

ATK TÄHTITIETEESSÄ: IDL:n perusteita

Heikki Salo 1992/11-10.2005 (latex-käännös Raine Karjalainen 2004)

IDL on monipuolinen grafiikka-ohjelmointikieli, jota voidaan käyttää sekä yksinkertaisessa datan plottauksessa ja muokkaamisessa, sekä esim. kuvankäsittelytehtävissä. IDL:n oleellisia piirteitä ovat mm.:

- yksinkertainen, johdonmukainen komentosyntaksi
- mahdollisuus sekä interaktiiviseen että ohjelmalliseen käyttöön
- erilaisten datamuotojen samankaltainen käsittely (skalaarit, vektorit jne.)
- monipuoliset input/output komennot esim. FORTRAN ohjelman tuottaman datan lukuun
- helppokäyttöinen ja nopea muuttujien talletus ja luku IDL:n omiin save-tiedostoihin
- runsaat valmiit ohjelmakirjastot
- suhteellisen nopea: hyvä IDL-koodi sopivassa ongelmassa on lähes yhtä tehokasta kuin optimoitu FORTRAN-koodi

Huom!

- EI menu-ohjattu kuten esim. monet MacIntosh piirto- ja kuvankäsittelypaketit (tällaisia systeemejä voi kuitenkin IDL:n avulla rakentaa)
- tekee numeerista manipulaatiota, ei symbolista (kuten esim. MATHEMATICA tai REDUCE)

Atk tähtitieteessä kurssin IDL-osuudessa tutustutaan tämän kielen perusteisiin, käytettynä UNIX-työasemilta. Lähinnä käsitellään yksinkertaiset peruskomennot, joiden avulla pääsee alkuun kielen omakohtaisessa opettelussa manuaalien avulla. Tavoitteena on että kurssin jälkeen on valmiudet käyttää IDL:ää esim. tähtitieteen harjoituksissa. IDL on käytettävissä mm. ATK-keskuksen lastuNN ja sun4-koneissa. Perusteellinen kuvaus IDL-paketista on manuaaleissa:

- IDL user's guide - Introduction to IDL
- IDL basics - IDL user's guide:Addendum

joita on myös käytetty apuna tätä opasta kirjoitettaessa.

1 IDL:n käynnistäminen

-Oletetaan, että käyttäjän .cshrc tiedostoon on määritelty tarpeelliset unix ympäristömuuttujat PATH ja IDL_PATH (sisältää hakemistopolut, joista idl etsii proseduuritiedostoja).

-> Tällöin IDL käynnistyy kirjoittamalla *idl*, jolloin päädytään IDL-komentotilaan ja voidaan esim. antaa interaktiivisia plottauskomentoja. IDL-komentotilasta poistutaan komennolla *exit*. IDL-komentotilasta voidaan, siitä poistumatta, aina antaa UNIX-komentoja laittamalla niiden eteen \$-merkki: esim. *\$ls -altr*. Useimmat IDL komennot ja proseduurit voidaan keskeyttää antamalla *ctrl-c*, jolloin palaudutaan IDL-komentotilaan.

2 Esimerkkejä interaktiivisesta käytöstä

Parhaiten IDL:n perusominaisuudet tulevat tutuksi kokeilemalla erilaisia interaktiivisia komentoja:

<i>print, 3*5</i>	tulostaa 3*5:n arvon
<i>a=3*5</i>	luo muuttujan a ja sijoittaa arvoksi 3*5
<i>help, a</i>	antaa tietoa a:sta: skalaari, kokonaisluku, arvo=15
<i>a=sqrt(a) & help, a</i>	& -merkillä voi erottaa useita komentoja rivillä
<i>a=[1,2,3,4,5]</i>	määrittelee a:n vektoriksi
<i>print, a, 2*a</i>	
<i>a=a/total(a)</i>	normeerataan a: total on IDL:n funktio joka antaa alkioiden summan
<i>plot, a</i>	plotataan (avaa ikkunan mikäli sellaista ei vielä ole)
<i>print, a</i>	tulostaa a:n alkiot
<i>x=fltarr(100)</i>	määrittelee x:n 100 alkion reaalityyppiseksi
<i>for i=0,99 do x(i)=i</i>	määrittelee x(i)=i
	Huom! vektoreitten indexit alkavat 0:sta
<i>plot, x</i>	
<i>y=sin(6*pi/100.*x)/exp(x/50.)</i>	!pi syteemimuuttuja, joka sisältää π :n
<i>plot, y &</i>	
<i>x=findgen(100)</i>	määrittelee vektorin x=[0.,1.,...,99.]
<i>plot, x</i>	
<i>x=x/100.*6*pi</i>	x saa arvoja välillä 0 - 6π
<i>plot, x, sin(x), xtitle='x', ytitle='sin(x)'</i>	
<i>help</i>	
<i>y=sin(x)</i>	
<i>plot, x, y</i>	
<i>z=x#y</i>	määrittelee 2-ul. taulukon z=(i,j)=x(i)*y(j)
<i>surface, z</i>	piirtää z:n pintana
<i>contour, z</i>	piirtää z:n contour-kuvana
<i>shade_surf, z</i>	
<i>tek_color</i>	määrittelee tektronix-väripaletin
<i>plot, x, y, color=2</i>	piirtää punaisella värillä
<i>oplot, x, y, color=3</i>	piirtää päälle vihreällä
<i>oplot, x, abs(y), color=4</i>	piirtää päälle sinisellä
<i>oplot, x, sqrt(abs(y)), color=5, psym=4</i>	piirtää päälle keltaisella
	psym=4 plottaa pisteet laatikoilla
<i>x=randomu(seed, 1000)</i>	
<i>y=randomu(seed, 1000)</i>	1000 satunnaislukua väliltä (0,1)
<i>plot, x, y, psym=3</i>	

Esimerkeistä näkee, että IDL sisältää normaali matemaattiset funktiot ja laskutoimitukset, esim. $y = \sin(x)$. Koska IDL aina käsittelee numeerista dataa, komennon argumenttina voi olla matemaattinen lauseke, esim. $plot, abs(y)$, jolloin lausekkeen numeroarvo lasketaan ennen komennon suoritusta. IDL käsittelee erityyppisiä muuttujia (skalaarit, vektorit, moniulotteiset taulukot) samalla tavalla:

```
a=1 & b=2 & c=a+b & print,c  
print,2*c  
c=[1,2,3] & print,2*c
```

Harjoittele erilaisten lausekkeiden plottaamista:

- Muotoa $x = \text{findgen}(1000) * 0.001$ tyyppisillä määrittelyillä saat sopivia argumentteja, josta voi sitten rakentaa erilaisia muuttujia y.

- 2-ulotteisia funktioita saa kätevästi vektoreiden ulkotulo-operaation `#` avulla (voi kokeilla `surface`, `contour`, `shade_surf` -komentoja)

- Yhtä riviä pitempi komento: käytetään `$` merkkiä rivin lopussa:

```
x=findgen(1000)*.1 & y=sqrt(x)*$  
sin(x) & plot,x,y
```

-Huom! IDL ei tee eroa isojen ja pienien kirjaimien välillä.

Esimerkkien mukaisesti kaikkien IDL komentojen syntaksi on muotoa

komento, parametri1, parametri2, ..., keyword1=value, keyword2=value, ...

eli komento ja sen jälkeen komennon vaatimat parametrit (tietty järjestys) pilkulla erotettuna. Lisäksi komennolle voi antaa erilaisia keyword-parametrejä (järjestyksellä ei väliä), samoin pilkulla erotettuna. Useat komennot eivät vaadi kaikkien parametrien antamista, tai sitten ne toimivat eri tavalla parametrien lukumäärän mukaan: esim. $plot, x$ piirtää taulukon x arvot käyttäen taulukko-indeksiä abskissana, $plot, x, y$ piirtää y :n x :n funktiona. Yleensä komennot tekevät käyttäjän puolesta tiettyjä oletuksia: mm. $plot$ -komento automaattisesti olettaa tietyt oletusarvot (defaults) esim. plottaus-symbolille ja akseliväleille, ellei niitä muuteta avainsanojen (keywords) perusteella: esim. $plot, x, psym=4$ plottaa kunkin datapisteen erikseen, yhdistämättä niitä viivalla kuten oletusarvo $psym=0$ tekee. Muotoa $keyword=1$ olevat avainsana-määrittelyt voi lyhentää $/keyword$.

IDL sisältää myös suuren määrän erilaisia proseduureja, edellisissä esimerkeissä esim. $total$ -funktio, tek_color -proseduuri, joiden tekemiseen palataan myöhemmin.

Huom! IDL on rivin kerrallaan kääntävä ohjelmointikieli (kuten yksinkertaiset BASIC-tulkkit, päinvastoin kuin FORTRAN tai C).

3 IDL-muuttujista

3.1 Data-typing

byte-muuttujat	<i>100b</i>	1-tavun muuttuja	0-255
integer	<i>100</i>	2-tavun kok.luku	0-32767
long integer	<i>100l</i>	4-tavun kok.luku	0-(2 ³¹ -1)
floating	<i>100., 1e2</i>	4-tavun liukuluku	6 numeron tarkkuus
double floating	<i>100d, 100.d0</i>	8-tavun liukuluku	12 numeron tarkkuus
string	<i>'100'</i>	merkkimuuttuja	

Kokeile esim. antamalla

$x=100b$ & $print, x$ & on erotinmerkki joka mahdollistaa
monta komentoa samalla rivillä

Huom! kokonaislukumuuttuijen suhteen oltava varovainen:

long integer, ei integer, vastaa esim. normaalia FORTRAN integer-muuttujaa lukualueeltaan! Oltava varovainen, koska IDL ei valita lukualueen ylityksestä: kokeile mitä antavat komennot

```
print,30000+30000
print,30000l+30000l
print,30000+30000l
```

Tyyppi-konversiot:

$b = \text{byte}(a)$	$b = \text{fix}(a)$
$b = \text{long}(a)$	$b = \text{float}(a)$
$b = \text{double}(a)$	$b = \text{complex}(a)$
$b = \text{string}(a)$	

3.2 Matemaattiset operaatiot

Suoritetaan normaalissa aritmeettisessa järjestyksessä:

eksponentti	\wedge
kerto- ja jakolasku, modulo	$*, /, \text{mod}$
yhteen- ja vähennyslasku, minimum, maximum	$+, -, <, >$

$a \geq 2^*3$	a potenssiin 2, kerrotaan kolmella
$\hat{a}(\geq 2^*3)$	a potenssiin 6
$a < 2$	pienempi a:sta ja 2:sta
$a > 2^*3$	suurempi a:sta ja 6:sta

laskutoimitus suoritetaan suurimmalla mahdollisella tarkkuudella:

esim. 2+3. suoritetaan siten että 2 muutetaan liukuluvuksi ja lisätään 3.:een

Huom: oltava varovainen

$a=5$ *kokonaisluku*

$a/2=2$ $a/2.=2.5$ $a/2+1.=3.$ $a/2.+1=3.5$

3.3 Matemaattisia funktioita

IDL sisältää normaali matemaattiset funktiot, esim.

$\sin(x), \cos(x), \tan(x)$	argumentit radiaaneina
$\operatorname{asin}(x), \operatorname{acos}(x), \operatorname{atan}(x)$	
$\operatorname{atan}(y,x)$	palauttaa kulman α , jolle $y=\sin(\alpha)$ ja $x=\cos(\alpha)$
$\sinh(x), \cosh(x), \tanh(x)$	
$\exp(x), \sqrt{x}, x^2, \operatorname{abs}(x)$	
$\operatorname{alog}(x), \operatorname{alog10}(x)$	luonnollinen ja 10-kantainen logaritmi

3.4 Vektorit, moniulotteiset taulukot

Huom! indeksien numerointi alkaa aina 0:sta, ei 1:sta!

Vektorien luonti:

$a=[0,1,2,3,4]$	määrittelee a:n kokonaisluku-vektoriksi (0,1,2,3,4)
$a=\operatorname{indgen}(5)$	sama vaikutus: vektori, jonka alkion arvo=alkion indeksi
$a=\operatorname{findgen}(5)$	reaalilukuektori
$a=\operatorname{lindgen}(5)$	long integer
$a=\operatorname{dindgen}(5)$	

$a=\operatorname{intarr}(5)$	määrittelee a:n kokonaislukuektoriksi (alkiot=0)
$a=\operatorname{fltarr}(5)$	
$a=\operatorname{lonarr}(5)$	
$a=\operatorname{dblarr}(5)$	
$a=\operatorname{bytarr}(5)$	

$b=a$	b:n tyyppi määritellään samaksi kuin a:n tyyppi ja lisäksi a:n arvot sijoitetaan b:n arvoiksi
-------	---

$a=\operatorname{findgen}(10)$	
$b=2*a$	$b(i)=2*a(i)$ kaikilla i
$b=a*a$	$b(i)=a(i)*a(i)$ kaikilla i

Alkioihin viittaaminen:

Oletetaan a on 10-alkion vektori

```
b=a(5)    a:n viides alkio
b=a(2:6)  sama kuin [a(2),a(3),a(4),a(5),a(6)]
b=a(7:*)  sama kuin [a(7),a(8),a(9)]
b=a(*)    b=a
```

$a(3:6)=0$ asettaa a(3), a(4), a(5), a(6) nolllaksi, muut alkiot ennallaan

arvoalue voidaan antaa myös vektorina:

```
a=[1,2,3,4,5,6]
b=[1,3,4]
c=a(b)          c=[2,4,5]
```

where -funktio:

```
a=[1,2,3,4,5,0]
index=where(a >= 4)      index=[3,4]
```

$data(where(data \leq 0)) = 0$ muuttaa taulukon data negatiiviset arvot nollliksi
(sama kuin $data=data>0$)

```
bad=where(data < 0)      korvataan negatiiviset alkiot
data(bad)=(data(bad-1)+data(bad+1))/2 vierekkäisten alkioden keskiarvolla
```

Moniulotteiset taulukot

```
a=ftarr(10,20)  a määritellään 2-ul. 10*20 taulukoksi
b=a(5:9,15:19)  b tulee 5*5 taulukoksi johon sijoitetaan a:n arvot annetulta osaväliltä
c=a(*,0)        a:n ensimmäinen pystyrivi [a(0,0),a(1,0),...,a(9,0)]
c=a(*,0:4)      a:n 5 ensimmäistä pystyriviä
a(5:6,7:10)=100
a(*,7:10)=100
```

Talletusjärjestys

```
olet a=2*3 matriisi:  a(0,0)  a(0,1)
                      a(1,0)  a(1,1)
                      a(2,0)  a(2,1)
```

järjestys muistissa a(0,0),a(1,0),a(0,1),a(1,1),a(2,0),a(2,1)

Moniulotteiseen taulukkoon voidaan viitata myös yksiulotteisena:

esim.

```
a=[[1,2],[3,4],[5,6]]      on 2*3 matriisi
```

```
print,a
```

```
print,a(1,2)
```

```
print,a(6)
```

```
print,a(*)
```

4 IDL-proseduurit

IDL:n hyvin käyttökelpoinen piirre on ns. proseduuritiedostot, jotka sisältävät useita IDL-komentoja ja laskutoimituksia. Ne voivat olla joko pääohjelma-, aliohjelma- tai funktio-tyyppisiä.

Huom! Esimerkeissä olevat ohjelmat löytyvät lastu:n hakemistosta `~hsalo/IDL_ESIM.dir`, josta ne voidaan kopioida `cp` komennolla:

```
mkdir IDL_ESIM.dir    tee itsellesi hakemisto
cp ~hsalo/IDL_ESIM.dir/* ./IDL_ESIM.dir    tulevat nykyisen hakemiston
alapuolella olevaan IDL_ESIM.dir-hakemistoon
```

Jos annat komennon IDL komentoriviltä, niin lisää `$` komennon eteen.

4.1 Pääohjelma-proseduurit

Avataan emacs-editori kirjoittamalla `$emacs idlesim1.pro`

Kirjoitetaan sinne seuraavat rivit (`;` = IDL:n kommenttirivin aloitusmerkki)

```
;-----
;idlesim1.pro
;esimerkki IDL-pääohjelmasta
;-----
r=10.
ala=!pi*r^2
piiri=2*!pi*r
print,'sade,piiri,ala'
print,r,piiri,ala
end           ;proseduurin pitää päättyä end-riviin
;-----
```

ja poistutaan emacsista tallettaen tiedosto (`ctrl -x -c`llä, vastataan `y`)

- tiedoston sisältö voidaan tarkistaa antamalla

```
$cat idlesim1.pro
```

- eo. pääohjelma tiedosto suoritetaan komennolla

```
.run idlesim1
```

jolloin ohjelman tulostuksen pitäisi tulla näkyviin

- IDL-proseduurin oletustarkennin on muotoa `.pro`, joten sitä ei tarvitse antaa mikäli tiedoston nimi on tätä muotoa (hyvä ohjelmointitapa)

Pääohjelma-tyyppinen proseduurit käyttää samaa työtilaa kuin interaktiivisesti annetut komennot: antamalla `eo.` ohjelman suorituksen jälkeen komento `help` nähdään että muuttujat `ala` ja `piiri` on määriteltynä.

Editoidaan idlesim1.pro tiedostoa:

-Kirjoita *\$emacs idlesim1.pro &*

Tällöin & kertoo normaaliin tapaan UNIX:lle, että editointi tehdään taustalla (IDL-ikkunaan voidaan antaa uusia komentoja).

-Lisätään kommentti-merkki rivin *r=10.* eteen (;) ja talletetaan tiedosto ctrl -x -s:llä, eli poistumatta emacsista.

-anna nyt IDL-ikkunassa uudelleen komento *.run idlesim1*

-tulostuksen pitäisi olla entisen kaltainen, sillä *r:n* arvo on ennallaan työtilassa.

-kirjoita nyt *r=20.* IDL-ikkunassa ja suorita komento *.run idlesim1*

-nyt *r:n* arvo on erilainen työtilassa, ja pääohjelma-proseduuri lukee sen sieltä

Huom! kannattaa omaksua tämä ohjelmien kehitystapa: editoidaan ohjelmaa erillisessä taustalla pyörivässä emacs-ikkunassa, talletetaan muutokset ja katsotaan mitä on saatu aikaan.

Eo. yksinkertaisen esimerkin perusteella ei oikein vielä saa kuvaa pääohjelma-proseduurien hyödyllisyydestä, mutta ne ovat erittäin käyttökelpoisia esim. tulostettaessa kuvia, joissa tehdään paljon erilaisia määrittelyjä ja halutaan tallettaa ne myöhempää käyttöä varten. Ohjelmaan voidaan nimittäin sisällyttää erilaisia silmukka- ja hyppyrakenteita, sillä toisin kuin interaktiivisessa käytössä, IDL käsittelee koko ohjelmaproseduurin kerralla (tosin tulkkauksessa tapahtuu edelleen rivi kerrallaan) ja siinä voi siten olla viittauksia eri ohjelman osien välillä. Pääohjelmien debuggaus on myös erittäin helppoa, sillä jos suoritus pysähtyy virheeseen voidaan helposti katsoa eri muuttujien arvoja virheen sattuessa.

4.2 Aliohjelma-proseduurit

Toinen IDL-proseduurityyppi on aliohjelmaproseduuri, joka poikkeaa pääohjelmasta siinä, että sille syötettävät arvot ja sen laskemat arvot siirretään kutsussa määrittelyjen argumenttien avulla. Aliohjelmalla on oma muuttuja-alueensa.

Aliohjelma-proseduuri määritellään sen alussa olevalla rivillä,

`PRO aliohjelma, arg1, arg2, ..., keyword1=keyword1, keyword2=keyword2,..`

jossa kerrotaan aliohjelman nimi, sen (mahdolliset) argumentit ja avainsanat.

Huom! ohjelman nimen ei tarvitse olla sama kuin sen tiedoston johon se on kirjoitettu, samassa tiedostossa voi olla myös useita aliohjelmia. Aliohjelmat voivat kutsua toisia aliohjelmia. Muokataan edellä olevasta idlesim1.pro-pääohjelmasta idlesim2.pro-aliohjelma:

```
;-----  
;idlesim2.pro  
;esimerkki IDL-aliohjelmasta  
pro ympyra,r,ala,piiri,print=print  
;-----
```



```

ala=!pi*r^2
piiri=2*!pi*r
if keyword_set(print) then begin
print,'säde,piiri,ala'
print,r,piiri,ala
endif
return
end
;-----
;proseduurin pitää päätyä end-riviin
;-----

```

Aliohjelma käännetään `.run idlesim2.pro` -komennolla, joka ei kuitenkaan suorita sitä. Suoritus tapahtuu kirjoittamalla proseduurin nimi ja argumentit pilkulla erotettuna, samoin keywordit. Esim.

```

ympyra,10,ala,piiri  palauttaa muuttujat ala,piiri
ympyra,10,/print     tulostaa säteen, alan ja piirin, mutta ei palauta niiden arvoja
                     koska niitä ei ole kutsussa argumentteina
ympyra               antaa virheilmoituksen sillä r on
                     määrittelemätön aliohjelmassa

```

Huom!

- 1) Mikäli proseduuria muutetaan, on se muistettava kääntää uudelleen.
- 2) Mikäli aliohjelma-proseduuri päättyy virheeseen, jäädään aliohjelman varaamaan työtilaan. Siten varsinaisen IDL-työtilan muuttujat eivät ole käytettävissä (hävinneet salaperäisesti!). Takaisin päästään `retall`-komennolla. Kannattaa muistaa, sillä tämä on käytännön ohjelmoinnissa erittäin tarpeellinen komento!
- 3) Aliohjelmaan syötettävät input-parametrit voivat olla joko muuttujia tai vakioita. Output-parametrien on oltava muuttujia, mutta niiden kaikkien ei tarvitse olla mukana kutsussa. Huomaa kuitenkin, että argumenttien järjestyksellä on väliä.
- 4) Avainsanojen testaamiseen käytetään `keyword_set(keyword)` rakennetta. Ehtorakenteista myöhemmin lisää.

Joskus omien aliohjelmien vaatimat argumentit ja niiden järjestys voi olla hankala muistaa. Käytännössä hyvä apukeino on lisätä ohjelmiinsa seuraavan kaltainen rakenne, jossa testataan proseduuriin syötettävien argumenttien lukumäärää, ja tulostetaan proseduurin käyttöohje mikäli argumentteja on liian vähän.

Esim. `idlesim2.pro` ohjelmaan voidaan lisätä heti PRO-määritteen jälkeen rivit

```

if n_params() eq 0 then begin
print,'pro ympyra,r,ala,piiri,print=print'
return
endif
;ohjelmaa kutsuttu ilman parametreja?
;kerrotaan mitä ohjelma haluaa
;hypataan pois

```

`n_params()` on IDL:n funktio, joka palauttaa aliohjelmaan syötettyjen argumenttien määrän. Sitä voi käyttää myös esim. tilanteissa joissa ohjelman toiminta on erilainen riippuen argumenttien määrästä (vrt. IDL'n `atan`-funktio).

4.3 Funktio-proseduurit

Kolmas proseduurityyppi on funktio-aliohjelmat. Ne määritellään rivillä

FUNCTION funktion_nimi, arg1, arg2,..., keyword1=keyword1, keyword2=keyword2,...

ja erona aliohjelmaan on, että funktiot palauttavat vain yhden arvon, joka määritellään *return*-lauseessa.

Esim. luodaan tiedosto idlesim3.pro, joka sisältää rivit

```
;-----  
;idlesim3.pro  
;esimerkki funktio-aliohjelmasta  
;-----  
function area,r  
ala=pi*r^2  
return,ala  
end  
;-----
```

Nyt komento $y=area(x)$ palauttaa x-säteisen ympyrän alan.

4.4 Erilaisia komentorakenteita

IDL proseduureissa voidaan käyttää mm. seuraavia esimerkein esitettyjä rakenteita:

FOR-silmukat

```
FOR i=0,10 DO print,i    ;printtaa luvut 0,1,...,10  
FOR i=0,10,2 DO print,i ;printtaa luvut 0,2,4,6,8,10  
  
FOR i=0,10 DO BEGIN    ;BEGIN ilmoittaa että seuraa useita lausekkeita  
j=i*i  
print,i,j  
ENDFOR                ;päättää FOR-silmukan
```

Voidaan tehdä myös sisäkkäisiä silmukoita:
(ohjelmapätkä on talletettu idlesim4.pro nimelle).

```
;-----  
;idlesim4.pro  
;esimerkki pinnan määrittelemisestä sisäkkäisten for-silmukoiden avulla  
;sijoittamalla erilaisia funktioita saadaan mielivaltaisia  $z=z(x,y)$  pintoja  
;Huom! aiemmissa esimerkeissä on luotu 2-ulotteisia taulukoita ulkotulo-operaation  
; # avulla jolloin  $z(x,y)$  on ollut aina muotoa  $z(x,y)=f(x)*f(y)$ .  
;-----  
  
x=fltarr(51,51) & y=x & z=x    ;kokomäärittelyt
```

```

FOR i=0,50 DO BEGIN
FOR j=0,50 DO BEGIN
x(i,j)=float(i)
y(i,j)=float(j)
ENDFOR
ENDFOR
x=x/5.-5. ;x on vektori, sisältää arvoja väliltä -5,5
y=y/5.-5. ;y samoin
xlab=findgen(51)/5.-5. ;akselien merkintää varten
ylab=xlab
z=exp(-0.125*(x^2+y^2))*sin(2*y)*cos(y)
loadct,3 ;otetaan käyttöön eräs IDL-paleteista
shade_surf,z,xlab,ylab ;plotataan varjostettuna pintana
end
;-----

```

IF-rakenteet

```

IF x NE 2 then print,'x ei ole 2'
IF(x NE 2) then print,'x ei ole 2' ;sulkua saa käyttää
IF(x EQ 2) then print,'x on 2'
IF(x LE 2) then print,'x pienempi tai yhtäsuuri kuin 2'
IF(x GT 2 OR x LT 0) then print,'x suurempi kuin 2 tai negatiivinen'
IF(x GT 2 AND x LT 10) then print,'x suurempi kuin 2 ja pienempi kuin 10'

IF(x NE 0) THEN BEGIN
print,'x=0'
ENDIF ELSE BEGIN
y=1./x
ENDELSE

```

WHILE-rakenne:

WHILE ehto DO toiminta

suoritetaan toiminto kunnes ehto ei ole enää voimassa:

esim.

```

WHILE NOT EOF(1) DO readf,1,a,b,c ;luetaan laitenumeroa 1 vastaavaa
tiedostoa jos ei olla sen lopussa eli EOF(1) on false

```

GOTO-hyppy:

```

GOTO,jump1
...
jump1: print,'hypättiin kohtaan jump1'

```

5 Erilaisia grafiikka-datan esitysrutiineja

5.1 Ikkunat

WINDOW, WSET, WDELETE

IDL avaa ensimmäisen grafiikkakomennon yhteydessä ikkunan grafiikan tulostamista varten, ellei sellaista ole jo avattu. Yhtäaikaan voi olla käytössä jopa 128 ikkunaa (riippuu koneesta ja versiosta) joihin tulostuksen voi ohjata.

<i>WINDOW</i>	luo 640*512 kokoisen ikkunan: window-index=0
<i>WINDOW,10</i>	luo 640*512 kokoisen ikkunan: window-index=10
<i>WINDOW,/FREE</i>	luo 640*512 kokoisen ikkunan: window-index on pienin vapaa ikkunnumero

*WINDOW, XSIZE=600, YSIZE=400, XPOS=50, YPOS=50, /FREE, \$
title='IDL', RETAIN=2*

Hyödyllisiä avainsanoja:

<i>XSIZE, YSIZE</i>	ikkunan koko (pikseleissä)
<i>XPOS, YPOS</i>	ikkunan sijainti (pikseleissä) ruudun oikeasta alanurkasta
<i>TITLE</i>	ikkunan otsikko (muutoin muotoa IDL_n, jossa n=window-index)
<i>PIXMAP</i>	ikkuna on näkymätön osa kuvamuistissa
<i>RETAIN</i>	kertoo mikä pitää huolen ikkunan sisällöstä <i>RETAIN=2</i> -> IDL huolehtii itse
<i>WSET,100</i>	ikkuna 100 valitaan aktiiviseksi (plottaus menee siihen)
<i>WDELETE,100</i>	tuhoaa ikkunan (ilman argumenttia aktiivisen ikkunan)
<i>ERASE</i>	tyhjentää aktiivisen ikkunan sisällön

Huom: avainsanoja ei tarvitse kirjoittaa kokonaan, riittää että ne ovat yksikäsitteisesti määrättyjä (esim. 'XPOS=' sijasta 'XP=')

5.2 PLOT-komento

PLOT-komennolla voidaan tehdä erilaisia xy-tyyppisiä plottauksia. Ilman avainsanoja se luo yksinkertaisen kuvan, jossa datapisteet esitetään viivoilla yhdistettynä, ja esim. akselien arvoalueet valitaan automaattisesti pisteiden arvojen avulla.

<i>PLOT,x</i>	plottaa x(i):n i:n funktiona
<i>PLOT,x,y</i>	plottaa x(i), y(i) pisteparit

Tulosta voidaan modifioida lukuisten avainsanojen avulla. Esim.

<i>XRANGE, YRANGE</i>	asettaa plottausvälin: esim. <code>plot,x,y,xrange=[0,10]</code> oletus: likimain minimi -> maksimi
<i>XTITLE, YTITLE</i>	akselien otsikot: esim. <code>plot,x,y,xtitle='x-arvo'</code>
<i>TITLE</i>	kuvan otsikko
<i>XSTYLE, YSTYLE</i>	akselityyppi: esim. <code>xstyle=1</code> käytetään <code>xrange:n</code> määräämää väliä tarkasti <code>xstyle=4</code> akseleita ei piirretä <code>xstyle=8</code> ei piirretä yläpuolista akselia (voidaan yhdistää esim. <code>5=1+4</code>)
<i>CHARSIZE</i>	akselien merkintöjen koko (1=normaali)
<i>LINESTYLE</i>	viivatyypit 0 yhtenäinen (oletus) 1 pisteviiva 2 katkoviiva 3 katko-piste viiva 4 katko-piste-piste-piste 5 pitkä katkoviiva
<i>COLOR</i>	väri riippuu käytettävästä paletista (ks. myöhemmin) yleensä 0 = musta, 255 = valkoinen
<i>PSYM</i>	datapisteiden symboli Oletus <code>psym=0</code> -> yhdistetään pisteet viivoilla negatiiviset arvot: käytetään symboleja+viivoja positiiviset: vain symbolit 1 plus-merkki 2 tähti 3 piste 4 salmiakki 5 kolmio 6 neliö 7 x 8 käyttäjän määräämä 10 histogrammi-moodi

Akselien piirtoon liittyy erilaisia avainsanoja, joilla voi muuntaa esim. akselimerkkien väliä, tai antaa data-arvoista riippumattomat akselimerkinnät:

<i>XTICKS, YTICKS</i>	varsinaisten akselimerkkivälien lukumäärä 0 tai määrittelemätön -> IDL tekee automaattisesti 3-6 väliä
<i>XMINOR, YMINOR</i>	varsinaisten akselimerkkien väliin merkittävien pikkuvälien lukumäärä 0 (oletusarvo) -> IDL määrää automaattisesti n -> n-1 pikkumerkkiä
<i>TICKLEN</i>	ison akselimerkin koko suhteessa kuvaan 0.02 (oletus) 1 -> ruudukko negatiivinen (esim. -.02) -> ulospäin osoittavat merkit
<i>XTICKV, YTICKV</i>	data-arvot joille akselimerkit laitetaan mahdollistaa ei-lineaariset merkinnät esim. <i>XTICKV</i> =[0,1,2,4,8], <i>XTICKS</i> =4 Huom! muistettava asettaa myös <i>XTICKS</i>
<i>XTICKNAME, YTICKNAME</i>	merkkivakio-vektori, joka sisältää akselimerkkejä vastaavat symbolit (oletus=data-arvot) Huom! muistettava asettaa myös <i>XTICKS</i>

Esim. plotataan $-\pi$ -> π ja merkitään arvot kirjaimin

```
PLOT,x,sin(x),XTICKS=4,XTICKV=[-!pi,-!pi/2,0,!pi/2,!pi],$
XTICKNAME=['PII','PII/2','0','PII/2','PII']
```

Plot-komennon kutsumisen yhteydessä IDL päättää kuvan sijoittelusta, akselivälistä jne. Piirrettäessä samaan kuvaan uusia käyriä käytä Oplot-komentoa. HUOM: Oplot-komennossa ei voi olla akseleihin liittyviä keywordeja.

AXIS-komento

Tämän avulla voidaan määritellä akseleita, esim. erilaiset y-akseliarvot kuvan vasempaan ja oikeen kulmaan. Käytännössä tämä tapahtuu estämällä kyseisen akselin luonti PLOT-komennossa, esim. käyttämällä YSTYLE=4 arvoa, ja luomalla akseli AXIS-komennolla käyttäen erilaisia YRANGE,YTITLE määrittelyjä.

YAXIS 0 vasen akseli, 1 oikea akseli

XAXIS 0 alareunan akseli, 1 yläreunan

Akselin paikka voidaan myös määrätä antamalla (X,Y) piste jonka kautta akseli kulkee. Esim.

AXIS,10,0,/XAXIS pisteen 10,0 kautta kulkeva x-akseli

Muita PLOT-muunnelmia:

<i>PLOT_IO</i>	linear-log
<i>PLOT_OI</i>	log-linear
<i>PLOT_OO</i>	log-log
<i>PLOT,/POLAR</i>	polar-plot: argumentit r ja fii (radiaaneina)

Esim:

```
;spiraalin piirtäminen  
r=findgen(100)  
fii=r/5.  
PLOT,/POLAR,r,fii,XSTYLE=4,YSTYLE=4 plotataan ilman akseleita  
AXIS,0,0,XAXIS=0 piirretään x-akseli origon kautta  
AXIS,0,0,YAXIS=0 piirretään y-akseli origon kautta
```

Normaali PLOT-komento piirtää vektoreita. Yksittäisen pisteen piirtäminen:

```
PLOTS,x,y
```

5.3 Plottaukseen liittyvät systeemimuuttujat !P, !X, !Y

Nämä IDL-systeemimuuttujat sisältävät plottaukseen liittyviä oletusarvoja, ja esim. useimmat avainsanat vastaavat jonkin näiden rakenteiden sisältämän muuttujan arvon komentokohtaista muuttamista. Niitä voidaan myös suoraan muokata, jolloin muutetut oletusarvot ovat voimassa. Esim:

<i>!P.COLOR=2</i> & <i>plot,x</i>	vastaa <i>plot,x,color=2</i> komentoa, paitsi että värimäärittely jää pysyvästi voimaan
<i>!P.CHARSIZE=0.7</i>	pienentää plottaus-symbolia
<i>!X.RANGE=[0,1]</i>	asettaa x-akseliväliksi 0,1

Aluksi näistä ei tarvitse paljoakaan välittää, koska samat vaikutukset voi tehdä lisäämällä komentoon avainsanoja.

Poikkeuksena on !P.MULTI-systeemimuuttuja, jonka avulla voidaan sijoittaa useita kuvia samaan ikkunaan. Esim.

<i>!P.MULTI=[0,2,3]</i>	2 kuvaa vaakasuoraan, 3 pystysuoraan, aloitetaan kuvasta 0 = ylhäällä vasemmalla
<i>!P.MULTI=[1,3,1]</i>	3 vaakasuoraan, 1 pystysuoraan, aloitetaan kuvasta 1 = ylärivin keskimmäinen
<i>!P.MULTI=0</i>	takaisin oletusarvoon: yksi kuva/ikkuna

KUVAN MUOTO

Normaalisti IDL automaattisesti valitsee grafiikkakomennoilla tuotetun kuvan koon ja sijoittelun ikkunaan. Siten esim. oletuskokoiseen 640*512 ikkunaan tuleva kuva on tyypillisesti x-suunnassa venytetty suorakaide. IDL-sisältää joukon systeemi-muuttujia, joiden avulla voidaan tarkasti määrittää kuvan koko ja sijoittelu suhteessa ikkunaan: näihin ei kuitenkaan tarkemmin puututa tässä tiivistelmässä.

Usein halutaan tuottaa tarkasti neliömäinen kuva, jossa x ja y akseleilla on tarkasti sama skaala, esim. simulaatiotulostuksen yhteydessä. Tavanomaisessa 640*512 ikkunassa on mahdollista tuottaa likimain neliömäinen kuva käyttämällä PLOT-komenossa $POSITION=[X1,Y1,X2,Y2]$ avainsanaa, jossa X1,Y1 on akselien vasen alakulma, X2, Y2 oikea yläkulma (kuvaikkunan normalisoiduissa koordinaateissa):

$POSITION=[0.18,0.1,0.82,0.9]$

Tällöin kompensoidaan ikkunan muoto $640/512=1.25$ määrittelemällä kuvan normalisoitu x-leveys 0.64:ksi ja y-leveys 0.8:ksi: $0.8/0.64=1.25$.

Plot-komennossa voi avainsanalla $ASPECT=1$ varmistaa että x ja y skaalat ovat samat

5.4 Tekstien lisääminen kuviin

XYOUTS-komento

$XYOUTS,x,y,string$ kirjoittaa merkkijonon string,
x,y määrittelee merkkijonon vasemman alakulman

Oletuksena on, että X,Y on annettu data-koordinaateissa.

Esim. $x=indgen(10)$ & $y=2*x$ & $xyouts,5,95,'y=x:n\ neliö'$ laittaa tekstin kuvan vasempaan alareunaan.

Avainsanoja:

<i>SIZE</i>	tekstin koko (oletus = 1 -> normaali koko)
<i>COLOR</i>	väri
<i>ALIGNMENT</i>	0 (oletus) -> tekstin vasen reuna alkaa pisteestä x 1. -> tekstin oikea reuna päättyy pisteeseen x 0.5 -> teksti keskitetty pisteeseen x
<i>ORIENTATION</i>	tekstin orientaatio suhteessa vaakasuoraan (asteina) esim. orientation=45 -> nousee 45 asteen kulmassa
<i>DEVICE</i>	x,y ovat ns. device-koordinaateissa (ruutu-> pikseleissä)
<i>NORMAL</i>	x,y ovat kuvan normalisoiduissa koordinaateissa 0,0 = kuvan vasen alalaita: 1,1 oikea ylälaita
<i>DATA</i>	x,y data-koordinaateissa (oletus)

TEKSTI-FONTIT

IDL grafiikka käyttää normaalisti ns. vektori-fontteja merkkien tuottamisiin: tekstit 'piirretään'. Tällöin tuotettava grafiikka on laite-riippumaton (esim. näyttö ja paperitulostus varmasti samoja). Käytettävissä on useita ns. Hershey-fontteja, mm. normaali ROMAN-fontti, erilaiset kreikkalaiset merkit, jopa kyriiliset kirjaimet. Samoin voidaan määritellä ylä- ja alaindeksejä.

Haluttujen tekstien tuottamiseksi varsinaisiin merkkijonojen lisätään !-merkillä alkavia fontti-määrittelyjä ja tekstin sijoituskomentoja. Tekstin sijoitteluun vaikuttavia komentoja ovat mm.:

<i>!N</i>	palaa normaalitasolle	(ja normaaliin merkkikokoon)
<i>!E</i>	siirry eksponentti-tasolle	(ja pienennä kokoa 0.44)
<i>!I</i>	siirry indeksitasolle	(ja pienennä kokoa 0.44)
<i>!D</i>	siirry alas ensimmäiselle indeksitasolle	(ja pienennä 0.62)
<i>!L</i>	siirry alas toiselle indeksitasolle	(ja pienennä 0.62)
<i>!U</i>	siirry ylös indeksitasolle	(ja pienennä 0.62)
Esim. ' <i>x!li!E2!N</i> ' tulostuu muodossa x_i^2		

Fonttimäärittelyt ovat muotoa !n, jossa n on fontin numero. Esim. !3 on normaali Roman-font, !7 antaa kreikkalaiset symbolit.

Esim.

<i>!7a</i> α	<i>!7h</i> Δ	<i>!7p</i> π	<i>!7v</i> χ
<i>!7b</i> β	<i>!7j</i> κ	<i>!7q</i> ρ	<i>!7w</i> Ψ
<i>!7c</i> γ	<i>!7k</i> λ	<i>!7r</i> σ	<i>!7x</i> ω
<i>!7d</i> δ	<i>!7l</i> μ	<i>!7s</i> τ	<i>!9A</i> \sim
<i>!7e</i> ϵ	<i>!7m</i> ν	<i>!7u</i> ϕ	<i>!9R</i> \surd

Näitä samoja määrittelyjä voidaan käyttää kaikessa grafiikka-tekstissä: PLOT-komennon akselien merkinnöissä (XTITLE, YTITLE, TITLE), XYOUTS-komennolla tulostettavissa teksteissä ja CONTOUR-komennon teksteissä. Huom! PRINT-komento ei tulosta grafiikkatekstiä, vaan käyttää IDL:n teksti-ikkunaa.

5.5 Kaksiulotteisten taulukoiden piirtäminen

CONTOUR-komento

Tuottaa kaksiulotteisen taulukon tasa-arvokäyrän kuvan, jonka oletuksena on 6 likimain tasavälein laskettua käyrää

<i>CONTOUR,Z</i>	2-ul. taulukko z,x ja y akselit merkitään z:n indeksien mukaan
<i>CONTOUR,X,Y,Z</i>	X,Y taulukot kertovat Z:n alkioden koordinaatit
	X,Y vektoreita -> Z(i,j):koordinaatti (X(i),Y(j))
	X,Y taulukoita -> Z(i,j):n koordinaatit (X(i,j),Y(i,j))

Avainsanat:

<i>LEVELS</i>	vektori joka sisältää plotattavat contour-arvot esim. <i>LEVELS</i> =[10,20,30]
<i>NLEVELS</i>	tasavälein laskettavien contour-käyrien määrä
<i>C_ANNOTATION</i>	vektori joka sisältää contour-käyrien merkinnät, yleensä contour-käyrän arvo
<i>C_CHARSIZE</i>	merkkikoko, jota käytetään contour-käyrien arvojen ilmaisemiseen (oletus 3/4 akselimerkkien koosta)
<i>C_COLORS</i>	vektori joka sisältää contour-käyrien värit. Käytetään syklisesti: esim. jos 10 contour arvoa ja <i>C_COLORS</i> =[1,2,3], niin käyrät 1,4,7,10 piirretään värillä 1, 2,5,8 värillä 2, 3,6,9 värillä 3.
<i>C_LABELS</i>	vektori joka kertoo mitkä contour-arvot merkitään: esim. 5 contour-arvoa, <i>C_LABELS</i> =[1,0,1,0,1] -> merkitään 1,3 ja 5.
<i>C_LINESTYLE</i>	vektori joka sisältää viivatyypit kullekin contour käyrälle. Käytetään syklisesti kuten <i>C_COLORS</i> :a.
<i>C_THICK</i>	vektori joka sisältää viivanpaksuudet. Käytetään syklisesti.
<i>SPLINE</i>	contour-käyrät tasoitetaan ennen piirtämistä (pienet taulukot!).
<i>FILL=1</i>	contour-käyrien värit täytetään

Lisäksi contour-kuvan akselien merkintöjä yms. voidaan muokata samoilla avainsanoilla mitä käytetään PLOT-komennon yhteydessä.

```
*****  
;  
;idlesim_ contour.pro  
;esimerkki väreillä täytetystä contour-kuvasta  
*****  
  
x=findgen(50)/50.*10  
data=cos(x)#(sin(x)*exp(-.25*x))  
  
clev=[-0.1, 0., 0.1, 0.3, 0.5, 0.7, 0.9] ;contour-arvot  
ccol=[ 2, 3, 4, 5, 6, 7] ;vastaavat värit  
clab=[1, 1, 1, 1, 0, 0, 1] ;label vai ei  
  
;HUOM: ensimmäisen contour arvon alittavat arvot  
;piirretään varilla 0 (musta ruudulla)  
  
tek_color ;perusvari-paletti  
contour,data,x,x,xrange=[0,10],yrange=[0,10],xstyle=1,ystyle=1,$  
levels=clev,c_label=clab,c_col=ccol,fill=1  
  
;piirretään paalle contour-kayrat valkoisella + arvot  
contour,data,x,x,xrange=[0,10],yrange=[0,10],xstyle=1,ystyle=1,$  
levels=clev,c_label=clab,col=1,chars=2,/over  
end  
*****  
;
```

SURFACE-komento:

Luo rautalanka-kuvion kaksi-ulotteisesta taulukosta (takana olevat viivat poistettu), käyttäen pitkälti samoja avainsanoja kuin esim. *contour*-komento. Lisäksi on mahdollista käyttää seuraavia lisä-avainsanoja:

<i>AX</i>	pintaa käännetään tämän kulman verran x-akselin suhteen katsojaan päin (asteina). Oletuarvo 30 astetta.
<i>AZ</i>	pintaa käännetään tämän kulman verran z-akselin suhteen, vastapäivään katsottaessa pitkin z-akselia origoa kohti. Oletusarvo 30 astetta.
<i>BOTTOM</i>	väri-indeksi joilla piirretään pinnan alapuoli. Oletus=sama väri-indeksi kuin yläpuolelle.
<i>UPPER_ONLY</i>	piirretään vain pinnan yläpuoli
<i>LOWER_ONLY</i>	piirretään vain alapuoli
<i>HORIZONTAL</i>	piirretään vain horisontaaliset viivat. Oletus: molempiin suuntiin.
<i>SKIRT</i>	piirretään reunat annetulla z-arvolla (<i>SKIRT</i> =z-arvo).

SHADE_SURF-komento:

Piirretään varjostettu kuva kaksi-ulotteisesta taulukosta. Avainsanat samat kuin **SURFACE**-komennossa.

SHOW3D-komento: IDL:n User Library -rutiini, joka yhdistää 2-ulotteisen taulukon esittämisen kuvana, *contour*-kuvana ja pintana.

5.6 Grafiikan tulostaminen printterille

Komennolla **SET_PLOT**, *device_name* määrätään mille laitteelle IDL:n grafiikatulostus ohjautuu. Käynnistettäessä oletuksena on 'X' eli grafiikka ohjautuu X-window grafiikkaikkunaan. Tulostettaessa PS-printterille annetaan komennot:

<i>SET_PLOT, 'PS'</i>	määritellään laite PS eli postscript
<i>DEVICE, FILE='psfile'</i>	ohjataan tiedostoon psfile
...plottaus-komennot	
<i>DEVICE, /CLOSE_FILE</i>	suljetaan tiedosto

\$lpr psfile unix-komento joka tulostaa tiedoston psfile oletusprintterille

DEVICE-komennolle voidaan antaa erilaisia avainsanoja, jotka vaikuttavat tulostettavan kuvan kokoon, sijoitteluun paperille, fontteihin, väri/mustavalko-tulostukseen, bittikarttojen tulostustarkkuuteen jne.

Käytännössä edellä olevat komennot on helpompi sijoittaa omiin proseduuritiedostoihin. Kopioi seuraavat tiedostot omaan hakemistoosi, komennolla *\$cp ~hsalo/IDL_ESIM/ps*.pro* . Tällöin saat käyttöösi proseduurit **PSOPEN**, **PSCLOSE** ja **PSDUMP**, ja voit yksinkertaistaa tulostusta muotoon

<i>PSOPEN, 'file'</i>	;ohjaa tulostuksen tiedostoon file (landscape-moodi)
...plot-komennot...	
<i>PSCLOSE</i>	;sulkee tiedoston
<i>\$lpr file</i>	;printtaa tiedoston

Proseduurissa on avainsanat COLOR ja VFONT:

-mikäli printtaat väritulostimelle käytä COLOR avainsanaa
 -VFONT taas ilmoittaa että käytetään vektorigrafiikkaa, eikä printterin hardware-fontteja, jolloin ruudulla näkyvät symbolit tulevat paperille varmasti samanlaisina. Mikäli kuvassa ei ole erikoisia symboleja, jätä /VFONT pois, sillä hardware-fontit antavat useimmiten paremman näköisen lopputuloksen.

-Edellä olevassa tulostustavassa postscript-tiedostoon tallentuvat kaikki piirtokäskyt: koska paperitulostinlaitteiden resoluutio (yleensä 300 dpi eli 300 pistettä/tuuma) on huomattavasti suurempi kuin X-päätteen resoluutio (n. 60 merkkiä/tuuma), lopputulos on paremman näköinen kuin ruudulla. Varjopuolena on postscript-tulostuksen hitaus, esim. tulostettaessa kuvia jotka sisältävät suuren määrän yksittäisiä pisteitä (luo esim. 1000 pistettä ja tulosta).

-Haluttaessa tulostaa nopeasti mutta huonommalla resoluutiolla, voidaan ruudulla oleva kuva tallentaa bitti-karttana postscript tiedostoon (TVRD-komennolla, palataan myöhemmin) jolloin sen tarkkuus tulee olemaan täsmälleen samaa kuin ruudulla, mutta tulostus on paljon nopeampaa:

<i>PSDUMP, file</i>	tallentaa aktiivisen ruudun ja 'dumpkaa' sen tiedostoon file
	luo myös väliaikaisen ikkunan jossa näkyy dumpattu ruutu
<i>\$lpr file</i>	unix-komento joka tulostaa oletusprintterille
<i>PSDUMP</i>	dumpkaa tiedostoon junk.ps ja tulostaa samantien

Tässäkin komennossa on runsaasti avainsanoja, mm. värikuvan tulostukseen jne. Kokeile /INV, /TEK -avainsanoja

Muista hävittää PS-tiedostot kun et enää tarvitse niitä (voivat olla yllättävän kookkaita).

6 Kuvanäyttö-rutiinit

Viivagrafikan lisäksi IDL sisältää runsaasti apuvälineitä varsinaiseen kuvien muokkaamiseen ja kuvankäsittelyyn. Kuvien esittäminen tapahtuu erilaisten **TV**-komentojen avulla, ja käsittelyyn on runsaasti erilaisia IDL-komentoja ja kirjastorutiineja.

Keskitytään seuraavassa harmaasävykuvien käsittelyyn (= normaali muoto, jossa tähtitieteellisiä kuvia analysoidaan) ja käsitellään sitten erikseen 3-kanavaisten värikuvien näyttäminen.

Seuraavalla komennolla saat luettua työtilaan Cassini-luotaimen Saturnuksesta ot-
taman kuvan:

```
restore, '~hsalo/IDL_ESIM.dir/saturn.save', /verbose
```

Sisältää 5 kuvaa:

```
Saturn = (3,1024,1024) rgb värikuva:3 kanavaa,kukin 1024*1024 pikseliä
red= eo. kuvan ensimmäinen kanava
green =eo. kuvan toinen kanava
red=eo. kuvan kolmas kanava
image=blue-kuvasta tehty 512*512 versio
```

Huom! koska kuvat ovat IDL:n kannalta tavanomaisia kaksiulotteisia taulukoita, joi-
hin pikseliarvot on talletettu, niiden käsittelyssä voidaan käyttää kaikkia normaaleja
matemaattisia- ja taulukko-operaatioita.

Erityisesti, kuvien kokoa voidaan muuttaa REBIN ja CONGRID-komennoilla, jotka
'venyttävät' taulukoita

```
ima512=REBIN(blue,512,512)          vanhan dimension oltava uuden monikerta
image256=REBIN(blue,256,256)
ima800=CONGRID(blue,800,800,interpol=1  mielivaltainen koko
```

6.1 TV-komennot

TV,image esittää 2-ulotteisen taulukon image kuvana. Kuvan
pikseliarvot vastaavat suoraan intensiteettiarvoja. Mikäli
pikseliarvot ylittävät ruudun maksimi-intensiteetin,
intensiteettejä käytetään syklisesti.

TVSCL,image skaalaa kuvan pikseliarvot näytön intensiteettiskaalalle
ennen esittämistä: min(image) arvoa vastaa intensiteetti 0,
max(image) -> maksimi-intensiteetti
Yleensä intensiteettiskaala on 0-255 (8-bittiiä)

TV-komentoihin voi antaa myös parametrejä, jotka kertovat kuvan sijoittelusta:

```
TV,image,x0,y0    kuvan vasen alakulma pikselissä x0,y0
TV,image,position position määrää sijainnin, esim.      0 1
                                                         2 3
```

Esim. oletetaan että image on 256*256 taulukko:

```
WINDOW,/FREE,XSIZE=512,YSIZE=512,/RETAIN  avaa 512*512 ikkunan

TV,image,0          tulostaa vasempaan yläkulmaan (0)
TVSCL,image,1       oikeaan yläkulmaan (position=1)
TVSCL,image<500,2   vasen alakulma (2)
TV,SQRT(image>0),3  oikea alakulma (3)
TV,image,50,60      tulostaa alkaen kohdasta 50,60
```

Kuvan intensiteettien skaalaus tulostuksessa:

-Kuvan kontrastin parantamiseksi voidaan kuvaa skaalata siten että haluttu pikseliarvo-väli tulee edustetuksi käyttäen koko ruudun intensiteettiskaalaa. Voidaan tehdä eri tavoin

1. tapa: rajoitetaan kuvan pikseliarvot välille $pmin$, $pmax$; sen jälkeen skaalataan ruudun intensiteettiminimin ja maksimin välille. Kuvan image sisältö ei muutu.

$TVSCL, image > pmin < pmax$

2. tapa: nopeampi tapa päästä samaan lopputulokseen käyttää BYTSCL-funktiota, joka skaalaa argumenttinsa välille 0-255. Ilman MIN ja MAX avainsanoja funktio etsii taulukon minimin ja maksimin ja käyttää niitä. Tämän jälkeen tulostetaan suoraan pikseliarvot. Kuvan image sisältö ei muutu.

$TV, BYTSCL(image, MIN=pmin, MAX=pmax)$

HIST_EQUAL-funktio:

Modifioi kuvan pikseliarvoja siten, että siinä esiintyy kutakin arvoa yhtä paljon. Käytännössä levittää runsaasti esiintyviä pikseliarvoja vastaavaa intensiteettialuetta.

$TV, HIST_EQUAL(image)$

Kuvien lukeminen ruudulta:

TVRD-proseduurilla voidaan lukea intensiteettiarvot ruudulla olevasta suorakulmaisesta alueesta IDL-tilukoon:

$image = TVRD(x0, y0, NX, NY)$ $x0, y0$ = luettavan alueen aloituskohta
(pikseleissä kuvan vasemmasta alareunasta).
 NX, NY = luettavan alueen koko x- ja y-suunnassa

Esim.

$image2 = tvrd(50, 50, 200, 200)$ lue 200*200 taulukko ruudulta, alkaen pikselistä 50,50
 $tv, 255-image2, 50, 50$ tulostetaan samaan kohtaan negatiivisena

6.2 Väripaletit

Yleisin tapa värien esittämisessä on ns. RGB-systeemi, jossa kukin väri luodaan kombinoimalla punaista, vihreää ja sinistä väriä (red, green, blue). Kullekin värikomponentille voidaan erikseen antaa intensiteettiarvoja välillä 0-255, ja väri voidaan esittää 3:n alkion vektorina [r,g,b] joka ilmoittaa kunkin komponentin intensiteetin. Mustaa väriä vastaa [0,0,0], valkoista [255,255,255], kirkkaan punaista [255,0,0] jne. Näin saadaan yhteensä 2^{24} eli 16 777 216 erilaista väriä.

Koska 8-bittinen näyttö pystyy kuitenkin esittämään vain 2^8 erilaista väriä tai harmaasävyä samanaikaisesti, joudutaan käyttämään ns. väritaulukkoa (color translation table), joka ilmoittaa mitkä näistä mahdollisista väreistä on valittu esitettäväksi 256 värin näytöllä. Väritaulukko C koostuu siten kolmesta 256-alkion vektorista, jotka ilmoittavat kutakin väri-indeksin arvoa vastaavat RGB-arvot.

Komennolla **TVLCT** voidaan muuttaa väritaulukkoa:

TVLCT,red,green,blue red,green,blue ovat kukin 256 alkion taulukoita
joiden arvot sijoitetaan väritaulukon kutakin
indeksiä vastaaviksi RGB-arvoiksi

Esim.

a=indgen(256) luo vektorin 0,1,...,255
tvlct,a,a,a väritaulukon i:n tripletin R,G,B-arvoiksi tulee
kullekin i -> harmaasävyskala (oletus IDL:n käynnistyessä)
*tvlct,a,a*0,a*0* lataa punaisen eri kirkkaussävyt (G(i) ja B(i) kaikki nolliä)
*tvlct,a,2*a,-a* kokeile muitakin

Esim. luodaan väripaletti, jonka ensimmäiset värit ovat musta, valkoinen, punainen, vihreä, keltainen (vastaa tektronix-paletin alkua)

red=[0,1,1,0,0,1]
green=[0,1,0,1,0,1]
blue=[0,1,0,0,1,0]
*tvlct,255*red,255*green,255*blue*

Valmiiden väritaulukoiden käyttöönottoon on IDL:n proseduur **LOADCT**, jolle voidaan parametrinä antaa jokin luvuista 0-15. Ilman parametria antaa tietoa eri paleteista:

LOADCT,0 harmaasävy
LOADCT,3 red-temperature
LOADCT,6 prism
LOADCT

Lisäksi viivagrafiikan tulostuksessa on hyödyllinen ns. tektronix-paletti jonka saa aikaan komennolla **TEK_COLOR** (*LOADCT,0*:lla pääsee takaisin oletuspalettiin). Se lataa väritaulukon alkuun mm. päävärit.

Jo ladattua väripalettia voi muunnella **STRETCH**-komennolla, joka lineaarisesti levittää nykyisen väripaletin peittämään annetun pikseliarvo-välin.

STRETCH,100,200 pikseliarvo 100 esitetään väri-indeksillä 0 ja
pikseliarvo 200 väri-indeksillä !d.colors-1.
Näiden välillä lineaarinen interpolaatio.

TV,image
STRETCH palauttaa alkuperäisen paletin

Lisäksi IDL:ssä on suuri määrä muita proseduureja palettien interaktiiviseen modifioimiseen, samoin kuin muiden värisysteemien (HSV ja HLS) käyttöön. Näistä löytyy tietoa esim. User's Guide.

Systeemin varaamat värit:

Yleensä käyttöjärjestelmä varaa tietyn määrän värejä omiin tarkoituksiinsa (ikkunoiden ja ikonien esittäminen) jolloin käytettävissä ei ole aivan 255 väriä. IDL:n systeemimuuttuja **!d.colors** sisältää IDL:n käytössä olevien värien määrän. NY-KYISSÄ 24-bittisissä näytöissä ei suurikaan ongelma!

Väripaletin esittäminen:

Hyödyllisen proseduurin kulloisenkin voimassa olevan väripaletin näyttämiseen saat kopioimalla proseduurin **colortable.pro** (*\$cp ~hsalo/IDL_ESIM.dir./colortable.pro*). Tämän jälkeen, kirjoittamalla *COLORTABLE*, saat näkyviin ikkunan jossa on esitettynä kaikkien 256 väri-indeksiarvoa vastaavat värit. Avainsanalla */TEK* saa 32 ensimmäistä väriä.

Pikseliarvojen lukeminen ruudulta:

Interaktiivisen kursorin sijainti ikkunassa voidaan lukea seuraavalla komennolla, joka odottaa hiiren painallusta ja palauttaa koordinaatit. Komennolla voidaan myös testata mitä hiiren nappulaa on painettu jne.

```
TV,image
CURSOR,x,y,/DEVICE  DEVICE-avainsana -> pikselikoordinaatit
print,image(x,y)      tulostaa pikseli-arvon
```

Voidaan käyttää myös interaktiivista **RDPIX** -kirjastoproseduuria.

RGB-kuvien näyttäminen 24-bittisellä näytöllä:

```
TVSCL,/true,image
```

7 INPUT/OUTPUT-rutiinit

IDL pystyy lukemaan ja kirjoittamaan monentyyppisiä tiedostoja, esim. FORTRAN ja C-kielisten ohjelmien tulostusta. Siinä on myös käytettävissä monipuoliset tulostuksen formatointikomennot. Seuraavassa rajoitutaan vain perusasioihin.

7.1 Tiedoston avaus ja sulkeminen

Ennen lukemista tiedosto on avattava:

<i>OPENR</i> ,1,'datain.dat'	Avaa tiedoston datain.dat laitenumeralle 1 lukemista varten (ei voi kirjoittaa).
<i>OPENW</i> ,2,'dataout.dat'	Avaa tiedoston dataout.dat laitenumeralle 2 kirjoittamista varten. Mahdollinen sisältö tuhoutuu.
<i>OPENU</i> ,3,'data.dat'	Avaa tiedoston lukemista ja kirjoittamista varten
<i>CLOSE</i> ,1	Tiedosto suljetaan (Sulkee laitenumeron 1).

7.2 Tiedostosta lukeminen ja tiedostoon kirjoittaminen (FORMATOITU)

Normaalin formatoidun tekstitiedoston käsittelyssä käytetään komentoja, jotka vastaavat IDL-teksti-ikkunasta lukua ja kirjoitusta (READ, PRINT)

READF, lun, muuttuja1, muuttuja2,... lukee laitenumeralta lun
PRINTF, lun, muuttuja1, muuttuja2,... kirjoittaa laitenumeralle lun

Esim.1: yksinkertaisen skalaaridatan lukeminen
oletetaan että tiedosto testi1 sisältää rivit
10 20 30
30 60 70

ne voidaan lukea komennolla

```
OPENR,1,'testi1'  
readf,1,a,b,c      ;a=10 b=20 c=30  
readf,1,d,e,f      ;d=30 e=60 f=70  
close,1
```

Esim.2: Taulukkoalkioiden lukeminen: useita alkioita eri taulukoista samalla rivillä

Oletetaan että on luotu tiedosto testi2 joka sisältää kullakin rivillä samantyyppisiä muuttujia, esim. x,y,z-arvoja muodossa:

```
x1,y1,z2  
x2,y2,z3  
...  
xn,yn,zn
```

Tällöin luku voidaan suorittaa useallakin eri tavalla riippuen esim. siitä tiedetäänkö pisteiden lukumäärä.

Tapa 1: ei tiedetä rivien lukumäärää, luetaan rivi kerrallaan

```
OPENR,1,'testi2'  
x=FLTARR(100) & y=x & z=x      ;arvellaan ettei pisteitä ole ainakaan 100 enempää  
lask=0  
WHILE NOT EOF(1) DO BEGIN      ;jatketaan jos ei ole kohdattu tiedoston loppumerkkiä  
  READF,1,dum1,dum2,dum3      ;luetaan skalaareina,  
  x(lask)=dum1                ;sillä vektorin alkioita ei voi suoraan lukea  
  y(lask)=dum2  
  z(lask)=dum3  
  lask=lask+1  
ENDWHILE  
x=x(0:lask-1)                  ;siivotaan tyhjät alkiot  
y=y(0:lask-1)  
z=z(0:lask-1)  
close,1
```

Tapa 2: tiedetään rivien määrä, luetaan yhtenä taulukkona

```
OPENR,1,'testi2'  
N=50 ;oletetaan että 50 riviä  
TEMP=ftarr(3,N) ;luodaan väliaikainen 3*50 taulukko  
READF,1,TEMP  
x=temp(0,*) ;poimitaan muuttujat  
y=temp(1,*)  
z=temp(2,*)  
close,1
```

Esim.3: peräkkäin talletettuja skalaareja, taulukoita

Oletetaan, että on tiedosto 'testi3', joka sisältää eo. x,y,z taulukot seuraavassa muodossa

```
50      alkioiden määrä/taulukko  
x1,x2,...,xn  x-arvot kokonaisuudessaan  
y1,y2,...,yn  
z1,z2,...,zn
```

luku voidaan tehdä seuraavasti:

```
OPENR,1,'testi3'  
READF,1,N      luetaan alkioiden määrä/taulukko  
x=FLTARR(N) & y=x & z=x  
READF,1,x      luetaan koko x-taulukko  
READF,1,y  
READF,1,z  
close,1
```

Huom! luku on aina paljon nopeampaa jos se voidaan tehdä taulukko kerrallaan.

Esim.4: Tiedoston riveillä eri tyyppisiä suureita

Oletetaan, että on tiedosto testi4 joka sisältää seuraavanlaisia rivejä, eli lukuja ja tekstimuuttujia

```
x1,y1,z2 HUONO  
x2,y2,z3 HYVA  
...  
xn,yn,zn HUONO
```

Tällöin tiedostoa on luettava rivi kerrallaan

```

OPENR,1,'testi4'
x=FLTARR(100) & x=y & z=x      ;arvellaan ettei pisteitä ole ainakaan 100 enempää
laatu=STRARR(100)              ;luodaan merkkitieto-taulukko

lask=0
dums=""                        ;määrittelee dums=merkkimuuttujan

WHILE NOT EOF(1) DO BEGIN      ;jatketaan jos ei ole kohdattu tiedoston loppumerkkiä
  READF,1,dum1,dum2,dum3,dums ;luetaan skalaareina,
  x(lask)=dum1                 ;sillä vektorin alkioita ei voida suoraan lukea
  y(lask)=dum2
  z(lask)=dum3
  laatu(lask)=dums
  lask=lask+1
ENDWHILE
x=x(0:lask-1)                  ;siivotaan tyhjät alkiot
y=y(0:lask-1)
z=z(0:lask-1)
laatu=laatu(0:lask-1)
close,1

```

7.3 FORMATOIMATTOMAN tiedoston lukeminen

Formatoimaton tiedosto on koneen sisäisellä esitystavalla tallennettu tiedosto, jota ei esim. pysty cat, more tms. tekstitiedosto-komennoilla järkevästi käsittelemään. Niitä käytetään usein esim. kuvien tallentamiseen niiden kompaktisuuden vuoksi. Voidaan luoda esim. FORTRAN-ohjelmilla.

-Formatoimattoman tiedoston avaus ja sulkeminen tapahtuu samoilla komennoilla kuin edellä.

-luku ja kirjoitus:

```

READU,lun,muuttuja1,muuttuja2,...
WRITEU,lun,muuttuja1,muuttuja2,...

```

7.4 MUOTOILULAUSEET

Käytettäessä print tai printf-komentoa tulostukseen ilman erillistä muotoilulausetta, IDL tulostaa esim. liukuluvut niiden koko tarkkuudella, mikä on usein tarpeettoman tilaa vievää. Tulostuksen muokkaamiseen voidaan käyttää format-avainsanaa, muodossa format=string, jossa string on muotoilukomennot sisältävä merkkijono. Muotoilukomennot ovat hyvin samankaltaisia kuin FORTRAN-kielessä esim.:

```

x=2.
i=1234
print,x                               tulostuu muodossa 2.00000
print,x,format='(F6.2)'              tulostuu muodossa 2.00

```

Erilaisia format-määrittelyjä

liukuluvut:	Fn.k	n=luvulle varattava tila, k=desimaalien määrä
	En.k	eksponentti-esitys (huomaa lisätilan tarve)
kokonaisluvut	In	n=luvulle varattava tila
merkkitieto	An	n=tulostettavien merkkien määrä
tyhjää	nX	n=tyhjien kohtien määrä
merkkivakio	nHteksti	tulostaa merkkivakion joka on H:n jäljessä, pituus=n merkkiä.

Esim.

```
print,x,sqrt(x),format='(17Hluku ja sen neliö,10x,2f12.3)'
```

7.5 SAVE ja RESTORE -komennot

Erittäin käyttökelpoinen muuttujien talletustapa on IDL:n sisäisellä formaatilla tallennetut save-tiedostot. Niihin voidaan tallentaa mielivaltaisessa järjestyksessä skalaareja, vektoreita, taulukoita jne., eikä minkäänlaisista muoto-määrittelyistä tarvitse välittää. Tiedostoa luettaessa ei myöskään tarvitse tietää sen sisältöä tai muuttujien kokoa, vaan ne palautetaan kaikki alkuperäisessä muodossaan ja nimettyinä.

Talletus:

```
save,file_name='tiedosto.save',var1,var2,var3,...
```

talletetaan muuttujat
var1,var2,... tiedostoon
nimeltä 'tiedosto.save'
(hyvä tapa lisätä .save määre)

Avainsanalla /variables taltioituu kaikki työtilassa sillä hetkellä olevat muuttujat. Jos tiedostonimeä ei anneta, oletuksena on 'idl.save'.

Lukeminen:

```
restore,'tiedosto.save'
```

luetaan kaikki tiedostossa 'tiedosto.save'
olevat muuttujat

Huom! Muuttujat palautetaan sen hetkiseen työtilaan, eli jos esim. proseduuri on lopunut virheeseen, keskeytyksen jälkeen jäädään joskus ohjelman työtilaan -> restore lukee muuttujat sinne, eikä varsinaisen työtilaan kuten luultavasti oli tarkoitus.

Myös aliohjelmat voivat käyttää restore-lausetta. Vanhoissa IDL versioissa voi tulla virheilmoitus 'undefined variable ...', jos save-tiedostossa olevia muuttujia ei ole aiemmin nimetty. Ilmoituksesta ei kuitenkaan tarvitse välittää, muuttujat tulevat oikein luetuiksi.

8 Help, ? ja doc_library

IDL:n help-komentoa käytetään tiedon saamiseen työtilassa kulloinkin olevista muuttujista, niiden kokomäärittelyistä ja sisällöstä, sekä esim. kulloinkin käännettyinä olevista proseduureista. Help,variable taas antaa enemmän tietoa yksittäisestä muuttujasta.

Erilaisista IDL-komennoista saa tietoa kirjoittamalla kysymysmerkki komennon eteen, esim. ?plot antaa tietoa plot-rutiinin avainsanoista. Pelkkä ? vie tilaan jossa voi valita tietoa erilaisista IDL:n sisältämistä komennoista yms.

Edellä olevissa esimerkeissä on käytetty myös useita käskyjä, joista osa ei ole IDL:n sisäisiä komentoja vaan proseduuri-tiedostojen nimiä (esim. tek_color, loadct). Tällaisen ohjelmätiedoston ja varsinaisen komennon kutsussa ei ole nimittäin eroa. IDL:aa varten onkin olemassa suuri määrä ns. käyttökirjastoja, jotka sisältävät eri käyttäjien, eri tarkoituksiin tehtyjä rutiineja. Näistä rutiineista saa informaatiota seuraavalla tavalla

doc_library, 'flat' antaa tietoa astro-kirjaston sisältämästä proseduurista
flat.pro

9 Ohjelmien optimointi

Periaatteessa IDL:lla on mahdollista päästä monissa numeerisissa laskuissa lähes samaan nopeuteen kuin esim. FORTRAN-ohjelmalla, huolimatta sen tulkkaavasta luonteesta. Tällöin täytyy kuitenkin ottaa huomioon seuraavat seikat:

- 1) Käytä vektoreita aina kun se on mahdollista, silmukkarakenteiden sijaan.
- 2) Käytä IDL:n omia systeemikomentoja ja funktioita jos mahdollista
- 3) Isoja taulukoita käytettäessä ota huomioon missä järjestyksessä se talletetaan muistiin.

Esim. 512*512 kuvan kääntäminen ylösalaisin

```
FOR i=0,511 DO BEGIN ;Tämä vaatii n. 143 sekuntia (vuoden 1992) sun3:ssa
FOR i=0,511 DO BEGIN
temp=image(i,j) ;talleta väliaikaisesti pikseliarvo
image(i,j)=image(i,511,-j)
image(i,511-j)=temp
ENDFOR
ENDFOR
```

Nopeampi:

```
FOR j=0,255 DO BEGIN           ;vaatii 11 sek  
sw=511-j  
temp=image(*,j)             ;talleta väliaikaisesti koko rivi  
image(0,j)=image(*,sw)  
image(0,sw)=temp  
ENDFOR
```

Vielä parempi:

```
image2=image(*,511-inden(512)) ;vaatii 6 sekuntia
```

Paras:

```
image=rotate(image,7)         ;ROTATE on valmis IDL-funktio 0.6 sekuntia
```