مقدمه‌ای کوتاه بر **سایلب**

۳۰ اردیبهشت ۹۲

شناسنامه

عنوان(اصلی): A Short Introduction to Scilab

نویسنده(ها): Terence Leung & N.K. Tsing

مترجم: ابوالفضل خورشیدی

تاریخ انتشار و نسخه سند: ۳۰ اردیبهشت ۹۲ ؛ نسخه ۱،۲

با سپاس فراوان از:

آقای سلمان محمدی که آموزش جامعی را برای نصب سایلب در توزیع‌های مختلف گنو/لینوکس نوشته و در ویکی فارسی سایلب منتشر کرده‌اند. آموزش ایشان عیناً در پیوست همین سند آورده شده است.

Acknowledgement

I would like to thank the authors, Dr. Nam-Kiu Tsing and Dr. Terence Leung for their work and effort in writing a valuable tutorial.

پیش‌گفتار

سایلب (Scilab) و شبیه‌ساز دینامیکی آن Xcos، جایگزین‌های آزاد و متن‌باز برای متلب (Matlab) و سیمولینک (Simulink) محسوب می‌شوند. از دیگر قابلیت‌های سایلب می‌توان به پردازش تصویر، پردازش سیگنال، بهینه‌سازی، تولید واسط گرافیکی کاربر، پردازش نمادین و محاسبات آماری اشاره کرد.

این آموزش یکی از پروژه‌های ویکی فارسی سایلب می‌باشد و شما با عضویت در ویکی به آسانی می‌توانید به ویرایش و اصلاح آن بپردازید. ما از تلاش مهربانانه‌ی شما در نسخه‌های بعدی این سند سپاسگزاری خواهیم کرد. قسمت‌هایی از آموزش به زبان انگلیسی با نسخه‌های جدید سایلب همخوانی نداشتند که در خلال ترجمه ضمن اشاره به آن‌ها تصحیح گردیدند. همچنین قسمت‌هایی که گمان می‌رفت برای کاربر تازه‌کار جای سوال باشد در پاورقی توضیح داده شده است. در ترجمه‌ی این آموزش از نسخه **۵٫۳٫۳** سایلب استفاده گردید. به دلیل مشابهت دستورات سایلب و متلب، این راهنما برای کاربران متلب نیز می‌تواند مفید باشد. ضمناً در خلال ترجمه به برخی از تفاوت‌های آنها اشاره شده است.

در پایان از زحمات نویسندگان این اثر آقایان دکتر چینگ و دکتر ترنس به‌خاطر آموزش بسیار ارزشمندشان سپاسگزاری می‌کنم. این آموزش با کسب اجازه از آقای دکتر چینگ به فارسی ترجمه و منتشر گردید.

دیدگاه‌ها، پیشنهادها و انتقادهای خود را می‌توانید از طریق ایمیل به نشانی akhorshidi@Live.com و یا Scilab.ir@Gmail.com با ما در میان بگذارید یا برای ارسال مستقیم دیدگاه خود به صفحه‌ی تماس با ما در وب‌سایت فارسی سایلب مراجعه کنید.

فهرست مطالب

۱. درباره‌ی سایلب

سایلب یک بسته نرم‌افزاری علمی آزاد و متن‌باز مشابه نرم‌افزار تجاری متلب می‌باشد که ابتدا توسط پژوهشگران دو موسسه‌ی فرانسوی INRIA[۱] و ENPC توسعه پیدا کرد و هم‌اکنون توسط Scilab Enterprises در حال توسعه می‌باشد[۲].

سایلب و متلب هر دو نرم‌افزارهایی قدرتمند هستند. سایلب از سه قسمت اصلی تشکیل شده است:

۱- یک مفسر

۲- کتابخانه‌های توابع (رویه‌های سایلب)

۳- کتابخانه‌های روال‌های C و Fortran

سایلب ویژه‌ی کار با ماتریس‌ها (عملیات اصلی ماتریس‌ها، الحاق، ترانهاده، معکوس و ...) و نیز محاسبات عددی ساخته شده است. همچنین سایلب یک محیط برنامه‌نویسی باز دارد که کاربران را قادر می‌سازد تا توابع و کتابخانه‌های خودشان را بسازند.

برای مطالعه بیشتر و مستندات به وب‌سایت رسمی سایلب مراجعه کنید:

http://www.scilab.org/

---

[۱] بنیاد ملی تحقیق در علوم کامپیوتر و کنترل فرانسه.

[۲] تا ژوئن ۲۰۱۲ میلادی کار توسعه و ارتقای نرم‌افزار توسط کنسرسیوم سایلب (Scilab Consortium) در داخل شبکه تحقیقاتی Digito انجام می‌شد. کنسرسیوم سایلب با نام جدید IPC (کمیته شرکای بین‌المللی) از جولای ۲۰۱۲ توسط Scilab Enterprises میزبانی می‌شود.

۲. نصب و اجرای سایلب

قبل از هرچیز، شما باید نسخه‌ای از نرم‌افزار را در اختیار داشته‌باشید. به قسمت دانلود وب‌سایت رسمی سایلب رفته و نسخه‌ای مناسب سیستم‌عامل‌تان (پلتفرم) پیدا کرده و دانلود کنید. برای نصب آسان، شما می‌توانید نصب‌کننده برنامه (برای نسخه‌های باینری) را دانلود کنید. سپس روی فایل دانلودشده دوکلیک کرده و مراحل پیش‌رو را طی نموده تا نصب برنامه کامل شود.

برای اجرای سایلب با برنامه خط فرمان در پوشه‌ی `bin` از دایرکتوری نصب، تایپ کنید:

```
scilex
```

یا اگر از ویندوز استفاده می‌کنید روی میانبری که در منوی شروع قرار دارد کلیک کنید.

برای خارج شدن از برنامه تایپ کنید (در خط فرمان سایلب):

```
exit
```

و یا پنجره‌ی برنامه اصلی را ببندید[۱].

---

[۱] واضح است که نویسنده این آموزش را بر اساس سیستم‌عامل ویندوز نوشته است.

در توزیع‌های مختلف گنو/ لینوکس (در اوبونتو امتحان شده‌است) بعد از دانلود پرونده‌ی دودویی (`scilab-5.4.0.bin.linux-i686.tar.gz`) آن را به دایرکتوری‌ای که می‌خواهید سایلب را در آن نصب کنید، انتقال دهید.

اکنون دستورات زیر را در ترمینال (برنامه اعلان فرمان) وارد کنید:

```
cd <scilab-path>
tar xzvf scilab-5.4.0.bin.linux-i686.tar.gz
```

برای اجرای سایلب در پوشه‌ی `bin` از دایرکتوری نصب تایپ کنید:

```
./scilab
```

راهنمای جامعی برای نصب سایلب در گنو/ لینوکس در **پیوست** آورده شده است.

۳. مستندات و راهنما

برای یافتن کاربرد هر تابع، تایپ کنید:

```
help function_name
```

برای مثال:

```
help sum
```

اگر می‌خواهید توابعی را که نمی‌دانید، پیدا کنید می‌توانید فقط تایپ کنید:

```
help
```

و سپس به جست‌وجوی کلمات‌کلیدی آن توابع بپردازید[۱]. سرانجام اگر به اطلاعات بیشتری نیاز داشتید می‌توانید وب‌سایت رسمی سایلب را بازدید نمایید. در آن‌جا قسمتی با عنوان مستندات (Documentation) قرار دارد که بسیار چاره‌ساز می‌باشد.

[۱] برای جستجوی کلمات کلیدی در راهنمای سایلب، می‌توانید در خط فرمان تایپ کنید:

```
apropos keyword
```

٤. اصول سایلب

٤-١. عملگرهای متداول

در اینجا فهرستی از عملگرهای متداول در سایلب آمده است:

+	جمع
-	تفریق
*	ضرب
/	تقسیم
^	توان
'	ترانهاده مزدوج [1]

٤-٢. توابع متداول

تعدادی از توابع متداول در سایلب عبارت‌اند از:

sin, cos, tan, asin, acos, atan, abs, min, max, sqrt, sum

برای مثال، وقتی وارد می‌کنیم:

```
sin(0.5)
```

در نتیجه نمایش داده می‌شود:

```
ans =
   0.4794255
```

مثال دیگر:

```
max(2, 3, abs(-5), sin(1))
ans =
   5
```

---

[1] ترانهاده مزدوج (ترانهاده‌ی مزدوج) یک ماتریس با نام های زیر در انگلیسی شناخته می شود:

conjugate transpose, Hermitian transpose, Hermitian conjugate, adjoint matrix

ماتریس الحاقی یک ماتریس (ترانهاده‌ی ماتریس همسازه‌ها) در انگلیسی adjugate matrix نامیده می‌شود که به آن classical adjoint matrix نیز می‌گویند.

لذا به تفاوت بین مفاهیم adjoint of a matrix و adjoint matrix توجه داشته‌باشید. اولی به ماتریس الحاقی اشاره دارد و دومی به ترانهاده مزدوج یک ماتریس اطلاق می‌شود.

٤-٣. ثابت‌های ویژه

شاید بخواهیم تا تعدادی از ثابت‌های ویژه مانند π و (sqrt(-1)) i و e را وارد کنیم. بدین‌منظور وارد می‌کنیم (به ترتیب)[1]:

```
%pi  %i  %e
```

همچنین ثابت‌های زیر نیز وجود دارند:

```
%t  %f
```

این‌ها ثابت‌های بولی بوده که به ترتیب درست (true) و نادرست (false) را نشان می‌دهند. متغیرهای بولی بعداً معرفی خواهند شد.

٤-٤. خط فرمان

دستورات چندگانه به‌وسیله‌ی ویرگول (کاما) از یکدیگر جدا شده و می‌توان آن‌ها را در یک خط وارد کنیم:

```
A = [1 2 3],  s = tan(%pi/4) + %e
 A  =
    1.   2.   3.
 s  =
    3.7182818
```

قرار دادن نقطه‌ویرگول (سیمی‌کولن) در انتهای فرمان از نمایش نتیجه (پاسخ عبارت) جلوگیری می‌کند:

```
A = [1 2 3];  s = tan(%pi/4) + %e;
```

در این‌جا بردار [1 2 3] در متغیر A ذخیره شده و عبارت tan(%pi/4) + %e محاسبه شده و (مقدار آن) در متغیر s ذخیره می‌شود؛ اما نتایج بروی صفحه نمایش داده نمی‌شود.

دستورات طولانی می‌توانند در چند خط شکسته شوند و استفاده از سه‌نقطه (...) در پایان هر خط نشان می‌دهد که دستور درحقیقت در خط بعدی ادامه می‌یابد:

```
s =  1 -1/2 + 1/3 -1/4 + 1/5 - 1/6 + 1/7 ...
     - 1/8 + 1/9 - 1/10 + 1/11 - 1/12;
```

---

[1] برای تعریف کلیه ثابت‌های ویژه از جمله عدد پی π و رادیکال منفی یک i ، در متلب به ترتیب داریم:

```
pi  i
```

در سایلب آنها را به صورتی که در بالا گفته شد، تعریف می‌کنیم(یک علامت درصد% قبلشان).

این ثابت‌ها محافظت شده بوده و شما نمی‌توانید مقدار آنها را تغییر دهید. یکی از مزایایی چنین تعریفی آن است که شما می‌توانید از متغیرهای i و e و pi به صورت دلخواه در کد نویسی خود بهره ببرید. برای مثال مقدار pi=-3.14 .

حال اگر در متلب pi=-3.14 تعریف کنید متلب تا انتهای کدنویسی این مقدار جدید را برای عدد پی می‌شناسد.

برای مشاهده‌ی فهرستی از این ثابت‌ها در خط فرمان سایلب تایپ کنید:

```
whos  -name %
```

هر چیزی‌که بعد از یک جفت علامت اسلش // تایپ شود، در نظر گرفته نمی‌شود (به عنوان دستور)، بنابرین می‌توانیم برای بیان دیدگاه‌ها و یا توضیحات مربوط به کدنویسی آن‌را به کار ببریم[۱].

```
%e^(%pi * %i) + 1  // should equal 0, as in the Euler's
identity
ans  =
   1.225D-16i
```

استفاده از پیکان‌های بالا (up) و پایین (down)[۲] در خط فرمان، دستورات قبلی را فراخوانی می‌کند.

٤-٥. ساختمان داده‌ها

سایلب بسیاری از ساختمان داده‌ها را پشتیبانی می‌کند. برای مثال:

(real or complex) matrix, polynomial, Boolean, string, function, list, tlist, sparse, library.

برای جزئیات بیشتر مستندات سایلب را مطالعه کنید.

برای پرس‌وجو از نوع یک شئ ، تایپ کنید:

```
typeof(object)
```

٤-٦. رشته‌ها

برای واردکردن یک رشته آن را بین علامت‌های یگانه (') یا دوگانه (") نقل‌قول قرار دهید. برای مثال:

```
'This is a string'
```

یا

```
"this is also a string"
```

برای اتصال (الحاق) رشته‌ها از عملگر «+» استفاده کنید:

```
"Welcome " + "to " + "Scilab!"
ans =
   Welcome to Scilab!
```

تعدادی تابع اصلی برای مدیریت رشته‌ها وجود دارند، ازجمله:

Strindex, strsplit, strsubst, part

برای دانستن جزئیات لطفا به مستندات سایلب رجوع کنید.

---

[۱] در متلب برای این منظور از یک علامت درصد % استفاده می‌کنیم.

[۲] منظور کلیدهای جهت‌نمای صفحه‌کلید

٤-٧. ذخیره و بارگذاری متغیرها

برای ذخیره و بارگذاری متغیرها ما از توابع save و load بهره می‌بریم:

```
save('file_name', var1, var2, ...);
load('file_name', 'var1', 'var2', ...);
```

جایی‌که file_name نام فایلی است که ذخیره یا بارگذاری می‌شود و ... ,var1, var2 نام متغیرها می‌باشند.

توجه داشته باشید که متغیر باید با نام خودش ذخیره شود.

در ادامه چند مورد شرح داده شده است.

```
a = 3; b = %f; s = 'scilab';
save('save.dat', a, b, s);
clear a;  // delete the variable a
clear b;
clear s;
load('save.dat', 'a', 'b', 's');
// load all the saved variables
load('save.dat','b');
// It loads only variable b, but not
// variable a in the name of b
load('save.dat','d');
// It will not show any error messages[1].
// Variable d is undefined, not empty.
listvarinfile('save.dat');
// list variables in a file saved by
// the function save
Name              Type         Size          Bytes
-------------------------------------------------
a                 constant     1 by 1        24
b                 boolean      1 by 1        20
s                 string       1 by 1        44
```

[1] چون متغیر d در فایل مفروض وجود ندارد پس پیغام خطایی نمایش داده می‌شود درحالی که نویسنده گفته است که پیغام خطایی نمایش داده نخواهد شد!

```
!--error 999
load: variable 'd' does not exist in 'save.dat'.
at line     823 of function %_sodload called by :
load('save.dat','d');
```

۵. کار با ماتریس‌ها

۵-۱. وارد کردن ماتریس‌ها

روش‌های متعددی برای وارد کردن ماتریس‌ها وجود دارد. ساده‌ترین روش این است که:

۱) هر عنصر در یک سطر را با یک جای خالی یا یک ویرگول (کاما) جدا می‌کنیم،

۲) هر سطر از عناصر را با یک نقطه ویرگول (سِمی‌کولن) جدا کرده، و

۳) تمامی فهرست عناصر را در داخل یک جفت کروشه قرار می‌دهیم.

برای مثال، برای وارد کردن مربع جادویی و اختصاص آن به متغیر M داریم:

```
M = [8 1 6; 3 5 7; 4 9 2]
M =
   8.   1.   6.
   3.   5.   7.
   4.   9.   2.
```

۵-۲. محاسبه‌ی مجموع

ما یک مربع جادویی داریم، شاید بخواهیم مجموع عناصر روی هر سطر، هر ستون و هر قطر آن را بررسی کنیم. برای این کار وارد می‌کنیم:

```
sum(M,'c') // column sums
ans =
   15.
   15.
   15.
sum(M,'r') // row sums
ans =
   15.  15.  15.
```

مجموع عناصر روی قطر اصلی ماتریس با کمک تابع diag براحتی بدست می‌آید[۱].

```
diag(M)
ans =
   8.
   5.
   2.
```

[۱] برای این منظور دستور زیر را وارد کنید:

```
sum(diag(M))
```

۵-۳. زیروندها

پیدا کردن مجموع عناصر روی قطرهای دیگر یه‌کمی مشکل است. ما دو روش را برای انجام آن نشان می‌دهیم. روش اول پیدا کردن مجموع به صورت دستی می باشد. یعنی، خواندن عناصر معین و سپس جمع کردن آن‌ها.

```
M(1,3) + M(2,2) + M(3,1)
ans =
   15.
```

می‌توانیم با استفاده از یک اندیس یگانه به عناصر یک ماتریس دسترسی پیدا کنیم. در این حالت یک ماتریس را به عنوان برداری طویل حاصل از انباشته شدن ستون‌های ماتریس بروی یکدیگر مورد بررسی قرار می‌دهیم. برای مثال، مقادیر M(1) و M(2) و M(3) و M(4) و M(5) به ترتیب 8 و 3 و 4 و 1 و 5 هستند.

در دسترسی به عناصر خارج از ماتریس با یک پیغام خطا روبرو می‌شوید. مثلا:

```
M(3,4)
    !--error 21
invalid index
```

یک روش زیرکانه برای بدست آوردن مجموع عناصر روی قطرهای دیگر استفاده از تابع mtlb_fliplr می‌باشد. جایی‌که mtlb به Matlab اشاره دارد. با این دستور یک ماتریس را از چپ به راست وارو (flip) می‌کنیم.

```
mtlb_fliplr(M)
ans =
   6.   1.   8.
   7.   5.   3.
   2.   9.   4.
```

سپس نتیجه‌ی مطلوب با وارد کردن عبارت زیر حاصل می‌شود.

```
sum(diag(mtlb_fliplr(M)))
```

۵-٤. عملگر دونقطه (کولن)

عملگر دونقطه (کولن) یکی از مهمترین عملگرها در سایلب می‌باشد. عبارت 1:10 یک ماتریس سطری با عناصر ۱, ۲, ... , ۱۰، را نتیجه می‌دهد. یعنی:

```
1:10
ans =
   1.  2.  3.  4.  5.  6.  7.  8.  9.  10.
```

برای فاصله‌بندی ناواحد، مقدار افزایش (نمو) را مشخص می‌کنیم:

```
10 : -2 : 2
ans =
   10.  8.  6.  4.  2.
```

متذکر می‌شویم که عبارت‌هایی چون 10:-2:1 و 10:-2:0.3 نتیجه‌ی یکسانی را تولید می‌کنند اما عبارت 11:-2:2 چنین نتیجه‌ای نخواهد داشت.

عبارت‌های زیروندار همراه با دونقطه (کولن) به قسمتی از یک ماتریس اشاره می‌کنند. M(i:j, k) سطر i–ام تا سطر j–ام از ستونk–ام را نشان می‌دهد. به طور مشابه،

```
M(3,2:3)
ans =
   9.  2.
```

مثال‌های بیشتر:

```
M(3,[3,2])
ans =
   2.    9.
M([2,1], 3:-1:1)
ans =
   7.   5.   3.
   6.   1.   8.
```

عملگر $ به بزرگترین مقدار یک اندیس تخصیص داده می‌شود، و برای رسیدن به آخرین درایه‌ی یک بردار یا ماتریس سودمند خواهد بود. برای دسترسی به تمام عناصرِ ستون آخر ماتریس M به جز آخرین عنصر ، داریم:

```
M(1:$-1, $)
ans =
   7.
   6.
```

برخی اوقات همه‌ی یک سطر یا یک ستون را می‌خواهیم. برای مثال، برای بدست آوردن تمام عناصر روی سطر دوم ماتریس M داریم:

```
M(2,:)
ans =
   3.   5.   7.
```

هم‌اکنون روش جدیدی برای اجرای عملیاتی چون mtlb_fliplr(M) داریم. برای این کار M(:, $:-1:1) را وارد می‌کنیم. به هر حال تابع mtlb_fliplr(M) نتیجه‌ی سریعتری (از نظر زمان محاسبه) را نسبت به هنگامی‌که از زیروندها استفاده می‌کنیم، می‌دهد.

۵-۵. ایجاد آسان ماتریس‌ها

تعدادی از ماتریس‌های اصلی تنها با یک دستور ایجاد می‌شوند:

zeros	تمام درایه‌ها صفر
ones	تمام درایه‌ها یک
eye	ماتریس همانی (درایه‌های واقع روی قطر اصلی همگی یک و بقیه درایه‌ها همگی صفر)
rand	عناصر تصادفی (یا از توزیع نرمال یا از توزیع یکنواخت)

چند مثال:

```
zeros(2,3)
ans =
   0.   0.   0.
   0.   0.   0.
8 * ones(2,2)
ans =
   8.   8.
   8.   8.
eye(2,3)
ans =
   1.   0.   0.
   0.   1.   0.
rand(1,3,'uniform') // same as rand(1,3)
ans =
   0.2113249   0.7560439   0.0002211
```

۶-۵. الحاق (اتصال)

الحاق (Concatenation) فرایندی است که ماتریس‌ها با اندازه کوچکتر را به هم متصل می‌کند و یک ماتریس با اندازه بزرگتر می‌سازد. برای این منظور ماتریس‌های کوچکتر را به عنوان عناصر ماتریس بزرگتر قرار می‌دهیم:

```
a = [1 2 3]; b = [4 5 6]; c = [7 8 9];
d = [a b c]
d =
   1.  2.  3.  4.  5.  6.  7.  8.  9.
e = [a; b; c]
e =
   1.   2.   3.
   4.   5.   6.
   7.   8.   9.
```

فرایند الحاق باید سطر یا ستون سازگار باشد:

```
x = [1 2]; y = [1 2 3];
z = [x; y]
       !-error 6
inconsistent row/column dimensions
```

ماتریس‌های جعبه‌ای را نیز می‌توان بهم متصل کرد:

```
[eye(2,2) 5*ones(2,3); zeros(1,3) rand(1,2)]
ans =
   1.   0.   5.   5.          5.
   0.   1.   5.   5.          5.
   0.   0.   0.   0.6525135   0.3076091
```

۷-۵. حذف سطرها و ستون‌ها

یک جفت کروشه که بین آن چیزی نباشد یک **ماتریس تهی** را نمایش می‌دهد؛ که از آن می‌توان برای حذف سطرها یا ستون‌های یک ماتریس استفاده کرد. برای حذف سطر اوّل و سوم یک ماتریس همانی از مرتبه ٤، درایم:

```
A = eye(4,4);
A([1 3],:) = []
A =
   0.   1.   0.   0.
   0.   0.   0.   1.
```

اگر با این روش بخواهیم تنها یک عنصر از ماتریس را حذف کنیم با پیغام خطا روبرو می‌شویم[۱].

```
A(1,2) = []
         !-error 15
submatrix incorrectly defined
```

اگر عناصر را با استفاده از عبارت اندیس یگانه حذف کنیم، در نتیجه یک بردار ستونی خواهیم داشت:

```
B=[1 2 3; 4 5 6];
B(1:2:5)=[]
B =
   4.
   5.
   6.
```

[۱] اگر بخواهیم تنها یک عنصر ماتریس را حذف کنیم باید از اندیس یگانه بهره ببریم. برای درک بهتر، خروجی دستور A(2)=[ ] را بررسی کنید.

۸-۵. معکوس ماتریس و حل سیستم‌های خطی

دستور inv(M) معکوس ماتریس M را نتیجه می‌دهد. اگر ماتریس **بد مقیاس** (badly scaled)[۱] باشد و یا ماتریسی تقریباً **تکین**(singular matrix)[۲] باشد، پیغام هشداری نمایش داده می‌شود:

```
inv([1 2;2 4.0000001])
 warning
 matrix is close to singular or badly scaled. rcond =
2.7778D-09
 ans  =
    40000001.   - 20000000.
  - 20000000.    10000000.
```

منظور از حل دستگاه معادلات خطی، Ax=b، یعنی یافتن مقادیر x که در معادله صدق کند. A یک ماتریس مربعی معکوس پذیر بوده و b یک بردار می‌باشد. در سایلب برای این منظور وارد می‌کنیم A\b :

```
A = rand(3,3), b = rand(3,1)
A =
   0.2113249   0.3303271   0.8497452
   0.7560439   0.6653811   0.6857310
   0.0002211   0.6283918   0.8782165
b =
   0.0683740
   0.5608486
   0.6623569
x = A \ b
x =
   - 0.3561912
     1.7908789
   - 0.5271342
```

روش دیگر inv(A)*b می‌باشد. این روش نتیجه یکسانی دارد اما کندتر خواهد بود؛ زیرا دستور A\b اساساً از روش حذفی گاوس که موجب صرفه‌جویی در محاسبات بوده، استفاده می‌کند. لطفا راهنمای نرم‌افزار را برای جزئیات بیشتر درباره‌ی عملگر اسلش (/) و چگونگی پیدا کردن معکوس ماتریس A وقتی مربعی نباشد، مطالعه نمایید.

A/b معادله‌ی xb=A را برحسب x حل می‌کند.

---

[۱] یک ماتریس را بد مقیاس (دارای مقیاس‌بندی ناجور) می‌نامند اگر بعضی از درایه‌های آن خیلی بزرگتر از بقیه باشند. برای نمونه:

$$\begin{bmatrix} 2*10^{9} & 10^{9} \\ 10^{-9} & 2*10^{-9} \end{bmatrix}$$ و $$\begin{bmatrix} 10^{100} & 0 \\ 0 & 10^{-100} \end{bmatrix}$$.

[۲] ماتریسی که معکوس نداشته باشد (دترمینان آن برابر صفر باشد) را ماتریس تکین (منفرد) می‌نامند.

۹-۵. عملیات درایه‌به‌درایه[۱]، اندازه ماتریس

برای اضافه‌کردن ٤ به تمام درایه‌های ماتریس M می‌توانیم از دستور M + 4 * ones(M) بهره ببریم که البته پرزحمت است. در واقع این کار با استفاده از دستور M+4 به راحتی انجام می‌شود. برای تفریق داریه‌به‌درایه یک ماتریس از اسکالر نیز به‌همین ترتیب عمل می‌کنیم.

برای آنکه عدد ۲ را در ستون دوم و عدد ۳ را در ستون سوم ماتریس M ضرب کنیم از عملگر داریه‌به‌درایه *. بهره می‌بریم:

```
M .* [1:3; 1:3; 1:3]
```

عملیات ریاضی درایه‌به‌درایه برای آرایه‌ها عبارت‌اند از:

+	جمع
-	تفریق
.*	ضرب
.^	توان
./	تقسیم راست
.\	تقسیم چپ

بنابرین به‌جای نوشتن

```
s =  1 -1/2 + 1/3 -1/4 + 1/5 - 1/6 + 1/7 ...
     - 1/8 + 1/9 - 1/10 + 1/11 - 1/12;
```

کافی است بنویسیم:

```
s = sum((1:2:12) .\ 1) - sum((2:2:12) .\ 1)
```

یا

```
s = sum(1 ./ (1:2:12)) - sum(1 ./ (2:2:12))
```

توجه داشته‌باشید که 1./1:2:12 به صورت (1./1):2:12 تفسیر می‌شود. بطور مشابه 1\1:2:12. به صورت 1\(1.):2:12 تفسیر می‌شود.

برای وارد کردن ماتریس M = [1 2 3 4 5; 6 7 8 9 10] می‌توان داشت:

```
M = zeros(2,5);
M(:) = 1:10;
```

[۱] به انگلیسی: entry-wise operations یا element-wise operations یا component-wise operations.

با این‌حال یک روش تقریبا بی‌دردسر استفاده از تابع matrix می‌باشد که یک ماتریس را از نو در اندازه‌ای مطلوب می‌ریزد.

```
M = matrix(1:10,5,2)'
```

چگونه ماتریس N = [1 2; 3 4; 5 6; 7 8; 9 10] را به‌آسانی وارد کنیم؟ راهنمایی: درباره‌ی چند عملیات ساده روی ماتریس‌ها فکر کنید[۱]!

یک تابع آماده در سایلب، تابع size می‌باشد که ابعاد ماتریس موردِ پرسش را بر می‌گرداند.

```
size(M)
ans =
   2.    5.
```

در عین‌حال size(M,1) مقدار ۲ (تعداد سطرها) را گزارش می‌دهد و size(M,2) مقدار ۵ (تعداد ستون‌ها) را گزارش می‌دهد.

[۱] یک جواب می‌توانید به صورت زیر باشد:

```
N = zeros(5,2);
N(1:5)=1:2:10; N(6:10)=2:2:10
```

٦. محیط برنامه‌نویسی

٦-١. ایجاد توابع

سایلب یک محیط برنامه‌نویسی باز دارد که کاربران را قادر می‌سازد تا توابع و کتابخانه‌های خودشان را بسازند. این کار با کمک ویرایشگر درونی سایلب به نام SciNotes انجام می‌شود. برای فراخوانی ویرایشگر تایپ کنید scinotes() یا editor() و یا بر روی ویرایشگر در نوار منو کلیک کنید[۱].

پسوند فایل‌های سایلب sci. و sec. می‌باشند. برای ذخیره یک فایل بروی منوی File کلیک کرده و گزینه‌ی Save را انتخاب کنید. برای بارگذاری یک فایل در منوی مفروض بروی Load کلیک کنید. برای اجرای یک فایل در خط‌فرمان تایپ کنید[۲]:

```
exec('function_file_name');
```

یا از همان منوی فایل گزینه ...Execute را انتخاب کنید[۳].

برای آغاز نوشتن یک تابع، تایپ کنید:

```
function [out1, out2, ...] = name(in1, in2, ...)
```

جایی‌که کلیدواژه function شروع یک تابع را نمایش می‌دهد؛ ...,out1,out2 و ...,in1,in2 متغیرهایی هستند که به‌ترتیب نشان‌دهنده‌ی خروجی‌ها و ورودی‌های یک تابع می‌باشند؛ این متغیرها می‌توانند بولی، اعداد، ماتریس‌ها و ... باشند. name، نام تابع موردنظر می‌باشد. در پایان برای نشان‌دادن پایان یک تابع تایپ می‌کنیم:

```
endfunction
```

خطوط توضیحی با دو علامت اسلش // شروع می‌شوند. در زیر یک تابع ساده داده شده است:

```
function [d] = distance(x, y)
// this function computes the distance
// between the origin and the point (x, y)
d = sqrt(x^2 + y^2);
endfunction
```

---

[۱] در نسخه اصلی آموزش نام ویرایشگر سایلب، SciPad ذکر شده است و دستور فراخوانی آن نیز scipad() آمده‌است.

[۲] به خاطر داشته باشید که برای اجرای هر فایل ابتدا باید به دایرکتوری که فایل مورد نظر در آن قرار دارد بروید. برای این منظور می‌توانید از پنجره‌ی File Browser (از نسخه ۵٫٤٫۰ به بعد) استفاده کنید یا دستور زیر را در خط فرمان تایپ کنید:

```
cd "Full Path";
```

توجه داشته باشید که دستور .. cd در سایلب کار نمی‌کند (امّا در متلب کار می‌کند)؛ به جای آن می‌توانید از دستور ../ cd استفاده کنید.

[۳] در نسخه اصلی آموزش آمده است که از منوی Execute گزینه‌ی load into Scilab را انتخاب کنید.

یک تابع سایلب می‌تواند به‌طور بازگشتی خودش را فرابخواند. در ادامه مثالی آورده شده است.

بر خلاف متلب، سایلب این اجازه را می‌دهد که چندین اعلان تابع (توابع با نام‌های مختلف) را در یک فایل واحد داشته باشیم. سایلب اجازه سربارگذاری (overloading) را نیز می‌دهد (برای کاربران تازه‌کار توصیه نمی‌شود). برای جزئیات بیشتر فصل سربارگذاری را در راهنمای سایلب بخوانید.

۲-٦.روند کنترل

در جدول زیر عبارت‌های منطقی (logical expressions) آورده شده‌اند:

==	مساوی با
~=	نامساوی با
>=	بزرگتر از یا مساوی با
<=	کوچکتر از یا مساوی با
>	بزرگتر از
<	کوچکتر از
~	NOT

اگر یک عبارت منطقی درست باشد، نتیجه متغیر بولی T (درست) خواهد بود در غیر این صورت F (نادرست) به عنوان خروجی نمایش داده می‌شود.

دستور شرطی if

ساختار اصلی این دستور این‌چنین است:

```
if condition
   body
end
```

Body (بدنه دستور) فقط هنگامی که condition (عبارت شرطی) برقرار باشد اجرا خواهد شد. دستورات if تودرتو ساختار زیر را دارند:

```
if condition_1
   body_1
elseif condition_2
   body_2
elseif condition_3
   body_3
elseif ...
   ...
end
```

به عنوان یک مثال، برای نوشتن یک تابع که n–امین عدد فیبوناتچی را در دنباله ...،۰،۱،۱،۲،۳،۵،۸،۱۳ نتیجه بدهد، خود را ملزوم می‌کنیم به استفاده از این واقعیت که یک تابع سایلب می‌تواند به‌طور بازگشتی خودش را فرابخواند.

```
function [K] = fibonacci(n)
//function [K] = fibonacci(n)
//Gives the n-th term of the Fibonacci sequence
0,1,1,2,3,5,8,13,...
if n==1
  K = 0;
elseif n==2
  K = 1;
elseif n>2 & int(n)==n  // check if n is an integer greater
than 2
  K = fibonacci(n-1) + fibonacci(n-2);
else
  disp('error! -- input is not a positive integer');
end
endfunction
```

توجه داشته باشید که وقتی n بزرگ باشد این روشی کارآمد برای پیدا کردن n–امین عدد فیبوناتچی نیست.

حلقه for

ساختار اصلی این دستور این‌چنین است:

```
for variable = initial_value : step : final_value
   body
end
```

این حلقه به تعداد دفعات معینی که توسط تعداد عناصر در متغیر آرایه مشخص می‌شوند، اجرا خواهد شد. در ادامه نسخه‌ای از آن با اندکی تغییر آورده شده است:

```
str = 'abcdr';
s = ''; // an empty string
for i = [1 2 5 1 3 1 4 1 2 5 1]
   s = s + part(str, i);
end
disp(s);
s = abracadabra
```

حلقه while

ساختار اصلی این دستور این‌چنین است:

```
while condition
   body
end
```

این حلقه تا مادامی‌که عبارت شرطی درست باشد ادامه خواهد یافت. در اینجا مثالی از الگوریتم اقلیدسی را می‌آوریم.

```
function [n1] = hcf(n1, n2)
// n1 and n2 are positive integers
if n2 > n1
   tem = n2; n2 = n1; n1 = tem; // to ensure n1>=n2
end
r = pmodulo(n1, n2); // remainder when n2 divides n1
n1 = n2;  n2 = r;
while r ~= 0
   r = pmodulo(n1, n2);
   n1 = n2;  n2 = r;
end
endfunction
```

دستورات **break** و **continue**

این دستورات به ترتیب برای پایان دادن به یک حلقه و برای شروع فوری تکرار بعدی می‌باشند. برای مثال:

```
// user has to input 10 numbers and for those which
// are integers are summed up, the program are
// prematurely once a negative number is entered.
// It is not well written but just to illustrate the
// use of the "break" and "continue" commands
result = 0;
for i = 1:10
   tem = input('please input a number');
   if tem < 0
      break;
   end
   if tem ~= int(tem) //integral part
      continue;
   end
   result = result + tem;
end
disp(result);
```

۳-۶. چند نکته‌ی برنامه‌نویسی

مفهوم بردارها و ماتریس‌های بولی بسیار مهم است. همچنین تابع find مفید خواهد بود. این تابع اندیس عناصر بردارها و ماتریس‌های بولی درست را گزارش می‌دهد. برای مثال:

```
M = [-1 2; 4 9]; M > 0
ans =
   F  T
   T  T
M(M > 0)'
ans =
   4.   2.   9.
end
```

در مقایسه با

```
find(M > 0)
ans =
   2.   3.   4.

M(find(M>0))'
ans =
   4.   2.   9.
```

باید متذکر شویم که M(M>0) نتایج سریع‌تری را نسبت به M(find(M>0)) می‌دهد. زیرا در اینجا تابع find غیرضروری می‌باشد.

بسیار مهم است که & و and ، و نیز | و or ، را از یکدیگر تمیز دهیم. در هر جفت، اوّلی عمل داریه‌به‌درایه بوده و دیگری بر اساس تمام داریه‌های یک ماتریس بولی مقدار درستی[۱] را گزارش می‌دهد.

```
M = [0 -2; 1 0]; M==0 | M==1
ans =
   T  F
   T  T
and(M >= 0) // true iff[۲] all entries are true
ans =
   F
or(M == -2) // false iff all entries are false
ans =
   T
```

[۱] درست(T) اگر و تنها اگر همه‌ی داریه‌ها درست(T) باشند و نادرست(F) اگر و تنها اگر همه‌ی درایه‌ها نادرست(F) باشند.

[۲] کوتاه شده‌ی عبارت if and only if.

٤-٦. خطایابی

یکی از کسل‌کننده‌ترین کارها در برنامه‌نویسی اشکال‌زدایی می‌باشد. که به دو روش قابل انجام است: یا استفاده از خطایاب درونی سایلب، یا اصلاح و تغییر برنامه به گونه‌ای که همان هدف خطایاب را برای ما انجام دهد.

خطایاب سایلب مشابه خطایاب‌های سایر زبان‌های برنامه نویسی بوده و کار با آن آسان می‌باشد. ما روش دوم را ارائه می‌کنیم تا برنامه نویس ازانعطاف‌پذیری بیشتری در حین اشکال‌زدایی برخوردار باشد.

برای ایجاد توقف‌گاه (وارد کردن نقطه انفصال) از دستور pause استفاده می‌کنیم[۱]. برای خاتمه دادن به آن از دستور abort استفاده می‌کنیم. برای تنظیم خروجی تابع می‌توانیم از دستور return استفاده کنیم[۲].

برای نمایش متغیرها داریم:

```
disp (variable)
```

برای جزئیات بیشتر مستندات سایلب را مطالعه کنید.

[۱] با اجرای دستور pause به محیط جدیدی منتقل شده و اعلان جدیدی <-1- بدست می‌آوریم که نشان دهنده‌ی سطح محیط جدید می‌باشد (سطح ۱). تمامی متغیرهایی که در محیط اول در دسترس هستند در این محیط جدید نیز قابل دسترسی می‌باشند. متغیرهایی که در محیط جدید ایجاد می‌شوند با کمک دستور return به محیط اصلی بازگردانده می‌شوند. کاربرد return بدون هیچ آرگومانی، تمام متغیرهایی که در محیط جدید ایجاد شده‌اند را قبل از بازگشت به محیط قدیمی از بین می‌برد.

```
-->s=poly(0,'s'); p=1+2*s+s^2;
-->pause
-1->pt=return(s*p)
-->pt
pt  =
          2   3
    s + 2s + s
```

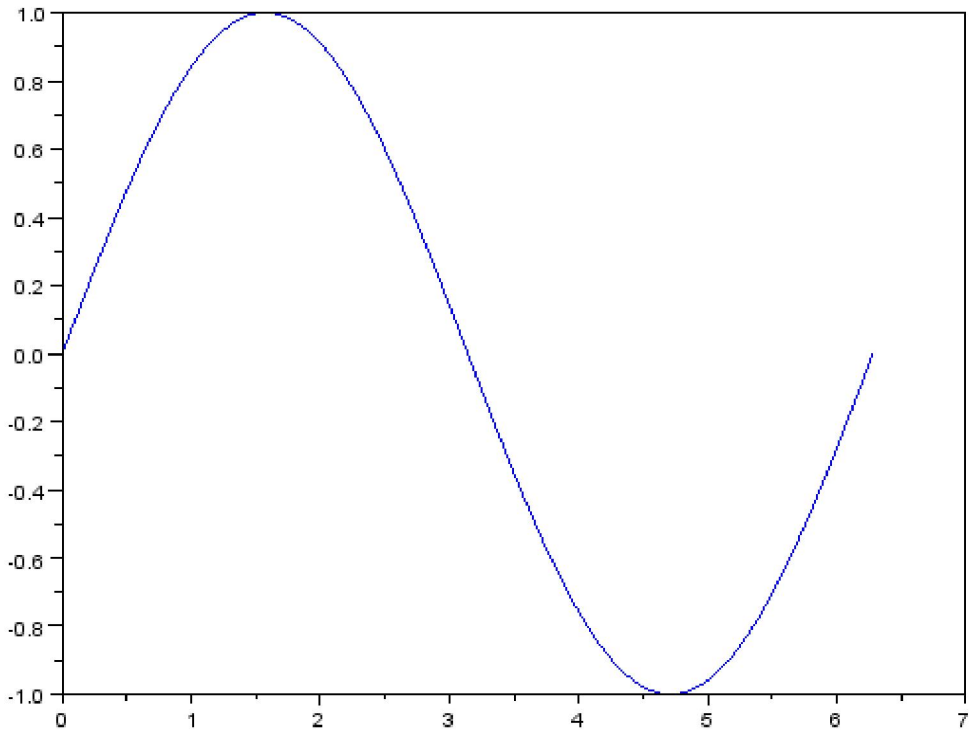
[۲] استفاده از عبارت return در داخل بدنه‌ی یک تابع، فوراً تابع را متوقف کرده یعنی بلافاصله تابع جاری را از دستور کار خارج می‌کند. این عبارت می‌تواند در مواردی که اجرای باقیمانده‌ی الگوریتم ضروری نباشد، مورد استفاده قرار گیرد.

۷. رسم نمودارها

۷-۱. نمودارهای دوبعدی

تابع plot بسته به نوع آرگومان‌های (متغیرهای) ورودی انواع مختلفی دارد. اگر y یک بردار باشد، plot(y) یک نمودار خطی تکه‌به‌تکه از عناصر بردار y برحسب زیروند هر عنصر y تولید می‌کند. وقتی x و y بردارهایی با اندازه یکسان باشند plot(x,y) نمودار y برحسب x را تولید می‌کند. برای مثال، برای رسم مقادیر تابع سینوس از صفر تا 2π داریم:

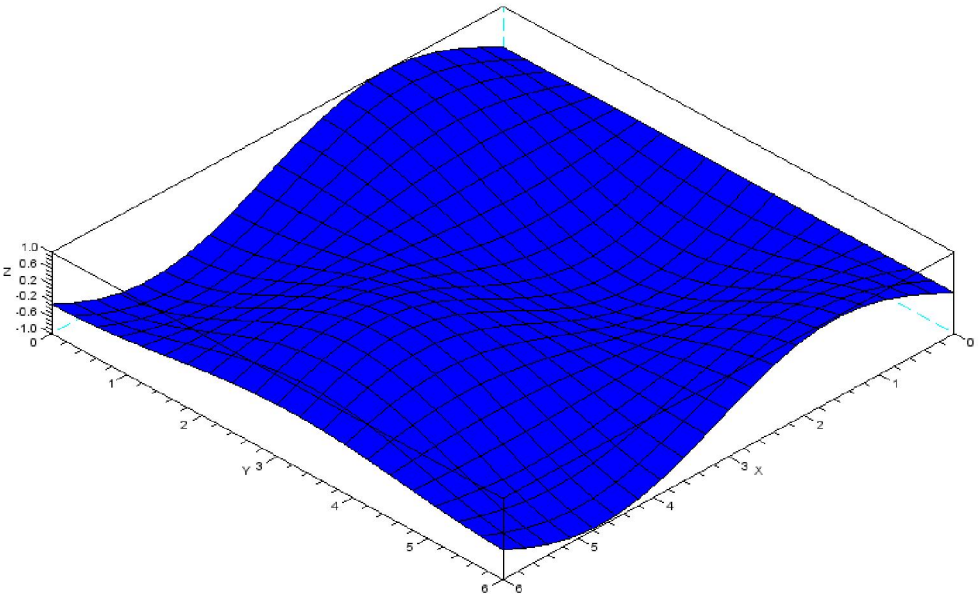
```
t = (0:1/100:2) * %pi;
y = sin(t);
plot(t,y);
```



۲-۷. رویه‌های سه بعدی

دستور plot3d(x,y,z) رویه‌های سه‌بعدی را رسم می‌کند. در این‌جا x و y (مختصات محورهای x و y) بردارهای سطری با اندازه‌های n1 و n2 بوده و این مختصات باید یکنوا (monotone)[1] باشند. z ماتریسی با اندازه n1×n2 بوده و z(i,j) نشان دهنده‌ی مقدار (ارتفاع) رویه در نقطه‌ی(x(i),y(j)) می‌باشد.

```
// A simple plot of z = f(x,y)
t=[0:0.3:2*%pi]';
z=sin(t)*cos(t');
plot3d(t,t,z)
```



[1] یک تابع یا یک دنباله که صعودی یا نزولی باشد.

البته می‌توان با دستور شرطی if مقادیر آن را کنترل کرد؛ دستورا ت زیر را در سایلب اجرا کنید:

```
t=[0:-0.3:-2*%pi]';
if or(diff(t)<=0),error('t not ascending order'), end
z=sin(t)*cos(t');
plot3d(t,t,z)
```

۸. سایلب، اُکتاو یا متلب[۱]

هر سه در دسته نرم‌افزارهای محاسبات عددی طبقه‌بندی می‌شوند. البته اصطلاح « محاسبات عددی » یک اصطلاح کلی می‌باشد. در واقع بهتر است هر سه را نرم‌افزارهای عددی همه منظوره (Full Purpose Numerical Software) بنامیم.
متلب یک نرم‌افزار تجاری بوده و بسیار گران‌قیمت(!!) می‌باشد. شرکت مث‌ورکس (Mathworks)، سازنده نرم‌افزار متلب، یک شرکت آمریکایی بوده و تابع قوانین، مقررات و سیاست‌های کشور آمریکا می‌باشد. پس قاعدتاً نباید (یا نمی‌تواند) به کاربران ایرانی سرویس دهد. به همین دلیل این شرکت سایت خود را برای IP ایران مسدود کرده است و جامعه علمی ما از منابع بی‌نظیر این سایت محروم می‌باشد. در طرف مقابل سایلب را داریم که روزبه‌روز پشتیبانی و سرویس‌های آنلاین خود را برای تمامی کاربرانش در سراسر دنیا توسعه داده و هیچ محدودیتی را برای هیچ کشوری قرار نداده است.
در ادامه به مقایسه‌ی اجمالی جایگزین‌های آزاد و متن‌باز برای متلب یعنی سایلب و گنو اُکتاو (GNU Octave) می‌پردازیم.
سایلب امکان مدل‌سازی و شبیه‌سازی سیستم‌های پیچیده را با کمک ابزار Xcos فراهم آورده است. و تنها جایگزین آزاد و متن‌باز برای سیمولینک متلب می‌باشد.
گنو اُکتاو بر سازگاری و مطابقت هرچه بیشتر با متلب تمرکز دارد در حالی که سایلب می‌گوید: "متلب منبعی برای الهام گرفتن است البته تا زمانی که آنها چیزهای خوبی انجام می‌دهند".
گنو اُکتاو دارای اکوسیستم بزرگتری است (تولباکس‌های بیشتر)، شاید به دلیل آن‌که سایلب برای مدت زمانی آزاد نبوده است.
گنو اُکتاو ساختاری یکپارچه را در عقبه‌ی خود ندارد. در حالی که سایلب مهندسانی را تمام وقت برای اهداف خویش به کار گرفته است (البته با صرف هزینه). یعنی: community driven **vs** integrated team driven.

---

[۱] در نسخه انگلیسی این آموزش، عنوان این قسمت « سایلب درمقابل متلب » می‌باشد. نویسنده آن‌را را در سال ۲۰۰۶ میلادی با توجه به نظرات کاربران در اینترنت جمع‌آوری کرده است. لذا آنچه که بیان شده بود لزوماً درست نیست. مترجم این قسمت را به مقایسه‌ی سه نرم‌افزار **سایلب**، **گنو اُکتاو** و **متلب** اختصاص داده است.

۹. مراجع و متون پیشنهادی[۱]

1. Scilab Help File (its own documentation).
http://help.scilab.org/
2. Scilab for Dummies.
http://www-irma.u-strasbg.fr/%7Esonnen/SCILAB_HELP/scilab_for_dummies.htm
3. Matlab Primer.
http://math.ucsd.edu/%7Edriver/21d-s99/matlab-primer.html
4. A Pratical Introduction to Matlab.
http://www.math.mtu.edu/%7Emsgocken/intro/intro.html
5. Introduction to Scilab, Scilab Group (2001).
http://www.snv.jussieu.fr/~wensgen/Doc/scilab-2.6/Intro-html/index.html
6. Introduction to Scilab, Michaël Baudin (Sep 2011).
http://forge.scilab.org/index.php/p/docintrotoscilab/
7. Scilab Official Wiki.
http://wiki.scilab.org/
8. Equalis - Scilab Online Support.
http://www.equalis.com/
9. Scilab Google Groups.
http://groups.google.com/forum/?fromgroups#!forum/comp.soft-sys.math.scilab

[۱] موارد ۱تا٤ مراجع اصلی نویسنده برای نوشتن این آموزش بوده است. باقی موارد مراجع و متون پیشنهادی‌ای می‌باشد که مترجم مطالعه‌ی آن‌ها را مفید می‌داند.

پیوست

نصب سایلب در گنو/لینوکس[۱]

برای نصب نرم‌افزار سایلب در توزیع‌های مختلف گنو/لینوکس (شامل: فدورا، اوبونتو، دبیان و غیره)، بهینه‌ترین راه استفاده از مخازن توزیع مورد استفاده است. البته کسانی که به اینترنت سریع دسترسی ندارند، امکان نصب برنامه از طریق دانلود کردن سورس برنامه از وب‌گاه سایلب و نصب نرم‌افزار وجود دارد.

نصب از طریق مخازن

با توجه به موجود بودن سایلب در مخازن فدورا، اوبونتو و دبیان، امکان نصب آن‌ها از طریق مخازن وجود دارد.

نصب در اوبونتو و دبیان

برای نصب این نرم‌افزار به همراه راهنمای آن در اوبونتو و یا دبیان، بایستی که دستور زیر را در ترمینال وارد نموده و سپس کلمه‌ی عبور حساب خود را وارد نمایید.

```
sudo apt-get install scilab scilab-doc
```

نصب در فدورا

برای نصب این نرم‌افزار به همراه راهنمای آن در فدورا، بایستی که ابتدا دستور زیر را در ترمینال وارد نموده و سپس کلمه‌ی عبور حساب ریشه‌ی خود را وارد نمایید.

```
su
```

و سپس دستور زیر را در ترمینال وارد نمایید.

```
yum install scilab scilab-doc
```

نصب از طریق دانلود کردن سورس از وب‌گاه سایلب

این روش بهترین روش برای کسانی است که به اینترنت سریع دسترسی ندارند و مراحل آن برای تمامی توزیع‌های گنو/لینوکس یکسان است.

برای این‌کار ابتدا به وب‌گاه زیر رفته و سایلب ۳۲ یا ٦٤ بیتی را متناسب با سیستم‌عامل خود تحت عنوان Linux دانلود نمایید.

http://www.scilab.org/products/scilab/download

حال فرض می‌کنیم که سیستم‌عامل ما یک توزیع ٦٤ بیتی است. بنابراین نسخه‌ی متناسب را دانلود می‌نماییم که به صورت پیش‌فرض پرونده‌های دانلود شده در پوشه‌ی Downloads قرار می‌گیرند. اسم پرونده مورد نظر چیزی شبیه به این است:
scilab-5.4.0.bin.linux-x86_64.tar.gz

---

[۱] این قسمت توسط آقای سلمان محمدی نوشته شده است.

حال ترمینال را باز می‌نماییم و سپس دستورهای زیر را به ترتیب در آن اجرا می‌نماییم (توجه: این مرحله ممکن است که تا حدود سه ساعت به طول بیانجامد).

```
cd Downloads
tar xzvf scilab-5.4.0.bin.linux-x86_64.tar.gz
```

حال هربار برای اجرا کردن سایلب بایستی که دستورهای زیر را در ترمینال وارد نماییم.

```
cd Downloads/scilab-5.4.0
./bin/scilab
```

پیوست ۲

راهنمای نصب جعبه‌ابزارهای سایلب

سایلب به گونه‌ای طراحی شده است تا کاربران آن بتوانند با نوشتن توابع جدید به گسترش و توسعه‌ی آن بپردازند. «جعبه‌ابزار» و یا «ماژول خارجی» نامی است که بر این افزونه‌ها می‌گذاریم. ایجاد یک ماژول جدید با کمک توابع از پیش تعریف شده‌ی سایلب و همچنین ایجاد صفحات راهنما برای این ماژول جدید کار بسیار آسانی بوده و یکی از دلایل موفقیت سایلب نیز همین است. برای آشنایی با نحوه‌ی ساخت یک ماژول جدید به ویکی رسمی سایلب مراجعه کنید.

از طرفی، بسیاری از ماژول‌های سایلب را نمی‌توان به صورت مستقیم و در حالت کد منبع استفاده کرد. ماژول‌هایی که براساس C و یا Fortarn کدنویسی شده‌اند ابتدا باید کامپایل شده و سپس فایل‌های باینری ایجادشده می‌توانند در سایلب لُود شوند. مرحله‌ی کامپایل‌کردن دردسرهای خاص خودش را داشته و حتی غیرممکن خواهد بود اگر کامپایلر مناسب بروی سیستم شما نصب نباشد. ATOMS ابزاریست که این گونه مسائل را برای ما حل کرده است. با کمک این ابزار شما می‌توانید ماژول‌های از پیش کامپایل‌شده را با توجه به نوع سیستم‌عامل (پلتفرم‌تان) دانلود، نصب و لُود کنید. همانند Packaging System در بسیاری از توزیع‌های گنو/لینوکس، در ATOMS نیز وابستگی‌ها (dependencies) به گونه‌ای مدیریت می‌شود که مثلا اگر ماژول «ب» پیش نیاز ماژول «الف» باشد، ماژول «الف» به طور خودکار ماژول «ب» را نصب می‌کند (البته در ATOMS GUI).

برای نصب یک ماژول علاوه بر ATOMS GUI می‌توانید به صورت دستی و با دانلود بسته‌های مناسب از وبسایت ATOMS اقدام به نصب آن نمایید.

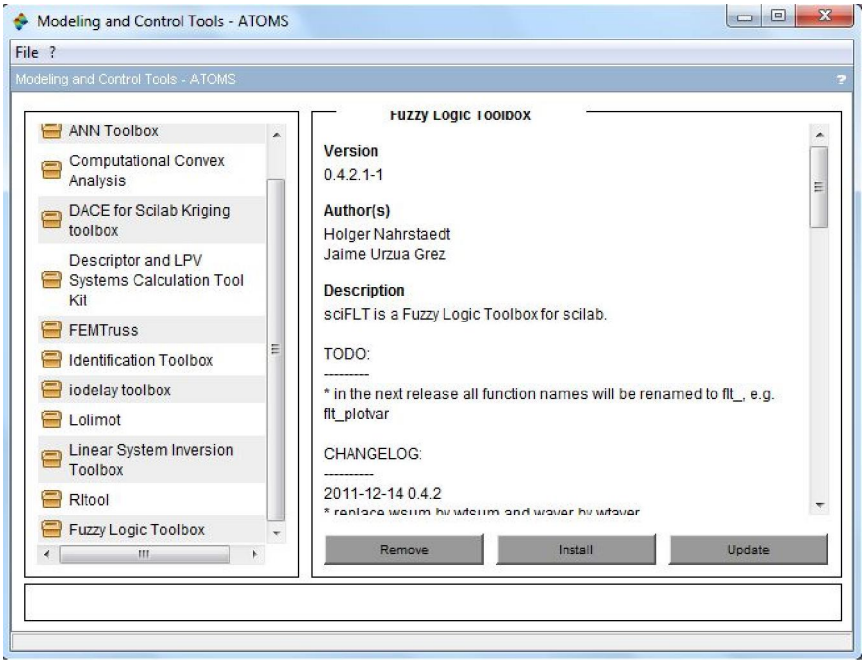
http://atoms.scilab.org

روش‌های نصب یک ماژول:

الف) استفاده از ATOMS GUI

برای این منظور پس از باز کردن نرم‌افزار سایلب در پنجره‌ی کنسول بروی آیکون کلیک کنید. توجه داشته باشید که حتما سایلب را از طریق مدیر سیستم اجرا کنید. کابران گنو/لینوکس با دستور su و کاربران ویندوز با کیلک راست بر روی آیکون برنامه و انتخاب گزینه‌ی Run as administrator. در غیر این‌صورت با پیغام خطایی مشابه آنچه در زیر آمده است روبه‌رو می‌شوید.

فرض کنید می‌خواهیم جعبه‌ابزار منطق فازی سایلب با نام sciFLT را نصب کنیم. این ماژول در دسته‌ی Modeling and Control Tools قرار دارد. در پنجره‌ی ATOMS GUI از ستون سمت چپ این دسته را انتخاب کرده و سپس بر روی Fuzzy Logic Toolbox کلیک کنید. در ستون سمت راست توضیحات مختصری درباره‌ی این جعبه‌ابزار برای شما نمایش داده می‌شود. برای نصب این جعبه‌ابزار فقط کافیست بروی دکمه‌ی install کلیک کنید.



ب) نصب یک ماژول به صورت دستی

ابتدا با توجه به نوع سیستم‌عامل‌تان فایل باینری مناسب را از وب‌سایت ATOMS دانلود کنید.

http://atoms.scilab.org/toolboxes/sciFLT

فایلی را که دانلود کرده‌اید به دیرکتوری زیر انتقال داده و در آنجا اکسترکت کنید:

<Scilab Installation Root>\contrib

بعد از اکسترکت کرد شما یک دیرکتوری جدید با نام افزونه‌ی موردنظر دارید (در اینجا مثلا sciFLT_0.4.2.1). دیرکتوری هر ماژول شامل چند زیر–دیرکتوری و نیز فایل‌های مختلفی می‌باشد. برای لُود کردن یک ماژول باید اسکریپت loader.sce را اجرا کنید. برای این منظور دیرکتوری جاری سایلب را به دیرکتوری ماژول موردنظر تغییر دهید.

دستور pwd آدرس دیرکتوری جاری سایلب را برمی‌گرداند. دستور SCI مسیر نصب نرم‌افزار سایلب را نشان می دهد. برای رفتن به دیرکتوری نصب ماژول موردنظر در پنجره‌ی کنسول تایپ کنید:

```
cd ('SCI'+filesep()+'contrib'+filesep()+'sciFLT_0.4.2.1')
```

در ادامه با دستور exec جعبه‌ابزار منطق فازی سایلب را اجرا کرده و می‌توانیم از توابع آن استفاده کنیم.

```
exec ('loader.sce')
```

برای دسترسی به راهنمای جعبه‌ابزار منطق فازی سایلب تایپ کنید:

```
help sciFLT
```

پیشنهاد می‌کنم ماژول‌ها را به صورت دستی نصب کنید زیرا در این حالت می‌توانید از منوی Toolboxes فقط ماژول‌هایی را که نیاز دارید لُود کرده و استفاده کنید. اما در صورت نصب اتوماتیک با کمک ATOMS GUI با هربار اجرای سایلب ابتدا باید منتظر بمانید تا تمامی ماژول‌ها لُود شوند، سپس می‌توانید از سایلب استفاده کنید.

تاریخچه سند

نسخه ۱،۰ (۸ آبان ۹۱):

- انتشار اولین نسخه.

نسخه ۱،۱ (۲۷ آذر ۹۱):

- افزودن بخش «پیوست» و آموزش جامع نصب سایلب در گنو/لینوکس؛
- اصلاحات ناچیز.

نسخه ۱،۲ (۳۰ اردیبهشت ۹۲):

- اضافه‌شدن «پیوست ۲» و آموزش نصب تولباکس‌های سایلب.