

MATLAB®

数据导入和导出



MATLAB®

R2022b



如何联系 MathWorks



最新动态: www.mathworks.com
销售和服务: www.mathworks.com/sales_and_services
用户社区: www.mathworks.com/matlabcentral
技术支持: www.mathworks.com/support/contact_us



电话: 010-59827000



迈斯沃克软件 (北京) 有限公司
北京市朝阳区望京东园四区 6 号楼
北望金辉大厦 16 层 1604

MATLAB® 数据导入和导出

© COPYRIGHT 2009–2022 by The MathWorks, Inc.

The software described in this document is furnished under a license agreement. The software may be used or copied only under the terms of the license agreement. No part of this manual may be photocopied or reproduced in any form without prior written consent from The MathWorks, Inc.

FEDERAL ACQUISITION: This provision applies to all acquisitions of the Program and Documentation by, for, or through the federal government of the United States. By accepting delivery of the Program or Documentation, the government hereby agrees that this software or documentation qualifies as commercial computer software or commercial computer software documentation as such terms are used or defined in FAR 12.212, DFARS Part 227.72, and DFARS 252.227-7014. Accordingly, the terms and conditions of this Agreement and only those rights specified in this Agreement, shall pertain to and govern the use, modification, reproduction, release, performance, display, and disclosure of the Program and Documentation by the federal government (or other entity acquiring for or through the federal government) and shall supersede any conflicting contractual terms or conditions. If this License fails to meet the government's needs or is inconsistent in any respect with federal procurement law, the government agrees to return the Program and Documentation, unused, to The MathWorks, Inc.

商标

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See www.mathworks.com/trademarks for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

专利

MathWorks products are protected by one or more U.S. patents. Please see www.mathworks.com/patents for more information.

修订历史记录

2009 年 9 月	仅限在线版本	MATLAB 7.9 (版本 2009b) 中的新增内容
2010 年 3 月	仅限在线版本	MATLAB 7.10 (版本 2010a) 中的修订内容
2010 年 9 月	仅限在线版本	MATLAB 7.11 (版本 2010b) 中的修订内容
2011 年 4 月	仅限在线版本	MATLAB 7.12 (版本 2011a) 中的修订内容
2011 年 9 月	仅限在线版本	MATLAB 7.13 (版本 2011b) 中的修订内容
2012 年 3 月	仅限在线版本	MATLAB 7.14 (版本 2012a) 中的修订内容
2012 年 9 月	仅限在线版本	MATLAB 8.0 (版本 2012b) 中的修订内容
2013 年 3 月	仅限在线版本	MATLAB 8.1 (版本 2013a) 中的修订内容
2013 年 9 月	仅限在线版本	MATLAB 8.2 (版本 2013b) 中的修订内容
2014 年 3 月	仅限在线版本	MATLAB 8.3 (版本 2014a) 中的修订内容
2014 年 10 月	仅限在线版本	MATLAB 8.4 (版本 2014b) 中的修订内容
2015 年 3 月	仅限在线版本	MATLAB 8.5 (版本 2015a) 中的修订内容
2015 年 9 月	仅限在线版本	MATLAB 8.6 (版本 2015b) 中的修订内容
2015 年 10 月	仅限在线版本	MATLAB 8.5.1 (版本 2015aSP1) 中的再发布内容
2016 年 3 月	仅限在线版本	MATLAB 9.0 (版本 2016a) 中的修订内容
2016 年 9 月	仅限在线版本	MATLAB 9.1 (版本 2016b) 中的修订内容
2017 年 3 月	仅限在线版本	MATLAB 9.2 (版本 2017a) 中的修订内容
2017 年 9 月	仅限在线版本	MATLAB 9.3 (版本 2017b) 中的修订内容
2018 年 3 月	仅限在线版本	MATLAB 9.4 (版本 2018a) 中的修订内容
2018 年 9 月	仅限在线版本	MATLAB 9.5 (版本 2018b) 中的修订内容
2019 年 3 月	仅限在线版本	MATLAB 9.6 (版本 2019a) 中的修订内容
2019 年 9 月	仅限在线版本	MATLAB 9.7 (版本 2019b) 中的修订内容
2020 年 3 月	仅限在线版本	MATLAB 9.8 (版本 2020a) 中的修订内容
2020 年 9 月	仅限在线版本	MATLAB 9.9 (版本 2020b) 中的修订内容
2021 年 3 月	仅限在线版本	MATLAB 9.10 (版本 2021a) 中的修订内容
2021 年 9 月	仅限在线版本	MATLAB 9.11 (版本 2021b) 中的修订内容
2022 年 3 月	仅限在线版本	MATLAB 9.12 (版本 2022a) 中的修订内容
2022 年 9 月	仅限在线版本	MATLAB 9.13 (版本 2022b) 中的修订内容

1

文件打开、加载和保存

支持的导入和导出的文件格式	1-2
以编程方式导入文件	1-2
专用数据格式的工作流	1-5
以交互方式导入图像、音频和视频	1-6
查看文件的内容	1-6
指定变量	1-6
生成可重用的 MATLAB 代码	1-7
导入或导出一系列文件	1-9
在 MAT 文件中保存和加载部分变量	1-10
使用 matfile 函数保存和加载	1-10
动态加载部分变量	1-11
避免意外加载整个变量	1-12
部分加载和保存功能要求 7.3 版本的 MAT 文件	1-12
MAT 文件版本	1-14
MAT 文件版本概述	1-14
保存为非默认 MAT 文件版本	1-15
数据压缩	1-15
加快 7.3 版本的 MAT 文件的保存和加载操作的速度	1-15
使用 matfile 函数增大数组	1-17
在函数中加载变量时出现意外结果	1-19
创建临时文件	1-21

文本文件

2

导入文本文件	2-2
以表的形式导入数据	2-2
将数据以时间表形式导入	2-2
以矩阵的形式导入数据	2-3
将数据以元胞数组的形式导入	2-3
将数据作为字符串数组导入	2-4
使用导入选项导入数据以进行更多控制	2-4
以交互方式导入数据	2-4

使用导入工具读取文本文件数据	2-6
以交互方式选择数据	2-6
从多个文本文件中导入数据	2-8
从文本文件中导入日期时间	2-10
将文本文件中的数值数据导入矩阵	2-14
导入逗号分隔的数据	2-14
导入分隔的数值数据	2-14
将文本文件中的混合数据导入表	2-16
将文本文件中的混合数据块导入表或元胞数组	2-19
将数据写入文本文件	2-22
将表导出到文本文件	2-22
将元胞数组导出到文本文件	2-23
将数值数组导出到文本文件	2-24
写入到 Diary 文件	2-25
从文本文件导入数值数据块	2-26

电子表格

3

导入电子表格	3-2
使用导入工具导入电子表格数据	3-2
使用 readtable 导入电子表格数据	3-2
将电子表格数据导入为其他数据类型	3-3
使用导入工具读取电子表格数据	3-4
以交互方式选择数据	3-4
从多个电子表格导入数据	3-5
粘贴剪贴板中的数据	3-5
将电子表格数据读入到表中	3-7
将数据写入 Excel 电子表格	3-10
将表格数据写入到电子表格文件	3-10
将数值和文本数据写入到电子表格文件	3-10
添加新工作表时禁用警告	3-11
对 Excel 文件中的单元格设置格式	3-11
定义表的导入选项	3-12

4

使用低级 I/O 导入文本数据文件	4-2
概述	4-2
读取格式化模式的数据	4-2
逐行读取数据	4-4
检测文件末尾 (EOF)	4-5
使用不同的字符编码打开文件	4-7
通过低级 I/O 导入二进制数据	4-8
用于导入数据的低级函数	4-8
读取文件中的二进制数据	4-8
读取文件的一部分	4-10
读取在其他系统上创建的文件	4-12
使用低级 I/O 导出到文本数据文件	4-13
使用 fprintf 写入到文本文件	4-13
追加到或覆盖现有的文本文件	4-14
使用不同的字符编码打开文件	4-16

5

6

导入图像	6-2
获取有关图像文件的信息	6-2
从 TIFF 文件中读取图像数据和元数据	6-2
导出到图像	6-5
将图像数据和元数据导出到 TIFF 文件	6-5

7

导入 NetCDF 文件和 OPeNDAP 数据	7-2
MATLAB NetCDF 功能	7-2
连接到 OPeNDAP 服务器时的安全注意事项	7-2
使用高级函数读取 NetCDF 文件	7-2
在 NetCDF 文件中查找所有无限维度	7-4
使用低级函数读取 NetCDF 文件	7-5

导出到 NetCDF 文件	7-8
MATLAB NetCDF 功能	7-8
基于现有文件或模板新建 NetCDF 文件	7-8
合并两个 NetCDF 文件	7-9
使用低级函数将数据写入 NetCDF 文件	7-11
以编程方式导入 HDF4 文件	7-13
概述	7-13
使用 MATLAB HDF4 高级函数	7-13

音频和视频

8

读取和写入音频文件	8-2
录制和播放音频	8-4
录制音频	8-4
播放音频	8-6
在函数内录制或播放音频	8-6
读取视频文件	8-8
支持的视频和音频文件格式	8-12
MATLAB 中的视频数据	8-12
MATLAB 中的音频数据	8-13
图像序列与视频之间的转换	8-15

XML 文档

9

将 XML 文件导入文档对象模型中	9-2
XML 文档对象模型	9-2
使用 MAXP 解析器读取 XML 文件	9-3
使用 xmlread 读取 XML 文件	9-4
将文档对象模型导出为 XML 文件	9-6
创建 DOM 文档	9-6
使用 MAXP DOMWriter 对象将 DOM 文档节点写入 XML 文件	9-6
使用 xmlwrite 将 DOM 文档节点写入 XML 文件	9-7
更新现有的 XML 文件	9-8

10

内存映射概述	10-2
什么是内存映射?	10-2
内存映射的优势	10-2
何时使用内存映射	10-3
内存映射的最大大小	10-4
字节排序	10-4

Internet 文件访问和 JSON

11

MATLAB 和 Web 服务安全	11-2
MATLAB 不校证书链	11-2
从 Web 服务下载数据	11-3
下载网页和文件	11-6
示例 - 使用 webread 函数	11-6
示例 - 使用 websave 函数	11-6
发送电子邮件	11-7

串行端口 I/O

12

配置串行端口通信设置	12-2
使用串行端口通信从 Arduino 读取流数据	12-3

大型数据

13

MapReduce 快速入门	13-2
什么是 MapReduce?	13-2
MapReduce 算法阶段	13-2
MapReduce 计算示例	13-3
编写 map 函数	13-7
map 函数在 MapReduce 中的角色	13-7
map 函数的要求	13-8
map 函数示例	13-8

数据存储快速入门	13-10
什么是数据存储?	13-10
创建和读取数据存储	13-11
处理远程数据	13-14
Amazon S3	13-14
Azure Blob 存储	13-15
Hadoop 分布式文件系统	13-16
读取和分析大型表格文本文件	13-19
读取和分析图像文件	13-21
使用 tall 数组处理无法放入内存的数据	13-25
什么是 tall 数组?	13-25
tall 数组的好处	13-25
创建 tall 表	13-25
创建 tall 时间表	13-26
创建 tall 数组	13-26
延迟计算	13-27
用 gather 计算	13-28
保存、加载和检查 tall 数组	13-29
支持函数	13-29
在 MATLAB 中使用 tall 数组分析大数据	13-30

MATLAB 中的 TCP/IP 支持

14

创建 TCP/IP 客户端并配置设置	14-2
使用主机名称创建对象	14-2
使用 IP 地址创建对象	14-2
设置超时属性	14-3
设置连接超时属性	14-3
设置传输延迟属性	14-4
查看 TCP/IP 对象属性	14-4
通过 TCP/IP 接口写入和读取数据	14-6
写入数据	14-6
读取数据	14-6
从气象站服务器获取数据	14-7
从网站读取网页	14-7

15	Bluetooth 低功耗通信
16	Bluetooth 通信
17	硬件管理器

文件打开、加载和保存

- “支持的导入和导出的文件格式” (第 1-2 页)
- “以交互方式导入图像、音频和视频” (第 1-6 页)
- “导入或导出系列文件” (第 1-9 页)
- “在 MAT 文件中保存和加载部分变量” (第 1-10 页)
- “MAT 文件版本” (第 1-14 页)
- “使用 matfile 函数增大数组” (第 1-17 页)
- “在函数中加载变量时出现意外结果” (第 1-19 页)
- “创建临时文件” (第 1-21 页)

支持的导入和导出的文件格式

本节内容
“以编程方式导入文件” （第 1-2 页）
“专用数据格式的工作流” （第 1-5 页）

将数据导入 MATLAB 的理想工作流取决于数据格式以及您的个人预设项。您可以通过编程方式或使用专用工作流来导入数据。最常见的解决办法是使用为数据定制的函数以编程方式导入数据。

当您将数据导入至 MATLAB 工作区时，您创建的新变量将会覆盖工作区中同名的任何现有变量。

以编程方式导入文件

MATLAB 包含针对导入特定文件格式定制的函数。当您要导入整个文件或仅导入文件的一部分时，请考虑使用格式特定的函数。许多格式特定的函数都提供了用于选择数据范围或数据部分的选项。一些格式特定的函数允许您请求多种可选的输出。

下表显示您可以在 MATLAB 应用程序导入和导出的文件格式。

文件内容	扩展名	描述	导入函数	导出函数
MATLAB 格式化数据	MAT	已保存的 MATLAB 工作区	load	save
		访问 MATLAB 工作区中的部分变量	matfile	matfile
文本	任何扩展名，包括： CSV TXT	分隔数字	readmatrix	writematrix
		带分隔符的数字，或者文本和数字的混合	textscan	无
		列向分隔数字或者文本和数字混合	readtable readcell readvars	writetable writecell
		纯文本	readlines	writelines
电子表格	XLS XLSX XLSM	工作表或电子表格范围中的列向数据	readmatrix	writematrix
	XLSB（仅限安装了 Windows® 版 Microsoft® Excel® 的系统） XLTM（仅导入） XLTX（仅导入） ODS（仅限安装了 Windows 版 Microsoft Excel 的系统）		readtable readcell readvars	writetable writecell

文件内容	扩展名	描述	导入函数	导出函数
可扩展标记语言	XML	XML 格式的文本	readstruct readtable readtimetable	writestruct writetable writetimetable
Parquet 格式数据	PARQUET	Parquet 格式的列向数据	parquetread	parquetwrite
Data Acquisition Toolbox™ 文件	DAQ	Data Acquisition Toolbox	daqread	无
科学数据	CDF	常用数据格式 (CDF)	请参阅 “常用数据格式 (CDF)”	请参阅 cdflib
	FITS	普适图像传输系统	请参阅 “FITS 文件”	请参阅 “FITS 文件”
	HDF	HDF4 或 HDF-EOS2	请参阅 “HDF4 文件”	请参阅 “HDF4 文件”
	H5	HDF5	请参阅 “HDF5 文件”	请参阅 “HDF5 文件”
	NC	网络通用数据格式 (netCDF)	请参阅 “NetCDF 文件”	请参阅 “NetCDF 文件”
图像数据	BMP	Windows 位图	imread	imwrite
	GIF	图形交换格式		
	HDF	分层数据格式		
	JPEG JPG	联合图像专家组		
	JP2 JPF JPX J2C J2K	JPEG 2000		
	PBM	可移植位图		
	PCX	画笔		
	PGM	可移植灰度图		
	PNG	可移植网络图形		
	PNM	可移植图		
	PPM	可移植像素图		
	RAS	Sun™ 光栅		
	TIFF TIF	标记图像文件格式		
	XWD	X 窗口转储		
	CUR	Windows 光标资源	imread	无
	ICO	Windows 图标资源		

文件内容	扩展名	描述	导入函数	导出函数
音频 (所有平台)	AU SND	NeXT/Sun 声音	audioread	audiowrite
	AIFF	音频交换文件格式		
	AIFC	带压缩编码的音频交换文件格式		
	FLAC	免费的无损压缩音频编码		
	OGG	Ogg Vorbis		
	OPUS	Ogg Opus		
	WAV	Microsoft WAVE 声音		
音频 (Windows)	M4A MP4	MPEG-4	audioread	audiowrite
	任意	Microsoft Media Foundation 支持的格式	audioread	无
音频 (Mac)	M4A MP4	MPEG-4	audioread	audiowrite
音频 (Linux®)	任意	GStreamer 支持的格式	audioread	无
视频 (所有平台)	AVI	音频视频交错格式	VideoReader	VideoWriter
	MJ2	Motion JPEG 2000		
视频 (Windows)	MPG	MPEG-1	VideoReader	无
	ASF ASX WMV	Windows Media®		
	任意	Microsoft DirectShow® 支持的格式		
视频 (Windows 7) 或更高版本	MP4 M4V	MPEG-4	VideoReader	VideoWriter
	MOV	QuickTime	VideoReader	无
	任意	Microsoft Media Foundation 支持的格式		
视频 (Mac)	MP4 M4V	MPEG-4	VideoReader	VideoWriter
	MPG	MPEG-1		
	MOV	QuickTime		
	任意	QuickTime 支持的格式, 包括 .3gp、.3g2 和 .dv		
视频 (Linux)	任意	安装的 GStreamer 插件支持的格式, 包括 .ogg	VideoReader	无
三角剖分	STL	立体光刻	stlread	stlwrite
低级文件	任何文本格式	低级二进制文本数据	fread	fwrite
	任意	低级二进制	fscanf	fprintf

文件内容	扩展名	描述	导入函数	导出函数
	任何文本格式	文本文件或字符串中的格式化数据	<code>textscan</code>	无

专用数据格式的工作流

二进制数据的内存映射

对于二进制数据文件，请考虑“内存映射概述”（第 10-2 页）。通过内存映射，可以使用标准 MATLAB 索引操作访问文件数据。内存映射是将磁盘上某文件的一部分或整个文件映射到应用程序地址空间内某个地址范围的一种机制。然后，应用程序可采用与访问动态内存相同的方法访问磁盘上的文件。内存映射的主要优势体现在效率、更快的文件访问速度、能够在应用程序之间共享内存，以及更高效的编码。

使用 MATLAB 工具箱进行专用导入

MATLAB 工具箱执行专用导入操作。例如，使用 Database Toolbox™ 软件从关系数据库导入数据。请参阅有关特定工具箱的文档以查看可用的导入功能。

用于读写数据的 Web 服务

可以使用 RESTful 或 WSDL 等 Web 服务来读取和写入 Internet 媒体类型格式的数据，例如 JSON、XML、图像或文本。有关详细信息，请参阅：

- “Web 服务”
- “将 WSDL 与 MATLAB 结合使用”

使用低级 IO 读取数据

如果格式特定的函数无法读取您的数据，并且专用工作流不符合您的要求，请使用低级 I/O 函数，如 `fscanf` 或 `fread`。低级函数支持最大程度地控制对文件的读取，但它们需要详细了解您的数据结构。此工作流不常用。

另请参阅

相关示例

- “标准文件格式”


以交互方式导入图像、音频和视频

以交互方式将数据导入到 MATLAB 工作区。

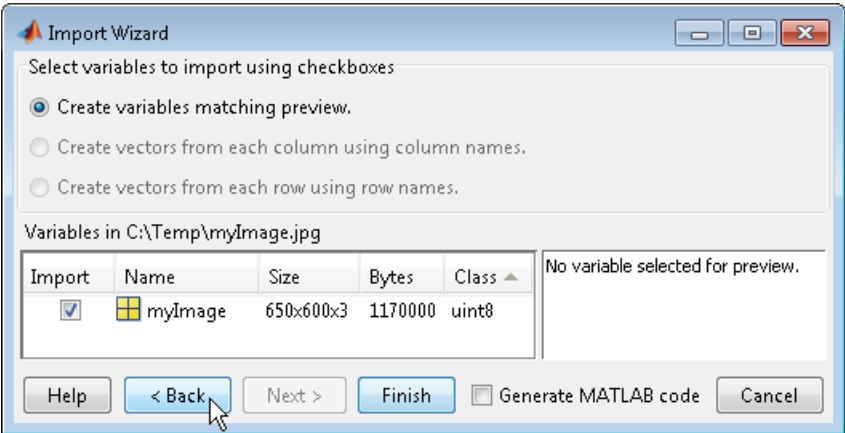
本节内容
“查看文件的内容” （第 1-6 页）
“指定变量” （第 1-6 页）
“生成可重用的 MATLAB 代码” （第 1-7 页）

注意 有关导入文本文件的信息，请参阅“使用导入工具读取文本文件数据”（第 2-6 页）。有关导入电子表格的信息，请参阅“使用导入工具读取电子表格数据”（第 3-4 页）。

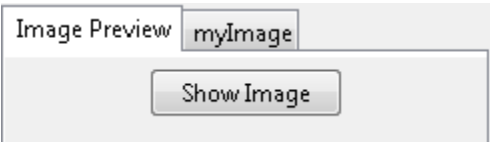
查看文件的内容

通过点击**导入数据**  按钮或调用 `uiimport`，启动导入向导。

要查看图像或视频，或者收听音频，请点击导入向导显示的第一个窗口上的 **< 返回**按钮。



新窗口的右窗格中包含一个预览按钮。在预览选项卡中点击该按钮以显示图像或者播放音频或视频。



指定变量

导入向导会基于数据的格式和内容生成默认变量名称。您可以按下列任何方法更改这些变量：

- “重命名或取消选择变量” （第 1-7 页）
- “导入到结构体数组” （第 1-7 页）

从系统剪贴板中导入的数据的默认变量名称为 `A_pastespecial`。

如果导入向导在文件中检测到单个变量，则默认变量名称为文件名。否则，导入向导将使用对应于 **importdata** 函数的输出字段的默认变量名称。有关输出字段的详细信息，请参阅 **importdata** 函数参考页。

重命名或取消选择变量

要覆盖默认变量名称，请选择相应名称，然后键入新名称。

Variables in C:\Temp\logo.mat

Import	Name ▲	Size	Bytes	Class
<input checked="" type="checkbox"/>	ExpoMapFigurePos	1x4	32	double
<input checked="" type="checkbox"/>	L	43x43	14792	double
<input checked="" type="checkbox"/>	M	60x3	1440	double
<input checked="" type="checkbox"/>	R	43x43	14792	double
<input checked="" type="checkbox"/>	axon	1x1	8	double
<input checked="" type="checkbox"/>	facet	1x1	8	double
<input checked="" type="checkbox"/>	light	1x1	8	double
<input checked="" type="checkbox"/>	source	3x1	24	double
<input checked="" type="checkbox"/>	xb	7x1	56	double
<input checked="" type="checkbox"/>	xg	7x1	56	double
<input checked="" type="checkbox"/>	xh	7x1	56	double

要避免导入特定变量，请清除**导入**列中的相应复选框。

导入到结构体数组

要将数据导入到结构体数组的字段中而不是作为单个变量，可带一个输出参数调用 **uiimport** 来启动导入向导。例如，样本文件 **durer.mat** 包含三个变量：**X**、**caption** 和 **map**。如果发出以下命令

```
durerStruct = uiimport('durer.mat')
```

并点击**完成**按钮，则导入向导将返回一个包含三个字段的标量结构体：

```
durerStruct =
    X: [648x509 double]
   map: [128x3 double]
caption: [2x28 char]
```

要访问特定字段，可使用圆点表示法。例如，查看 **caption** 字段：

```
disp(durerStruct.caption)
```

MATLAB 返回：

```
Albrecht Durer's Melancholia.
Can you find the matrix?
```

有关详细信息，请参阅“结构体数组”。

生成可重用的 MATLAB 代码

要创建读取相似文件而不重新启动导入向导的函数，请选中**生成 MATLAB 代码**复选框。当点击**完成**以完成初始导入操作时，MATLAB 将打开一个包含未保存的函数的编辑器窗口。默认函数名称为 **importfile.m** 或 **importfileN.m**，其中 **N** 为整数。

所生成代码中的函数包括以下功能：

- 对于文本文件，如果从行或列请求向量，则生成的代码也会返回向量。
- 当导入文件中的数据时，该函数包括一个对应于要导入的文件名的输入参数 `fileToRead1`。
- 当导入到结构体数组中时，该函数包括一个对应于结构体的名称的输出参数 `newData1`。

但是，生成的代码具有以下局限：

- 如果在导入向导中重命名或取消选择任何变量，则生成的代码不会反映这些更改。
- 如果未导入到结构体数组，则生成的函数会在基础工作区中创建变量。如果计划从您自己的函数中调用生成的函数，则您的函数无法访问这些变量。要允许您的函数访问数据，可带一个输出参数调用 `uiimport` 来启动导入向导。带输出参数调用生成的函数以在您的函数的工作区中创建结构体数组。

MATLAB 不会自动保存该函数。要保存文件，请选择**保存**。为了获得最佳结果，请使用具有 `.m` 扩展名的函数名称作为文件名。

另请参阅

`imread` | `VideoReader` | `audioread`

详细信息

- “读取视频文件”（第 8-8 页）
- “读取和写入音频文件”（第 8-2 页）
- “导入图像”（第 6-2 页）

导入或导出系列文件

要导入或导出多个文件，可创建一个控制循环，从而一次处理一个文件。在构造循环时：

- 要构建有序的文件名，请使用 `sprintf`。
- 要查找与模式匹配的文件，请使用 `dir`。
- 使用函数语法将文件名传递到导入或导出函数。（有关详细信息，请参阅“选择命令语法或函数语法”。）

例如，要使用 `importdata` 读取名为 `file1.txt` 至 `file20.txt` 的文件：

```
numfiles = 20;
mydata = cell(1, numfiles);

for k = 1:numfiles
    myfilename = sprintf('file%d.txt', k);
    mydata{k} = importdata(myfilename);
end
```

要使用 `imread` 读取匹配 `*.jpg` 的所有文件：

```
jpegFiles = dir('*.jpg');
numfiles = length(jpegFiles);
mydata = cell(1, numfiles);

for k = 1:numfiles
    mydata{k} = imread(jpegFiles(k).name);
end
```

在 MAT 文件中保存和加载部分变量

本节内容
“使用 matfile 函数保存和加载” （第 1-10 页）
“动态加载部分变量” （第 1-11 页）
“避免意外加载整个变量” （第 1-12 页）
“部分加载和保存功能要求 7.3 版本的 MAT 文件” （第 1-12 页）

您可以在 MAT 文件中直接保存和加载部分变量，而无需使用 `matfile` 函数将其加载到内存中。相对于 `load` 或 `save` 函数，使用 `matfile` 函数的主要优点是对于太大而无法装入内存中的数据，可以仅处理该数据集的一部分。当处理这些大型变量时，应一次将尽可能多的数据读取和写入到内存中。否则，重复的文件访问会严重降低代码的性能。

使用 matfile 函数保存和加载

此示例说明如何使用 `matfile` 函数在现有 MAT 文件中加载、修改和保存变量的一部分。

创建具有两个变量 A 和 B 的 7.3 版 MAT 文件。

```
A = rand(5);
B = magic(10);
save example.mat A B -v7.3;
clear A B
```

基于 MAT 文件 `example.mat` 构造一个 `MatFile` 对象。`matfile` 函数创建一个对应于 MAT 文件的 `MatFile` 对象，并包含 `MatFile` 对象的属性。默认情况下，`matfile` 仅允许从现有 MAT 文件加载。

```
exampleObject = matfile('example.mat');
```

要启用保存，请使用 `Writable` 参数调用 `matfile`。

```
exampleObject = matfile('example.mat','Writable',true);
```

或者，构造该对象并在单独的步骤中设置 `Properties.Writable`。

```
exampleObject = matfile('example.mat');
exampleObject.Properties.Writable = true;
```

将 B 的第一行从 `example.mat` 加载到变量 `firstRowB` 中，并修改数据。当索引与 7.3 版的 MAT 文件关联的对象时，MATLAB® 仅加载所指定的变量部分。

```
firstRowB = exampleObject.B(1,:);
firstRowB = 2 * firstRowB;
```

使用存储在 `firstRowB` 中的值更新 `example.mat` 中变量 B 的第一行中的值。

```
exampleObject.B(1,:) = firstRowB;
```

对于非常大的文件，最佳做法是应一次将尽可能多的数据读取和写入到内存中。否则，重复的文件访问会严重影响代码的性能。例如，假设文件包含许多行和列，并且加载一行就需要占用大部分可用内存。这种情况下不要一次更新一个元素，而应该更新一行。

```
[nrowsB,ncolsB] = size(exampleObject,'B');
for row = 1:nrowsB
    exampleObject.B(row,:) = row * exampleObject.B(row,:);
end
```

如果内存大小不是问题，则可以一次更新一个变量的完整内容。

```
exampleObject.B = 10 * exampleObject.B;
```

或者，通过使用 `-append` 选项调用 `save` 函数来更新变量。`-append` 选项要求 `save` 函数仅替换指定的变量 `B`，并保留文件中的其他变量不变。此方法始终要求您加载并保存整个变量。

```
load('example.mat','B');
B(1,:) = 2 * B(1,:);
save('example.mat','-append','B');
```

使用 `matlab.io.MatFile` 对象向文件中添加变量。

```
exampleObject.C = magic(8);
```

还可以通过使用 `-append` 选项调用 `save` 函数来添加变量。

```
C = magic(8);
save('example.mat','-append','C');
clear C
```

动态加载部分变量

此示例说明如何动态地从 MAT 文件访问部分变量。在处理变量名称并不始终已知的 MAT 文件时，此功能很有用。

假设 MAT 文件示例 `topography.mat` 包含一个或多个具有未知名称的数组。构造一个 `MatFile` 对象，该对象对应于文件 `topography.mat`。调用 `who` 以获取文件中的变量名称。

```
exampleObject = matfile('topography.mat');
varlist = who(exampleObject)
```

```
varlist = 4x1 cell
    {'topo'   }
    {'topolegend'}
    {'topomap1' }
    {'topomap2' }
```

`varlist` 是一个元胞数组，包含 `topography.mat` 中的四个变量的名称。

第三个和第四个变量（即 `topomap1` 和 `topomap2`）都是包含颜色图数据的数组。将每个变量的第三列中的颜色图数据加载到结构体数组 `S` 的一个字段中。对于每个字段，指定一个由原始变量名称加前缀 `colormap_` 构成的字段名称。然后，访问每个变量中作为 `exampleObject` 的属性的数据。由于 `varName` 为变量，因此将其括在圆括号中。

```
for index = 3:4
    varName = varlist{index};
    S(1).([colormap_',varName]) = exampleObject.(varName)(:,3);
end
```

查看结构体数组 `S` 的内容。

`S`

```
S = struct with fields:  
    colormap_topomap1: [64x1 double]  
    colormap_topomap2: [128x1 double]
```

`S` 具有两个字段，即 `colormap_topomap1` 和 `colormap_topomap2`，每个字段包含一个列向量。

避免意外加载整个变量

如果您不知道 MAT 文件中一个大型变量的大小，并且想要一次仅加载该变量的一部分，请不要使用 `end` 关键字。使用 `end` 关键字会将相关变量的全部内容临时加载到内存中。对于非常大的变量，加载过程可能会花费较长时间，也可能产生 `Out of Memory` 错误。在这种情况下，应对 `MatFile` 对象调用 `size` 方法。

例如，以下代码会将 `B` 的完整内容临时加载到内存中：

```
lastColB = exampleObject.B(:,end);
```

改用以下代码可提高其性能：

```
[nrows,ncols] = size(exampleObject,'B');  
lastColB = exampleObject.B(:,ncols);
```

类似地，任何时候使用 `matObj.varName` 形式的语法引用变量，例如 `exampleObject.B`，MATLAB 都会临时将整个变量加载到内存中。因此，请确保使用以下语法对 `MatFile` 对象调用 `size` 方法：

```
[nrows,ncols] = size(exampleObject,'B');
```

而不是将 `exampleObject.B` 的完整内容传递到 `size` 函数，

```
[nrows,ncols] = size(exampleObject.B);
```

语法的差异是细微的，但却是重要的。

部分加载和保存功能要求 7.3 版本的 MAT 文件

如果使用的是与 7 版本或更低版本的 MAT 文件关联的 `MatFile` 对象，任何加载或保存操作都会临时将整个变量加载到内存中。

使用 `matfile` 函数创建 7.3 版本格式的文件。例如，下面的代码

```
newfile = matfile('newfile.mat');
```

创建支持部分加载和保存的 MAT 文件。

但是，默认情况下，`save` 函数创建 7 版本的 MAT 文件。通过使用 `-v7.3` 选项调用 `save` 函数，将现有的 MAT 文件转换为 7.3 版本，例如：

```
load('durer.mat');  
save('mycopy_durer.mat','-v7.3');
```


要更改预设以将新文件保存为 7.3 版本格式，请访问**主页**选项卡上的**环境**部分，然后点击  **预设**。选择 **MATLAB > 常规 > MAT 文件**。MATLAB Online™ 中未提供此预设。

另请参阅

[matfile](#) | [save](#) | [load](#)

详细信息

- “保存和加载工作区变量”
- “使用 matfile 函数增大数组”（第 1-17 页）
- “MAT 文件版本”（第 1-14 页）

MAT 文件版本

本节内容


- “MAT 文件版本概述” (第 1-14 页)
- “保存为非默认 MAT 文件版本” (第 1-15 页)
- “数据压缩” (第 1-15 页)
- “加快 7.3 版本的 MAT 文件的保存和加载操作的速度” (第 1-15 页)

MAT 文件版本概述

MAT 文件是二进制 MATLAB 文件，用于存储工作区变量。从 MAT 文件版本 4 开始，随后的几个 MAT 文件版本都支持一组增加的功能。MATLAB 版本 R2006b 和更高版本均支持所有的 MAT 文件版本。

默认情况下，所有保存操作都会创建 7 版本的 MAT 文件。唯一的例外是使用 `matfile` 函数创建新的 MAT 文件时。在这种情况下，默认的 MAT 文件版本是 7.3。

要确定或更改默认 MAT 文件版本，请访问 MAT 文件预设。

- 在主页选项卡上的环境部分中，点击  预设。
- 选择 **MATLAB > 常规 > MAT 文件**。

预设适用于 `save` 函数和保存菜单选项。

MAT 文件的最大大小由您的本机文件系统限定。

下表列出并比较了所有 MAT 文件版本。

MAT 文件版本	支持的 MATLAB 发行版	支持功能	压缩	每个变量大小的最大值	save 函数中 version 参数的值	预设选项
版本 7.3	R2006b (版本 7.3) 或更高版本	保存并加载部分变量，以及版本 7 的所有功能	是 (默认值)	≥ 2 GB (64 位计算机)	'-v7.3'	MATLAB 版本 7.3 或更高版本 (save -v7.3)
版本 7	R14 (版本 7.0) 或更高版本	Unicode® 字符编码。通过编码，可以在使用不同默认字符编码方案的系统之间共享文件，还可以使用版本 6 的所有功能。	是 (默认值)	每变量 2 ³¹ 字节	'-v7'	MATLAB 版本 7 或更高版本 (save -v7)
版本 6	R8 (版本 5) 或更高版本	N 维数组、元胞数组、结构体数组、大于 19 个字符的变量名称，以及版本 4 的所有功能。	否	每变量 2 ³¹ 字节	'-v6'	MATLAB 版本 5 或更高版本 (save -v6)

MAT 文件版本	支持的 MATLAB 发行版	支持功能	压缩	每个变量大小的最大值	save 函数中 version 参数的值	预设选项
版本 4	全部	二维 double、字符和稀疏数组	否	每数组 100,000,000 个元素，每变量 2^{31} 字节	'-v4'	不适用

注意 7.3 版本的 MAT 文件使用基于 HDF5 的格式，该格式要求使用一些存储空间开销来描述文件内容。对于元胞数组、结构体数组或可以存储异构数据类型或其他容器的其他容器，7.3 版本的 MAT 文件有时比版本 7 的 MAT 文件要大。

保存为非默认 MAT 文件版本

当想要执行下列操作时，可保存为默认版本之外的其他 MAT 文件版本：

- 允许使用较低版本的 MATLAB 访问文件。
- 利用 7.3 版本的 MAT 文件功能。
- 通过存储未压缩的数据来降低加载和保存某些文件需要的时间。
- 通过存储压缩后的数据来降低某些文件的大小。

要保存为默认版本之外的其他 MAT 文件版本，请指定一个 **version** 作为 **save** 函数的最后一个输入。例如，要创建名为 **myfile.mat** 的 6 版本的 MAT 文件，请键入：

```
save('myfile.mat','-v6')
```

数据压缩

从 7 版本开始，当将数据写入到 MAT 文件时，MATLAB 会对数据进行压缩以节省存储空间。数据压缩和解压缩会降低所有保存操作和部分加载操作的速度。在大多数情况下，降低文件大小所花费的额外时间是值得的。

在某些情况下，加载压缩数据实际上可能比加载未压缩数据更快。例如，假设数值数组中的数据块分别保存为一个 10 MB 的压缩文件和一个 100 MB 的未压缩文件。加载每个文件的前 10 MB 花费的时间相同。但加载未压缩文件的剩余 90 MB 还要花费加载前 10 MB 的九倍时间，而压缩文件只需要相对较少的解压缩时间即可完成加载。

在下列情况下数据压缩的好处可以忽略：

- 相对于容器的复杂性，数据量不大。例如，压缩和解压缩简单数值数组比压缩和解压缩相同大小的元胞数组或结构体数组花费的时间更少。若数组产生的未压缩文件大小少于 3 MB，则对数组进行压缩的好处非常有限，除非您要通过网络传输这些数据。
- 数据是随机的，没有重复的模式或一致的值。

加快 7.3 版本的 MAT 文件的保存和加载操作的速度

7.3 版本的 MAT 文件使用基于 HDF5 的格式，此格式将数据存储在压缩分块中。从 7.3 版本的 MAT 文件中加载变量的一部分所需的时间取决于如何跨一个或多个分块存储该部分数据。包含所要加载数据的任何

部分的每个分块都必须完全解压缩，才能访问这些数据。对数据重新分块可提高加载操作的性能。要对数据重新分块，请使用 HDF5 分发版本所带的 HDF5 命令行工具。

另请参阅

`save` | `matfile`

详细信息

- “保存和加载工作区变量”

使用 matfile 函数增大数组

当将大量较大的值写入 MAT 文件时，文件大小以非递增方式增加。这种增加方式是预期的结果。为了尽可能减少文件必须增长的次数并确保最佳性能，在为数组填充数据之前将初始值赋给数组。

例如，假设您有一个可写入的 MatFile 对象。

```
fileName = 'matFileOfDoubles.mat';
matObj = matfile(fileName);
matObj.Properties.Writable = true;
```

定义要写入的值的参数。在此例中，将写入一百万个值，一次写入五万个。这些值的均值为 123.4，标准差为 56.7。

```
size = 1000000;
chunk = 50000;
mean = 123.4;
std = 56.7;
```

在使用数据填充数组之前，将初始值零赋给数组中的最后一个元素。

```
matObj.data(1,size) = 0;
```

查看文件的大小。

- 在 Windows 系统上，使用 `dir`。

```
system('dir matFileOfDoubles.mat');
```

- 在 UNIX® 系统上，使用 `ls -ls`：

```
system('ls -ls matFileOfDoubles.mat');
```

在这种情况下，`matFileOfDoubles.mat` 少于 5000 个字节。将一个初始值赋给数组的最后一个元素不会创建一个大文件。但它却可以让系统为 `matFileOfDoubles.mat` 可能较大幅度的大小增加做好准备。

将数据写入数组，一次写入一个分块。

```
nout = 0;
while(nout < size)
    fprintf('Writing %d of %d\n',nout,size);
    chunkSize = min(chunk,size-nout);
    data = mean + std * randn(1,chunkSize);
    matObj.data(1,(nout+1):(nout+chunkSize)) = data;
    nout = nout + chunkSize;
end
```

查看文件的大小。

```
system('dir matFileOfDoubles.mat');
```

由于使用数据填充了数组，因此文件大小现在更大。

另请参阅

`matfile`

详细信息

- “在 MAT 文件中保存和加载部分变量” (第 1-10 页)

在函数中加载变量时出现意外结果

如果有一个从 MAT 文件中加载数据的函数，但发现 MATLAB 未返回预期的结果，请检查 MAT 文件中是否有任何变量与 MATLAB 函数具有相同的名称。与函数名称冲突的公共变量名称包括 `i`、`j`、`mode`、`char`、`size` 和 `path`。

出现这些意外结果是因为在执行函数时，MATLAB 会在运行函数之前对函数中的所有代码进行预处理。但它不会对 `load` 的调用进行预处理，这意味着 MATLAB 对 MAT 文件中的变量并不知悉。因此，与 MATLAB 函数具有相同名称的变量被预处理为 MATLAB 函数，从而导致意外结果。这与脚本不同，MATLAB 会预处理脚本并逐行执行，类似于命令行窗口。

例如，以包含变量 `height`、`width` 和 `length` 的 MAT 文件为例。如果将这些变量加载到诸如 `findVolume` 的函数中，则 MATLAB 会将 `length` 的引用解释为对 MATLAB `length` 函数的调用，并返回错误。

```
function vol = findVolume(myfile)
    load(myfile);
    vol = height * width * length;
end
```

使用长度时出错，
输入参数的数目不足。

为了避免冲突，在定义函数时，请选择下列一种或多种方法：

- 将变量加载到结构体数组中。例如：

```
function vol = findVolume(myfile)
    dims = load(myfile);
    vol = dims.height * dims.width * dims.length;
end
```

- 在对 `load` 函数的调用中明确包括变量名称。例如：

```
function vol = findVolume(myfile)
    load(myfile,'height','width','length')
    vol = height * width * length;
end
```

- 在调用 `load` 之前，初始化函数中的变量。要初始化变量，请将其分配给空矩阵或空字符向量。例如：

```
function vol = findVolume(myfile)
    height = [];
    width = [];
    length = [];
    load(myfile);
    vol = height * width * length;
```

要确定特定变量名称是否与 MATLAB 函数相关联，请使用 `exist` 函数。返回值 5 表示该名称是内置的 MATLAB 函数的名称。

另请参阅

`load`

详细信息

- “保存和加载工作区变量”

创建临时文件

使用 `tempdir` 函数可返回用于在系统上存储临时文件的文件夹的名称。例如，在 The Open Group UNIX 系统上发出 `tempdir` 将返回 `/tmp` 文件夹。

使用 `tempname` 函数可返回临时文件夹中的文件名。返回的文件名是适合保存临时数据的目标。例如，如果需要在临时文件中存储某些数据，可以首先发出以下命令：

```
fileID = fopen(tempname,'w');
```

在大多数情况下，`tempname` 生成唯一标识符 (UUID)。但是，如果您运行 MATLAB 而不运行 JVM™，则 `tempname` 会使用 CPU 计数器和时间生成一个随机名称，不能保证该名称是唯一的。

某些系统在每次重启时都会删除临时文件。在其他系统上，如果将一个文件指定为临时文件，则意味着将不备份该文件。

文本文件

- “导入文本文件” (第 2-2 页)
- “使用导入工具读取文本文件数据” (第 2-6 页)
- “从文本文件中导入日期时间” (第 2-10 页)
- “将文本文件中的数值数据导入矩阵” (第 2-14 页)
- “将文本文件中的混合数据导入表” (第 2-16 页)
- “将文本文件中的混合数据块导入表或元胞数组” (第 2-19 页)
- “将数据写入文本文件” (第 2-22 页)
- “写入到 Diary 文件” (第 2-25 页)
- “从文本文件导入数值数据块” (第 2-26 页)

导入文本文件

MATLAB 能够读写分隔文本文件和格式化文本文件（包括 .csv 和 .txt 文件）中的数值数据和非数值数据。文本文件通常包含数值和文本数据以及变量名称和行名称的混合。您可以在 MATLAB 中将这

些数据表示为表、时间表、矩阵、元胞数组或字符串数组。以编程方式或以交互方式从文本文件导入数据。以编程方式导入以使用定制的导入函数，并进一步控制如何使用导入选项导入数据。以交互方式导入，以使用**导入工具**及其用户界面。

以表的形式导入数据

如果您的文本文件包含表格数据，您可以以表的形式导入数据。**table** 由包含相同类型数据行的列向变量组成。表中的每个变量可以保留不同数据类型和大小，但每个变量必须有相同的行数。有关表的详细信息，请参阅“创建表并为其分配数据”。

使用 **readtable** 函数并指定文件名，从文本文件中将表格数据导入表中。例如，根据示例文件 **airlinesmall.csv** 创建一个表。

```
T = readtable('airlinesmall.csv');
```

显示表的前五行和前五列。

```
T(1:5,1:5)
```

ans =

5×5 table

Year	Month	DayofMonth	DayOfWeek	DepTime
1987	10	21	3	{'642'}
1987	10	26	1	{'1021'}
1987	10	23	5	{'2055'}
1987	10	23	5	{'1332'}
1987	10	22	4	{'629'}

将数据以时间表形式导入

如果文本文件包含表格数据，其中每行都与时间相关联，则可以将数据以时间表的形式导入。与表一样，时间表允许您存储表格数据变量，这些变量可以有不同数据类型和大小，只要它们有相同的行数。此外，时间表还提供特定于时间的函数，以便使用一个或多个时间表中的时间戳数据执行对齐、组合和计算。有关时间表的详细信息，请参阅“创建时间表”。

使用 **readtimetable** 函数将表格数据从文本文件导入时间表。例如，从示例文件 **outages.csv** 创建一个时间表。

```
TT = readtimetable('outages.csv');
```

显示时间表的前五行和前五列。

```
TT(1:5,1:5)
```

ans =

5×5 timetable

OutageTime	Region	Loss	Customers	RestorationTime	Cause
2002-02-01 12:18	{'SouthWest'}	458.98	1.8202e+06	2002-02-07 16:50	{'winter storm' }
2003-01-23 00:49	{'SouthEast'}	530.14	2.1204e+05	NaT	{'winter storm' }
2003-02-07 21:15	{'SouthEast'}	289.4	1.4294e+05	2003-02-17 08:14	{'winter storm' }
2004-04-06 05:44	{'West' }	434.81	3.4037e+05	2004-04-06 06:10	{'equipment fault'}
2002-03-16 06:18	{'MidWest' }	186.44	2.1275e+05	2002-03-18 23:23	{'severe storm' }

以矩阵的形式导入数据

如果您的文本文件包含统一的数据（均为同一类型），您可以将数据以矩阵的形式导入。通过将数据导入矩阵中，您可以处理具有最基本格式的数组。

使用 `readmatrix` 将表格数据从文本文件导入矩阵。例如，从示例文件 `basic_matrix.txt` 中将数据导入矩阵中。

```
M = readmatrix('basic_matrix.txt')
```

M = 5×4

```
6   8   3   1
5   4   7   3
1   6   7  10
4   2   8   2
2   7   5   9
```

将数据以元胞数组的形式导入

元胞数组是一种包含名为元胞的索引数据容器的数据类型，其中的每个元胞都可以包含任意类型的数据。元胞数组通常包含文本列表、文本和数字的组合或者不同大小的数值数组。

您可以使用 `readcell` 将非均匀数据（每列有不同类型）从文本文件导入元胞数组。例如，显示 `basic_cell.txt` 的内容，然后将混合数据导入元胞数组。

```
type basic_cell.txt
```

```
1,2,3
hello,world,NaN
10-Oct-2018 10:27:56,1,
```

```
C = readcell('basic_cell.txt')
```

```
C=3×3 cell array
    {[      1]} {[      2]} {[      3]}
    {'hello'   } {'world'  } {[      NaN]}
    {[10-Oct-2018 10:27:56]} {[      1]} {1×1 missing}
```

您也可以使用 `textscan` 函数和低级 I/O 工作流将格式化数据从文本文件导入元胞数组。低级 I/O 工作流允许对导入数据进行最大程度地的控制。对于大多数工作流来说，这种程度的控制是不必要的。有关使用低级 I/O 导入文本数据的详细信息，请参阅“使用低级 I/O 导入文本数据文件”（第 4-2 页）。

将数据作为字符串数组导入

如果您的文本文件包含纯文本行，您可以将 MATLAB 中的纯文本表示为字符串数组。字符串数组可存储文本片段，并提供一组用于将文本按数据进行处理的功能。例如，您可以对字符串数组进行索引、重构和进行串联，就像处理任何其他类型的数组一样。

使用 `readlines` 将文本文件中的纯文本行导入字符串数组。例如，从示例文本文件 `badpoem.txt` 创建一个字符串数组。由于文本文件有四行纯文本，因此该函数创建了一个 4×1 字符串数组。

```
lines = readlines("badpoem.txt")
```


```
lines = 4x1 string  
    "Oranges and lemons,"  
    "Pineapples and tea."  
    "Orangutans and monkeys,"  
    "Dragonflies or fleas."
```

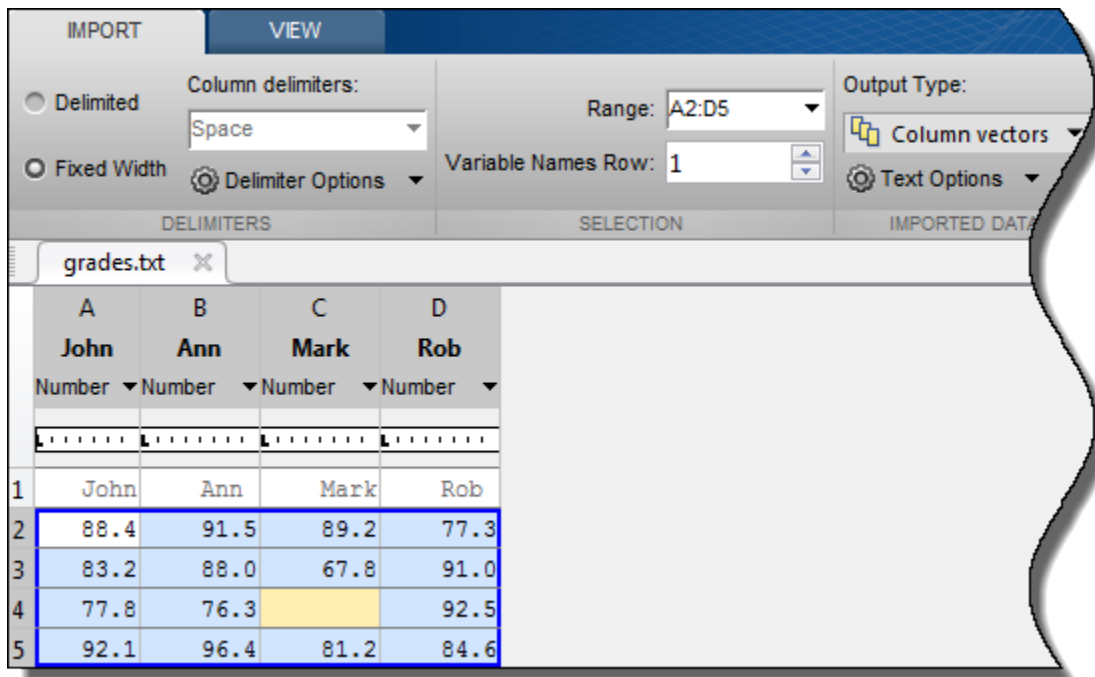
使用导入选项导入数据以进行更多控制

导入表格数据有时需要对导入过程进行更多控制。要自定义导入过程，您可以创建导入选项对象。该对象具有您可以根据自己的导入需求进行调整的属性。例如，您可以更改变量的数据类型或仅导入变量的子集。有关导入选项的详细信息，请参阅 `detectImportOptions`。

以交互方式导入数据

如果您更喜欢使用用户界面，您可以使用**导入工具**以交互方式将数据导入表或其他数据类型。

要打开**导入工具**，请在**主页**选项卡的**变量**部分中，点击**导入数据** 。或者，在当前文件夹浏览器中右键单击文件名，然后选择**导入数据**。然后，选择要导入的文件。使用**导入工具**窗口，设置导入选项，然后点击**导入所选内容**，将数据导入 MATLAB。有关详细信息，请参阅“使用导入工具读取文本文件数据”（第 2-6 页）。



另请参阅

导入工具 | readtable | table | readtimetable | readmatrix | readcell | readlines | textscan | detectImportOptions

详细信息

- “创建表并为其分配数据”
- “访问表中的数据”
- “将文本文件中的混合数据导入表” (第 2-16 页)
- “创建时间表”
- “使用导入工具读取文本文件数据” (第 2-6 页)

使用导入工具读取文本文件数据

本节内容
“以交互方式选择数据” （第 2-6 页）
“从多个文本文件中导入数据” （第 2-8 页）

通过以交互方式选择数据，从文本文件中导入数据。也可以使用导入工具的生成代码功能，对多个文本文件重复此导入操作。

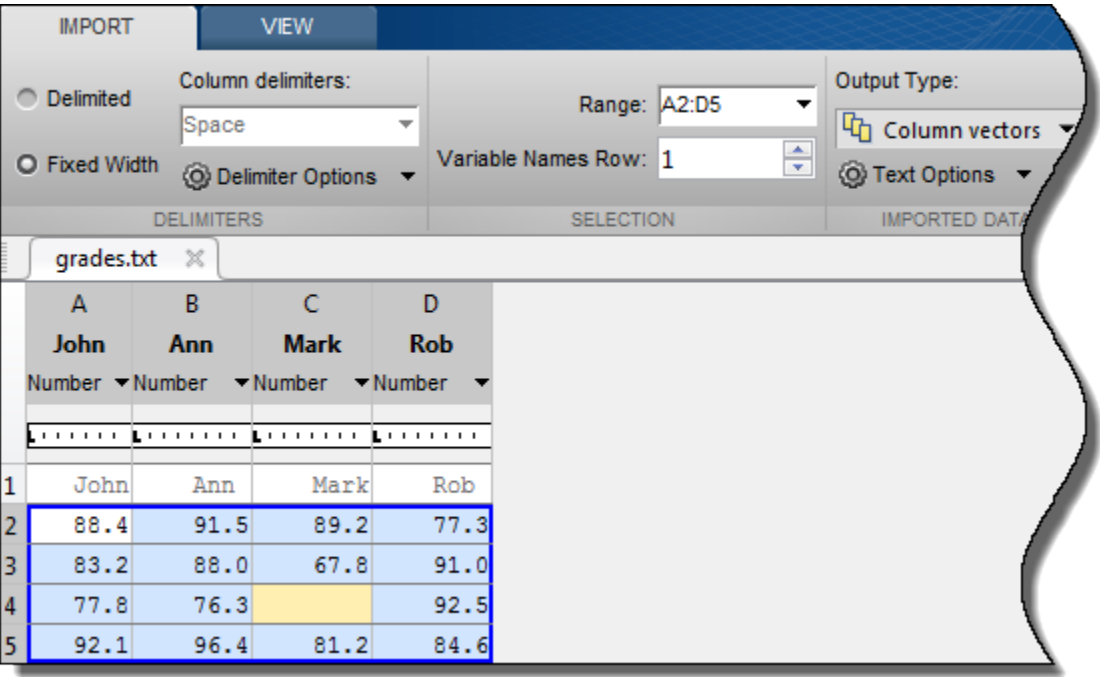
以交互方式选择数据

以下示例演示如何使用导入工具导入具有列标题和数值数据的文本文件中的数据。示例中的文件 `grades.txt` 包含以下数据：

John	Ann	Mark	Rob
88.4	91.5	89.2	77.3
83.2	88.0	67.8	91.0
77.8	76.3	92.5	
92.1	96.4	81.2	84.6

要创建文件，请使用任意文本编辑器复制并粘贴数据。

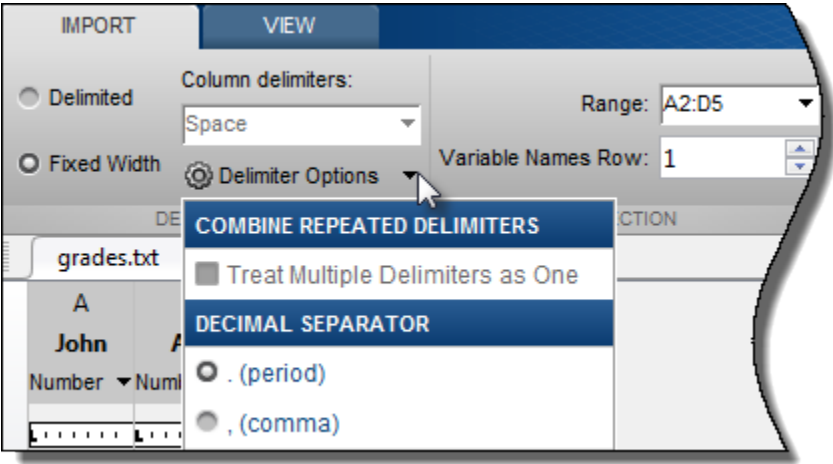
在**主页**选项卡上的**变量**部分中，点击**导入数据** 。或者，在当前文件夹浏览器中右键点击文件名，然后选择**导入数据**。此时将打开导入工具。



导入工具识别出 `grades.txt` 为等宽文件。在**导入的数据**部分中，选择要导入数据的方式。下表指示如何根据所选的选项导入数据。

选择的选项	导入数据的方式
表	将所选数据导入为表。
列向量	将所选数据的每一列导入为单个 $m \times 1$ 向量。
数值矩阵	将所选数据导入为 $m \times n$ 数值数组。
字符串数组	将所选数据导入为包含文本的字符串数组。
元胞数组	将所选数据导入为可包含多种数据类型的元胞数组，例如数值数据和文本。

在**分隔符选项**下，可指定导入工具是使用句点还是逗号作为数值的小数分隔符。

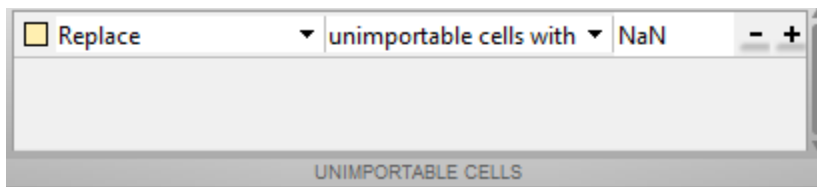


双击变量名称进行重命名。

	A	B	C	D
	John	Ann	Mark	Rob
	N...	NUMBER	NUMBER	NUMBER
1	John	Ann	Mark	Rob
2	88.4	91.5	89.2	77.3
3	83.2	88.0	67.8	91.0
4	77.8	76.3		92.5
5	92.1	96.4	81.2	84.6

此外，还可以使用**所选内容**部分中的**变量名称行**框，来选择希望导入工具将其用作变量名称的文本文件行。


导入工具会突出显示无法导入的单元格。无法导入的单元格是指所包含的数据无法按为该列指定的格式导入的单元格。在以下示例中，第 3 行 C 列的单元格被视为是无法导入的单元格，因为空白单元格不是数值。高亮显示颜色对应于建议的使数据适合数值数组的规则。您可以添加、删除、重新排序或编辑规则，例如将替换值从 NaN 更改为其他值。



所有规则仅应用于导入的数据，不会更改文件中的数据。无论何时，在导入到矩阵或导入到数值列向量时，如果范围包括非数值数据，则必须指定规则。

要查看数据的导入方式，请将光标放在单个单元格上。

	A	B	C	D
	John	Ann	Mark	Rob
	N...	NUMBER	NUMBER	NUMBER
1	John	Ann	Mark	Rob
2	88.4	91.5	89.2	77.3
3	83.2	88.0	Replaced by:NaN	
4	77.8	76.3	NaN	92.5
5	92.1	96.4	81.2	84.6


当点击**导入所选内容**按钮  时，导入工具将会在工作区中创建变量。

有关与导入工具交互的详细信息，请观看此视频。

从多个文本文件中导入数据

要对多个文件执行相同的导入操作，请使用导入工具的代码生成功能。如果通过导入工具逐个导入文件并生成代码，则可以使用此代码来简化重复的操作过程。导入工具可生成程序脚本（您可以编辑并运行该脚本以导入文件）或函数（您可以对每个文件调用该函数）。

假设当前文件夹中有一组文本文件。文件被命名为 **myfile01.txt** 至 **myfile25.txt**，并且您要导入各个文件中从第二行开始的数据。生成代码以导入整个文件集，如下所示：

- 1 在导入工具中打开一个文件。
- 2 点击**导入所选内容** ，然后选择**生成函数**。导入工具生成类似如下脚本的代码，并在编辑器中打开代码。

```
function data = importfile(filename,startRow,endRow)
%IMPORTFILE Import numeric data from a text file as a matrix.
...
```

- 3 保存函数。
- 4 在单独的程序文件或命令行中，创建一个 **for** 循环，将每个文本文件中的数据导入到名为 **myData** 的元胞数组中：

```
numFiles = 25;
startRow = 2;
```

```
endRow = inf;
myData = cell(1,numFiles);

for fileNum = 1:numFiles
    fileName = sprintf('myfile%02d.txt',fileNum);
    myData{fileNum} = importfile(fileName,startRow,endRow);
end
```

`myData` 中的每个元胞包含对应文本文件中的数据数组。例如，`myData{1}` 包含第一个文件 `myfile01.txt` 中的数据。

另请参阅

[readtable](#) | [textscan](#) | [readmatrix](#) | [readcell](#) | [readvars](#) | [readtimetable](#)

详细信息

- “导入文本文件” (第 2-2 页)

从文本文件中导入日期时间

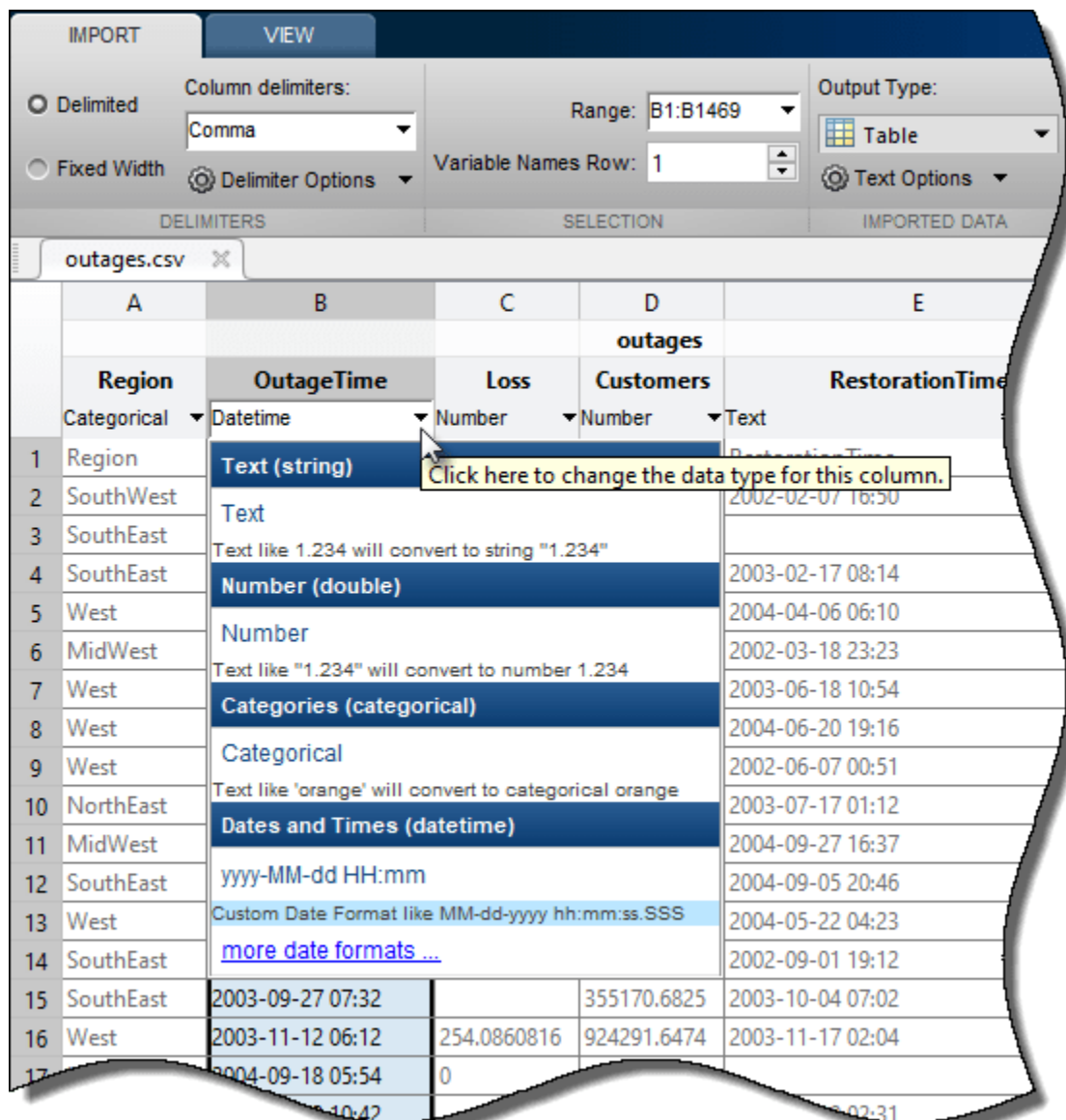
要从列向表格数据中导入格式化的日期时间（例如 '01/01/01' 或 '12:30:45'），有如下三种方法可以选择。

- 导入工具 - 以交互方式选择并导入日期时间。
- `readtable` 函数 - 自动检测包含日期时间的变量，并将其导入到表中。
- 导入选项 - 将 `readtable` 函数与 `detectImportOptions` 配合使用，以便对日期时间变量的导入实行更多控制。例如，可以指定 `FillValue` 和 `DatetimeFormat` 等属性。

此示例说明如何使用以上各种方法从文本文件中导入日期时间。

导入工具

使用导入工具打开文件 `outages.csv`。使用每列的下拉菜单指定日期时间的格式。您可以从预定义的日期格式中选择，或者输入自定义格式。要导入 `OutageTime` 列，请指定自定义格式 `yyyy-MM-dd HH:mm`。然后，点击**导入所选内容**按钮，将数据导入到工作区中。



readtable 函数

使用 `readtable` 函数并显示 `OutageTime` 变量的前 10 行。`readtable` 会自动检测日期时间变量和格式。

```
filename = 'outages.csv';
T = readtable(filename);
T.OutageTime(1:10)
```

```
ans = 10x1 datetime
    2002-02-01 12:18
    2003-01-23 00:49
    2003-02-07 21:15
    2004-04-06 05:44
    2002-03-16 06:18
    2003-06-18 02:49
```

```
2004-06-20 14:39
2002-06-06 19:28
2003-07-16 16:23
2004-09-27 11:09
```

导入选项

使用导入选项对象，对日期时间变量的导入实行更多控制。例如，更改日期时间显示格式，或指定缺失日期的填充值。

为 `outages.csv` 文件创建导入选项对象，并显示变量 `RestorationTime` 的变量导入选项。`detectImportOptions` 函数会自动检测变量的数据类型。

```
opts = detectImportOptions(filename);
getvaropts(opts,'RestorationTime')
```

```
ans =
    DatetimeVariableImportOptions with properties:
```

```
Variable Properties:
    Name: 'RestorationTime'
    Type: 'datetime'
    FillValue: NaT
    TreatAsMissing: {}
    QuoteRule: 'remove'
    Prefixes: {}
    Suffixes: {}
    EmptyFieldRule: 'missing'
```

```
Datetime Options:
    DatetimeFormat: 'default'
    DatetimeLocale: 'en_US'
    InputFormat: ''
    TimeZone: ''
```

导入数据并显示变量 `RestorationTime` 的前 10 行。第二行包含一个 `NaT`，指示缺失的日期时间值。

```
T = readtable(filename,opts);
T.RestorationTime(1:10)
```

```
ans = 10x1 datetime
    2002-02-07 16:50
    NaT
    2003-02-17 08:14
    2004-04-06 06:10
    2002-03-18 23:23
    2003-06-18 10:54
    2004-06-20 19:16
    2002-06-07 00:51
    2003-07-17 01:12
    2004-09-27 16:37
```

要使用其他日期时间显示格式，请更新 `DatetimeFormat` 属性，然后通过使用 `FillValue` 属性，将缺失的值替换为当前的日期时间。显示更新的变量选项。

```
opts = setvaropts(opts,'RestorationTime', ...
    'DatetimeFormat','MMMM d, yyyy HH:mm:ss Z',...
    'FillValue','now');
getvaropts(opts,'RestorationTime')
```

```
ans =
    DatetimeVariableImportOptions with properties:
```

```
    Variable Properties:
```

```
        Name: 'RestorationTime'
        Type: 'datetime'
        FillValue: January 30, 2023 15:23:28 *
        TreatAsMissing: {}
        QuoteRule: 'remove'
        Prefixes: {}
        Suffixes: {}
        EmptyFieldRule: 'missing'
```

```
    Datetime Options:
```

```
        DatetimeFormat: 'MMMM d, yyyy HH:mm:ss Z'
        DatetimeLocale: 'en_US'
        InputFormat: ''
        TimeZone: ''
```

使用更新的导入选项读取数据，并显示变量的前 10 行。

```
T = readtable(filename,opts);
T.RestorationTime(1:10)
```

```
ans = 10x1 datetime
    2002-02-07 16:50
    2023-01-30 15:23
    2003-02-17 08:14
    2004-04-06 06:10
    2002-03-18 23:23
    2003-06-18 10:54
    2004-06-20 19:16
    2002-06-07 00:51
    2003-07-17 01:12
    2004-09-27 16:37
```

有关 `datetime` 变量选项的详细信息，请参阅 `setvaropts` 参考页。

另请参阅

导入工具 | `readtable` | `detectImportOptions` | `setvaropts` | `readmatrix` | `readcell` | `readvars` | `readtimetable`

详细信息

- “将文本文件中的混合数据导入表” (第 2-16 页)

将文本文件中的数值数据导入矩阵

从存储为逗号分隔的或带分隔符的文本文件的文件中，将数值数据导入为 MATLAB 数组。

导入逗号分隔的数据

以下示例说明如何从文本文件中导入逗号分隔的数值数据。创建样本文件，读取文件中的所有数据，然后仅读取从指定位置开始的部分数据。

创建一个名为 **ph.dat** 且包含逗号分隔的数据的样本文件，然后显示该文件的内容。

```
rng('default')
A = 0.9*randi(99,[3 4]);
writematrix(A,'ph.dat','Delimiter','.')
type('ph.dat')

72.9,81.9,25.2,86.4
81,56.7,49.5,14.4
11.7,9,85.5,87.3
```

使用 **readmatrix** 函数读取该文件。函数将返回一个包含文件中的数据的 **3×4 double** 数组。

```
M = readmatrix('ph.dat')

M = 3×4

    72.9000    81.9000    25.2000    86.4000
    81.0000    56.7000    49.5000    14.4000
    11.7000     9.0000    85.5000    87.3000
```

仅导入文件中从第一行、第三列开始的数据的矩形部分。创建一个导入选项对象，并使用 **SelectedVariableNames** 和 **DataLines** 属性指定要导入的列和行。然后从文件中导入所选部分的数据。

```
opts = detectImportOptions('ph.dat');
opts.SelectedVariableNames = {'Var3','Var4'};
opts.DataLines = [1 3];
readmatrix('ph.dat',opts)

ans = 3×2

    25.2000    86.4000
    49.5000    14.4000
    85.5000    87.3000
```

导入分隔的数值数据

以下示例演示如何使用 **writematrix** 函数导入由任何单个字符分隔的数值数据。创建样本文件，读取整个文件，然后读取文件从指定位置开始的部分数据。

创建一个名为 **num.txt** 的制表符分隔的文件，其中包含一个 **4×4** 数值数组，然后显示该文件的内容。


```
rng('default')
A = randi(99,[4,4]);
writematrix(A,'num.txt','Delimiter','\t')
type('num.txt')
```

```
81  63  95  95
90  10  96  49
13  28  16  80
91  55  97  15
```

读取整个文件。`readmatrix` 函数会自动确定分隔符并返回一个 4×4 的 `double` 数组。

```
M = readmatrix('num.txt')
```

```
M = 4×4
```

```
81  63  95  95
90  10  96  49
13  28  16  80
91  55  97  15
```

仅读取文件中从第二行、第三列开始的矩形数据块。创建一个导入选项对象，并使用 `SelectedVariableNames` 和 `DataLines` 属性指定要导入的列和行。然后从文件中导入所选部分的数据。

```
opts = detectImportOptions('num.txt');
opts.SelectedVariableNames = {'Var3','Var4'};
opts.DataLines = [2 4];
readmatrix('num.txt',opts)
```

```
ans = 3×2
```

```
96  49
16  80
97  15
```

另请参阅

`readmatrix` | `readcell` | `readvars` | `readtimetable`

详细信息

- “导入文本文件” (第 2-2 页)

将文本文件中的混合数据导入表

此示例说明如何使用 `readtable` 函数将混合的文本和数值数据导入到表中，指定变量的数据类型，然后将新变量附加到表。

样本文件概述

样本文件 `outages.csv` 包含表示美国电力中断的数据。文件的前几行如下：

```
Region,OutageTime,Loss,Customers,RestorationTime,Cause
SouthWest,2002-01-20 11:49,672,2902379,2002-01-24 21:58,winter storm
SouthEast,2002-01-30 01:18,796,336436,2002-02-04 11:20,winter storm
SouthEast,2004-02-03 21:17,264.9,107083,2004-02-20 03:37,winter storm
West,2002-06-19 13:39,391.4,378990,2002-06-19 14:27,equipment fault
```

读取文本文件

使用 `readtable` 导入数据，并显示前五。 `readtable` 函数会自动检测分隔符和变量类型。

```
T = readtable('outages.csv');
head(T,5)
```

Region	OutageTime	Loss	Customers	RestorationTime	Cause
{'SouthWest'}	2002-02-01 12:18	458.98	1.8202e+06	2002-02-07 16:50	{'winter storm' }
{'SouthEast'}	2003-01-23 00:49	530.14	2.1204e+05	NaT	{'winter storm' }
{'SouthEast'}	2003-02-07 21:15	289.4	1.4294e+05	2003-02-17 08:14	{'winter storm' }
{'West' }	2004-04-06 05:44	434.81	3.4037e+05	2004-04-06 06:10	{'equipment fault'}
{'MidWest' }	2002-03-16 06:18	186.44	2.1275e+05	2002-03-18 23:23	{'severe storm' }

在导入之前指定变量数据类型

根据文件中的变量类型，将变量数据类型更新为相应的 MATLAB® 数据类型可能有益于数据。例如， `outages.csv` 中的第一列和第六列为分类列。通过将这两个列指定为 `categorical` 数组，您可以运用 MATLAB 函数处理分类数据。

要指定变量的数据类型，有如下方法可以选择：

- 指定 `readtable` 中的 `Format` 名称-值对组
- 设置文件导入选项的 `VariableTypes` 属性

使用 `Format` 名称-值对组指定变量的数据类型，读取数据，并显示前五行数据。在 `formatSpec` 设定符的 `%{yyyy-MM-dd HH:mm}D` 部分中，花括号之间的文本描述了日期时间数据的格式。 `Format` 中指定的值指定了：

- 文件中的第一列和最后一列为分类数据
- 第二列和第五列为格式化的日期时间数据
- 第三列和第四列为浮点值

```
formatSpec = '%C%{yyyy-MM-dd HH:mm}D%f%f%{yyyy-MM-dd HH:mm}D%C';
T = readtable('outages.csv','Format',formatSpec);
head(T,5)
```

Region	OutageTime	Loss	Customers	RestorationTime	Cause
SouthWest	2002-02-01 12:18	458.98	1.8202e+06	2002-02-07 16:50	winter storm
SouthEast	2003-01-23 00:49	530.14	2.1204e+05	NaT	winter storm
SouthEast	2003-02-07 21:15	289.4	1.4294e+05	2003-02-17 08:14	winter storm
West	2004-04-06 05:44	434.81	3.4037e+05	2004-04-06 06:10	equipment fault
MidWest	2002-03-16 06:18	186.44	2.1275e+05	2002-03-18 23:23	severe storm

或者，通过使用导入选项的 `setvartype` 函数，指定变量的数据类型。首先，为文件创建一个导入选项对象。数据文件包含了不同类型的变量。指定第一个和最后一个变量为 `categorical` 数组，第二个和第五个变量为 `datetime` 数组，剩余变量为 `double`。

```
opts = detectImportOptions('outages.csv');
varNames = opts.VariableNames;
varTypes = {'categorical','datetime','double',...
            'double','datetime','categorical'};
opts = setvartype(opts,varNames,varTypes);
```

将 `readtable` 与 `opts` 配合使用以导入数据，然后显示前五。

```
T = readtable('outages.csv',opts);
head(T,5)
```

Region	OutageTime	Loss	Customers	RestorationTime	Cause
SouthWest	2002-02-01 12:18	458.98	1.8202e+06	2002-02-07 16:50	winter storm
SouthEast	2003-01-23 00:49	530.14	2.1204e+05	NaT	winter storm
SouthEast	2003-02-07 21:15	289.4	1.4294e+05	2003-02-17 08:14	winter storm
West	2004-04-06 05:44	434.81	3.4037e+05	2004-04-06 06:10	equipment fault
MidWest	2002-03-16 06:18	186.44	2.1275e+05	2002-03-18 23:23	severe storm

将新变量追加到表中

表 `T` 中包含了 `OutageTime` 和 `RestorationTime`。计算每次电力中断的持续时间并将此数据追加到表中。

```
T.Duration = T.RestorationTime - T.OutageTime;
head(T,5)
```

Region	OutageTime	Loss	Customers	RestorationTime	Cause	Duration
SouthWest	2002-02-01 12:18	458.98	1.8202e+06	2002-02-07 16:50	winter storm	148:32:00
SouthEast	2003-01-23 00:49	530.14	2.1204e+05	NaT	winter storm	NaN
SouthEast	2003-02-07 21:15	289.4	1.4294e+05	2003-02-17 08:14	winter storm	226:59:00
West	2004-04-06 05:44	434.81	3.4037e+05	2004-04-06 06:10	equipment fault	00:26:00
MidWest	2002-03-16 06:18	186.44	2.1275e+05	2002-03-18 23:23	severe storm	65:05:00

另请参阅

`readtimetable` | `readtable` | `detectImportOptions` | `setvaropts` | `setvartype` | `preview` | `head`

详细信息

- “创建表并为其分配数据”
- “从文本文件中导入日期时间”（第 2-10 页）
- “访问表中的数据”

将文本文件中的混合数据块导入表或元胞数组

本示例从文本文件中读取一个包含文本和数值的混合数据块，然后将该数据块导入到表中或元胞数组中。

数据文件概述

样本文件 **bigfile.txt** 中包含以 **##** 开头的注释行。数据排列在五列中：第一列包含指示时间戳的文本。第二、第三和第四列包含数值数据，指示温度、湿度和风速。最后一列包含说明性文本。显示 **bigfile.txt** 文件内容。

```
type('bigfile.txt')

## A ID = 02476
## YKZ Timestamp Temp Humidity Wind Weather
06-Sep-2013 01:00:00 6.6 89 4 clear
06-Sep-2013 05:00:00 5.9 95 1 clear
06-Sep-2013 09:00:00 15.6 51 5 mainly clear
06-Sep-2013 13:00:00 19.6 37 10 mainly clear
06-Sep-2013 17:00:00 22.4 41 9 mostly cloudy
06-Sep-2013 21:00:00 17.3 67 7 mainly clear
## B ID = 02477
## YVR Timestamp Temp Humidity Wind Weather
09-Sep-2013 01:00:00 15.2 91 8 clear
09-Sep-2013 05:00:00 19.1 94 7 n/a
09-Sep-2013 09:00:00 18.5 94 4 fog
09-Sep-2013 13:00:00 20.1 81 15 mainly clear
09-Sep-2013 17:00:00 20.1 77 17 n/a
09-Sep-2013 18:00:00 20.0 75 17 n/a
09-Sep-2013 21:00:00 16.8 90 25 mainly clear
## C ID = 02478
## YYZ Timestamp Temp Humidity Wind Weather
```

将数据块导入为表

要将数据导入为表，请将 **readtable** 与导入选项配合使用。

使用 **detectImportOptions** 函数，为文件创建一个导入选项对象。使用 **DataLines** 属性指定数据位置。例如，第 3 行至第 8 行包含第一个数据块。也可选择使用 **VariableNames** 属性来指定变量的名称。最后，将 **readtable** 与 **opts** 对象配合使用，导入第一个数据块。

```
opts = detectImportOptions('bigfile.txt');
opts.DataLines = [3 8];
opts.VariableNames = {'Timestamp','Temp',...
    'Humidity','Wind','Weather'};
T_first = readtable('bigfile.txt',opts)
```

T_first=6×5 table

Timestamp	Temp	Humidity	Wind	Weather
06-Sep-2013 01:00:00	6.6	89	4	{'clear' }
06-Sep-2013 05:00:00	5.9	95	1	{'clear' }
06-Sep-2013 09:00:00	15.6	51	5	{'mainly clear' }
06-Sep-2013 13:00:00	19.6	37	10	{'mainly clear' }
06-Sep-2013 17:00:00	22.4	41	9	{'mostly cloudy' }
06-Sep-2013 21:00:00	17.3	67	7	{'mainly clear' }

通过将 **DataLines** 属性更新为第二个块的位置，读取第二个块。

```
opts.DataLines = [11 17];
T_second = readtable('bigfile.txt',opts)
```

```
T_second=7×5 table
      Timestamp      Temp  Humidity  Wind      Weather
    _____  _
09-Sep-2013 01:00:00  15.2    91      8  {'clear'   }
09-Sep-2013 05:00:00  19.1    94      7  {'n/a'     }
09-Sep-2013 09:00:00  18.5    94      4  {'fog'     }
09-Sep-2013 13:00:00  20.1    81     15  {'mainly clear'}
09-Sep-2013 17:00:00  20.1    77     17  {'n/a'     }
09-Sep-2013 18:00:00   20     75     17  {'n/a'     }
09-Sep-2013 21:00:00  16.8    90     25  {'mainly clear'}
```

将数据块导入为元胞数组

要以元胞数组形式导入数据，您可以将 **readcell** 函数与 **detectImportOptions** 结合使用，或使用 **textscan** 函数。首先使用 **readcell** 函数导入数据块，然后使用 **textscan** 执行相同的导入。

要使用 **readcell** 函数执行导入，请使用 **detectImportOptions** 函数为文件创建一个导入选项对象。使用 **DataLines** 属性指定数据位置。然后，使用 **readcell** 函数和导入选项对象 **opts** 执行导入操作。

```
opts = detectImportOptions('bigfile.txt');
opts.DataLines = [3 8]; % fist block of data
C = readcell('bigfile.txt',opts)
```

```
C=6×5 cell array
    {[06-Sep-2013 01:00:00]} {[ 6.6000]} {[89]} {[ 4]} {'clear'   }
    {[06-Sep-2013 05:00:00]} {[ 5.9000]} {[95]} {[ 1]} {'clear'   }
    {[06-Sep-2013 09:00:00]} {[15.6000]} {[51]} {[ 5]} {'mainly clear'}
    {[06-Sep-2013 13:00:00]} {[19.6000]} {[37]} {[10]} {'mainly clear'}
    {[06-Sep-2013 17:00:00]} {[22.4000]} {[41]} {[ 9]} {'mostly cloudy'}
    {[06-Sep-2013 21:00:00]} {[17.3000]} {[67]} {[ 7]} {'mainly clear'}
```

要使用 **textscan** 函数执行导入，请使用 **N** 指定块大小，并使用 **formatSpec** 指定数据字段的格式。例如，将 **%s** 用于文本变量，将 **%D** 用于日期和时间变量，或将 **%c** 用于分类变量。将 **'DateLocale'** 名称-值参数设置为 **'en_US'**，以确保月份名称用英语解释。使用 **fopen** 打开文件。然后，函数将返回文件标识符 **fileID**。下一步，使用 **textscan** 函数读取文件。

```
N = 6;
formatSpec = '%D %f %f %f %c';
fileID = fopen('bigfile.txt');
```

读取第一个块并显示变量 **Humidity** 的内容。

```
C_first = textscan(fileID,formatSpec,N,'CommentStyle','##','Delimiter','\t','DateLocale','en_US')
```

```
C_first=1×5 cell array
    {6×1 datetime} {6×1 double} {6×1 double} {6×1 double} {6×1 char}
```

```
C_first{3}
```

```
ans = 6×1
```

```

89
NaN
95
NaN
51
NaN

```

更新块大小 N，并读取第二个块。显示第五个变量 **Weather** 的内容。

```

N = 7;
C_second = textscan(fileID,formatSpec,N,'CommentStyle','##','Delimiter','\t','DateLocale','en_US')

```

```

C_second=1×5 cell array
    {7×1 datetime}  {7×1 double}  {7×1 double}  {7×1 double}  {7×1 char}

```

```

C_second{5}

```

```

ans = 7×1 char array

```

```

'm'
'↵'
'm'
'↵'
'm'
'↵'
'c'

```

关闭文件。

```

fclose(fileID);

```

另请参阅

[readcell](#) | [readtable](#) | [textscan](#) | [fopen](#) | [detectImportOptions](#)

详细信息

- “访问元胞数组中的数据”
- “在文件中移动” (第 4-10 页)

将数据写入文本文件

本节内容
“将表导出到文本文件” （第 2-22 页）
“将元胞数组导出到文本文件” （第 2-23 页）
“将数值数组导出到文本文件” （第 2-24 页）

将表、元胞数组或数值数组中包含的表格数据从 MATLAB 工作区导出到文本文件。

将表导出到文本文件

可使用 `writetable` 函数将表格数据从 MATLAB® 工作区导出到文本文件。创建样本表，将表写入文本文件，然后指定更多选项并再次将表写入文本文件。

创建包含变量 `Pitch`、`Shape`、`Price` 和 `Stock` 的样本表 `T`。

```
Pitch = [0.7;0.8;1;1.25;1.5];
Shape = {'Pan';'Round';'Button';'Pan';'Round'};
Price = [10.0;13.59;10.50;12.00;16.69];
Stock = [376;502;465;1091;562];
T = table(Pitch,Shape,Price,Stock)
```

T=5×4 table

Pitch	Shape	Price	Stock
0.7	{'Pan' }	10	376
0.8	{'Round' }	13.59	502
1	{'Button' }	10.5	465
1.25	{'Pan' }	12	1091
1.5	{'Round' }	16.69	562

将表 `T` 导出到一个名为 `tabledata.txt` 的文本文件。查看文件的内容。默认情况下，`writetable` 会写入逗号分隔的数据，将表变量名称作为列标题。

```
writetable(T,'tabledata.txt');
type tabledata.txt
```

```
Pitch,Shape,Price,Stock
0.7,Pan,10,376
0.8,Round,13.59,502
1,Button,10.5,465
1.25,Pan,12,1091
1.5,Round,16.69,562
```

创建表 `T2`，使用 `RowNames` 名称-值对组参数指定行名称。

```
rowNames = {'M4';'M5';'M6';'M8';'M10'};
T2 = table(Pitch,Shape,Price,Stock,'RowNames',rowNames)
```

T2=5×4 table

Pitch	Shape	Price	Stock
-------	-------	-------	-------


```
M4    0.7  {'Pan' }    10   376
M5    0.8  {'Round'}  13.59  502
M6     1   {'Button'} 10.5   465
M8    1.25 {'Pan' }    12   1091
M10   1.5  {'Round' }  16.69  562
```

将 T2 导出到名为 **tabledata2.txt** 并以制表符分隔的文本文件。使用 **Delimiter** 名称-值对组参数指定制表符分隔符，并使用 **WriteRowNames** 名称-值对组参数以包括行名称。查看文件的内容。

```
writetable(T2,'tabledata2.txt','Delimiter','\t','WriteRowNames',true);
type tabledata2.txt
```

```
Row Pitch Shape Price Stock
M4  0.7  Pan   10   376
M5  0.8  Round 13.59 502
M6  1   Button 10.5  465
M8  1.25 Pan   12   1091
M10 1.5  Round 16.69 562
```

将元胞数组导出到文本文件

可采用以下方法之一，将元胞数组从 MATLAB® 工作区导出到文本文件：

- 使用 **writecell** 函数将元胞数组导出到文本文件。
- 通过指定输出数据的格式，使用 **fprintf** 导出元胞数组。

创建样本元胞数组 **C**。

```
C = {'Atkins',32,77.3,'M';'Cheng',30,99.8,'F';'Lam',31,80.2,'M'}
```

```
C = 3×4 cell array
{'Atkins'} {[32]} {[77.3000]} {'M'}
{'Cheng' } {[30]} {[99.8000]} {'F'}
{'Lam' }   {[31]} {[80.2000]} {'M'}
```

使用 **writecell** 导出元胞数组。

```
writecell(C,'data.dat')
```

查看文件的内容。

```
type data.dat
```

```
Atkins,32,77.3,M
Cheng,30,99.8,F
Lam,31,80.2,M
```

或者，使用 **fprintf** 导入元胞数组。打开一个可供写入的名为 **celldata.dat** 的文件。使用格式设定符定义 **formatSpec**，以描述文件中的数据模式。典型的格式设定符包括：表示字符向量的 '%s'，表示整数的 '%d' 或者表示浮点数的 '%f'。使用空格分隔每个格式设定符，以指示对输出文件使用空格分隔符。在每行数据的末尾包括换行符 ('\n')。

```
fileID = fopen('celldata.dat','w');  
formatSpec = '%s %d %2.1f %s\n';
```

确定 C 的大小，并使用 `fprintf` 函数一次导出一行数据。然后关闭文件。`fprintf` 将写入一个空格分隔的文件。

```
[nrows,ncols] = size(C);  
for row = 1:nrows  
    fprintf(fileID,formatSpec,C{row,:});  
end  
fclose(fileID);
```

查看文件的内容。

```
type celldata.dat
```

```
Atkins 32 77.3 M  
Cheng 30 99.8 F  
Lam 31 80.2 M
```

将数值数组导出到文本文件

可使用 `writematrix` 将数值数组导出到文本文件。

创建数值数组 A。

```
A = magic(5)/10
```

```
A = 5×5
```

```
1.7000  2.4000  0.1000  0.8000  1.5000  
2.3000  0.5000  0.7000  1.4000  1.6000  
0.4000  0.6000  1.3000  2.0000  2.2000  
1.0000  1.2000  1.9000  2.1000  0.3000  
1.1000  1.8000  2.5000  0.2000  0.9000
```

将该数值数组写入到 `myData.dat`，并将分隔符指定为 ';'。然后，查看文件的内容。

```
writematrix(A,'myData.dat','Delimiter',';')  
type myData.dat
```

```
1.7;2.4;0.1;0.8;1.5  
2.3;0.5;0.7;1.4;1.6  
0.4;0.6;1.3;2;2.2  
1;1.2;1.9;2.1;0.3  
1.1;1.8;2.5;0.2;0.9
```

另请参阅

`writematrix` | `writecell` | `writetimetable` | `fprintf` | `type` | `writetable`

写入到 Diary 文件

要保留 MATLAB 会话的活动日志，请使用 **diary** 函数。**diary** 在磁盘文件中创建 MATLAB 会话的逐字副本（不包括图形）。

例如，如果工作区中具有数组 A，

```
A = [ 1 2 3 4; 5 6 7 8];
```

在 MATLAB 提示符处执行以下命令以使用 **diary** 导出此数组：

- 1 启用 **diary** 函数。您也可以为 **diary** 创建的输出文件命名：

```
diary my_data.out
```

- 2 显示要导出的数组的内容。以下示例显示数组 A。您也可以显示元胞数组或其他 MATLAB 类：

```
A =  
    1    2    3    4  
    5    6    7    8
```

- 3 关闭 **diary** 函数：

```
diary off
```

diary 创建文件 **my_data.out**，并记录您在关闭该文件前在 MATLAB 会话中执行的所有命令。

```
A =
```

```
    1    2    3    4  
    5    6    7    8
```

```
diary off
```

- 4 在文本编辑器中打开 **diary** 文件 **my_data.out**，如果需要，删除外部文本。

从文本文件导入数值数据块

此示例说明如何读取文本文件中以数据块形式组织的数值数据。文件中的每个数据块可以有不同格式。您可以使用 `textscan` 以元胞数组形式读取所有数据块，一次读取一个。

文件格式概述

示例文本文件 `test80211.txt` 中的信息是无线网络通信质量测试的结果。该示例文件由四行介绍和后续的几个数据块组成。每个数据块代表一个不同环境（例如，移动、室内、室外），并具有以下格式：

- 开头的两行说明文字
- 文本 `Num SNR=` 后跟数值 `m`
- 以 `m` 列和任意行数的表形式组织的数值数据（数据以逗号分隔。）
- 文本 `*EOB`，表示该数据块的结束

例如，某个数据块的格式如下：

* Indoor2

* SNR Vs test No

Num SNR=3

,-5.00E+00,-4.00E+00,

1.00E+00,3.32E-07,9.12E-07

2.00E+00,1.49E-07,2.44E-07

3.00E+00,6.04E-07,2.53E-07

4.00E+00,1.53E-07,4.25E-07

5.00E+00,1.82E-07,1.83E-07

6.00E+00,6.27E-07,8.21E-07

7.00E+00,9.10E-08,1.53E-08

8.00E+00,8.73E-07,6.45E-07

9.00E+00,4.40E-07,1.33E-07

*EOB

数值数据表示若干独立测试在一系列噪声级别上的误差率。第一列表示测试编号。要查看整个示例文件，请在命令行中键入：

```
open test80211.txt
```

打开要读取的文本文件

打开文件并创建一个文件标识符。

```
fileID = fopen('test80211.txt','r');
```

读取简介行

读取四个简介行，其中包含由换行符分隔的文本。`textscan` 返回一个 1×1 元胞数组，其中包含一个 4×1 字符向量元胞数组。

```
Intro = textscan(fileID,'%s',4,'Delimiter','\n')
```

```
Intro = 1x1 cell array
      {4x1 cell}
```

查看第一个元胞的内容。

```
disp(Intro{1})
```

```
{'*CCX'          }
{'*CCX WiFi conformance test'}
{'*CCX BER Results'      }
{'*CCX'          }
```

读取每个数据块

对于每个数据块，我们要读取数据的标题、数值 `m`、列标题，然后读取数据本身。首先，初始化数据块索引。

```
Block = 1;
```

以 `while` 循环方式读取每个数据块。该循环一直执行到文件结束且 `~feof` 返回 `false`。`textscan` 函数以名为 `InputText` 的元胞数组形式返回每个数据块中的数据。使用 `cell2mat` 将每个元胞数组转换为数值数组，并将数值数组存储在名为 `Data` 的元胞数组中。元胞数组允许存储不同大小的数据块。

```
while (~feof(fileID))                % For each block:

    fprintf('Block: %s\n', num2str(Block))    % Print block number to the screen
    InputText = textscan(fileID,'%s',2,'delimiter','\n'); % Read 2 header lines
    HeaderLines{Block,1} = InputText{1};
    disp(HeaderLines{Block});              % Display header lines

    InputText = textscan(fileID,'Num SNR = %f'); % Read the numeric value
                                           % following the text, Num SNR =
    NumCols = InputText{1};                % Specify that this is the
                                           % number of data columns

    FormatString = repmat('%f',1,NumCols);    % Create format string
                                           % based on the number
                                           % of columns
    InputText = textscan(fileID,FormatString, ... % Read data block
                          'delimiter',';');

    Data{Block,1} = cell2mat(InputText);
    [NumRows,NumCols] = size(Data{Block});    % Determine size of table
    disp(cellstr(['Table data size: ' ...
                  num2str(NumRows) ' x ' num2str(NumCols)]));
    disp(' ');                               % New line

    eob = textscan(fileID,'%s',1,'delimiter','\n'); % Read and discard end-of-block marker
```

```
Block = Block+1;           % Increment block index
end
```

Block: 1

```
{'*    Mobile1'    }
{'*    SNR Vs test No'}

{'Table data size: 30 x 19'}
```

Block: 2

```
{'*    Mobile2'    }
{'*    SNR Vs test No'}

{'Table data size: 30 x 9'}
```

Block: 3

```
{'*    Mobile3'    }
{'*    SNR Vs test No'}

{'Table data size: 31 x 15'}
```

Block: 4

```
{'*    Mobile4'    }
{'*    SNR Vs test No'}

{'Table data size: 28 x 19'}
```

Block: 5

```
{'*    Mobile5'    }
{'*    SNR Vs test No'}

{'Table data size: 32 x 18'}
```

Block: 6

```
{'*    Mobile6'    }
{'*    SNR Vs test No'}

{'Table data size: 30 x 19'}
```

Block: 7

```
{'*    Mobile7'    }
{'*    SNR Vs test No'}

{'Table data size: 30 x 11'}
```

Block: 8

```
{'*    Mobile8'    }  
{'*    SNR Vs test No'}  
  
{'Table data size: 20 x 18'}
```

Block: 9

```
{'*    Indoor0'    }  
{'*    SNR Vs test No'}  
  
{'Table data size: 9 x 3'}
```

Block: 10

```
{'*    Indoor1'    }  
{'*    SNR Vs test No'}  
  
{'Table data size: 22 x 6'}
```

Block: 11

```
{'*    Indoor2'    }  
{'*    SNR Vs test No'}  
  
{'Table data size: 25 x 3'}
```

Block: 12

```
{'*    Indoor3'    }  
{'*    SNR Vs test No'}  
  
{'Table data size: 21 x 18'}
```

Block: 13

```
{'*    Outdoor1'    }  
{'*    SNR Vs test No'}  
  
{'Table data size: 20 x 18'}
```

Block: 14

```
{'*    Outdoor2'    }  
{'*    SNR Vs test No'}  
  
{'Table data size: 23 x 3'}
```

Block: 15

```
{'*    Outdoor3'    }
{'*    SNR Vs test No'}

{'Table data size: 22 x 18'}
```

Block: 16

```
{'*    Outdoor4'    }
{'*    SNR Vs test No'}

{'Table data size: 21 x 18'}
```

Block: 17

```
{'*    Outdoor5'    }
{'*    SNR Vs test No'}

{'Table data size: 18 x 5'}
```

关闭文本文件

```
fclose(fileID);
```

数据块总数

确定文件中的数据块数。

```
NumBlocks = Block-1
```

```
NumBlocks = 17
```

查看数值数据

使用短科学记数法显示其中一个数据块中的数值数据。

首先，存储当前命令行窗口输出显示格式。

```
user_format = get(0, 'format');
```

将显示格式更改为短科学记数法。

```
format shortE
```

显示第九个数据块的标题行和数值数据。

```
Block = 9;
disp(HeaderLines{Block});
```

```
{'*    Indoor0'    }
{'*    SNR Vs test No'}
```

```
fprintf('SNR    %d    %d\n',Data{Block,1}(1,2:end))
```



```
SNR      -7      -6
```

```
disp(Data{Block,1}(2:end,2:end));
```

```
9.0600e-07  6.7100e-07  
3.1700e-07  3.5400e-07  
2.8600e-07  1.9600e-07  
1.4800e-07  7.3400e-07  
3.9500e-08  9.6600e-07  
7.9600e-07  7.8300e-07  
4.0000e-07  8.8100e-07  
3.0100e-07  2.9700e-07
```

还原原始命令行窗口输出显示格式。

```
set(0, 'format', user_format);
```

另请参阅

textscan

详细信息

- “将文本文件中的混合数据块导入表或元胞数组” (第 2-19 页)

电子表格


- “导入电子表格” (第 3-2 页)
- “使用导入工具读取电子表格数据” (第 3-4 页)
- “将电子表格数据读入到表中” (第 3-7 页)
- “将数据写入 Excel 电子表格” (第 3-10 页)
- “定义表的导入选项” (第 3-12 页)

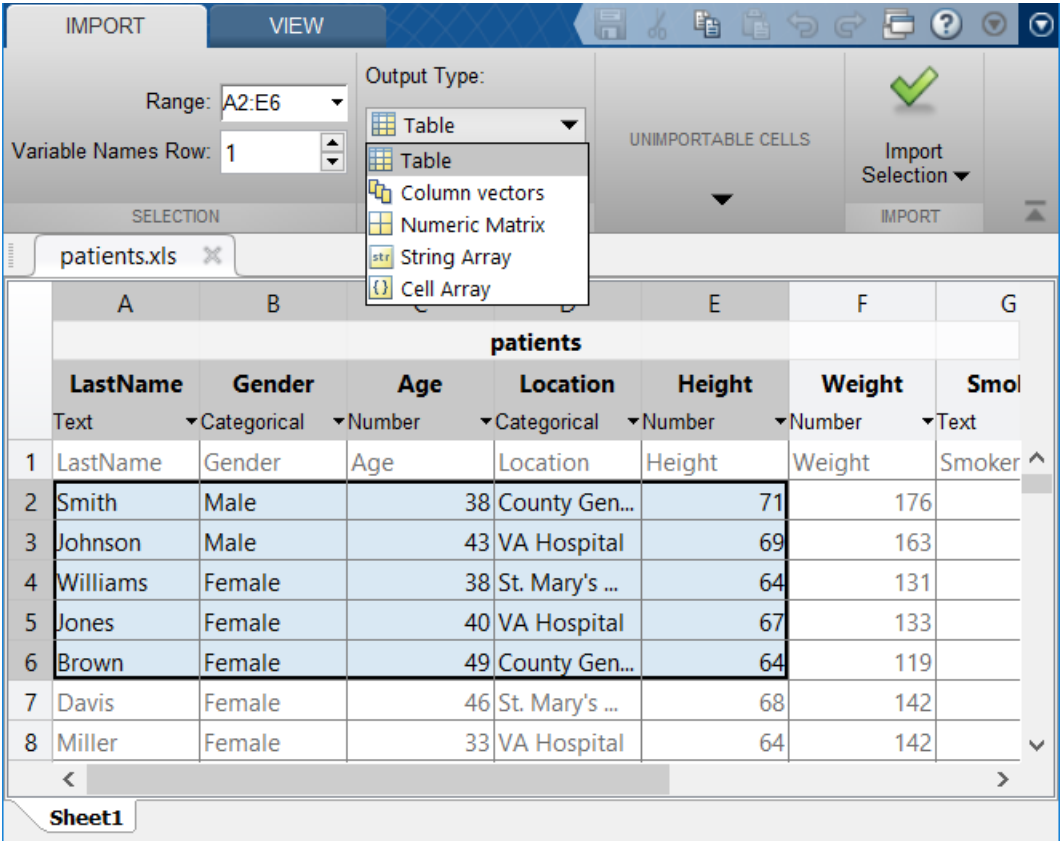
导入电子表格

电子表格通常包含数值和文本数据以及变量名称和行名称的混合，在 MATLAB 中的最佳表示形式是表。您可以使用**导入工具**或 `readtable` 函数将数据导入表中。

使用导入工具导入电子表格数据

导入工具允许您导入到表或其他数据类型中。例如，将示例电子表格文件 `patients.xls` 中的数据以表的形式读取到 MATLAB 中。使用**导入工具**打开该文件，并选择数据范围和输出类型等选项。然后，点击**导入**所

选内容按钮 ，将数据导入到 MATLAB 工作区中。



使用 readtable 导入电子表格数据

您也可以使用 `readtable` 函数通过读取文件名将电子表格数据读入表中，例如：

```
T = readtable('patients.xls');
```

您也可以通过指定范围参数来选择要导入的数据范围。例如，读取电子表格的前五行五列。请用 Excel 表示法将范围指定为 'A1:E5'。

```
T = readtable('patients.xls','Range','A1:E5')
```

T =

4×5 table

LastName	Gender	Age	Location	Height
{'Smith' }	{'Male' }	38	{'County General Hospital' }	71
{'Johnson'}	{'Male' }	43	{'VA Hospital' }	69
{'Williams'}	{'Female'}	38	{'St. Mary's Medical Center'}	64
{'Jones' }	{'Female'}	40	{'VA Hospital' }	67

将电子表格数据导入为其他数据类型

除了表之外，您还可以将电子表格数据作为时间表、数值矩阵、元胞数组或单独的列向量导入到 MATLAB 工作区中。根据您需要的数据类型，使用以下函数之一。

输出的数据类型	函数
时间表	readtimetable
数值矩阵	readmatrix
元胞数组	readcell
单独的列向量	readvars

另请参阅
导入工具 | readtable

详细信息

- “使用导入工具读取电子表格数据” （第 3-4 页）
- “将电子表格数据读入到表中” （第 3-7 页）
- “访问表中的数据”

使用导入工具读取电子表格数据

本节内容
“以交互方式选择数据” （第 3-4 页）
“从多个电子表格导入数据” （第 3-5 页）
“粘贴剪贴板中的数据” （第 3-5 页）

此示例说明如何使用导入工具将数据从电子表格导入工作区中，以及如何从剪贴板中导入数据。

以交互方式选择数据

在**主页**选项卡上的**变量**部分中，点击**导入数据** 。或者，在当前文件夹浏览器中，双击扩展名为.xls、.xlsx、.xlsb 或 .xlsm 的文件的名称。此时将打开导入工具。

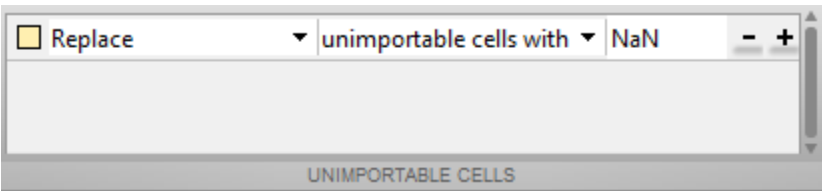
选择要导入的数据。例如，下图中的数据对应于三个列向量的数据。您可以在选项卡中编辑变量名称，并且可以为同一变量选择不连续的数据部分。

	A Station Number	B Temp Number	C Date Datetime
1	Station	Temp	Date
2	12	Replaced by:NaN	13
3	13	NaN	10/23/2013
4	14	97	12/1/2013

在**导入**选项卡的**输出类型**部分中，选择需要的数据导入方式。您选择的选项决定了所导入数据的数据类型。


选择的选项	导入数据的方式
列向量	将所选数据的每一列导入为单个 $m \times 1$ 向量。
数值矩阵	将所选数据导入为 $m \times n$ 数值数组。
字符串数组	将所选数据导入为 $m \times n$ 字符串数组。
元胞数组	将所选数据导入为可包含多种数据类型的元胞数组，例如数值数据和文本。
表	将所选数据导入为表。

如果选择将数据导入为矩阵或数值列向量，工具将突出显示工作表中的所有非数值数据。每个突出显示颜色对应于一个使数据适合数值数组的建议规则。例如，您可以用 NaN 替换非数值的值。此外，将鼠标置于单个单元格上时，可以看到数据的导入方式。



您可以添加、删除、重新排序或编辑规则，例如将替换值从 NaN 更改为其他值。所有规则仅应用于导入的数据，不会更改文件中的数据。只要范围包括非数值数据并且将导入到矩阵或数值列向量，就需要指定规则。

包含 #Error? 的所有单元格都对应于电子表格文件中的公式错误，例如除以零。导入工具将这些单元格视为非数值。

当点击**导入所选内容**按钮  时，导入工具将会在工作区中创建变量。

有关与导入工具交互的详细信息，请观看此视频。

从多个电子表格导入数据

如果计划对多个文件执行相同的导入操作，则可以从导入工具生成代码，以便更轻松地重复操作。在所有平台上，导入工具都可以生成一个程序脚本，您可以编辑和运行该脚本以导入文件。在安装有 Excel 软件的 Microsoft Windows 系统上，导入工具可以生成一个可为每个文件调用的函数。

例如，假设在当前文件夹中有一组名为 myfile01.xlsx 到 myfile25.xlsx 的电子表格，并且要从每个文件中的第一个工作表导入相同范围的数据 A2:G100。生成代码以导入整个文件集，如下所示：

- 1 在导入工具中打开一个文件。
- 2 从**导入所选内容**按钮中，选择**生成函数**。导入工具生成类似如下脚本的代码，并在编辑器中打开代码。

```
function data = importfile(workbookFile, sheetName, range)
%IMPORTFILE Import numeric data from a spreadsheet
...
```

- 3 保存函数。
- 4 在单独的程序文件或命令行中，创建一个 for 循环，以将每个电子表格中的数据导入到名为 myData 的元胞数组中：

```
numFiles = 25;
range = 'A2:G100';
sheet = 1;
myData = cell(1,numFiles);


for fileNum = 1:numFiles
    fileName = sprintf('myfile%02d.xlsx',fileNum);
    myData{fileNum} = importfile(fileName,sheet,range);
end
```

myData 中的每个元胞包含来自对应工作表的数据数组。例如，myData{1} 包含来自第一个文件 myfile01.xlsx 的数据。

粘贴剪贴板中的数据

除了以交互方式导入数据以外，您还可以将剪贴板中的电子表格数据粘贴到 MATLAB 中。

首先，在 Microsoft Excel 中选择并复制您的电子表格数据，然后使用以下方法之一：

- 在工作区浏览器标题栏中，点击 ，然后选择**粘贴**。

- 在变量编辑器中打开一个现有变量，右键点击，然后选择**粘贴 Excel 数据**。
- 调用 `uiimport -pastespecial`。

另请参阅

`readmatrix` | `readcell` | `readvars` | `readtable` | `detectImportOptions`

详细信息

- “定义表的导入选项”（第 3-12 页）
- “Read Spreadsheet Data into Array or Individual Variables”

将电子表格数据读入到表中

MATLAB® 中表示电子表格数据的最佳方法是在表中表示，表可以存储混合的数值和文本数据，以及变量和行名称。可采用交互方式或编程方式将数据读入到表中。要以交互方式选择数据，请在[主页](#)选项卡中的变量部分中点击**导入数据**。要以编程方式导入数据，请使用以下函数之一：

- `readtable` - 读取一个工作表。
- `spreadsheetDatastore` - 读取多个工作表或文件。

以下示例说明如何同时使用这两个函数以编程方式导入电子表格数据。样本数据 `airlinesmall_subset.xlsx` 包含与 1996 年到 2008 年之间的每一年——对应的工作表。工作表名称与年份对应，例如 2003。

读取工作表中的所有数据

调用 `readtable`，读取名为 2008 的工作表中的所有数据，然后仅显示前 10 个行和列。使用 `Sheet` 名称-值对组参数指定工作表名称。如果数据位于文件的第一个工作表中，则不需要指定 `Sheet`。

```
T = readtable('airlinesmall_subset.xlsx','Sheet','2008');
T(1:10,1:10)
```

ans=10×10 table

Year	Month	DayofMonth	DayOfWeek	DepTime	CRSDepTime	ArrTime	CRSArrTime	UniqueCarrier
2008	1	3	4	1012	1010	1136	1135	{'WN'}
2008	1	4	5	1303	1300	1411	1415	{'WN'}
2008	1	6	7	2134	2115	2242	2220	{'WN'}
2008	1	7	1	1734	1655	54	30	{'WN'}
2008	1	8	2	1750	1755	2018	2035	{'WN'}
2008	1	9	3	640	645	855	905	{'WN'}
2008	1	10	4	1943	1945	2039	2040	{'WN'}
2008	1	11	5	1303	1305	1401	1400	{'WN'}
2008	1	13	7	1226	1230	1415	1400	{'WN'}
2008	1	14	1	1337	1340	1623	1630	{'WN'}

读取特定工作表中的选定范围

通过指定范围 'A1:E11'，从名为 1996 的工作表中仅读取前 5 列中的 10 行数据。 `readtable` 函数将返回一个 10×5 的表。

```
T_selected = readtable('airlinesmall_subset.xlsx','Sheet','1996','Range','A1:E11')
```

T_selected=10×5 table

Year	Month	DayofMonth	DayOfWeek	DepTime
1996	1	18	4	2117
1996	1	12	5	1252
1996	1	16	2	1441
1996	1	1	1	2258
1996	1	4	4	1814
1996	1	31	3	1822
1996	1	18	4	729
1996	1	26	5	1704

```
1996 1 11 4 1858
1996 1 7 7 2100
```

将变量转换为日期时间、持续时间或分类

在导入过程中，`readtable` 会自动检测变量的数据类型。但如果数据中包含非标准的日期、持续时间或重复的标签，则可以将这些变量转换为其正确的数据类型。通过将变量转换为其正确的数据类型，可以执行高效的计算和比较，并改善内存的使用。例如，将变量 `Year`、`Month` 和 `DayofMonth` 表示为一个 `datetime` 变量，将 `UniqueCarrier` 表示为 `categorical`，将 `ArrDelay` 表示为以分钟为单位的 `duration`。

```
data = T(:,{'Year','Month','DayofMonth','UniqueCarrier','ArrDelay'});
data.Date = datetime(data.Year,data.Month,data.DayofMonth);
data.UniqueCarrier = categorical(data.UniqueCarrier);
data.ArrDelay = minutes(data.ArrDelay);
```

查找当年延迟最长的一天，然后显示该日期。

```
ind = find(data.ArrDelay == max(data.ArrDelay));
data.Date(ind)
```

```
ans = datetime
    07-Apr-2008
```

读取电子表格文件中的所有工作表

数据存储适合处理分布在多个工作表或多个电子表格文件中的任意大的数据量。可通过数据存储执行数据导入和数据处理。

从 `airlinesmall_subset.xlsx` 中的工作表集合创建一个数据存储，选择要导入的变量，然后预览数据。

```
ds = spreadsheetDatastore('airlinesmall_subset.xlsx');
ds.SelectedVariableNames = {'Year','Month','DayofMonth','UniqueCarrier','ArrDelay'};
preview(ds)
```

```
ans=8×5 table
    Year  Month  DayofMonth  UniqueCarrier  ArrDelay
    _____  _____  _____  _____  _____
    1996     1     18      {'HP'}         6
    1996     1     12      {'HP'}        11
    1996     1     16      {'HP'}       -13
    1996     1      1      {'HP'}         1
    1996     1      4      {'US'}        -9
    1996     1     31      {'US'}         9
    1996     1     18      {'US'}        -2
    1996     1     26      {'NW'}       -10
```

在导入数据之前，可以指定要使用的数据类型。在此示例中，以分类变量的形式导入 `UniqueCarrier`。

```
ds.SelectedVariableTypes(4) = {'categorical'};
```

使用 `readall` 或 `read` 函数导入数据。`readall` 函数要求所有数据都能放入内存中，样本数据符合此要求。导入后，计算此数据集的最大到港延误。

```
alldata = readall(ds);  
max(alldata.ArrDelay)/60
```

```
ans = 15.2333
```

对于大型数据集，请使用 `read` 函数导入文件的部分内容。有关详细信息，请参阅[读取电子表格文件集合或序列](#)。

另请参阅

`readtable` | `spreadsheetDatastore`

详细信息

- “使用导入工具读取电子表格数据”（第 3-4 页）
- “Read Spreadsheet Data into Array or Individual Variables”
- “Read Collection or Sequence of Spreadsheet Files”

将数据写入 Excel 电子表格

本节内容
“将表格数据写入到电子表格文件” （第 3-10 页）
“将数值和文本数据写入到电子表格文件” （第 3-10 页）
“添加新工作表时禁用警告” （第 3-11 页）
“对 Excel 文件中的单元格设置格式” （第 3-11 页）

将表格数据写入到电子表格文件

要将工作区中的表导出到 Microsoft® Excel® 电子表格文件中，请使用 `writetable` 函数。您可以将工作区中的数据导出到文件中的任何工作表，以及导出到该工作表中的任何位置。默认情况下，`writetable` 将表数据写入到文件中的第一张工作表，并从单元格 A1 处开始。

例如，创建一个包含列向数据的样本表，并显示前五。

```
load patients.mat
T = table(LastName, Age, Weight, Smoker);
T(1:5,:)

ans=5x4 table
    LastName    Age    Weight    Smoker
    _____  _  _  _
    {'Smith' }  38    176    true
    {'Johnson'} 43    163    false
    {'Williams'} 38    131    false
    {'Jones' }  40    133    false
    {'Brown' }  49    119    false
```

将表 `T` 写入到名为 `patientdata.xlsx` 新电子表格文件中的第一张工作表，并从单元格 D1 处开始。要指定您想要写入到的工作表部分，请使用 `Range` 名称-值对组参数。默认情况下，`writetable` 将表变量名称写入为电子表格文件中的列标题。

```
filename = 'patientdata.xlsx';
writetable(T,filename,'Sheet',1,'Range','D1')
```

将不带变量名称的表 `T` 写入名为 'MyNewSheet' 的新工作表。要写入不带变量名称的数据，请将名称-值对组 `WriteVariableNames` 指定为 `false`。

```
writetable(T,filename,'Sheet','MyNewSheet','WriteVariableNames',false);
```

将数值和文本数据写入到电子表格文件

要将数值数组和元胞数组导出到 Microsoft Excel 电子表格文件，请使用 `writematrix` 或 `writecell` 函数。您可以将单个数值工作区变量和文本工作区变量中的数据导出到文件中的任何工作表，以及导出到工作表中的任何位置。默认情况下，导入函数将矩阵数据写入到文件中的第一张工作表，并从单元格 A1 处开始。

例如，创建一个由数值数据组成的样本数组 `A`，以及一个由文本和数值数据组成的样本元胞数组 `C`。

```
A = magic(5)
C = {'Time', 'Temp'; 12 98; 13 'x'; 14 97}
```

```
A =
```

```
17 24 1 8 15
23 5 7 14 16
4 6 13 20 22
10 12 19 21 3
11 18 25 2 9
```

```
C =
```

```
'Time' 'Temp'
[ 12] [ 98]
[ 13] 'x'
[ 14] [ 97]
```

将数组 A 写入名为 `testdata.xlsx` 的新电子表格文件中第一张工作表上的 5×5 矩形区域 E1:I5。

```
filename = 'testdata.xlsx';
writematrix(A,filename,'Sheet',1,'Range','E1:I5')
```

将元胞数组 C 写入到名为 `Temperatures` 的工作表上的一个矩形区域（从单元格 B2 处开始）。可以仅使用第一个单元格指定范围。

```
writecell(C,filename,'Sheet','Temperatures','Range','B2');
```

`writecell` 将会显示警告，因为工作表 `Temperatures` 之前不存在，但您可以禁用此警告。

添加新工作表时禁用警告

如果文件中不存在目标工作表，则 `writetable` 和 `writecell` 函数将会显示以下警告：

Warning: Added specified worksheet.

有关如何隐藏警告消息的信息，请参阅“[隐藏警告](#)”。

对 Excel 文件中的单元格设置格式

要将数据写入到 Windows 系统上具有自定义格式（例如字体或颜色）的 Excel 文件，请直接使用 `actxserver` 而不是 `writetable`、`writetimetable`、`writematrix` 或 `writecell` 访问 COM 服务器。例如，技术解决方案 1-QLD4K 使用 `actxserver` 在 MATLAB 和 Excel 之间建立连接，将数据写入工作表，以及指定单元格的格式。

有关详细信息，请参阅“[Get Started with COM](#)”。

另请参阅

`writematrix` | `writecell` | `writetable`

定义表的导入选项

通常情况下，可以使用 `readtable` 函数来导入表。但有时导入表格数据需要对导入过程施加更多控制。例如，您可能希望选择变量，以导入或处理缺失数据或存在导致错误的数据的行。要控制导入过程，您可以创建导入选项对象。该对象具有您可以根据自己的导入需求进行调整的属性。

创建导入选项

要为样本数据集 `airlinesmall.csv` 创建导入选项对象，请使用 `detectImportOptions` 函数。`detectImportOptions` 函数将为此文本文件创建一个 `DelimitedTextImportOptions` 对象。有关导入选项对象的完整属性列表，请参阅 `detectImportOptions` 参考页。

```
opts = detectImportOptions('airlinesmall.csv');
```

自定义表级别的导入选项

您可以调整导入选项对象所包含的属性，以便控制导入过程。一些属性适用于整个表，而另一些则适用于特定的变量。影响整个表的属性包括用于管理导致错误的的数据或缺失数据的规则。例如，通过将 `ImportErrorRule` 设为 `'omitrow'`，删除包含会导致导入错误的的数据的行。通过将 `MissingRule` 设为 `'fill'` 替换缺失值。`FillValue` 属性值决定了用来替换缺失值的值。例如，您可以用 `NaN` 来替换缺失值。

```
opts.ImportErrorRule = 'omitrow';
opts.MissingRule = 'fill';
```

自定义变量级别的导入选项

要获取和设置特定变量的选项，请使用 `getvaropts`、`setvartype` 和 `setvaropts` 函数。例如，使用 `getvaropts` 函数查看名为 `FlightNum`、`Origin`、`Dest` 和 `ArrDelay` 的变量的当前选项。

```
getvaropts(opts,{'FlightNum','Origin','Dest','ArrDelay'});
```

使用 `setvartype` 函数更改变量的数据类型：

- 由于变量 `FlightNum` 中的值为航班标识符而不是数值，因此需要将其数据类型更改为 `char`。
- 由于变量 `Origin` 和 `Dest` 指定了重复文本值的有限集合，因此需要将其数据类型更改为 `categorical`。

```
opts = setvartype(opts,{'FlightNum','Origin','Dest','ArrDelay'},...
    {'char','categorical','categorical','single'});
```

使用 `setvaropts` 函数更改其他属性：

- 对于 `FlightNum` 变量，通过将 `WhiteSpaceRule` 属性设为 `trimleading`，删除文本中的任何前导空白。
- 对于 `ArrDelay` 变量，通过设置 `TreatAsMissing` 属性，将包含 0 或 NA 的字段替换为在 `FillValue` 属性中指定的值。

```
opts = setvaropts(opts,'FlightNum','WhiteSpaceRule','trimleading');
opts = setvaropts(opts,'ArrDelay','TreatAsMissing',{0,'NA'});
```

导入表

指定要获取的变量，使用 `readtable` 导入这些变量，并显示表的前 8 行。

```
opts.SelectedVariableNames = {'FlightNum','Origin','Dest','ArrDelay'};
T = readtable('airlinesmall.csv',opts);
T(1:8,:)
```

ans=8×4 table

FlightNum	Origin	Dest	ArrDelay
{'1503'}	LAX	SJC	8
{'1550'}	SJC	BUR	8
{'1589'}	SAN	SMF	21
{'1655'}	BUR	SJC	13
{'1702'}	SMF	LAX	4
{'1729'}	LAX	SJC	59
{'1763'}	SAN	SFO	3
{'1800'}	SEA	LAX	11

另请参阅

[detectImportOptions](#) | [getvaropts](#) | [setvaropts](#) | [setvartype](#) | [readtable](#) | [SpreadsheetImportOptions](#) | [DelimitedTextImportOptions](#) | [readmatrix](#) | [readcell](#) | [readvars](#)

详细信息

- “使用导入工具读取电子表格数据” (第 3-4 页)
- “将电子表格数据读入到表中” (第 3-7 页)

低级文件 I/O

- “使用低级 I/O 导入文本数据文件” (第 4-2 页)
- “通过低级 I/O 导入二进制数据” (第 4-8 页)
- “使用低级 I/O 导出到文本数据文件” (第 4-13 页)

使用低级 I/O 导入文本数据文件

本节内容
“概述”（第 4-2 页）
“读取格式化模式的数据”（第 4-2 页）
“逐行读取数据”（第 4-4 页）
“检测文件末尾 (EOF)”（第 4-5 页）
“使用不同的字符编码打开文件”（第 4-7 页）

概述

通过低级文件 I/O 函数，可以最大程度地控制文件数据的读取和写入。但是，相对于更易于使用的高级函数，例如 `importdata`，这些低级函数要求指定更为详细的文件信息。有关读取文本文件的高级函数的详细信息，请参阅“导入文本文件”（第 2-2 页）。

如果高级函数无法导入数据，请使用下列函数之一：

- `fscanf`，读取文本或 ASCII 文件（即可以在文本编辑器中查看的文件）中的格式化数据。有关详细信息，请参阅“读取格式化模式的数据”（第 4-2 页）。
- `fgetl` 和 `fgets`，一次读取文件中的一行，其中每一行通过换行符分隔。有关详细信息，请参阅“逐行读取数据”（第 4-4 页）。
- `fread`，读取从相应字节或位级开始的数据流。有关详细信息，请参阅“通过低级 I/O 导入二进制数据”（第 4-8 页）。

有关其他信息，请参阅：

- “检测文件末尾 (EOF)”（第 4-5 页）
- “使用不同的字符编码打开文件”（第 4-7 页）

注意 低级文件 I/O 函数基于 ANSI® 标准 C 库中的函数。但是，MATLAB 包括向量化版本的函数，通过最少的控制循环在数组中读取和写入数据。

读取格式化模式的数据

要导入 `importdata` 和 `textscan` 无法读取的文本文件，可考虑使用 `fscanf`。`fscanf` 函数要求对文件的格式进行描述，但包括用于描述格式的众多选项。

例如，按如下所示创建文本文件 `my meas.dat`。`my meas.dat` 中的数据包括重复的时间、日期和测量数据集。标题文本包括测量数据集数 N：

```
Measurement Data
N=3

12:00:00
01-Jan-1977
4.21 6.55 6.78 6.55
9.15 0.35 7.57 NaN
7.92 8.49 7.43 7.06
9.59 9.33 3.92 0.31
```

```

09:10:02
23-Aug-1990
2.76 6.94 4.38 1.86
0.46 3.17 NaN 4.89
0.97 9.50 7.65 4.45
8.23 0.34 7.95 6.46
15:03:40
15-Apr-2003
7.09 6.55 9.59 7.51
7.54 1.62 3.40 2.55
NaN 1.19 5.85 5.05
6.79 4.98 2.23 6.99

```

打开文件

与任何低级 I/O 函数一样，在读取前，使用 `fopen` 打开文件，并获取文件标识符。默认情况下，`fopen` 使用 'r' 权限打开文件进行读取访问。

处理完文件后，使用 `fclose(fid)` 关闭文件。

描述数据

使用格式设定符描述文件中的数据，例如 '%s' 表示文本，'%d' 表示整数，或者 '%f' 表示浮点数。（有关设定符的完整列表，请参阅 `fscanf` 参考页。）

要跳过文件中的字面字符，请在格式说明中包括这些字符。要跳过数据字段，请在设定符中使用星号 ('*')。

例如，`my meas.dat` 的标题行如下所示：

```

Measurement Data % skip the first 2 words, go to next line: %*s %*s\n
N=3              % ignore 'N=', read integer: N=%d\n
                % go to next line: \n
12:00:00
01-Jan-1977
4.21 6.55 6.78 6.55
...

```

读取标题并返回 N 的值：

```
N = fscanf(fid, '%*s %*s\nN=%d\n\n', 1);
```

指定要读取的值数

默认情况下，`fscanf` 会重复应用您的格式描述，直至它无法匹配数据，或者它到达文件的末尾。

您也可以指定要读取的值数，这样 `fscanf` 不会尝试读取整个文件。例如，在 `my meas.dat` 中，每个测量数据集包含固定数目的行和列：

```

measrows = 4;
meascols = 4;
meas = fscanf(fid, '%f', [measrows, meascols]);

```

在工作区中创建变量

有多种方式可将 `my meas.dat` 存储在 MATLAB 工作区中。在此例中，将值读取到一个结构体中。该结构体的每个元素都有三个字段：`mtime`、`mdate` 和 `meas`。

注意 `fscanf` 按列顺序使用数值填充数组。要使输出数组与文件中数值数据的方向匹配，请对数组进行转置操作。

```
filename = 'my meas.dat';
measrows = 4;
meascols = 4;

% open the file
fid = fopen(filename);

% read the file headers, find N (one value)
N = fscanf(fid, '%*s %*s\nN=%d\n\n', 1);

% read each set of measurements
for n = 1:N
    mystruct(n).mtime = fscanf(fid, '%s', 1);
    mystruct(n).mdate = fscanf(fid, '%s', 1);

    % fscanf fills the array in column order,
    % so transpose the results
    mystruct(n).meas = ...
        fscanf(fid, '%f', [measrows, meascols]);
end

% close the file
fclose(fid);
```

逐行读取数据

MATLAB 提供了下列两个函数分别用于读取文件中的行和将这些行存储为字符向量：`fgetl` 和 `fgets`。`fgets` 函数会将数据行和换行符都复制到输出，但 `fgetl` 不会。

以下示例使用 `fgetl` 按一次读取一行的方式读取整个文件。函数 `litcount` 确定给定的字符序列 (literal) 是否将显示在每一行中。如果是，则该函数将输出整行，并在行前加上该文字在行中出现的次数。

```
function y = litcount(filename, literal)
% Count the number of times a given literal appears in each line.

fid = fopen(filename);
y = 0;
tline = fgetl(fid);
while ischar(tline)
    matches = strfind(tline, literal);
    num = length(matches);
    if num > 0
        y = y + num;
        fprintf(1, '%d:%s\n', num, tline);
    end
    tline = fgetl(fid);
end
fclose(fid);
```

创建一个名为 `badpoem` 的输入数据文件：

```
Oranges and lemons,
Pineapples and tea.
```

Orangutans and monkeys,
Dragonflys or fleas.

要确定 'an' 显示在此文件中的次数，请调用 `litcount`：

```
litcount('badpoem','an')
```

这将返回：

```
2: Oranges and lemons,  
1: Pineapples and tea.  
3: Orangutans and monkeys,  
ans =  
    6
```

检测文件末尾 (EOF)

当一次读取数据的一部分时，可以使用 `feof` 检查是否已到达文件的末尾。当文件指针位于文件末尾时，`feof` 返回值 1。否则，将返回 0。

注意 打开一个空文件不会将文件位置指示符移到文件末尾。读取操作以及 `fseek` 和 `frewind` 函数可移动文件位置指示符。

使用 `feof` 测试 EOF

当使用 `textscan`、`fscanf` 或 `fread` 一次读取数据的一部分时，请使用 `feof` 检查是否已到达文件的末尾。

例如，假定所假设的文件 `my meas.dat` 具有下列形式，并且没有有关测量集数的信息。将数据读取到具有表示 `mtime`、`mdate` 和 `meas` 的字段的结构体：

```
12:00:00  
01-Jan-1977  
4.21 6.55 6.78 6.55  
9.15 0.35 7.57 NaN  
7.92 8.49 7.43 7.06  
9.59 9.33 3.92 0.31  
09:10:02  
23-Aug-1990  
2.76 6.94 4.38 1.86  
0.46 3.17 NaN 4.89  
0.97 9.50 7.65 4.45  
8.23 0.34 7.95 6.46
```

读取文件：

```
filename = 'my meas.dat';  
measrows = 4;  
meascols = 4;  
  
% open the file  
fid = fopen(filename);  
  
% make sure the file is not empty  
finfo = dir(filename);
```

```

fsize = finfo.bytes;

if fsize > 0

    % read the file
    block = 1;
    while ~feof(fid)
        mystruct(block).mtime = fscanf(fid, '%s', 1);
        mystruct(block).mdate = fscanf(fid, '%s', 1);

        % fscanf fills the array in column order,
        % so transpose the results
        mystruct(block).meas = ...
            fscanf(fid, '%f', [measrows, meascols]);

        block = block + 1;
    end

end

% close the file
fclose(fid);

```

使用 fgetl 和 fgets 测试 EOF

如果在控制循环中使用 `fgetl` 或 `fgets`，`feof` 并不始终是检测文件末尾的最佳方式。作为备选方法，可以考虑检查 `fgetl` 或 `fgets` 返回的值是否为字符向量。

例如，“逐行读取数据”（第 4-4 页）中描述的 `litcount` 函数包括下列 `while` 循环和 `fgetl` 调用：

```

y = 0;
tline = fgetl(fid);
while ischar(tline)
    matches = strfind(tline, literal);
    num = length(matches);
    if num > 0
        y = y + num;
        fprintf(1, '%d: %s\n', num, tline);
    end
    tline = fgetl(fid);
end

```

此方法比检测 `~feof(fid)` 更稳健，原因如下：

- 如果 `fgetl` 或 `fgets` 找到数据，则它们返回字符向量。否则，它们返回数值 (-1)。
- 每次读取操作之后，`fgetl` 和 `fgets` 都会检查文件中的下一个字符是否为文件结尾标记。因此，这些函数有时会在返回值 -1 之前设置文件结尾指示符。例如，假设以下包含三行的文本文件。前两行的每一行都以换行符结尾，第三行仅包含文件结尾标记：

```

123
456

```

对 `fgetl` 的三个连续调用生成以下结果：

```

t1 = fgetl(fid); % t1 = '123', feof(fid) = false
t2 = fgetl(fid); % t2 = '456', feof(fid) = true
t3 = fgetl(fid); % t3 = -1, feof(fid) = true

```

此行为不符合相关 C 语言函数的 ANSI 设定。

使用不同的字符编码打开文件

编码方案支持特定字母所需的字符，例如日语或欧洲语言字符。常见的编码方案包括 US-ASCII 或 UTF-8。

如果在打开文件进行读取时没有指定编码方案，**fopen** 会使用自动字符集检测来确定编码。如果在打开文件进行写入时没有指定编码方案，**fopen** 默认使用 UTF-8，以便在所有平台和区域设置之间提供互操作性，而不会丢失或损坏数据。

要确定默认编码方案，请打开文件，并再次使用以下语法调用 **fopen**：

```
[filename, permission, machineformat, encoding] = fopen(fid);
```

如果在打开文件时指定了编码方案，以下函数将应用该方案：**fscanf**、**fprintf**、**fgetl**、**fgets**、**fread** 和 **fwrite**。

有关支持的编码方案的完整列表以及用于指定编码的语法，请参阅 **fopen** 参考页。

通过低级 I/O 导入二进制数据

本节内容
“用于导入数据的低级函数” (第 4-8 页)
“读取文件中的二进制数据” (第 4-8 页)
“读取文件的一部分” (第 4-10 页)
“读取在其他系统上创建的文件” (第 4-12 页)

用于导入数据的低级函数

通过低级文件 I/O 函数，可以最大程度地直接控制对文件数据的读取和写入。但是，相对于更易于使用的高级函数，这些低级函数要求指定更为详细的文件信息。有关高级函数及其支持的文件格式的完整列表，请参阅“支持的导入和导出的文件格式” (第 1-2 页)。

如果高级函数无法导入数据，请使用下列函数之一：

- **fscanf**，读取文本或 ASCII 文件（即可以在文本编辑器中查看的文件）中的格式化数据。有关详细信息，请参阅“读取格式化模式的数据” (第 4-2 页)。
- **fgetl** 和 **fgets**，一次读取文件中的一行，其中每一行通过换行符分隔。有关详细信息，请参阅“逐行读取数据” (第 4-4 页)。
- **fread**，读取从相应字节或位级开始的数据流。有关详细信息，请参阅“读取文件中的二进制数据” (第 4-8 页)。

注意 低级文件 I/O 函数基于 ANSI 标准 C 库中的函数。但是，MATLAB 包括向量化版本的函数，通过最少的控制循环在数组中读取和写入数据。

读取文件中的二进制数据

与任何低级 I/O 函数一样，在导入前，使用 **fopen** 打开文件，并获取文件标识符。处理完文件后，使用 **fclose(fileID)** 关闭文件。

默认情况下，**fread** 每次读取文件的 1 个字节，并将每个字节解释为一个 8 位无符号整数 (**uint8**)。**fread** 创建一个列向量，文件中每个字节对应一个元素。列向量中的值属于 **double** 类。

例如，按以下方式创建文件 **nine.bin**：

```
fid = fopen('nine.bin','w');
fwrite(fid, [1:9]);
fclose(fid);
```

要将该文件中的所有数据读取到一个 **double** 类的 9×1 列向量中：

```
fid = fopen('nine.bin');
col9 = fread(fid);
fclose(fid);
```

更改数组的维度

默认情况下，**fread** 将文件中的所有值读取到一个列向量中。但是，您可以指定要读取的值的数量，或者描述一个二维输出矩阵。

例如，要读取前面示例中所述的 `nine.bin`：

```
fid = fopen('nine.bin');

% Read only the first six values
col6 = fread(fid, 6);

% Return to the beginning of the file
frewind(fid);

% Read first four values into a 2-by-2 matrix
frewind(fid);
two_dim4 = fread(fid, [2, 2]);

% Read into a matrix with 3 rows and
% unspecified number of columns
frewind(fid);
two_dim9 = fread(fid, [3, inf]);

% Close the file
fclose(fid);
```

描述输入值

如果文件中的值不是 8 位无符号整数，则需指定值的大小。

例如，使用以下双精度值创建文件 `fpoint.bin`：

```
myvals = [pi, 42, 1/3];

fid = fopen('fpoint.bin','w');
fwrite(fid, myvals, 'double');
fclose(fid);
```

读取文件：

```
fid = fopen('fpoint.bin');

% read, and transpose so samevals = myvals
samevals = fread(fid, 'double');

fclose(fid);
```

有关精度描述的完整列表，请参阅 `fread` 函数参考页。

节省内存

默认情况下，`fread` 创建一个 `double` 类的数组。在数组中存储双精度值比存储字符、整数或单精度值需要更多的内存。

要减少存储数据所需的内存量，请使用以下方法之一指定数组的类：

- 用星号 (*) 匹配输入值的类。例如，要将单精度值读入一个 `single` 类的数组，请使用命令：

```
mydata = fread(fid, '*single')
```
- 用 '=>' 符号将输入值映射到一个新类。例如，要将 `uint8` 值读入一个 `uint16` 数组，请使用命令：

```
mydata = fread(fid, 'uint8=>uint16')
```

有关精度描述的完整列表，请参阅 **fread** 函数参考页。

读取文件的一部分

MATLAB 低级函数包括几个用于读取文件中二进制数据部分的选项：

- 一次读取指定数量的值，如“更改数组的维度”（第 4-8 页）中所述。考虑将此方法与“检测文件末尾”（第 4-10 页）结合使用。
- 移动到文件中的特定位置以开始读取。有关详细信息，请参阅“在文件中移动”（第 4-10 页）。
- 在每次读取元素后跳过一定数量的字节或位。有关示例，请参阅“Write and Read Complex Numbers”。

检测文件末尾

打开文件时，MATLAB 创建一个指针，指示文件中的当前位置。

注意 打开一个空文件不会将文件位置指示符移到文件末尾。读取操作以及 **fseek** 和 **frewind** 函数可移动文件位置指示符。

使用 **feof** 函数检查是否已到达文件的末尾。当文件指针在文件的末尾时，**feof** 返回值 1。否则，将返回 0。

例如，分几部分来读取大文件：

```
filename = 'largedata.dat';    % hypothetical file
segsz = 10000;
```

```
fid = fopen(filename);
```

```
while ~feof(fid)
    currData = fread(fid, segsz);
    if ~isempty(currData)
        disp('Current Data:');
        disp(currData);
    end
end
```

```
fclose(fid);
```

在文件中移动

要读取或写入数据的所选部分，请将文件位置指示符移动到文件中的任意位置。例如，使用以下语法调用 **fseek**

```
fseek(fid,offset,origin);
```

其中：

- **fid** 是从 **fopen** 获取的文件标识符。
- **offset** 是一个以字节指定的正或负偏移值。
- **origin** 指定定位所采用的基本位置：

'bof'	文件的开头
'cof'	文件中的当前位置
'eof'	文件的结尾

或者，要轻松移至文件的开头：

```
frewind(fid);
```

使用 `ftell` 查找在给定文件中的当前位置。`ftell` 返回从文件开头算起的字节数。

例如，创建文件 `five.bin`：

```
A = 1:5;
fid = fopen('five.bin','w');
fwrite(fid,A,'short');
fclose(fid);
```

因为调用 `fwrite` 会指定 `short` 格式，所以 `A` 的每个元素在 `five.bin` 中使用两个存储字节。

重新打开 `five.bin` 以便于读取：

```
fid = fopen('five.bin','r');
```

将文件位置指示符从文件开头向前移动 6 个字节：

```
status = fseek(fid,6,'bof');
```

File Position	bof	1	2	3	4	5	6	7	8	9	10	eof
File Contents		0	1	0	2	0	3	0	4	0	5	
File Position Indicator								↑				

读取下一个元素：

```
four = fread(fid,1,'short');
```

读取操作使文件位置指示符前进。要确定当前文件位置指示符，请调用 `ftell`：

```
position = ftell(fid)
```

```
position =
      8
```

File Position	bof	1	2	3	4	5	6	7	8	9	10	eof
File Contents		0	1	0	2	0	3	0	4	0	5	
File Position Indicator									↑			

要将文件位置指示符向后移动 4 个字节，请再次调用 `fseek`：

```
status = fseek(fid,-4,'cof');
```

File Position	bof	1	2	3	4	5	6	7	8	9	10	eof
File Contents		0	1	0	2	0	3	0	4	0	5	
File Position Indicator					↑							

读取下一个值：

```
three = fread(fid,1,'short');
```

读取在其他系统上创建的文件

不同的操作系统在字节级别或位级别以不同的方式存储信息：

- Big-endian 系统从内存中的最大地址开始存储字节（即以大端开头）。
- Little-endian 系统从最小地址（小端）开始存储字节。

Windows 系统使用 little-endian 字节顺序，而 UNIX 系统使用 big-endian 字节顺序。

要读取在相反字节序系统上创建的文件，请指定用于创建该文件的字节顺序。您可以在打开文件的调用中指定顺序，也可以在读取文件的调用中指定顺序。

例如，假设在 little-endian 系统上创建了一个名为 **little.bin** 的双精度值文件。要在 big-endian 系统上读取此文件，请使用以下命令之一（或两者）：

- 打开文件

```
fid = fopen('little.bin', 'r', 'l')
```

- 读取文件

```
mydata = fread(fid, 'double', 'l')
```

其中 'l' 表示 little-endian 顺序。

如果不确定系统使用哪种字节顺序，请调用 **computer** 函数：

```
[cinfo, maxsize, ordering] = computer
```

返回的 **ordering** 为 'l' 表示 little-endian 系统，为 'B' 表示 big-endian 系统。

使用低级 I/O 导出到文本数据文件

本节内容
“使用 fprintf 写入到文本文件” (第 4-13 页)
“追加到或覆盖现有的文本文件” (第 4-14 页)
“使用不同的字符编码打开文件” (第 4-16 页)

使用 fprintf 写入到文本文件

以下示例演示如何使用低级 fprintf 函数创建文本文件，其中混合了数值、字符数据和非矩形文件。

fprintf 基于其在 ANSI® 标准 C 库中的名称。但是，MATLAB® 使用向量化版本的 fprintf，后者使用最少的控制循环写入数组中的数据。

打开文件

创建一个包含两行的样本矩阵 y。

```
x = 0:0.1:1;
y = [x; exp(x)];
```

使用 fopen 打开文件以进行写入并获取文件标识符 fileID。默认情况下，fopen 以只读方式打开文件，所以您必须指定写入或追加权限，例如 'w' 或 'a'。

```
fileID = fopen('exptable.txt','w');
```

写入到文件

使用 fprintf 函数写入标题，并后跟一个空行。要移动到文件中的新行，请使用 '\n'。

```
fprintf(fileID, 'Exponential Function\n\n');
```

注意：某些 Windows® 文本编辑器，包括 Microsoft® 记事本，需要换行符序列 '\r\n' 而不是 '\n'。但是，'\n' 满足 Microsoft Word 或 WordPad 的要求。

按列顺序将值写入到 y，这样两个值显示在文件的每一行中。fprintf 根据您的设定将数组输入中的数值或字符转换为文本。指定 '%f' 可输出浮点数。

```
fprintf(fileID, '%f %f\n',y);
```

其他常见的转换设定符包括表示整数的 '%d' 或表示字符的 '%s'。fprintf 重新应用转换信息以按列顺序循环使用输入数组的所有值。

写入完成后，使用 fclose 关闭文件。

```
fclose(fileID);
```

使用 type 函数查看文件的内容。

```
type exptable.txt

Exponential Function

0.000000 1.000000
```

```
0.100000 1.105171
0.200000 1.221403
0.300000 1.349859
0.400000 1.491825
0.500000 1.648721
0.600000 1.822119
0.700000 2.013753
0.800000 2.225541
0.900000 2.459603
1.000000 2.718282
```

其他格式化选项

可以选择在对 `fprintf` 的调用中包括其他信息以描述字段宽度、精度或者输出值的顺序。例如，指定指数表中的字段宽度和小数点右侧的位数。

```
fileID = fopen('exptable_new.txt', 'w');

fprintf(fileID, 'Exponential Function\n\n');
fprintf(fileID, '%6.2f %12.8f\n', y);

fclose(fileID);
```

查看文件的内容。

type `exptable_new.txt`

Exponential Function

```
0.00 1.000000000
0.10 1.10517092
0.20 1.22140276
0.30 1.34985881
0.40 1.49182470
0.50 1.64872127
0.60 1.82211880
0.70 2.01375271
0.80 2.22554093
0.90 2.45960311
1.00 2.71828183
```

追加到或覆盖现有的文本文件

以下示例演示如何将值追加到现有的文本文件、重写整个文件，以及仅覆盖文件的一部分。

默认情况下，`fopen` 使用只读访问权限打开文件。要更改文件访问权限的类型，请在对 `fopen` 的调用中使用权限设定符。可能的权限设定符包括：

- `'r'`，用于读取
- `'w'`，用于写入、放弃文件的任何现有内容
- `'a'`，用于追加到现有文件的结尾

要打开文件进行读取和写入或追加，请将一个加号附加到权限，例如 `'w+'` 或 `'a+'`。如果打开文件进行读取和写入，则必须在读操作和写操作之间调用 `fseek` 或 `frewind`。

追加到现有的文本文件

创建一个名为 `changing.txt` 的文件。

```
fileID = fopen('changing.txt','w');
fmt = '%5d %5d %5d %5d\n';
fprintf(fileID,fmt, magic(4));
fclose(fileID);
```

`changing.txt` 的当前内容为：

```
16 5 9 4
```

```
2 11 7 14
```

```
3 10 6 15
```

```
13 8 12 1
```

打开有权进行追加的文件。

```
fileID = fopen('changing.txt','a');
```

在文件末尾写入值 `[55 55 55 55]`：

```
fprintf(fileID,fmt,[55 55 55 55]);
```

关闭文件。

```
fclose(fileID);
```

使用 `type` 函数查看文件的内容。

```
type changing.txt
```

```
16   5   9   4
 2  11   7  14
 3  10   6  15
13   8  12   1
55  55  55  55
```

覆盖整个文本文件

文本文件由一系列连续字符组成，包括换行符。要将文件的某行替换为不同数量的字符，必须重写要更改的行和该文件中的所有后续行。

将 `changing.txt` 的第一行替换为更长的描述性文本。由于更改将应用到第一行，因此将重写整个文件。

```
replaceLine = 1;
numLines = 5;
newText = 'This file originally contained a magic square';
```

```
fileID = fopen('changing.txt','r');
mydata = cell(1, numLines);
for k = 1:numLines
    mydata{k} = fgetl(fileID);
end
fclose(fileID);
```

```
mydata{replaceLine} = newText;

fileID = fopen('changing.txt','w');
fprintf(fileID,'%s\n',mydata{:});
fclose(fileID);
```

查看文件的内容。

type **changing.txt**

```
This file originally contained a magic square
 2  11  7  14
 3  10  6  15
13   8 12   1
55  55 55  55
```

覆盖文本文件的一部分

将 changing.txt 的第三行替换为 [33 33 33 33]。如果要使用完全相同数量的字符替换文本文件的一部分，则无需重写文件中的任何其他行。

```
replaceLine = 3;
myformat = '%5d %5d %5d %5d\n';
newData = [33 33 33 33];
```

将文件位置标记移动到正确行。

```
fileID = fopen('changing.txt','r+');
for k=1:(replaceLine-1);
    fgetl(fileID);
end
```

在读写操作之间调用 fseek。

```
fseek(fileID,0,'cof');
```

```
fprintf(fileID, myformat, newData);
fclose(fileID);
```

查看文件的内容。

type **changing.txt**

```
This file originally contained a magic square
 2  11  7  14
33  33 33  33
13   8 12   1
55  55 55  55
```

使用不同的字符编码打开文件

编码方案支持特定字母所需的字符，例如日语或欧洲语言字符。常见的编码方案包括 US-ASCII 或 UTF-8。

如果在打开文件进行读取时没有指定编码方案，**fopen** 会使用自动字符集检测来确定编码。如果在打开文件进行写入时没有指定编码方案，**fopen** 默认使用 UTF-8，以便在所有平台和区域设置之间提供互操作性，而不会丢失或损坏数据。

要确定默认编码方案，请打开文件，并再次使用以下语法调用 **fopen**：

```
[filename, permission, machineformat, encoding] = fopen(fid);
```

如果在打开文件时指定了编码方案，以下函数将应用该方案：**fscanf**、**fprintf**、**fgetl**、**fgets**、**fread** 和 **fwrite**。

有关支持的编码方案的完整列表以及用于指定编码的语法，请参阅 **fopen** 参考页。

另请参阅

fprintf | **fopen** | **fseek**

详细信息

- “格式化文本”
- “将数据写入文本文件” (第 2-22 页)

物联网 (IoT) 数据

图像

- “导入图像” (第 6-2 页)
- “导出到图像” (第 6-5 页)

导入图像

要将图形文件中的数据导入到 MATLAB 工作区，请使用 **imread** 函数。使用此函数可以从许多具有标准文件格式的文件中导入数据，这些文件格式包括：标记图像文件格式 (TIFF)、图形交换格式 (GIF)、联合图像专家组 (JPEG) 和可移植网络图形 (PNG) 格式。关于支持格式的完整列表，请参阅 **imread** 参考页。

以下示例将 JPEG 格式的文件中存储的图像数据以数组 **I** 的形式读取到 MATLAB 工作区：

```
I = imread('ngc6543a.jpg');
```

imread 将工作区中的图像表示为 **uint8** 类的多维数组。数组的维度取决于数据的格式。例如，**imread** 使用三个维度表示 RGB 颜色图像：

```
whos I
  Name      Size              Bytes Class
  I        650x600x3          1170000 uint8 array
```

Grand total is 1170000 elements using 1170000 bytes

要更大程度地控制 TIFF 文件的读取，请使用 **Tiff** 对象 - 有关详细信息，请参阅“从 TIFF 文件中读取图像数据和元数据”（第 6-2 页）。

获取有关图像文件的信息

如果有标准图形格式的文件，可使用 **imfinfo** 函数获取有关其内容的信息。**imfinfo** 函数返回包含有关该文件的信息的一个结构体。结构体中的字段根据文件格式的不同而不同，**imfinfo** 始终返回一些基本信息，包括文件名、上次修改日期、文件大小和格式。

以下示例返回联合图像专家组 (JPEG) 格式的文件的相关信息：

```
info = imfinfo('ngc6543a.jpg')
```

```
info =
```

```
    Filename: 'matlabroot\toolbox\matlab\demos\ngc6543a.jpg'
    FileModDate: '01-Oct-1996 16:19:44'
    FileSize: 27387
    Format: 'jpg'
    FormatVersion: ''
    Width: 600
    Height: 650
    BitDepth: 24
    ColorType: 'truecolor'
    FormatSignature: ''
    NumberOfSamples: 3
    CodingMethod: 'Huffman'
    CodingProcess: 'Sequential'
    Comment: {'CREATOR: XV Version 3.00b Rev: 6/15/94 Quality =...'}
    ...
```

从 TIFF 文件中读取图像数据和元数据

虽然可以使用 **imread** 从 TIFF 文件中导入图像数据和元数据，但该函数有一些局限性。例如，TIFF 文件可以包含多个图像，并且每个图像可以有多个子图像。虽然可以使用 **imread** 从多图像 TIFF 文件中读取

所有图像，但无法访问子图像。使用 **Tiff** 对象可以从 TIFF 文件中读取图像数据、元数据和子图像。构造的 **Tiff** 对象表示与 TIFF 文件的连接，并提供对 LibTIFF 库中的众多例程的访问。

下面的示例分步介绍了如何使用 **Tiff** 对象的方法和属性从 TIFF 文件读取子图像。为充分利用 **Tiff** 对象，请熟悉 TIFF 规范和技术说明。请参阅 LibTIFF - TIFF 库和实用工具。

从 TIFF 文件中读取子图像

TIFF 文件可以包含一个或多个图像文件目录 (IFD)。每个 IFD 都包含与图像关联的图像数据和元数据（标记）。每个 IFD 可以包含一个或多个 subIFD，后者也可以包含图像数据和元数据。这些子图像通常是包含 subIFD 的 IFD 中图像数据的降分辨率版本（缩略图）。

要读取 IFD 中的子图像，必须从 **SubIFD** 标记中获取子图像的位置。**SubIFD** 标记包含由指向子图像的字节偏移量构成的数组。然后，可以将 subIFD 的地址传递到 **setSubDirectory** 方法以将该 subIFD 设置为当前 IFD。大多数 **Tiff** 对象方法作用于当前 IFD。

- 1 使用 **Tiff** 对象构造函数打开包含图像和子图像的 TIFF 文件。以下示例使用在“创建 TIFF 文件子目录”（第 6-7 页）中创建的 TIFF 文件，该文件包含一个具有两个 subIFD 的 IFD 目录。**Tiff** 构造函数打开该 TIFF 文件，并将文件中的第一个 subIFD 设置为当前 IFD：

```
t = Tiff('my_subimage_file.tif','r');
```

- 2 检索与当前 IFD 关联的 subIFD 的位置。使用 **getTag** 方法获取 **SubIFD** 标记的值。此方法将返回指定 subIFD 位置的字节偏移量数组：

```
offsets = getTag(t,'SubIFD')
```

- 3 导航到第一个子图像。首先，将 **currentIFD** 设为包含第一个子图像的目录：

```
dirNum = 1;
setDirectory(t,dirNum);
```

- 4 然后，使用 **setSubDirectory** 方法导航到第一个 subIFD。指定该 subIFD 的字节偏移量作为参数。此调用会将该 subIFD 设置为当前 IFD：

```
setSubDirectory(t,offsets(1));
```

- 5 采用与读取文件中的任何其他 IFD 一样的方式，读取当前 IFD（第一个 subIFD）中的图像数据：

```
subimage_one = read(t);
```

- 6 查看第一个子图像：

```
imagesc(subimage_one)
```

- 7 导航到第二个子图像。首先，将 **currentIFD** 重置为包含第二个子图像的目录：

```
setDirectory(t,dirNum);
```

- 8 然后使用 **setSubDirectory** 方法导航到第二个 subIFD。指定第二个 subIFD 的字节偏移量：

```
setSubDirectory(t,offsets(2));
```

- 9 与读取文件中的任何其他 IFD 一样，读取当前 IFD（第二个 subIFD）中的图像数据。

```
subimage_two = read(t);
```

- 10 查看第二个子图像：

```
imagesc(subimage_two)
```

- 11 关闭 **Tiff** 对象：

```
close(t);
```

另请参阅

Tiff

外部网站

- “导出到图像” (第 6-5 页)

导出到图像

要使用某一标准图形文件格式从 MATLAB 工作区导出数据，请使用 `imwrite` 函数。使用此函数，可以以诸如带标记的图像文件格式 (TIFF)、联合图像专家组 (JPEG) 和可移植网络图形 (PNG) 格式导出数据。有关所支持格式的完整列表，请参阅 `imwrite` 参考页。

以下示例将一个 `uint8` 数据的多维数组 `I` 从 MATLAB 工作区写入一个 TIFF 格式的文件。写入该文件的输出图像的类取决于指定的格式。对于大多数格式，如果输入数组为 `uint8` 类，则 `imwrite` 将以 8 位值输出数据。有关详细信息，请参阅 `imwrite` 参考页。

```
whos I
Name      Size      Bytes Class

I         650x600x3    1170000 uint8 array

Grand total is 1170000 elements using 1170000 bytes
imwrite(I, 'my_graphics_file.tif','tif');
```

注意 `imwrite` 支持对多种标准格式使用不同语法。例如，对于 TIFF 文件格式，您可以指定 MATLAB 用于存储图像的压缩类型。有关详细信息，请参阅 `imwrite` 参考页。

在将数据写入 TIFF 文件的过程中若要获得更多控制，请使用 `Tiff` 对象 - 有关详细信息，请参阅“将图像数据和元数据导出到 TIFF 文件”（第 6-5 页）。

将图像数据和元数据导出到 TIFF 文件

虽然可以使用 `imwrite` 将图像数据和元数据（标记）导出到带标记的图像文件格式 (TIFF) 文件，但此函数有一些限制。例如，当要修改文件中的图像数据或元数据时，必须将所有数据写入文件。您不能只写入更新的部分。使用 `Tiff` 对象，可以写入部分图像数据，并修改单个标记或将其添加到 TIFF 文件。构造的 `Tiff` 对象表示与 TIFF 文件的连接，并提供对 LibTIFF 库中的众多例程的访问。

以下各节提供了使用 `Tiff` 对象方法和属性来执行关于 TIFF 文件的一些常见任务的分步示例。为充分利用 `Tiff` 对象，您必须熟悉 TIFF 规范和技术说明。请访问 LibTIFF - TIFF 库和实用工具查看此文档。

创建新的 TIFF 文件

- 1 创建一些图像数据。此示例从 MATLAB 附带的 JPEG 文件中读取图像数据：

```
imgdata = imread('ngc6543a.jpg');
```

- 2 通过构造 `Tiff` 对象来创建新的 TIFF 文件，并将新文件名指定为参数。要创建文件，必须指定写入模式 ('w') 或追加模式 ('a')：

```
t = Tiff('myfile.tif','w');
```

当创建新的 TIFF 文件时，`Tiff` 构造函数会创建一个包含图像文件目录 (IFD) 的文件。TIFF 文件使用此 IFD 来组织与特定图像相关的所有数据和元数据。一个 TIFF 文件可以包含多个 IFD。`Tiff` 对象将它创建的 IFD 设为当前 IFD。`Tiff` 对象方法对当前 IFD 进行操作。可以使用 `Tiff` 对象方法在 TIFF 文件中的各 IFD 之间导航并指定哪个 IFD 是当前 IFD。

- 3 使用 `Tiff` 对象的 `setTag` 方法设置所需的 TIFF 标记。这些所需的标记指定图像的相关信息，如长度和宽度。要将图像数据分割为条带，请指定 `RowsPerStrip` 标记的值。要将图像数据分割为图块，请指定 `TileWidth` 和 `TileLength` 标记的值。以下示例创建一个包含标记名称和值的结构体，并将其传递给 `setTag`。您还可以单独设置每个标记。

```

tagstruct.ImageLength = size(imgdata,1);
tagstruct.ImageWidth = size(imgdata,2);
tagstruct.Photometric = Tiff.Photometric.RGB;
tagstruct.BitsPerSample = 8;
tagstruct.SamplesPerPixel = 3;
tagstruct.RowsPerStrip = 16;
tagstruct.PlanarConfiguration = Tiff.PlanarConfiguration.Chunky;
tagstruct.Software = 'MATLAB';
tagstruct % display tagstruct
setTag(t,tagstruct)

```

有关支持的 TIFF 标记以及如何设置其值的信息，请参阅“设置标记值”（第 6-9 页）。例如，Tiff 对象支持一些属性，您可以用这些属性来设置某些特定属性的值。此示例使用 Tiff 对象的 **PlanarConfiguration** 属性为块配置指定正确的值：**Tiff.PlanarConfiguration.Chunky**。

- 4 使用 Tiff 对象的 **write** 方法将图像数据和元数据写入当前目录。

```
write(t,imgdata);
```

如果要多个图像放入文件，请在执行此写入操作后立即调用 **writeDirectory** 方法。**writeDirectory** 方法在文件中设置一个新的图像文件目录，并将该新目录设为当前目录。

- 5 通过关闭 Tiff 对象来关闭与文件的连接：

```
close(t);
```

- 6 通过使用 **imread** 函数读取文件，然后显示图像，来测试是否创建了有效的 TIFF 文件：

```
imagesc(imread('myfile.tif'));
```

写入图像数据的条带或图块

注意 如果数据未压缩，则只能修改图像数据的一个条带或图块。

- 1 通过创建 Tiff 对象来打开现有的 TIFF 文件进行修改。此示例使用在“创建新的 TIFF 文件”（第 6-5 页）中创建的文件。Tiff 构造函数返回 Tiff 对象的句柄。

```
t = Tiff('myfile.tif','r+');
```

- 2 生成一些数据以写入图像中的条带。此示例创建一个由零值构成的三维数组，其大小与条带大小相同。代码使用条带中的行数、图像宽度和每个像素的样本数作为维度。该数组由 **uint8** 值构成。

```
width = getTag(t,'ImageWidth');
height = getTag(t,'RowsPerStrip');
numSamples = getTag(t,'SamplesPerPixel');
stripData = zeros(height,width,numSamples,'uint8');
```

如果图像数据具有分块布局，则可以使用 **TileWidth** 和 **TileLength** 标记来指定维度。

- 3 使用 **writeEncodedStrip** 方法将数据写入文件中的条带。指定标识要修改的条带的索引编号。此示例选取条带 18，因为它更容易反映图像中的变化。

```
writeEncodedStrip(t,18,stripData);
```

如果图像具有分块布局，则可以使用 **writeEncodedTile** 方法修改图块。

- 4 通过关闭 Tiff 对象来关闭与文件的连接。

```
close(t);
```

- 5 通过使用 **imread** 函数读取文件，然后显示图像，来测试是否修改了 TIFF 文件中图像的条带。

```
modified_imgdata = imread('myfile.tif');
imagesc(modified_imgdata)
```

注意贯穿图像中间的黑色条带。

修改 TIFF 文件元数据 (标记)

- 1 使用 Tiff 对象打开现有 TIFF 文件进行修改。此示例使用在“创建新的 TIFF 文件”（第 6-5 页）中创建的文件。Tiff 构造函数返回 Tiff 对象的句柄。

```
t = Tiff('myfile.tif','r+');
```

- 2 使用 getTag 方法确认文件不包含 Artist 标记。此代码应该发出一条错误消息，指出它无法检索标记。

```
artist_value = getTag(t,'Artist');
```

- 3 使用 setTag 方法添加 Artist 标记。

```
setTag(t,'Artist','Pablo Picasso');
```

- 4 使用 rewriteDirectory 方法将新的标记数据写入 TIFF 文件。在修改文件中的现有元数据或向文件添加新元数据时，使用 rewriteDirectory 方法。

```
rewriteDirectory(t);
```

- 5 通过关闭 Tiff 对象来关闭与文件的连接。

```
close(t);
```

- 6 通过重新打开 TIFF 文件并使用 getTag 方法获取 Artist 标记的值来测试所做的工作。

```
t = Tiff('myfile.tif','r');
```

```
getTag(t,'Artist')
```

```
ans =
```

```
Pablo Picasso
```

```
close(t);
```

创建 TIFF 文件子目录

- 1 创建一些图像数据。此示例从 MATLAB 附带的 JPEG 文件中读取图像数据。然后，该示例在图像数据的基础上创建两个低分辨率（缩略图）版本。

```
imgdata = imread('ngc6543a.jpg');
```

```
%
```

```
% Reduce number of pixels by a half.
```

```
img_half = imgdata(1:2:end,1:2:end,:);
```

```
%
```

```
% Reduce number of pixels by a third.
```

```
img_third = imgdata(1:3:end,1:3:end,:);
```

- 2 通过构造 Tiff 对象来创建新的 TIFF 文件，并将新文件的名称指定为参数。要创建文件，必须指定写入模式 ('w') 或追加模式 ('a')：Tiff 构造函数返回 Tiff 对象的句柄。

```
t = Tiff('my_subimage_file.tif','w');
```

- 3 使用 Tiff 对象的 setTag 方法设置所需的 TIFF 标记。这些所需的标记指定图像的相关信息，如长度和宽度。要将图像数据分割为条带，请指定 RowsPerStrip 标记的值。要将图像数据分割为图块，请使用 TileWidth 和 TileLength 标记。以下示例创建一个包含标记名称和值的结构体，并将其传递给 setTag。也可以单独设置每个标记。

要创建子目录，必须设置 **SubIFD** 标记，指定要创建的子目录的数量。注意，指定的数字不是 **SubIFD** 标记的值。该数字告知 Tiff 软件创建一个指向两个子目录的 **SubIFD**。**SubIFD** 标记的实际值将是两个子目录的字节偏移量。

```
tagstruct.ImageLength = size(imgdata,1);
tagstruct.ImageWidth = size(imgdata,2);
tagstruct.Photometric = Tiff.Photometric.RGB;
tagstruct.BitsPerSample = 8;
tagstruct.SamplesPerPixel = 3;
tagstruct.RowsPerStrip = 16;
tagstruct.PlanarConfiguration = Tiff.PlanarConfiguration.Chunky;
tagstruct.Software = 'MATLAB';
tagstruct.SubIFD = 2 ; % required to create subdirectories
tagstruct % display tagstruct
setTag(t,tagstruct)
```

有关支持的 TIFF 标记以及如何设置其值的信息，请参阅“设置标记值”（第 6-9 页）。例如，Tiff 对象支持一些属性，您可以用这些属性来设置某些特定属性的值。此示例使用 Tiff 对象的 **PlanarConfiguration** 属性为块配置指定正确的值：**Tiff.PlanarConfiguration.Chunky**。

- 4 使用 Tiff 对象的 **write** 方法将图像数据和元数据写入当前目录。

```
write(t,imgdata);
```

- 5 通过调用 **writeDirectory** 方法设置第一个子目录。**writeDirectory** 方法设置子目录并将新目录设为当前目录。因为您指定要创建两个子目录，**writeDirectory** 会设置一个子目录。

```
writeDirectory(t);
```

- 6 设置所需的标记，就像对常规目录所做的那样。根据 LibTIFF API，子目录不能包含 **SubIFD** 标记。

```
tagstruct2.ImageLength = size(img_half,1);
tagstruct2.ImageWidth = size(img_half,2);
tagstruct2.Photometric = Tiff.Photometric.RGB;
tagstruct2.BitsPerSample = 8;
tagstruct2.SamplesPerPixel = 3;
tagstruct2.RowsPerStrip = 16;
tagstruct2.PlanarConfiguration = Tiff.PlanarConfiguration.Chunky;
tagstruct2.Software = 'MATLAB';
tagstruct2 % display tagstruct2
setTag(t,tagstruct2)
```

- 7 使用 Tiff 对象的 **write** 方法将图像数据和元数据写入子目录。

```
write(t,img_half);
```

- 8 通过调用 **writeDirectory** 方法设置第二个子目录。**writeDirectory** 方法设置子目录并将其设为当前目录。

```
writeDirectory(t);
```

- 9 设置所需的标记，就像对任何目录所做的那样。根据 LibTIFF API，子目录不能包含 **SubIFD** 标记。

```
tagstruct3.ImageLength = size(img_third,1);
tagstruct3.ImageWidth = size(img_third,2);
tagstruct3.Photometric = Tiff.Photometric.RGB;
tagstruct3.BitsPerSample = 8;
tagstruct3.SamplesPerPixel = 3;
tagstruct3.RowsPerStrip = 16;
tagstruct3.PlanarConfiguration = Tiff.PlanarConfiguration.Chunky;
tagstruct3.Software = 'MATLAB';
tagstruct3 % display tagstruct3
setTag(t,tagstruct3)
```

10 使用 **Tiff** 对象的 **write** 方法将图像数据和元数据写入子目录：

```
write(t,img_third);
```

11 通过关闭 **Tiff** 对象来关闭与文件的连接：

```
close(t);
```

设置标记值

下表列出了 **Tiff** 对象支持的所有 TIFF 标记，并包括有关其 MATLAB 类和大小的信息。对于某些标记，该表还指示了 **Tiff** 对象支持的值集合，它是 TIFF 规范定义的所有可能值的子集。您可以使用 **Tiff** 属性结构体为这些标记指定支持的值。例如，用 **Tiff.Compression.JPEG** 指定 JPEG 压缩。有关完整的属性列表，请参阅 **Tiff** 参考页。

表 1: 支持的 TIFF 标记

TIFF 标记	类	大小	支持的值	注释
Artist	char	1xN		
BitsPerSample	double	1x1	1、8、16、32、64	请参阅表 2 (第 6-13 页)
ColorMap	double	256x3	值应归一化为 0 - 1 之间的值。在内部存储为 uint16 值。	Photometric 必须是 Palette
Compression	double	1x1	None: 1 CCITTRLE: 2 CCITTFax3: 3 CCITTFax4: 4 LZW: 5 JPEG: 7 CCITTRLEW: 32771 PackBits: 32773 Deflate: 32946 AdobeDeflate: 8	请参阅表 3 (第 6-14 页)。
Copyright	char	1xN		
DateTime	char	1x19	如果需要, 将填充返回值以使其包含 19 个字符。	
DocumentName	char	1xN		
DotRange	double	1x2		Photometric 必须是 Separated
ExtraSamples	double	1xN	Unspecified: 0 AssociatedAlpha: 1 UnassociatedAlpha: 2	请参阅表 4 (第 6-14 页)。
FillOrder	double	1x1		
GeoAsciiParamsTag	char	1xN		
GeoDoubleParamsTag	double	1xN		
GeoKeyDirectoryTag	double	Nx4		
Group3Options	double	1x1		Compression 必须是 CCITTFax3
Group4Options	double	1x1		Compression 必须是 CCITTFax4
HalfToneHints	double	1x2		
HostComputer	char	1xn		
ICCProfile	uint8	1xn		
ImageDescription	char	1xn		
ImageLength	double	1x1		

TIFF 标记	类	大小	支持的值	注释
ImageWidth	double	1x1		
InkNames	char cell array	1x NumInks		Photometric 必须是 Separated
InkSet	double	1x1	CMYK:1 MultiInk: 2	Photometric 必须是 Separated
JPEGQuality	double	1x1	介于 1 和 100 之间的值	
Make	char	1xn		
MaxSampleValue	double	1x1	0 - 65,535	
MinSampleValue	double	1x1	0 - 65,535	
Model	char	1xN		
ModelPixelScaleTag	double	1x3		
ModelTiepointTag	double	Nx6		
ModelTransformationMatrixTag	double	1x16		
NumberOfInks	double	1x1		必须等于 SamplesPerPixel
Orientation	double	1x1	TopLeft:1 TopRight: 2 BottomRight: 3 BottomLeft: 4 LeftTop: 5 RightTop: 6 RightBottom: 7 LeftBottom: 8	
PageName	char	1xN		
PageNumber	double	1x2		
Photometric	double	1x1	MinIsWhite:0 MinIsBlack: 1 RGB: 2 Palette: 3 Mask: 4 Separated: 5 YCbCr: 6 CIELab: 8 ICCLab: 9 ITULab: 10	请参阅表 2 (第 6-13 页)。
Photoshop	uint8	1xN		
PlanarConfiguration	double	1x1	Chunky:1 Separate: 2	
PrimaryChromaticities	double	1x6		
ReferenceBlackWhite	double	1x6		

TIFF 标记	类	大小	支持的值	注释
ResolutionUnit	double	1x1		
RICTIFFIPTC	uint8	1xN		
RowsPerStrip	double	1x1		
RPCCoefficientTag	double	1x92	92 元素行向量	请参阅表 6 (第 6-15 页)
SampleFormat	double	1x1	UInt:1 Int: 2 IEEEFP: 3	请参阅表 2 (第 6-13 页)
SamplesPerPixel	double	1x1		
SMaxSampleValue	double	1x1	为图像数据指定的 MATLAB 数据类型的范 围	
SMinSampleValue	double	1x1	为图像数据指定的 MATLAB 数据类型的范 围	
Software	char	1xN		
StripByteCounts	double	1xN		只读
StripOffsets	double	1xN		只读
SubFileType	double	1x1	Default :0 ReducedImage: 1 Page: 2 Mask: 4	
SubIFD	double	1x1		
TargetPrinter	char	1xN		
Thresholding	double	1x1	BiLevel:1 HalfTone: 2 ErrorDiffuse: 3	Photometric 可以是 MinIsWhite 或 MinIsBlack
TileByteCounts	double	1xN		只读
TileLength	double	1x1	必须是 16 的倍数	
TileOffsets	double	1xN		只读
TileWidth	double	1x1	必须是 16 的倍数	
TransferFunction	double	请参阅注 释 ¹	每个值应在 0 - 2 ¹⁶ -1 内	SamplePerPixel 可以 是 1 或 3
WhitePoint	double	1x2		Photometric 可以是: RGB Palette YCbCr CIELab ICCLab ITULab
XMP	char	1xn		N>5
XPostion	double	1x1		

TIFF 标记	类	大小	支持的值	注释
XResolution	double	1x1		
YCbCrCoefficients	double	1x3		Photometric 必须是 YCbCr
YCbCrPositioning	double	1x1	Centered:1 Cosited: 2	Photometric 必须是 YCbCr
YCbCrSubSampling	double	1x2		Photometric 必须是 YCbCr
YPosition	double	1x1		
YResolution	double	1x1		
ZipQuality	double	1x1	介于 1 和 9 之间的值	

¹ 大小是 $1 \times 2^{\text{BitsPerSample}}$ 或 $3 \times 2^{\text{BitsPerSample}}$ 。

表 2: BitsPerSample 设置的有效 SampleFormat 值

BitsPerSample	SampleFormat	MATLAB 数据类型
1	Uint	logical
8	Uint, Int	uint8, int8
16	Uint, Int	uint16, int16
32	Uint, Int, IEEEFP	uint32, int32, single
64	IEEEFP	double

表 3: BitsPerSample 和 Photometric 组合的有效 SampleFormat 值

	BitsPerSample 值				
Photometric 值	1	8	16	32	64
MinIsWhite	Uint	Uint/Int	Uint Int	Uint Int IEEEFP	IEEEFP
MinIsBlack	Uint	Uint/Int	Uint Int	Uint Int IEEEFP	IEEEFP
RGB		Uint	Uint	Uint IEEEFP	IEEEFP
Palette		Uint	Uint		
Mask	Uint				
Separated		Uint	Uint	Uint IEEEFP	IEEEFP
YCbCr		Uint	Uint	Uint IEEEFP	IEEEFP
CIELab		Uint	Uint		
ICCLab		Uint	Uint		
ITULab		Uint	Uint		

表 4: BitsPerSample 和 Compression 组合的有效 SampleFormat 值

	BitsPerSample 值				
Compression 值	1	8	16	32	64
None	UInt	UInt Int	UInt Int	UInt Int IEEEFP	IEEEFP
CCITTRLE	UInt				
CCITTFax3	UInt				
CCITTFax4	UInt				
LZW	UInt	UInt Int	UInt Int	UInt Int IEEEFP	IEEEFP
JPEG		UInt Int			
CCITTRLEW	UInt				
PackBits	UInt	UInt Int	UInt Int	UInt Int IEEEFP	IEEEFP
Deflate	UInt	UInt Int	UInt Int	UInt Int IEEEFP	IEEEFP
AdobeDeflate	UInt	UInt Int	UInt Int	UInt Int IEEEFP	IEEEFP

表 5: Photometric 设置的有效 SamplesPerPixel 值

Photometric 值	SamplesPerPixel ¹
MinIsWhite	1+
MinIsBlack	1+
RGB	3+
Palette	1
Mask	1
Separated	1+
YCbCr	3
CIELab	3+
ICCLab	3+
ITULab	3+

表 6: RPCCoefficientTag 值说明列表

92 元素向量中的索引值	值说明 ¹	单位
1	均方根偏差误差	每个水平轴的米数
2	均方根随机误差	每个水平轴的米数
3	线偏移量	像素
4	样本偏移量	像素
5	地理纬度偏移量	度
6	地理经度偏移量	度
7	地理高度偏移量	米
8	线缩放因子	像素
9	样本缩放因子	像素
10	地理纬度标度	度
11	地理经度标度	度
12	地理高度缩放因子	米
13 至 32	有理多项式方程 $r(n)$ 的分子系数 ²	
33 至 52	有理多项式方程 $r(n)$ 的分母系数	
53 至 72	有理多项式方程 $c(n)$ 的分子系数 ²	
73 至 92	有理多项式方程 $c(n)$ 的分母系数	

¹ 要使用 RPCCoefficientTag 对象指定此向量中的值，请参阅 Mapping Toolbox™ 中的 RPCCoefficientTag (Mapping Toolbox)。

² 方程 $r(n)$ 和 $c(n)$ 表示通用严格投影模型的归一化行和列值。

另请参阅

Tiff

外部网站

- “导入图像” (第 6-2 页)

科学数据

- “导入 NetCDF 文件和 OPeNDAP 数据” (第 7-2 页)
- “导出到 NetCDF 文件” (第 7-8 页)
- “以编程方式导入 HDF4 文件” (第 7-13 页)

导入 NetCDF 文件和 OPeNDAP 数据

本文先后使用 `netcdf` 包中的高级函数和低级函数读取 NetCDF 文件。

本节内容
“MATLAB NetCDF 功能” (第 7-2 页)
“连接到 OPeNDAP 服务器时的安全注意事项” (第 7-2 页)
“使用高级函数读取 NetCDF 文件” (第 7-2 页)
“在 NetCDF 文件中查找所有无限维度” (第 7-4 页)
“使用低级函数读取 NetCDF 文件” (第 7-5 页)

MATLAB NetCDF 功能

网络通用数据表 (NetCDF) 是一组软件库及与机器无关的数据格式，支持创建、访问和共享面向数组的科学数据。NetCDF 广泛用于工程和科学领域，这些领域需要一种标准的数据存储方式，以便共享数据。

MATLAB 高级函数简化了从 NetCDF 文件或 OPeNDAP NetCDF 数据源导入数据的过程。MATLAB 低级函数通过提供对 NetCDF C 库中例程的访问权限来实现对导入过程的更多控制。要有效地使用低级函数，您应该熟悉 NetCDF C 接口。NetCDF 文档可在 Unidata 网站上获取。

注意 有关导入具有单独的、不兼容格式的通用数据格式 (CDF) 文件的信息，请参阅 “Import CDF Files Using Low-Level Functions”。

连接到 OPeNDAP 服务器时的安全注意事项

强烈建议您只连接到受信任的 OPeNDAP 服务器。在 R2020b 中，默认情况下，MATLAB NetCDF 接口仅通过执行服务器证书和主机名验证连接到受信任的数据访问协议 (DAP) 端点。以前，当您访问 OPeNDAP 服务器时，默认情况下服务器证书和主机名验证都处于禁用状态。

如果要禁用服务器证书和主机名验证，请在当前目录的 `.dodsrc` 文件中添加以下行：

```
[mylocaltestserver.lab] HTTP.SSL.VALIDATE=0
```

这样，MATLAB NetCDF 接口会连接到在 URI `mylocaltestserver.lab` 中指定名称的 OPeNDAP 服务器，而无需对服务器证书或主机名执行任何验证。此更改将在以后的 MATLAB 会话中持续存在。有关 OPeNDAP 服务器身份验证和主机名验证的详细信息，请参阅 netCDF 授权支持。

使用高级函数读取 NetCDF 文件

此示例说明如何使用高级函数显示和读取 NetCDF 文件的内容。

显示示例 NetCDF 文件 `example.nc` 的内容。

```
ncdisp('example.nc')

Source:
    \\matlabroot\toolbox\matlab\demos\example.nc
Format:
    netcdf4
```

```

Global Attributes:
  creation_date = '29-Mar-2010'
Dimensions:
  x = 50
  y = 50
  z = 5
Variables:
  avagadros_number
    Size: 1x1
    Dimensions:
    Datatype: double
    Attributes:
      description = 'this variable has no dimensions'
  temperature
    Size: 50x1
    Dimensions: x
    Datatype: int16
    Attributes:
      scale_factor = 1.8
      add_offset = 32
      units = 'degrees_fahrenheit'
  peaks
    Size: 50x50
    Dimensions: x,y
    Datatype: int16
    Attributes:
      description = 'z = peaks(50);'
Groups:
  /grid1/
    Attributes:
      description = 'This is a group attribute.'
    Dimensions:
      x = 360
      y = 180
      time = 0 (UNLIMITED)
    Variables:
      temp
        Size: []
        Dimensions: x,y,time
        Datatype: int16
  /grid2/
    Attributes:
      description = 'This is another group attribute.'
    Dimensions:
      x = 360
      y = 180
      time = 0 (UNLIMITED)
    Variables:
      temp
        Size: []
        Dimensions: x,y,time
        Datatype: int16

```

ncdisp 显示文件中的所有组、维度和变量定义。无限维度用标签 **UNLIMITED** 标识。

从 **peaks** 变量中读取数据。

```
peaksData = ncread('example.nc','peaks');
```

显示有关 `peaksData` 输出的信息。

```
whos peaksData
```

```

Name          Size          Bytes Class Attributes
peaksData     50x50          5000 int16

```

读取与变量相关联的 `description` 属性。

```
peaksDesc = ncreadatt('example.nc','peaks','description')
```

```
peaksDesc =
```

```
z = peaks(50);
```

创建变量数据的三维曲面图。用 `description` 属性的值作为图标题。

```
surf(double(peaksData))
title(peaksDesc);
```

读取与 `/grid1/` 组相关联的 `description` 属性。将组名指定为 `ncreadatt` 函数的第二个输入。

```
g = ncreadatt('example.nc','/grid1/','description')
```

```
g =
```

```
This is a group attribute.
```

读取全局属性 `creation_date`。对于全局属性，将 `ncreadatt` 的第二个输入参数指定为 `'/'`。

```
creation_date = ncreadatt('example.nc','/','creation_date')
```

```
creation_date =
```

```
29-Mar-2010
```

在 NetCDF 文件中查找所有无限维度

此示例说明如何使用高级函数在 NetCDF 文件中查找一个组的所有无限维度。

使用 `ncinfo` 函数获取示例文件 `example.nc` 中 `/grid2/` 组的相关信息。

```
ginfo = ncinfo('example.nc','/grid2/')
```

```
ginfo =
```

```

Filename: '\\matlabroot\toolbox\matlab\demos\example.nc'
Name: 'grid2'
Dimensions: [1x3 struct]
Variables: [1x1 struct]
Attributes: [1x1 struct]
Groups: []
Format: 'netcdf4'

```

`ncinfo` 返回一个包含组信息的结构体数组。

获取表示该组的无限维度的布尔值的向量。

```
unlimDims = [ginfo.Dimensions.Unlimited]
```

```
unlimDims =
```

```
    0    0    1
```

用 `unlimDims` 向量显示无限维度。

```
disp(ginfo.Dimensions(unlimDims))
```

```
    Name: 'time'  
    Length: 0  
    Unlimited: 1
```

使用低级函数读取 NetCDF 文件

此示例显示如何使用 `netcdf` 包中的 MATLAB 低级函数获取有关 NetCDF 文件中的维度、变量和属性的信息。要有效使用这些函数，应该熟悉 NetCDF C 接口。

打开 NetCDF 文件

使用 `netcdf.open` 函数，以只读访问权限打开示例 NetCDF 文件 `example.nc`。

```
ncid = netcdf.open('example.nc','NC_NOWRITE')
```

```
ncid = 65536
```

`netcdf.open` 返回文件标识符。

获取有关 netCDF 文件的信息

使用 `netcdf.inq` 函数获取有关文件内容的信息。此函数对应于 netCDF 库 C API 中的 `nc_inq` 函数。

```
[ndims,nvars,natts,unlimdimID] = netcdf.inq(ncid)
```

```
ndims = 3
```

```
nvars = 3
```

```
natts = 1
```

```
unlimdimID = -1
```

`netcdf.inq` 返回文件中维度、变量和全局属性的数量，并返回文件中无限维度的标识符。无限维度可以增长。

使用 `netcdf.inqAttName` 函数获取文件中全局属性的名称。此函数对应于 netCDF 库 C API 中的 `nc_inq_attname` 函数。要获取属性的名称，必须指定该属性所关联变量的 ID 和属性编号。要访问与特定变量不相关联的全局属性，请使用常量 `'NC_GLOBAL'` 作为变量 ID。

```
global_att_name = netcdf.inqAttName(ncid,...  
    netcdf.getConstant('NC_GLOBAL'),0)
```

```
global_att_name =  
'creation_date'
```

使用 `netcdf.inqAtt` 函数获取有关属性的数据类型和长度的信息。此函数对应于 netCDF 库 C API 中的 `nc_inq_att` 函数。再次使用 `netcdf.getConstant('NC_GLOBAL')` 指定变量 ID。

```
[xtype,attlen] = netcdf.inqAtt(ncid,...
    netcdf.getConstant('NC_GLOBAL'),global_att_name)
```

```
xtype = 2
```

```
attlen = 11
```

使用 `netcdf.getAtt` 函数获取属性值。

```
global_att_value = netcdf.getAtt(ncid,...
    netcdf.getConstant('NC_GLOBAL'),global_att_name)
```

```
global_att_value =
'29-Mar-2010'
```

使用 `netcdf.inqDim` 函数获取文件中第一个维度的相关信息。此函数对应于 netCDF 库 C API 中的 `nc_inq_dim` 函数。`netcdf.inqDim` 的第二个输入是维度 ID，它是一个标识维度的从 0 开始的索引。第一个维度的索引值为 0。

```
[dimname,dimlen] = netcdf.inqDim(ncid,0)
```

```
dimname =
'x'
```

```
dimlen = 50
```

`netcdf.inqDim` 返回维度的名称和长度。

使用 `netcdf.inqVar` 函数获取文件中第一个变量的相关信息。此函数对应于 netCDF 库 C API 中的 `nc_inq_var` 函数。`netcdf.inqVar` 的第二个输入是变量 ID，它是一个标识变量的从 0 开始的索引。第一个变量的索引值为 0。

```
[varname,vartype,dimids,natts] = netcdf.inqVar(ncid,0)
```

```
varname =
'avagadros_number'
```

```
vartype = 6
```

```
dimids =
```

```
[]
```

```
natts = 1
```

`netcdf.inqVar` 返回名称、数据类型、维度 ID 以及与变量相关联的属性的数量。`vartype` 中返回的数据类型信息是 NetCDF 数据类型常量的数值，例如，`NC_INT` 和 `NC_BYTE`。有关这些常量的信息，请参阅 NetCDF 文档。

读取 NetCDF 文件中的数据

使用 `netcdf.getVar` 函数读取示例文件中与变量 `avagadros_number` 相关联的数据。`netcdf.getVar` 的第二个输入是变量 ID，它是一个标识变量的从 0 开始的索引。`avagadros_number` 变量的索引值为 0。

```
A_number = netcdf.getVar(ncid,0)
```

```
A_number = 6.0221e+23
```

查看 `A_number` 的数据类型。

```
whos A_number
```

Name	Size	Bytes	Class	Attributes
A_number	1x1	8	double	

`netcdf` 包中的函数自动选择与 NetCDF 数据类型最匹配的 MATLAB 类，但您也可以使用 `netcdf.getVar` 的可选参数指定返回数据的类。

读取与 `avagadros_number` 相关联的数据，并以 `single` 类返回数据。

```
A_number = netcdf.getVar(ncid,0,'single');
whos A_number
```

Name	Size	Bytes	Class	Attributes
A_number	1x1	4	single	

关闭 NetCDF 文件

关闭 NetCDF 文件 `example.nc`。

```
netcdf.close(ncid)
```

另请参阅

`ncread` | `ncreadatt` | `ncdisp` | `ncinfo`

详细信息

- “Map NetCDF API Syntax to MATLAB Syntax”

外部网站

- NetCDF C 接口

导出到 NetCDF 文件

使用高级函数和 `netcdf` 包低级函数创建、合并和写入 NetCDF 文件。

本节内容
“MATLAB NetCDF 功能” (第 7-8 页)
“基于现有文件或模板新建 NetCDF 文件” (第 7-8 页)
“合并两个 NetCDF 文件” (第 7-9 页)
“使用低级函数将数据写入 NetCDF 文件” (第 7-11 页)

MATLAB NetCDF 功能

网络通用数据表 (NetCDF) 是一组软件库及与机器无关的数据格式，支持创建、访问和共享面向数组的科学数据。NetCDF 广泛用于工程和科学领域，这些领域需要一种标准的数据存储方式，以便共享数据。

使用 MATLAB 高级函数，将数据导出到 netCDF 文件的操作变得容易。MATLAB 低级函数提供对 NetCDF C 库中例程的访问权限。要有效地使用低级函数，您应该熟悉 NetCDF C 接口。NetCDF 文档可在 Unidata 网站上获取。

注意 有关导出到常用数据格式 (CDF) 文件（这类文件具有单独的不兼容格式）的信息，请参阅 “Export to CDF Files”。

基于现有文件或模板新建 NetCDF 文件

此示例说明如何创建一个新 NetCDF 文件，该文件包含某个现有文件的变量、维度和组定义，但使用不同的格式。

使用 `nccreate` 函数创建包含一个变量的文件。

```
nccreate('myfile.nc','myvar')
```

向该文件写入数据。

```
A = 99;  
ncwrite('myfile.nc','myvar',A)
```

使用 `ncinfo` 从该文件中读取变量、维度和组定义。此信息定义文件的架构。

```
S = ncinfo('myfile.nc');
```

获取文件的格式。

```
file_fmt = S.Format  
  
file_fmt =  
'netcdf4_classic'
```

将结构体 S 中 `Format` 字段的值更改为另一种受支持的 NetCDF 格式。

```
S.Format = 'netcdf4';
```

使用 `ncwritschema` 函数，创建一个使用新格式的新文件版本。架构定义文件的结构，但不包含原始文件中的任何数据。

```
ncwritschema('newfile.nc',S)
S = ncinfo('newfile.nc');
```

注意：当您使用 `ncwritschema` 转换文件格式时，如果原始文件格式包含新格式不支持的字段，您可能会收到警告消息。例如，`netcdf4` 格式支持填充值，但 NetCDF 经典格式不支持。在这些情况下，`ncwritschema` 仍然会创建文件，但会忽略在新格式中未定义的字段。

查看新文件的格式。

```
new_fmt = S.Format

new_fmt =
'netcdf4'
```

新文件 `newfile.nc` 包含 `myfile.nc` 的变量和维度定义，但不包含数据。

将数据写入新文件。

```
ncwrite('newfile.nc','myvar',A)
```

合并两个 NetCDF 文件

此示例说明如何使用高级函数合并两个 NetCDF 文件。合并后的文件包含所合并的原始文件的变量和维度定义，但不包含这些原始文件中的数据。

创建名为 `ex1.nc` 的 NetCDF 文件，并定义名为 `myvar` 的变量。然后，将数据写入该变量并显示文件内容。

```
nccreate('ex1.nc','myvar');
ncwrite('ex1.nc','myvar',55)
ncdisp('ex1.nc')
```

```
Source:
  pwd\ex1.nc
Format:
  netcdf4_classic
Variables:
  myvar
    Size:    1x1
    Dimensions:
    Datatype: double
```

创建第二个文件并定义名为 `myvar2` 的变量。然后，将数据写入该变量并显示文件内容。

```
nccreate('ex2.nc','myvar2');
ncwrite('ex2.nc','myvar2',99)
ncdisp('ex2.nc')
```

```
Source:
  pwd\ex2.nc
Format:
  netcdf4_classic
Variables:
  myvar2
```

```

Size:    1x1
Dimensions:
Datatype: double

```

使用 `ncinfo` 函数获取每个文件的架构。

```
info1 = ncinfo('ex1.nc')
```

```
info1 =
```

```

Filename: 'pwd\ex1.nc'
Name: '/'
Dimensions: []
Variables: [1x1 struct]
Attributes: []
Groups: []
Format: 'netcdf4_classic'

```

```
info2 = ncinfo('ex2.nc')
```

```
info2 =
```

```

Filename: 'pwd\ex2.nc'
Name: '/'
Dimensions: []
Variables: [1x1 struct]
Attributes: []
Groups: []
Format: 'netcdf4_classic'

```

使用 `ncwritschema` 函数，创建使用第一个示例文件的架构的新 NetCDF 文件。然后，显示文件内容。

```
ncwritschema('combined.nc',info1)
ncdisp('combined.nc')
```

```

Source:
    pwd\combined.nc
Format:
    netcdf4_classic
Variables:
    myvar
        Size:    1x1
        Dimensions:
        Datatype: double
        Attributes:
            _FillValue = 9.969209968386869e+36

```

使用 `ncwritschema` 将 `ex2.nc` 的架构添加到 `combined.nc`。

```
ncwritschema('combined.nc',info2)
```

查看合并后的文件的内容。

```
ncdisp('combined.nc')
```

```

Source:
    pwd\combined.nc
Format:

```

```

    netcdf4_classic
Variables:
  myvar
    Size:    1x1
    Dimensions:
    Datatype: double
    Attributes:
      _FillValue = 9.969209968386869e+36
  myvar2
    Size:    1x1
    Dimensions:
    Datatype: double
    Attributes:
      _FillValue = 9.969209968386869e+36

```

该文件包含第一个示例文件中定义的 `myvar` 变量和第二个文件中定义的 `myvar2` 变量。

使用低级函数将数据写入 NetCDF 文件

此示例说明如何使用低级函数向 NetCDF 文件写入数据。MATLAB® 低级函数提供对 NetCDF C 库中例程的访问。MATLAB 将这些函数组合到一个称为 `netcdf` 的包中。要调用该包中的某个函数，必须使用包名称作为该函数名称的前缀。

为了有效地使用 MATLAB NetCDF 函数，您需要熟悉 NetCDF C 接口的有关信息。

要运行本例，必须对当前目录拥有写入权限。

在 MATLAB 工作区中创建名为 `my_data` 的 1×50 数值变量。该向量属于 `double` 类。

```
my_data = linspace(0,49,50);
```

使用 `netcdf.create` 函数创建名为 `my_file.nc` 的 NetCDF 文件。`NOCLOBBER` 参数是 NetCDF 文件访问常量，它指示不要覆盖同名的现有文件。

```
ncid = netcdf.create('my_file.nc','NOCLOBBER');
```

`netcdf.create` 返回文件标识符 `ncid`。创建 NetCDF 文件时，文件将以定义模式打开。必须在定义模式下才能定义维度和变量。

使用 `netcdf.defDim` 函数在文件中定义维度。此函数对应于 netCDF 库 C API 中的 `nc_def_dim` 函数。在定义变量并向文件写入数据之前，您必须先先在文件中定义维度。在本例中，定义名为 `my_dim`、长度为 50 的维度。

```
dimid = netcdf.defDim(ncid,'my_dim',50)
```

```
dimid = 0
```

`netcdf.defDim` 返回对应于新维度的维度标识符。标识符是从 0 开始的索引。

使用 `netcdf.defVar` 函数，在维度上定义名为 `my_var` 的变量。此函数对应于 netCDF 库 C API 中的 `nc_def_var` 函数。指定变量的 NetCDF 数据类型，在本例中为 `NC_BYTE`。

```
varid = netcdf.defVar(ncid,'my_var','NC_BYTE',dimid)
```

```
varid = 0
```

`netcdf.defVar` 返回对应于 `my_var` 的变量标识符。

使 NetCDF 文件退出定义模式。要向文件写入数据，您必须处于数据模式。

```
netcdf.endDef(ncid)
```

使用 `netcdf.putVar` 函数，将来自 MATLAB 工作区的数据写入 NetCDF 文件中的变量。工作区中的数据属于 `double` 类，但 NetCDF 文件中的变量属于 `NC_BYTE` 类型。MATLAB NetCDF 函数会自动进行转换。

```
netcdf.putVar(ncid,varid,my_data)
```

使用 `netcdf.close` 函数关闭文件。

```
netcdf.close(ncid)
```

通过打开文件并将数据从变量中读取到 MATLAB 工作区的新变量中，验证数据已写入文件。

```
ncid2 = netcdf.open('my_file.nc','NC_NOWRITE');
x = netcdf.getVar(ncid2,0);
```

查看 `x` 的数据类型。

```
whos x
```

Name	Size	Bytes	Class	Attributes
x	50x1	50	int8	

MATLAB 以列优先顺序存储数据，而 NetCDF C API 使用行优先顺序。`x` 表示存储在 NetCDF 文件中的数据，因此它是 `50×1`，即使在 MATLAB 工作区中的原始向量 `my_data` 是 `1×50`。由于您将数据作为 `NC_BYTE` 存储在 NetCDF 文件中，因此 MATLAB 将变量中的数据作为 `int8` 类读入工作区中。

关闭文件。

```
netcdf.close(ncid2)
```

另请参阅

详细信息

- “Map NetCDF API Syntax to MATLAB Syntax”

外部网站

- NetCDF C 接口

以编程方式导入 HDF4 文件

本节内容
“概述” (第 7-13 页)
“使用 MATLAB HDF4 高级函数” (第 7-13 页)

概述

分层数据格式 (HDF4) 是由美国国家超级计算应用中心 (NCSA) 开发的用于存储文件中科学数据的通用型机器无关标准。有关这些文件格式的详细信息，请阅读 HDF 网站 (www.hdfgroup.org) 上的 HDF 文档。

HDF-EOS 是 HDF4 的一种扩展，由美国国家航空和航天局 (NASA) 开发，用于存储从地球观测系统 (EOS) 返回的数据。有关此 HDF4 扩展的详细信息，请参阅 NASA 网站 (www.hdfEOS.org) 上的 HDF-EOS 文档。

MATLAB 包括几个导入 HDF4 文件的选项，将在以下各节讨论。

注意 有关导入单独的不兼容格式 HDF5 数据的信息，请参阅 “Import HDF5 Files” 。

使用 MATLAB HDF4 高级函数

要从 HDF 或 HDF-EOS 文件导入数据，可以使用 MATLAB HDF4 高级函数 `hdfread`。`hdfread` 函数提供了一种从 HDF4 或 HDF-EOS 文件导入数据的编程方法，它隐藏了许多在使用 “Import HDF4 Files Using Low-Level Functions” 中描述的低级 HDF 函数导入数据时需要知道的细节。

本节介绍这些高级 MATLAB HDF 函数，包括

- “使用 `hdfinfo` 获取有关 HDF4 文件的信息” (第 7-13 页)
- “使用 `hdfread` 从 HDF4 文件导入数据” (第 7-14 页)

要将数据导出到 HDF4 文件，必须使用 MATLAB HDF4 低级函数。

使用 `hdfinfo` 获取有关 HDF4 文件的信息

要获取有关 HDF4 文件内容的信息，请使用 `hdfinfo` 函数。`hdfinfo` 函数返回一个结构体，其中包含有关文件及文件中数据的信息。

此示例返回有关 MATLAB 附带的示例 HDF4 文件的信息：

```
info = hdfinfo('example.hdf')

info =

    Filename: 'matlabroot\example.hdf'
  Attributes: [1x2 struct]
        Vgroup: [1x1 struct]
         SDS: [1x1 struct]
        Vdata: [1x1 struct]
```

要获取有关存储在文件中的数据集的信息，请查看 `SDS` 字段。

使用 hdfread 从 HDF4 文件导入数据

要使用 `hdfread` 函数，必须指定要读取的数据集。可以指定文件名和数据集名称作为参数，也可以指定包含此信息的 `hdfinfo` 函数返回的结构体。以下示例显示了这两种方法。有关如何导入数据集中数据子集的信息，请参阅“读取数据集中数据的子集”（第 7-15 页）。

- 1 使用 `hdfinfo` 函数确定 HDF4 文件中数据集的名称。

```
info = hdfinfo('example.hdf')

info =

    Filename: 'matlabroot\example.hdf'
  Attributes: [1x2 struct]
    Vgroup: [1x1 struct]
      SDS: [1x1 struct]
    Vdata: [1x1 struct]
```

要确定文件中数据集的名称及其他相关信息，请查看 `SDS` 字段的内容。`SDS` 结构体中的 `Name` 字段给出了数据集的名称。

```
dsets = info.SDS

dsets =

    Filename: 'example.hdf'
      Type: 'Scientific Data Set'
    Name: 'Example SDS'
    Rank: 2
  DataType: 'int16'
  Attributes: []
    Dims: [2x1 struct]
    Label: {}
  Description: {}
    Index: 0
```

- 2 使用 `hdfread` 函数从 HDF4 文件读取数据集。将数据集的名称指定为函数的一个参数。请注意，数据集名称区分大小写。此示例返回一个 16×5 数组：

```
dset = hdfread('example.hdf', 'Example SDS')

dset =

     3     4     5     6     7
     4     5     6     7     8
     5     6     7     8     9
     6     7     8     9    10
     7     8     9    10    11
     8     9    10    11    12
     9    10    11    12    13
    10    11    12    13    14
    11    12    13    14    15
    12    13    14    15    16
    13    14    15    16    17
    14    15    16    17    18
    15    16    17    18    19
    16    17    18    19    20
    17    18    19    20    21
    18    19    20    21    22
```

也可以在由包含此信息的 `hdfinfo` 返回的结构体中指定该特定字段。例如，要读取科学数据集，请使用 `SDS` 字段。

```
dset = hdfread(info.SDS);
```

读取数据集中数据的子集

要读取数据集的子集，可以使用可选的 `'index'` 参数。`index` 参数的值是一个包含三个向量的元胞数组，这些向量指定数据集中开始读取的位置、跳过间隔（例如，每隔一个数据项读取）以及读取的数据量（例如，沿每个维度的长度）。在 HDF4 术语中，这些参数称为 `start`、`stride` 和 `edge` 值。

例如，下面的代码

- 在第 3 行第 3 列处开始读取数据 (`[3 3]`)。
- 读取数组中的每个元素 (`[]`)。
- 读取 10 行 2 列的内容 (`[10 2]`)。

```
subset = hdfread('Example.hdf','Example SDS',...
    'Index',{[3 3],[],[10 2]})
```

```
subset =
```

```

7    8
8    9
9   10
10   11
11   12
12   13
13   14
14   15
15   16
16   17
```


音频和视频

- “读取和写入音频文件” (第 8-2 页)
- “录制和播放音频” (第 8-4 页)
- “读取视频文件” (第 8-8 页)
- “支持的视频和音频文件格式” (第 8-12 页)
- “图像序列与视频之间的转换” (第 8-15 页)

读取和写入音频文件

将数据写入到音频文件，获取文件信息，然后将数据读回到 MATLAB 工作区。

写入音频文件

从文件 `handel.mat` 加载示例数据

```
load handel.mat
```

工作区现在包含音频数据矩阵 `y` 和采样率 `Fs`。

使用 `audiowrite` 函数将数据写入当前文件夹中名为 `handel.wav` 的 WAVE 文件。

```
audiowrite("handel.wav",y,Fs)
clear y Fs
```

`audiowrite` 函数也可以写入其他音频文件格式。有关支持格式的完整列表，请参阅“支持的导入和导出的文件格式”（第 1-2 页）。

获取有关音频文件的信息

使用 `audioinfo` 函数获取有关 WAVE 文件 `handel.wav` 的信息。

```
info = audioinfo("handel.wav")

info =
    Filename: 'pwd\handel.wav'
  CompressionMethod: 'Uncompressed'
    NumChannels: 1
    SampleRate: 8192
   TotalSamples: 73113
    Duration: 8.9249
        Title: []
       Comment: []
        Artist: []
   BitsPerSample: 16
```

`audioinfo` 返回一个 1×1 结构体数组。`SampleRate` 字段指示音频数据的采样率，以赫兹为单位。`Duration` 字段指示文件的持续时间，以秒为单位。

读取音频文件

使用 `audioread` 函数读取文件 `handel.wav`。`audioread` 函数可以支持其他文件格式。有关支持格式的完整列表，请参阅“支持的导入和导出的文件格式”（第 1-2 页）。

```
[y,Fs] = audioread("handel.wav");
```

播放音频。

```
sound(y,Fs)
```

您也可以通过交互方式读取文件。选择  **导入数据** 或双击当前文件夹浏览器中的文件名。

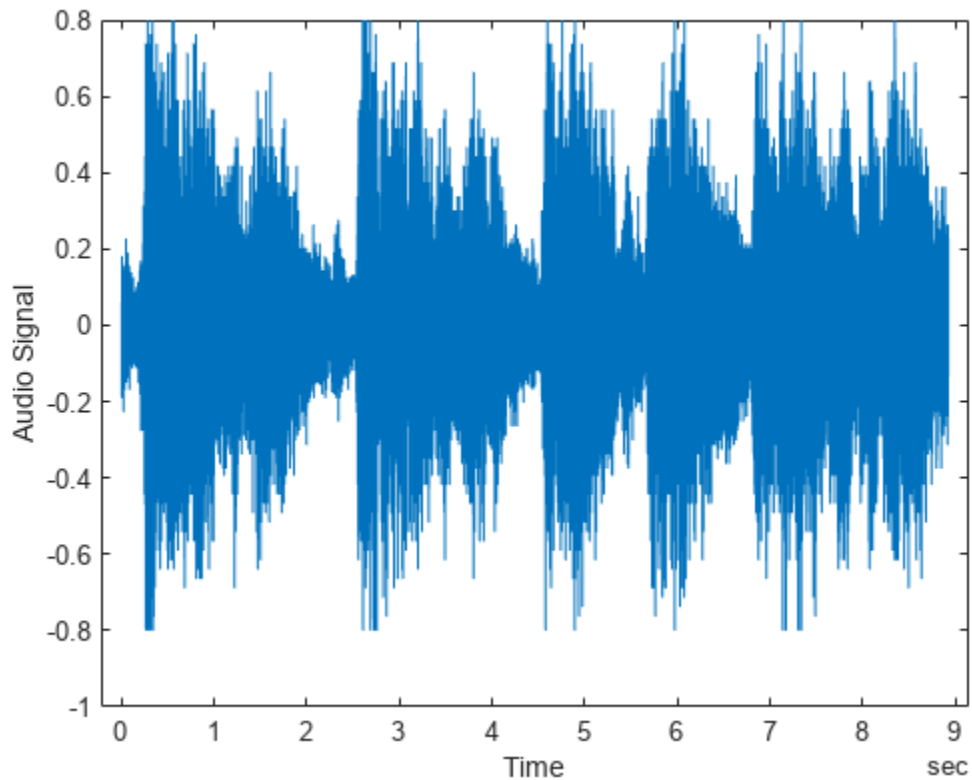
绘制音频数据图

创建一个与 `y` 长度相同的向量 `t`，表示经过的时间。

```
t = 0:seconds(1/Fs):seconds(info.Duration);  
t = t(1:end-1);
```

将音频数据绘制为时间的函数。

```
plot(t,y)  
xlabel('Time')  
ylabel('Audio Signal')
```



另请参阅

[audioinfo](#) | [audioread](#) | [audiowrite](#)

相关示例

- “以交互方式导入图像、音频和视频” (第 1-6 页)

录制和播放音频

通过系统音频输入和输出设备录制和播放音频数据，以便在 MATLAB 中进行处理。Google Chrome® 支持在 MATLAB Online 和 MATLAB Web App Server™ 中进行音频播放和音频录制。

本节内容

- “录制音频” (第 8-4 页)
- “播放音频” (第 8-6 页)
- “在函数内录制或播放音频” (第 8-6 页)

录制音频

从音频输入设备（例如与系统相连的麦克风）录制数据：

- 1 创建 **audiorecorder** 对象。
- 2 调用 **record** 或 **recordblocking** 方法，其中：
 - **record** 将当前的控制权返回给调用函数或命令提示符，甚至在录制进行过程中也是如此。可以指定录制时长（秒）或者使用 **stop** 方法结束录制。也可以选择调用 **pause** 和 **resume** 方法。录制以异步方式进行。
 - **recordblocking** 会一直保留控制权，直至录制完成。指定录制时长（秒）。录制以同步方式进行。
- 3 使用 **getaudiodata** 方法创建一个与信号数据对应的数值数组。

以下示例演示如何使用 **recordblocking** 和 **record** 方法。

录制麦克风输入

以下示例演示如何录制麦克风输入、播放录音，以及将录制的音频信号存储在数值数组中。您必须首先将麦克风连接到您的系统。

使用名为 **recObj** 的默认属性创建一个 **audiorecorder** 对象，用于录制音频输入。

```
recObj = audiorecorder
```

```
recObj =
```

```
audiorecorder with properties:
```

```
    SampleRate: 8000
    BitsPerSample: 8
    NumChannels: 1
    DeviceID: -1
    CurrentSample: 1
    TotalSamples: 0
    Running: 'off'
    StartFcn: []
    StopFcn: []
    TimerFcn: []
    TimerPeriod: 0.0500
    Tag: ''
    UserData: []
    Type: 'audiorecorder'
```


audiorecorder 创建 8000 Hz、8 位、1 通道的 **audiorecorder** 对象。

将您的语音录制 5 秒。

```
recDuration = 5;
disp("Begin speaking.")
recordblocking(recObj,recDuration);
disp("End of recording.")
```

播放该录音。

```
play(recObj);
```

将数据存储于双精度数组 **y** 中。

```
y = getaudiodata(recObj);
```

绘制音频样本图。

```
plot(y);
```

录制来自不同声卡的两个通道

要独立地录制来自两个不同的声卡（并且每个声卡都连接了一个麦克风）的音频：

- 1 调用 **audiodevinfo** 以列出可用的声卡。例如，以下代码返回一个结构体数组，其中包含系统上的所有输入和输出音频设备。

```
info = audiodevinfo;
```

通过名称确定您要使用的声卡，并记录其 ID 值。

- 2 创建两个 **audiorecorder** 对象。例如，以下代码创建 **audiorecorder** 对象 **recorder1**，用于以 44.1 kHz 和每样本 16 位的方式录制设备 3 的单个通道。然后，代码创建 **audiorecorder** 对象 **recorder2**，用于以 48 kHz 录制设备 4 的单个通道。

```
recorder1 = audiorecorder(44100,16,1,3);
recorder2 = audiorecorder(48000,16,1,4);
```

- 3 独立录制每个音频通道。

```
record(recorder1);
record(recorder2);
pause(5);
```

当对 **record** 的首个调用不排他时，会同时进行录制。

- 4 停止录制。

```
stop(recorder1);
stop(recorder2);
```

指定录制质量

默认情况下，**audiorecorder** 对象使用 8000 Hz 的采样率、8 位（每个样本 8 位）的深度和单个音频通道。使用这些设置时，所需的数据存储量很低。要获取更高质量的录音，可增加采样率或位深。

例如，光盘使用 44,100 Hz 的采样率、16 位位深和两个音频通道。创建一个 **audiorecorder** 对象以使用这些设置进行录制。

```
myRecObj = audiorecorder(44100,16,2);
```

有关可用的属性和值的详细信息，请参阅 `audiorecorder` 参考页。

播放音频

在导入或录制音频后，MATLAB 支持以多种方式收听数据：

- 对于使用单个函数调用的简单播放，可使用 `sound` 或 `soundsc`。例如，加载一个包含信号和采样率数据的样本 MAT 文件并收听音频。

```
load chirp.mat
sound(y,Fs)
```

- 要在播放过程中获得更大的灵活性，包括暂停、恢复或定义回调的功能，可使用 `audioplayer` 函数。创建一个 `audioplayer` 对象，然后调用播放音频的方法。例如，收听 `gong` 样本文件。

```
load gong.mat
gong = audioplayer(y,Fs);
play(gong);
```

有关其他示例，请参阅“在函数内录制或播放音频”（第 8-6 页）。

如果未指定采样率，则 `sound` 将以 8192 Hz 播放。对于任何播放，指定的采样率越小，则播放的速度越慢；采样率越大，则播放的速度越快。

注意 大多数声卡支持大约 5000 到 192,000 Hz 之间的采样率。指定此范围之外的采样率可能会产生意外结果。

在函数内录制或播放音频

如果在函数内部创建 `audioplayer` 或 `audiorecorder` 对象，则该对象仅在函数的持续时间内存在。例如，创建一个名为 `playFile` 的播放器函数和一个简单的回调函数 `showSeconds`。

```
function playFile(myfile)
    load(myfile)

    obj = audioplayer(y,Fs);
    obj.TimerFcn = 'showSeconds';
    obj.TimerPeriod = 1;

    play(obj);
end

function showSeconds
    disp("tick")
end
```

从命令提示符调用 `playFile` 以播放文件 `handel.mat`。

```
playFile("handel.mat")
```

按照录音采样率（每秒 8192 个样本），播放文件中的 73.113 个样本大约需要 8.9 秒。但是，`playFile` 函数通常会在播放完成前结束，并清除 `audioplayer` 对象 `obj`。

对于完整的播放或录制，请考虑以下选项：

- 使用 **playblocking** 或 **recordblocking** 而不是 **play** 或 **record**。排他方法会一直保留控制权，直至播放或录制完成。如果对控制权实施排他，则无法在播放或录制过程发出任何其他命令或方法（例如 **pause** 或 **resume**）。
- 为在基础工作区中生成对象的函数创建一个输出参数。例如，修改 **playFile** 函数以包括输出参数。

```
function obj = playFile(myfile)
```

调用函数。

```
h = playFile("handel.mat");
```

因为 **h** 存在于基础工作区中，所以可以从命令提示符暂停播放。

```
pause(h)
```

另请参阅

audioplayer | **sound** | **soundsc** | **audiorecorder**

详细信息

- “读取和写入音频文件” （第 8-2 页）

读取视频文件

从特定时间或帧索引开始读取视频帧，读取指定区间内的帧，或读取视频中的所有帧。

从指定时间或帧索引开始读取帧

从距离视频文件开头 0.5 秒处开始读取文件的一部分。然后，从帧索引 100 开始读取到视频文件结束。

构造一个与样本文件 'xylophone.mp4' 关联的 `VideoReader` 对象。

```
vidObj = VideoReader('xylophone.mp4');
```

通过设置 `CurrentTime` 属性，指定应从距离文件开头 0.5 秒处开始读取。

```
vidObj.CurrentTime = 0.5;
```

使用 `readFrame` 方法读取视频帧，直至到达文件结束。

```
while hasFrame(vidObj)
    vidFrame = readFrame(vidObj);
    imshow(vidFrame)
    pause(1/vidObj.FrameRate);
end
```



您也可以使用 `read` 方法从指定的帧索引开始读取视频帧，直到视频结束。将要读取的索引指定为 `[100 Inf]`。`read` 方法返回从索引 100 开始到视频文件结束的所有帧。

```
vidframes = read(vidObj,[100 Inf]);
```

读取指定区间内的帧

通过指定时间或帧区间来读取视频文件的一部分。

读取 0.6 到 0.9 秒之间的视频帧。首先，创建一个视频读取器对象和一个用来保存帧的结构体数组。

```
vidObj = VideoReader('xylophone.mp4');
s = struct('cdata',zeros(vidObj.Height,vidObj.Width,3,'uint8'),'colormap',[]);
```

然后，通过设置 `CurrentTime` 属性，指定应从距离文件开头 0.6 秒处开始读取。

```
vidObj.CurrentTime = 0.6;
```

一次读取一帧，直至 `CurrentTime` 到达 0.9 秒处。将每个视频帧中的数据追加到结构体数组。查看结构体数组中的帧数。`s` 是 1×10 结构体，表示读取了 10 个帧。有关将结构体 `s` 中的帧显示为影片的信息，请参阅 `movie` 函数参考页。

```
k = 1;
while vidObj.CurrentTime <= 0.9
    s(k).cdata = readFrame(vidObj);
    k = k+1;
end
whos s
```

Name	Size	Bytes	Class	Attributes
s	1x10	2305344	struct	

您也可以通过使用帧索引来读取指定区间内的所有帧。例如，将 `read` 的第二个参数指定为 `[18 27]`。`read` 方法返回 `FrameSize×10` 数组，表示读取了 10 个帧。

```
frames = read(vidObj,[18 27]);
whos frames
```

Name	Size	Bytes	Class	Attributes
frames	240x320x3x10	2304000	uint8	

读取所有帧

从视频中读取所有帧，一次读取一帧或一次读取所有帧。

创建一个视频读取器对象，并显示视频中的总帧数。

```
vidObj = VideoReader('xylophone.mp4');
vidObj.NumFrames
```

```
ans = 141
```

使用 `readFrame` 方法读取所有帧，一次读取一帧，并显示这些帧。

```
while hasFrame(vidObj)
    frame = readFrame(vidObj);
    imshow(frame)
    pause(1/vidObj.FrameRate);
end
```



您也可以一次读取所有视频帧。`read` 方法返回由视频帧组成的 `FrameSize×141` 数组。

```
allFrames = read(vidObj);  
whos allFrames
```

Name	Size	Bytes	Class	Attributes
allFrames	240x320x3x141	32486400	uint8	

视频读取的故障排除和提示

- 当 `CurrentTime` 属性的值等于 `Duration` 属性的值时，`hasFrame` 方法可能会返回逻辑值 1 (true)。这是由于所使用的基础 API 的局限性所致。
- 不推荐通过将 `CurrentTime` 属性设置为接近于 `Duration` 值的值来查找视频文件中的最后一帧。对于某些文件，即使 `CurrentTime` 值小于 `Duration` 值，该操作也会返回错误，即指示已到达文件结尾。如果文件持续时间长于视频流的持续时间，并且在靠近文件结尾时没有可读取的视频，通常会发生此情况。
- 不推荐使用 `Duration` 属性限制从视频文件中读取数据。请使用 `hasFrame` 方法检查是否存在可读取的帧。最好是连续读取数据，直至文件报告不再存在可读取的帧。
- Windows® 系统上的视频读取性能：为了在 Windows 上达到更好的 MP4 和 MOV 视频文件读取性能，MATLAB® 使用系统的图形硬件进行解码。但是，在某些情况下，使用图形卡进行解码可能会降低系统性能，具体取决于系统上的特定图形硬件。如果您注意到系统上的视频读取性能降低，可以通过键入以下命令来禁用硬件加速：`matlab.video.read.UseHardwareAcceleration('off')`。您可以通过键入以下命令重新启用硬件加速：`matlab.video.read.UseHardwareAcceleration('on')`。

另请参阅

`VideoReader` | `mmfileinfo` | `movie` | `read` | `readFrame`

详细信息

- “支持的视频和音频文件格式” (第 8-12 页)

支持的视频和音频文件格式

MATLAB 中的视频和音频文件及其支持的文件格式和编解码器。

MATLAB 中的视频数据

什么是视频文件？

对于视频数据而言，“文件格式”通常指的是容器格式或编解码器。容器格式描述文件的布局，而编解码器则描述如何对视频数据进行编码/解码。许多容器格式可以容纳使用不同编解码器编码的数据。

要读取视频文件，任何应用程序都必须：

- 能够识别容器格式（例如 AVI）。
- 可以访问能够对文件中所存储的视频数据进行解码的编解码器。标准 Windows 和 Macintosh 系统安装中包含了一些编解码器，您可以播放 Windows Media Player 或 QuickTime® 格式的视频。在 MATLAB 中，**VideoReader** 可访问其中大多数（但不是全部）编解码器。
- 正确使用编解码器对文件中的视频数据进行解码。**VideoReader** 不一定始终能读取与原始系统安装中未包含的编解码器相关联的文件。

VideoReader 支持的格式

在 MATLAB 中使用 **VideoReader** 读取视频文件。**VideoReader** 支持的文件格式视平台而异，对文件扩展名没有任何限制。

平台	文件格式
所有平台	AVI，包括未压缩、索引、灰度和 Motion JPEG 编码的视频 (.avi) Motion JPEG 2000 (.mj2)
所有 Windows	MPEG-1 (.mpg) Windows Media 视频 (.wmv、.asf) Microsoft DirectShow 支持的任何格式
Windows 7 或更高版本	MPEG-4，包括 H.264 编码视频 (.mp4、.m4v) Apple QuickTime 影片 (.mov) Microsoft Media Foundation 支持的任何格式
Macintosh	QuickTime Player 支持的大多数格式，包括： MPEG-1 (.mpg) MPEG-4，包括 H.264 编码视频 (.mp4、.m4v) Apple QuickTime 影片 (.mov) 3GPP 3GPP2 AVCHD DV 注意： 对于 OS X Yosemite (10.10 版) 和更高版本来说，使用 VideoWriter 编写的 MPEG-4/H.264 文件能正常播放，但显示的帧速率不精确。

平台	文件格式
Linux	GStreamer 1.0 或更高版本的已安装插件支持的任何格式，如 https://gstreamer.freedesktop.org/documentation/plugins_doc.html 中所列，包括 Ogg Theora (.ogg)。

查看视频文件所关联的编解码器

此示例说明如何使用 `mmfileinfo` 函数查看视频文件所关联的编解码器。

将有关样本视频文件 `shuttle.avi` 的信息存储在一个名为 `info` 的结构体数组中。`info` 结构体包含以下字段：`Filename`、`Path`、`Duration`、`Audio` 和 `Video`。

```
info = mmfileinfo('shuttle.avi');
```

通过在命令行窗口中显示 `info` 结构体的字段来显示属性。例如，要查看 `Video` 字段下的信息，请键入 `info.Video`

```
info.Video
ans = struct with fields:
    Format: 'Motion JPEG'
    Height: 288
    Width: 512
```

文件 `shuttle.avi` 使用 Motion JPEG 编解码器。

故障排除：读取视频文件时出错

如果 MATLAB 无法访问相应的编解码器，您将无法读取视频文件。64 位应用程序使用 64 位编解码器库，32 位应用程序则使用 32 位编解码器库。例如，使用 64 位 MATLAB 时，您将无法读取需要访问系统上安装的 32 位编解码器的视频文件。要读取这些文件，请尝试以下方法之一：

- 安装支持此文件格式的 64 位编解码器。然后，尝试使用 64 位 MATLAB 读取文件。
- 使用您的计算机上安装的 64 位编解码器将文件重新编码为其他格式。

有时，`VideoReader` 无法在 Windows 平台上打开视频文件进行读取。如果您安装的第三方编解码器覆盖了系统设置，就可能会出现这种情况。卸载该编解码器，然后再次尝试在 MATLAB 中打开视频文件。

MATLAB 中的音频数据

什么是音频文件？

文件中的音频信号代表一系列样本，其中包含随时间变化的声音幅值。采样率是每秒采集的离散样本数量，以赫兹为单位。样本精度按位深（采样位数）测量，具体取决于可用的音频硬件。

MATLAB 音频函数使用 $m \times 1$ 列向量读取和存储单通道（单声道）音频数据，使用 $m \times 2$ 矩阵读取和存储立体声数据。在这两种情况下， m 均为样本数量。对于立体声数据，第一列包含左声道，第二列包含右声道。

通常，每个样本都是介于 -1 和 1 之间的双精度值。在某些情况下，尤其当音频硬件不支持高位深时，音频文件会以 8 位或 16 位整数形式存储值。样本值的范围取决于可用的位数。例如，存储为 `uint8` 值的样本

范围为 0 到 255 ($2^8 - 1$)。MATLAB `sound` 和 `soundsc` 函数仅支持介于 -1 和 1 之间的单精度或双精度值。其他音频函数支持多种数据类型，请参阅函数参考页了解详情。

audioread 支持的格式

在 MATLAB 中使用 `audioread` 读取音频文件。`audioread` 函数支持以下文件格式。

平台支持	文件格式
所有平台	AIFC (.aifc)
	AIFF (.aiff、.aif)
	AU (.au)
	FLAC (.flac)
	OGG (.ogg)
	OPUS (.opus)
	WAVE (.wav)
Windows 7（或更高版本）、Macintosh 和 Linux	MP3 (.mp3)
	MPEG-4 AAC (.m4a、.mp4)

在低于 Windows 7 的 Windows 平台上，`audioread` 不读取包含 MP3 编码数据的 WAVE 文件。

在 Windows 7（或更高版本）平台中，`audioread` 可能还读取 Windows Media Foundation 支持的任何文件。

在 Linux 平台中，`audioread` 可能还读取 GStreamer 支持的任何文件。

`audioread` 可以从 Windows 7 或更高版本、Macintosh 和 Linux 上的 MPEG-4 (.mp4、.m4v) 视频文件、从 Windows 7（或更高版本）和 Linux 平台上的 Windows Media 视频 (.wmv) 和 AVI (.avi) 文件中提取音频。

另请参阅

`VideoReader` | `audioread` | `mmfileinfo` | `audioinfo`

详细信息

- “读取视频文件”（第 8-8 页）
- “读取和写入音频文件”（第 8-2 页）

图像序列与视频之间的转换

使用 `VideoReader` 和 `VideoWriter` 在视频文件和图像文件序列之间转换。

名为 `shuttle.avi` 的示例文件包含 121 帧。使用 `VideoReader` 和 `imwrite` 函数将这些帧转换为图像文件。然后，使用 `VideoWriter` 将图像文件转换为 AVI 文件。

设置

创建一个用于存储图像序列的临时工作文件夹。

```
workingDir = tempname;
mkdir(workingDir)
mkdir(workingDir,'images')
```

创建 VideoReader

创建一个用于从文件中读取帧的 `VideoReader`。

```
shuttleVideo = VideoReader('shuttle.avi');
```

创建图像序列

循环读取视频，将所有帧都读入到名为 `img` 的一个宽×高×3 的数组中。将每个图像写入一个以 `imgN.jpg` 格式命名的 JPEG 文件，其中 `N` 是帧编号。

```
| img001.jpg|
| img002.jpg|
| ...|
| img121.jpg|

ii = 1;

while hasFrame(shuttleVideo)
    img = readFrame(shuttleVideo);
    filename = [sprintf('%03d',ii) '.jpg'];
    fullname = fullfile(workingDir,'images',filename);
    imwrite(img,fullname) % Write out to a JPEG file (img1.jpg, img2.jpg, etc.)
    ii = ii+1;
end
```

查找图像文件名称

找到 `images` 文件夹中的所有 JPEG 文件名。将这组图像名称转换为元胞数组。

```
imageNames = dir(fullfile(workingDir,'images','*.jpg'));
imageNames = {imageNames.name};
```

使用图像序列创建新视频

构造一个 `VideoWriter` 对象，默认情况下该对象会创建一个 Motion-JPEG AVI 文件。

```
outputVideo = VideoWriter(fullfile(workingDir,'shuttle_out.avi'));  
outputVideo.FrameRate = shuttleVideo.FrameRate;  
open(outputVideo)
```

循环读取图像序列，加载每个图像，然后将其写入到视频。

```
for ii = 1:length(imageNames)  
    img = imread(fullfile(workingDir,'images',imageNames{ii}));  
    writeVideo(outputVideo,img)  
end
```

最终生成视频文件。

```
close(outputVideo)
```

查看最终生成的视频

构造一个 reader 对象。

```
shuttleAvi = VideoReader(fullfile(workingDir,'shuttle_out.avi'));
```

根据视频帧创建一个 MATLAB® 影片结构体。

```
ii = 1;  
while hasFrame(shuttleAvi)  
    mov(ii) = im2frame(readFrame(shuttleAvi));  
    ii = ii+1;  
end
```

基于视频的宽度和高度调整当前图窗及其坐标区的大小，并查看影片的第一帧。

```
figure  
imshow(mov(1).cdata, 'Border', 'tight')
```

按照视频的帧速率播放影片一次。

```
movie(mov,1,shuttleAvi.FrameRate)
```



感谢

感谢 NASA 提供的航天飞机视频。

XML 文档

- “将 XML 文件导入文档对象模型中” (第 9-2 页)
- “将文档对象模型导出为 XML 文件” (第 9-6 页)

将 XML 文件导入文档对象模型中

您可以使用 `matlab.io.xml.dom.Parser` 对象或 `xmlread` 函数将 XML 文件导入文档对象模型 (DOM) 文档节点。

`matlab.io.xml.dom.Parser` 类属于用于 XML 处理的 MATLAB API (MAXP)。当您使用 MAXP Parser 对象读取 XML 文件时，生成的 DOM 文档节点表示为 `matlab.io.xml.dom.Document` 对象。有关可用于处理 Document 对象的类的列表，请参阅 `matlab.io.xml.dom`。您不需要 Java® 软件便可使用 MAXP 类。

要使用由 `xmlread` 创建的 DOM 文档节点对象，您必须使用用于 XML 处理的 Java API (JAXP)。有关 JAXP 方法和属性的列表，请参阅 <https://docs.oracle.com/javase/7/docs/api> 上提供的 `org.w3c.dom` 包说明。

XML 文档对象模型

在文档对象模型中，XML 文件中的每一项都对应于一个节点。您用于创建和访问节点的属性和方法遵循万维网联合会设定的标准。

例如，考虑以下样本 XML 文件：

```
<productinfo>

<!-- This is a sample info.xml file. -->

<list>

<listitem>
<label color="blue">Import Wizard</label>
<callback>uiimport</callback>
<icon>ApplicationIcon.GENERIC_GUI</icon>
</listitem>

<listitem>
<label color="red">Profiler</label>
<callback>profile viewer</callback>
<icon>ApplicationIcon.PROFILER</icon>
</listitem>

</list>
</productinfo>
```

文件中的信息映射到以下类型的 DOM 节点：

- 元素节点 - 对应于标记名称。在 `info.xml` 文件中，这些标记对应于元素节点：
 - `productinfo`
 - `list`
 - `listitem`
 - `label`
 - `callback`
 - `icon`

在此情况下，`list` 元素是 `listitem` 元素子节点的父级。`productinfo` 元素是根元素节点。

- 文本节点 - 包含与元素节点关联的值。每个文本节点都是一个元素节点的子节点。例如，`Import Wizard` 文本节点是第一个 `label` 元素节点的子节点。

- 属性节点 - 包含与元素节点关联的名称-值对组。例如，在第一个 `label` 元素节点中，`color` 是属性的名称，`blue` 是其值。属性节点不是任何节点的父节点或子节点。
- 注释节点 - 在文件中包含附加文本，格式为 `<!--Sample comment-->`。
- 文档节点 - 对应于整个文件。使用文档节点上的方法可创建新的元素、文本、属性或注释节点。

使用 MAXP 解析器读取 XML 文件

此示例使用 `matlab.io.xml.dom.Parser` 对象将 `info.xml` 文件读入 `matlab.io.xml.dom.Document` 节点。该文件包含几个 `listitem` 元素。每个 `listitem` 元素包含一个 `label` 和 `callback` 元素。该示例使用 MAXP 方法来查找 `callback` 元素的文本内容，该元素对应于具有文本内容 Plot Tools 的 `label`。

将该文件读入一个 `Document` 对象中。

```
infoFile = fullfile(matlabroot,'toolbox/matlab/general/info.xml');
infoLabel = 'Plot Tools';
infoCbK = '';
itemFound = false;
```

```
import matlab.io.xml.dom.*
xDoc = parseFile(Parser,infoFile);
```

通过调用 `getElementsByTagName` 方法查找所有 `listitem` 元素，该方法返回 `matlab.io.xml.dom.NodeList` 对象。

```
allListItems = getElementsByTagName(xDoc,'listitem');
```

对于每个 `listitem` 元素，都会将 `label` 元素的文本与 Plot Tools 进行比较。当找到正确的标签时，将获取 `callback` 的文本。要访问 `NodeList` 对象中的元素，请使用 `node` 方法，该方法使用从 1 开始的索引。您也可以使用 `item` 方法，该方法使用从 0 开始的索引。

```
length = allListItems.Length;
for i=1:length

    thisListItem = node(allListItems,i);
    childNode = getFirstChild(thisListItem);

    while ~isempty(childNode)
        %Filter out text, comments, and processing instructions.

        if isa(childNode,'matlab.io.xml.dom.Element')
            %Assume that each element has a single Text child

            childText = getData(getFirstChild(childNode));

            switch getTagName(childNode)
                case 'label'
                    itemFound = strcmp(childText,infoLabel);
                case 'callback'
                    infoCbK = childText;
            end
            childNode = getNextSibling(childNode);
        end
        if itemFound
            break
        else

```

```

        infoCbk = "";
    end
end

```

显示结果。

```
fprintf('Item "%s" has a callback of "%s".\n', infoLabel,infoCbk);
```

Item "Plot Tools" has a callback of "figure; plottools".

使用 xmlread 读取 XML 文件

此示例使用 `xmlread` 将 `info.xml` 文件读入 DOM 文档节点，并使用用于 XML 处理的 Java API 方法来查找 `callback` 元素的文本内容，该元素对应于具有文本内容 `label` 的 `Plot Tools`。

```

infoFile = fullfile(matlabroot,'toolbox/matlab/general/info.xml');
infoLabel = 'Plot Tools';
infoCbk = "";
itemFound = false;

xDoc = xmlread(infoFile);

allListItems = getElementsByTagName(xDoc,'listitem');

%The item list index is zero-based.
length = allListItems.getLength-1;
for i=0:length

    thisListItem = item(allListItems,i);
    childNode = getFirstChild(thisListItem);

    while ~isempty(childNode)
        %Filter out text, comments, and processing instructions.

        if childNode.getNodeType == childNode.ELEMENT_NODE
            %Assume that each element has a single org.w3c.dom.Text child

            childText = char(childNode.getFirstChild.getData);

            switch char(childNode.getTagName)
                case 'label'
                    itemFound = strcmp(childText,infoLabel);
                case 'callback'
                    infoCbk = childText;
            end
            childNode = getNextSibling(childNode);
        end
        if itemFound
            break
        else
            infoCbk = "";
        end
    end
end
fprintf('Item "%s" has a callback of "%s".\n', infoLabel,infoCbk);

```

Item "Plot Tools" has a callback of "figure; plottools".

另请参阅

`matlab.io.xml.dom.Document` | `xmlread`

相关示例

- “将文档对象模型导出为 XML 文件” (第 9-6 页)

外部网站

- <https://docs.oracle.com/javase/7/docs/api>

将文档对象模型导出为 XML 文件

您可以使用 `matlab.io.xml.dom.DOMWriter` 对象或 `xmlwrite` 函数将文档对象模型 (DOM) 文档节点导出为 XML 文件。

`matlab.io.xml.dom.DOMWriter` 类属于用于 XML 处理的 MATLAB API (MAXP)。要使用 MAXP `DOMWriter` 对象，请将 DOM 文档节点表示为 `matlab.io.xml.dom.Document` 对象。要创建元素、文本和其他节点并将其添加到文档节点，请使用 MAXP 类和方法。请参阅 `matlab.io.xml.dom`。您不需要 Java 软件便可使用 MAXP 类。

要创建可以使用 `xmlwrite` 进行编写的 DOM 文档，请使用 `com.mathworks.xml.XMLUtils.createDocument`。要创建节点并将其添加到文档节点，请使用用于 XML 处理的 Java API (JAXP) 的方法。请参阅 [https://docs.oracle.com/javase/7/docs/api](https://docs.oracle.com/javase/7/docs/api/org.w3c.dom) 上提供的 `org.w3c.dom` 包说明。

创建 DOM 文档

创建 XML 文档的通用步骤包括：

- 1 创建文档节点并定义根元素。以下代码通过创建 MAXP `matlab.io.xml.dom.Document` 对象来创建文档节点：

```
import matlab.io.xml.dom.*
docNode = Document('root_element');
```

以下代码创建一个可与 JAXP 方法结合使用的文档节点：

- ```
docNode = com.mathworks.xml.XMLUtils.createDocument('root_element');
```
- 2 通过调用 `getDocumentElement`，获取与根元素对应的节点。添加子节点需要根元素节点。
  - 3 通过调用文档节点的方法，添加元素、文本、注释和属性节点。有用的方法包括：

- `createElement`
- `createTextNode`
- `createComment`
- `setAttribute`

- 4 要将子节点追加到父节点，请使用 `appendChild`。

---

**提示** 文本节点始终是元素节点的子节点。要添加文本节点，请将 `createTextNode` 用于文档节点，然后将 `appendChild` 用于父元素节点。

---

### 使用 MAXP DOMWriter 对象将 DOM 文档节点写入 XML 文件

此示例使用 `matlab.io.xml.dom.DOMWriter` 对象为 Upslope Area Toolbox 创建一个 `info.xml` 文件，如“显示自定义文档”中所述。

创建文档节点和根元素 `toc`。

```
import matlab.io.xml.dom.*
docNode = Document('toc');
```

获得根元素并设置 `version` 属性。

```
toc = docNode.getDocumentElement;
setAttribute(toc,'version','2.0');
```

为产品页添加 tocitem 元素。此文件中的每个 tocitem 元素都有一个 target 属性和一个子文本节点。

```
product = createElement(docNode,'tocitem');
setAttribute(product,'target','upslope_product_page.html');
appendChild(product,createTextNode(docNode,'Upslope Area Toolbox'));
appendChild(toc,product);
```

添加注释。

```
appendChild(product,createComment(docNode,' Functions '));
```

为每个函数添加一个 tocitem 元素节点。

```
functions = {'demFlow','facetFlow','flowMatrix','pixelFlow'};
n = numel(functions);
for idx = 1:n
 curr_node = createElement(docNode,'tocitem');
 curr_file = [functions{idx} '_help.html'];
 setAttribute(curr_node,'target',curr_file);

 % Child text is the function name.
 appendChild(curr_node,createTextNode(docNode,functions{idx}));
 appendChild(product,curr_node);
end
```

将 DOM 节点导出至 info.xml，并查看文件。

```
xmlFileName = 'info.xml';
writer = matlab.io.xml.dom.DOMWriter;
writer.Configuration.FormatPrettyPrint = true;
```

```
writeToFile(writer,docNode,xmlFileName);
```

```
type('info.xml');
```

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<toc version="2.0">
```

```
 <tocitem target="upslope_product_page.html">Upslope Area Toolbox
 <!-- Functions -->
 <tocitem target="demFlow_help.html">demFlow</tocitem>
 <tocitem target="facetFlow_help.html">facetFlow</tocitem>
 <tocitem target="flowMatrix_help.html">flowMatrix</tocitem>
 <tocitem target="pixelFlow_help.html">pixelFlow</tocitem>
 </tocitem>
```

```
</toc>
```

## 使用 xmlwrite 将 DOM 文档节点写入 XML 文件

此示例使用 xmlwrite 为 Upslope Area Toolbox 创建一个 info.xml 文件，如“显示自定义文档”中所述。

```
docNode = com.mathworks.xml.XMLUtils.createDocument('toc');
toc = docNode.getDocumentElement;
```

```

toc.setAttribute('version','2.0');
product = docNode.createElement('tocitem');
product.setAttribute('target','upslope_product_page.html');
product.appendChild(docNode.createTextNode('Upslope Area Toolbox'));
toc.appendChild(product)
product.appendChild(docNode.createComment(' Functions '));
functions = {'demFlow','facetFlow','flowMatrix','pixelFlow'};
for idx = 1:numel(functions)
 curr_node = docNode.createElement('tocitem');

 curr_file = [functions{idx} '_help.html'];
 curr_node.setAttribute('target',curr_file);

 % Child text is the function name.
 curr_node.appendChild(docNode.createTextNode(functions{idx}));
 product.appendChild(curr_node);
end
xmlwrite('info.xml',docNode);
type('info.xml');

<?xml version="1.0" encoding="utf-8"?>
<toc version="2.0">
 <tocitem target="upslope_product_page.html">Upslope Area Toolbox<!-- Functions --><tocitem target="demFlow_help.html">demFlow</tocitem>
 <tocitem target="facetFlow_help.html">facetFlow</tocitem>
 <tocitem target="flowMatrix_help.html">flowMatrix</tocitem>
 <tocitem target="pixelFlow_help.html">pixelFlow</tocitem>
</tocitem>
</toc>

```

## 更新现有的 XML 文件

要更改现有文件中的数据，请执行下列步骤：

- 1 使用 `matlab.io.xml.dom.Parser` 对象或 `xmlread` 将文件导入 DOM 文档节点中。
- 2 遍历该节点并使用下列方法添加或更改数据：
  - `getElementsByTagName`
  - `getFirstChild`
  - `getNextSibling`
  - `getNodeName`
  - `getNodeType`

如果您使用 `matlab.io.xml.dom.Parser` 将 XML 文件读入 `matlab.io.xml.dom.Document` 中，请使用用于 XML 处理的 MATLAB API (MAXP) 类和方法。请参阅 `matlab.io.xml.dom`。如果您使用 `xmlread`，请使用用于 XML 处理的 Java API (JAXP) 方法。请参阅 <https://docs.oracle.com/javase/7/docs/api> 上提供的 `org.w3c.dom` 包说明。

- 3 当 DOM 文档包含您的所有更改时，请编写该文件。对于 MAXP DOM 文档，请使用 `matlab.io.xml.DOMWriter` 对象。对于 JAXP DOM 文档，请使用 `xmlwrite`。

## 另请参阅

`matlab.io.xml.dom.Document` | `matlab.io.xml.dom.DOMWriter` | `xmlwrite`

## 相关示例

- “将 XML 文件导入文档对象模型中” (第 9-2 页)

## 外部网站

- <https://docs.oracle.com/javase/7/docs/api>





## 内存映射数据文件

---

# 内存映射概述

本节内容
“什么是内存映射？” (第 10-2 页)
“内存映射的优势” (第 10-2 页)
“何时使用内存映射” (第 10-3 页)
“内存映射的最大大小” (第 10-4 页)
“字节排序” (第 10-4 页)

## 什么是内存映射？

内存映射是将磁盘上某文件的一部分或整个文件映射到应用程序地址空间内某个地址范围的一种机制。然后，应用程序可采用与访问动态内存相同的方法访问磁盘上的文件。与使用 `fread` 和 `fwrite` 等函数相比，这会加快文件的读取和写入速度。

## 内存映射的优势

内存映射的主要优势体现在效率、更快的文件访问速度、能够在应用程序之间共享内存，以及更高效的编码。

### 更快的文件访问速度

通过内存映射访问文件的速度比使用 `fread` 和 `fwrite` 等 I/O 函数访问文件更快。系统使用操作系统内置的虚拟内存功能读取和写入数据，而不必分配数据缓冲区，将数据复制到数据缓冲区，然后取消分配进程所拥有的数据缓冲区。

在先行构造了映射的情况下，MATLAB 不会从磁盘访问数据。它仅在内存映射的指定部分被访问时才会读取或写入磁盘上的文件，之后便仅读取该特定部分。这可以提高对映射数据的随机访问速度。

### 效率

将文件映射到内存后，访问文件中的数据时就如同该数据已被读入到应用程序地址空间中的数组。最初，MATLAB 仅为该数组分配地址空间；在您访问映射的区域之前，它不会从文件中实际读取数据。因此，利用内存映射的文件所提供的机制，应用程序无需先将整个文件读入内存，便可访问超大型文件中的数据段。

### 高效的编码样式

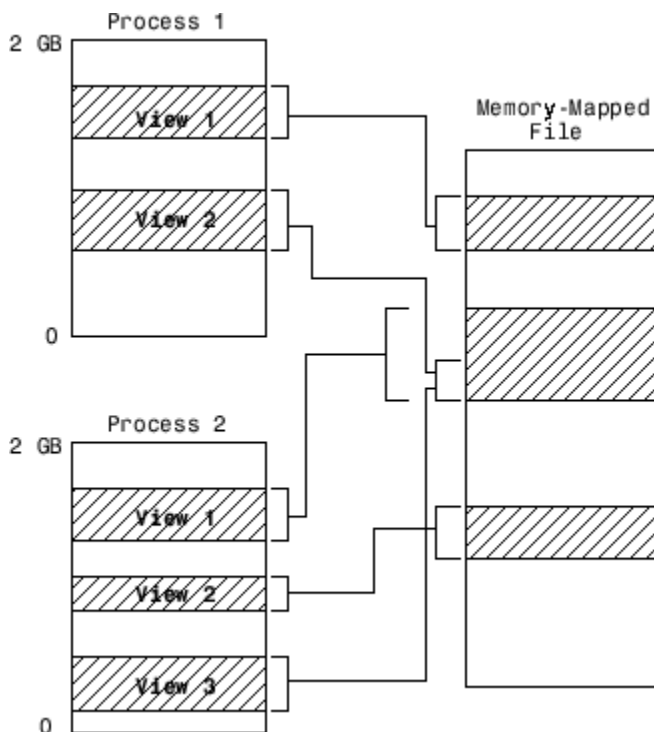
通过 MATLAB 应用程序中的内存映射，您可以使用标准 MATLAB 索引操作来访问文件数据。将文件映射到内存后，您可以使用与读取 MATLAB 工作区中变量所用的相同类型的 MATLAB 语句，来读取该文件的内容。所映射文件的内容就像是当前活动工作区的数组一样。您只需创建此数组的索引，便可在文件中读取或写入所需的数据。因此，您不需要显式调用 `fread` 和 `fwrite` 函数。

在 MATLAB 中，如果 `x` 是内存映射变量，`y` 是要写入到文件的数据，则只需执行以下命令即可写入到文件：

```
x.Data = y;
```

## 在应用程序之间共享内存

内存映射文件还提供了在应用程序之间共享数据的机制（如下图所示）。实现方法是让各个应用程序映射同一文件的多个部分。您可以使用此功能在 MATLAB 与其他应用程序之间传输大型数据集。



此外，在单一应用程序内，您也可以多次映射文件的同一数据段。

## 何时使用内存映射

通过将文件映射到内存可以获得多大的优势，这主要取决于该文件的大小和格式、文件中数据的使用方式，以及您所使用的计算机平台。

### 内存映射何时最有用

内存映射最适合在以下情形中用于二进制文件：

- 对于需要随机访问一次或多次的大型文件
- 对于需要一次读入内存然后频繁访问的小文件
- 对于需要在应用程序之间共享的数据
- 需要将文件中的数据当作 MATLAB 数组一样进行处理时

### 何时优势不太明显

以下文件类型无法充分利用内存映射的优势：

- 需要不适合进行内存映射的自定义读取器的二进制文件，例如 HDF 或 TIFF。描述这些文件中包含的数据可能是一项非常复杂的任务。此外，您也不能直接访问映射段中的数据，而是必须创建数组来保存数据。

- 需要将映射区内的文本转换为相应类型才能让数据有意义的文本文件或 ASCII 文件。这会占用额外的地址空间。
- 大小超过几百 MB 的文件，这些文件会消耗大量虚拟地址空间，而 MATLAB 在处理您的程序时需要这些空间。映射如此大的文件可能导致 MATLAB 更频繁地报告内存不足的错误。当 MATLAB 已运行一段时间或 MATLAB 所用的内存变得碎片化时，出现这种情况的可能性更高。

### 内存映射的最大大小

由于操作系统和 MATLAB 所设的限制，使用单一内存映射实例可以映射的最大数据量为 2 GB（32 位系统）和 256 TB（64 位系统）。如果需要映射的数据量超过此限制，您可以针对文件的不同区域创建单独的映射，也可以将一个映射的窗口移至文件中的不同位置。

### 字节排序

内存映射仅适用于字节排序方案与操作系统本机字节排序相同的数据。例如，由于 Linus Torvalds 的 Linux 和 Microsoft Windows 系统均使用 little-endian 字节排序，因此在 Linux 系统上创建的数据可以在 Windows 系统上读取。您可以使用 `computer` 函数来确定当前系统的本机字节排序。

## Internet 文件访问和 JSON

---

## **MATLAB 和 Web 服务安全**

本主题描述 MATLAB 如何处理 Web 服务的安全性。有关计算机安全的完整说明，需要咨询外部资源。

### **MATLAB 不校证书链**

对于 HTTPS 连接，`webread`、`webwrite` 和 `websave` 函数校证书域是否与 Web 服务的主机名匹配。这些函数不校证书链。有关计算机安全的完整说明，需要咨询外部资源。

### **另请参阅**

`webread` | `webwrite` | `websave`

## 从 Web 服务下载数据

以下示例说明如何使用 `webread` 函数从 Web 服务下载数据。世界银行通过世界银行气候数据 API 提供各种气候数据。调用此 API 将返回 JSON 格式的数据。`webread` 则将 JSON 对象转换为方便在 MATLAB 中进行分析的结构体。

使用 `webread` 将美国年平均温度读入一个结构体数组。

```
api = 'http://climatedataapi.worldbank.org/climateweb/rest/v1/';
url = [api 'country/cru/tas/year/USA'];
S = webread(url)
```

S =

112x1 struct array with fields:

```
year
data
```

`webread` 将该数据转换为一个包含 112 个元素的结构体数组。每个结构体包含 1901 年到 2012 年期间一个给定年份的温度。

S(1)

ans =

```
year: 1901
data: 6.6187
```

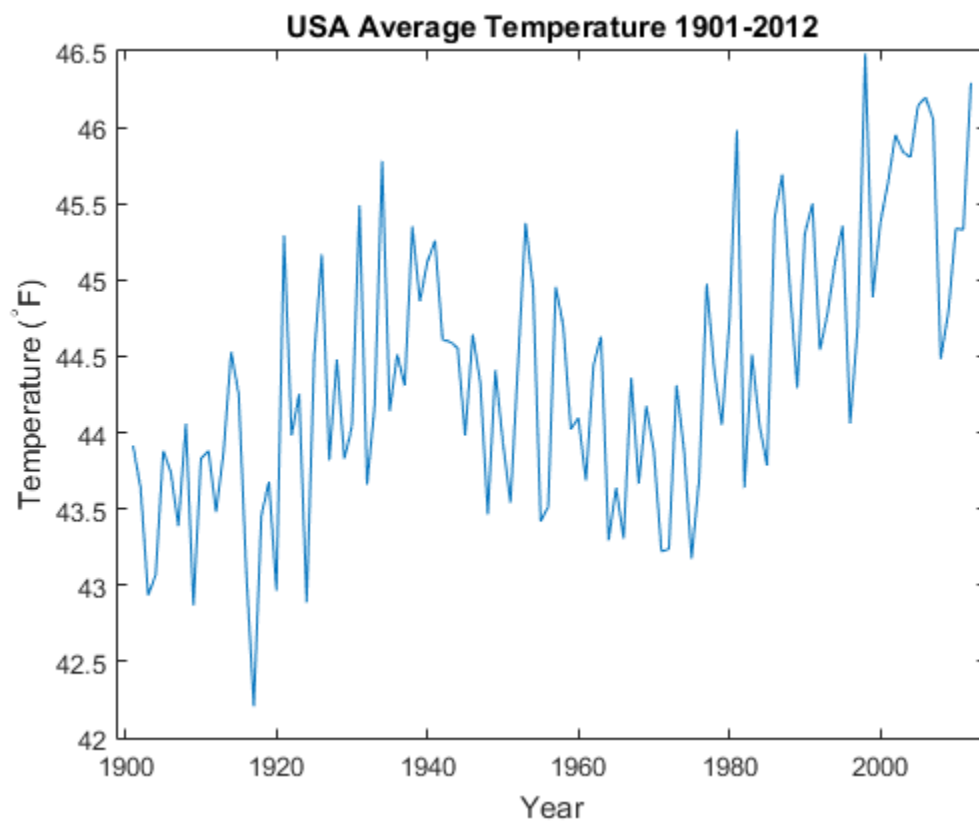
S(112)

ans =

```
year: 2012
data: 7.9395
```

绘制年平均温度图。将温度和年份转换为数值数组。将年份转换为日期时间对象以方便绘图，并将温度转换为华氏度。

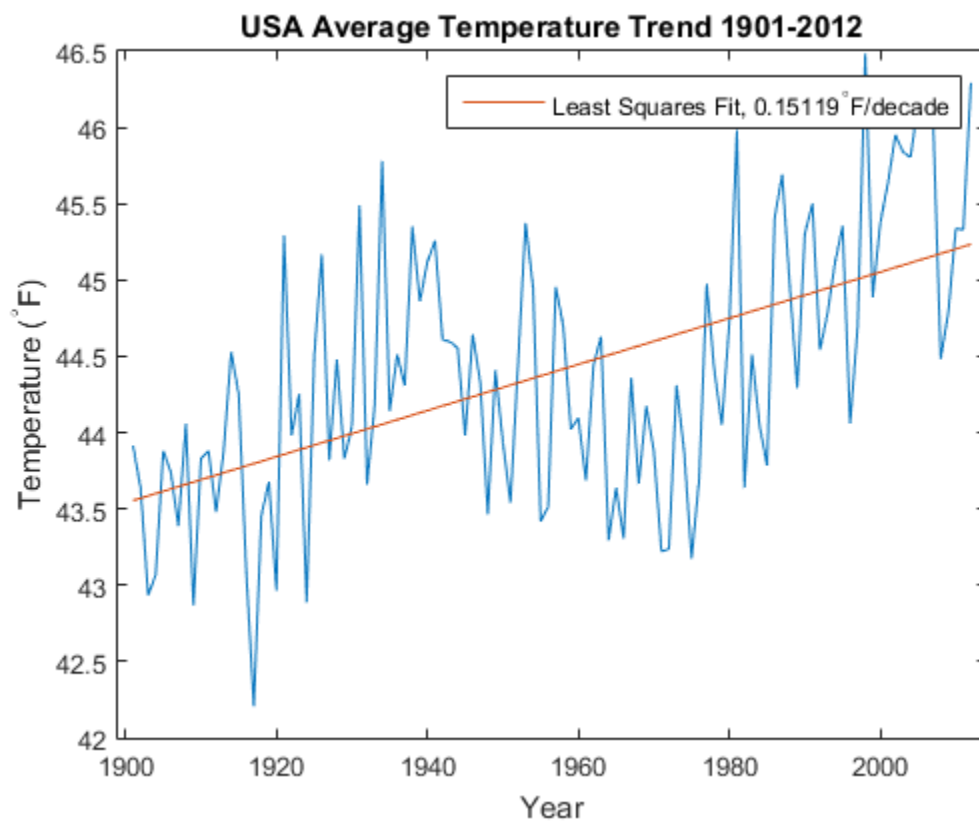
```
temps = [S.data];
temps = 9/5 * temps + 32;
years = [S.year];
yearstoplot = datetime(years,1,1);
figure
plot(yearstoplot, temps);
title('USA Average Temperature 1901-2012')
xlabel('Year')
ylabel('Temperature (^{\circ}F)')
xmin = datetime(1899,1,1);
xmax = datetime(2014,1,1);
xlim([xmin xmax])
```



在图上叠加绘制温度的最小二乘拟合线。

```
p = polyfit(years,temps,1);
ptemps = polyval(p,years);
deltat = p(1);
hold on
fl = plot(yearstoplot, ptemps);
xlim([xmin xmax])
title('USA Average Temperature Trend 1901-2012')
xlabel('Year')
ylabel('Temperature (^{\circ}F)')
deltat = num2str(10.0*deltat);
legend(fl,['Least Squares Fit, ', deltat, '^{\circ}F/decade'])
hold off
```





世界银行提供的 API 和数据：气候数据 API。（请参阅世界银行：气候数据 API 以了解有关 API 的更多信息；并参阅世界银行：使用条款。）

## 下载网页和文件

MATLAB 提供了两个用于从 RESTful Web 服务读取内容的函数：**webread** 和 **websave**。利用 **webread** 函数，您可以将网页内容读取到 MATLAB 工作区内的字符数组中。利用 **websave** 函数，您可以将网页内容保存到文件。

由于 **webread** 函数可以在工作区内创建字符数组，因此它非常适合在 MATLAB 中处理网页内容。**websave** 函数适用于将网页保存到本地文件夹。

---

**注意** 在 **webread** 以字符数组形式返回 HTML 时，请记住，它仅检索了该特定网页中的 HTML。超链接目标、图像等均未检索。

---

如果您需要向网页传递参数，利用 **webread** 和 **websave** 函数可以将这些参数定义为 Name, Value 对组参数。有关详细信息，请参阅 **webread** 和 **websave** 参考页。

### 示例 - 使用 webread 函数

以下过程展示了如何检索网页内容，该网页中列出了提交到 MATLAB Central™ File Exchange <https://www.mathworks.com/matlabcentral/fileexchange/> 的文件。它将结果赋给字符数组 **fullList**：

```
filex = 'https://www.mathworks.com/matlabcentral/fileexchange/';
fullList = webread(filex);
```

仅检索过去 7 天内上传至 File Exchange 且包含单词 Simulink® 的文件的列表。将 **duration** 和 **term** 设为 **webread** 将传递至网页的参数。

```
filex = 'https://www.mathworks.com/matlabcentral/fileexchange/';
recent = webread(filex,'duration',7,'term','simulink');
```

### 示例 - 使用 websave 函数

以下示例基于前面部分的过程构建，但将内容保存至文件：

```
% Locate the list of files at the MATLAB Central File Exchange
% uploaded within the past 7 days, that contain "Simulink."
filex = 'https://www.mathworks.com/matlabcentral/fileexchange/';
```

```
% Save the Web content to a file.
recent = websave('contains_simulink.html',filex, ...
 'duration',7,'term','simulink');
```

MATLAB 将网页另存为 **contains\_simulink.html**。输出参数 **recent** 包含 **contains\_simulink.html** 的完整路径。调用 **web** 函数，以便在浏览器中显示 **contains\_simulink.html**。

```
web(recent)
```

此页面包含了已上传至 MATLAB Central File Exchange 的文件的链接。

## 发送电子邮件

要从 MATLAB 发送电子邮件，请使用 `sendmail` 函数。也可以将文件附加到电子邮件，从而直接从 MATLAB 以邮件形式发送文件。要使用 `sendmail`，请用 `setpref` 函数设置电子邮件地址和 SMTP 服务器信息。

`setpref` 函数定义两个与邮件相关的预设项：

- 电子邮件地址：此预设项设置要在邮件中显示的电子邮件地址。
- SMTP 服务器：此预设项设置传出 SMTP 服务器地址，它可以是几乎任何支持邮局协议 (POP) 或互联网邮件访问协议 (IMAP) 的电子邮件服务器。

```
setpref('Internet','E_mail','youraddress@yourserver.com');
```

```
setpref('Internet','SMTP_Server','mail.server.network');
```

在电子邮件客户端应用程序的电子邮件帐户设置中查找传出 SMTP 服务器地址。也可以与系统管理员联系以获取这些信息。

一旦正确配置了 MATLAB，就可以使用 `sendmail` 函数。`sendmail` 函数需要至少两个参数：收件人的电子邮件地址和电子邮件主题。

```
sendmail('recipient@someserver.com','Hello From MATLAB!');
```

使用字符向量的元胞数组可以提供多个电子邮件地址。

```
sendmail({'recipient@someserver.com','recipient2@someserver.com'}, ...
 'Hello From MATLAB!');
```

可以指定消息主体。

```
sendmail('recipient@someserver.com','Hello From MATLAB', ...
 'Thanks for using sendmail.');
```

可以将文件附加到电子邮件。

```
sendmail('recipient@someserver.com','Hello from MATLAB', ...
 'Thanks for using sendmail.','C:\yourFileSystem\message.txt');
```

必须包含消息文件，才能添加附件。不过消息可以为空。

可以将多个文件附加到一个电子邮件。

```
sendmail('recipient@someserver.com','Hello from MATLAB', ...
 'Thanks for using sendmail',{'C:\yourFileSystem\message.txt', ...
 'C:\yourFileSystem\message2.txt'});
```

### 另请参阅

`sendmail` | `setpref`



# 串行端口 I/O

---

- “配置串行端口通信设置” (第 12-2 页)
- “使用串行端口通信从 Arduino 读取流数据” (第 12-3 页)

## 配置串行端口通信设置

串行端口对象和设备必须具有相同的通信设置，才能写入或读取数据。配置串行端口通信涉及指定用于控制波特率和“Serial Data Format”的属性的值。这些属性如下所示。

### 串行端口通信属性

属性名称	说明
BaudRate	指定位传输速率。
Parity	指定奇偶校验检查的类型。
DataBits	指定要传输的数据位数。
StopBits	指定用于指示字节结尾的位数。
Terminator	指定终止符字符。

**小心** 如果串行端口对象和仪器通信设置不相同，则可能无法成功读取或写入数据。

有关仪器支持的通信设置的说明，请参考仪器文档。

您可以显示在“Create Serial Port Object”中创建的串行端口对象 `s` 的通信属性值。

```
s = serialport("COM4",9600)
```

```
s =
```

Serialport with properties:

```
Port: "COM4"
BaudRate: 9600
NumBytesAvailable: 0
```

Show all properties, all methods

```
Port: "COM4"
BaudRate: 9600
NumBytesAvailable: 0

ByteOrder: "little-endian"
DataBits: 8
StopBits: 1
Parity: "none"
FlowControl: "none"
Timeout: 10
Terminator: "LF"
```

```
BytesAvailableFcnMode: "off"
BytesAvailableFcnCount: 64
BytesAvailableFcn: []
NumBytesWritten: 0
```

```
ErrorOccurredFcn: []
UserData: []
```

## 使用串行端口通信从 Arduino 读取流数据

此示例说明如何使用 `serialport` 接口启用回调以从 Arduino® Due 读取以 ASCII 字符结尾的流数据。

### 在 Arduino 上加载程序

将 Arduino Due 插入您的计算机。

使用 Arduino IDE 在 Arduino Due 上加载以下程序。此程序输出连续正弦波点，然后是“回车”和“换行”终止符。

```
/*
SineWavePoints

Write sine wave points to the serial port, followed by the Carriage Return and LineFeed terminator.
*/

int i = 0;

// The setup routine runs once when you press reset:
void setup() {
 // Initialize serial communication at 9600 bits per second:
 Serial.begin(9600);
}

// The loop routine runs over and over again forever:
void loop() {
 // Write the sinewave points, followed by the terminator "Carriage Return" and "Linefeed".
 Serial.print(sin(i*50.0/360.0));
 Serial.write(13);
 Serial.write(10);
 i += 1;
}
```

### 与 Arduino 建立连接

创建一个 `serialport` 实例来连接到您的 Arduino Due。

找到 Arduino 连接到的串行端口。您可以从 Arduino IDE 中识别该端口。

```
serialportlist("available")'
```

```
ans = 3x1 string
"COM1"
"COM3"
"COM13"
```

通过使用 Arduino 代码中指定的端口和波特率创建 `serialport` 对象，连接到 Arduino Due。

```
arduinoObj = serialport("COM13",9600)
```

```
arduinoObj =
Serialport with properties
```

```
 Port: "COM13"
 BaudRate: 9600
```

```
NumBytesAvailable: 0
NumBytesWritten: 0
```

Show all properties

### 准备 serialport 对象以开始流式传输数据

通过清除旧数据并配置其属性来配置 serialport 对象。

将 Terminator 属性设置为与您在 Arduino 代码中指定的终止符匹配。

```
configureTerminator(arduinoObj,"CR/LF");
```

清空 serialport 对象以删除所有旧数据。

```
flush(arduinoObj);
```

准备 UserData 属性来存储 Arduino 数据。结构体的 Data 字段保存正弦波值，Count 字段保存正弦波的 x 轴值。

```
arduinoObj.UserData = struct("Data",[],"Count",1)
```

```
arduinoObj =
Serialport with properties
```

```
Port: "COM13"
BaudRate: 9600
NumBytesAvailable: 10626
NumBytesWritten: 0
```

Show all properties

创建一个回调函数 readSineWaveData，它读取前 1000 个以 ASCII 字符结尾的正弦波数据点并绘制结果。

```
function readSineWaveData(src, ~)
```

```
% Read the ASCII data from the serialport object.
data = readline(src);
```

```
% Convert the string data to numeric type and save it in the UserData
% property of the serialport object.
src.UserData.Data(end+1) = str2double(data);
```

```
% Update the Count value of the serialport object.
src.UserData.Count = src.UserData.Count + 1;
```

```
% If 1001 data points have been collected from the Arduino, switch off the
% callbacks and plot the data.
```

```
if src.UserData.Count > 1001
 configureCallback(src, "off");
 plot(src.UserData.Data(2:end));
```

```
end
end
```



将 BytesAvailableFcnMode 属性设置为 "terminator", 将 BytesAvailableFcn 属性设置为 @readSineWaveData。当可以从 Arduino 读取新正弦波数据（带终止符）时，会触发回调函数 readSineWaveData。

```
configureCallback(arduinoObj,"terminator",@readSineWaveData);
```

该回调函数会打开 MATLAB® 图窗窗口，图窗窗口中会显示前 1000 个正弦波数据点的绘图。



# 大型数据

---

- “MapReduce 快速入门” (第 13-2 页)
- “编写 map 函数” (第 13-7 页)
- “数据存储快速入门” (第 13-10 页)
- “处理远程数据” (第 13-14 页)
- “读取和分析大型表格文本文件” (第 13-19 页)
- “读取和分析图像文件” (第 13-21 页)
- “使用 tall 数组处理无法放入内存的数据” (第 13-25 页)
- “在 MATLAB 中使用 tall 数组分析大数据” (第 13-30 页)

# MapReduce 快速入门

随着数据采集设备的数量和类型逐年增加，所收集的绝对数据大小和速率也在快速增长。这些大型数据集可能包含数 GB 或数 TB 的数据，并且可能以每天 MB 或 GB 的数量级增长。收集这类信息不仅提供了获取深入见解的机会，也带来了许多挑战。大多数算法的设计无法在合理的时间内或使用合理的内存量处理大型数据集。利用 MapReduce，您可以应对从大型数据集获取重要见解时所面临的诸多挑战。

本节内容
“什么是 MapReduce?” (第 13-2 页)
“MapReduce 算法阶段” (第 13-2 页)
“MapReduce 计算示例” (第 13-3 页)

## 什么是 MapReduce?

MapReduce 是一种用于分析无法放入内存的数据集的编程方法。您可能很熟悉 Hadoop® MapReduce，它是一种用于 Hadoop 分布式文件系统 (HDFS™) 的常用实现。MATLAB 使用 `mapreduce` 函数提供了一种略有不同的 MapReduce 方法实现。

`mapreduce` 使用数据存储，基于可分别放入内存的小数据块来处理数据。每个数据块会经历映射阶段，此阶段对要处理的数据进行格式化。之后，中间数据块将经历化简 (Reduce) 阶段，此阶段对中间结果进行聚合，以生成最终结果。映射和化简阶段使用 `map` 和 `reduce` 函数进行编码，这些函数是 `mapreduce` 的主要输入。`map` 函数和 `reduce` 函数有无限多种用于处理数据的组合，因此该方法不仅灵活，而且非常强大，可用于处理大型数据处理任务。

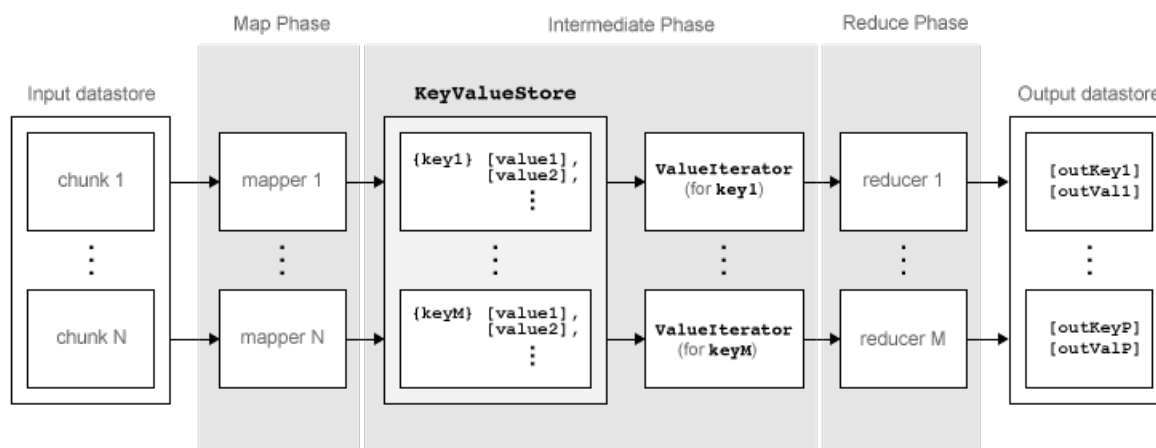
`mapreduce` 支持自我扩展，以便在多种环境中运行。有关这些功能的详细信息，请参阅 “Speed Up and Deploy MapReduce Using Other Products”。

`mapreduce` 函数的功用在于它可对大型数据集执行计算。因此，`mapreduce` 不太适合对正常大小的数据集执行计算，这类数据集可直接加载到计算机内存并使用传统方法进行分析。应将 `mapreduce` 用于对无法放入内存的数据集执行统计或解析计算。

`mapreduce` 每次调用的 `map` 或 `reduce` 函数都是独立于所有其他函数的。例如，对 `map` 函数的调用不能依赖于上一次 `map` 函数调用的输入或结果。最好将此类计算分解为多次 `mapreduce` 调用。

## MapReduce 算法阶段

在到达最终输出之前，`mapreduce` 会移动输入数据存储中的各个数据块，使其经历多个阶段。下图概述了 `mapreduce` 的算法阶段。



该算法包含以下步骤：

- 1 **mapreduce** 使用 `[data,info] = read(ds)` 从输入数据存储读取数据块，然后调用 `map` 函数处理该数据块。
- 2 `map` 函数接收数据块，组织数据块或执行前驱计算，然后使用 `add` 和 `addmulti` 函数将键-值对组添加到名为 **KeyValueStore** 的中间数据存储对象。**mapreduce** 对 `map` 函数的调用次数等于输入数据存储中的数据块数目。
- 3 `map` 函数处理完数据存储中的所有数据块后，**mapreduce** 按照唯一键对中间 **KeyValueStore** 对象中的所有值进行分组。
- 4 接下来，**mapreduce** 针对 `map` 函数添加的每个唯一键调用一次 `reduce` 函数。每个唯一键可以有多个关联的值。**mapreduce** 将这些值以 **ValueIterator** 对象（用于循环访问这些值）的形式传递给 `reduce` 函数。每个唯一键的 **ValueIterator** 对象包含了该键的所有关联值。
- 5 `reduce` 函数使用 `hasnext` 和 `getnext` 函数，逐一遍历 **ValueIterator** 对象中的值。然后，在聚合 `map` 函数的所有中间结果后，`reduce` 函数使用 `add` 和 `addmulti` 函数将最终的键-值对组添加到输出。输出中的键顺序与 `reduce` 函数将其添加到最终 **KeyValueStore** 对象的顺序相同。即，**mapreduce** 不会显式对输出进行排序。

**注意** `reduce` 函数将最终键-值对组写入到最终 **KeyValueStore** 对象。**mapreduce** 将键-值对组从该对象拉入输出数据存储（默认为 **KeyValueDatastore** 对象）。

## MapReduce 计算示例

以下示例使用一项简单的计算（某航班数据集中的平均航程）说明运行 **mapreduce** 所需的步骤。

### 准备数据

使用 **mapreduce** 的第一步是为数据集构造数据存储。数据集的数据存储与 `map` 和 `reduce` 函数一样，都是 **mapreduce** 的必要输入，因为 **mapreduce** 需要利用数据存储来处理数据块中的数据。

**mapreduce** 可处理大多数数据存储类型。例如，为 `airlinesmall.csv` 数据集创建一个 **TabularTextDatastore** 对象。

```
ds = tabularTextDatastore('airlinesmall.csv','TreatAsMissing','NA')
```

```
ds =
```

```
TabularTextDatastore with properties:
```

```

Files: {
 '...\matlab\toolbox\matlab\demos\airlinesmall.csv'
}
Folders: {
 '...\matlab\toolbox\matlab\demos'
}
FileEncoding: 'UTF-8'
AlternateFileSystemRoots: {}
PreserveVariableNames: false
ReadVariableNames: true
VariableNames: {'Year', 'Month', 'DayofMonth' ... and 26 more}
DatetimeLocale: en_US

Text Format Properties:
NumHeaderLines: 0
Delimiter: ','
RowDelimiter: '\r\n'
TreatAsMissing: 'NA'
MissingValue: NaN

Advanced Text Format Properties:
TextscanFormats: {'%f', '%f', '%f' ... and 26 more}
TextType: 'char'
ExponentCharacters: 'eEdD'
CommentStyle: ""
Whitespace: '\b\t'
MultipleDelimitersAsOne: false

Properties that control the table returned by preview, read, readall:
SelectedVariableNames: {'Year', 'Month', 'DayofMonth' ... and 26 more}
SelectedFormats: {'%f', '%f', '%f' ... and 26 more}
ReadSize: 20000 rows
OutputType: 'table'
RowTimes: []

Write-specific Properties:
SupportedOutputFormats: ["txt" "csv" "xlsx" "xls" "parquet" "parq"]
DefaultOutputFormat: "txt"

```

之前描述的多个选项在 **mapreduce** 的上下文中非常有用。**mapreduce** 函数对数据存储执行 **read**，以检索要传递到 **map** 函数的数据。因此，您可以使用 **SelectedVariableNames**、**SelectedFormats** 和 **ReadSize** 选项来直接配置 **mapreduce** 传递给 **map** 函数的数据块大小和数据类型。

例如，要选择 **Distance**（总航程）变量作为唯一要关注的变量，请指定 **SelectedVariableNames**。

```
ds.SelectedVariableNames = 'Distance';
```

现在，不论何时对 **ds** 执行 **read**、**readall** 或 **preview** 函数，它们都仅返回 **Distance** 变量的信息。要确认这一点，您可以预览数据存储中的前几行数据。这样，您可以检查 **mapreduce** 函数将要传递给 **map** 函数的数据的格式。

```
preview(ds)
```

```
ans =
```

```
8×1 table
```

```
Distance
```

```

308
296
480
296
373
308
447
954

```

要查看 `mapreduce` 将要传递给 `map` 函数的确切数据，请使用 `read`。

有关可用选项的更多信息和完整摘要，请参阅“数据存储”。

## 写入 `map` 函数和 `reduce` 函数

`mapreduce` 函数会在执行期间自动调用 `map` 函数和 `reduce` 函数，因此这些函数必须满足特定的要求才能正确运行。

### 1 `map` 函数的输入包括 `data`、`info` 和 `intermKVStore`：

- `data` 和 `info` 是对输入数据存储调用 `read` 函数的结果，`mapreduce` 在每次调用 `map` 函数之前都会自动执行该函数。
- `intermKVStore` 是 `KeyValueStore` 中间对象的名称，`map` 函数需要使用该名称添加键-值对组。`add` 和 `addmulti` 函数使用此对象名称添加键-值对组。如果对 `map` 函数的所有调用都没有向 `intermKVStore` 中添加键-值对组，则 `mapreduce` 不会调用 `reduce` 函数，并且结果数据存储为空。

下面是一个简单的 `map` 函数示例：

```
function MeanDistMapFun(data, info, intermKVStore)
 distances = data.Distance(~isnan(data.Distance));
 sumLenValue = [sum(distances) length(distances)];
 add(intermKVStore, 'sumAndLength', sumLenValue);
end
```

此 `map` 函数只有三行，分别执行了一些简单的操作。第一行过滤出了航程数据块中的所有 NaN 值。第二行使用块的总航程和计数创建了一个二元素向量，第三行将该值向量添加到键为 'sumAndLength' 的 `intermKVStore`。对 `ds` 中的所有数据块运行此 `map` 函数后，`intermKVStore` 对象将包含各个航程数据块的总航程和计数。

在您的当前文件夹中将此函数另存为 `MeanDistMapFun.m`。

### 2 `reduce` 函数的输入包括 `intermKey`、`intermValIter` 和 `outKVStore`：

- `intermKey` 用于 `map` 函数所添加的活动键。`mapreduce` 每次调用 `reduce` 函数都会根据中间 `KeyValueStore` 对象中的键指定新的唯一键。
- `intermValIter` 是与活动键 `intermKey` 相关的 `ValueIterator`。这个 `ValueIterator` 对象包含与活动键相关的所有值。使用 `hasnext` 和 `getnext` 函数滚动这些值。
- `outKVStore` 是最终 `KeyValueStore` 对象的名称，`reduce` 函数需要使用该名称添加键-值对组。`mapreduce` 从 `outKVStore` 中获取输出键-值对组并在输出数据存储中返回它们，默认情况下是一个 `KeyValueDatastore` 对象。如果对 `reduce` 函数的所有调用都没有向 `outKVStore` 中添加键-值对组，则 `mapreduce` 将返回空的数据存储。

下面是一个简单的 `reduce` 函数示例：

```
function MeanDistReduceFun(interKey, intermValIter, outKVStore)
 sumLen = [0 0];
 while hasnext(intermValIter)
 sumLen = sumLen + getnext(intermValIter);
 end
 add(outKVStore, 'Mean', sumLen(1)/sumLen(2));
end
```

此 `reduce` 函数会遍历 `intermValIter` 中每组航程和计数的值，并在每次执行后保留航程和计数的实时总和。完成此循环后，`reduce` 函数使用简单的除法计算出总平均航程，然后向 `outKVStore` 中添加一个键。

在您的当前文件夹中将此函数另存为 `MeanDistReduceFun.m`。

有关编写更高级的 `map` 函数和 `reduce` 函数的信息，请参阅“编写 `map` 函数”（第 13-7 页）和“Write a Reduce Function”。

## 运行 `mapreduce`

有了数据存储、`map` 函数和 `reduce` 函数之后，您便可以调用 `mapreduce` 来执行计算。要计算数据集中的平均航程，请使用 `ds`、`MeanDistMapFun` 和 `MeanDistReduceFun` 来调用 `mapreduce`。

```
outds = mapreduce(ds, @MeanDistMapFun, @MeanDistReduceFun);
```

```

* MAPREDUCE PROGRESS *

Map 0% Reduce 0%
Map 16% Reduce 0%
Map 32% Reduce 0%
Map 48% Reduce 0%
Map 65% Reduce 0%
Map 81% Reduce 0%
Map 97% Reduce 0%
Map 100% Reduce 0%
Map 100% Reduce 100%
```

默认情况下，`mapreduce` 函数会在命令行中显示进度信息，并返回一个指向当前文件夹中的文件的 `KeyValueDatastore` 对象。您可以使用 `Name, Value` 对组参数来调整 '`OutputFolder`'、'`OutputType`' 和 '`Display`' 这三个选项。有关详细信息，请参阅 `mapreduce` 的参考页。

## 查看结果

使用 `readall` 函数从输出数据存储读取键-值对组。

```
readall(outds)
```

```
ans =
```

```
1×2 table
```

Key	Value
-----	-------

{ 'Mean' }	{ [702.1630] }
------------	----------------

## 另请参阅

`tabularTextDatastore` | `mapreduce`

## 相关示例

- “Build Effective Algorithms with MapReduce”



## 编写 map 函数

### 本节内容

“map 函数在 MapReduce 中的角色” (第 13-7 页)

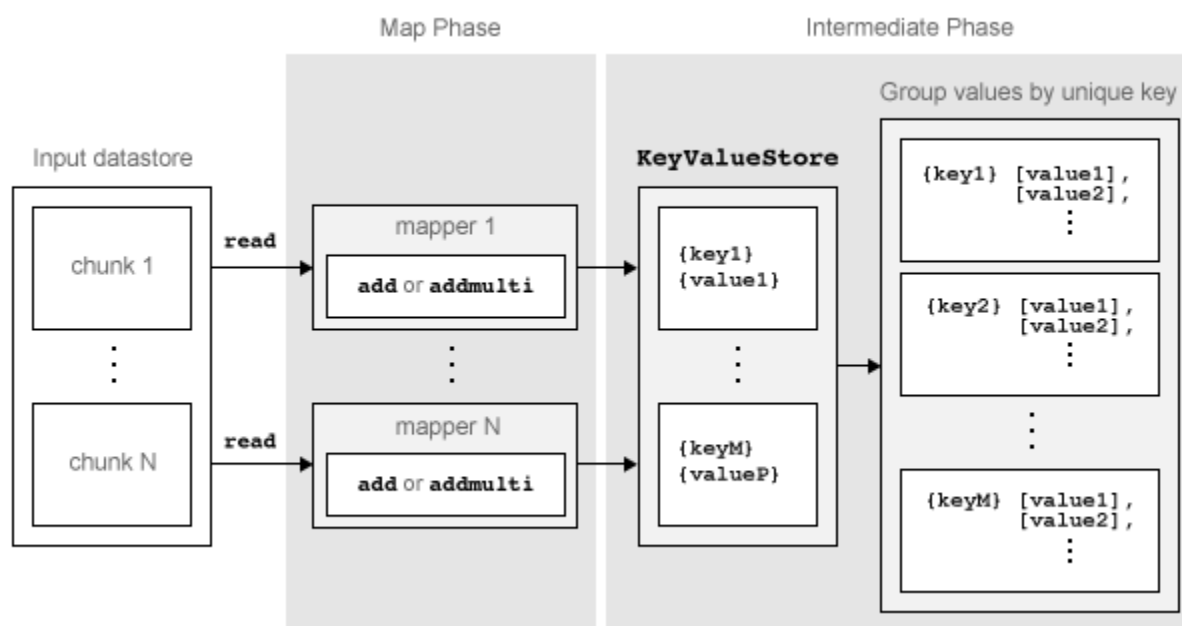
“map 函数的要求” (第 13-8 页)

“map 函数示例” (第 13-8 页)

## map 函数在 MapReduce 中的角色

**mapreduce** 既需要接收数据块并输出中间结果的输入 map 函数，也需要读取中间结果并生成最终结果的输入 reduce 函数。因此，通常将计算分解成两个相关部分，分别由 map 和 reduce 函数来完成。例如，要求数据集中的最大值，可以使用 map 函数求出每个输入数据块中的最大值，然后使用 reduce 函数求出所有中间最大值中的单个最大值。

下图显示 **mapreduce** 算法的映射阶段。



**mapreduce** 算法的映射阶段包含以下步骤：

- 1 **mapreduce** 使用输入数据存储上的 **read** 函数读取单个数据块，然后调用 map 函数对该数据块进行处理。
- 2 然后，map 函数对单个数据块进行处理，并使用 **add** 或 **addmulti** 函数向中间 **KeyValueStore** 对象添加一个或多个键-值对组。
- 3 **mapreduce** 对输入数据存储中的每个数据块重复此过程，因此对 map 函数的调用总数等于数据块数。数据存储的 **ReadSize** 属性决定数据块数。

当 map 函数处理输入数据存储中的每个数据块时，**mapreduce** 算法的映射阶段完成。**mapreduce** 算法的此阶段的结果是一个 **KeyValueStore** 对象，它包含 map 函数添加的所有键-值对组。在映射阶段后，**mapreduce** 通过按唯一键对 **KeyValueStore** 对象中的所有值进行分组，为归约阶段做准备。

## map 函数的要求

**mapreduce** 自动为输入数据存储中的每个数据块调用 **map** 函数。**map** 函数必须满足某些基本要求，才能在这些自动调用期间正常运行。这些要求共同确保数据正确通过 **mapreduce** 算法的映射阶段。

**map** 函数的输入包括 **data**、**info** 和 **intermKVStore**：

- **data** 和 **info** 是对输入数据存储调用 **read** 函数的结果，**mapreduce** 在每次调用 **map** 函数之前都会自动执行该函数。
- **intermKVStore** 是 **KeyValueStore** 中间对象的名称，**map** 函数需要使用该名称添加键-值对组。**add** 和 **addmulti** 函数使用此对象名称添加键-值对组。如果 **map** 函数没有向 **intermKVStore** 对象添加任何键-值对组，则 **mapreduce** 不会调用 **reduce** 函数，并且得到的结果数据存储为空。

除了 **map** 函数的这些基本要求之外，**map** 函数添加的键-值对组还必须满足以下条件：

- 1 键必须为数值标量、字符向量或字符串。数值键不能为 NaN、复数、逻辑值或稀疏矩阵。
- 2 由 **map** 函数添加的所有键必须具有相同的类。
- 3 值可以是任何 MATLAB 对象，包括所有有效的 MATLAB 数据类型。

---

**注意** 在使用包含 **mapreduce** 的其他产品时，上述键-值对组要求可能不同。请参阅相应产品的文档以获得产品特定的键-值对组要求。

---

## map 函数示例

以下是在 **mapreduce** 示例中使用的一些 **map** 函数示例。

### 恒等映射函数

如果 **map** 函数只是简单地返回 **mapreduce** 传递给它的内容，则该 **map** 函数称为恒等映射函数。在 **reduce** 函数中进行计算之前，可以利用恒等映射函数按唯一键对值进行分组。**identityMapper** 映射函数文件是示例 “Tall Skinny QR (TSQR) Matrix Factorization Using MapReduce” 中使用的映射函数之一。

```
function identityMapper(data, info, intermKVStore)
% This mapper function simply copies the data and add them to the
% intermKVStore as intermediate values.
x = data.Value{:,};
add(intermKVStore,'Identity',x);
end
```

### 简单 map 函数

非恒等映射函数的一个最简单示例是 **maxArrivalDelayMapper**，这是示例 “Find Maximum Value with MapReduce” 的映射函数。对于输入数据的每个分块，此映射函数都会计算最大到港延误，并向中间 **KeyValueStore** 添加一个键-值对组。

```
function maxArrivalDelayMapper (data, info, intermKVStore)
partMax = max(data.ArrDelay);
add(intermKVStore, 'PartialMaxArrivalDelay',partMax);
end
```

### 高级 map 函数

**statsByGroupMapper** 是一个更高级的映射函数示例，它是示例 “Compute Summary Statistics by Group Using MapReduce” 的映射函数。此映射函数使用嵌套函数计算每个输入数据分块的几个统计量

(计数、均值、方差等)，然后向中间 `KeyValueStore` 对象添加几个键-值对组。此外，此映射函数使用四个输入参数，而 `mapreduce` 只接受具有三个输入参数的 `map` 函数。为了解决此问题，请在调用 `mapreduce` 的过程中使用匿名函数传入一个额外参数，如示例所示。

```
function statsByGroupMapper(data, ~, intermKVStore, groupVarName)
% Data is a n-by-3 table. Remove missing values first
delays = data.ArrDelay;
groups = data.(groupVarName);
notNaN = ~isnan(delays);
groups = groups(notNaN);
delays = delays(notNaN);
% Find the unique group levels in this chunk
[intermKeys, ~, idx] = unique(groups, 'stable');
% Group delays by idx and apply @grpstatsfun function to each group
intermVals = accumarray(idx, delays, size(intermKeys), @grpstatsfun);
addmulti(intermKVStore, intermKeys, intermVals);
function out = grpstatsfun(x)
 n = length(x); % count
 m = sum(x)/n; % mean
 v = sum((x-m).^2)/n; % variance
 s = sum((x-m).^3)/n; % skewness without normalization
 k = sum((x-m).^4)/n; % kurtosis without normalization
 out = {n, m, v, s, k};
end
end
```

## 更多 map 函数

有关 `map` 或 `reduce` 函数中常见编程模式的详细信息，请参阅“Build Effective Algorithms with MapReduce”。

## 另请参阅

`mapreduce` | `tabularTextDatastore` | `add` | `addmulti`

## 详细信息

- `KeyValueStore`
- “Write a Reduce Function”
- “MapReduce 快速入门” (第 13-2 页)

# 数据存储快速入门

本节内容
“什么是数据存储？” （第 13-10 页）
“创建和读取数据存储” （第 13-11 页）

## 什么是数据存储？

数据存储是一个用于读取单个文件或者文件或数据集的对象。它相当于一个存储库，用来存储具有相同结构和格式的数据。例如，数据存储中每个文件包含的数据必须具有相同的类型（如数字或文本）、以相同顺序显示并用相同的分隔符分隔。

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R				
1	Year	Month	DayOfMor	DayOfWe	DepTime	CRSDepTii	ArrTime	CRSArrTin	UniqueCa	FlightNum	TailNum	ActualElap	CRSElapse	AirTime	ArrDelay	DepDelay	Origin	Dest	Distance			
2	2000		3	2	4	1641		1830	WN	1726	N353	50	55	41	1	6	ELP	LBB				
3	2000		a	5	7	2004		1950	AA	2307	2249	AA	123	119	98	18	14	BNA	LGA			
4	2000																					
5	2000	1	Year	Month	DayOfMor	DayOfWe	DepTime	CRSDepTii	ArrTime	CRSArrTin	UniqueCa <td>FlightNum</td> <td>TailNum</td> <td>ActualElap</td> <td>CRSElapse</td> <td>AirTime</td> <td>ArrDelay</td> <td>DepDelay</td> <td>Origin</td> <td>Dest</td> <td>Distance</td>	FlightNum	TailNum	ActualElap	CRSElapse	AirTime	ArrDelay	DepDelay	Origin	Dest	Distance	
6	2000	2	1990		10	31	6	1655	1655	1742	1755	PI	903	NA	47	60	NA	-13	0	LGA	SYR	
7	2000	3	1990		10	11	7	1042	1042	1107	1107	PI	929	NA	25	25	NA	0	0	SYR	ITH	
8	2000	4	1990																			
9	2000	5	1990																			
10	2000	6	1990	2	1987	10	21	3	642	630	735	727	PS	1503	NA	53	57	NA	8	12	LAX	SJC
11	2000	7	1990	3	1987	10	26	1	1021	1020	1124	1116	PS	1550	NA	63	56	NA	8	1	SJC	BUR
12	2000	8	1990	4	1987	10	23	5	2055	2035	2218	2157	PS	1589	NA	83	82	NA	21	20	SAN	SMF
13	2000	9	1990	5	1987	10	23	5	1332	1320	1431	1418	PS	1655	NA	59	58	NA	13	12	BUR	SJC
14	2000	10	1990	6	1987	10	22	4	629	630	746	742	PS	1702	NA	77	72	NA	4	-1	SMF	LAX
15	2000	11	1990	7	1987	10	28	3	1446	1343	1547	1448	PS	1729	NA	61	65	NA	59	63	LAX	SJC
16	2000	12	1990	8	1987	10	8	4	928	930	1052	1049	PS	1763	NA	84	79	NA	3	-2	SAN	SFO
17	2000	13	1990	9	1987	10	10	6	859	900	1134	1123	PS	1800	NA	155	143	NA	11	-1	SEA	LAX
18	2000	14	1990	10	1987	10	20	2	1833	1830	1929	1926	PS	1831	NA	56	56	NA	3	3	LAX	SJC
19	2000	15	1990	11	1987	10	15	4	1041	1040	1157	1155	PS	1864	NA	76	75	NA	2	1	SFO	LAS
20	2000	16	1990	12	1987	10	15	4	1608	1553	1656	1640	PS	1907	NA	48	47	NA	16	15	LAX	FAT
21	2000	17	1990	13	1987	10	21	3	949	940	1055	1052	PS	1939	NA	66	72	NA	3	9	LGB	SFO
22	2000	18	1990	14	1987	10	22	4	1902	1847	2030	1951	PS	1973	NA	88	64	NA	39	15	LAX	OAK
23	2000	19	1990	15	1987	10	16	5	1910	1838	2052	1955	TW	19	NA	162	137	NA	57	32	STL	DEN
24	2000	20	1990	16	1987	10	2	5	1130	1133	1237	1237	TW	59	NA	187	184	NA	0	-3	STL	PHX
25	2000	21	1990	17	1987	10	30	5	1400	1400	1920	1934	TW	102	NA	200	214	NA	-14	0	SNA	STL
26	2000	22	1990	18	1987	10	28	3	841	830	1233	1218	TW	136	NA	172	168	NA	15	11	UTA	STL
27	2000	23	1990	19	1987	10	5	1	1500	1445	1703	1655	TW	183	NA	243	250	NA	8	15	STL	SFO
28	2000	24	1990	20	1987	10	27	2	1647	1640	1914	1903	TW	220	NA	87	83	NA	11	7	STL	DTW
29	2000	25	1990	21	1987	10	15	4	1709	1710	1752	1749	TW	251	NA	103	99	NA	3	-1	PIT	STL
30	2000	26	1990	22	1987	10	24	6	1515	1515	1544	1545	TW	283	NA	29	30	NA	-1	0	SRQ	RSW
31	2000	27	1990	23	1987	10	25	7	2017	2017	2347	2329	TW	318	NA	150	132	NA	18	0	STL	BDL
32	2000	28	1990	24	1987	10	25	7	2218	2220	2335	2322	TW						13	-2	STL	PHX

文件或数据的类型	数据存储类型
作为 <code>mapreduce</code> 的输入或输出的键-值对组数据。	<code>KeyValueDatastore</code>
包含列向数据的 Parquet 文件。	<code>ParquetDatastore</code>
自定义文件格式。需要提供用于读取数据的函数。	<code>FileDatastore</code>
用于存放 <code>tall</code> 数组的检查点的数据存储。	<code>TallDatastore</code>

创建和读取数据存储

使用 `tabularTextDatastore` 函数从示例文件 `airlinesmall.csv` 创建一个数据存储，其中包含有关各航空公司航班的出发和到达信息。结果是一个 `TabularTextDatastore` 对象。

```
ds = tabularTextDatastore('airlinesmall.csv')

ds =
 TabularTextDatastore with properties:
 Files: {
 '...\matlab\toolbox\matlab\demos\airlinesmall.csv'
 }
 Folders: {
 '...\matlab\toolbox\matlab\demos'
 }
 FileEncoding: 'UTF-8'
 AlternateFileSystemRoots: {}
 PreserveVariableNames: false
 ReadVariableNames: true
 VariableNames: {'Year', 'Month', 'DayofMonth' ... and 26 more}
 DatetimeLocale: en_US

 Text Format Properties:
 NumHeaderLines: 0
 Delimiter: ','
 RowDelimiter: '\r\n'
 TreatAsMissing: ''
 MissingValue: NaN

 Advanced Text Format Properties:
 TextscanFormats: {'%f', '%f', '%f' ... and 26 more}
 TextType: 'char'
 ExponentCharacters: 'eEdD'
 CommentStyle: ''
 Whitespace: '\b\t'
 MultipleDelimitersAsOne: false

 Properties that control the table returned by preview, read, readall:
 SelectedVariableNames: {'Year', 'Month', 'DayofMonth' ... and 26 more}
 SelectedFormats: {'%f', '%f', '%f' ... and 26 more}
 ReadSize: 20000 rows
 OutputType: 'table'
 RowTimes: []

 Write-specific Properties:
 SupportedOutputFormats: ['txt' 'csv' 'xlsx' 'xls' 'parquet' 'parq']
 DefaultOutputFormat: 'txt'
```

创建数据存储后，可以预览数据而无需将其全部加载到内存中。可以使用 `SelectedVariableNames` 属性指定相关变量（列），以预览或只读这些变量。

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	
1	Year	Month	DayofMonth	DayOfWeek	DepTime	CRSDepTime	ArrTime	CRSArrTime	UniqueCarrier	FlightNum	TailNum	ActualElapsedTime	CRSElapseTime	AirTime	ArrDelay	DepDelay	Origin	Dest	Distance
2	1987	10	21	3	642	630	735	727	PS	1503	NA	53	57	NA	8	12	LAX	SJC	159
3	1987	10	26	1	1021	1020	1124	1116	PS	1550	NA	63	56	NA	8	1	SJC	BUR	105
4	1987	10	23	5	2055	2035	2218	2157	PS	1589	NA	83	82	NA	21	20	SAN	SMF	116
5	1987	10	23	5	1332	1320	1431	1418	PS	1655	NA	59	58	NA	13	12	BUR	SJC	105
6	1987	10	22	4	629	630	746	742	PS	1702	NA	77	72	NA	4	-1	SMF	LAX	116
7	1987	10	28	3	1446	1343	1547	1448	PS	1729	NA	61	65	NA	59	63	LAX	SJC	159
8	1987	10	8	4	928	930	1052	1049	PS	1763	NA	84	79	NA	3	-2	SAN	SFO	116
9	1987	10	10	6	859	900	1134	1123	PS	1800	NA	155	143	NA	11	-1	SEA	LAX	159
10	1987	10	20	2	1833	1830	1929	1926	PS	1831	NA	56	56	NA	3	3	LAX	SJC	159
11	1987	10	15	4	1041	1040	1157	1155	PS	1864	NA	76	75	NA	2	1	SFO	LAS	145
12	1987	10	15	4	1608	1553	1656	1640	PS	1907	NA	48	47	NA	16	15	LAX	FAT	100
13	1987	10	21	3	949	940	1055	1052	PS	1939	NA	66	72	NA	3	9	LGB	SFO	116
14	1987	10	22	4	1902	1847	2030	1951	PS	1973	NA	88	64	NA	39	15	LAX	OAK	100
15	1987	10	16	5	1910	1838	2052	1955	TW	19	NA	162	137	NA	57	32	STL	DEN	160
16	1987	10	2	5	1130	1133	1237	1237	TW	59	NA	187	184	NA	0	-3	STL	PHX	160
17	1987	10	30	5	1400	1400	1920	1934	TW	102	NA	200	214	NA	-14	0	SNA	STL	160
18	1987	10	28	3	841	830	1233	1218	TW	136	NA	172	168	NA	15	11	TUS	STL	160
19	1987	10	5	1	1500	1445	1703	1655	TW	183	NA	243	250	NA	8	15	STL	SFO	160
20	1987	10	27	2	1647	1640	1914	1903	TW	220	NA	87	83	NA	11	7	STL	DTW	160
21	1987	10	27	2	1708	1710	1752	1749	TW	220	NA	87	83	NA	3	-1	PIT	STL	160

```
ds.SelectedVariableNames = {'DepTime','DepDelay'};
preview(ds)
```

```
ans =
```

8×2 table

DepTime	DepDelay
642	12
1021	1
2055	20
1332	12
629	-1
1446	63
928	-2
859	-1

可以指定数据中表示缺失值的值。在 `airlinesmall.csv` 中，缺失值由 `NA` 表示。

```
ds.TreatAsMissing = 'NA';
```

如果相关变量的数据存储中的所有数据都放入内存，则可以使用 `readall` 函数读取。

```
T = readall(ds);
```

否则，使用 `read` 函数读取放入内存的较小子集中的数据。默认情况下，`read` 函数一次从 `TabularTextDatastore` 读取 20,000 行。但是，可以通过为 `ReadSize` 属性分配一个新值来更改此值。

```
ds.ReadSize = 15000;
```

在重新读取之前，使用 `reset` 函数将数据存储重置为初始状态。通过在 `while` 循环中调用 `read` 函数，可以对每个数据子集执行中间计算，然后在结束时汇总中间结果。以下代码计算 `DepDelay` 变量的最大值。

```
reset(ds)
X = [];
while hasdata(ds)
```

```

 T = read(ds);
 X(end+1) = max(T.DepDelay);
end
maxDelay = max(X)

maxDelay =

 1438

```

如果每个单独文件中的数据都能放入内存，则可以指定每次调用 `read` 时应读取一个完整文件，而不是特定行数。

```

reset(ds)
ds.ReadSize = 'file';
X = [];
while hasdata(ds)
 T = read(ds);
 X(end+1) = max(T.DepDelay);
end
maxDelay = max(X);

```

除了读取数据存储中的数据子集之外，还可以使用 **mapreduce** 将 `map` 和 `reduce` 函数应用于数据存储，或使用 **tall** 创建一个 **tall** 数组。有关详细信息，请参阅“MapReduce 快速入门”（第 13-2 页）和“使用 **tall** 数组处理无法放入内存的数据”（第 13-25 页）。

## 另请参阅

[tabularTextDatastore](#) | [imageDatastore](#) | [spreadsheetDatastore](#) | [KeyValueDatastore](#) | [fileDatastore](#) | [tall](#) | [mapreduce](#)

## 相关示例

- “Select Datastore for File Format or Application”
- “读取和分析大型表格文本文件”（第 13-19 页）
- “读取和分析图像文件”（第 13-21 页）

## 处理远程数据

您可以使用 MATLAB 函数和对象（例如文件 I/O 函数和一些数据存储对象）从远程位置读写数据。这些示例说明如何设置、读取和写入以下云存储平台上的远程位置：

- Amazon S3（简单存储服务）
- Azure Blob 存储（以前称为 Windows Azure® Storage Blob (WASB)）
- Hadoop 分布式文件系统 (HDFS)

### Amazon S3

MATLAB 允许使用 Amazon Web Services 提供的 Amazon S3 作为在线文件存储 Web 服务。当指定数据位置时，必须使用以下格式的统一资源定位器 (URL) 指定文件或文件夹的完整路径：

```
s3://bucketname/path_to_file
```

**bucketname** 是容器的名称，**path\_to\_file** 是文件或文件夹的路径。

Amazon S3 通过 Web 服务接口提供数据存储。您可以在 Amazon S3 中使用桶作为容器来存储对象。

#### 设置访问权限

要处理 Amazon S3 中的远程数据，必须先设置访问权限：

- 1 注册一个 Amazon Web Services (AWS) 根帐户。请参阅 Amazon Web Services：帐户。
- 2 使用 AWS 根帐户，创建 IAM（身份识别和访问管理）用户。请参阅在 AWS 帐户中创建 IAM 用户。
- 3 生成访问密钥以接收访问密钥 ID 和秘密访问密钥。请参阅管理 IAM 用户的访问密钥。
- 4 使用 <https://aws.amazon.com/cli/> 上提供的 AWS 命令行界面工具，为机器配置 AWS 访问密钥 ID、访问密钥和地区。或者，通过使用 `setenv` 来直接设置环境变量：
  - **AWS\_ACCESS\_KEY\_ID** 和 **AWS\_SECRET\_ACCESS\_KEY** - 对 Amazon S3 服务进行身份验证并启用服务。（在步骤 3 中生成了这对访问密钥变量。）
  - **AWS\_DEFAULT\_REGION**（可选）- 选择桶的地理区域。此环境变量的值通常自动确定，但桶所有者可能要求手动设置此变量的值。
  - **AWS\_SESSION\_TOKEN**（可选）- 如果您使用的是临时安全凭据，例如 AWS® 联合身份验证，请指定会话令牌。

如果正在使用的是 Parallel Computing Toolbox™，则必须确保已配置集群以访问 S3 服务。可通过在 `parpool`、`batch`、`createJob` 中或在 Cluster Profile Manager 中设置 **EnvironmentVariables**，将客户端环境变量复制到集群上的工作进程。

#### 从 Amazon S3 读取数据

以下示例说明如何使用 **ImageDatastore** 对象从 Amazon S3 中读取指定的图像，然后将图像显示到屏幕上。

```
setenv('AWS_ACCESS_KEY_ID', 'YOUR_AWS_ACCESS_KEY_ID');
setenv('AWS_SECRET_ACCESS_KEY', 'YOUR_AWS_SECRET_ACCESS_KEY');

ds = imageDatastore('s3://bucketname/image_datastore/jpegfiles', ...
 'IncludeSubfolders', true, 'LabelSource', 'foldernames');
img = ds.readimage(1);
imshow(img)
```



## 将数据写入到 Amazon S3

以下示例说明如何使用 `tabularTextDatastore` 对象将表格数据从 Amazon S3 读入到 `tall` 数组中，通过移除缺失的条目并排序进行预处理，然后将其写回到 Amazon S3。

```
setenv('AWS_ACCESS_KEY_ID', 'YOUR_AWS_ACCESS_KEY_ID');
setenv('AWS_SECRET_ACCESS_KEY', 'YOUR_AWS_SECRET_ACCESS_KEY');

ds = tabularTextDatastore('s3://bucketname/dataset/airlinesmall.csv', ...
 'TreatAsMissing', 'NA', 'SelectedVariableNames', {'ArrDelay'});
tt = tall(ds);
tt = sortrows(rmmissing(tt));
write('s3://bucketname/preprocessedData/', tt);
```

要读回 `tall` 数据，请使用 `datastore` 函数。

```
ds = datastore('s3://bucketname/preprocessedData/');
tt = tall(ds);
```

## Azure Blob 存储

MATLAB 允许您使用 Azure Blob 存储进行在线文件存储。当指定数据位置时，必须使用以下格式的统一资源定位器 (URL) 指定文件或文件夹的完整路径：

```
wasbs://container@account/path_to_file/file.ext
```

`container@account` 是容器的名称，`path_to_file` 是文件或文件夹的路径。

Azure 通过 Web 服务接口提供数据存储。可使用 blob 来存储 Azure 中的数据文件。有关详细信息，请参阅[什么是 Azure](#)。

### 设置访问权限

要处理 Azure 存储中的远程数据，必须先设置访问权限：

- 1 注册一个 Microsoft Azure 帐户，请参阅[Microsoft Azure 帐户](#)。
- 2 使用 `setenv` 设置以下两个环境变量之一，以设置身份验证详细信息：
  - **MW\_WASB\_SAS\_TOKEN** - 通过共享访问签名 (SAS) 进行身份验证

获得 SAS。有关详细信息，请参阅[使用存储资源管理器管理 Azure Blob 存储资源中的“获取 Blob 容器的 SAS”部分](#)。

在 MATLAB 中，将 **MW\_WASB\_SAS\_TOKEN** 设置为 SAS 查询字符串。例如，

```
setenv MW_WASB_SAS_TOKEN '?st=2017-04-11T09%3A45%3A00Z&se=2017-05-12T09%3A45%3A00Z&sp=rl&sv=2015-12-11&sr=c&sig=
```

您必须将此字符串设置为从 Azure Storage Web UI 或 Explorer 生成的有效 SAS 令牌。

- **MW\_WASB\_SECRET\_KEY** - 通过帐户的两个密钥之一进行身份验证

每个存储帐户都有两个允许管理特权访问权限的密钥。通过设置 **MW\_WASB\_SECRET\_KEY** 环境变量，无需创建 SAS 令牌即可对 MATLAB 授予相同的访问权限。例如：

```
setenv MW_WASB_SECRET_KEY '1234567890ABCDEF1234567890ABCDEF1234567890ABCDEF'
```

如果您使用 Parallel Computing Toolbox，则必须通过在 `parpool`、`batch`、`createJob` 中或 Cluster Profile Manager 中设置 `EnvironmentVariables`，将客户端环境变量复制到集群上的工作进程。

有关详细信息，请参阅[将 Azure 存储与 Azure HDInsight 集群配合使用](#)。

## 从 Azure 读取数据

要从 Azure Blob 存储位置读取数据，请使用以下语法指定位置：

```
wasbs://container@account/path_to_file/file.ext
```

**container@account** 是容器的名称，**path\_to\_file** 是文件或文件夹的路径。

例如，如果您在测试存储帐户 **wasbs://blobContainer@storageAccount.blob.core.windows.net/** 上的文件夹 **/airline** 中有一个文件 **airlinesmall.csv**，则可以使用以下方式创建数据存储：

```
location = 'wasbs://blobContainer@storageAccount.blob.core.windows.net/airline/airlinesmall.csv';
```

```
ds = tabularTextDatastore(location, 'TreatAsMissing', 'NA', ...
 'SelectedVariableNames', {'ArrDelay'});
```

您可以使用 Azure 进行数据存储支持的所有计算，包括直接读取、**mapreduce**、**tall** 数组和深度学习。例如，创建 **ImageDatastore** 对象，从数据存储中读取指定的图像，然后将图像显示到屏幕上。

```
setenv('MW_WASB_SAS_TOKEN', 'YOUR_WASB_SAS_TOKEN');
ds = imageDatastore('wasbs://YourContainer@YourAccount.blob.core.windows.net/', ...
 'IncludeSubfolders', true, 'LabelSource', 'foldernames');
img = ds.readimage(1);
imshow(img)
```

## 将数据写入到 Azure

以下示例说明如何使用 **tabularTextDatastore** 对象将表格数据从 Azure 读入到 **tall** 数组，通过移除缺失的条目并排序进行预处理，然后将其写回到 Azure。

```
setenv('MW_WASB_SAS_TOKEN', 'YOUR_WASB_SAS_TOKEN');
```

```
ds = tabularTextDatastore('wasbs://YourContainer@YourAccount.blob.core.windows.net/dataset/airlinesmall.csv',
 'TreatAsMissing', 'NA', 'SelectedVariableNames', {'ArrDelay'});
tt = tall(ds);
tt = sortrows(rmmissing(tt));
write('wasbs://YourContainer@YourAccount.blob.core.windows.net/preprocessedData/', tt);
```

要读回 **tall** 数据，请使用 **datastore** 函数。

```
ds = datastore('wasbs://YourContainer@YourAccount.blob.core.windows.net/preprocessedData/');
tt = tall(ds);
```

## Hadoop 分布式文件系统

### 指定数据的位置

MATLAB 允许将 Hadoop 分布式文件系统 (HDFS) 用作在线文件存储 Web 服务。当指定数据位置时，必须使用以下格式之一的统一资源定位器 (URL) 指定文件或文件夹的完整路径：

```
hdfs:/path_to_file
```

```
hdfs:///path_to_file
```

```
hdfs://hostname/path_to_file
```

**hostname** 是主机或服务器的名称，**path\_to\_file** 是文件或文件夹的路径。指定 **hostname** 是一个可选操作。未指定 **hostname** 时，Hadoop 使用与 MATLAB 中的 Hadoop 分布式文件系统 (HDFS) 安装相关联的默认主机名称。

例如，您可以使用以下任一命令在 `myserver` 主机上的 `data` 文件夹中为文件 `file1.txt` 创建一个数据存储：

- `ds = tabularTextDatastore('hdfs:///data/file1.txt')`
- `ds = tabularTextDatastore('hdfs://myserver/data/file1.txt')`

如果指定了 `hostname`，则它必须对应于由 Hadoop 集群的 Hadoop XML 配置文件中的 `fs.default.name` 属性定义的名称节点。

您还可以在位置中包括端口号。例如，此位置指定了一台名为 `myserver` 的主机，端口号为 `7867`，在其文件夹 `data` 中包含文件 `file1.txt`：

```
'hdfs://myserver:7867/data/file1.txt'
```

指定的端口号必须与在 HDFS 配置中设置的端口号相匹配。

### 设置 Hadoop 环境变量

在从 HDFS 读取数据之前，先使用 `setenv` 函数将适当的环境变量设置为安装 Hadoop 的文件夹。此文件夹必须可从当前计算机访问。

- 仅使用 Hadoop v1 - 设置 `HADOOP_HOME` 环境变量。
- 仅使用 Hadoop v2 - 设置 `HADOOP_PREFIX` 环境变量。
- 如果同时使用 Hadoop v1 和 Hadoop v2，或者未设置 `HADOOP_HOME` 和 `HADOOP_PREFIX` 环境变量，则设置 `MATLAB_HADOOP_INSTALL` 环境变量。

例如，使用此命令设置 `HADOOP_HOME` 环境变量。`hadoop-folder` 是安装 Hadoop 的文件夹，`/mypath/` 是该文件夹的路径。

```
setenv('HADOOP_HOME','/mypath/hadoop-folder');
```

### Hortonworks 或 Cloudera 上的 HDFS 数据

如果您当前的计算机可以访问 Hortonworks 或 Cloudera® 上的 HDFS 数据，则无需设置 `HADOOP_HOME` 或 `HADOOP_PREFIX` 环境变量。MATLAB 会在使用 Hortonworks 或 Cloudera 应用程序边缘节点时自动分配这些环境变量。

### 防止从内存中清除代码

当从 HDFS 读取时或在本地读取序列文件时，`datastore` 函数调用 `javaaddpath` 命令。此命令将执行以下操作：

- 清除动态类路径中的文件定义的所有 Java 类的定义
- 从基础工作区中删除所有全局变量和变量
- 从内存中删除所有编译的脚本、函数和 MEX 函数

要防止清除持久变量、代码文件或 MEX 文件，请使用 `mlock` 函数。

### 将数据写入到 HDFS

此示例说明如何使用 `tabularTextDatastore` 对象将数据写入 HDFS 位置。使用 `write` 函数将 `tall` 数组和分布式数组写入到 Hadoop 分布式文件系统。当对分布式数组或 `tall` 数组调用此函数时，必须指定一个 HDFS 文件夹的完整路径。以下示例说明如何将表格数据从 HDFS 读入到 `tall` 数组，通过移除缺失的条目并排序进行预处理，然后将其写回到 HDFS。

```
ds = tabularTextDatastore('hdfs://myserver/some/path/dataset/airlinesmall.csv', ...
 'TreatAsMissing', 'NA', 'SelectedVariableNames', {'ArrDelay'});
tt = tall(ds);
tt = sortrows(rmmissing(tt));
write('hdfs://myserver/some/path/preprocessedData', tt);
```

要读回 tall 数据，请使用 `datastore` 函数。

```
ds = datastore('hdfs://myserver/some/path/preprocessedData');
tt = tall(ds);
```

### 另请参阅

`datastore` | `tabularTextDatastore` | `write` | `imageDatastore` | `imread` | `imshow` | `javaaddpath` | `mlock` | `setenv`

### 相关示例

- “Read and Analyze Hadoop Sequence File”
- “将深度学习数据上传到云” (Deep Learning Toolbox)

## 读取和分析大型表格文本文件

以下示例说明如何为包含表格数据的大型文本文件创建数据存储，然后采用逐个块或逐个文件的方式读取和处理数据。

### 创建数据存储

使用 `tabularTextDatastore` 函数基于样本文件 `airlinesmall.csv` 创建一个数据存储。创建数据存储后，可以指定将数据中的文本 NA 作为缺失数据处理。

```
ds = tabularTextDatastore('airlinesmall.csv','TreatAsMissing','NA');
```

您可以通过更改数据存储的属性来修改其属性。修改 `MissingValue` 属性，指定将缺失值处理为 0。

```
ds.MissingValue = 0;
```

在此示例中，选择到港延误变量 `ArrDelay` 作为要关注的变量。

```
ds.SelectedVariableNames = 'ArrDelay';
```

使用 `preview` 函数预览数据。此函数不影响数据存储的状态。

```
data = preview(ds)
```

```
data=8×1 table
```

```
ArrDelay
```

```
8
8
21
13
4
59
3
11
```

### 读取数据子集

默认情况下，`read` 一次从 `TabularTextDatastore` 读取 20000 行。要在每次调用 `read` 时读取不同的行数，请修改 `ds` 的 `ReadSize` 属性。

```
ds.ReadSize = 15000;
```

在 `while` 循环中使用 `read` 函数从 `ds` 中读取数据子集。该循环会持续执行，直到 `hasdata(ds)` 返回 `false`。

```
sums = [];
counts = [];
while hasdata(ds)
 T = read(ds);

 sums(end+1) = sum(T.ArrDelay);
 counts(end+1) = length(T.ArrDelay);
end
```

计算平均到港延误。

```
avgArrivalDelay = sum(sums)/sum(counts)
```

```
avgArrivalDelay = 6.9670
```

重置数据存储以允许重新读取数据。

```
reset(ds)
```

### 一次读取一个文件

一个数据存储可以包含多个文件，每个文件具有不同的行数。可通过将 **ReadSize** 属性设为 'file'，一次从数据存储读取一个完整的文件。

```
ds.ReadSize = 'file';
```

在将 **ReadSize** 的值从数值更改为 'file' 时，MATLAB® 会重置数据存储（反之亦然）。

像前面一样，在 **while** 循环中使用 **read** 函数读取 **ds**，并计算平均到港延误。

```
sums = [];
counts = [];
while hasdata(ds)
 T = read(ds);

 sums(end+1) = sum(T.ArrDelay);
 counts(end+1) = length(T.ArrDelay);
end
avgArrivalDelay = sum(sums)/sum(counts)

avgArrivalDelay = 6.9670
```

### 另请参阅

[tabularTextDatastore](#) | [tall](#) | [mapreduce](#)

### 相关示例

- “使用 **tall** 数组处理无法放入内存的数据”（第 13-25 页）
- “MapReduce 快速入门”（第 13-2 页）

## 读取和分析图像文件

此示例说明如何为图像集合创建数据存储，读取图像文件，并找到具有最大平均色调、饱和度和亮度 (HSV) 的图像。有关使用 `mapreduce` 函数进行图像处理的类似示例，请参阅“Compute Maximum Average HSV of Images with MapReduce”。

确定两个 MATLAB® 目录并创建一个数据存储，其中包含在这些目录中具有 `.jpg`、`.tif` 和 `.png` 扩展名的图像。

```
location1 = fullfile(matlabroot,'toolbox','matlab','demos');
location2 = fullfile(matlabroot,'toolbox','matlab','imagesci');

ds = imageDatastore([location1,location2],'FileExtensions',{'jpg','tif','png'});
```

初始化最大平均 HSV 值和对应的图像数据。

```
maxAvgH = 0;
maxAvgS = 0;
maxAvgV = 0;
```

```
dataH = 0;
dataS = 0;
dataV = 0;
```

对于集合中的每个图像，读取图像文件并计算所有图像像素的平均 HSV 值。如果平均值大于先前图像的平均值，则将其记录为新的最大值 (`maxAvgH`、`maxAvgS` 或 `maxAvgV`)，并记录对应的图像数据 (`dataH`、`dataS` 或 `dataV`)。

```
for i = 1:length(ds.Files)
 data = readimage(ds,i); % Read the ith image
 if ~ismatrix(data) % Only process 3-dimensional color data
 hsv = rgb2hsv(data); % Compute the HSV values from the RGB data

 h = hsv(:,:,1); % Extract the HSV values
 s = hsv(:,:,2);
 v = hsv(:,:,3);

 avgH = mean(h(:)); % Find the average HSV values across the image
 avgS = mean(s(:));
 avgV = mean(v(:));

 if avgH > maxAvgH % Check for new maximum average hue
 maxAvgH = avgH;
 dataH = data;
 end

 if avgS > maxAvgS % Check for new maximum average saturation
 maxAvgS = avgS;
 dataS = data;
 end

 if avgV > maxAvgV % Check for new maximum average brightness
 maxAvgV = avgV;
 dataV = data;
 end
 end
end
```

```
end
end
```

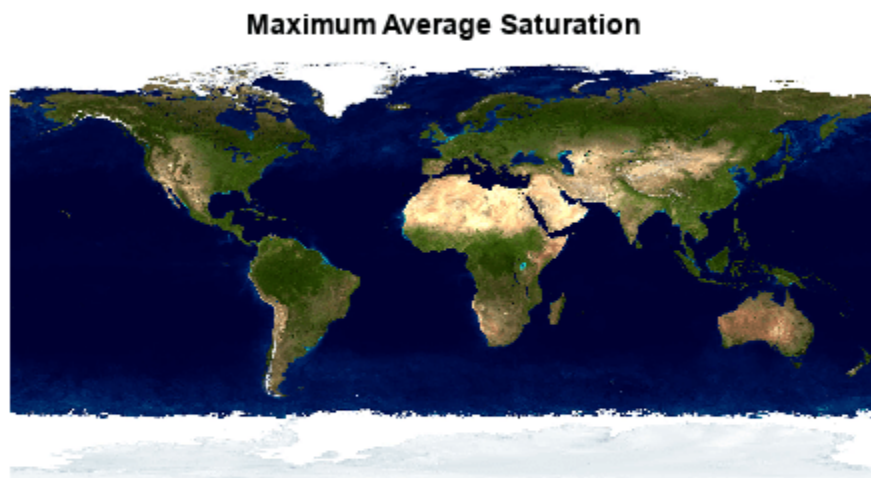
查看具有最大平均色调、饱和度和亮度的图像。

```
imshow(dataH,'InitialMagnification','fit');
title('Maximum Average Hue')
```



```
figure
imshow(dataS,'InitialMagnification','fit');
title('Maximum Average Saturation');
```





```
figure
imshow(dataV,'InitialMagnification','fit');
title('Maximum Average Brightness');
```

Maximum Average Brightness



### 另请参阅

`imageDatastore` | `tall` | `mapreduce`

### 相关示例

- “使用 `tall` 数组处理无法放入内存的数据” (第 13-25 页)
- “MapReduce 快速入门” (第 13-2 页)
- “Compute Maximum Average HSV of Images with MapReduce”

## 使用 tall 数组处理无法放入内存的数据

tall 数组用于处理由 **datastore** 支持的无法放入内存的数据。数据存储允许以单个放入内存的小块形式处理大型数据集，而不是一次将整个数据集加载到内存中。tall 数组扩展了此功能，允许使用公共函数处理无法放入内存的数据。

### 什么是 tall 数组？

由于数据不是一次加载到内存中，因此 tall 数组在第一维中可以是任意大的（即可以具有任意数量的行）。tall 数组允许您以一种直观方式处理大型数据集，这与处理内存 MATLAB 数组的方式类似；您不必为超大的数据量而专门编写特殊的代码（例如使用 MapReduce 之类的技术）。许多核心运算符和函数能够像处理内存数组一样处理 tall 数组。MATLAB 一次只处理一小块数据，而在后台执行所有的数据分块和处理，因此可以对大型数据集应用常见的运算表达式，如 **A+B**。

### tall 数组的好处

与内存数组不同，tall 数组通常不执行计算，直到您使用 **gather** 函数请求执行计算后才会真正计算。这种延迟计算可快速处理大型数据集。当使用 **gather** 最终请求输出时，MATLAB 尽可能合并排队的计算，并执行最少次数的数据遍历操作。遍历数据的次数会极大地影响执行时间，因此建议仅在必要时才请求输出。

**注意** 由于 **gather** 将结果作为内存 MATLAB 数组返回，因此遵循标准内存考量即可。如果 **gather** 返回的结果太大，MATLAB 可能会耗尽内存。

### 创建 tall 表

除了可以包含任意数量的行以外，tall 表与内存 MATLAB 常规表并无不同。要基于大型数据集创建 tall 表，首先需要为数据创建数据存储。如果数据存储 **ds** 包含表格数据，则 **tall(ds)** 返回包含这些数据的 tall 表或 tall 时间表。有关创建数据存储的详细信息，请参阅“数据存储”。

创建指向航空公司航班数据表格文件的电子表格数据存储。对于包含文件集合的文件夹，可以指定整个文件夹位置，或使用通配符 **'\*.csv'** 来包括数据存储中具有相同文件扩展名的多个文件。通过将 **'NA'** 值视为缺失数据来清理数据，以便 **tabularTextDatastore** 用 **NaN** 值替换它们。另外，将几个文本变量的格式设置为 **%s**，以便 **tabularTextDatastore** 将它们作为字符向量的元胞数组读取。

```
ds = tabularTextDatastore('airlinesmall.csv');
ds.TreatAsMissing = 'NA';
ds.SelectedFormats{strcmp(ds.SelectedVariableNames,'TailNum')} = '%s';
ds.SelectedFormats{strcmp(ds.SelectedVariableNames,'CancellationCode')} = '%s';
```

基于数据存储创建 tall 表。当对此 tall 表执行计算时，基础数据存储读取数据块并将其传递给 tall 表进行处理。数据存储和 tall 表都不保留任何基础数据。

```
tt = tall(ds)
```

```
tt =
```

```
M×29 tall table
```

Year	Month	DayofMonth	DayOfWeek	DepTime	CRSDepTime	ArrTime	CRSArrTime	UniqueCarrier				
1987	10	21	3	642	630	735	727	'PS'	1503	'NA'	53	57

1987	10	26	1	1021	1020	1124	1116	'PS'	1550	'NA'	63	56
1987	10	23	5	2055	2035	2218	2157	'PS'	1589	'NA'	83	82
1987	10	23	5	1332	1320	1431	1418	'PS'	1655	'NA'	59	58
1987	10	22	4	629	630	746	742	'PS'	1702	'NA'	77	72
1987	10	28	3	1446	1343	1547	1448	'PS'	1729	'NA'	61	65
1987	10	8	4	928	930	1052	1049	'PS'	1763	'NA'	84	79
1987	10	10	6	859	900	1134	1123	'PS'	1800	'NA'	155	143
:	:	:	:	:	:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:	:	:	:	:	:

上面显示的结果指示行数 M 当前未知。MATLAB 显示了一些行，垂直省略号 : 则表示 tall 表中还存在许多当前未显示的行。

创建 tall 时间表

如果正在处理的数据有与每一行数据相关联的时间，则您可以使用 tall 时间表来处理这些数据。有关创建 tall 时间表的信息，请参阅扩展功能 (timetable)。

在本例中，tall 表 tt 具有与每一行相关联的时间，但这些时间分解为几个表变量，例如 Year、Month、DayofMonth 等。将所有这些日期时间信息组合为一个基于出发时间 DepTime 的新 tall 日期时间变量 Dates。然后，使用 Dates 作为行时间创建 tall 时间表。由于 Dates 是表中唯一的日期时间变量，因此 table2timetable 函数自动将其用于行时间。

```
hrs = (tt.DepTime - mod(tt.DepTime,100))/100;
mins = mod(tt.DepTime,100);
tt.Dates = datetime(tt.Year, tt.Month, tt.DayofMonth, hrs, mins, 0);
tt(:,1:8) = [];
TT = table2timetable(tt)
```

TT =

M×21 tall timetable

Dates	UniqueCarrier	FlightNum	TailNum	ActualElapsedTime	CRSElapsedTime	AirTime	ArrDelay
21-Oct-1987 06:42:00	'PS'	1503	'NA'	53	57	NaN	8
26-Oct-1987 10:21:00	'PS'	1550	'NA'	63	56	NaN	8
23-Oct-1987 20:55:00	'PS'	1589	'NA'	83	82	NaN	21
23-Oct-1987 13:32:00	'PS'	1655	'NA'	59	58	NaN	13
22-Oct-1987 06:29:00	'PS'	1702	'NA'	77	72	NaN	4
28-Oct-1987 14:46:00	'PS'	1729	'NA'	61	65	NaN	59
08-Oct-1987 09:28:00	'PS'	1763	'NA'	84	79	NaN	3
10-Oct-1987 08:59:00	'PS'	1800	'NA'	155	143	NaN	11
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:

创建 tall 数组

当从 tall 表或 tall 时间表中提取变量时，其结果是一个具有适当基础数据类型的 tall 数组。tall 数组可以是数字、逻辑、日期时间、持续时间、日历持续时间、分类、字符串或元胞数组。此外，可以按照 tA = tall(A) 条件将内存数组 A 转换为 tall 数组。内存数组 A 必须具有受支持的数据类型之一。

从 tall 时间表 TT 提取到港延误 ArrDelay。这将创建一个新的 tall 数组变量，其基础数据类型为 double。

```

a = TT.ArrDelay

a =

 M×1 tall double column vector

 8
 8
 21
 13
 4
 59
 3
 11
 :
 :

```

`classUnderlying` 和 `isaUnderlying` 函数可用于确定 tall 数组的基础数据类型。

## 延迟计算

tall 数组的一个重要特征是，在处理它们时大多数运算不会立即执行。这些运算看起来执行速度很快，是因为实际计算被延迟，直到特别请求时才执行计算。可以使用 `gather` 函数（将结果放入内存）或 `write` 函数（将结果写入磁盘）来启动 tall 数组的计算。这种延迟计算非常重要，因为对一个包含十亿行的 tall 数组执行一个类似 `size(X)` 的简单命令都是一次耗时的计算过程。

在处理 tall 数组时，MATLAB 跟踪要执行的所有操作。然后，此信息用于优化在使用 `gather` 函数请求输出时需要的遍历数据次数。因此，通常情况下应使用未经计算的 tall 数组，并仅在需要时请求输出。有关详细信息，请参阅“Deferred Evaluation of Tall Arrays”。

计算到港延误的均值和标准差。使用这些值来构建在均值的一个标准差内的延误的上限和下限阈值。请注意，每个运算的结果表示该数组尚未真正计算。

```
m = mean(a,'omitnan')
```

```
m =
```

```
tall double
```

```
?
```

Preview deferred. Learn more.

```
s = std(a,'omitnan')
```

```
s =
```

```
tall
```

```
?
```

Preview deferred. Learn more.

```
one_sigma_bounds = [m-s m m+s]
```

```
one_sigma_bounds =
```

M×N×... tall array

```
? ? ? ...
? ? ? ...
? ? ? ...
: : :
: : :
```

Preview deferred. Learn more.

## 用 gather 计算

延迟计算的好处是，当到了 MATLAB 执行计算的时间，通常可以合并多个运算，这样能最大限度地降低遍历数据的次数。因此，即使您执行大量运算，MATLAB 也仅在必要时才会对数据进行额外次数的遍历。

**gather** 函数强制执行所有排队的运算，并将生成的输出载入内存中。因此，您可以考虑将 **gather** 作为 tall 数组和内存数组之间的桥梁。例如，您不能使用 tall 逻辑数组控制 **if** 或 **while** 循环，但一旦使用 **gather** 计算该数组后，它就成为内存逻辑数组，从而可用于上述控制目的。

由于 **gather** 在 MATLAB 中返回整个结果，因此应确保结果可放入内存中。

使用 **gather** 计算 **one\_sigma\_bounds** 并将结果放入内存中。在本例中，**one\_sigma\_bounds** 需要几个运算来计算，但 MATLAB 将这些运算合并在一次遍历数据中进行。由于此示例中的数据较小，因此 **gather** 可快速执行。然而，随着数据大小的增加，消除数据遍历会变得更 valuable。

```
sig1 = gather(one_sigma_bounds)
```

Evaluating tall expression using the Local MATLAB Session:

- Pass 1 of 1: Completed in 1.5 sec

Evaluation completed in 1.8 sec

```
sig1 =
```

```
-23.4572 7.1201 37.6975
```

如果要同时计算几个 tall 数组，可以给 **gather** 指定多个输入和输出。这种方法比多次调用 **gather** 更快。例如，计算最小和最大到港延误。如果单独计算，则计算每个值都需要遍历一次数据，总共两次。然而，同时计算两个值则仅需要一次数据遍历。

```
[max_delay, min_delay] = gather(max(a),min(a))
```

Evaluating tall expression using the Local MATLAB Session:

- Pass 1 of 1: Completed in 1.1 sec

Evaluation completed in 1.1 sec

```
max_delay =
```

```
1014
```

```
min_delay =
```

```
-64
```

这些结果表明，大多数航班平均延误约 7 分钟。但是在一个标准差之内，航班最多可能延误 37 分钟或早到 23 分钟。数据集中最快的航班大约早到一小时，而最迟的航班延误了许多小时。

## 保存、加载和检查 tall 数组

`save` 函数保存 tall 数组的状态，但不复制任何数据。生成的 `.mat` 文件通常很小。但是，原始数据文件必须保留在原位置，以便以后使用 `load`。

`write` 函数创建数据的副本，并将副本保存为文件集合，这可能会占用大量磁盘空间。`write` 对 tall 数组执行所有待处理运算，以在写入前计算出值。一旦 `write` 完成数据复制，就与原始裸数据再无关联。因此，即使原始裸数据不再可用，也可以基于已写入的文件重新创建 tall 数组。

您可以创建一个指向文件写入位置的新数据存储，基于已写入的文件重新创建 tall 数组。此功能支持创建 tall 数组数据的检查点或快照。创建检查点是保存数据预处理结果的好方法，这样数据就能以更有效的方式加载。

如果有一个 tall 数组 `TA`，则可以用以下命令将它写入文件夹 `location`：

```
write(location,TA);
```

稍后，要基于写入的文件重新构造 `TA`，请使用命令：

```
ds = datastore(location);
TA = tall(ds);
```

此外，可以使用 `write` 函数触发 tall 数组的计算，并将结果写入磁盘。`write` 的这种用法类似于 `gather`，但 `write` 不会将任何结果写入内存中。

## 支持函数

大多数核心函数采用与处理内存数组相同的方法处理 tall 数组。但在某些情况下，函数处理 tall 数组的方法可能较为特殊，或存在局限性。可以通过查看函数参考页底部的**扩展功能**部分（例如，查看 `filloutliers`），判断函数是否支持 tall 数组，以及是否具有任何限制。

有关支持 tall 数组的所有 MATLAB 函数的筛选列表，请参阅函数列表（tall 数组）。

有几个工具箱也支持 tall 数组，您可以使用这些工具箱完成诸如编写机器学习算法、部署独立 App 以及并行运行或在集群上运行计算之类的任务。有关详细信息，请参阅“Extend Tall Arrays with Other Products”。

## 另请参阅

`gather` | `tall` | `datastore` | `table` | `mapreducer`

## 详细信息

- “Index and View Tall Array Elements”
- “Visualization of Tall Arrays”

## 在 MATLAB 中使用 tall 数组分析大数据

此示例说明如何在 MATLAB® 中使用 tall 数组处理大数据。您可以使用 tall 数组对无法放入内存的不同类型的数据执行各种计算。这些计算包括基本计算，以及 Statistics and Machine Learning Toolbox™ 中的机器学习算法。

此示例在一台计算机上对一小部分数据进行运算，然后扩展到分析所有数据集。不过，这种分析方法可以进一步扩展，以处理无法读入内存的大型数据集，或在像 Apache Spark™ 这样的系统上进行处理。

### tall 数组简介

tall 数组和 tall 表用于处理包含任意行数的无法放入内存的数据。使用 tall 数组和表，您能够以类似内存 MATLAB® 数组的方式处理大型数据集，而不必为超大的数据量专门编写代码。不同之处在于，tall 数组在您请求执行计算之前通常不执行计算。

这种延迟计算使 MATLAB 能够尽可能合并排队的计算，并执行最少次数的数据遍历操作。由于遍历数据的次数会极大地影响执行时间，建议仅在必要时才请求输出。

### 为文件集合创建数据存储

创建数据存储使您能够访问数据集。数据存储可以处理任意大的数据量，数据甚至可以分布在多个文件夹的多个文件中。您可以为大多数类型的文件创建数据存储，包括表格文本文件（此处演示）、电子表格、图像、SQL 数据库（需要 Database Toolbox™）或 Hadoop® 序列文件等的集合。

为包含航班数据的 .csv 文件创建一个数据存储。将 'NA' 值视为缺失，以便 tabularTextDatastore 用 NaN 值替换它们。选择关注的变量，并为 Origin 和 Dest 变量指定分类数据类型。预览内容。

```
ds = tabularTextDatastore('airlinesmall.csv');
ds.TreatAsMissing = 'NA';
ds.SelectedVariableNames = {'Year','Month','ArrDelay','DepDelay','Origin','Dest'};
ds.SelectedFormats(5:6) = {'%C','%C'};
pre = preview(ds)
```

```
pre=8x6 table
 Year Month ArrDelay DepDelay Origin Dest

1987 10 8 12 LAX SJC
1987 10 8 1 SJC BUR
1987 10 21 20 SAN SMF
1987 10 13 12 BUR SJC
1987 10 4 -1 SMF LAX
1987 10 59 63 LAX SJC
1987 10 3 -2 SAN SFO
1987 10 11 -1 SEA LAX
```

### 创建 tall 数组

除了可以包含任意数量的行以外，tall 数组与内存 MATLAB 数组并无不同。tall 数组可以包含数值、逻辑值、日期时间、持续时间、calendarDuration、分类或字符串数据。此外，您也可以将任何内存数组转换为 tall 数组。（内存数组 A 必须是受支持的数据类型之一。）

tall 数组的基础类基于支持它的数据存储的类型。例如，如果数据存储 ds 包含表格数据，则 tall(ds) 返回包含这些数据的 tall 表。



```
tt = tall(ds)

tt =

Mx6 tall table

Year Month ArrDelay DepDelay Origin Dest

? ? ? ? ? ?
? ? ? ? ? ?
? ? ? ? ? ?
: : : : : :
: : : : : :
```

显示内容指示基础数据类型，并包括前几行数据。表的大小显示为 “M×6”，表示 MATLAB 尚不了解有多少行数据。

对 tall 数组执行计算

您可以像处理内存 MATLAB 数组和表一样，处理 tall 数组和 tall 表。

tall 数组的一个重要特征是，在处理它们时，MATLAB 不会立即执行大多数运算。这些运算看起来执行速度很快，是因为实际计算被延迟，直到特别请求输出时才执行计算。这种延迟计算非常重要，因为对一个包含十亿行的 tall 数组执行一个类似 `size(X)` 的简单命令都是一次耗时的计算过程。

当您处理 tall 数组时，MATLAB 会跟踪要执行的所有运算并优化数据遍历次数。因此，通常情况下应使用未经计算的 tall 数组，并仅在需要时请求输出。在您请求计算和显示数组之前，MATLAB 不知道未计算的 tall 数组的内容或大小。

计算离港延误均值。

```
mDep = mean(tt.DepDelay,'omitnan')

mDep =

tall double

?
```

将结果收集到工作区中

延迟计算的好处是，当到了 MATLAB 执行计算的时间，通常可以合并多个运算，这样能最大限度地降低遍历数据的次数。因此，即使您执行大量运算，MATLAB 也仅在必要时才会对数据进行额外次数的遍历。

`gather` 函数强制执行所有排队的运算，并将生成的输出载入内存中。由于 `gather` 在 MATLAB 中返回整个结果，因此应确保结果可放入内存中。例如，对使用 `sum`、`min` 和 `mean` 等函数进行了归约的 tall 数组使用 `gather`。

使用 `gather` 计算离港延误均值并将计算结果放入内存中。此计算只需遍历一次数据，但其他计算可能需要多次遍历数据。MATLAB 确定计算的最佳遍历次数，并在命令行中显示该信息。

```
mDep = gather(mDep)

Evaluating tall expression using the Local MATLAB Session:
- Pass 1 of 2: Completed in 0.58 sec
```

- Pass 2 of 2: Completed in 0.5 sec  
Evaluation completed in 1.5 sec

mDep = 8.1860

选择 tall 数组的子集

您可以通过下标或索引从 tall 数组中提取值。您可以从顶部或底部开始对数组进行索引，或使用逻辑索引对数组进行索引。函数 `head` 和 `tail` 可以替代索引，您可以用它们来探查 tall 数组的开头部分和结尾部分。同时收集这两个变量可避免对数据进行多余的遍历。

```
h = head(tt);
tl = tail(tt);
[h,tl] = gather(h,tl)
```

Evaluating tall expression using the Local MATLAB Session:  
- Pass 1 of 1: Completed in 0.41 sec  
Evaluation completed in 0.55 sec

h=8×6 table

Year	Month	ArrDelay	DepDelay	Origin	Dest
1987	10	8	12	LAX	SJC
1987	10	8	1	SJC	BUR
1987	10	21	20	SAN	SMF
1987	10	13	12	BUR	SJC
1987	10	4	-1	SMF	LAX
1987	10	59	63	LAX	SJC
1987	10	3	-2	SAN	SFO
1987	10	11	-1	SEA	LAX

tl=8×6 table

Year	Month	ArrDelay	DepDelay	Origin	Dest
2008	12	14	1	DAB	ATL
2008	12	-8	-1	ATL	TPA
2008	12	1	9	ATL	CLT
2008	12	-8	-4	ATL	CLT
2008	12	15	-2	BOS	LGA
2008	12	-15	-1	SFO	ATL
2008	12	-12	1	DAB	ATL
2008	12	-1	11	ATL	IAD

在扩展到完整数据集之前，请使用 `head` 从针对原型代码的数据中选择包含 10000 行数据的子集。

```
ttSubset = head(tt,10000);
```

按条件选择数据

您可以对 tall 数组使用典型的逻辑运算，这对于使用逻辑索引选择相关数据或删除离群值非常有用。逻辑表达式会创建一个 tall 逻辑向量，然后系统使用它进行下标索引，从而标识条件为 `true` 的行。

通过将分类变量 `Origin` 的元素与值 `'BOS'` 进行比较，仅选择飞离波士顿的航班。

```
idx = (ttSubset.Origin == 'BOS');
bosflights = ttSubset(idx,:)
```

```
bosflights =
```

```
207x6 tall table
```

Year	Month	ArrDelay	DepDelay	Origin	Dest
1987	10	-8	0	BOS	LGA
1987	10	-13	-1	BOS	LGA
1987	10	12	11	BOS	BWI
1987	10	-3	0	BOS	EWR
1987	10	-5	0	BOS	ORD
1987	10	31	19	BOS	PHL
1987	10	-3	0	BOS	CLE
1987	11	5	5	BOS	STL
:	:	:	:	:	:
:	:	:	:	:	:

您可以使用相同的索引方法从 tall 数组中删除包含缺失数据或 NaN 值的行。

```
idx = any(ismissing(ttSubset),2);
ttSubset(idx,:) = [];
```

### 确定最大延误

由于大数据的性质，使用 `sort` 或 `sortrows` 等传统方法对所有数据进行排序的效率比较低。但是，用于 tall 数组的 `topkrows` 函数会以排序后的顺序返回前 `k` 行。

计算前 10 大离港延误。

```
biggestDelays = topkrows(ttSubset,10,'DepDelay');
biggestDelays = gather(biggestDelays)
```

```
Evaluating tall expression using the Local MATLAB Session:
Evaluation completed in 0.047 sec
```

```
biggestDelays=10x6 table
```

Year	Month	ArrDelay	DepDelay	Origin	Dest
1988	3	772	785	ORD	LEX
1989	3	453	447	MDT	ORD
1988	12	397	425	SJU	BWI
1987	12	339	360	DEN	STL
1988	3	261	273	PHL	ROC
1988	7	261	268	BWI	PBI
1988	2	257	253	ORD	BTW
1988	3	236	240	EWR	FLL
1989	2	263	227	BNA	MOB
1989	6	224	225	DFW	JAX

### 可视化 tall 数组中的数据

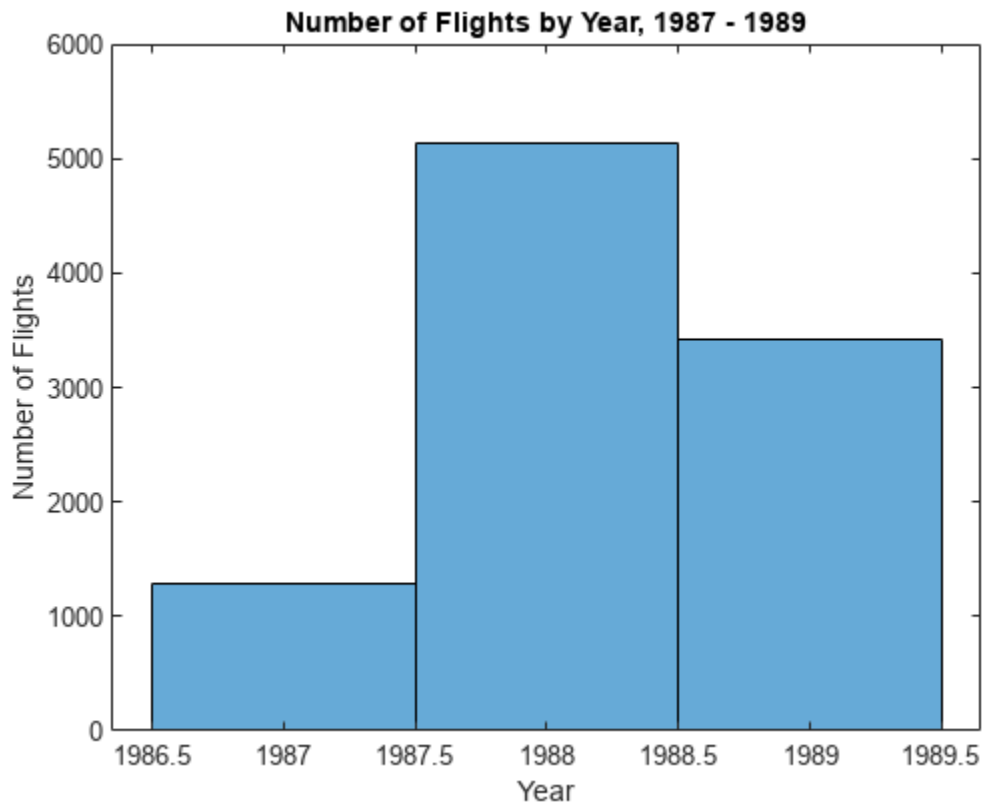
对于大型数据集而言，绘制其中的每个数据点并不可行。因此，tall 数组的可视化需要使用采样或分 bin 来减少数据点的数量。

使用直方图可视化每年的航班数量。可视化函数会在您调用它们时立即遍历数据并计算解，因此不需要 `gather`。

```
histogram(ttSubset.Year,'BinMethod','integers')
```

Evaluating tall expression using the Local MATLAB Session:  
Evaluation completed in 0.32 sec

```
xlabel('Year')
ylabel('Number of Flights')
title('Number of Flights by Year, 1987 - 1989')
```



### 扩展到整个数据集

除了使用从 `head` 返回的较小数据之外，您还可以使用 `tall(ds)` 的结果来扩展到对整个数据集执行计算。

```
tt = tall(ds);
idx = any(ismissing(tt),2);
tt(idx,:) = [];
mnDelay = mean(tt.DepDelay,'omitnan');
biggestDelays = topkrows(tt,10,'DepDelay');
[mnDelay,biggestDelays] = gather(mnDelay,biggestDelays)
```

Evaluating tall expression using the Local MATLAB Session:  
- Pass 1 of 2: Completed in 0.27 sec  
- Pass 2 of 2: Completed in 0.51 sec  
Evaluation completed in 0.87 sec

```
mnDelay = 8.1310
```

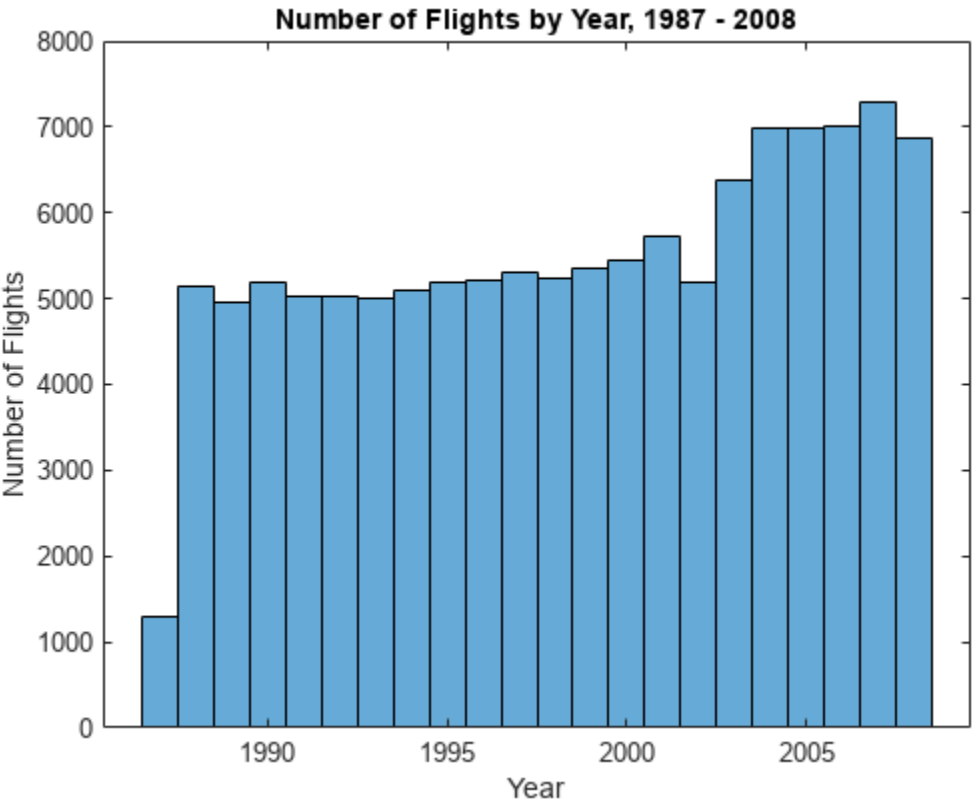
biggestDelays=10×6 table

Year	Month	ArrDelay	DepDelay	Origin	Dest
1991	3	-8	1438	MCO	BWI
1998	12	-12	1433	CVG	ORF
1995	11	1014	1014	HNL	LAX
2007	4	914	924	JFK	DTW
2001	4	887	884	MCO	DTW
2008	7	845	855	CMH	ORD
1988	3	772	785	ORD	LEX
2008	4	710	713	EWR	RDU
1998	10	679	673	MCI	DFW
2006	6	603	626	ABQ	PHX

```
histogram(tt.Year,'BinMethod','integers')
```

Evaluating tall expression using the Local MATLAB Session:  
- Pass 1 of 2: Completed in 0.72 sec  
- Pass 2 of 2: Completed in 0.33 sec  
Evaluation completed in 1.2 sec

```
xlabel('Year')
ylabel('Number of Flights')
title('Number of Flights by Year, 1987 - 2008')
```



使用 `histogram2` 进一步按月细分整个数据集的航班数量。由于 `Month` 和 `Year` 的 bin 事先已知，请指定 bin 边界以避免多余的数据遍历。

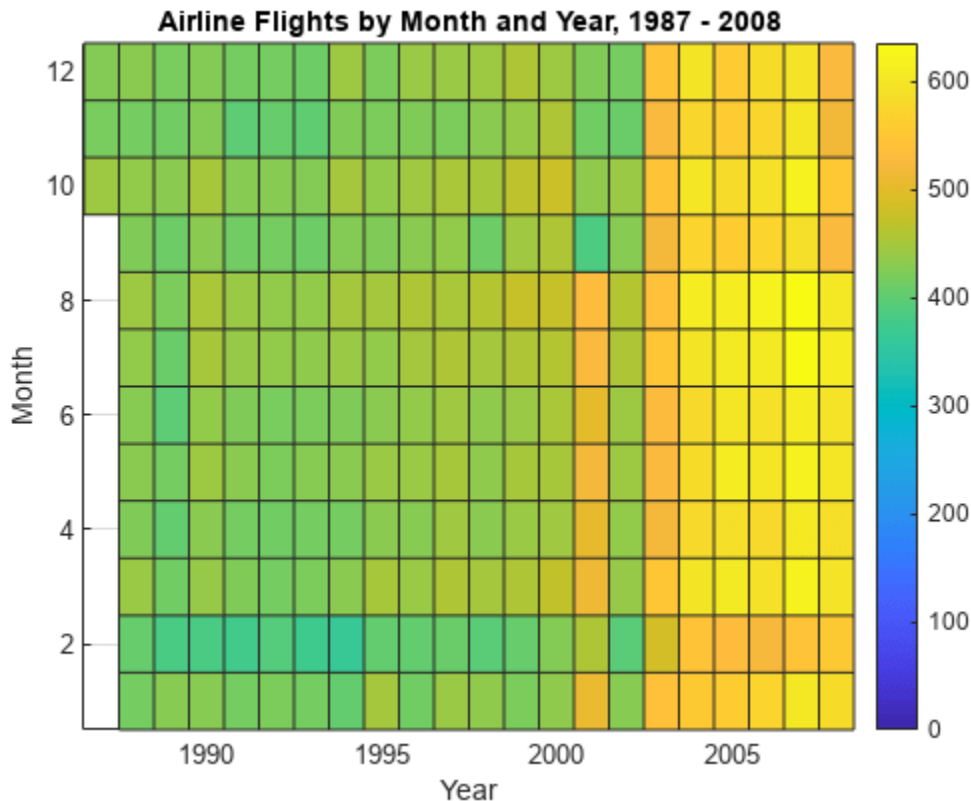
```
year_edges = 1986.5:2008.5;
month_edges = 0.5:12.5;
histogram2(tt.Year,tt.Month,year_edges,month_edges,'DisplayStyle','tile')
```

Evaluating tall expression using the Local MATLAB Session:

- Pass 1 of 1: Completed in 0.55 sec

Evaluation completed in 0.6 sec

```
colorbar
xlabel('Year')
ylabel('Month')
title('Airline Flights by Month and Year, 1987 - 2008')
```



### 使用 tall 数组进行数据分析和机器学习

使用 Statistics and Machine Learning Toolbox™ 中的函数，您可以对 tall 数组执行更复杂的统计分析，包括计算预测分析和执行机器学习。

有关详细信息，请参阅“使用 tall 数组的大数据分析” (Statistics and Machine Learning Toolbox)。

### 扩展到大数据系统

MATLAB 中 tall 数组的一个关键功能是与大数据平台的连接，例如计算集群和 Apache Spark™。

此示例仅粗略涉及如何使用 tall 数组处理大数据。有关使用以下产品的详细信息，请参阅“Extend Tall Arrays with Other Products”：

- Statistics and Machine Learning Toolbox™
- Database Toolbox™
- Parallel Computing Toolbox™
- MATLAB® Parallel Server™
- MATLAB Compiler™

## 另请参阅

tall

## 详细信息

- “使用 tall 数组处理无法放入内存的数据”（第 13-25 页）





# MATLAB 中的 TCP/IP 支持

---

- “创建 TCP/IP 客户端并配置设置” (第 14-2 页)
- “通过 TCP/IP 接口写入和读取数据” (第 14-6 页)

## 创建 TCP/IP 客户端并配置设置

MATLAB TCP/IP 客户端支持从 MATLAB 连接到远程主机或硬件以读取和写入数据。典型的工作流如下：

- 创建到服务器或硬件的 TCP/IP 连接。
- 配置连接（如有必要）。
- 执行读取和写入操作。
- 清除并关闭连接。

要通过 TCP/IP 接口进行通信，请首先创建一个 `tcpclient` 对象。

```
t = tcpclient(address,port);
```

地址可以为远程主机名称或远程 IP 地址。在这两种情况下，端口必须为 1 到 65535 之间的正整数。

### 使用主机名称创建对象

使用所示的主机地址和端口 80 创建 TCP/IP 对象 `t`。

```
t = tcpclient('www.mathworks.com',80)
```

```
t =
tcpclient with properties:
```

```
 Address: 'www.mathworks.com'
 Port: 80
NumBytesAvailable: 0
```

```
Show all properties, functions
```

当您使用主机名（如指定的 Web 地址或 `'localhost'`）连接时，IP 地址默认为 IPv6 格式。如果您要连接的服务器需要 IPv4 格式，连接将失败。对于 IPv4，您可以通过指定显式 IP 地址而不是主机名来创建连接。

### 使用 IP 地址创建对象

使用所示的 IP 地址和端口 80 创建 TCP/IP 对象 `t`。

```
t = tcpclient('144.212.130.17',80)
```

```
t =
tcpclient with properties:
```

```
 Address: '144.212.130.17'
 Port: 80
NumBytesAvailable: 0
```

```
Show all properties, functions
```

## 设置超时属性

创建对象并使用名称-值对组参数来设置 **Timeout** 值。**Timeout** 参数以秒为单位指定完成读写操作的等待时间，默认值为 10。您可以在创建对象期间或在创建对象之后更改该值。

创建一个超时值为 20 秒的 TCP/IP 对象。

```
t = tcpclient("144.212.130.17",80,"Timeout",20)
```

```
t =
 tcpclient with properties:

 Address: '144.212.130.17'
 Port: 80
 NumBytesAvailable: 0

 Show all properties, functions
```

查看 **Timeout** 属性。

```
t.Timeout
```

```
ans =

 20
```

输出会反映 **Timeout** 属性更改。

## 设置连接超时属性

创建对象并使用名称-值对组参数来设置 **ConnectTimeout** 值。**ConnectTimeout** 参数指定等待对指定远程主机的连接请求是成功还是失败的最长时间（以秒为单位）。该值必须大于或等于 1。如果未指定 **ConnectTimeout**，则默认值为 **Inf**。仅可在对象创建期间指定该值。

创建一个 TCP/IP 对象，并将 **ConnectTimeout** 指定为 10 秒。

```
t = tcpclient("144.212.130.17",80,"ConnectTimeout",10)
```

```
t =
 tcpclient with properties:

 Address: '144.212.130.17'
 Port: 80
 NumBytesAvailable: 0

 Show all properties, functions
```

查看 **ConnectTimeout** 属性。

```
t.ConnectTimeout
```

```
ans =

 10
```

输出会反映 `ConnectTimeout` 属性更改。

---

**注意** 如果指定的地址或端口无效，或者无法建立与服务器的连接，则不会创建对象。

---

## 设置传输延迟属性

创建对象并使用名称-值对组参数来设置 `EnableTransferDelay` 值。`EnableTransferDelay` 参数指定 Nagle 算法是打开还是关闭。如果启用传输延迟，客户端将收集小段未完成的数据，并在收到来自服务器的确认 (ACK) 时以单个数据包的形式发送这些小数据段。如果禁用传输延迟，客户端会立即向网络发送数据。如果不指定 `EnableTransferDelay`，默认情况下会设置为 `true`。仅可在对象创建期间指定该值。

创建一个禁用了传输延迟的 TCP/IP 对象。

```
t = tcpclient("144.212.130.17",80,"EnableTransferDelay",false)
```

```
t =
 tcpclient with properties:

 Address: '144.212.130.17'
 Port: 80
 NumBytesAvailable: 0

 Show all properties, functions
```

查看 `EnableTransferDelay` 属性。

```
t.EnableTransferDelay
```

```
ans =

 logical

 0
```

输出会反映 `EnableTransferDelay` 属性更改。

## 查看 TCP/IP 对象属性

创建 `tcpclient` 对象后，您可以查看属性及其值的完整列表。在 `tcpclient` 输出中点击 `properties`。

```
t = tcpclient("www.mathworks.com",80)
```

```
t =
 tcpclient with properties:

 Address: 'www.mathworks.com'
 Port: 80
 NumBytesAvailable: 0

 Show all properties, functions

 Address: 'www.mathworks.com'
 Port: 80
```

```
NumBytesAvailable: 0

ConnectTimeout: Inf
 Timeout: 10
 ByteOrder: "little-endian"
 Terminator: "LF"

BytesAvailableFcnMode: "off"
BytesAvailableFcnCount: 64
 BytesAvailableFcn: []
 NumBytesWritten: 0

EnableTransferDelay: 1
 ErrorOccurredFcn: []
 UserData: []
```

有关如何配置这些属性的详细信息，请参阅“属性”。

您可以使用 `configureTerminator` 和 `configureCallback` 函数来配置某些属性。

## 另请参阅

`tcpclient`

## 详细信息

- “通过 TCP/IP 接口写入和读取数据”（第 14-6 页）

## 通过 TCP/IP 接口写入和读取数据

### 本节内容

“写入数据” (第 14-6 页)  
“读取数据” (第 14-6 页)  
“从气象站服务器获取数据” (第 14-7 页)  
“从网站读取网页” (第 14-7 页)

### 写入数据

`write` 函数以同步方式将数据写入与 `tcpclient` 对象连接的远程主机。首先指定数据，然后写入该数据。此函数一直等到指定数量的值写入远程主机。

在此示例中，`tcpclient` 对象 `t` 已存在。

```
% Create a variable called data
data = 1:10;

% Write the data to the object t
write(t, data)
```

**注意** 对于任何读写操作，该数据类型将转换为 `uint8` 以便传输数据。如果指定了其他数据类型，则可以将其重新转换为所设置的数据类型。

### 读取数据

`read` 函数以同步方式从连接到 `tcpclient` 对象的远程主机读取数据和返回数据。有三个读取选项：

- 读取所有可用字节（无参数）。
- （可选）指定要读取的字节数。
- （可选）指定数据类型。

如果未指定大小，则默认读取将使用 `BytesAvailable` 属性值，该值等于输入缓冲区中可用的字节数。

在这些示例中，`tcpclient` 对象 `t` 已存在。

```
% Read all bytes available.
read(t)

% Specify the number of bytes to read, 5 in this case.
read(t,5)

% Specify the number of bytes to read, 10, and the data type, double.
read(t,10,"double")
```

**注意** 对于任何读写操作，该数据类型将转换为 `uint8` 以便传输数据。如果指定了其他数据类型，则可以将其重新转换为所设置的数据类型。

## 从气象站服务器获取数据

TCP/IP 通信的主要用途之一是从服务器获取数据。此示例显示如何从远程气象站获取数据并对数据绘图。

**注意** 此示例中的 IP 地址不是工作 IP 地址。该示例显示如何连接到远程服务器。请使用您要与之通信的服务器的 IP 地址或主机名替换此处所示的地址。

- 1 使用此处显示的地址和端口 1045 创建 `tcpclient` 对象。

```
t = tcpclient("172.28.154.231",1045)
```

```
t =
tcpclient with properties:
```

```
Address: '172.28.154.231'
```

```
Port: 1045
```

```
NumBytesAvailable: 0
```

```
Show all properties, functions
```

- 2 使用 `read` 函数获取数据。对于来自三个传感器（温度、压力和湿度）的 10 个样本，指定要读取的字节数为 30。将数据类型指定为 `double`。

```
data = read(t,30,"double");
```

- 3 将 1×30 数据重构为 10×3 数据，各用一列来显示温度、压力和湿度。

```
data = reshape(data,[3,10]);
```

- 4 绘制温度图。

```
subplot(311);
plot(data(:,1));
```

- 5 绘制压力图。

```
subplot(312);
plot(data(:,2));
```

- 6 绘制湿度图。

```
subplot(313);
plot(data(:,3));
```

- 7 通过清除 TCP/IP 客户端对象关闭该对象与远程主机之间的连接。

```
clear t
```

## 从网站读取网页

在此示例中，您使用 TCP/IP 对象从 RFC Editor 网站读取一个网页。

- 1 创建一个 TCP/IP 对象。端口 80 是 Web 服务器的标准端口。

```
t = tcpclient("www.rfc-editor.org",80);
```

设置 TCP/IP 对象的 `Terminator` 属性。

```
configureTerminator(t,"LF","CR/LF");
```

- 2 您现在可以使用 `writeline` 和 `readline` 函数与服务器通信。

若要求 Web 服务器发送网页，请使用 GET 命令。您可以使用 'GET (path/filename)' 从 RFC Editor 网站获取文本文件。

```
writeline(t,"GET /rfc/rfc793.txt");
```

服务器接收命令并发回网页。您可以通过查看对象的 `NumBytesAvailable` 属性来查看是否发回了数据。

```
t.NumBytesAvailable
```

现在您可以开始读取网页数据。默认情况下，`readline` 一次读取一行。您可以读取多行数据，直到 `NumBytesAvailable` 值为 0。请注意，您看不到生成的网页；HTML 文件数据仅在屏幕上滚动显示。

```
while (t.NumBytesAvailable > 0)
 A = readline(t)
end
```

- 3 如果您要进行更多通信，可以继续读写数据。如果您不再使用该对象，请清除它。

```
clear t
```

### 另请参阅

`read` | `readline` | `write` | `writeline`



## Bluetooth 低功耗通信

---



## Bluetooth 通信

---



## 硬件管理器

---

