另一个参量方法，它使用一个预测误差滤波器对数据进行预白化。这个谱估计的结果反比于滤波器的功率频率响应。用户可以选择预测误差滤波器的阶数。

14.1.6 诊断

除了频谱之外，一些诊断函数也可以计算并标绘出来。预测误差可以被标绘为阶的函数。这个 图可用来为应用于 MEM 方法的预测误差滤波器选择一个较好的尺寸。由于进行 PDS 估计的算法已经众所周知，所以在 SPE 中给出了关于这种方法的更多的诊断信息。90% 置信区间以及估计的频率分辨率可以通过理论进行估算。这些值都可以在 PDS 的频谱上显示出来。

14.1.7 同主程序的区别

在 SPE 和 OBSPY 主程序之间有两个主要的区别。SPE 一次只能处理一个数据文件，这是因为 SPE 在运行期间生成并保存了大量的辅助函数 (例如：相关函数、预测误差函数以及谱估计函数自 身)。这种对单个数据文件的限制将在未来的版本中去掉。第二个不同点是，与 OBSPY 不同，SPE 中 具有自己特有的执行不同指令的次序。

14.1.8 初始化

执行 SPE 命令时即调用了 SPE 软件包。调用的同时也定义了各种 SPE 参数的缺省值。数据文件在进入 SPE 之前或进入 SPE 的任何时间均可以使用 READ 命令读入，一旦读入新的文件，系统中将为前面所述的辅助函数生成一个空间。

14.1.9 相关

可以使用 `cor` 命令计算相关函数，用 `writecor` 命令可以激昂相关函数作为 OBSPY 的数据文件 保存起来，还可以用 `readcor` 命令再将它们读回 SPE 中去，这比每次都重复计算相关函数要更为 简单。在数据文件很长的时候尤为如此。此时用户也可以使用 `plotcor` 命令来看一下相关函数。如 果用户准备使用MEM方法的话，还可以使用`plotpe`命令来看一下预测误差函数。

14.1.10 估计

用户可以使用 `pds`、`mlm`、`mem` 命令来选择三种频谱估计中的任何一种。每一种方法都有自己的选项，你可以使用 `plotspe` 命令来检验谱分析结果。有几种确定比例的选项可以使用。同样的你也可以使用`writespe`命令将谱估计的结果作为OBSPY的数据文件保存起来。

14.1.11 终止

可以使用 quitsub 命令终止谱估计子程序，或使用 `quit` 命令终止整个 OBSPY 程序的运行。

14.2 cor

概要

计算相关函数

语法

```
COR [NUMBER n|ON|OFF] [LENGTH v] [PREWHITEN ON|OFF|n]
    [STOCASTIC|TRANSIENT]
    [TYPE   HAMMING|HANNING|COSINE|RECTANGLE|TRIANGLE]
```

输入

NUMBER n 设定窗口数为 n
NUMBER ON 设定窗口数为先前值
NUMBER OFF 根据数据长度和窗长计算窗口数。使用该选项时没有数据重叠
LENGTH v 设置窗长为 v 秒
TYPE type 设置窗类型
PREWHITEN ON|OFF 打开/关闭预白化选项
PREWHITEN n 打开预白化选项，并设置系数的个数为 n
STOCHASTIC 设置相关定标，假定数据是随机的
TRANSIENT 设置相关定标，假定数据是瞬态信号

缺省值

```
cor number off type hamming prewhiten off
```

说明

这个命令假定数据是稳态的。基于这一假定，数据被分段成许多窗口，并且对每一个窗口计算 一个相关函数。求这些函数的平均值即生成基于随机过程的相关函数的更加稳定的估计。窗口数和 窗口长度以及窗口类型都由用户控制。如果窗口长度诚意窗口数目超过了数据总长度则窗口会发生 重叠，重叠的部分不在用户的控制之下。

很明显，对于一个固定的数据长度，在窗口数和窗口尺寸之间存在一个折衷。这种折衷最终决 定了使用相关函数做出的谱估计的混淆和方差之间的折衷。一个频谱估计算法的频率域分辨率依赖 于可以使用的相关的长度，因而也就间接地依赖于数据窗口的大小。相关窗口越大，由频率平滑产 生的频谱估计的混淆就越小。然而随着数据窗口尺寸加大，在平均计算只能怪可以使用的窗口的数 量会减少，其结果是，相关函数估计的方差会增加，从而频谱估计的方差也会随之增加。

适当地选择窗口类型可以用来协调混淆和方差之间的折衷。较平滑的窗口使窗口边缘附近的数 据平滑地过渡到零，从而实际上相当于减少了窗口的长度。于是窗口可以更多的重叠，也就是说可 以有更多的窗口，折衷选择是在以增加混淆的代价减小方差。

在频谱的动态范围相当大的时候混淆还有另外一个来源，即窗口泄漏效应。在使用 PDS 估计的时候这一点尤为明显。经由相关窗口的 FFT 的旁瓣功率给求出的频谱带来了一个台阶。在典型的地震数据中，这种台阶是规则的，而且出现在高频段上，在这个频段上的频谱通常较小。相关函数 估计有一个可选的预白化功能，它可以减缓旁瓣泄露问题。一个低阶的预测误差滤波器可以用来在 计算相关函数之前平滑数据的频谱，滤波器的效果将在频谱的计算过程中得到补偿。

数据进行预白化改变了原始信号，如果用户使用了预白化，退出子程序并且想要在别的操作中 再次使用原始的信号，则必须重新读入原始信号到 OBSPY。

这种相关函数用于频谱的计算，cor 必须在执行 pds、mlm、mem 之前执行，用户可以执行 plotcor 命令绘制相关函数并且可以使用 writecor 命令将其作为 OBSPY 文件进行保存。 **头段**

变量改变

depmin、depmax、depmin

14.3 mem

概要

利用最大熵方法计算谱估计

语法

```
MEM [ORDER n] [NUMBER n]
```

输入

ORDER n 设置预测误差滤波器的时滞阶数为 n
NUMBER n 设置用于谱估计的点数

缺省值

```
mem order 25
```

说明

该命令实现了最大熵谱估计法，该方法使用一个预测误差滤波器对数据进行白化处理，得到的 谱估计正比于滤波器的能量频率响应的倒数。用户可以自由选择预测误差滤波器的阶数，详情参考 plotpe 命令。

该方法的主要优点是用相对少量的数据即可获得相当高的分辨率，它的缺点是跟传统方法相比 没什么理论好说。

14.4 mlm

概要

利用最大似然方法计算谱估计

语法

```
MLM [ORDER n] [NUMBER n]
```

输入

ORDER n 设置预测误差滤波器的时滞阶数为 n
NUMBER n 设置用于谱估计的点数

缺省值

```
mlm order 25
```

说明

该命令实现了能量密度谱的最大似然法。该方法生成的谱估计能够代表一个窄带通滤波器的能量输出，最终得到一个平滑的、参数化的能量密度谱，这些参数是有限脉冲响应窄带滤波器的系数。 用户可以指定这些参数。

该方法的特点在于其一般比传统方法有更高的分辨率。算法的阶数限制在 100，因为它需要一 个维度与阶数相同的矩阵的逆。对于这个求逆运算，存在更快的方法，但对于大阶数估计可能存在 数值噪声。

14.5 pds

概要

用能量密度谱方法计算谱估计

语法

```
PDS [SECONDS v|LAGS n] [NUMBER n]
    [TYPE   HAMMING|HANNING|COSINE|RECTANGLE|TRIANGLE]
```

输入

SECONDS v 设置窗长为 v 秒
LAGS n 设置窗长为 n?
NUMBER n 设置谱估计使用的数据点数
TYPE type 设置要使用的窗类型

缺省值

```
pds type hamming
```

说明

该命令实现了传统的谱估计方法。样本相关函数首先进行相关窗截窗，生成的函数再使用 FFT 获得谱估计。正如在 cor 命令文档中提到的，在估计偏差 (即丧失分辨率) 与估计方差之间存在 tradeoff。随着窗长增加，频率域分辨率增加，进而偏差减少；然而，样本相关函数在大延迟时值较 大，谱估计的方差也会增加，这是由于在大延迟时用于估计的数据点数变少了。

相关窗类型的选取与 cor 文档中描述的数据窗有不同的效果。这是两种不同类型的偏差之间的选择。

该谱估计方法用相关窗的 Fourier 变换来逼近真实谱的卷积。窗变换由两个特性控制，一个是控制分辨率的中心瓣，一个是控制带外能量泄露的旁瓣。通常来说，用户想要尽可能窄的主瓣以及 尽可能小的旁瓣。

大的旁瓣会在谱估计上产生一个虚假的低值，可能会掩盖高动态范围的频谱衰减。窗类型的选 取在主瓣分辨率与旁瓣能量泄露之间存在 tradeoff。

矩形窗有最窄的主瓣，因此也具有最好的分辨率，同时有最大的旁瓣。余弦窗稍微减小了旁瓣， 但基本不太影响主瓣宽度。这两种窗主要用于估计瞬变信号的谱，以获取尽可能小的时间域失真。 Hamming 和 Hanning 窗则具有很小的旁瓣和较宽的主瓣，当用户有大量数据的时候比较有用，可以通过增加窗的尺寸来控制分辨率。Hamming 和 Hanning 窗都是余弦窗的衍生，但 Hamming 窗

经过优化以尽可能地减少最大旁瓣的尺寸，因而该命令默认使用 Hamming 窗。三角窗也有很好的旁瓣结构，但是它有特别好的性质，即保证了谱估计总是非负值。

通常情况下，当用户需要处理大量数据时，PDS 方法要优于另外两种参数化方法。因为在这种情况下，分辨率不受限制，并且对这种算法的了解要比对其他方法更多一些。例如，理论上可以给 出该方法的置信度和分辨率。这些指标包括在 SPE 中。参量方法通常显示出比 PDS 更好的分辨率 (尤其是在估计线性谱的时候)，并且在数据量有限时更加有用。

14.6 plotcor

概要

绘制相关函数

语法

```
PLOTCOR [XLIM v|ON|OFF]
```

输入

XLIM v 打开 X 轴范围选项，并设置上限为 v 秒，下限一直为 0
XLIM ON 打开 X 轴范围选项，并使用原先设置的上限
XLIM OFF 关闭 X 轴范围选项，即绘制全部相关函数

缺省值

```
plotcor xlim off
```

14.7 plotpe

概要

绘制 RMS 预测误差函数

语法

```
PLOTPE
```

说明

该命令生成一个诊断绘图，用来选择 MEM 谱估计的阶数。该绘图是归一化的预测误差函数，表示成算法的阶的函数。一般说来，预测误差对于低阶数来说是很大的。随着阶数的增加误差会迅速下降。预测误差是预测滤波器对数据滤波之后的残余功率。理论表明，这个数量很小时，频谱中的主要部分都被滤波器的功率响应所捕获。残余数据为白噪声，结果是用户可以检测预测误差函数以寻找曲线中的 “膝部”。在该位置处，函数会急剧下跌到某些值，在这些点上即使进一步增加阶数，函数也不会随之进一步下降。在 “膝部” 的预测算法的阶数经常被用来作为 MEM 频谱估计的阶数。

14.8 plotspe

概要

绘制谱估计

语法

```
PLOTSPE [POWER|LOG|AMPLITUDE] [CONFIDENCE [ON|OFF]]
```

输入

POWER 用线性插值绘制功率响应
LOG 用对数内插法绘制功率响应
AMPLITUDE 绘制振幅响应
CONFIDENCE ON 在绘图上包含置信区间
CONFIDENCE OFF 在绘图上不包含置信区间

缺省值

```
plotsp power confidence off
```

说明

这个绘图中还包括用于计算相关函数和频谱估计的参数的描述。

14.9 readcor

概要

读取 OBSPY 文件到内存中

语法

```
READCOR file
```

输入

file 合法的文件名

说明

该命令与 OBSPY 主程序中的 `read` 命令相同，存在如下两个例外。

首先，在 SPE 中只能读取一个文件；其次，执行该命令会删除内存中已经计算的任何相关函数或谱估计。当该命令执行时，SPE 中的参数，如预白化系数的数目或窗类型及长度，不会被改变。

为了重新初始化所有 SPE 参数，你需要使用 quitsub 命令终止 SPE 子程序，再重新启动。

14.10 writecor

概要

写一个包含相关函数的 OBSPY 文件

语法

```
WRITECOR [file]
```

输入

file 要写入的 OBSPY 文件名

缺省值

```
writecor cor
```

说明

由此命令写出的相关函数的结构取决于用来计算它的算法。由于数据被分配到几个窗口中，并且样本相关函数是从每一个窗口算出的然后再去平均，则相关函数的长度由数据窗口的尺寸决定， 它实际上容纳的样本数则为数据窗内数据的二倍减一。然而，由于计算样本相关函数时使用了 FFT 算法，文件中的样本数是 2 的乘方。也就是说实际上是数据窗口尺寸的 2 倍。附加的样本都是零。 相关函数也是在文件内循环回转的。这是由于使用 FFT 算法计算相关函数的特殊性。这意味着零 滞后样本是文件在中的第一个样本，负滞后的样本紧随着正滞后的样本。

14.11 writespe

概要

写一个包含谱估计的 OBSPY 文件

语法

```
WRITESPE [file]
```

输入

file 要写入的 OBSPY 文件名

缺省值

```
writespe spe
```

说明

频谱估计文件包含从 0 到截止频率的频谱。频谱估计由 FFT 算法计算得出，文件中的采样点数是 FFT 使用的长度的一半再加上 1，选择这种格式以使由 SPE 计算出的多个频谱能通过 plot2 绘制函数来进行比较而无需事先为绘图而剪裁文件。

第三部分

附录

附录 A 错误与警告

0131 Obspy/datagen data directory not found

datagen 在生成波形数据时需要从 ${OBSPYAUX}/datagen 目录下读取 OBSPY 文件，出现该错误的 原因是该目录不存在。

1028 External command does not exist

执行 load 命令时找不到外部命令。产生该错误的原因很多，可能是环境变量 LD_LIBRARY_PATH 不包含要载入的共享库的所在目录，或 $OBSPYSOLIST 中不包含要载入的共享库的名字。

1103 No help package is available

没有可用的帮助文档包，可能原因是没找到 ${OBSPYAUX}/help 目录。

1106 Not a valid OBSPY command.

对于每一行命令，OBSPY 首先会检查是否是 OBSPY 内部的命令，如果不是，则检查是否是系统自 带的命令，比如 ls、cp 等。

一个例外是系统命令 rm。在 OBSPY 中直接执行 rm 命令会出现如上所示的错误。出现该错误的 原因是 OBSPY 禁用了系统命令 rm，主要目的是为了防止将 r *.OBSPY 误敲成 rm *.OBSPY 而导致数据的 误删除。可以使用 systemcommand 命令显式调用系统命令，如下：

```
OBSPY> rm BAD*.OBSPY
 ERROR 1106: Not a valid OBSPY
command. OBSPY> sc rm BAD*.OBSPY
```

1301 No data files read in

内存中未读入数据。可能是未指定要读取的文件列表，或列表中的文件不可读。

1303 Overwrite flag is not on for file

该错误主要出现在写 OBSPY 文件时，出现该错误的原因是 OBSPY 文件的头段变量 lovrok 的值为
FALSE，即磁盘中的数据不允许被覆盖。解决该问题的方法有两种：

- 以其他文件名写入磁盘，不覆盖磁盘文件；
- 修改 lovrok 的值为 TRUE；

1304 Illegal operation on data file

对数据文件的非法操作。

1305 Illegal operation on time series file

某些命令无法对时间序列数据进行操作。

1306 Illegal operation on unevenly spaced file

某些命令无法对不等间隔数据进行操作。

1307 Illegal operation on spectral file

命令不能对内存中的谱文件进行操作。

1311 No list of filenames to write

没有要写入的文件列表。

1312 Bad number of files in write file list

通常在使用 `write` 命令时会出现该问题。出现该错误的原因是内存中的波形文件的数目与 write 命令给出的文件名的数目不想匹配。在该错误信息的后面，紧接着会给出 write 命令中给出的文件数目以及内存中的波形数目。

1317 The following file is not a OBSPY data file

试图读入非 OBSPY 格式的文件所产生的错误。

1320 Available memory too small to read file

系统内存不足。

1322 Undefined starting cut for file

`cut` 命令中时间窗的起始参考头段未定义。默认情况下，会使用磁盘文件的起始时间代替，也 可以使用 `cuterr` 命令控制该错误的处理方式。

1323 Undefined stop cut for file

`cut` 命令中时间窗的结束参考头段未定义。默认情况下，会使用磁盘文件的结束时间代替，也 可以使用 `cuterr` 命令控制该错误的处理方式。

1324 Start cut less than file begin for file

`cut` 命令中时间窗的开始时间早于磁盘文件的开始时间。默认情况下，会使用磁盘文件的开始时间代替，也可以使用 `cuterr` 命令控制该错误的处理方式。

1325 Stop cut greater than file end for file

`cut` 命令中时间窗的结束时间晚于磁盘文件的结束时间。默认情况下，会使用磁盘文件的结束时间代替，也可以使用 `cuterr` 命令控制该错误的处理方式。

1326 Start cut greater than file end for file

`cut` 命令中时间窗的开始时间晚于文件结束时间。

1340 data points outside allowed range contained in file

文件中数据点的值超过了所允许的范围。比如 `log` 中要求数据为正。

1379 No SORT parameters given

使用了 `sort` 命令，但未指定按照哪个参数排序。

1380 Too many SORT parameters

`sort` 命令中用于排序的参数太多。

1381 Not a valid SORT parameter

无效的 sort 参数。

1383 SORT failed

排序失败。

1606 Maximum allowable DFT is 16777216

OBSPY 中与 FFT 相关的命令，所能允许的最大数据点数是 $2^{24} = 16777216$。

1611 Corner frequency greater than Nyquist for file

对数据进行滤波时，拐角频率超过了文件的 Nyquist 采样率。

1613 Minimum size of data file for Hilbert transform is 201

在做 Hilbert 变换时，要求数据的最小长度是 201 个数据点。

1701 Can't divide by zero

除零的非法操作。

1702 Non-positive values found in file

数据文件中存在非正值。

1801 Header field mismatch

该错误出现在 addf、subf、divf、mulf 以及 merge 和 beam 中。 出现该错误的原因是多个数据文件中的头段变量不匹配。该命令会明确给出不匹配的头段变量名，以及出现不匹配的数据文件，以供用户查错。会出现不匹配的头段变量包括 npts、delta、kstnm、knetwk、kcmpnm。

1802 Time overlap

要进行操作的两个数据的时间段不完全重合。

1803 No binary data files read in.

addf、subf、merge 等命令需要先读入二进制数据，再对数据做操作。

1805 Time gap

使用 merge 命令时，两段数据间存在时间间断。

2001 Command requires an even number of data files

使用 rotate 命令进行数据对的旋转时，需要内存中有成对的数据文件（即偶数个），若内存中数目不为偶数个，则报错。注意检查是否有数据漏读。

2002 Following files are not an orthogonal pair

出现该错误的原因是使用 rotate 旋转的两个分量不完全正交，此时应注意检查两个分量的头段变量 cmpinc 和 cmpaz。若两个头段变量未定义，则需要根据仪器的其他信息确定两个头段变量的值；若两个头段变量有定义，但的确不正交，则无法进行分量旋转。

2003 Following files are not both horizontals

rotate 命令的 TO 选项只能用于将两个水平的分量旋转到某个角度，出现该错误时应注意检查两个分量的头段变量 cmpinc 是否等于 90 度。

2004 Insufficient header information for rotation

`rotate` 命令的 `TO GCP` 选项要求头段变量中 `stla`、`stlo`、`evla` 和 `evlo` 必须定义。该选项会读取一对分量中的第一个文件中的头段变量，并计算反方位角，而不是直接使用头段变量中已有的 反方位角值。

2008 Requested begin time is less than files begin time. Output truncated.

`interpolate` 命令中指定的开始时间小于文件起始时间，此时输出会被截断。

2111 Taper frequency limits are invalid. No taper applied.

该警告出现在 `transfer` 命令中，出现该错误的原因是 `freqlimits` 选项的参数设置有误。四个频率应满足 `f1<f2<f3<f4`。

出现该警告时，`transfer` 会忽略 `freqlimits` 选项，即在去仪器响应时，不使用 taper 函数，进而可能导致去仪器响应后的波形出现问题。

2405 Cannot PRINT: no SGF files produced

无法打印，原因是未生成 SGF 文件。

5003 No correlation function calculated

未计算相关函数。

5004 No spectral estimate calculated

未计算谱估计。

9005 Amplitude mismatch

要合并的两段数据之间存在时间重叠，且在重叠的时间段内振幅不匹配。出现这样的错误可能 是数据的绝对时间有问题。

Environmental variable OBSPYAUX not defined.

出现该错误的原因是环境变量 `$OBSPYAUX` 未定义，参考 “在 Linux 下安装 OBSPY”。

Enviornment variable OBSPYAUX too long: max: 128 OBSPYAUX: xxx

环境变量 `$OBSPYAUX` 超过 128 字符。

附录 B　仪器响应

B.1　仪器响应简介

地震仪观测到的地面运动记录可以表示为

$$u(t) = s(t) * g(t) * i(t)$$

其中 $s(t)$ 代表震源项，$g(t)$ 代表路径效应，$i(t)$ 代表仪器响应，星号代表卷积。翻译成中文就是“地面运动记录是震源项、路径项以及仪器响应三者的卷积”。

四个量中：

- $u(t)$ 是地震仪的数字记录，已知；
- $i(t)$ 在仪器设计的时候会给出各种参数，已知；
- $s(t)$ 为震源项，包括震源机制、震源时间函数等，未知；
- $g(t)$ 为路径项，即地球内部结构，未知。

其中的两个未知项，是地震学研究的两个主要内容：震源和结构。因而理解仪器响应 $i(t)$ 的基本原理并准确地去除仪器响应是研究的关键一步。

真实世界里，有很多因素会引起地面的运动，比如地震波、长周期潮汐波以及人类活动等等，因而真实的地面运动是极其复杂的。图 B.1 给出了 COLA 台站记录到的持续时间为 22 000 s 左右的地面运动情况，即 $s(t) * g(t)$ 所对应的波形：

从图 B.1 可以看到，整个地面运动中最明显的地方是弯弯曲曲类似正弦的波形，其周期大概在 6000 s，也就是两个小时，这很可能是潮汐波引起的长周期地面运动。波形的最大振幅大概是 2×10^7。在 1000 s 左右有一个明显的振动信号，这是一个震级为 Mw 8.3 级的大地震的信号，其振幅大概在 2×10^6 的量级，比长周期波小了一个量级。

这就是真实的地面运动。即便是如此大的地震，其引起的地面运动相对来说也是很小的。这样的记录因为有太多其它类型的地面运动的干扰，因而对于地震学来说是没有太大用处的，所以就需要设计合适的仪器尽量去除其它类型地面运动的干扰，也就是地震仪。

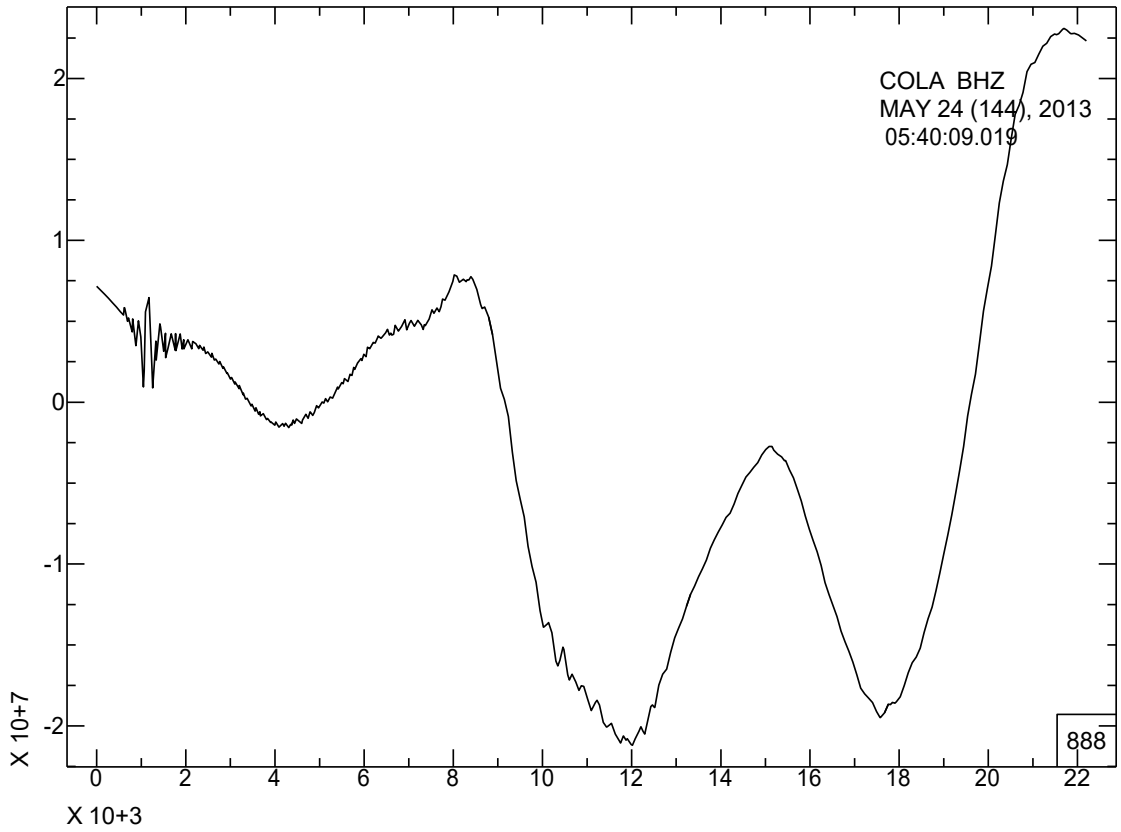

图 B.1: COLA 台站处的原始地面运动

从信号处理的角度来看，常见的地震仪是一个带通滤波器，对地震学不关心的超高频和超低频 的信号进行压制，只保留感兴趣的周期段。下图给出了该台站的仪器响应，即 $i(t)$：

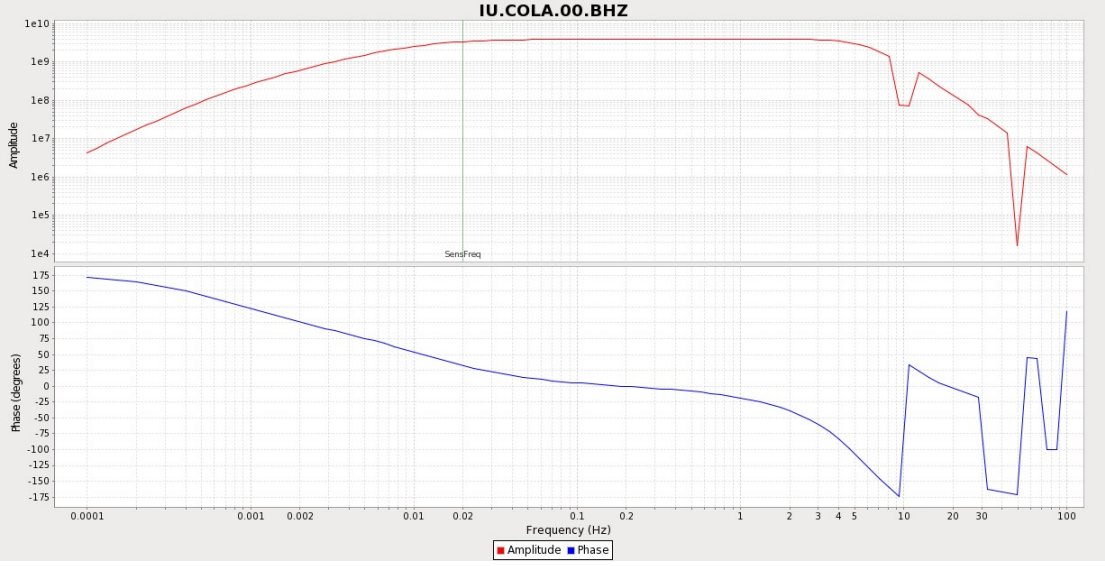

图 B.2: 仪器响应频谱图

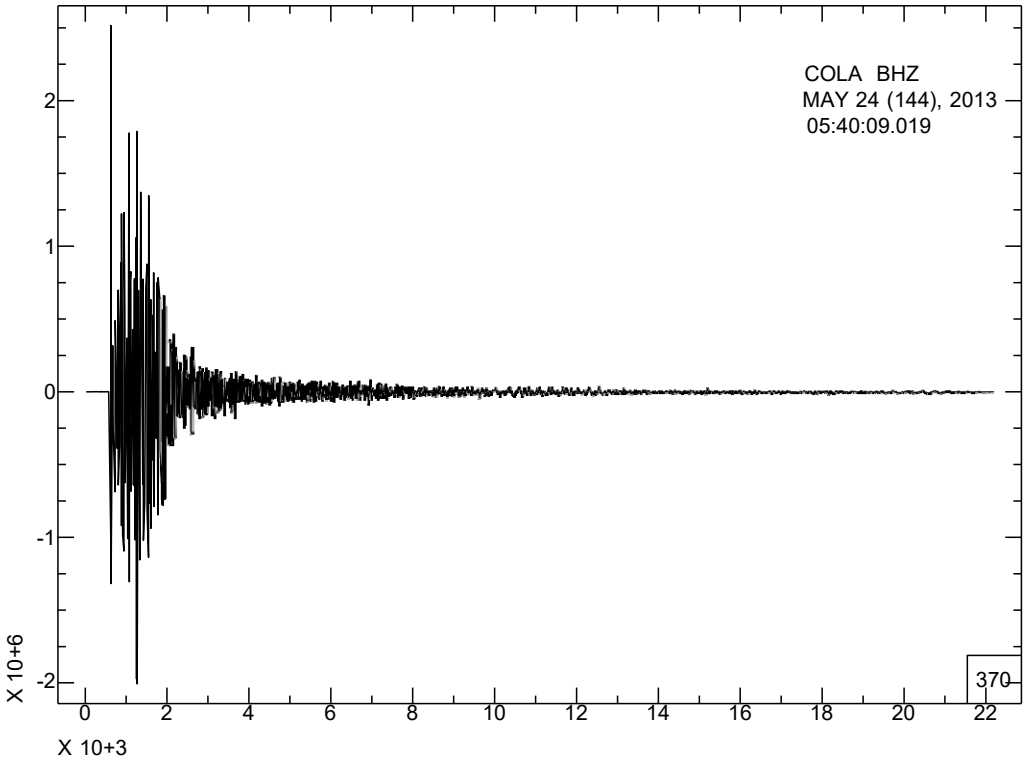

图 B.3: COLA 台站的地震记录

从图 B.2 中振幅谱可以看出，频率在 0.02 Hz 到 8 Hz 内的信号具有相同的振幅增益（被增强），而小于 0.02 Hz 或大于 8 Hz 的信号则被压制。图 B.1 中的周期为 1000 s 量级的信号被压制到了原来的千分之一。

原始的地面运动 $s(t) * g(t)$ 在经过仪器 $i(t)$ 之后，即得到地震仪的数字记录，如下图。超低频和超高频的信号被压制，留下地震学感兴趣的频段，也就是前面说的 $u(t)$：

与图 B.1 相比，长周期的类正弦信号没了。在 0 s 到 300 s 内，“地面” 很安静，300 s 左右，强烈的地震信号开始出现，最大振幅约为 2.4×10^6，持续了很长一段时间后，又恢复了 “平静”。这里可以很明显地看到 “平静 → 震动 → 平静” 的过程。这才是地震数据处理理想的波形。

为什么要去仪器响应呢？哪些时候需要去仪器响应呢？下面列举出若干需要去仪器响应的场景：

- 需要获取某个台站绝对振幅值；
- 仪器响应不同的台站之间的波形对比；
- 待补充...

B.2　物理与数学

地震仪一般固定在地表或地下浅层，也有放置在钻孔深处或海底的，因而地震仪可以直接感知 到地面的运动。在地面运动物理量被地震仪接收后，首先要将其转换为电信号，然后对电信号振幅 进行放大以及滤波，再将连续的时间序列离散化，最终以常见的波形数据的形式表现出来。

概括地说，地面的运动在被地震仪感知之后，需要经历如下三个阶段，最终成为用户拿到的波 形数据：

1. 模拟信号阶段
2. 模数转换
3. 数字信号阶段

地震仪的仪器响应可以表示为频率的复函数，即常说的振幅响应和频率响应。上面提到的三个 阶段，每个阶段的响应函数都是频率的复函数，表示为 $G_i(f)$，整个仪器的响应函数是各个阶段响应函数的乘积：

$$G(f) = \prod_i G_i(f)$$

B.2.1　预备知识

在介绍地震仪的三个阶段之前，需要先了解一些基础的数学知识。

响应函数

模拟信号的响应函数用 Laplace 变换表示：

$$H(s) = \int_0^{\infty} h(t)e^{-st}dt$$

数字信号的响应函数用 Z 变换表示：

$$H(z) = \sum_{-\infty}^{+\infty} h_m z^{-m}$$

归一化

频率响应可以表示为

$$G(f) = S_d R(f)$$

其中 $R(f)$ 是频率的函数。在某个特定的频率 f_s，有 $|R(f_s)| = 1.0$，即 $R(f)$ 在频率 f_s 处进行归一化。S_d 是放大系数，也称为 Sensitivity 或者 Gain。

R(f) 可以表示为

$$R(f) = A_0 H_p(s)$$

其中 $H_p(s)$ 是用零极点表示的 transfer 函数，一般来说在频率 f_s 处这个 transfer 函数的振幅响应不为 1，所以需要归一化因子 A_0。

FIR 滤波器

FIR 滤波器从数学上看就是对数据采样点做加权平均，设计不同的加权系数就得到不同的 FIR 滤波器。FIR 滤波器一般设计为振幅响应为方阶跃函数（boxcar），因而其在通带内有较平的振幅响应，在拐角频率处有很尖锐很陡峭的振幅响应变化。与此同时，滤波器具有线性相位，在时间域造成 信号的时间延迟，一般数据采集系统会对这个时间延迟做校正。因此，用户基本不需要考虑 FIR 滤波器对仪器响应的影响。

B.2.2　模拟信号阶段

模拟信号阶段会将连续的地面运动物理量（比如速度）转换为连续的电压信号，并进行电压放 大。因而此阶段的输入单位是运动物理量的单位（比如　　m/s），输出单位是伏特（V）。

这个阶段的响应函数可以表示为

$$G(f) = S_d A_0 \frac{\prod_{i=1}^{n}(s - r_i)}{\prod_{j=1}^{m}(s - p_j)} = S_d A_0 H_p(s)$$

其中，S_d 是放大系数，$H_p(s)$ 是用零极点表示的 transfer 函数，这里有 n 个零点和 m 个极点。A_0 是归一化因子，且 $s = i2\pi f$。

假定某仪器的自然频率为 $f_0 = 1$Hz, 阻尼常数为 $\lambda = 0.7$。如果该仪器是一个相对简单的仪器，则该仪器的加速度传递函数为：

$$H(s) = \frac{s}{s^2 + 2\lambda\omega_0 s + \omega_0^2}$$

几点说明：

- 该传递函数仅表示某一类地震仪的传递函数，现代地震仪的传递函数可能比这个要复杂；
- 根据该传递函数，很容易计算出它的一个零点和两个极点；

已知该仪器在 $f_s = 1\,\text{Hz}$ 时的增益为 $Sd = 150\,\text{V}/(\text{m/s}^2)$，即若仪器接收到 1Hz 频率的加速度 $1\,\text{m/s}^2$，则仪器的输出电压为 150V。因而仪器真实的传递函数还需要把增益加进去：

$$G(f) = R(f) * S_d = A_0 * H_p(s) * S_d$$

其中 A_0 是归一化因子，保证在频率 $f = f_s$ 处有 $|R(f_s)| = 1.0$。

B.2.3　模数转换

模数转换器，将上一阶段产生的连续电压信号转换为离散的电压信号。输入的单位是伏特　　V，输出单位是　　counts。这个阶段，所有频段有相同的振幅响应，即只存在一个放大系数，同时可能存在一个时间延迟。

假设模数转换器是 m 位，电压范围是 $\pm n\,\text{V}$，那么 $1\,\text{counts} = \frac{2n}{2^m}\text{V}$，电压也能精确到 $\frac{2n}{2^m}\text{V}$。例如，24 位模数转换器的输入电压范围是 $\pm 20\,\text{V}$，则输出的最大范围是 $\pm 2^{23}$ counts。因而 ADC 的放大系数为

$$S_d = 2^{24}/40\,\text{counts/V} = 4.1943 \times 10^5\,\text{counts/V}$$

即若 ADC 接收到的输入电压为 1V，则其输出为 4.1983×10^5 counts。结合地震仪的放大系数可知，频率为 1Hz 的 $1\,\text{m/s}^2$ 的地面运动加速度在 ADC 的输出为：

$$1\,\text{m/s}^2 * 150\,\text{V}/(\text{m/s}^2) * 4.1943 \times 10^5\,\text{counts/V} = 6.29145 \times 10^7\,\text{counts}$$

另一方面，由于 ADC 的输入电压的上限为 20V，因而仪器所能记录的最大加速度为 $20/150 = 0.13\ \text{m/s}^2$。

B.2.4　数字信号阶段

这个阶段会对数据信号进行进一步的处理，主要包含三个部分，即离散信号滤波、数据重采样、 时间延迟校正。

离散信号滤波可以采用 FIR 滤波器，也可以采用 IIR 滤波器。多数情况下采用 FIR 滤波器，而 FIR 滤波器的振幅响应函数可以认为在全频段内为 1[1]，因而这个阶段只需要考虑放大系数，而不需要再考虑由于滤波引入的响应函数。同样，对于数据重采样以及时间校正也不会引入新的响应函数。

B.2.5　小结

综上所述，三个阶段中，第一个阶段最为复杂，需要给出放大系数 Sd_1、归一化因子 A_0 以及零极点信息；第二个阶段以及第三个阶段都只需要给出放大系数 Sd_2 和 Sd_3。最终得到仪器的响应函数为

$$G(f) = Sd_1 A_0 H_p(s) Sd_2 Sd_3 = Sd_0 A_0 H_p(s)$$

即需要仪器在第一个阶段的零极点信息、归一化因子 A_0 以及三个阶段的放大系数的乘积 Sd_0 即可以近似表示地震仪的仪器响应。

[1] FIR 滤波器在 Nyquist 频率附近会有 5% 左右的震荡，因而若感兴趣的频率与 Nyquist 频率相差较大，则可以忽略这一阶段的响应函数

B.3　仪器响应文件

在 OBSPY 诞生的 80 年代，模拟地震仪的类型很少，OBSPY 将这些地震仪器的响应函数都内置到 程序中，可以直接使用。随着数字地震仪的出现和不断发展，地震仪器的类型越来越丰富，不可能把 这些地震仪器的响应函数都内置到 OBSPY 中，这就需要有更通用的方式来描述仪器响应，即仪器响 应文件：RESP、PZ 和 FAP。

B.3.1　内置仪器响应

OBSPY 内置了很多标准地震仪器的仪器响应，如表 B.1 所示。部分仪器类型还拥有子类型，如表 B.2 所示。在 OBSPY 命令中，可以直接使用这些仪器类型。

除了表 B.1 中列出的众多仪器类型之外，还有几个特别的仪器类型：

- `none`：即位移，也是OBSPY的默认值
- `vel`：速度
- `acc`：加速度

B.3.2　RESP 文件

RESP　　文件是用于描述仪器响应的文件，其包含了描述仪器响应所需要的全部信息。

RESP 仪器响应文件可以通过如下几种方式获得：

- 用 rdseed 程序从 SEED 数据中提取；
- 用 evalresp 程序从 SEED 数据中提取；
- 从 IRIS DMC resp Web Service[1] 下载；
- 手写 RESP 文件；

一个 RESP 文件中可以只包含一个仪器响应函数，也可以包含多个台站、多通道、多时间段的多个仪器响应函数。每个仪器响应函数中包含了台站名、台网名、通道名、开始时间和结束时间等台 站的基本信息。具体的仪器响应函数部分又分成多个 Stage，每个 Stage 中又分为多个 block，包含了仪器响应的不同信息。

- Stage1 一般对应模拟信号阶段，从中可以提取中这一阶段的输入单位、零极点、归一化因子 A_0 以及第一阶段的增益。
- Stage2 一般对应 ADC 阶段，从中可以提取出这一阶段的放大系数。
- Stage3 一般对应于数字滤波和减采样阶段。通常需要对数字信号多次滤波或减采样，因而 Stage3 后面可能会接多个类似的 Stage。从这几个 Stage 中提取的信息是增益，一般值为 1。
- Stage0 是会给出前面所有 Stage 的增益的乘积，主要是起到了辅助验证的作用。

B.3.3　OBSPY PZ 文件

RESP 文件中包含了仪器响应的完整信息，同时也包含了不少冗余信息。OBSPY 从 RESP 文件 中提取处仪器响应中的重要信息，定义了新的零极点响应文件（即 OBSPY PZ）。相对于 RESP 文件而 言，PZ　　文件中仅包含仪器响应中的零极点和增益信息，在去仪器响应时更方便。

OBSPY PZ 文件可以用 rdseed 程序从 SEED 文件中提取，也可以从 IRIS DMC OBSPY PZ Web Service [2] 获取，当然也可以手写 OBSPY PZ 文件。

下面是某个台站的 OBSPY PZ 文件：

```
*  **********************************
```

[1] http://service.iris.edu/irisws/resp/1/

[2] http://service.iris.edu/irisws/Obspypz/1/

表 B.1: OBSPY 内置仪器类型列表

type	说明
BBDISP	Blacknest specification of Broadband Displacement
BBVEL	Blacknest specification of Broadband Velocity
BENBOG	Blacknest specification of Benioff by Bogert
DSS	LLNL Digital Seismic System
DWWSSN	Digital World Wide Standard Seismograph Station
EKALP6	Blacknest specification of EKA LP6
EKASP2	Blacknest specification of EKA SP2
ELMAG	Electromagnetic
GBALP	Blacknest specification of GBA LP
GBASP	Blacknest specification of GBA SP
GENERAL	General seismometer
GSREF	USGS Refraction
HFSLPWB	Blacknest specification of HFS LPWB
IW	EYEOMG-spectral differentiation
LLL	LLL broadband analog seismometer
LLSN	LLSN L-4 seismometer
LNN	Livermore NTS Network instrument
LRSMLP	Blacknest specification of LRSM LP
LRSMSP	Blacknest specification of LRSM SP
NORESS	NORESS (NRSA)
NORESSHF	NORESS high frequency element
OLDBB	Old Blacknest specification of BB
OLDKIR	Old Blacknest specification of Kirnos
PORTABLE	Portable seismometer with PDR2
PTBLLP	Blacknest specification of PTBL LP
REDKIR	Blacknest specification of RED Kirnos
REFTEK	Reftek 97-01 portable instrument
RSTN	Regional Seismic Test Network
S750	S750 Seismometer
SANDIA	Sandia system 23 instrument
SANDIA3	Sandia new system with SL-210
SRO	Seismic Research Observatory
WA	Wood-Anderson
WABN	Blacknest specification of Wood-Anderson
WIECH	Wiechert seismometer
WWLPBN	Blacknest specification of WWSSN long period
WWSP	WWSSN short period
WWSPBN	Blacknest specification of WWSSN short period
YKALP	Blacknest specification of YKA long period
YKASP	Blacknest specification of YKA short period

表 B.2: 部分仪器子类型

主类型	子类型
LLL	LV, LR, LT, MV, MR, MT, EV, ER, ET, KV, KR, KT
LNN	BB|HF
NORESS	LP|IP|SP
RSTN	[CP|ON|NTR|NY|SD][KL|KM|KS|7S][Z|N|E]
SANDIA	[N|O][T|L|B|D|N|E][V|R|T]
SRO	BB|SP|LPDE
FREEPERIOD v	ELMAG, GENERAL, IW, LLL SUBTYPE BB, REFTEK
MAGNIFICATION n	ELMAG, GENERAL
NZEROS n	GENERAL, IW
DAMPING v	GENERAL, LLL SUBTYPE BB, REFTEK
CORNER v	LLL SUBTYPE BB, REFTEK
GAIN v	
HIGHPASS v	REFTEK

```
* NETWORK   (KNETWK): IU
* STATION    (KSTNM): COLA
* LOCATION   (KHOLE): 00
* CHANNEL   (KCMPNM): BHZ
* CREATED           :  2013-06-22T14:12:09
* START             :  2012-09-14T04:00:00
* END               :  2599-12-31T23:59:59
* DESCRIPTION       : College Outpost, Alaska, USA
* LATITUDE          : 64.873599
* LONGITUDE         : -147.861600
* ELEVATION         : 84.0
* DEPTH             : 116.0
* DIP               : 0.0
* AZIMUTH           : 0.0
* SAMPLE RATE       : 20.0
* INPUT UNIT        : M
* OUTPUT UNIT       : COUNTS
* INSTTYPE          : Geotech KS-54000 Borehole Seismometer
* INSTGAIN          : 2.013040e+03 (M/S)
* COMMENT           : N/A
* SENSITIVITY       : 3.377320e+09 (M/S)
* A0                : 8.627050e+04
*  **********************************
ZEROS   3
        +0.000000e+00   +0.000000e+00
        +0.000000e+00   +0.000000e+00
        +0.000000e+00   +0.000000e+00
POLES   5
```

```
        -5.943130e+01    +0.000000e+00
        -2.271210e+01    +2.710650e+01
        -2.271210e+01    -2.710650e+01
        -4.800400e-03    +0.000000e+00
        -7.384400e-02    +0.000000e+00
CONSTANT        +2.913631e+14
```

OBSPY PZ 文件中，以星号开始的行为注释行，给出了该 PZ 文件所对应的台站信息，其中 INPUT UNIT 表明了该 PZ 文件的输入是位移、速度还是加速度。用 rdseed 从 SEED 数据中提取出来的 PZ 文件，输入都是位移，且单位为 m。

以关键字 ZEROS 起始的行给出了零点数目，接下来几行列出了每个零点的实部和虚部。以关键字 POLES 起始的行给出了极点数目，接下来几行列出了每个极点的实部和虚部。最后一行给出了仪器响应中的常数 CONSTANT。

根据零极点以及 CONSTANT，即可计算得到仪器响应函数：

$$H(s) = C_0 * \frac{(s-z_1)(s-z_2)\ldots(s-z_{nz})}{(s-p_1)(s-p_2)\ldots(s-p_{nz})}$$

其中 $s = 2\pi if$。 一些说明：

- 若有零点 (0.0,0.0)，则这样的“零”零点可以省略。因而列出的零点数可能会少于“ZEROS”行给出的零点数；上例中的三个零点可以不列出；
- CONSTANT 对应于 RESP 文件中所有阶段的增益 Sd_0 以及归一化因子 A_0 的乘积；
- 若未指定 CONSTANT，则默认值为 1.0；

B.3.4 FAP 文件

FAP 文件是响应函数的另一种表现形式，其包含了很多记录行，每行三个字段，分别是频率（Hz）、振幅及相位。

频率不需要等间隔分段。在执行 transfer 时，低于第一行频率的频段将使用第一行的振幅和相位；同理大于最后一行频率的频段将使用最后一行的振幅和相位。

FAP 文件可以从程序 evalresp v3.3.2 中获得，FAP 相对于 PZ 文件的优势在于，其给出了每个频率的振幅和相位响应，因而包含更丰富的信息，且方便人工修改以控制需要校正的频率段。

B.3.5 RESP vs PZ vs FAP

RESP、PZ 和 FAP 都可以用于表征仪器的响应函数，常见的是 RESP 和 PZ，而这两种还是有很大区别的：

- RESP 文件包含了仪器响应的完整信息，而 PZ 文件中仅包含了零极点和增益信息，二者的主要差异在于 PZ 文件中未包含 FIR 滤波器的信息；
- RESP 文件中可以知道输入数据是位移、速度还是加速度，而 PZ 文件默认输入为位移。因而若 RESP 文件中输入是速度，则 PZ 文件中会多一个“零”零点；若 RESP 文件中输入是加速度，则 PZ 文件中会多两个“零”零点；
- OBSPY 中的默认位移单位是 nm，RESP 文件中有指定输入单位为 m，因而在用 RESP 去仪 器响应时，transfer 会在去除仪器响应之后在对数据做单位上的变换以使得得到的位移数据 的单位是 nm，即与 OBSPY 的标准相一致。而 PZ 文件中并未提供输入单位信息，或者说即便 提供了也没有被利用到，故而用 PZ 文件去除仪器响应得到的位移物理量单位是 m，为了与 OBSPY 标准相一致，需要对数据乘以 10 的 9 次方将数据单位由 m 转换成 nm；

对于大多数情况，建议使用 PZ 文件，数据处理速度要快很多。

附录 C　数据命名规则

用 rdseed 程序从标准 SEED 格式中解压得到的 OBSPY 文件，通常都具有固定格式的文件名。具体格式为：

```
yyyy.ddd.hh.mm.ss.ffff.NN.SSSSS.LL.CCC.Q.OBSPY
```

其中

- yyyy.ddd.hh.mm.ss.ffff 是 OBSPY 文件中第一个数据点对应的时间
  - yyyy 为年；
  - ddd 为一年的第多少天；
  - hh.mm.ss 为时、分、秒；
  - ffff 为毫秒；需要注意的是 1s = 1000 ms，这里毫秒用了 4 位来表示。
- NN 为台网名 [1]，长度不超过 2 个字符；
- SSSSS 为台站名，长度不超过 5 个字符；
- LL 为位置码，为空或两字符；
- CCC 为通道名；
- Q 为质量控制标识；
- OBSPY 为文件后缀；

C.1　位置码

位置码（Location ID）用于区分同一个台站处几套类似的仪器，这些仪器可能是相同的型号，但位于不同的深度或者指向不同的方位；也有可能是不同型号的仪器。

位置码通常用两位字母或数字表示，比如常见的 00、01、10 等。对于一个台站只有一套仪器的情况，位置码通常是空值。

C.2　质量控制

质量控制符 Q 用于表征当前 OBSPY 数据的数据质量。该标识符可以取如下四种：

- D 不确定状态的数据
- M 已合并的数据
- R 原始波形数据
- Q 经过质量控制的数据 常见

的质量控制符为 M 或 Q。

[1] 所有永久或临时台网的台网名列表：http://www.fdsn.org/networks/

C.3　通道名

通道名用三个字符来表示，这三个字符分别代表了频带码（Band　　Code）、仪器码（Instrument Code）和方位码（Orientation　Code）。

C.3.1　频带码

频带码是通道名的第一个字符，代表了仪器的采样率以及响应频带等信息。

表 C.1: 频带码

频带码	频带类型	采样率（Hz）	拐角周期（sec）
F	...	1000-5000	> 10
G	...	1000-5000	< 10
D	...	250-1000	< 10
C	...	250-1000	> 10
E	Extremely Short Period	80-250	< 10
S	Short Period	10-80	< 10
H	High Broad Band	80-250	< 10
B	Broad Band	10-80	> 10
M	Mid Period	1-10	> 10
L	Long Period	≈ 1	
V	Very Long Period	≈ 0.1	
U	Ultra Long Period	≈ 0.01	
R	Extremely Long Period	0.0001-0.001	
P	Order of 0.1 to 1 days	0.00001-0.0001	
T	Order of 1 to 10 days	0.000001-0.00001	
Q	Greater than 10 days	< 0.000001	
Q	Administrative Instrument Channel	variable	NA
O	Opaque Instrument Channel	variable	NA

C.3.2　仪器码

仪器码是通道名的第二个字符，代表了不同的仪器传感器。

表 C.2: 仪器码

仪器码	说明
H	High Gain Seismometer
L	Low Gain Seismometer
G	Gravimeter
M	Mass position Seismometer
N	Accelerometer

常见的是高增益（H）仪器，记录地面运动速度。

C.3.3　方位码

方位码表示了传感器记录的地面运动的方向。地震学中常见的方位码有如下几种：

表 C.3: 方位码

方位码	说明
N E Z	南北向、东西向、垂向
1 2 3	3 为垂向；1、2 为水平方向，正交但与正东西、正南北向有偏离
T R Z	T 为切向、R 为径向，通常 R 方向是震中到台站的大圆连线方向

通常情况下，若仪器的方向与正东西方向的夹角小于 2 度时，方位码取为 E；当与正东西方向夹角大于 2 度时，方位码取为 1 或 2。因而，方位码为 E 并不意味着分量是正东西方向的，真实的分量方向应以 OBSPY 头段中的 cmpaz 和 cpminc 为准。更进一步，由于仪器放置过程中的技术问题， OBSPY 头段中的 cmpaz 在某些情况下也会产生一定的误差。

保护环境，从阅读电子文档开始！

附录 D　数据获取

D.1　数据来源

地震波形数据的来源有很多，下面列举并简单介绍常见的数据来源。

D.1.1 国家测震台网数据备份中心

国家测震台网数据备份中心，隶属于中国地震局地球物理研究所。到　2014　年底，国家测震台网已建成由 170 个台站和 3 个小孔径台阵（共 30 个子台）组成的国家地震台网；859 个台站组成的 31 个区域地震台网，33 个子台组成的 6 个火山监测台网；291 套地震仪器组成的 32 个应急流动观测台网。国家测震台网数据备份中心可以提供从 2007 年 8 月起的全球 M5.5 级以上地震事件以及国内及周边地区 M3.5 级以上地震事件的波形数据。

要申请数据备份中心的数据，需要注册账户并升级账户属性。要升级账户属性，需要按程序进　行申请，并且签署和遵守 相关协议。

D.1.2　IRIS DMC

IRIS DMC 是世界上最大的地震波形数据中心。

IRIS DMC 的大部分数据是完全公开的，无需注册即可直接申请下载波形数据。从 IRIS DMC 申请数据的工具有很多：BREQ_FAST、Wilber III、Web Service、irisfetch.m、Web Service Fetch scripts、SOD 和 JWEED。

D.1.3　NIED

NIED 是日本的国家防灾科学技术研究所。其下包含若干台网：高感度地震观测网　Hi-net、宽频带地震台网 F-net、强地面运动地震台网 K-net 和 KiK-net 和火山观测网 V-net。

D.1.4　Natural Resources Canada

加拿大政府的网站 Natural Resources Canada 提供了 Canadian National Seismic Network、Yellowknife Seismic Array、POLARIS Network 等台网/台阵的连续波形数据以及这些台网/台阵 1975 年至今的事件波形数据。

事件波形数据可以直接 点击下载；连续波形数据则可以通过工具 AutoDRM 下载。

D.1.5　其他

- Northern California Earthquake Data Center
- Pacific Northwest Seismic Network
- Southern California Seismic Network
- Southern California Seismic Network at Caltech
- 中国地震科学探测台阵数据中心

D.2　IRIS 地震数据申请工具

从 IRIS 申请地震波形数据时，有很多工具可供选择与使用。这一节不会去介绍工具的具体用法，而是试着总结不同工具各自的优缺点以及适用范围。读者应根据实际需求选择合适的工具，并 自行阅读相应工具的说明文档。

D.2.1　Wilber III 主页：

http://ds.iris.edu/wilber3/find_event 教程：
http://seismman.info/wilber3.html

- 适用范围：仅用于申请基于事件的事件波形数据
- 特色及优点：
  - 网页端
  - 提供地震目录，筛选条件：经纬度范围、发震时刻、震级范围
  - 基于 Google 地图服务显示地震分布
  - 台站筛选条件：台网名、虚拟台网名、通道名、震中距范围、方位角范围
  - 基于 Google 地图服务显示台站分布
  - 提供波形预览
  - 根据发震时刻、P 波到时、S 波到时确定数据时间窗
  - 数据格式：OBSPY、SEED、miniSEED、ASCII、dataless SEED
  - 申请得到的数据位于 IRIS 的 FTP 中
  - 申请得到的数据已经写入事件信息
- 缺点：
  - 一次只能下载一个事件的波形数据，难以自动化
  - 只能用于基于事件的波形数据，无法用于基于台站的波形数据或连续波形数据

D.2.2　BREQ_FAST 主页：

https://ds.iris.edu/ds/nodes/dmc/manuals/breq_fast/ 教程：
http://seismman.info/iris-breq-fast.html

- 适用范围：连续波形数据
- 特色及优点：
  - 发送特定格式的邮件到特定邮箱即可申请数据
  - 邮件格式相对简单
  - 易于自动化和批量处理
  - 数据格式：发送到不同的邮箱可分别得到 SEED、miniSEED 和 dataless 格式的数据
  - 申请的数据位于 IRIS 的 FTP 里
- 缺点：需要一定的编程基础

D.2.3　DMC Web Service

主页：http://service.iris.edu/fdsnws/dataselect/1/

- 适用范围：连续波形数据
- 特色及优点：
  - 基于 Web 的服务，是其他多个工具的基础
  - 按照固定的格式构建一个 URL 贴到浏览器中即可申请相应的数据
  - 数据格式：miniSEED

- 缺点：Web 服务太原始，仅在个别情况下比较适用，大多数情况下都需要编程调用该服务

D.2.4　JWEED

主页：https://ds.iris.edu/ds/nodes/dmc/software/downloads/jweed/

- 适用范围：基于事件的事件波形数据 + 基于台站的事件波形数据
- 特色及优点：
  - Java 语法写的 GUI 客户端
  - 基于 DMC Web Service
  - 跨平台，但兼容性较差，某些平台下无法正常使用
  - 运行速度稍慢
  - 地震目录类型：NEIC　PDE、GCMT、ISC、ANF
  - 地震事件筛选条件：时间范围、震级范围、深度范围
  - 台站筛选条件：台网名、虚拟台网名、通道名
  - 地震台站对筛选条件：方位角范围、反方位角范围、震中距范围
  - 数据格式：OBSPY、RESP、PZ、ASCII、miniSEED
  - 可以生成 `BREQ_FAST` 格式的文件，供发邮件申请数据
  - 数据会直接下载保存到本地
- 缺点：无法下载连续数据

D.2.5　Web Service fetch scripts

主页：https://seiscode.iris.washington.edu/projects/ws-fetch-scripts

- 适用范围：连续波形数据
- 特色及优点：
  - Perl 语言写的脚本，在命令行使用
  - 基于 DMC Web Service
  - 数据格式：miniSEED
- 缺点：仅适用于连续波形数据。相对于原始的 Web Service 的优点在于，可编程实现批量申请

D.2.6　irisfetch.m

主页：https://ds.iris.edu/ds/nodes/dmc/software/downloads/irisfetch.m/

- 适用范围：连续波形数据
- 特色及优点：
  - Matlab 脚本，可以在 Matlab 中直接调用相关函数获取数据
  - 基于 DMC Web Service
  - 数据格式：保存为 Matlab 自定义结构体

D.2.7　SOD

主页：http://www.seis.sc.edu/sod/

- 适用范围：基于事件的事件波形数据 + 基于台站的事件波形数据
- 特色及优点：
  - 命令行工具，易于批量处理
  - 申请数据的同时可以对数据进行预处理
  - 数据格式：OBSPY、miniSEED
  - 可生成 `BREQ_FAST` 格式的文件
- 缺点：学习成本较高

D.2.8 SeismiQuery

主页：http://ds.iris.edu/SeismiQuery/breq_fast.phtml

- 适用范围：连续波形数据
- 特色及优点：
  - 网页工具
  - 可以生成BREQ_FAST所需的文件

附录 E　文档更新与维护

E.1　手册简介

《OBSPY 参考手册》（也就是你正在阅读的文档）是一本详细介绍 OBSPY 用法的中文电子书。

2011 年暑假，seisman 用 20 天左右的时间翻译了官方的英文帮助文档；2012 年初将翻译的文档整理成 PDF 并发布 1.0 版；2013 年 3 月用 LaTeX 重新对文档做了排版并发布 2.0 版；2014 年 4 月重新设计了整个文档的结构并重写了文档的大部分内容，并发布 3.0 版；目前，文档依旧在不断更新中。

该手册的源码使用 LaTeX，并开源托管在 GitHub 上：

- 项目主页：https://github.com/seisman/OBSPY_Docs_zh
- 文档发布页：http://seisman.info/Obspy_manual.html

E.2　文档更新历史

2012-01-08 1.0 版

- 第一版发布，由 DOC 转换为 PDF
- 参考《数字地震波形分析》一书，翻译了大部分官方文档中的内容
- 结合 OBSPY 101.4 版本，增加、删除和修改了一些命令
- 增加了书签，方便定位，支持全文搜索

2012-09-03 1.1 版

- 重新格式化整个文档，使得其看上去更规范，也易于以后的修改
- 代码从 NotePad++ 中直接导出，支持语法高亮
- 代码及正文英文字体采用 Consolas 字体
- 增加了“在脚本中调用 OBSPY”一节
- 新增命令 `transfer`、`traveltime`、`saveimg`、`datagen`
- 更新至 OBSPY v101.5c
- 公式用公式编辑器编辑

2012-09-18 1.2 版

- 增加了封面配图

2013-03-29 2.0 版

用 LaTeX 重新排版文档

- 操作系统：CentOS 6.4
- 编译环境：TeX Live 2012

- 编译命令：XeLaTeX
- 中文实现：CTEX 宏包
- 中文字体：宋体 + 黑体
- 英文主字体：Liberation Sans
- 代码字体：Courier 10 Pitch

2013-04-06 2.1 版

- 重新整理了第一章
- 修复 bugs

2013-04-12 2.2 版

- 重新排版了全部命令
- 重新设计了封面

2014-02-22 2.3 版

- 使用 git 管理源码
- 整理结构和布局的修改
- 新的小节：“OBSPY IO 升级版”、“黑板变量的读写”、“OBSPY 保存图像”
- 修复 bugs；

2014-04-18 3.0 版

- 源码托管在 GitHub 上，正式开源
- 丢弃了之前的提交历史，重新开始
- 重写了 LATEX 导言区
- 重新设计了整个文档的结构
- 重写了教程部分的大多数内容
- 教程部分根据 OBSPY v101.6a 进行修正
- 修复 bugs

2014-09-25 3.1 版

- 重新整理了大部分命令的语法说明
- 对“OBSPY 图像”一章进行了修订
- 新增“信号迭加子程序”一章
- 新增“谱估计子程序”一章
- 新增“在 Python 中调用 OBSPY”一节
- 修复 bugs

2015-05-02 3.2 版

对于用户：

- 修复 bugs 和 typos
- 命令整理：`systemcommand`、`transfer`
- 新增章节
  - 波形排序
  - 标记震相理论走时的三种方法
  - 图像格式转换
  - OBSPY 初始化宏文件

  - OBSPY 命令的长度上限
  - 字节序
  - 新增附录 “仪器响应”，整理了 “去仪器响应” 一节
- 新增示例：调用 OBSPY 的 Hilbert

函数 对于维护者：

- 新增 ChangeLog
- 更新 README，可根据说明自行编译源码生成 PDF
- 修改 Makefile，对依赖的检测更加智能
- 简化用于调用绘图脚本的 Makefile
- 英文使用 TeX 默认字体，中文使用开源中文字体 Fandol
- 使用 `minted` 实现代码的语法高亮，替代 `listings`
- `datetime` 宏包升级至 `datetime2`

2015-06-06 3.3 版

对于用户：

- 修改 bugs 和 typos
- 命令整理：`hilbert`、`transfer`
- 新增内容：
  - 四个文件重命名脚本
  - 读取某个目录下全部文件遇到的问题
  - 使用 Tab 遇到的问题
  - 数据命名规则
  - 时区校正
  - 错误与警告消息
  - 未定义变量
  - OBSPY debug
  - `wh` 与 `w over` 的区别

对于维护者：

- 删除之前的提交历史，精简 repo 尺寸，维护者需要重新 Fork
- 删除了 scons 脚本

2015-09-15 3.4 版

对于用户：

- 调整与修订：
  - 将命令的 “错误消息” 和 “警告消息” 集中整理到附录中
  - 将文件重命名脚本移动到 “在脚本中调用 OBSPY” 一章
  - 重新整理了 “震相拾取” 一节的内容
- 新增内容：
  - 在 Mac OS X 10.10 中安装 OBSPY
  - 在 C 程序中调用 OBSPY 提供的 `distaz` 函数
  - 数据处理中使用 `decimate` 和 `interpolate` 进行数据重采样
  - Python 中修改发震时刻
  - 在 C 程序中读写 OBSPY 文件
  - 在 Fortran 程序中读写 OBSPY 文件

  - 在 Python 脚本中读写 OBSPY 文件
  - 在 matlab 中读写 OBSPY 文件
  - 修改OBSPY所能读取的文件数目的上限
  - 文档维护与更新并征集维护者
- 命令整理：`mtw`、`markptp`、`markvalue`、`readcss`
- 修正 Bugs 和 Typos

对于维护者：

- 更新至 TEX Live 2015
- CTEX宏包版本要求 ≥ 2.2
- listings宏包版本要求 ≥ 1.6
- 新增“文档维护与更新”一章以介绍文档维护的流程
- 使用longtable宏包处理跨页长表格
- 使用 siunitx 宏包处理数字与单位之间的间隙
- 小括号统一使用英文输入法下的半角括号，而不是中文输入法下的全角符号
- Tips 环境改成 note 环境

2016-01-09 3.5 版

对于用户：

- 增加示例：绘制滤波器的时间响应和频率响应
- 增加示例：一次性修改多个波形数据的发震时刻
- 新增章节：`rdseed` 的选项及其用法
- 新增章节：介绍IRIS等地震数据中心
- 新增章节：介绍数据申请：连续波形数据和事件波形数据
- 新增章节：IRIS波形数据申请工具
- 新增章节：OBSPY 与脚本运行速度差异导致的陷阱
- 新增 Perl 脚本：数据提取、合并、重命名、修改发震时刻、去仪器响应、分量旋转、重采样
- 新增 Python 脚本：数据提取、合并、重命名、修改发震时刻、去仪器响应、分量旋转、重采样
- 更新命令说明：`plotpk`、`plot1`、`plot2`、`datagen`
- 新增命令：`writecss`
- 修正 Bugs 和 Typos

E.3 征集维护者

从 2012 年的 1.0 版到 2015 年的 3.3 版，本手册一直由 seisman 一人维护并持续更新。个人的精力有限，因而希望有志愿者参与到这个开源项目中，一起维护并完善该手册。

E.3.1 维护的内容

本手册需要维护的部分可以分为如下几类（按由简到繁、由易到难排序）：

1. 错别字；
2. 个别行或页面的排版问题；
3. 需要解释清楚或易引起歧义的部分；
4. 部分未翻译的命令，以及个别命令中未翻译的部分；
5. 某些命令在不同 OBSPY 版本间的语法不兼容；
6. 随着 OBSPY 新版本的发布，更新手册中相应的内容；

7. 补充命令示例以及脚本示例；
8. 补充其他尚未包含在手册中的与 OBSPY 相关的知识点；
9. 文档结构的调整；
10. 项目源码的优化；

E.3.2　参与方式

想要参与到本手册的维护，有三种方式（从易到难依次为）：

- 通过邮件发送给 seisman.info@gmail.com
- 在项目主页提交 Issue
- 在项目主页提交 Pull Request

方式一仅用于修正错别字等简单的维护，方式二和方式三则适用于全部的维护。其中，方式一对维护者没有其他要求；方式二要求维护者注册 GitHub 账户并懂得如何提交 Issue；方式三要求维护者注册 GitHub 账户，会使用版本控制工具 Git，会在 GitHub 上提交 Issue 和 Pull Request，也要求掌握 LaTeX 及编译的基础知识。关于方式三的具体如何操作，在后面的章节会专门介绍。

E.3.3　维护者的收益

维护者全凭志愿，没有任何物质报酬。除此之外，维护者将会：

1. 被加入到维护者列表中 [1]；
2. 结识更多的 OBSPY 用户以及地震学同行；
3. 对 OBSPY 有更深的理解；
4. 学会 Git，参与开源项目；
5. 了解 MarkDown 和 LaTeX；

E.3.4　对维护者的要求

所有 OBSPY 用户，都可以通过方式一参与到该项目的维护中，没有其他额外的需求。 如果想要对项目有更多的贡献，则需要满足如下要求：

1. 使用 Linux 或 Mac 操作系统；
2. 拥有 GitHub 账户；
3. 了解 Git 的基础知识；
4. 了解 LaTeX 的基础知识；

E.4　GitHub 和 Git

Git 是一个非常强大的版本管理工具，GitHub 则是一个基于 Git 的开源项目托管库。 本手册使用 Git 作为版本管理工具，并将源码托管在 GitHub 上。因而用户需要学会使用 GitHub 和 Git 才能更好地参与到项目的维护中来。

E.4.1　熟悉 GitHub

维护者需要首先注册一个 GitHub 账户，并向 GitHub 账户中添加当前机器的 SSH 秘钥，以使得当前机器拥有 GitHub 账户的读写权限。具体流程可以参考 http://www.worldhello.net/gotgithub/02-join-github/010-account-setup.html。

[1] 只有具有足够维护量的维护者才会被加入到列表中，最终解释权归 SeisMan 所有。

E.4.2　安装和配置 Git

Linux 下直接用系统自带的包管理器即可安装 `git`，同时也建议安装用于用于查看 git 提交信息的可视化工具 `gitk`。

CentOS/RHEL 用户：

```
$ sudo yum install git gitk
```

Ubuntu/Debian 用户：

```
$ sudo apt-get install git gitk
```

安装完 git 之后还需要告诉 git 你的名字和邮箱，这些信息会出现在每次的提交历史中：

```
$ git config --global user.name "Your Name"
$ git config --global user.email "you@example.com"
```

E.4.3　准备工作

在准备维护该项目之前，建议先阅读相关书籍以了解 git 的原理以及 GitHub 的使用说明。推荐的参考书目包括（由易到难排序）：

- git 简明指南
- GotGitHub
- 廖雪峰的 Git 教程
- Pro Git

首先，进入该手册的项目主页　`https://github.com/seisman/OBSPY_Docs_zh`，点击右上角的 Fork 按钮，将该项目复制到你的 GitHub 账户下。

下面假定你的 GitHub 账户名为 `USER`，在终端执行如下操作：

1. Clone 源码到当前机器

   ```
   $ git clone git@github.com:USER/OBSPY_Docs_zh.git
   $ cd OBSPY_Docs_zh/
   ```

2. 将 seisman 账户下的官方 repo 添加为远程 repo，并命名为 seisman，这样方便以后与官方 repo 同步进度

   ```
   $ git remote add seisman https://github.com/seisman/OBSPY_Docs_zh.git
   ```

E.4.4　正式维护该项目

至此，可以开始正式维护该项目了。使用 Git 进行协作的方式有很多中，这里参考了阮一峰的 Git 使用规范流程 一文中的协作方式。请按照如下流程参与到该项目的维护中：

1. 从 seisman 的官方 repo 中拉取源码的最新版本，并合并到本地 master 分支，以保证本地的 master 分支与官方 master 分支同步

   ```
   # 切换到本地的 master 分支
   $ git checkout master
   # pull = fetch + merge
   $ git pull seisman master
   ```

2. 不要在 master 分支中修改文档。对文档进行修改，应新建一个单独在单独的分支。分支名任意，但应尽量反映要维护的内容。这里假定分支名为　`mydev`

```
$ git checkout -b mydev
```

3. 对文档做修改，并提交 commit，此过程可以循环多次

```
$ git status
$ git add --all
$ git status
$ git commit -m " 此处填写本次提交的注释信息"
```

4. 分支开发的过程中，可能 seisman 的官方 master 分支已经更新，可以经常与主干保持同步

```
# 获取 seisman 的更新
$ git fetch seisman
# rebase 使得当前分支的提交基于最新的 seisman/master
$ git rebase seisman/master
```

5. 分支开发的过程中，可能会有很多次 commit，某些 commit 可能不那么重要，可以将多个 commit 压缩成一个或若干个 commit，这样不仅清晰，也容易管理

```
# 以 seisman/master 作为基准，对当前分支的 commit 做 rebase
$ git rebase -i seisman/master
```

rebase 操作相对比较复杂，可以参考前面提到的阮一峰的博文或者其他 git 相关书籍。

6. 将分支推送到远程仓库

```
# 将 mydev 分支推送到 USER 的 repo 下
$ git push -u origin mydev
```

7. 进入 https://github.com/USER/OBSPY_Docs_zh，点击 Pull Request 即可提交 PR 8. seisman 在收到 PR 后，会对代码进行审核、评论以及修改，并决定是否结束该 PR 9. 若 PR 已被接受，则可以删除本地和 GitHub 上的 mydev 分支

```
# 删除本地分支
$ git branch -D mydev
# 删除 GitHub 上的远程分支，也可以在 GitHub 上点击按钮删除分支
$ git push origin :mydev
```

E.4.5　对维护的若干说明

为了简化维护流程，避免重复或不必要的劳动，请遵循如下原则。 若只是对文档做简单的微调，比如修改简单的 bug 或 typo，整理部分语句等，可以直接修改并提交 PR。若需要对文档做大量修改，比如新增章节、调整文档结构等，请先到项目主页中提交 Issue，由众多维护者一起讨论是否有必要做修改。

若暂时不打算解决某 Issue，则该 Issue 会有标签 “Pull Request Welcomed”，维护者可以随意认领具有 “Pull Request Welcomed” 标签的 Issue，若某 Issue 已经被认领，则设置标签为 “In Progress”。标签的设置和修改只能由 seisman 完成。

E.5　编译文档

在维护文档的同时，你肯定想要将源码编译成 PDF 以检查自己的修改效果。这一节简单介绍如何编译文档。

该项目源码是用 LaTeX 写成的，Linux 下可以用 TeX Live 2015 编译。

E.5.1 安装 TEX Live 2015

TEX Live 是由国际 TEX 用户组（TEX Users Group，TUG）整理和发布的 TEX 软件发行套装，包含与 TEX 系统相关的各种程序、编辑与查看工具、常用宏包及文档、常用字体及多国语言支持。

参考 http://seisman.info/install-texlive-under-linux.html 安装 TeX Live 2015 并更新所有宏包至最新版本。

E.5.2 安装 pygment

目前，源码中的代码高亮使用 `minted` 宏包实现，该宏包依赖于 Python 的第三方模块 `pygments`。

CentOS/RHEL 下可以用如下命令安装：

```
$ sudo yum install python-pygments
```

Ubuntu/Debian 下可以用如下命令安装：

```
$ sudo apt-get install python-pygments
```

E.5.3 编译文档

进入项目所在目录，输入命令进行编译：

```
$ make
```

若编译时出现错误，请到项目主页处提交 Issue。