ATK TÄHTITIETEESSÄ: IDL:n perusteita

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 data 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 (nopeampi kuin esim. MATLAB?)

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. Lähinna käsitellään yksinkertaiset peruskomennot, joiden avulla pääsee alkuun kielen omakohtaisessa opettelussa manuaalien avulla. Tavoitteena on että kurssin jälkeen on valmiudet käyttää IDL:ää esim. tähtitieteen harjoitustöissä. IDL on käytettävissä mm. ATK-keskuksen haapa2 ja sun4-koneissa. Perusteellinen kuvaus IDL-paketista on manuaaleissa:

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

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 -al*. 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:

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

Esimerkeistä näkee, että IDL sisältää normaalit 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 evaluoidaan 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=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)

-Mikäli halutaan antaa yhtä riviä pitempi komento käytetään $ merkkiä rivin lopussa:

x=indgen(10) & y=sqrt(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-tulkit, päinvastoin kuin FORTRAN tai C).

3 IDL-muuttujista

3.1 Data-tyypit

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

Kokeile esim. antamalla

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

Huom! kokonaislukumuuttujien 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=byte(a)
b=fix(a)
b=long(a)
b=float(a)
b=double(a)
b=string(a)
b=complex(a)

3.2 Matemaattiset operaatiot

Suoritetaan normaalissa aritmeettisessa järjestyksessä:

eksponentti	^
kerto- ja jakolasku, modulo	*,/,mod
yhteen- ja vähennyslasku, minimum, maximum	+,-,<,>

*a^2*3*	a potenssiin 2, kerrotaan kolmella
*a^(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!
a=5
a/2=2
a/2.=2.5
a/2+1.=3.
a/2.+1=3.5

3.3 Matemaattisia funktioita

IDL sisältää normaalit matemaattiset funktiot, esim.

sin(x), cos(x), tan(x)	argumentit radiaaneina
asin(x), acos(x), atan(x)	
atan(y,x)	palauttaa kulman α, jolle y=sin(α) ja x=cos(α)
sinh(x), cosh(x), tanh(x)	
exp(x), sqrt(x), x^2, abs(x)	
alog(x), 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=indgen(5)	sama vaikutus: vektori, jonka alkion arvo=alkion indeksi
a=findgen(5)	reaalilukuvektori
a=lindgen(5)	long integer
a=dindgen(5)	
a=intarr(5)	määrittelee a:n kokonaislukuvektoriksi (alkiot=0)
a=fltarr(5)	
a=lonarr(5)	
a=dblarr5)	
a=bytarr(5)	
b=a	b:n tyyppi määritellään samaksi kuin a:n tyyppi ja lisäksi a:n arvot sijoitetaan b:n arvoiksi
a=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) nollaksi, 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 ge 4) index=[3,4]

data(where(data le 0)) = 0 muuttaa taulukon data negatiiviset arvot nolliksi (sama kuin data=data>0)

bad=where(data lt 0) korvataan negatiiviset alkiot vierekkäisten keskiarvolla

data(bad)=(data(bad-1)+data(bad+1))/2

Moniulotteiset taulukot

a=fltarr(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. proseduuri-tiedostot, jotka sisältävät useita IDL-komentoja ja laskutoimituksia. Ne voivat olla joko pääohjelma, aliohjelma tai funktio tyyppisiä.

Huom! Esimerkeissä olevat ohjelmat löytyvät haapa2:n hakemistosta /wrk2/rkarjala/IDL, josta ne voi kopioda itselleen copy komennolla.

mkdir idlesim tee itsellesi idlesim-hakemisto
cp /wrk2/rkarjala/IDL/ ./idlesim* tulevat nykyisen hakemiston alapuolella olevaan idlesim-hakemistoon

IDL:n sisällä 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,'säde,piiri,ala'
print,r,piiri,ala
end                ;proseduurin pitää päättyä end-riviin
```

ja poistutaan emacsista (ctrl -x -c:llä, vastataan y)

-tiedoston sisältö voidaan tarkistaa antamalla
$cat idlesim1.pro
-pääohjelma tiedosto suoritetaan komennolla
.run idlesim1
-IDL-proseduurin oletustarkennin on muotoa .pro, joten sitä ei tarvitse antaa mikäli tiedoston nimi on tätä muotoa (hyvä ohjelmointitapa)

Pääohjelma-tyyppinen proseduuri 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 tulkkaus 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 määriteltyjen argumenttien avulla. Aliohjelmalla on myös oma muuttuja-alueensa.

Aliohjelma-proseduuri määritellään sen alussa olevalla rivillä,

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

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

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 edella olevasta idlesim1.pro-pääohjelmasta idlesim2.pro-aliohjelma:

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

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 päättyä 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 voidaan käyttää 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 niitä on liian vähän.

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

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

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. atan-funktio).

4.3 Funktio-proseduurit

Kolmas proseduurityyppi on funktion-aliohjelmat. Ne määritellään rivillä

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

ja erona aliohjelmaan on, että ne palauttavat 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 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=fltarr(51,51)
FOR i=0,50 DO BEGIN
FOR j=0,50 DO BEGIN
x(i+j*51)=float(i)
y(i+j*51)=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
zz=exp(-0.125*(x^2+y^2))*sin(2*y)*cos(y)  ;zz vektori, zz=zz(x,y)
z(0:50,0:50)=zz                           ;muutetaan zz pinnaksi z (1-ulotteinen
; -> 2-ul.) sijoittamalla 2-ulotteiseen taulukkoon samankokoinen 1-ulotteinen vektori
```

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' ;sulkuja 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-hypyt
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. Yhtäaikaa voi olla käytössä jopa 128 ikkunaa (riippuu koneesta ja versiosta) joihin tulostuksen voi ohjata.

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

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

Hyödyllisiä avainsanoja:

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

5.2 PLOT-komento

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

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

Tulosta voidaan modifioida lukuisten avainsanojen avulla. Esim.

XRANGE,YRANGE	asettaa plottausvälin: esim. plot,x,y,xrange=[0,10]
	oletus: likimain minimi -> maksimi
XTITLE,YTITLE	akselien otsikot: esim. plot,x,y,xtitle='x-arvo'
TITLE	kuvan otsikko
XSTYLE,YSTYLE	akselityyppi: esim.
	xstyle=1 käytetään xrange:n määräämää väliä tarkasti
	xstyle=4 akseleita ei piirretä
	xstyle=8 ei piirretä yläpuolista akselia
	(voidaan yhdistää esim. 5=1+4)
CHARSIZE	akselien merkintöjen koko (1=normaali)
LINESTYLE	viivatyyppi
	0 yhtenäinen (oletus)
	1 pisteviiva
	2 katkoviiva
	3 katko-piste viiva
	4 katko-piste-piste-piste
	5 pitkä katkoviiva
COLOR	väri
	riippuu käytettävästa paletista (ks. myöhemmin)
	yleensä 0 = musta, 255 = valkoinen

PSYM	datapisteiden symboli Oletus psym=0 -> yhdistetään pisteet viivoilla negatiiviset arvot: käytetään symboleja+viivoja positiivieset: 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:

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

Esim. plotataan -π -> 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.

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:

PLOT_IO	linear-log
PLOT_OI	log-linear
PLOT_OO	log-log
PLOT/POLAR	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

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:

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

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

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

!P.MULTI=[0,2,3]	2 kuvaa vaakasuoraan, 3 pystysuoraan
!P.MULTI=[1,3,1]	3 vaakasuoraan, 1 pystysuoraan, aloitetaan kuvasta 1 = keskimmäinen
!P.MULTI=0	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 systeemimuuttujia, joiden avulla voidaan tarkasti määrittää kuvan koko ja sijoittelu suhteessa ikkunaan: näihin ei kuitenkaan tarkemmin puututa tässä tiivistelmässä.

Usein halutaan tuottaa tarkasti neliömäinen kuva, esim. simulaatiotulostuksen yhteydessä. Esim. 640*512 ikkunassa on mahdollista tuottaa likimain neliömäinen kuva käyttämällä PLOT-komenossa seuraavaa POSITION=[X1,Y1,X2,Y2] avain sanaa, 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.

Voit käyttää myös proseduuria *sbox* ($cp /wrk2/rkarjala/IDL/sbox.pro .)

SBOX,1	neliömäinen kuva (käyttää koko kuva-alaa)
SBOX,0.5	pieni neliömäinen kuva
SBOX,0	takaisin oletusmuotoon

5.4 Tekstien lisääminen kuviin

XYOUTS-komento

XYOUTS,x,y,string	kirjoittaa merkkijonon string, x,y määrittelee 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:

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

TEKSTI-FONTIT

IDL grafiikka käyttää normaalisti ns. vektori-fontteja merkkien tuottamisiin: tekstit 'piirretään'. Tällöin tuotettava grafiikka on laite-riippumatonta (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.:

!N	palaa normaalitasolle	(ja normaaliin merkkikokoon)
!E	siirry eksponentti-tasolle	(ja pienennä kokoa 0.44)
!I	siirry indeksitasolle	(ja pienennä kokoa 0.44)
!D	siirry alas ensimmäiselle indeksitasolle	(ja pienennä 0.62)
!L	siirry alas toiselle indeksitasolle	(ja pienennä 0.62)
!U	siirry ylös indeksitasolle	(ja pienennä 0.62)
Esim. *'x!li!E2!N'*	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.

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

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 Kaksi-ulotteisten taulukoiden tulostus

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

CONTOUR,Z	2-ul. taulukko z,x ja y akselit merkitään z:n indeksien mukaan
CONTOUR,X,Y,Z	X,Y taulukot kertovat Z:n alkioiden 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:

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

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

Contour-käyrien väliin jäävien alueiden värittäminen:

Käyttämällä *PATH_FILENAME='tiedosto_nimi'* avainsanaa voidaan contour-komennon tuottamat käyrät tallettaa tiedostoon piirtämisen sijaan. Tätä tiedostoa voidaan käyttää yhdessä POLYCONTOUR-proseduurin kanssa täyttämään halutuilla väreillä käyrien väliset alueet. Huom! POLYCONTOUR pystyy täyttämään vain suljettujen käyrien väliset alueet mutta tämä on helppo ottaa huomioon, allaolevan täydellisen piirtoesimerkin mukaisesti.

```
;*******************************************
;idlesim_polycontour.pro
:esimerkki väreillä täytetystä contour-kuvasta

x=findgen(50)/50.*10
data=cos(x)#(sin(x)*exp(-.25*x))          ;alkuperäinen data 50*50
data2=-10+0*fltarr(52,52)                 ;52*52 taulukko jonka
data2(1,1)=data                           ;keskelle data laitetaan
x2=[2*x(0)-x(1),x,2*x(49)-x(48)]          ;vastaavat x-arvot (52)

clev=[-0.1,0.,.1,.3,.5,.7,.9]             ;contour-arvot
cin=[0,2,3,4,5,6,7]                       ;vastaavat värit
clab=[1,1,1,1,0]                          ;label vai ei
```

```
;***** lasketaan contour-käyrät, ei piirretä mutta talletetaan tiedostoon junk
contour,data2,x2,x2,xrange=[-1,11],yrange=[-1,11],xstyle=1,ystyle=1,$
levels=clev,C_label=clab,PATH_filename='junk'

;***** piirretään alueet täytettynä väreillä
polycontour,'junk',color_index=cin

;***** piirretään contour-käyrät päälle. NOERASE -> ei pyyhi ennen piirtämistä.
contour,data2,x2,x2,xrange=[-1,11],yrange=[-1,11],xstyle=1,ystyle=1,$

levels=clev,C_label=clab,/noerase
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:

AX	pintaa käännetään tämän kulman verran x-akselin suhteen katsojaan päin (asteina). Oletuarvo 30 astetta.
AZ	pintaa käännetään tämän kulman verran z-akselin suhteen, vastapäivään katsottaessa pitkin z-akselia origoa kohti. Oletusarvo 30 astetta.
BOTTOM	väri-indeksi joilla piirretään pinnan alapuoli. Oletus=sama väri-indeksi kuin yläpuolelle.
UPPER_ONLY	piirretään väin pinnan yläpuoli
LOWER_ONLY	piirretään vain alapuoli
HORIZONTAL	piirretään vain horisontaaliset viivat. Oletus: molempiin suuntiin.
SKIRT	piirretään reunat annetulla z-arvolla (SKIRT=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 grafiikkatulostus ohjautuu. Käynnistettäessä oletuksena on 'X' eli grafiikka ohjautuu X-window grafiikkaikkunaan. Tulostettaessa PS-printterille annetaan komennot:

SET_PLOT,'PS'	määritellään laite PS eli postscript
DEVICE,FILE='psfile'	ohjataan tiedostoon psfile
...plottaus-komentoja	
DEVICE,/CLOSE_FILE	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 /wrk2/rkarjala/IDL/ps*.pro .* Tällöin saat käyttöösi proseduurit **PSOPEN, PSCLOSE** ja **PSDUMP**, ja voit yksinkertaistaa tulostusta muotoon

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

Proseduurissa on avainsanat COLOR ja VFONT:
-mikäli printtaat väritulostimelle käytä COLOR avainsanaa
-VFONT taas ilmoittaa että käytetään vektorigrafiikkaa, eikä printterin hardwarefontteja, 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:

PSDUMP,file	tallentaa aktiivisen ruudun ja 'dumppaa' sen tiedostoon file luo myös väliaikaisen ikkunan jossa näkyy dumpattu ruutu
$lpr file	unix-komento joka tulostaa oletusprintterille
PSDUMP	dumppaa 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

Viivagrafiikan 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.

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

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-bittinen näyttö)

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 argumenttinsä 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)

Kuvien lukeminen ruudulta:

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

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 muutella 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 -> harmaasävyskaala (oletus IDL:n käynnistyessä)
*tvlct,a,a*0,a*0*	lataa punaisen eri kirkkaussävyt (G(i) ja B(i) kaikki nollia)
*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 proseduuri **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.

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

Väripaletin esittäminen:
Hyödyllisen proseduurin kulloisenkin voimassa olevan väripaletin näyttämiseen saat kopioimalla proseduurin **COLORTABLE** (*$cp /wrk2/rkarjala/IDL/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.

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:

OPENR,1,'datain.dat'	Avaa tiedoston datain.dat laitenumerolle 1 lukemista varten (ei voi kirjoittaa).
OPENW,2,'dataout.dat'	Avaa tiedoston dataout.dat laitenumerolle 2 kirjoittamista varten. Mahdollinen sisältö tuhoutuu.
OPENU,3,'data.dat'	Avaa tiedoston lukemista ja kirjoittamista varten
CLOSE,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 laitenumerolta lun
PRINTF,lun,muuttuja1,muuttuja2,... kirjoittaa laitenumerolle 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 alkiota 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=fltarr(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! Muttujat palautetaan sen hetkiseen työtilaan, eli jos esim. proseduuri on loppunut virheeseen, keskeytyksen jälkeen jäädään joskus ohjelman työtilaan -> restore lukee muuttujat sinne, eikä varsinaisen työtilaan kuten luultavasti oli tarkoitus.

Myös aliohjelmat voivat käyttää restore-lausetta: kuitenkin mikäli save-tiedostossa olevia muuttujia ei ole aiemmin nimetty, tulee virheilmoitus 'undefined variable ...'. Siitä 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 ohjelmatiedoston 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 kirjasto-ohjelmasta flat

9 Ohjelmien optimointi

Periaatteessa IDL:lla on mahdollista päästä monissa numeerisissa laskuissa lähes samaan nopeuteen kuin esim. FORTRAN-ohjelmalla, huolimatta sen tulkkaavasta luonteesta. Tällöin taytyy 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 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
```