Nonlinear Spectral Mixture Effects for Photosynthetic/Non-photosynthetic Vegetation Cover Estimates of Typical Desert Vegetation in Western China Version 4

Cuicui JI

Abstract

Citation: Cuicui JI Nonlinear Spectral Mixture Effects for Photosynthetic/Non-photosynthetic Vegetation Cover Estimates of Typical Desert Vegetation in Western China. **protocols.io**
dx.doi.org/10.17504/protocols.io.ia5cag6
Published: 15 Dec 2017

Protocol

Spectral Data and Preprocessing

Step 1.

In order to remove the effects of endmember variability, the experimental setup was designed so that endmembers were allowed to vary among plots. For each of the experimental plots, plot-specific bare soil (BS), shadow, PV, and NPV endmembers were defined and used in further analyses. In this research, spectra data for each endmember and canopy mixed spectral were obtained from ground-based field experiments; this involved obtaining specific PV/NPV/BS/shadow endmember spectra in each sample plot so that we could remove the effect of variability among the endmember spectra . Reflectance spectral measurements were acquired on August 25, 2014 (a clear-sky day), within 1 h of local solar noon by using a full-range (350–2500 nm) spectroradiometer with a 25° ASD (Analytic Spectral Devices, Boulder, CO) Spec Pro Field spectrometer. The reflectance was calibrated by using a white spectral panel (Labsphere Inc., North Sutton, NH). During windless, cloudless, and full sunshine conditions, we collected data throughout a stable time period (10:00–14:00) from 20 *Nitraria* plots and 20 *Haloxylon* plots with different ratios of PV to NPV cover, and we measured the canopy spectra and the pure endmembers spectra. We used an orthogonal ruler to determine the center of the plots, and we marked out 1 m diameter circular area to ensure the sample range was consistent. Measurements were taken from nadir at a height of 2.3 m above the earth's surface, which resulted in a field of view (FOV) or a pixel/plot diameter of 1 m from which we obtained the mixed spectral data for the fields; meanwhile, we placed the probe above the various typical species endmember surfaces (e.g., *Nitraria*, *Haloxylon*, dry branches, fine sand, shadows) between 0.1 m and 0.02 m so that pure endmember spectra were acquired in each field. In this way, each mixed canopy pixel and the specific pure endmember spectra for the field sites were obtained.

Linear Spectral Mixture Model

Step 2.

We have constructed the Linear Spectral Mixture Model based on the principles of linear mixtures.the endmember fraction estimates are obtained by the fully constrained least squares (FCLS).

Reference Fraction

Step 3.

In this study, photographs were acquired two times by using a digital camera at each field site, in order to avoid out of focus. The photograph with higher quality will be selected, so that the photograph could be used to get the reference fraction accurately. Information on the ground cover composition of each of the measured mixed pixels was extracted from the digital photographs (positioned at nadir) so that the ground cover fraction distribution could be determined. Two cross rulers were used to mark the FOV while the digital image sensor was used to obtain RGB (red, green, and blue) photographs. According to the training samples for supervised classification, neural network classification (NNC) was applied to classify the digital photos with ENVI 5.3 software[26], and this yielded the 3-EM classes and 4-EM classes . The classification results were validated through visual interpretations. The observed endmember cover would not contribute equally to the ASD probe due to the point spread function (PSF) of the ASD sensor. Hence, it was necessary to calibrate the fractional cover.

The PSF is the Gaussian function, , where *d* is the distance from a pixel center (the unit is in meters) and σ illustrates the instantaneous FOV (IFOV) of a detector, which means that measurements closer to the center of the FOV will contribute more to the mixed reflectance signal, and thus, more weight should be given to the objects in the center. Each binary classified image was convolved with a Gaussian filter (i.e., *PSF* of the ASD sensor) to compute weighted averages of the image. Endmember reference fractions could as such be expressed as a function of their actual contribution to the mixed pixel reflectance. Here, all the actual ground cover fractions were corrected to correspond to the PSF of the ASD sensor. In summary, all the classification images were convolved with a Gaussian filter, and then, the weighted average adjusted fractions of two photographs were taken as the reference fraction for the different endmembers in a plot.

Bilinear spectral mixture model

Step 4.

this study proposes the use of a bilinear spectral mixture model (BSMM). Each term accounts for multiple interactions between endmembers and is represented by the cross-product of the interacting endmembers. BSMMs consider multi-order interactions between endmembers j and t (for j,t=1,...,m). The FCLS algorithm is employed to unmixing fj,t and fj. Part of the interaction fraction fj,t should be assigned to each of the contributing physical entities.

Kernel Nonlinear Spectral Mixture Model

Step 5.

We used two common kernel functions, namely, the radial basis function (RBF) kernel and the polynomial kernel function (PKF).

Parameters of kernel function

Step 6.

The optimal parameters in the RBF and PKF are determined by the minimum model unmixing RMSE.The parameter σ in the RBF is determined by the gradient descent method,and the parameters of PKF are determined by the Cross validation ,which means that the fitting process optimizes the model parameters to make the model fit the input data as well as possible.

Kernel Nonlinear Spectral Mixture Model unmixing

Step 7.

The KFCLS aims to find the abundance vectors via the objective function. A detailed step-by-step implementation of KFCLS is available..

Evaluate unmixing Accuracy

Step 8.

In cases where accurate ground reference data are available, the quality of the sub-pixel abundance estimates can be assessed more reliably by checking the discrepancy between the estimated and reference endmember fractions. The spectral mixture model fit was checked by using the unmixing error of the spectral mixture model, the PV/NPV/BS/shadow ground validation RMSE , the R2 , and the Relative RMSE (RRMSE) . The use of the RMSE of the spectral mixture model is mainly aimed at validating the accuracy of the model in unmixing the mixture spectral. The RMSE of endmembers was calculated to quantify the difference between the measured fraction and estimated fraction for the fields. RRMSE is a measure of the deviation rate with percent as the unit, and values closer to zero are indicative of a better fit.

Bilinea Spectral Mixture Model unmixiIDL CODE

Step 9.

```
pro BSMM_unmixing_414
COMPILE_OPT idl2
envi, /restore_base_save_files

;□**********************************************************************************************************
**************************
```

```
;; ************************************************************************************************************************
emMixNB=1;3EM:from 0to5 (0表示线性的; 1 表示有一组端元混合; 2 有 2组端元迭代; 3 有3组端元迭代; 4 有4组端元迭代; 5 有五组端元混合)
;4EM:from 0to8(0表示线性的; 1 表示有一组端元混合; 2 有 2组端元迭代; 3 有3组端元迭代; 4 有4组端元迭代; 5 有五组端元混合 .....8 有五组端元混合)
em_num=4; 3表示3端元模型; 4表示4端元模型
; kernel=0
fcls_method=2 ; 1:jinying's ;NCLS Reference:"Fully Constrained Least Squares Linear Spectral Mixture Analysis Method for Material Quantification in Hyperspectral Imagery"
;2:jicc's NCLS Reference:"Constrained Subpixel Target Detection for Remotely Sensed Imagery"

; m_num=10 ;需要修改 ,请输入method方法组合数目
; em_num=3 ;请输入纯端元数目
; mixem_num=5 ;非线性纯端元+混合端元数目
; band_num=1653 ;请输入波段数目
; sample_num=20
;; ************************************************************************************************************************
;; ************************************************************************************************************************

;开始设置路径, 读取txt(n列, n行)
txtname2='D:\hyperion_paper\data_idl\ss_data\SS_pixcel.txt' ;输入混合像元反射率,全部样本混合端元数据, 多少样本多少列, 例如白刺为20列

if file_test(txtname2)then begin
nLines2=file_lines(txtname2) ;获取行
tmp2=''
; 打开文件
openr,lun2,txtname2,/get_lun
while(EOF(lun2))do begin
readf,lun2,tmp2 ;获取第一行
;print,tmp2
;help,tmp2
var2=strsplit(tmp2,/extract) ;拆分第一行获取列数
rowNum=N_elements(var2)
vararr2=fltarr(rowNum,nLines2-1) ;定义数组(n, n-1)
readf,lun2,vararr2
vararr2=[[var2],[vararr2]] ;重新组合数组(n, n-1)+第一行=（n,n） over!
band_num=N_elements(vararr2[0,*])
pixels_group=reform(TRANSPOSE(vararr2),band_num,rowNum)
; pixels=reform(TRANSPOSE(vararr2),band_num,1) ; 只有一个样本时采用
endwhile
endif
txtname1='D:\hyperion_paper\data_idl\ss_data\4EM_SS_EM_2.txt' ;输入数据为PV,SOIL,NPV.....反射率, 根据（端元数量*样本数量）决定多少列
;ss顺序为PV npv BS shadow
if file_test(txtname1)then begin
nLines1=file_lines(txtname1) ;获取行
```

```
tmp1=''
; 打开文件
openr,lun,txtname1,/get_lun
while(EOF(lun))do begin
readf,lun,tmp1 ;获取第一行
;print,tmp1
;help,tmp
var1=strsplit(tmp1,/extract) ;拆分第一行获取列数
rowNum1=N_elements(var1)
vararr1=fltarr(rowNum1,nLines1-1) ;定义数组(n，n-1)
readf,lun,vararr1
vararr1=[[var1],[vararr1]] ;重新组合数组(n，n-1)+第一行=（n,n） over!
endwhile
endif
; em_num=N_elements(vararr1[*,0])

; txtname4='D:\hyperion_paper\data_idl\bc_data\RMSE_PERCENT\ttttttttt.txt'
txtname4='D:\hyperion_paper\data_idl\bc_data\遥感学报\bc_4em_bsmm.txt'
openw,lun4,txtname4,/get_lun,/append ;/append连续写入函数
printf,lun4,'PV NPV BS (SHADOW) model_RMSE model_rmse(%)'
printf,lun4,em_num,'EM','+',emMixNB

;print fraction continue to check the accuracy of fraction
txtname5='D:\hyperion_paper\data_idl\ss_data\ss_1020_gauss\mix_pixel\ss_4em_bsmm_pix.txt'
openw,lun5,txtname5,/get_lun,/append ;/append连续写入函数

case em_num of
3:begin
case emMixNB of
0:m_num=1
1:m_num=5
2:m_num=10
3:m_num=10
4:m_num=5
5:m_num=1
endcase
end
4:begin
case emMixNB of
0:m_num=1
1:m_num=8
2:m_num=28
3:m_num=56
4:m_num=70
5:m_num=56
6:m_num=28
7:m_num=8
8:m_num=1
endcase
```

```
end
endcase

result_group=fltarr(em_num+2,m_num,rowNum)
g=0



;一组组的读取样本数据，一个循环为一个样本数据
for sm=0,rowNum-1 do begin

;openw,lun4,txtname4,/get_lun,/append ;/append连续写入函数
case em_num of
3:begin
;3EM
pixels=reform((TRANSPOSE(pixels_group))[sm,*],band_num,1)

pv1=vararr1[sm*em_num+0,*] ;获取第一列
bs1=vararr1[sm*em_num+1,*] ;获取第二列
npv1=vararr1[sm*em_num+2,*]
pvpv=TRANSPOSE(vararr1[sm*em_num+0,*]*vararr1[sm*em_num+0,*])
npvnpv=TRANSPOSE(vararr1[sm*em_num+2,*]*vararr1[sm*em_num+2,*])
pvbs=TRANSPOSE(vararr1[sm*em_num+0,*]*vararr1[sm*em_num+1,*])
pvnpv=TRANSPOSE(vararr1[sm*em_num+0,*]*vararr1[sm*em_num+2,*])
bsnpv=TRANSPOSE(vararr1[sm*em_num+1,*]*vararr1[sm*em_num+2,*])

;; ********************************************************************************************************
*
case emMixNB of
0:begin
;3EM+0 1组
m_num=1
;mixem_num=3
em=transpose([pv1,bs1,npv1])
;printf,lun4,'3em+0'
end
1:begin
;3em+1em 5组
m_num=5
; mixem_num=4

vararr71=[[transpose([pv1,bs1,npv1])],[pvpv]]
vararr72=[[transpose([pv1,bs1,npv1])],[npvnpv]]
vararr73=[[transpose([pv1,bs1,npv1])],[pvbs]]
vararr74=[[transpose([pv1,bs1,npv1])],[pvnpv]]
vararr75=[[transpose([pv1,bs1,npv1])],[bsnpv]]
em=[[vararr71],[vararr72],[vararr73],[vararr74],[vararr75]]
;printf,lun4,'3em+1'
end
;; ********************************************************************************************************
```

```
*
2:begin
m_num=10
; mixem_num=5
;3em+2em 10
vararr71=[[transpose([pv1,bs1,npv1])],[pvpv],[npvnpv]]
vararr72=[[transpose([pv1,bs1,npv1])],[pvpv],[bsnpv]]
vararr73=[[transpose([pv1,bs1,npv1])],[pvpv],[pvbs]]
vararr74=[[transpose([pv1,bs1,npv1])],[pvpv],[pvnpv]]
vararr75=[[transpose([pv1,bs1,npv1])],[pvnpv],[bsnpv]]
vararr76=[[transpose([pv1,bs1,npv1])],[pvnpv],[pvbs]]
vararr77=[[transpose([pv1,bs1,npv1])],[bsnpv],[pvbs]]
vararr78=[[transpose([pv1,bs1,npv1])],[npvnpv],[pvnpv]]
vararr79=[[transpose([pv1,bs1,npv1])],[npvnpv],[bsnpv]]
vararr710=[[transpose([pv1,bs1,npv1])],[npvnpv],[pvbs]]
em=[[vararr71],[vararr72],[vararr73],[vararr74],[vararr75],[vararr76],[vararr77],[vararr78],[vararr79]
,[vararr710]]
;printf,lun4,'3em+2'
end
;**********************************************************************************************************
******************
;3em+3em 10
3:begin
m_num=10
; mixem_num=6
vararr71=[[transpose([pv1,bs1,npv1])],[pvpv],[npvnpv],[pvnpv]]
vararr72=[[transpose([pv1,bs1,npv1])],[pvpv],[npvnpv],[bsnpv]]
vararr73=[[transpose([pv1,bs1,npv1])],[pvpv],[npvnpv],[pvbs]]
vararr74=[[transpose([pv1,bs1,npv1])],[pvpv],[pvnpv],[bsnpv]]
vararr75=[[transpose([pv1,bs1,npv1])],[pvpv],[pvnpv],[pvbs]]
vararr76=[[transpose([pv1,bs1,npv1])],[pvpv],[bsnpv],[pvbs]]
vararr77=[[transpose([pv1,bs1,npv1])],[npvnpv],[pvnpv],[bsnpv]]
vararr78=[[transpose([pv1,bs1,npv1])],[npvnpv],[pvnpv],[pvbs]]
vararr79=[[transpose([pv1,bs1,npv1])],[npvnpv],[bsnpv],[pvbs]]
vararr710=[[transpose([pv1,bs1,npv1])],[pvnpv],[bsnpv],[pvbs]]
em=[[vararr71],[vararr72],[vararr73],[vararr74],[vararr75],[vararr76],[vararr77],[vararr78],[vararr79]
,[vararr710]]
;printf,lun4,'3em+3'
end
;**********************************************************************************************************
******************
;3em+4em 5
4:begin
m_num=5
; mixem_num=7
vararr71=[[transpose([pv1,bs1,npv1])],[pvpv],[npvnpv],[pvnpv],[bsnpv]]
vararr72=[[transpose([pv1,bs1,npv1])],[pvpv],[npvnpv],[pvnpv],[pvbs]]
vararr73=[[transpose([pv1,bs1,npv1])],[pvpv],[npvnpv],[bsnpv],[pvbs]]
vararr74=[[transpose([pv1,bs1,npv1])],[pvpv],[pvnpv],[bsnpv],[pvbs]]
```

```
vararr75=[[transpose([pv1,bs1,npv1])],[npvnpv],[pvnpv],[bsnpv],[pvbs]]
em=[[vararr71],[vararr72],[vararr73],[vararr74],[vararr75]]
;printf,lun4,'3em+4'
end
;*****************************************************************************************************
**********************************************
5:begin
;3em+5em 1□
m_num=1
; mixem_num=8
vararr71=[[transpose([pv1,bs1,npv1])],[pvpv],[npvnpv],[pvnpv],[bsnpv],[pvbs]] ;3□□
em=[vararr71]
;printf,lun4,'3em+5'
end
;*****************************************************************************************************
**********************************************
endcase

end
4:begin
;4EM
; pixels=TRANSPOSE((TRANSPOSE(pixels_group))[sm,*])
pixels=reform((TRANSPOSE(pixels_group))[sm,*],band_num,1)
pv1=vararr1[sm*em_num+0,*] ;□□□□□
bs1=vararr1[sm*em_num+1,*] ;□□□□□
npv1=vararr1[sm*em_num+2,*]
shadow1=vararr1[sm*em_num+3,*]
pp=TRANSPOSE(vararr1[sm*em_num+0,*]*vararr1[sm*em_num+0,*])
nn=TRANSPOSE(vararr1[sm*em_num+2,*]*vararr1[sm*em_num+2,*])
pb=TRANSPOSE(vararr1[sm*em_num+0,*]*vararr1[sm*em_num+1,*])
pn=TRANSPOSE(vararr1[sm*em_num+0,*]*vararr1[sm*em_num+2,*])
ps=TRANSPOSE(vararr1[sm*em_num+0,*]*vararr1[sm*em_num+3,*])
bn=TRANSPOSE(vararr1[sm*em_num+1,*]*vararr1[sm*em_num+2,*])
bs=TRANSPOSE(vararr1[sm*em_num+1,*]*vararr1[sm*em_num+3,*])
ns=TRANSPOSE(vararr1[sm*em_num+2,*]*vararr1[sm*em_num+3,*])


; 4EM
case emMixNB of

0:begin
m_num=1
em=transpose([pv1,bs1,npv1,shadow1])
;printf,lun4,'4em+0'
END
; 4+1EM
1:begin
m_num=8
vararr71=[[transpose([pv1,bs1,npv1,shadow1])],[pp]]
```

```
vararr72=[[transpose([pv1,bs1,npv1,shadow1])],[nn]]
vararr73=[[transpose([pv1,bs1,npv1,shadow1])],[pb]]
vararr74=[[transpose([pv1,bs1,npv1,shadow1])],[pn]]
vararr75=[[transpose([pv1,bs1,npv1,shadow1])],[ps]]
vararr76=[[transpose([pv1,bs1,npv1,shadow1])],[bn]]
vararr77=[[transpose([pv1,bs1,npv1,shadow1])],[bs]]
vararr78=[[transpose([pv1,bs1,npv1,shadow1])],[ns]]
em=[[vararr71],[vararr72],[vararr73],[vararr74],[vararr75],[vararr76],[vararr77],[vararr78]]
;printf,lun4,'4em+1'
END
; 4+2EM
2:begin
m_num=28
vararr71=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn]]
vararr72=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[pb]]
vararr73=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[pn]]
vararr74=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[ps]]
vararr75=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[bn]]
vararr76=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[bs]]
vararr77=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[ns]]
vararr78=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[pb]]
vararr79=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[pn]]
vararr80=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[ps]]
vararr81=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[bn]]
vararr82=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[bs]]
vararr83=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[ns]]
vararr84=[[transpose([pv1,bs1,npv1,shadow1])],[pb],[pn]]
vararr85=[[transpose([pv1,bs1,npv1,shadow1])],[pb],[ps]]
vararr86=[[transpose([pv1,bs1,npv1,shadow1])],[pb],[bn]]
vararr87=[[transpose([pv1,bs1,npv1,shadow1])],[pb],[bs]]
vararr88=[[transpose([pv1,bs1,npv1,shadow1])],[pb],[ns]]
vararr89=[[transpose([pv1,bs1,npv1,shadow1])],[pn],[ps]]
vararr90=[[transpose([pv1,bs1,npv1,shadow1])],[pn],[bn]]
vararr91=[[transpose([pv1,bs1,npv1,shadow1])],[pn],[bs]]
vararr92=[[transpose([pv1,bs1,npv1,shadow1])],[pn],[ns]]
vararr93=[[transpose([pv1,bs1,npv1,shadow1])],[ps],[bn]]
vararr94=[[transpose([pv1,bs1,npv1,shadow1])],[ps],[bs]]
vararr95=[[transpose([pv1,bs1,npv1,shadow1])],[ps],[ns]]
vararr96=[[transpose([pv1,bs1,npv1,shadow1])],[bn],[bs]]
vararr97=[[transpose([pv1,bs1,npv1,shadow1])],[bn],[ns]]
vararr98=[[transpose([pv1,bs1,npv1,shadow1])],[bs],[ns]]
em=[[vararr71],[vararr72],[vararr73],[vararr74],[vararr75],[vararr76],[vararr77],[vararr78],[vararr79]
,[vararr80],$
[vararr81],[vararr82],[vararr83],[vararr84],[vararr85],[vararr86],[vararr87],[vararr88],[vararr89],[vara
rr90],$
[vararr91],[vararr92],[vararr93],[vararr94],[vararr95],[vararr96],[vararr97],[vararr98]]
;printf,lun4,'4em+2'
END
; 4+3EM
```

3:begin
m_num=56
vararr21=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[pb]]
vararr22=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[pn]]
vararr23=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[ps]]
vararr24=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[bn]]
vararr25=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[bs]]
vararr26=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[ns]]
vararr27=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[pb],[pn]]
vararr28=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[pb],[ps]]
vararr29=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[pb],[bn]]
vararr30=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[pb],[bs]]
vararr31=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[pb],[ns]]
vararr32=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[pn],[ps]]
vararr33=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[pn],[bn]]
vararr34=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[pn],[bs]]
vararr35=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[pn],[ns]]
vararr36=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[ps],[bn]]
vararr37=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[ps],[bs]]
vararr38=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[ps],[ns]]
vararr39=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[bn],[bs]]
vararr40=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[bn],[ns]]
vararr41=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[bs],[ns]]
vararr42=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[pb],[pn]]
vararr43=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[pb],[ps]]
vararr44=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[pb],[bn]]
vararr45=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[pb],[bs]]
vararr46=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[pb],[ns]]
vararr47=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[pn],[ps]]
vararr48=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[pn],[bn]]
vararr49=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[pn],[bs]]
vararr50=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[pn],[ns]]
vararr51=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[ps],[bn]]
vararr52=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[ps],[bs]]
vararr53=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[ps],[ns]]
vararr54=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[bn],[bs]]
vararr55=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[bn],[ns]]
vararr56=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[bs],[ns]]
vararr57=[[transpose([pv1,bs1,npv1,shadow1])],[pb],[pn],[ps]]
vararr58=[[transpose([pv1,bs1,npv1,shadow1])],[pb],[pn],[bn]]
vararr59=[[transpose([pv1,bs1,npv1,shadow1])],[pb],[pn],[bs]]
vararr60=[[transpose([pv1,bs1,npv1,shadow1])],[pb],[pn],[ns]]
vararr61=[[transpose([pv1,bs1,npv1,shadow1])],[pb],[ps],[bn]]
vararr62=[[transpose([pv1,bs1,npv1,shadow1])],[pb],[ps],[bs]]
vararr63=[[transpose([pv1,bs1,npv1,shadow1])],[pb],[ps],[ns]]
vararr64=[[transpose([pv1,bs1,npv1,shadow1])],[pb],[bn],[bs]]
vararr65=[[transpose([pv1,bs1,npv1,shadow1])],[pb],[bn],[ns]]
vararr66=[[transpose([pv1,bs1,npv1,shadow1])],[pb],[bs],[ns]]
vararr67=[[transpose([pv1,bs1,npv1,shadow1])],[pn],[ps],[bn]]

```
vararr68=[[transpose([pv1,bs1,npv1,shadow1])],[pn],[ps],[bs]]
vararr69=[[transpose([pv1,bs1,npv1,shadow1])],[pn],[ps],[ns]]
vararr70=[[transpose([pv1,bs1,npv1,shadow1])],[pn],[bn],[bs]]
vararr71=[[transpose([pv1,bs1,npv1,shadow1])],[pn],[bn],[ns]]
vararr72=[[transpose([pv1,bs1,npv1,shadow1])],[pn],[bs],[ns]]
vararr73=[[transpose([pv1,bs1,npv1,shadow1])],[ps],[bn],[bs]]
vararr74=[[transpose([pv1,bs1,npv1,shadow1])],[ps],[bn],[ns]]
vararr75=[[transpose([pv1,bs1,npv1,shadow1])],[ps],[bs],[ns]]
vararr76=[[transpose([pv1,bs1,npv1,shadow1])],[bn],[bs],[ns]]
em=[[vararr21],[vararr22],[vararr23],[vararr24],[vararr25],[vararr26],[vararr27],[vararr28],[vararr29]
,[vararr30],$
[vararr31],[vararr32],[vararr33],[vararr34],[vararr35],[vararr36],[vararr37],[vararr38],[vararr39],[vara
rr40],$
[vararr41],[vararr42],[vararr43],[vararr44],[vararr45],[vararr46],[vararr47],[vararr48],[vararr49],[vara
rr50],$
[vararr51],[vararr52],[vararr53],[vararr54],[vararr55],[vararr56],[vararr57],[vararr58],[vararr59],[vara
rr60],$
[vararr61],[vararr62],[vararr63],[vararr64],[vararr65],[vararr66],[vararr67],[vararr68],[vararr69],[vara
rr70],$
[vararr71],[vararr72],[vararr73],[vararr74],[vararr75],[vararr76]]
;printf,lun4,'4em+3'
END
;4+4EM
4:begin
m_num=70
vararr21=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[pb],[pn]]
vararr22=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[pb],[ps]]
vararr23=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[pb],[bn]]
vararr24=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[pb],[bs]]
vararr25=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[pb],[ns]]
vararr26=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[pn],[ps]]
vararr27=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[pn],[bn]]
vararr28=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[pn],[bs]]
vararr29=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[pn],[ns]]
vararr30=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[ps],[bn]]
vararr31=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[ps],[bs]]
vararr32=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[ps],[ns]]
vararr33=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[bn],[bs]]
vararr34=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[bn],[ns]]
vararr35=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[bs],[ns]]
vararr36=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[pb],[pn],[ps]]
vararr37=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[pb],[pn],[bn]]
vararr38=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[pb],[pn],[bs]]
vararr39=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[pb],[pn],[ns]]
vararr40=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[pb],[ps],[bn]]
vararr41=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[pb],[ps],[bs]]
vararr42=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[pb],[ps],[ns]]
vararr43=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[pb],[bn],[bs]]
vararr44=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[pb],[bn],[ns]]
```

```
vararr45=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[pb],[bs],[ns]]
vararr46=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[pn],[ps],[bn]]
vararr47=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[pn],[ps],[bs]]
vararr48=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[pn],[ps],[ns]]
vararr49=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[pn],[bn],[bs]]
vararr50=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[pn],[bn],[ns]]
vararr51=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[pn],[bs],[ns]]
vararr52=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[ps],[bn],[bs]]
vararr53=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[ps],[bn],[ns]]
vararr54=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[ps],[bs],[ns]]
vararr55=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[bn],[bs],[ns]]
vararr56=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[pb],[pn],[ps]]
vararr57=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[pb],[pn],[bn]]
vararr58=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[pb],[pn],[bs]]
vararr59=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[pb],[pn],[ns]]
vararr60=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[pb],[ps],[bn]]
vararr61=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[pb],[ps],[bs]]
vararr62=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[pb],[ps],[ns]]
vararr63=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[pb],[bn],[bs]]
vararr64=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[pb],[bn],[ns]]
vararr65=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[pb],[bs],[ns]]
vararr66=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[pn],[ps],[bn]]
vararr67=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[pn],[ps],[bs]]
vararr68=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[pn],[ps],[ns]]
vararr69=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[pn],[bn],[bs]]
vararr70=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[pn],[bn],[ns]]
vararr71=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[pn],[bs],[ns]]
vararr72=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[ps],[bn],[bs]]
vararr73=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[ps],[bn],[ns]]
vararr74=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[ps],[bs],[ns]]
vararr75=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[bn],[bs],[ns]]
vararr76=[[transpose([pv1,bs1,npv1,shadow1])],[pb],[pn],[ps],[bn]]
vararr77=[[transpose([pv1,bs1,npv1,shadow1])],[pb],[pn],[ps],[bs]]
vararr78=[[transpose([pv1,bs1,npv1,shadow1])],[pb],[pn],[ps],[ns]]
vararr79=[[transpose([pv1,bs1,npv1,shadow1])],[pb],[pn],[bn],[bs]]
vararr80=[[transpose([pv1,bs1,npv1,shadow1])],[pb],[pn],[bn],[ns]]
vararr81=[[transpose([pv1,bs1,npv1,shadow1])],[pb],[pn],[bs],[ns]]
vararr82=[[transpose([pv1,bs1,npv1,shadow1])],[pb],[ps],[bn],[bs]]
vararr83=[[transpose([pv1,bs1,npv1,shadow1])],[pb],[ps],[bn],[ns]]
vararr84=[[transpose([pv1,bs1,npv1,shadow1])],[pb],[ps],[bs],[ns]]
vararr85=[[transpose([pv1,bs1,npv1,shadow1])],[pb],[bn],[bs],[ns]]
vararr86=[[transpose([pv1,bs1,npv1,shadow1])],[pn],[ps],[bn],[bs]]
vararr87=[[transpose([pv1,bs1,npv1,shadow1])],[pn],[ps],[bn],[ns]]
vararr88=[[transpose([pv1,bs1,npv1,shadow1])],[pn],[ps],[bs],[ns]]
vararr89=[[transpose([pv1,bs1,npv1,shadow1])],[pn],[bn],[bs],[ns]]
vararr90=[[transpose([pv1,bs1,npv1,shadow1])],[ps],[bn],[bs],[ns]]
em=[[vararr21],[vararr22],[vararr23],[vararr24],[vararr25],[vararr26],[vararr27],[vararr28],[vararr29]
,[vararr30],$
[vararr31],[vararr32],[vararr33],[vararr34],[vararr35],[vararr36],[vararr37],[vararr38],[vararr39],[vara
```

```
rr40],$
[vararr41],[vararr42],[vararr43],[vararr44],[vararr45],[vararr46],[vararr47],[vararr48],[vararr49],[vara
rr50],$
[vararr51],[vararr52],[vararr53],[vararr54],[vararr55],[vararr56],[vararr57],[vararr58],[vararr59],[vara
rr60],$
[vararr61],[vararr62],[vararr63],[vararr64],[vararr65],[vararr66],[vararr67],[vararr68],[vararr69],[vara
rr70],$
[vararr71],[vararr72],[vararr73],[vararr74],[vararr75],[vararr76],[vararr77],[vararr78],[vararr79],[vara
rr80],$
[vararr81],[vararr82],[vararr83],[vararr84],[vararr85],[vararr86],[vararr87],[vararr88],[vararr89],[vara
rr90]]
;printf,lun4,'4em+4'
END
;4+5EM
5:BEGIN
m_num=56
vararr21=[[transpose([pv1,bs1,npv1,shadow1])],[pn],[ps],[bn],[bs],[ns]]
vararr22=[[transpose([pv1,bs1,npv1,shadow1])],[pb],[ps],[bn],[bs],[ns]]
vararr23=[[transpose([pv1,bs1,npv1,shadow1])],[pb],[pn],[bn],[bs],[ns]]
vararr24=[[transpose([pv1,bs1,npv1,shadow1])],[pb],[pn],[ps],[bs],[ns]]
vararr25=[[transpose([pv1,bs1,npv1,shadow1])],[pb],[pn],[ps],[bn],[ns]]
vararr26=[[transpose([pv1,bs1,npv1,shadow1])],[pb],[pn],[ps],[bn],[bs]]
vararr27=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[ps],[bn],[bs],[ns]]
vararr28=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[pn],[bn],[bs],[ns]]
vararr29=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[pn],[ps],[bs],[ns]]
vararr30=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[pn],[ps],[bn],[ns]]
vararr31=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[pn],[ps],[bn],[bs]]
vararr32=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[pb],[bn],[bs],[ns]]
vararr33=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[pb],[ps],[bs],[ns]]
vararr34=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[pb],[ps],[bn],[ns]]
vararr35=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[pb],[ps],[bn],[bs]]
vararr36=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[pb],[pn],[bs],[ns]]
vararr37=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[pb],[pn],[bn],[ns]]
vararr38=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[pb],[pn],[bn],[bs]]
vararr39=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[pb],[pn],[ps],[ns]]
vararr40=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[pb],[pn],[ps],[bs]]
vararr41=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[pb],[pn],[ps],[bn]]
vararr42=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[ps],[bn],[bs],[ns]]
vararr43=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[pn],[bn],[bs],[ns]]
vararr44=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[pn],[ps],[bs],[ns]]
vararr45=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[pn],[ps],[bn],[ns]]
vararr46=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[pn],[ps],[bn],[bs]]
vararr47=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[pb],[bn],[bs],[ns]]
vararr48=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[pb],[ps],[bs],[ns]]
vararr49=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[pb],[ps],[bn],[ns]]
vararr50=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[pb],[ps],[bn],[bs]]
vararr51=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[pb],[pn],[bs],[ns]]
vararr52=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[pb],[pn],[bn],[ns]]
vararr53=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[pb],[pn],[bn],[bs]]
```

```
vararr54=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[pb],[pn],[ps],[ns]]
vararr55=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[pb],[pn],[ps],[bs]]
vararr56=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[pb],[pn],[ps],[bn]]
vararr57=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[bn],[bs],[ns]]
vararr58=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[ps],[bs],[ns]]
vararr59=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[ps],[bn],[ns]]
vararr60=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[ps],[bn],[bs]]
vararr61=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[pn],[bs],[ns]]
vararr62=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[pn],[bn],[ns]]
vararr63=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[pn],[bn],[bs]]
vararr64=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[pn],[ps],[ns]]
vararr65=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[pn],[ps],[bs]]
vararr66=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[pn],[ps],[bn]]
vararr67=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[pb],[bs],[ns]]
vararr68=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[pb],[bn],[ns]]
vararr69=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[pb],[bn],[bs]]
vararr70=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[pb],[ps],[ns]]
vararr71=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[pb],[ps],[bs]]
vararr72=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[pb],[ps],[bn]]
vararr73=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[pb],[pn],[ns]]
vararr74=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[pb],[pn],[bs]]
vararr75=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[pb],[pn],[bn]]
vararr76=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[pb],[pn],[ps]]
em=[[vararr21],[vararr22],[vararr23],[vararr24],[vararr25],[vararr26],[vararr27],[vararr28],[vararr29]
,[vararr30],$
[vararr31],[vararr32],[vararr33],[vararr34],[vararr35],[vararr36],[vararr37],[vararr38],[vararr39],[vara
rr40],$
[vararr41],[vararr42],[vararr43],[vararr44],[vararr45],[vararr46],[vararr47],[vararr48],[vararr49],[vara
rr50],$
[vararr51],[vararr52],[vararr53],[vararr54],[vararr55],[vararr56],[vararr57],[vararr58],[vararr59],[vara
rr60],$
[vararr61],[vararr62],[vararr63],[vararr64],[vararr65],[vararr66],[vararr67],[vararr68],[vararr69],[vara
rr70],$
[vararr71],[vararr72],[vararr73],[vararr74],[vararr75],[vararr76]]
;printf,lun4,'4em+5'
END
;4+6EM
6:BEGIN
m_num=28
vararr71=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[pb],[pn],[ps],[bn]] ;4+2端元，将不同的列组合成新的矩阵
vararr72=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[pb],[pn],[ps],[ns]] ;4+2端元，将不同的列组合成新的矩阵
vararr73=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[pb],[pn],[ps],[bs]] ;4+2端元，将不同的列组合成新的矩阵
vararr74=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[pb],[pn],[bn],[ns]] ;4+2端元，将不同的列组合成新的矩阵
vararr75=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[pb],[pn],[bn],[bs]] ;4+2端元，将不同的列组合成新的矩阵
vararr76=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[pb],[pn],[bs],[ns]] ;4+2端元，将不同的列组合成新的矩阵
vararr77=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[pb],[ps],[bn],[ns]] ;4+2端元，将不同的列组合成新的矩阵
vararr78=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[pb],[ps],[bn],[bs]] ;4+2端元，将不同的列组合成新的矩阵
vararr79=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[pb],[ps],[bs],[ns]] ;4+2端元，将不同的列组合成新的矩阵
vararr80=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[pb],[bn],[bs],[ns]] ;4+2端元，将不同的列组合成新的矩阵
```

vararr81=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[ps],[bn],[bs],[ns]] ;4+2端元，将不同的列组合成新的矩阵
vararr82=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[pn],[bn],[bs],[ns]] ;4+2端元，将不同的列组合成新的矩阵
vararr83=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[pn],[ps],[bs],[ns]] ;4+2端元，将不同的列组合成新的矩阵
vararr84=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[pn],[ps],[bn],[bs]] ;4+2端元，将不同的列组合成新的矩阵
vararr85=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[pn],[ps],[bn],[ns]] ;4+2端元，将不同的列组合成新的矩阵
vararr86=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[pn],[ps],[bn],[bs],[ns]] ;4+2端元，将不同的列组合成新的矩阵
vararr87=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[pb],[ps],[bn],[bs],[ns]] ;4+2端元，将不同的列组合成新的矩阵
vararr88=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[pb],[pn],[bn],[bs],[ns]] ;4+2端元，将不同的列组合成新的矩阵
vararr89=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[pb],[pn],[ps],[bs],[ns]] ;4+2端元，将不同的列组合成新的矩阵
vararr90=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[pb],[pn],[ps],[bn],[ns]] ;4+2端元，将不同的列组合成新的矩阵
vararr91=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[pb],[pn],[ps],[bn],[bs]] ;4+2端元，将不同的列组合成新的矩阵
vararr92=[[transpose([pv1,bs1,npv1,shadow1])],[pb],[pn],[ps],[bn],[bs],[ns]] ;4+2端元，将不同的列组合成新的矩阵
vararr93=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[pn],[ps],[bn],[bs],[ns]] ;4+2端元，将不同的列组合成新的矩阵
vararr94=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[pb],[ps],[bn],[bs],[ns]] ;4+2端元，将不同的列组合成新的矩阵
vararr95=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[pb],[pn],[bn],[bs],[ns]] ;4+2端元，将不同的列组合成新的矩阵
vararr96=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[pb],[pn],[ps],[bs],[ns]] ;4+2端元，将不同的列组合成新的矩阵
vararr97=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[pb],[pn],[ps],[bn],[ns]] ;4+2端元，将不同的列组合成新的矩阵
vararr98=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[pb],[pn],[ps],[bn],[bs]] ;4+2端元，将不同的列组合成新的矩阵
em=[[vararr71],[vararr72],[vararr73],[vararr74],[vararr75],[vararr76],[vararr77],[vararr78],[vararr79]
,[vararr80],$
[vararr81],[vararr82],[vararr83],[vararr84],[vararr85],[vararr86],[vararr87],[vararr88],[vararr89],[vara
rr90],$
[vararr91],[vararr92],[vararr93],[vararr94],[vararr95],[vararr96],[vararr97],[vararr98]]
;printf,lun4,'4em+6'
END
;4+7EM
7:BEGIN
m_num=8
vararr71=[[transpose([pv1,bs1,npv1,shadow1])],[nn],[pb],[pn],[ps],[bn],[bs],[ns]] ;4+7端元，将不同的列组合成新的矩阵
vararr72=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[pb],[pn],[ps],[bn],[bs],[ns]] ;4+7端元，将不同的列组合成新的矩阵
vararr73=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[pn],[ps],[bn],[bs],[ns]] ;4+7端元，将不同的列组合成新的矩阵
vararr74=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[pb],[ps],[bn],[bs],[ns]] ;4+7端元，将不同的列组合成新的矩阵
vararr75=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[pb],[pn],[bn],[bs],[ns]] ;4+7端元，将不同的列组合成新的矩阵
vararr76=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[pb],[pn],[ps],[bs],[ns]] ;4+7端元，将不同的列组合成新的矩阵
vararr77=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[pb],[pn],[ps],[bn],[ns]] ;4+7端元，将不同的列组合成新的矩阵
vararr78=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[pb],[pn],[ps],[bn],[bs]] ;4+7端元，将不同的列组合成新的矩阵
em=[[vararr71],[vararr72],[vararr73],[vararr74],[vararr75],[vararr76],[vararr77],[vararr78]]
;printf,lun4,'4em+7'
END
;4+8EM 1组实验
8:BEGIN

```
m_num=1
vararr12=[[transpose([pv1,bs1,npv1,shadow1])],[pp],[nn],[pb],[pn],[ps],[bn],[bs],[ns]] ;4+8端元，将不同的列组合成新的矩阵
em=[vararr12]
;printf,lun4,'4em+8'
END

;*****************************************************************************************************************
*****

endcase

end
endcase

;
;*****************************************************************************************************************
*****************************************

;限制为1的，非负计算，所有端元的丰度结果及RMSE
n_num=N_elements(em[*,0])
nr_num=N_elements(em[0,*])
mixem_num=nr_num/m_num
result_a=fltarr(m_num,mixem_num)
rmse_a=fltarr(m_num)
Error_per=fltarr(m_num)
pre_pixPer=fltarr(m_num,n_num)
;
for nn=0,m_num-1 do begin
case fcls_method of ;
1:begin ;NCLS Reference:Fully Constrained Least Squares Linear Spectral Mixture Analysis Method for Material Quantification in Hyperspectral Imagery
FCLS_ncls_Heinz,em[*,(nn*mixem_num):(nn*mixem_num+mixem_num-1)],pixels,nb,result=result,rmse=rmse,R_Error3=R_Error3,pre_pix=pre_pix ;限制加和为1，并非负
end
2:begin ;NCLS Reference: Constrained Subpixel Target Detection for Remotely Sensed Imagery
FCLS_ncls_Chang_Heinz,em[*,(nn*mixem_num):(nn*mixem_num+mixem_num-1)],pixels,nb,result=result,rmse=rmse,R_Error3=R_Error3,pre_pix=pre_pix ;限制加和为1，并非负
;
KFCLS_ncls_Chang_Heinz,em[*,(nn*mixem_num):(nn*mixem_num+mixem_num-1)],pixels,nb,kernel,result=result,rmse=rmse,R_Error3=R_Error3,pre_pix=pre_pix ;限制加和为1，并非负
end
endcase
result_a[nn,*]=result
rmse_a[nn,*]=rmse
Error_per[nn,*]=R_Error3
pre_pixPer[nn,*]=pre_pix
```

```
endfor
;输出各端元丰度 结果矩阵及各模型针对一个样本的rmse矩阵
resultall=reform(transpose(result_a),mixem_num,m_num)
rmseall=transpose(rmse_a)

;将混合像元丰度分配到各纯净像元中
dims_R = SIZE(resultall, /dimensions)
r = dims_R[0] ;列数
t=dims_R[1] ;行数

case em_num of
3:begin
if r eq 3 then begin
resultall_sub=resultall[0:(r-1),*]
resultfraction=resultall_sub
; realresult=[[resultall], rmseall,transpose(Error_per)]
endif else begin
resultall_sub=resultall[em_num:(r-1),*]
Sum_other=total(resultall_sub,1)
PV_re1=resultall[0,*]/(1-sum_other)
BS_re1=resultall[1,*]/(1-sum_other)
NPV_re1=resultall[2,*]/(1-sum_other)
;输出结果为丰度 PV BS NPV RMSE
resultfraction=[PV_re1,BS_re1,NPV_re1]

; idx = WHERE((resultfraction LT 0) and (resultfraction ge -0.0001))
; resultfraction[idx]=0
; realresult=[[resultfraction],rmseall,transpose([Error_per])]
endelse
end
4:begin
if r eq 4 then begin
resultall_sub=resultall[0:(r-1),*]
resultfraction=resultall_sub
; realresult=[[resultall],rmseall,transpose(Error_per)]
endif else begin
resultall_sub=resultall[em_num:(r-1),*]
Sum_other=total(resultall_sub,1)
PV_re1=resultall[0,*]/(1-sum_other)
BS_re1=resultall[1,*]/(1-sum_other)
NPV_re1=resultall[2,*]/(1-sum_other)
SHADOW_re1=resultall[3,*]/(1-sum_other)
;输出结果为丰度 PV BS NPV SHADOW RMSE
resultfraction=[PV_re1,BS_re1,NPV_re1,SHADOW_re1]
endelse
end
endcase

idx = WHERE((resultfraction[*,*] LT 0.000001) and (resultfraction[*,*] ge -0.00001))
if min(idx) ne -1 then begin
```

```
resultfraction[idx]=0
endif
realresult=[[resultfraction],rmseall,transpose([Error_per])]
result_group[*,*,g]=realresult

g++
;print,result
; print,resultfraction
; print,realresult
; print,result_group
; printf,lun4,realresult

; free_lun,lun4
;文件输出结果为15、20个是一组数据的结果，这样下个程序可以直接调用TXT文本，每间隔一定数量行为一组新的实验结果数据
; txtname5='D:\hyperion_paper\data_idl\bc_data\RMSE_PERCENT\nnnnnnnnn.txt'
; openw,lun5,txtname5,/get_lun,/append ;/append连续写入函数
printf,lun5,'BSMM mixture pixel'
printf,lun5,transpose(pre_pixPer),format = '(1653F12)'
; free_lun,lun5

ENDFOR ;一组组的读取样本数据过程结束
;提取出各组样本数据同一组合方法的所有结果为一个矩阵
dims_R = SIZE(result_group, /dimensions)
r = dims_R[0] ; ;列数
t=dims_R[1]
g=dims_R[2] ;行数
fraction= fltarr(r,g,t)
for k=0,t-1 do begin
for i=0, g-1 do begin
fraction[*,i,k]=result_group[*,k,i] ;提取出各组样本数据同一组合方法的所有结果为一个矩阵，3em/4EM全部计算
endfor
endfor


print,"emMixNB",emMixNB
; print,result_a
print,fraction
;printf,lun4,fraction ;the same method with the samples

fraction_check=fltarr(r,g*t)

for k=0,t-1 do begin
for i=0, g-1 do begin

fraction_check[*,k*g+i]=result_group[*,k,i]
;print,fraction_check
endfor
endfor
; print,fraction_check
;printf,lun5,fraction_check
```

```
free_lun,lun4
free_lun,lun5
end
```

Linear and Kernel Nonlinear Spectral Mixture Model unmixing IDL CODE

Step 10.

```
pro test_20160314_KFCLS
COMPILE_OPT idl2
envi, /restore_base_save_files


;; ************************************************************************************************************************
;; ************************************************************************************************************************
; ;需要修改 ,请输入method方法组合数目
em_num=3 ;请输入纯端元数目
kernel=2;0:线性核函数; 1: 'KFCLS_RBF' :,径向基函数; 2: 'KFCLS_PKF' :,多项式核函数; 3: 神经网络核函数
; mixem_num=3 ;非线性纯端元+混合端元数目
; band_num=4 ;请输入波段数目
;; ************************************************************************************************************************
;; ************************************************************************************************************************

;开始设置路径，读取txt(n列，n行)
txtname2='D:\hyperion_paper\data_idl\bc_data\BC_PIXCLE.txt';'D:\hyperion_paper\data_idl\bc_data\BC_PIXCLE.txt'

if file_test(txtname2)then begin
nLines2=file_lines(txtname2) ;获取行
tmp2=''
; 打开文件
openr,lun2,txtname2,/get_lun
while(EOF(lun2))do begin
readf,lun2,tmp2 ;获取第一行
; print,tmp2
;help,tmp2
var2=strsplit(tmp2,/extract) ;拆分第一行获取列数
rowNum=N_elements(var2)
vararr2=fltarr(rowNum,nLines2-1) ;定义数组(n，n-1)
readf,lun2,vararr2
vararr2=[[var2],[vararr2]] ;重新组合数组(n，n-1)+第一行=（n,n） over!
band_num=N_elements(vararr2[0,*])
pixels_group=reform(TRANSPOSE(vararr2),band_num,rowNum)
endwhile
endif
```

```
txtname1='D:\hyperion_paper\data_idl\bc_data\3EM_BC_EM.txt';'D:\hyperion_paper\data_idl\bc_data\3EM_BC_EM.txt' ss_data\4EM_SS_EM_2.txt ;输入数据为PV,SOIL,NPV,反射率,端元数据，波段数*端元数
; txtname1='D:\hyperion_paper\data_idl\bc_data\test_baici4_l\6675_a.txt' ;输入数据为PV,SOIL,NPV,反射率
; txtname1='D:\hyperion_paper\data_idl\ss_data\test_ss3_l\6730_a.txt'
if file_test(txtname1)then begin
nLines1=file_lines(txtname1) ;获取行
tmp1=''
; 打开文件
openr,lun,txtname1,/get_lun
while(EOF(lun))do begin
readf,lun,tmp1 ;获取第一行
;print,tmp1
;help,tmp
var1=strsplit(tmp1,/extract) ;拆分第一行获取列数
rowNum1=N_elements(var1)
vararr1=fltarr(rowNum1,nLines1-1) ;定义数组(n，n-1)
readf,lun,vararr1
vararr1=[[var1],[vararr1]] ;重新组合数组(n，n-1)+第一行=（n,n） over!
endwhile
endif
rmse_aa=0
;rmse_bb=0

txtname4='D:\hyperion_paper\data_idl\bc_data\BC_20170511\BC_3em_kfcls_0516.txt';'D:\hyperion_paper\data_idl\bc_data\bc_1020_gauss\pixels_1030\bc_3em_kfcls_pixel.txt'
openw,lun3,txtname4,/get_lun,/append ;/append连续写入函数
printf,lun3,'PV BS NPV (SHADOW) model_RMSE model_rmse(%)'
;白刺，'PV BS NPV (SHADOW) model_RMSE model_rmse(%)'
;梭梭 'PV NPV BS (SHADOW) model_RMSE model_rmse(%)'
;白刺，'PV BS NPV (SHADOW) model_RMSE model_rmse(%)'
;梭梭 'PV NPV BS (SHADOW) model_RMSE model_rmse(%)'
IF kernel EQ 0 THEN BEGIN
printf,lun3,'FCLS_linear',em_num,'EM'
ENDIF
IF kernel EQ 1 THEN BEGIN
printf,lun3,'KFCLS_RBF',em_num,'EM'
ENDIF
IF kernel EQ 2 THEN BEGIN
printf,lun3,'KFCLS_PKF',em_num,'EM'
ENDIF
IF kernel EQ 3 THEN BEGIN
printf,lun3,'FCLS_nnc',em_num,'EM'
ENDIF
for sm=0,rowNum-1 do begin

pixels=reform((TRANSPOSE(pixels_group))[sm,*],band_num,1)
;; ****************************************************************************************************
*
case em_num of
```

```
3:begin
pv1=vararr1[sm*em_num+0,*] ;获取第一列
bs1=vararr1[sm*em_num+1,*] ;获取第二列
npv1=vararr1[sm*em_num+2,*]
em=transpose([pv1,bs1,npv1])
;pixel_lsma_1210,em,pixels,result=result,rmse=rmse ;最小二乘法，没有限制条件
end
4:begin
pv1=vararr1[sm*em_num+0,*] ;获取第一列
bs1=vararr1[sm*em_num+1,*] ;获取第二列
npv1=vararr1[sm*em_num+2,*]
shadow1=vararr1[sm*em_num+3,*]
em=transpose([pv1,bs1,npv1,shadow1])
end
endcase
;限制为1的，非负计算，所有端元的丰度结果及RMSE
m_num=1
result_a=fltarr(m_num,em_num)
rmse_a=fltarr(m_num)
result_b=fltarr(m_num,em_num)
rmse_b=fltarr(m_num)
Error_per=fltarr(m_num)
estmodel_pix=fltarr(m_num,band_num)
for nn=0,m_num-1 do begin
; kernel=1
; ;限制加和为1，并非负
;
pixel_kfcls_20160314,em,pixels,nb,kernel,result=result,rmse=rmse,R_Error3=R_Error3,pre_pix=pre_pix ;遥感学报采用算法
KFCLS_ncls_Chang_Heinz,em,pixels,nb,kernel,result=result,rmse=rmse,R_Error3=R_Error3,pre_pix=pre_pix
; KFCLS_LSOSP_unmixing,em,pixels,kernel,result=result,rmse=rmse
; FCLS_ncls_Chang_Heinz,em,pixels,nb,result=result,rmse=rmse,R_Error3=R_Error3,pre_pix=pre_pix
result_a[nn,*]=result
rmse_a[nn,*]=rmse
Error_per[nn,*]=R_Error3
; for px=0,rowNum-1 do begin
estmodel_pix[nn,*]=pre_pix
; endfor
; kernel2=2
; pixel_kfcls_20160314,em,pixels,nb,kernel2,result=result,rmse=rmse
; pixel_fcls_1210,em,pixels,nb,result=result,rmse=rmse,R_Error3=R_Error3,pre_pix=pre_pix
; result_b=result
; rmse_b=rmse
; kernel4=2
; ;
pixel_kfcls_20160314,em[*,(nn*mixem_num):(nn*mixem_num+mixem_num-1)],pixels,nb,kernel2,result=result,rmse=rmse
; pixel_kfcls_20160314,em,pixels,nb,kernel4,result=result,rmse=rmse
```

```
; result_d=result
; rmse_d=rmse
; kernel5=30
; ;
pixel_kfcls_20160314,em[*,(nn*mixem_num):(nn*mixem_num+mixem_num-1)],pixels,nb,kernel2,resu
lt=result,rmse=rmse
; pixel_kfcls_20160314,em,pixels,nb,kernel5,result=result,rmse=rmse
; result_e=result
; rmse_e=rmse

; kernel3=1
; pixel_kfcls_20160314_IGARS,em,pixels,nb,kernel3,result=result,rmse=rmse
; result_c=result
; rmse_c=rmse
endfor
;输出各端元丰度结果矩阵及各模型针对一个样本的rmse矩阵
; resultall=transpose(result_a)

rmsealla=transpose(rmse_a)
; rmseallb=transpose(rmse_b)
; rmseallc=transpose(rmse_c)
; rmsealld=transpose(rmse_d)
; rmsealle=transpose(rmse_e)
;输出结果为丰度 PV BS NPV RMSE
PRINT,transpose([[result_a],[rmsealla],[Error_per]]);,transpose([[result_b],[rmseallb]]);,$$
;transpose([[result_c],[rmseallc]]),
; transpose([[result_d],[rmsealld]]),transpose([[result_e],[rmsealle]])
;PRINT,


; writeu,lun,result
; printf,lun,'6630' ;改名称
; printf,lun,'PV BS NPV SHADOW RMSE'
; printf,lun,t_realresult
printf,lun3,transpose([[result_a],[rmsealla],[Error_per]]) ;高斯核函数结果,,
; printf,lun3,transpose(estmodel_pix),format='(1653f12)' ;输出由估算值计算所得模型反射率
; printf,lun3,transpose([[result_b],[rmseallb]])
;; printf,lun,transpose([[result_c],[rmseallc]])
; printf,lun3,transpose([[result_d],[rmsealld]])
; printf,lun3,transpose([[result_e],[rmsealle]])
rmse_aa=rmse_aa+rmse_a
;rmse_bb=rmse_bb+rmse_b^2

;文件输出结果为15、20个是一组数据的结果，这样下个程序可以直接调用TXT文本，每间隔一定数量行为一组新的实验结果数据
endfor

total_rmse_a=total(rmse_aa)/rowNum
; total_rmse_b=sqrt(total(rmse_bb)/rowNum)
PRINT,total_rmse_a;,total_rmse_b ;RMSE OF MODEL
free_lun,lun3
```

```
end



;;;基于核函数的全限制最小二乘法矩阵分解
pro
pixel_kfcls_20160314,em,pixels,nb,kernel,result=result,rmse=rmse,R_Error3=R_Error3,pre_pix=pre_p
ix
;usage:
;pixels:array(npixel,nb),������
; em:array(n_ems,nb),������
; nb:�������
;result:array(npixel,n_em),������
; rmse:array(npixel)
COMPILE_OPT idl2
;
M=TRANSPOSE(pixels);(N*nb)

U=TRANSPOSE(em); (n_em*nb)



IF SIZE(M, /n_dimensions) NE 2 THEN BEGIN
mytemp = DIALOG_MESSAGE('U must be 2D data')
return
ENDIF
IF SIZE(U, /n_dimensions) NE 2 THEN BEGIN
mytemp = DIALOG_MESSAGE('U must be 2D data')
return
ENDIF

dims_U = SIZE(U, /dimensions)
dims_M = SIZE(M, /dimensions)
;M ��  U ������
; IF dims_U[1] NE nb THEN BEGIN
; mytemp = DIALOG_MESSAGE('The spectral bands of Input endmember is wrong', /error)
; return
;ENDIF
;IF dims_M[1] NE nb THEN BEGIN
; mytemp = DIALOG_MESSAGE('The spectral bands of Input pixels is wrong', /error)
; return
;ENDIF

p = dims_M[1] ;像元矩阵的行数
N = dims_M[0] ;像元矩阵的列数
q = dims_U[0] ;端元矩阵的行数
X = FLTARR(N, q) ;求出矩阵的大小
```

```
Mbckp=U

FOR n1=0L,N-1 DO BEGIN

count = q
done = 0
ref = INDGEN(q)

r = M[n1,*]

U = Mbckp


WHILE done DO BEGIN
KWW1=kernel_matrix_kfcls(transpose(U),transpose(U),kernel=kernel);MAKE_ARRAY(q, q, /INTEGER,
VALUE = 1)
kwv=kernel_matrix_kfcls(transpose(U),transpose(r),kernel=kernel)

; KWW1=KERNEL_GS_F(U,U,kernel)
; kwv=KERNEL_GS_F(r,U,kernel)

; KWW1=KERNEL_MU_F(U,U)
; kwv4=KERNEL_MU_F(r,U)
; Kwv=transpose(kwv4[0,*])
EME=INV_k(KWW1)
als_hat =EME##kwv
ones = INTARR(1,count)
ones[*,*] = 1
s = TRANSPOSE(ones)#EME

;□□□□□□□□□□□□□□□□□□
; afcls_hat=als_hat-EME##(kwv-KWW1##als_hat)
afcls_hat=als_hat-
(als_hat#TRANSPOSE(ones)-1)#(INV_k(ones#EME#TRANSPOSE(ones)))#TRANSPOSE(ones)#EME

; ���öÏ����□��
alpha = FLTARR(1,q)
IF TOTAL(afcls_hat GT 0) EQ count THEN BEGIN
alpha[ref] = afcls_hat
BREAK
ENDIF

;��□�س��2Ö��s������□��
;ц�����abs(a_ij, s_ij)�� ��alpha���Ƴ�Entry.
idx = WHERE(afcls_hat LT 0)
IF idx[0] LT 0 THEN idx = 0;!null = MIN(afcls_hat, idx)
afcls_hat[idx] = afcls_hat[idx]/s[idx]
maxIdx = (SORT(-ABS(afcls_hat[idx])))[0]
maxIdx = idx[maxIdx]
alpha[maxIdx] = 0
idx_all = INDGEN(1, (SIZE(U, /dimensions))[0])
```

```
keep = idx_all[WHERE(idx_all NE maxIdx)]
U = U[keep, *]
count = count - 1
IF count EQ 0 THEN BREAK
ref = ref[keep]

ENDWHILE

X[n1,*] = alpha

endfor
result=x
U = Mbckp
R_Error2 =(x#U-pixels)^2
RMSE = FLOAT(SQRT(FCLS_MEAN_k(R_Error2, DIMENSION=2)))

R_Error3=mean(abs(X#U-pixels)/pixels) ;输出模型误差百分比
pre_pix=X#U ;输出估算后各波段反射率值


end

FUNCTION INV_k, data

;
COMPILE_OPT idl2
IF SIZE(data, /n_dimensions) LT 2 THEN BEGIN
RETURN, 1/data
ENDIF ELSE BEGIN
RETURN, INVERT(data)
ENDELSE

END
FUNCTION FCLS_MEAN_k, X, DIMENSION=dimension, DOUBLE = Double, NAN = nan

COMPILE_OPT idl2, hidden
ON_ERROR, 2

IF KEYWORD_SET(dimension) THEN BEGIN
xdims = SIZE(X, /DIMENSION)
IF (N_ELEMENTS(dimension) GT 1 || $
dimension[0] LT 1 || dimension[0] GT N_ELEMENTS(xdims)) THEN BEGIN
MESSAGE, 'Illegal keyword value for DIMENSION.'
ENDIF
IF (KEYWORD_SET(nan)) THEN BEGIN
nX = TOTAL(FINITE(x, /NAN), dimension[0], /INTEGER)
ENDIF ELSE BEGIN
nX = xdims[dimension[0]-1]
ENDELSE
RETURN, TOTAL(X, dimension, DOUBLE=double, NAN=nan)/nX
ENDIF ELSE BEGIN
```

```
IF (KEYWORD_SET(nan)) THEN BEGIN
nX = TOTAL(FINITE(x, /NAN), /INTEGER)
ENDIF ELSE BEGIN
nX = N_ELEMENTS(x)
ENDELSE
RETURN, TOTAL(X, DOUBLE=Double, NAN=nan)/nX
ENDELSE
END

function KERNEL_GS_F,V,W,sigma ;X,Z为变量, d为参数
dims_W= SIZE(W, /dimensions)
dims_V= SIZE(V, /dimensions)
M=dims_W[0]
N=dims_v[0]
VM = FLTARR(M,N)
for i=0,M-1 DO BEGIN
for j=0,N-1 DO BEGIN
VM(i,j)=(NORM(V(j,*)-W(i,*)))^2
endfor
endfor
KVM=EXP(-VM/(2*(sigma^2)))
return, KVM

END

function KERNEL_MU_F,V,W ;V,W为变量,
;KVM1=TRANSPOSE(V)##W ;线性核函数
; KVM1=TRANSPOSE(V)##W##TRANSPOSE(V)##W ;多项式
KVM1=((TRANSPOSE(V)##W)+1)##(TRANSPOSE(V)##W+1) ;多项式，有的文章中直接平方，有的是4个矩阵相乘，不确定哪种是对的了

return,KVM1
END

function
kernel_matrix_kfcls,G1,G2,KERNEL=kernel,GMA=gma,NSCALE=nscale,DEGREE=degree,BIAS=bias
if n_params() eq 1 then G2 = G1
if n_elements(nscale) eq 0 then nscale = 1.0
if n_elements(kernel) eq 0 then kernel = 1
if n_elements(degree) eq 0 then degree = 2
if n_elements(bias) eq 0 then bias = 1.0
case kernel of
0: begin;;线性核函数
gma = 0.0
return, G1##transpose(G2)
end
1: begin ;高斯核函数
m = n_elements(G1[0,*])
n = n_elements(G2[0,*])
K = transpose([total(G1^2,1)])##(intarr(n)+1)
K = K + transpose(intarr(m)+1)##total(G2^2,1)
```

```
K = K - 2*G1##transpose(G2)
if n_elements(gma) eq 0 then begin
scale=60 ;目前证明是最优的 4EM 用60最好, 3EM用20最好
; scale = total(sqrt(abs(K)))/(m^2-m+0.00001)+0.0001
; nscale=50
gma = 1/(2*(nscale*scale)^2)
endif
return, exp(-gma*K)
end
2: begin ;多项式核函数
if n_elements(gma) eq 0 then gma=1.0/n_elements(G1[*,0])
return, (gma*G1##transpose(G2)+bias)^2;##(gma*G1##transpose(G2)+bias);^degree
end
3: begin;神经网络核函数
K = G1##transpose(G2)
if n_elements(gma) eq 0 then gma = 1/mean(abs(K))
return, tanh(gma*K+bias)
end
endcase
end
```