Manual CIFOO v1.00
COALA/Eucalyptus IFU simulador
coala.ferrari.pro.br

Fabricio Ferrari
fabricio(a)ferrari.pro.br

9 de fevereiro de 2004

1 Objetivo e Utilidade

O objetivo do programa é simular os dados do espectrógrafo multifibra (IFU) Eucalyptus, protótipo do IFU do telescópio SOAR (www.soartelescope.org) que está em testes no OPD/LNA (www.lna.br).

Com os dados sintéticos do COALA/CIFOO é possível variar os parâmetros de entrada e saída como não se pode fazer na prática. Deste modo poder-se-á verificar quais são os limites práticos (amostragem, resolução, sensibilidade) antecipadamente à construção de instrumentos deste tipo, não só no caso do SOAR.

Por outro lado, será possível testar o software de redução dos espectros IFU, que dada a complexidade e abundância dos dados crus, as rotinas automatizadas de redução dos dados são tão importantes quanto o próprio instrumento. O COALA/CIFOO serve de simulador para o LIFU[1]

2 Instalação

O CIFOO é inteiramente escrito em Python 2 (www.python.org), com alguns adendos que tornam a vida mais fácil.

[1]Visite www.astro.ufsc.br/~sifus ou informe-se sobre o LIFU com Antônio Kanaan em kanaan@astro.ufsc.br

2.1 Python 2

O CIFOO precisa do Python 2 para funcionar. Em alguns sistemas coexistem duas versões de Python, a 1.5 e a 2.2. Em geral, o executável `python` é um elo simbólico para o `python1.5`. Assim, sempre que for executar o `python` explicite `% python2`. O mesmo vale para os programas escritos em Python: se a primeira linha do arquivo contiver somente `#!/usr/bin/python` mude-a para `#!/usr/bin/python2`. O CIFOO já vem ajustado para usar Python2.

Se você não o tem instalado, pode consegui-lo em www.python.org para várias plataformas (prefira Linux). Um arquivo RPM também pode ser obtido diretamente de www.rpmfind.net, pronto para ser instalado em Linux baseados em pacotes RPM. Se estiver num Debian, basta fazer `apt-get install python`

2.2 Numarray

O Numarray (numpy.sourceforge.net) é um módulo que facilita trabalhar com matrizes. Ele é derivado do Numeric Python e deverá substituí-lo. Para instalá-lo, depois de baixá-lo do link acima, basta fazer como superusuário (supondo que a versão em questão seja a 0.3.6)

```
% tar xzvf numarray-0.3.6.tgz
% cd numarray-0.3.6
% su
% python2 setupall.py
```

e ele deverá instalar-se na sua distribuição de Python. Se você tem a sorte de estar num sistema Debian (www.debian.org) basta fazer como superusuário

```
% apt-get install python-numarray
```

e o pacote será instalado.

2.3 Árvore CIFOO

Depois disso, basta descompactar a árvore do COALA e o CIFOO está pronto para ser usado. No diretório mais baixo do COALA, haverá os seguintes diretórios:

- `./docs/manual` contém este manual nos formatos LaTeX(original), dvi, ps, pdf, txt, html. Há também um `Makefile` que reconstrói a documentação a partir do original LaTeX.

- `cifoo` contém o programa em vários módulos python. Para executá-lo, chame `cifoo`, que uma breve ajuda será mostrada na tela. O CIFOO depende dos arquivos `.py` que estão neste diretório, além do subdiretório `estrelas_padrao` que contém os espectros dos vários tipos espectrais.

- `./docs/análise` contém uma breve descrição física do que está sendo feito, mais num estilo de uma colcha de retalhos do que propriamente um texto acabado. Em especial a parte da dispersão de luz está bem explicada.

- `./docs/reunioes` contém uma versão crua das atas das reuniões sobre o CIFOO.

- `./sitio` é um espelho do site do COALA coala.ferrari.pro.br.

3 Uso

O CIFOO pode ser usado de duas maneiras, explicadas abaixo. Detalhes sobre os comandos de linha, digite 'cifoo' sem parâmetros.

Ao executar, o CIFOO criará um diretório chamado `simul_sn30_a3.0_f2.0` indicado a relação sinal/ruído SN (=30), a distância entre as fibras (=3.0) e a máxima largura a meia altura FWHM das gaussianas das fibras (=2.0).

Dentro deste diretório estarão todos os arquivos resultantes. Os nome base dos arquivos gerados numa simulação estão listados na Tabela 1. Neste caso, o tipo espectral escolhido foi `A5-V7`. Os arquivos de extensão `.dat`[2] são a saída original do CIFOO, com um cabeçalho especificando os parâmetros relevantes e em seguida um vetor de com os valores de cada pixel, começando pelas linhas, depois colunas. Os arquivos `.fits` são convertidos pelo `ascii2fits` a partir dos `.dat`. Os arquivos `ifu.map` e `soar_ifu.cfg` são gerados para integrar o CIFOO com LIFU (veja Seção 4).

3.1 Interativo

O modo interativo foi desenvolvido para ser didático para novos usuários do CIFOO. A tendência é que desapareça no futuro e que seus resultados sejam mais

[2]Malogrado este nome. Todos arquivos são de dados, embora uns no formato texto/ASCII e outros em formato FITS/binário. De qualquer forma, é uma longa equivocada tradição chamá-los de `dat`.

Nome base	descrição
`A5-7V`	espectro IFU da estrela
`A5-7V_lentes`	imagem da estrela nas microlentes
`flat`	flat IFU (lentes igualmente iluminadas)
`mask-00-00`	flat de cada 5 fibras, começando na 0
`mask-00-01`	flat de cada 5 fibras, começando na 1
`mask-00-02`	flat de cada 5 fibras, começando na 2
`mask-00-03`	flat de cada 5 fibras, começando na 3
`mask-00-04`	flat de cada 5 fibras, começando na 4
`ifu.map`	arquivos de configuração para o LIFU
`soar_ifu.cfg`	
`simul_sn30_a3.0_f2.0.log`	arquivo com um relatório da simulação
`transmis`	arquivo com a transmissividade das fibras

Tabela 1: Nome base dos arquivos gerados numa simulação

imprevisíveis que no modo automático. De qualquer modo, utilize-o se quiser entender os vários parâmetros de entrada do CIFOO.

O modo interativo é especificado com

```
% cifoo -i
```

ou

```
% cifoo --interativo
```

Desta maneira, todos os parâmetros são perguntados ao usuário de maneira auto-explicativa. As perguntas feitas são

1. modo (objeto; máscara 0, 1, 2, 3 ou 4; flat)
2. posição
3. seeing
4. tipo espectral
5. distância entre as fibras
6. dispersão nas fibras

3.2 Automático

No modo automático os parâmetros são colocados num arquivo de inicialização (não confundir com arquivo de configuração; eu às vezes me confundo). Existe um modelo deste arquivo junto da distribuição chamado `cifoo_init_exemplo.py`. A sintaxe deste arquivo também pode ser mostrada pelo CIFOO com o parâmetro `-I`

```
% cifoo -I


#### Arquivo de inicializacao do Coala/Cifoo ####

#### Objeto ####
xs=2.0       # centro do objeto (0<xs<N)
ys=2.0       # centro do objeto (0<ys<M)
seeing=0.1   # seeing do objeto

###  Tipo Espectral ###
# Para colocar o tipo espectral no arquivo de inicializacao do Cifoo,
# use o indice abaixo (0-32) que corresponde ao tipo espectral.
#    00: A1--3V       01: A3III        02: A5--7V
#    03: A6--F0III    04: A7IV         05: A8V
#    06: A9--F0V      07: B3--4V       08: B5III
#    09: B6V          10: F3IV         11: F4-7III
#    12: F6--7V       13: F8--9V       14: G1--2V
#    15: G2IV         16: G5--6III     17: G5IV
#    18: G6--8V       19: G8IV         20: G8--K0III
#    21: G9--K0V      22: GO--4III     23: K2III
#    24: K4III        25: K4V          26: K5V
#    27: K7III        28: M2V          29: M4V
#    30: O5V          31: O7--B0V      32: O7-B1III
#
tipoespec=19

#### Fibras ####
a=3.0          # distancia entre as fibras
fwhmi=3.0      # dispersao das fibras, maxima larguma 1/2 altura (fwhm=2.35*sigma)

### RUIDO NO CCD ###
# Tipo de ruido
# (modulo random python)
#
#    uniforme = aleatorio em torno do valor fluxo do espectro entre (F-sigmar, F+sigmar)
#                    F_com_ruido = F * (1+ sigmar*(2*random()-1))
#               random()                    tem imagem entre 0 e 1
#               (2*random()-1)              tem imagem entre -1 e 1
#               sigmar*(2*random()-1)       tem imagem entre -sigmar e +sigmar
#               F*[1+sigmar*(2*random()-1)] tem imagem entre F-sigmar e F+sigmar
#
#      gauss  = distribuicao normal em torno do fluxo do objeto com dispersao sigmar*F
#               random.gauss(F,sigmar) retorna valores de uma distribuicao normal
#               centrada em F e com dispersao sigmar
#
tipor  = "uniforme"
#
# Limites do ruido, dada como fracao do fluxo
# Sinal/Ruido SN=1/sigmar
sigmar = 0.033333
```

Um arquivo com o conteúdo acima é gravado com a extensão `.py` (Python) e então seu nome fornecido ao CIFOO, da seguinte forma (assumindo que o nome escolhido para o arquivo seja `config_init.py`):

```
% cifoo -a config_init.py
```

ou então

```
% cifoo --auto-file=config_init.py
```

É importante que o nome do arquivo de inicialização seja terminado por `.py`[3] para que as variáveis do arquivo sejam incorporadas ao espaço de nomes principal do CIFOO. Os seguintes parâmetros que precisam ser especificados

1. posição do objeto (xs,ys) na matriz de lentes
2. seeing do objeto
3. tipo espectral
4. distância entre as fibras
5. dispersão (fwhm) nas fibras
6. ruído do sistema
7. a tranmissividade das fibras.

O tipo e a escala do ruído incorporado aos fluxos no CCD pode ser aleatório (`uniforme`), em que os valores estão uniformemente distribuídos de `sigmar` a mais ou a menos que o fluxo do objeto naquela posição, ou gaussiano (`gauss`), em que o ruído tem distribuição normal com média no fluxo do objeto e dispersão `sigmar`.

Atenção No modo automático, todos os arquivos da Tabela 1 são gerados. No modo interativo, cada um deles precisa ser gerado manualmente.

3.3 Arquivo de configuração

O CIFOO lê os parâmetros essenciais num arquivo chamado `cifoo_config.py`. Neste arquivo estão especificados as configurações que não mudam ao longo de várias simulações, por isso coloquei num arquivo separado. Os parâmetros são os seguintes:

- As dimensões em X (N) e Y (M) da matriz de microlentes.

[3] Esta é uma exigência do `Python`, não minha `:-)`

- O tamanho da imagem do CCD em x (`CCDx`). Este tamanho é quantas linhas de dados serão usadas do espectro da estrela, começando em 3505 A e indo em passos de 5 A em cada linha. Em geral é da ordem de algumas dezenas porque o interesse maior é na contaminação espacial, e não espectral, embora o limite seja `CCDx`<1000, pois são 1000 os pontos que há nos espectros.

- O número de fibras vizinhas no CCD (`Nvizinhos`) de quem deverá somar as contribuições ao avaliar a intensidade em uma dada posição, quando calculando a imagem. Para tornar o algoritmo mais eficiente, a contribuição de uma fibra muito distante não é incluída numa dada posição do CCD. `Nvizinhos` é quão longe vai esta tua política de boa vizinhança.

O arquivo é como o que segue:

```
### arquivo cifoo_config.py ###

# Se e' para imprimir tudo que faz
# ou somente os passos basicos
verbose=0

### Matriz de lentes ###
N  = 5       # X colunas da matriz de fibras
M  = 5       # Y linhas da matriz de fibras
NF = M * N   # numero de fibras

### Imagem no CCD ###
# Numero de pixies na direcao X da imagem do CCD
#
CCDx = 25  #<<<<< deve ser menor que 1000 <<<<<

### VIzinhos ###
# quantos vizinhos contrinuem numa dada posicao z
Nvizinhos = 5

### Transmissividade ###
# vetor com transmissividades de cada fibra, em ordem.
# se vazio [] sao escolhidas trans. aleatorias entre tranmin e transmax,
# senao, especificadas na ordem e no mesmo numero de fibras =[0.3, 0.4, ...] NF vezes
transmin=0.3 # minimo da transmissividade (>0)
transmax=1.0 # maximo da transmissividade (em geral =1)
transmis=[]  # vetor com as transmissividades
```

3.4 Modo mais-que-automático – GERSIMUL

Existe um pequeno utilitário chamado GERSIMUL que serve para produzir automaticamente os arquivos de inicialização de uma série de simulações e rodar o CIFOO em cada um deles. É prático para o caso de se querer rodar uma série de simulações alterando gradativamente os parâmetros de cada uma delas.

Para executar o GERSIMUL, componha um arquivo como mostrado abaixo (há um chamado `gersimul_init_exemplo.py` de exemplo)

```
##### PARAMETROS VARIAVEIS  #####
```

```
# ESPACAMENTO ENTRE AS FIBRAS (a)
# intervalo de (a)
a = [1, 2, 3 ]

# MAXIMA LARGURA A MEIA ALTURA DAS FIBRAS fwhm
# fwhm=2.35*sigma
# intervalo de (fwhm)
fwhm = [3, 4, 5]

### SINAL RUIDO (SN) ###
# intervalo de (ruido)
sn = [10, 100, 1000]


##### PARAMETROS FIXOS #####

# RUIDO - tipo (uniforme|gauss)
tipor = "uniforme"

## Objeto ##
xs=2.0       # centro do objeto (0<xs<N)
ys=2.0       # centro do objeto (0<ys<M)
seeing=0.1   # seeing do objeto

## Tipo Espectral ##
tipoespec=19
```

A sintaxe dos parâmetros variáveis especifica quais valores estes devem assumir. Em se rodando o GERSIMUL com o arquivo de parâmetros (p.ex. `gersimul parametros.py`), seriam criados arquivos de inicialização para $a = 1, 2, 3$, $fwhm = 3, 4, 5$, $SN = 10, 100, 1000$, ou seja, 27 simulações diferentes. Para manter um dos valores fixos, basta restringir em um só número o seu conjunto, p.ex. $a = [7]$

O GERSIMUL então cria os diretórios apropriados e dentro grava os arquivos de inicialização para cada um. Depois disso, grava um script do shell chamado `gersimul_executa.sh` que contém os comandos para rodar o CIFOO em cada um deles. Basta executá-lo e esperar que todas as simulações sejam completadas.

3.5 Unidades

Um dos problemas que deve ter chamado a atenção do usuário é o fato de não haver referências às unidades físicas. De fato, elas não são importantes no estudo da dispersão nas lentes e depois nas fibras. No caso das lentes, a convolução atmosférica (*seeing*) deve ser especificada em termos da distância entre as fibras. Por exemplo, um *seeing* de 2 significa de tamanho de 2 lentes. Da mesma forma no caso das fibras: a dispersão têm sentido quanto comparada com a separação entre as fibras. Entretanto, como notado por A. Kanaan, uma dispersão de 1 numa separação de 5 não é a mesma coisa que uma dispersão de 2 numa separação de 10, embora a razão seja a mesma. O mesmo se aplica ao *seeing* com relação às microlente. A diferença entre os dois casos é a amostragem. No domínio contínuo seriam idênticos, mas no discreto não são.

4 Interação com o LIFU

Para que utilizar os programas de redução de dados LIFU, desenhado para os dados da IFU Eucalyptus, com os dados sintéticos do COALA/CIFOO é necessário fazer alguns ajustes. Os arquivos que precisam ser modificados são `ifu.map` e `soar_ifu.cfg` que estão localizados no diretório `$SOAR_IFU/conf/`, onde `SOAR_IFU` é a raiz da estrutura de diretórios que compõem o LIFU. Este arquivo precisa ser ajustado antes de executar o LIFU.

Existe um modo de execução do CIFOO que gera estes arquivos, basta executá-lo

```
% cifoo -L
```

ou

```
% cifoo --lifu
```

Ele pega os parâmetros necessários do arquivo `cifoo_config.py` em voga. No modo automático, o CIFOO gera-os automaticamente, colocando-os dentro do diretório dos dados.

4.1 ifu.map

O arquivo `ifu.map` é um mapa da localização das fibras no CCD. A primeira linha contém o número de fibras, em seguida o número de blocos de fibras e finalmente os blocos ordenados e as fibras que os compõem.

Por exemplo, para uma situação típica da IFU Eucalyptus o arquivo `ifu.map` seria

```
510
16
00 30
01 32
02 32
03 32
04 32
05 32
06 32
```

```
07 32
08 32
09 32
10 32
11 32
12 32
13 32
14 32
15 32
```

No caso do CIFOO, para uma situação de 25 fibras, num arranjo de 5x5 fibras, o arquivo será

```
25
1
00 25
```

ou seja, 25 fibras arranjadas em 1 só bloco (00 25) no CDD.

No caso do Eucalyptus, a cada 32 fibras mais ou menos, há um espaço no CDD. Nos dados sintéticos do CIFOO não, assim o arquivo `ifu.conf` em geral contém a especificação de um só bloco com o número total de fibras, do tipo (NF=MxN é o número total de fibras)

```
NF
1
00 NF
```

4.2 `soar_ifu.cfg`

Este arquivo contém um mapeamento de qual fibra no CCD corresponde à fibra na matriz de microlentes. Ele especifica, na verdade, a maneira em que as fibras que saem da matriz de lentes são alinhadas na entrada do espectrógrafo. Assim como o `ifu.map`, o CIFOO também gera-o no modo LIFU, colocando no diretório dos dados.

No caso da matriz de 5x5 fibras o arquivo fica assim:

```
  0   0
  0   1
  0   2
  0   3
  0   4
  1   0
  .   .
  .   .
  .   .
```

```
4   1
4   2
4   3
4   4
```

5 Ultimas palavras

Deverá ser fácil, com ajuda deste manual, simular a IFU Eucalyptus com o COALA/CIFOO. De qualquer forma, comentários/sugestões/dúvidas/melhorias podem ser enviados a fabricio AT ferrari.pro.br.