

ComsolScript för MATLAB-användare

Elin Bergeås och Linda Östman

Fysikum, november 2006

Innehåll

1 Inledning	1
2 Hur man startar ComsolScript	2
3 Hur man installerar och använder texteditorn Crimson	2
3.1 Buggar i Crimson	3
4 Skillnader i syntax och implementering	3
4.1 Att beräkna fakultet	4
5 Kända buggar och några allmänna tips	4
5.1 Kommandot <code>axis</code>	5
5.2 Kommandot <code>errorbar</code>	5
5.3 Kommandot <code>legend</code>	5
5.4 Funktioner som tar sifferargument	6
5.5 Öppen fil efter körning från menyraden	7
5.6 Sätt sökvägen till dina filer	7
5.7 "...men det funkar inte!"	7

1 Inledning

Från och med vårterminen 2006 är programmet ComsolScript installerat på alla Fysikums studentdatorer. MATLAB finns fortfarande på några datorer, men inte alla. Denna text är avsedd för dig som är bekant med MATLAB och nu vill lära dig att använda även ComsolScript. Om du aldrig har använt något av programmen förut hänvisar vi istället till en nybörjarmanual, exempelvis Sten Hellmans ComsolScript-kompendium som finns att köpa på studentexpeditionen.

ComsolScript ska fungera på samma sätt som MATLAB. Det betyder inte att allt är precis likadant, men de flesta `m`-filer som fungerar i MATLAB ska utan vidare kunna köras även i ComsolScript. Undantaget är de allra senaste finesserna i MATLAB, som inte nödvändigtvis finns i ComsolScript. Den största skillnaden mellan programmen är egentligen hur de ser ut. Se även stycke 4, om skillnader i syntaxen.

Denna text är skriven för version 1.0.0.300 av ComsolScript och version 7.0.1 av MATLAB, vilket är de versioner av programmen som var installerade på

Fysikums studentdatorer i november 2006. Andra versioner kan skilja sig något åt från dessa.

2 Hur man startar ComsolScript

För att starta ComsolScript dubbelklickar man på ComsolScript-ikonen

(se bilden till höger) som ska finnas på skrivbordet. Om ikonen saknas kan du leta efter ComsolScript bland programmen i START-menyn.



Om ComsolScript saknas där med kan du aktivera programmet genom att gå till START-menyn, välja Run, skriva `fy11x` och klicka OK. Nu ska nya ikoner ha dykt upp på skrivbordet, däribland en ComsolScript-ikon.

ComsolScript påminner mycket om MATLAB som det såg ut i de tidiga versionerna (4 och 5). En del finesser saknas, som till exempel det speciella fönster som visar variablerna i "workspace". ComsolScript har en inbyggd texteditor, men den är inte så kraftfull, och ComsolScripts tillverkare rekommenderar att man använder en extern editor för att redigera `m`-filer. I princip kan man använda vilket textredigeringsprogram som helst, även Anteckningar, men Fysikum rekommenderar Crimson Editor som stöder MATLAB-syntax (se nästa stycke).

När ComsolScript startas kommer du inte automatiskt att befinna dig i din hemkatalog, utan du står i ComsolScripts arbetskatalog. För att byta aktuell katalog använder du kommandot `cd` (som står för 'Change Directory'):

```
cd Z:\minkatalog\
```

Du kan se vilken katalog du befinner dig i med kommandot `pwd` (Print Working Directory) och lista alla filer i den aktuella katalogen med `ls` (LiSt). Den som är van att arbeta i Unix-miljö känner igen dessa kommandon.

Om man arbetar i en katalog men vill att ComsolScript ska leta filer även i andra kataloger kan man lägga till dessa i sökvägen med kommandot `path`:

```
path('Z:\minkatalog\katalog2',path);
```

`path` utan argument listar alla de sökvägar som finns för tillfället.

Notera att ComsolScript inte sparar sökvägen om du avslutar programmet. Hur man kommer runt detta problem beskrivs i stycke 5.

3 Hur man installerar och använder texteditorn Crimson

För att öppna Crimson kan du dubbelklicka på Crimson-ikonen (se bilden till höger). Om ikonen saknas på skrivbordet i datorsalen kan du få fram den genom att skriva `fy11x` i Run på samma sätt som beskrevs i stycke 2.



Ett annat sätt att öppna en texteditor är med kommandot `edit` i ComsolScript. Om inget annat angetts öppnas då ComsolScripts inbyggda editor. För att sätta Crimson till standard-editor, om det inte redan har gjorts, går du till **File**-menyn och väljer **Preferences...** I fönstret som då dyker upp prickar du för **External editor** och skriver in sökvägen till Crimson i fältet bredvid. I datorsalen är sökvägen `C:\Program_Files\Crimson Editor\cedit.exe`. Istället för att skriva in sökvägen, kan du även trycka på **Browse...** och leta dig fram till `cedt.exe`, vilket är ikonen ovan med en kanin. Tryck sedan 'OK' varvid Crimson kommer att vara standardredigeringsprogram i fortsättningen. Nästa gång du skriver `edit` kommer Crimson att öppnas istället för ComsolScripts editor.

I MATLABs editor läggs filtillägget `.m` till automatiskt. I Crimson måste du dock komma ihåg att själv lägga till det. Annars kommer du inte att kunna köra filen i ComsolScript.

Om du vill få texten i din fil färgkodad med MATLAB-syntax för att lättare se om du skrivit något felaktigt kommando, kan du i menyn **Document** i Crimson gå till **Syntax type** och ställa in `matlab`.

Om du vill installera Crimson hemma kan du ladda hem programmet från www.crimsoneditor.com, vilket är gratis. På hemsidan finns även en del tips om hur Crimson kan användas, samt ett diskussionsforum.

Crimson finns bara tillgängligt för Windows. Linuxanvändare kan skriva sina `m`-filer i exempelvis Emacs istället.

3.1 Buggar i Crimson

I ComsolScript används enkla citattecken (`'`) bl.a. för att markera textsträngar. Om man i Crimson väljer att färgkoda sin `m`-fil, kommer alla strängar att få en och samma färg (lila). Det finns dock en bugg i Crimson som gör att editorn tolkar all text som börjar med ett enkelt citattecken (`'`) som en textsträng och således färgar den lila, även om inget slutcitattecken förekommer. Eftersom det enkla citattecknet även kan användas till andra funktioner än strängar, till exempel matristransponat, gör det att en del kodrader kan bli felaktigt färgade. Ett exempel på detta är kommandoraden

```
y = x'+2
```

där allt efter citattecknet (`'`) på samma rad blir lila trots att det inte är en textsträng. Denna bugg är dock bara kosmetisk och har ingen betydelse för hur programmet utförs i ComsolScript.

Det här problemet finns också i Emacs för Linux, och där blir även de påföljande raderna textsträngsfärgade. För att undvika det kan du avsluta raden med `%'`.

4 Skillnader i syntax och implementering

Enligt tillverkarna ska ComsolScript vara "fully MATLAB compatible". Nåja... I princip ska alla kommandon som fungerar i MATLAB även fungera i ComsolScript. I Tabell 1 har vi sammanställt några kända skillnader i hur programko-

den skrivs. Se även stycke 5 för en genomgång av de kända buggarna i ComsolScript.

MATLAB	ComsolScript
<code>bar(vektor, 'stack')</code>	<code>bar(vektor, 'stacked')</code>
<code>barh</code>	– saknas –
<code>factorial</code>	saknas, se nedan för alternativ
<code>set(gca, 'XTickLabel', {'a'; 'b'; 'c'})</code>	<pre>h1=gca; x_tick = 1:3; set(h1, 'XTick', x_tick); set(h1, 'XTickLabel', {'a'; 'b'; 'c'})</pre>

Tabell 1: Kända skillnader i programkoden i MATLAB respektive ComsolScript.

4.1 Att beräkna fakultet

I ComsolScript finns inte funktionen `factorial` som beräknar fakultet. Detta är dock enkelt att koda på egen hand. Om du vill beräkna fakulteten av ett tal a kan man skriva `prod([1:a])`, vilket i praktiken multiplicerar ihop talen i vektorn `[1:a]`, d.v.s. fakultet beräknas. `prod` kan även användas i MATLAB.

Man kan också skriva en egen m-fil `factorial.m`:

```
function f = factorial(a)
% function f = factorial(a)
% Beräknar fakulteten av heltalet a, f = a!
f=prod([1:a]);
```

(För att göra filen generell bör den kompletteras med specialfallet $0! = 1$, samt en kontroll att a är ett heltal).

5 Kända buggar och några allmänna tips

I Tabell 2 finns en lista över de kommandon som innehåller kända buggar i ComsolScript. Listan är inte på något vis komplett. Samtliga upptäckta buggar är rapporterade till utvecklarna av ComsolScript, så förhoppningsvis kommer de åtgärdas till nästa version.

<code>axis</code>	se stycke 5.1
<code>errorbar</code>	se stycke 5.2
<code>legend</code>	se stycke 5.3
<i>Funktioner som tar sifferargument:</i>	
<code>abs(x)</code> , <code>cos(x)</code> , <code>exp(x)</code> , <code>sin(x)</code> , <code>tan(x)</code>	se stycke 5.4

Tabell 2: Några av de kommandon i ComsolScript som innehåller kända buggar.

5.1 Kommandot axis

Kommandot `axis` ska returnera min- och maxvärdena för x - och y -axlarna i den aktuella figuren. Detta fungerar inte som det ska i ComsolScript. Uttrycket `variabel = axis` returnerar istället alltid $[-1 \ 1 \ -1 \ 1]$. Problemet beror antagligen på att figuren inte uppdateras efter kommandot `plot`.

Man kan komma runt detta genom att anropa figuren igen när allting är färdigritat och sedan manipulera axlarna.

Kodexempel:

```
clear all; close all;

x= -2.5: .05: 2.5;
y=sin(x);

figure(1);
plot(x,y);
hold on;
plot(x(2),y(2),'ro');
figure(1); % ==> Om denna rad tas bort fungerar inte
           % axis-kommandot som det ska!
current_axis = axis;
luft = 0.5;
ny = [current_axis(1)-luft, current_axis(2)+luft,...
      current_axis(3)-luft, current_axis(4)+luft];
axis(ny);
title('Omskalning av axlarna');
```

(Testa gärna vad som händer om man tar bort andra `figure(1)`; i koden ovan!)

5.2 Kommandot errorbar

Kommandot `errorbar` fungerar ungefär som `plot`, förutom att det även ritar ut felstaplar i y -led. `errorbar` kan förutom argumenten x , y och dy även ha ett fjärde argument med instruktioner om hur markörerna ska se ut. Detta argument kan innehålla en färg, en punktmarkör och/eller en linjestil. En bugg i ComsolScript gör att man alltid måste ange en färg om man vill använda sig av något av de andra alternativen. Det går alltså att skriva `errorbar(x,y,dy,'go')`, vilket ger gröna cirklar, men inte `errorbar(x,y,dy,'o')`. Kommandot `plot` har dock inte detta problem och där kan man skriva `plot(x,y,'o')`.

5.3 Kommandot legend

Kommandot `legend` används när man vill ha en ruta med förklarande text till de olika grafer man har ritat i sin figur. Kommandot har flera problem i ComsolScript. Textrutan som skapas då `legend` används läggs bredvid grafen

(och inte i grafen) vilket gör att ju längre text du skriver desto mindre utrymme får grafen i x -led. Försök därför att fatta dig kort om du använder `legend`.

Ett annat problem med `legend` uppstår i samband med `errorbar`. Om du använder `legend` efter `errorbar` skapas två rader i legend-rutan för felstaplarna, en med symbolen som markerar punkten (t.ex. `*` eller `o`) och en för stapeln.

Detta blir ännu mer förvirrande om du väljer att ha linjer mellan dina datapunkter, d.v.s. din kodrad har följande utseende

```
errorbar(x,y,dy,'b-')
legend('legend-text')
```

Då får du två identiska horisontella blå linjer i din legend-ruta som du förväntas skriva kommentarer till.

För att bara få en rad för punktmarkören och ingen för stapeln kan du göra på följande sätt istället:

```
plot(x,y,'b*')
legend('legend-text')
hold on
errorbar(x,y,dy,'b*')
```

Figuren kommer då att se likadan ut, bortsett från att du bara får en rad i legend-rutan. Detta beror på att `legend` endast lägger till rader för de `plot`- och `errorbar`-kommandon som finns ovanför `legend`. Observera att det är viktigt att du använder samma färg och punktmarkör i `plot` och `errorbar` i detta exempel eftersom texten i legend-rutan annars kommer att bli felaktig.

Till ComsolScripts försvar ska sägas att `legend` kan uppvisa liknande problem även i MATLAB, om man har ritat flera olika grafiska element i samma figur.

5.4 Funktioner som tar sifferargument

Om man glömmet parenteserna när man använder funktioner som tar sifferargument (exempelvis `abs(x)`, `cos(x)`, `exp(x)`, `sin(x)` eller `tan(x)`) räknar ComsolScript ut värdet för tecknets ASCII-kod¹ istället.

Exempel:

Tecknet 0 har ASCII-kod 48, 1 har 49

```
sin 0, sin(48)
ans =
-0.7683
ans =
-0.7683
```

Flera tecken genererar en vektor:

```
sin 01
ans =
-0.7683 -0.9538
```

Var alltid noggrann med att skriva ut parenteserna när du använder funktioner!

¹ASCII står för American Standard Code for Information Interchange. En dator kan inte direkt hantera tecken, så som de skrivs ut på skärmen, utan bara siffror (egentligen bara binära tal). Varje tecken får istället en speciell sifferkod, som representerar tecknet. Notera att det är skillnad på *talet* 0 och *tecknet* 0!

5.5 Öppen fil efter körning från menyraden

En bugg i ComsolScript gör att en m-fil inte stängs ordentligt om man kör den från menyraden ('File' → 'Run M-file...') vilket kan få som följd att filen inte går att spara. Detta är enkelt att kringgå genom att helt enkelt anropa filen direkt från Comsol-prompten istället, genom att skriva dess namn (utan .m-ändelsen) och trycka på 'Enter'.

5.6 Sätt sökvägen till dina filer

När ComsolScript avslutas sparas inte sökvägen till filerna. Man kan komma runt detta genom att skapa ett skript, `setmypath.m`² som sätter alla de sökvägar man vill ha. Exempel på hur detta skript kan se ut:

```
% setmypath.m
% Sätter sökvägen till dina filer.

% Sätt sökvägarna
path('Z:\labbar\funktionsfiler',path);
path('Z:\labbar\kvantlabbar',path);
path('Z:\labbar\kvantlabbar\zeeman',path);
path('Z:\labbar\kvantlabbar\compton',path);

%Skriv ut sökvägarna på skärmen
path
```

Byt ut `labbar`, `funktionsfiler` etc mot namnen på dina kataloger. Spara filen i din toppkatalog (alltså precis under `abcd1234`). Sätt sökvägen till dina filer genom att skriva i ComsolScripts prompt:

```
cd Z:\
setmypath
```

5.7 "...men det funkar inte!"

När ingenting annat fungerar, läs manualen.

ComsolScript har, liksom MATLAB omfattande hjälpfiler. `help kommando` returnerar en hjälpfil precis som i MATLAB, och man kan även söka mer övergripande i dokumentationen under "Hjälp"-menyn.

Denna text innehåller inte på något vis alla de skillnader som finns mellan programmen. Dessutom kan olika versioner av samma program ha olika egenskaper. Vi hoppas dock att texten har hjälpt dig att komma igång med ComsolScript så att du själv kan söka lösningar på eventuella problem.

²Denna fil kan även hämtas på www.physto.se/~elin/undervisning