



Universidade Federal do Rio de Janeiro  
Observatório do Valongo

# **IDL, PAHFIT e PAHdb**

OVL715

Tópicos de Tratamentos de Dados Extragalácticos  
Rayssa Guimarães Silva

# Sumário

<b>1</b>	<b>Instalação do IDL</b>	<b>2</b>
1.1	Passos da instalação . . . . .	2
1.2	Configurações pós-instalação . . . . .	2
1.2.1	Incluir tudo no .bashrc: . . . . .	2
1.2.2	O <i>startup file</i> : . . . . .	3
1.3	Biblioteca de funções astronômicas . . . . .	3
1.4	Kit de sobrevivência: coisas para saber . . . . .	3
<b>2</b>	<b>PAHFIT</b>	<b>4</b>
2.1	Instalação . . . . .	4
2.2	A função <i>pahfit</i> . . . . .	4
2.3	Decompondo um espectro . . . . .	5
2.4	Output do PAHFIT . . . . .	5
2.5	Decompondo vários espectros . . . . .	7
<b>3</b>	<b>PAHdb</b>	<b>9</b>
3.1	Instalação . . . . .	9
3.2	Ajustando um espectro . . . . .	9
3.3	Output do PAHdb . . . . .	11
3.4	Analisando o output no Python . . . . .	11

# 1 Instalação do IDL

## 1.1 Passos da instalação

Os passos da instalação supõem que você está usando algum sistema UNIX.

**1. Descompactar o arquivo de instalação:** O arquivo de instalação está compactado, para descompactar

```
:$ gunzip nome_do_arquivo.tar.gz
:$ tar -xf nome_do_arquivo.tar
```

Você pode usar o comando `ls` para verificar se os arquivos estão corretos, o resultado deverá ser uma pasta e um arquivo, respectivamente, `install` e `install.sh`.

**2. Criar o diretório de instalação:** Antes de instalar, é necessário escolher um local para instalar o IDL em seu sistema e criar um diretório. O caminho de instalação padrão é `/usr/local/itt`. Após criar o diretório, é necessário também certificar-se que as permissões estão corretas:

```
:$ mkdir /usr/local/itt #cria o diretório
:$ chmod a+rx /usr/local/itt #dá permissões
```

**3. Executar o script de instalação:** Finalmente, para instalar, basta executar o seguinte comando no terminal e ir respondendo às perguntas da instalação

```
:$ ./install.sh #executa o script de instalação
```

## 1.2 Configurações pós-instalação

Antes de começar a usar o IDL, é necessário configurar algumas variáveis de ambiente

- **ITT\_DIR:** caminho completo para o diretório do ITT
- **IDL\_DIR:** caminho completo para o diretório do IDL
- **IDL\_PATH:** caminho completo para o diretório contendo os arquivos `.pro` (funções, rotinas, bibliotecas etc.)

### 1.2.1 Incluir tudo no `.bashrc`:

No seu diretório `home` existe um arquivo oculto de texto chamado `.bashrc` que é lido toda vez que você abre um terminal. Nesse arquivo é possível incluir variáveis de ambiente, aliases e até pequenos comandos para que eles estejam disponíveis para você no terminal. Há duas opções nesse caso, incluir as seguintes linhas no `.bashrc`: usando um arquivo padrão ou definindo as variáveis de forma explícita.

Arquivo padrão:

```
#importa o arquivo que contém algumas definições padrão do IDL
source /usr/local/itt/idl/idl81/bin/idl_setup.bash

#exporta a variável IDL_PATH
export IDL_PATH='+/usr/local/itt/idl/idl81/lib':$IDL_PATH
```

Definição explícita:

```
#caminhos IDL
export ITT_DIR=/usr/local/itt
export IDL_DIR=/usr/local/itt/idl/idl81/

#exporta a variável IDL_PATH
export IDL_PATH='+/usr/local/itt/idl/idl81/lib':$IDL_PATH
```

A variável **IDL\_PATH** pode ser repetida diversas vezes dentro do arquivo `.bashrc`, sempre que desejarmos adicionar uma nova pasta que contenha arquivos `.pro`. O “+” significa que estamos incluindo o diretório e todas as suas subpastas.

### 1.2.2 O *startup file*:

Se você tiver alguma lista de comandos ou variáveis que devem estar disponíveis apenas dentro do IDL, crie um *startup file*, comumente nomeado de `.idl_startup` e crie uma variável de ambiente específica **IDL\_STARTUP** e a inclua no seu `.bashrc`:

```
export IDL_STARTUP=caminho/para/seu/startup/file
```

## 1.3 Biblioteca de funções astronômicas

existe uma coleção de funções que são comuns na Astronomia que podem ser baixadas em <https://idlastro.gsfc.nasa.gov/>. É interessante também instalar a biblioteca Coyote de funções gráficas em [http://www.idlcoyote.com/documents/programs.php#COYOTE\\_LIBRARY\\_DOWNLOAD](http://www.idlcoyote.com/documents/programs.php#COYOTE_LIBRARY_DOWNLOAD). Em ambos os casos, basta extrair os arquivos e incluir o caminho dos diretórios na variável **IDL\_PATH**.

## 1.4 Kit de sobrevivência: coisas para saber

- Abrir a linha de comando do IDL: digitar `idl` no terminal
- O IDL não diferencia maiúsculas e minúsculas em nomes de variáveis, então não precisa se preocupar com isso!
- O símbolo de comentário é “;”
- Programa deu algum erro e você não consegue dar novos comandos? Tente o `retall`
- Digitar `$` na frente de um comando faz com que ele seja executado fora do IDL (por exemplo, podemos digitar `$ ls`)
- Referência muito legal para uma introdução geral ao IDL [https://www.dartmouth.edu/~rc/classes/idl\\_intro/](https://www.dartmouth.edu/~rc/classes/idl_intro/).

## 2 PAHFIT

### 2.1 Instalação

O PAHFIT vem “pronto para uso”, bastando colocá-lo em algum lugar que o compilador de IDL seja capaz de encontrar. Para isso, basta incluir o caminho do diretório que o contém na variável de ambiente `IDL_PATH`. O jeito mais fácil é adicionar a linha

```
export IDL_PATH='+/caminho/do/pahfit':$IDL_PATH
```

no arquivo `.bashrc`.

### 2.2 A função `pahfit`

```
pahfit(lambda, flux, flux_uncertainty, [REDSHIFT=,  
    STARLIGHT_TEMPERATURE=,  
    CONTINUUM_TEMPERATURES=, LINES=, DUST_FEATURES=,  
    /PLOT_PROGRESS, /NO_EXTINCTION, /NO_FIT, PARINFO=, /NO_FIX,  
    /VALUES_ONLY, /NO_MEGAJANSKY_SR, REPORT=, _EXTRA=e])
```

Parâmetros de entrada obrigatórios

- **lambda**: um array de comprimento de onda em microns
- **flux\***: intensidade de fluxo em MJy/sr (ou outra unidade)
- **flux\_uncertainty**: incerteza da intensidade de fluxo. Apesar de o programa dizer que esse parâmetro é opcional e se não for passado, peso igual será dado para cada ponto, a experiência mostra que o programa dá erro.

Algumas observações e comentários sobre as palavras-chave opcionais

- **REDSHIFT**: tem que estar em km/s (i.e., cz)
- \* **NO\_MEGAJANSKY\_SR**: se as unidades de **flux** não forem MJy/sr, esse parâmetro deve ser incluído no input. A inclusão evita conversões errôneas no meio do programa.
- **PLOT\_PROGRESS**: controla se desejamos ou não ver o progresso do ajuste em uma janela gráfica
- **NO\_EXTINCTION**: fixa a profundidade óptica em 0. **REPORT**: se colocarmos o nome de um arquivo, salva os parâmetros do ajuste do PAHFIT nesse arquivo. Se usado na forma `/REPORT`, mostra os parâmetros no terminal apenas.
- **STARLIGHT\_TEMPERATURE**: é a temperatura usada para a emissão de corpo-negro associada ao contínuo estelar (por padrão é 5000 K).
- **CONTINUUM\_TEMPERATURES**: é o array de temperaturas usada para ajustar o contínuo da poeira (padrão é 300.D,200.D,135.D,90.D,65.D,50.D,40.D,35.D).

Uma descrição dos outros parâmetros — e mais detalhes em geral — pode ser encontrada na parte inicial do arquivo `pahfit.pro`.

## 2.3 Decompondo um espectro

O PAHFIT é bem direto para o ajuste, basta dar três inputs: um array de comprimento de onda, o de fluxo e o de incerteza. O comando abaixo é um exemplo de como ler uma tabela com três colunas pulando duas linhas de comentário e usando os arrays resultantes no PAHFIT:

```
IDL > cd, '/path/to/spectra'  
IDL > rdffloat, 'spectrum.txt', lambda, flux, error, SKIPLINE=2  
IDL > fit=pahfit(lambda, flux, error, /PLOT_PROGRESS, REPORT='pahfit.txt')
```

**Dica:** dependendo do formato do seu arquivo de entrada, o comando de leitura vai ser um pouco diferente. Por exemplo, arquivos no formato IPAC<sup>1</sup> têm essa aparência:

```
232 \ Processing Info  
233 \SOFTWARE = 'irs_merge v2.1' / program that created this file  
234 \ORIGIN = 'SSC - Spitzer Science Center' / this file created by  
235 \CREATOR = 'S18.18.0' / pipeline version of 1st merged file  
236 \DATE = '2011-12-08 19:36:18' / when this file created (Pacific)  
237 \ENHID = 12760 / Spitzer database Enhanced Product ID  
238 \CAMPID = 1094 / Spitzer database Campaign ID  
239 \FILENAME = 'SPITZER_S5_23012096_01_merge.tbl'  
240  
241 |order|wavelength|flux_density |error |bit-flag|  
242 |int |real |real |real |int |  
243 | |microns |Jy |Jy | |  
244 2 7.57612 0.001772 0.000302 0  
245 2 7.63660 0.002925 0.000281 0  
246 2 7.69709 0.004055 0.000289 0  
247 2 7.75757 0.004515 0.000276 0  
248 2 7.81805 0.003938 0.000267 0
```

Ao invés de converter a tabela externamente, podemos reescrever o comando de forma que ele pule todas as 243 linhas de comentários e também leve em conta que temos 5 colunas:

```
IDL > rdffloat, 'spectrum.txt', order, lambda, flux, error, flag, SKIPLINE=243
```

As colunas `order` e `flag` não serão utilizadas pelo PAHFIT, mas devem ser lidas para evitar erros.

## 2.4 Output do PAHFIT

Abaixo está um trecho do REPORT do PAHFIT, ele contém os valores finais dos parâmetros do ajuste. Além disso, o PAHFIT também gera um arquivo `components_model.dat` que contém diversas colunas com o “espectro” de cada componente. O PAHFIT também cria um arquivo que

<sup>1</sup>É um formato ASCII muito comum na Astronomia, em especial para dados do Spitzer. Mais sobre o formato em: [https://irsa.ipac.caltech.edu/applications/DDGEN/Doc/ipac\\_tbl.html](https://irsa.ipac.caltech.edu/applications/DDGEN/Doc/ipac_tbl.html)

contém as intensidades de cada componente em cada comprimento de onda, na próxima seção há um trecho de código para transformar esse arquivo em uma tabela mais fácil de ler e manipular.

```
=====
PAHFIT v1.2 JD Smith & Bruce Draine
Tue Dec  1 15:36:11 2020
=====

Reduced Chi-Square (statistical errors only):    0.339

Starlight:
T_star: 5.00e+03  tau_star: 2.31e-15 ( 3.74e-15 )

Dust Continuum:
T_dust:    300.  tau_dust: 1.30e-13 ( 6.16e-12 )
T_dust:    200.  tau_dust: 4.18e-11 ( 6.27e-11 )
T_dust:    135.  tau_dust:    0.00 (    0.00 )
T_dust:    90.0  tau_dust: 3.18e-10 ( 7.94e-08 )
T_dust:    65.0  tau_dust: 3.40e-07 ( 4.27e-06 )
T_dust:    50.0  tau_dust:    0.00 (    0.00 )
T_dust:    40.0  tau_dust:    0.00 (    0.00 )
T_dust:    35.0  tau_dust:    0.00 (    0.00 )

Lines:
H2 S(7) lam:    5.511 (    0.000 ) cen_inten: 0.00106 (    0.00 )
fwhm:  0.0530 (    0.00 )    power: 5.90e+08 (    0.00 )
eqw:   0.0678
```

## 2.5 Decompondo vários espectros

O arquivo a seguir pode ser usado para executar o ajuste para diversos espectros e também cria uma tabela com cabeçalho contendo cada componente ajustado.

```
; PROGRAM TO FIT MULTIPLE SOURCES
; PARAMETERS:
;     PATH_TO_SPECTRA: spectra directory's path
;     SPECTRA_EXTENSION: extension of the spectra files
;     SPECTRA: string pattern to search for spectra, default is *
;             (all files with the same extension)
;
; Code written by Rayssa Guimarães Silva - 2020, March
; (based on fit_pahfit from Carla Martinez Canelo- 2015, July)

pro pahfit_all

path_to_spectra = 'path/to/spectra/directory/'
CD,path_to_spectra

spectra_extension = '.tbl'

; You must use the wildcard expression with "*" to search for all files with the
; same extension. The wildcard can be combined with a string pattern,
; e.g, spectra = '*_redshiftcorrected' + spectra_extension will match all
; 'anything_redshiftcorrected.tbl',
spectra = '*' + spectra_extension

files = FILE_SEARCH(spectra,/FULLY_QUALIFY_PATH)

; For every file that matches our previous criteria
foreach element, files, idx do begin

; Extract the name of the file to be saved as output with another extension
name = STRMID(element, 0, STRLEN(element)-STRLEN(spectra_extension))

; Read the file. This command can be changed depending on the structure of the
; file to be read
readcol, element, lambda, flux, error, delimiter=" ", skipline=0, format=' '

; Create the name of the report
report_file = name + '_pahfit.txt'

fit=pahfit(lambda,flux,error,/PLOT_PROGRESS,/NO_MEGAJANSKY_SR,REPORT=report_file
)

; Transform PAHFIT fit result, 'components_model.dat', into a file that is
; easier to understand and read
```



```

readcol,'components_model.dat', lam_mod , cont_mod, feat_mod, lines_mod,
    stars_mod, dust_mod, bestfit_mod, ext_mod
openw,1,name+'_mod_components.dat',WIDTH=250
printf,1,'#',name
; STRING(9B) is the table separator
printf,1,$
'lam_mod',STRING(9B), $
'cont_mod',STRING(9B), $
'feat_mod',STRING(9B), $
'lines_mod', STRING(9B), $
'stars_mod',STRING(9B), $
'dust_mod',STRING(9B), $
'bestfit_mod',STRING(9B), $
'ext_mod'

for k=0,n_elements(lam_mod)-1 do begin
printf,1, lam_mod[k], cont_mod[k], feat_mod[k], lines_mod[k], stars_mod[k],
    dust_mod[k], bestfit_mod[k], ext_mod[k]
endfor

close,1
endforeach
end

```

A saída desse programa é uma tabela com as seguintes colunas

- lam\_mod: comprimento de onda
- cont\_mod: contínuo total
- feat\_mod: bandas dos PAHs
- lines\_mod: linhas atômicas ou moleculares
- stars\_mod: contínuo estelar
- dust\_mod: contínuo da poeira
- bestfit\_mod: modelo
- ext\_mod: extinção

## 3 PAHdb

### 3.1 Instalação

Similar ao PAHFIT, o PAHdb é baseado em um conjunto de funções do IDL no formato .pro. No momento esses arquivos compõem a AmesPAHdbIDLSuite e podem ser baixados na aba Download do site <https://www.astrochemistry.org/pahdb/>.

Além disso, é necessário baixar também a base de dados que desejamos usar, no formato .xml. Na mesma aba de Download, é possível baixar a base teórica ou a experimental completas, mas também é possível baixar um subconjunto menor dessas bases, definido na aba Browse.

PAH IR Spectral Database  
Ames Research Center

Browse Selection Download Tools Results — Help

magnesium=0 oxygen=0 iron=0 silicium=0 chx=0 ch2=0 c>20 hydrogen>0 Search

(e.g., C<=20 N=2 neutral) Advanced

Theory | Experiment — Versions

2962 PAHs matching query, showing PAHs 1 - 50

species per page view selection << previous 1 2 3 4 5 6 ... 60 next >>

C21H13 C21H13 C21H13 C21H13 C21H13  
C21H13 C21H13N C21H13N C21H13N C21H13N  
C21H13N+ C21H13N+ C21H13N+ C22H12- C22H12  
C22H12 C22H12 C22H14 C22H14 C22H14  
C22H14 C22H14 C22H14 C22H14 C22H14  
C22H14 C22H14 C22H14 C22H14+ C22H14+  
C22H14+ C22H14+ C22H14+2 C23H11N C23H13  
C23H13 C23H13 C23H13 C23H12N C23H12N  
C23H11N+ C23H12N+ C23H12N+ C23H12N+ C24H12-  
C24H14- C24H14- C24H12 C24H14 C24H14

Nesse caso, começamos com uma busca no Search, que pode ser feita por um filtro escrito ou indo em opções avançadas e usando caixas de seleção. O resultado para `magnesium=0 oxygen=0 iron=0 silicium=0 chx=0 ch2=0 c>20 hydrogen>0` é a página acima. Nela é possível escolher qual o tipo de base (teoria ou experimental) e também a versão da base de dados. Com isso determinado, vá ao botão cinza de Selection e escolha `all` para selecionar todas as espécies que correspondem aos seus critérios. Após isso, vá para a página de Download e escolha `Selected species` na hora de baixar o arquivo .xml.

Com esse arquivo em mãos, é interessante definir a variável de ambiente que guarda o caminho completo para a base de dados, para que você não precise digitar esse caminho toda vez que for usar o PAHdb — como no Passo 4 da próxima seção. O nome da variável é **AMESPAHDEFAULTDB**.

### 3.2 Ajustando um espectro

**Passo 1:** ler o espectro contido no arquivo “meuarquivo” e transformá-lo em um objeto de observação. Aceita espectros no formato ASCII de duas colunas (comprimento de onda e fluxo),

VOTABLE e ISO/Spitzer-YARR.

A função `AmesPAHdbIDLSuite_CREATE_OBSERVATION_UNITS_S()` é usada para associar as unidades corretas.

```
IDL > observation = OBJ_NEW('AmesPAHdbIDLSuite_Observation', 'seuarquivo',  
Units=AmesPAHdbIDLSuite_CREATE_OBSERVATION_UNITS_S())
```

**Passo 2:** converter as unidades da abscissa para número de onda

```
IDL > observation -> AbscissaUnitsTo,1
```

**Passo 3 (opcional):** rebinnar a observação para uma grade uniforme com espaçamento de  $5 \text{ cm}^{-1}$ . Fazer esse passo altera o número de pontos usados no ajuste e pode afetar adversamente o resultado. Não executar esse passo significa que o espectro de saída do programa terá o mesmo número de pontos do espectro de entrada.

```
IDL > observation -> Rebin,5D,/Uniform
```

**Passo 4:** ler a base de dados do PAHdb de um arquivo. Caso a variável **AMESPAHDEFAULTDB** esteja definida, o parâmetro `Filename` não precisa ser incluído.

```
IDL > pahdb = OBJ_NEW('AmesPAHdbIDLSuite', Filename='caminho/base/xml')
```

**Passo 5:** definir quais moléculas (UIDs) serão usadas no ajuste e obter as transições. O filtro é em forma de string (por exemplo, "mg=0 o=0 fe=0 si=0 chx=0 ch2=0 c>20"). Para buscar todos os UIDs basta escrever "C>0", i.e., todas as moléculas com mais de 0 carbonos. O Shift aplica um deslocamento para simular efeitos de anarmonicidade das transições. Atenção: essa função só deve ser aplicada após a convolução com um perfil de linha!

```
IDL > uids = pahdb -> Search("C>0")  
IDL > pahs = pahdb -> getSpeciesByUID(uids)  
IDL > transitions = pahdb -> getTransitionsByUID(uids)  
IDL > transitions -> Shift,-15D
```

**Passo 6 (opcional):** fixar a temperatura de um modelo em 550 Kelvin ou qualquer outra temperatura.

```
IDL > transitions -> FixedTemperature, 550D
```

**Passo 7:** convoluir as transições com algum perfil de linha determinado pelo tipo de perfil e a FWHM. Aqui usamos o Lorentziano com uma FWHM de  $20 \text{ cm}^{-1}$

```
IDL > spectrum = transitions -> Convolve(/Lorentzian, Grid = observation->
    getGrid(), FWHM=15D)
```

**Passo 8:** realizar o ajuste

```
IDL > fit = spectrum -> Fit(observation)
```

**Passo 9 (recomendado):** destruir objetos que não serão mais utilizados para liberar memória. É uma boa ideia não destruir o objeto pahdb, pois ele pode ser usado para fazer o ajuste com outros espectros de entrada.

```
IDL > OBJ_DESTROY, [observation, spectrum, fit, pahdb]
```

### 3.3 Output do PAHdb

O arquivo de saída do PAHdb é uma tabela com diversas colunas como no trecho abaixo:

```
1 AMESPAHDBIDLSUITE_FITTED_SPECTRUM
2 UID:                27
3 WEIGHT:             0.0160806
4 frequency [cm!U-1!N]  cross-section [x10!U5!N cm!U2!N/mol]
5 682.659             0.000465861
6 687.659             0.000488999
7 692.659             0.000518170
8 697.659             0.000554099
9 702.659             0.000598024
```

A primeira coluna é o número de onda em  $\text{cm}^{-1}$  e as colunas à direita são a contribuição de cada molécula (identificada pelo UID).

### 3.4 Analisando o output no Python

Recentemente o PAHdb começou a ser implementado no Python — mais informações no site do NASA Ames! — e criaram uma função capaz de interagir com a base de dados no arquivo `.xml`, disponibilizada na `AmesPAHdbPythonSuite`<sup>2</sup>. A seguir, um exemplo de como extrair informações a partir dos UIDs que são dados como saída pela `AmesPAHdbIDLSuite`

O primeiro passo é ler o `.xml`, verificar a integridade do arquivo e transformá-lo em um dicionário:

<sup>2</sup><https://github.com/rayssags/AmesPAHdbPythonSuite>

```
In [1]: xml = 'caminho/para/o/xml'
        parser = XMLparser(xml)
        parser.verify_schema()
        library = parser.to_pahdb_dict()
```

Agora, podemos checar quais informações podemos acessar a partir do UID (aqui usei o UID=210):

```
In [2]: library['species'][210].keys()
```

```
Out[2]: dict_keys(['comments', 'references', 'formula', 'charge', 'symmetry',
                  'weight', 'total_e', 'vib_e', 'method', 'n_solo', 'n_duo', 'n_trio',
                  'n_quartet', 'n_quintet', 'n_ch2', 'n_chx', 'geometry', 'transitions'])
```

Portanto, para encontrar a carga, basta fazer:

```
In [3]: library['species'][210]['charge']
```

```
Out[3]: -1
```

Outro parâmetro muito útil é o número total de carbonos. É possível descobri-lo fazendo uma busca na fórmula da molécula:

```
In [4]: import re
        formula = library['species'][210]['formula']
        formula = formula.replace(' ', '')
        if 'H' in formula:
            nc = int(re.sub('[+-]', '', re.findall('C(.+)H', formula)[0]))
        else:
            nc = int(re.sub('[+-]', '', re.findall('C(.+)', formula)[0]))

        nc
```

```
Out[4]: 18
```

```
In [5]: from astropy.table import Table
        pahdb = Table.read("caminho/para/output/do/pahdb", format='ascii',
                          header_start=1, data_start=4)
        pahdb
```

Para ler o arquivo de saída do PAHdb, podemos fazer:

```
Out[5]: UID:          27          481      ...
        float64    float64    float64    ...
        -----  -----  -----  ...
        682.659  0.000465861  0.000887875  ...
        687.659  0.000488999  0.00108891  ...
        692.659  0.00051817   0.0013851  ...
        697.659  0.000554099  0.00184616  ...
```

E convertendo para outras unidades:

```
In [6]: from astropy import units as u
pahdb.rename_column('UID:', 'wavenumber')
pahdb['wavenumber'].unit = 1/u.cm
pahdb['wavelength'] = pahdb['wavenumber'].to(u.um, equivalencies=u.
    spectral())
pahdb
```

```
Out [6]: wavenumber      27          481      ...      wavelength
          1 / cm
          float64      float64      float64      ...      float64
-----
          682.659  0.000465861  0.000887875  ...  14.648602010667114
          687.659  0.000488999  0.00108891  ...  14.54209135632632
          692.659  0.00051817   0.0013851  ...  14.437118408914056
          697.659  0.000554099  0.00184616  ...  14.3336501070007
```