Versão mais recente do programa com entrada de múltiplos elementos

Arquivos

Arquivos necessários e gerados durante a execução do programa.

Arquivo de entrada

Arquivo txt com as variáveis do NJOY e seus valores que serão usadas no teste. O nome do arquivo precisa constar no arquivo de configuração json na chave "ArquivoEntrada". Os valores para um mesmo elemento precisam estar separados por espaço:

nome_ variável = valor

nome_variável = valor0 valor1...

Para variáveis que podem conter múltiplos valores para o mesmo elemento, esses precisam estar limitados por parênteses:

nome_variável = (valor0) (valor1)

Caso haja menos valores que elementos, o último valor dado será atribuído a todos os outros. Strings precisam estar delimitadas por aspas duplas:

nome_variável = 'uma string'

Algumas variáveis possuem valores padrão que será mantido caso não ela não seja preenchida. A seguir as variáveis que podem ser usadas bem como seus valores, caso exista:

Variáveis do programa:

material; materiais que serão usados no processo. Deve-se seguir o padrão:

('Z-Símbolo-N', 'REAÇÃO'). As reações suportadas são *incident-neutron data,* N, e *photo-atomic interection data,* PHOTO. Exemplo: ('1-H-0', 'PHOTO')

Variáveis do NJOY:

Todas as variáveis são de valores único exceto onde indicado

tlabel[1]: 'fita pendf para o(s) material(is) : '+ lista de materiais usados. Vai até 66 caracteres.

title[1]: string com até 80 caracteres listando os materiais.

cards[2] 'processado pelo NJOY', 'Veja a original para mais detalhes'

temp2: 0

errthn : 0.005

ign : 3

igg : 3

iwt: 3

lord : 3

iprint : 1

sigz: 1.e+10

mtname[2]**: '**total'

egg[3] (múltiplos valores)

egn[3] (múltiplos valores)

mfd[4]**:**

mfd[4]**:**

[1]os valores são string e não mudam de acordo com o material

[2]os valores são string e podem mudar de acordo com o material

[3]os valores não mudam para os vários elementos e não é preciso estar entre parênteses.

[4]os valores são números e podem mudar de acordo com o material.

Quem faz a leitura é a função "leia", "getEntrada" a retorna em um dicionário, cujas chaves são os nomes das variáveis digitadas pelo usuário.

Arquivo padrão

O arquivo padrão é a "forma" usada para gerar os inputs. Nele estão escritos o nome das variáveis, dos módulos do NJOY e STOP, que indica o fim do input. Além da palavra VOLTE POSIÇÃO que para criar laços. O programa lê esse arquivo e substitui o nome das variáveis pelos seus valores, caso uma string não seja um nome de variável, ele simplesmente a transcreve para o input (com a exceção de VOLTE).

O usuário pode optar por diferentes padrões indicando o nome do arquivo na chave "padrao" no arquivo de configuração.

Arquivos para misturas ou substâncias

Há dois arquivos com informações que indicam a composição de uma mistura e informações sobre isótopos. Esses arquivos são usados para o cálculo da seção de choque macroscópica a partir da microscópica fornecida pela saída do NJOY e são opcionais.

O arquivo com informações sobre a mistura lista a composição, identificando o isótopo e a fração desse isótopo nela. A primeira linha do arquivo deve contém uma string seguida da densidade da mistura. Exemplo:

light dolomite 2.5

1-H-0 0.0082538

6-C-0 0.0839755

8-O-0 0.5098378

11-Na-0 0.002732

12-Mg-0 0.069146

13-Al-0 0.00064078

14-Si-0 0.05412

15-P-0 0.0000164

16-S-0 0.0004093

19-K-0 0.0002803

20-Ca-0 0.2611081

26-Fe-0 0.004129

Já o arquivo com os elementos serve para indicar o peso atômico dos isótopos. A estrutura é similar ao arquivo anterior. Por exemplo:

1-H-0 1.008

6-C-0 12.011

8-O-0 15.999

11-Na-0 23

...

Arquivo de configuração

O arquivo de configuração serve para indicar o caminho do executável do NJOY (chave njoy) e os nomes dos arquivos usados na execução. Exemplo:

output; string correspondente ao nome do arquivo onde o NJOY deve escrever a saída. Chave: output

{

"njoy" : "/NJOY21/bin/", *caminho do NJOY*

"teste_1" : *dicionário com as informações para um teste*

{

"caminho":"/NJOY21/bin/ic_testsJoao/testeConcretoAutomatizado", *caminho da pasta onde acontecerá o teste. Ela não precisa ser criada.*

"output": "outputConcretoAutomatizado1", nome do arquivo onde o NJOY deve escrever a saída.

"arquivoEntrada" : "testeAki.txt", *nome da entrada com as variáveis.*

"arquivoSaida" : "inputGeradoMAIN.txt", *nome do arquivo de input do NJOY*

"padrao" : "inputPraTestarGAMIRN.txt", *padrão usado para gerar o input*

"ingredientes" : "densidade_materiais.txt", *composição da mistura*

"elementos" : "peso_atômico.txt" *lista de isótopos com seus pesos atômicos*

}

}

Caso o usuário queira realizar diversos testes ao mesmo tempo, basta criar vários dicionários com as informações necessárias.

Classes e funções

Explicação sobre as classes e funções usadas pelo programa

Tapes

Responsável por ler os arquivos ENDF e obter o código mat do material. Seu construtor recebe como argumento uma lista contendo os materiais e o tipo de informação.

Para obter os códigos, usa-se a função procurarTAPE que lista as tapes disponíveis na pasta e abre os arquivos correspondentes a cada material. A função considera que todos os arquivos ENDF estão no formato txt e nomeados com o nome do material no formato usual. Cada arquivo ENDF deve conter um único material.

Após abrir o arquivo correto, procurarTAPE usa __getMat__ para procurar pela palavra-chave correspondente ao tipo de informação, assim distinguindo entre os possíveis mats diferentes. Após achar a primeira ocorrência da palavra-chave, __getMat__ pega o terceiro valor da linha apartir da direita.

A função obterMats, então, retorna um dicionário cujas chaves são os materiais e os valores seus mats, em ordem decrescente.

WebScrapping

A função webScrapping é responsável por buscar em: ENDF: Evaluated Nuclear Data File (iaea.org) os arquivos ENDF ausentes. Ela recebe como argumento a lista de materiais desejados e compara com os arquivos txt presentes no diretório, descartando os presentes. WebScrapping precisa do navegador Firefox.

Parametros

A classe Parametros recebe a entrada preenchida pelo usuário e preenche as variáveis do NJOY bem como gera valores para outras. Para tanto, a classe chama a função "receberParametros" que passa em sequência a entrada para as funções encarregadas do preenchimento específico dessas variáveis.

Para a maior parte dessas variáveis, o preenchimento é simplesmente chamar o dicionário; parametrosSimples (que contém variáveis com único valor para diversos materiais) e de acordo com a chave, o nome da variável, guardar o valor da entrada. Isso fica à cargo da função __valorUnico__.

Outras variáveis exigem verificações especiais ou preenchimento automático. São essas:

__preencheCards__: responsável por preencher a variável cards e, automaticamente, ncards. Ela preenche no máximo até 60 caracteres de cada string de cards, caso esteja vazia, a função preenche com o valor padrão. Em seguida, ela conta a quantidade de strings e assim, preenche ncards.

__title__: preenche a variável title com até 80 caracteres. Caso não esteja vazia, a função preenche com um valor padrão.

__tlabel__: preenche tlabel com uma string de até 66 caracteres. Caso esteja vazia, usa o valor padrão.

____preencheEgg__ e __preencheEgn__: responsáveis pelo preenchimento das variáveis egg quando igg for 1 e egn quando ign for 1 respectivamente. As duas funções preenchem automaticamente ngn com o tamanho de egn e ngg com o tamanho de egg.

__preencheReac__: preenche as variáveis mfd, mtd e mtname.

setMatb: recebe uma lista de código de materiais e os armazena em "matb".Em seguida, nomeia os arquivos de entrada de acordo com a quantidade de códigos fornecidas. Por exemplo, se há 4 materiais:

nin, que é a variável com o nome dos arquivos ENDF, será: [20, 21, 22, 23]. A partir do último valor, os demais são gerados ficando:

nout = -24

npend = -25. E assim por diante.

Nota: o sinal negativo indica arquivo binário.

"getMat" e "getParametros" retornam uma lista de mats e um dicionário de Parametros respectivamente.

ParametrosAdmin

A partir as variáveis preenchidas por Parametros e o padrão, cria o input para o NJOY. Primeiro, ParametrosAdmin recebe três argumentos: o padrão, os parâmetros e o nome do input a ser gerado. Em seguida, o construtor cria o arquivo inputGerado para a escrita, lê o arquivo padrão e chama a função __inicializarLaco__.

O objetivo de __inicializarLaco__ é escrever moder considerando se há um ou vários materiais, de acordo com o modelo abaixo:

moder

20 -21/

Caso haja n elementos, os comandos serão:

moder

1 (-20-n)/

tlabel

20 mat0/

...

(20 + n - 1) matn/

0/

Onde –n-1 é a tape de saída e 1 é uma *flag* para múltiplos materias. As outras funções são:

preencheInput: escreve o inputGerado substituindo as variáveis em padrão pelos seus valores obtidos de determinadas funções ou escrevendo o que estiver escrito nele. PreenchInput ainda lida com os possíveis laços de repetição internos que existem no padrão. Encerra a escrita ao chegar em STOP.

O primeiro deles é indicado por VOLTE POSICAO e usa a variável “ponteiro”, que recebe o valor POSICAO, como ponteiro assim, a leitura de uma linha em padrão é feita usando-o como indice; padrao[ponteiro]. Este laço só continua enquanto a variável quantidade (a quantidade de materiais) for não nula.

O segundo laço é interno em cada linha que serve para pegar os valores de cada material em variáveis com múltiplos valores. É controlado pelo ponteiro ponteiroLaco que é incrementado por 1 quando se entra no laço anterior.

Após pegar uma variável em padrão, PreencheInput invoca __preencheLinha__ e passa como argumento a variável e o ponteiroLaco. Essa função, por sua vez, chama a função correta para cada variável e depois retorna o valor obtido para PreencheInput que, finalmente, o escreve. A seguir, essas funções:

__card9__: retorna o valor das variáveis mfd, mtname e mtd. __Card9__ não usa ponteiroLaco para selecionar os valores para cada material, apenas verifica se ele já foi incrementado assim, consegue distinguir quando usar seu próprio ponteiro para selecionar diferentes valores. Ela obtém os valores para um material e os junta em uma única string no formato abaixo:

mfd mtd mtname/\n

Como o primeiro valor para o primeiro elemento é escrito em outro card, __card9__ verifica se ponteiroLaco foi incrementado e assim seleciona os próximos valores. Nesse caso, teremos o formato:

mat/\n

mfd mtd mtname/\n

__multiploElementos__: retorna o valor de cards, usando ponteiroLaco para selecionar os cards de cada elemento. Caso haja menos cards que há elementos, o último card é usado nos que restam.

__parametrosComuns__:

e verifica se nela há a palavra STOP, caso sim, ele escreve-a em inputGerado e encerra o programa, senão, ele passa o valor para __preencheLinha__. preencheInput

também é responsável pelo controle das repetições, assim ele verifica se na linha está escrito VOLTE, caso esteja, ele subtrai o contador e defi-

ne o valor de "ponteiro" para POSIÇÃO em VOLTE POSIÇÃO. Quando a quantidade de repetição é zero, o método reseta o contador, bem como o ponteiro

interno que seleciona os valores das linhas, além de escrever "0/" indicando o fim dos elementos para um determinado módulo.

__preencheLinha__ é um método responsável por determinar qual função retornará os valores de uma determinada variável no formato adequado além de

retornar o valor da variável "title". Para as variáveis: mfd, matname e mtd, chamadas de "card9" é utilizada o método "card9" que pega seus valores

em parametros e os retorna comoa uma string no formato:

mfd mtd mtname/\n

Quando há mais de um elemento, o método usa "ponteiroLaco" para selecionar os valores de cada um. Assim, o formato fica:

mat

mfd mtd mtname/\n

__parametrosComuns__: retorna o valor das demais variáveis, exceto tlabel e title que a própria preencheLinha o faz. O trabalho de __parametrosComuns__ é relativamente simples, pois os valores não exigem uma formatação especial. As exceções são:

egg e egn que são colocadas num formato de matriz para facilitar a leitura;

ncards e matb. Cada vez que a função encontra matb, ela acessa o próximo código mat usando ponteiroLaco + 1. Ncards exige um trabalho semelhante a cards; é preciso usar ponteiroLaco para acessar os diferentes valores e, quando há menos ncards que elementos, usa-se o último valor para os demais.

Por fim, __preencheLinha__ retorna o valor das funções ou o que estiver escrito no padrão à preencheInput que por sua vez, escreve no inputGerado.

MacroSeccao

MacroSeccao calcula a seção macroscópica do material a partir da seção microscópica obtida da saída do NJOY. Entre elas, está a seção macroscópica total, calculada pela fórmula:

$$\sum_{total} N_{total} \cdot \sigma_{total}$$

Onde N_{total} é a densidade atômica total de cada elemento e σ_{total} é a seção microscópica total. A densidade atômica total pode ser calculada por:

$$N = \frac{n\#\ de\ Avogrado\ \cdot\ densidade\ do\ material}{peso\ molecular} \cdot fração\ de\ peso$$

A classe recebe como argumento os nomes do output do NJOY, ingredientes e elementos. Os métodos da classe são:

CalculaNi: responsável pelo cálculo da densidade atômica dos materiais. Ela lê ingredientes e pega o valor correspondente à densidade da mistura e em seguida, para cada elemento no arquivo, ela armazena o valor da fração de peso em uma lista e o elemento em outra. Depois, a função obtem o peso atômico dos isótopos presentes nessa última lista. Por fim, é calculada a densidade atômica utilizando o numpy.

SecaoMicro: é responsável por obter a lista de seções microscópicas do output. Ela abre o arquivo e procura pela palavra-chave; ' group (barns)\n' que marca o início dos resultados gerados pelo NJOY. Em seguida, o programa lê os resultados, armazenando cada um em uma matriz. O programa utiliza o método “find” das listas do python para procurar a palavra chave e assim, poder ir saltado pelo output, além de encerrar sua execução quando ela não está presente.

CalcularMacroscopicaTotal: chama SecaoMicro e CalculaNI e calcula o valor da seção macroscópica total como uma simples multiplicação de vetor, a densidade atômica dos elementos, e matriz, a seção microscópica. Como a matriz armazenada tem a dimensão; n# de elementos x n# de grupos de energia, pode ocorrer de não ser possível multiplicar ambos, para contornar essa situação,

CalcularMacroscopicaTotal inverte as colunas e as linhas da matriz, faz a multiplicação e inverte novamente a matriz resultante.

ReturnMacroTotal; retorna a matriz calculada por CalcularMacroscopicaTotal.

Funções usadas na execução

São as funções usadas para chamar: as classes e funções acima, o NJOY e gerenciar os arquivos necessários. Antes delas serem chamadas, o programa pede ao usuário que digite o nome do arquivo json, podendo optar pelo nome padrão ao digitar ctrl+c. O programa abre o arquivo, lê as informações da chave e às passa para main, rodar e macro_seccao:

main: é a função principal do programa. Ela é responsável por ler a entrada e escrever o inputGerado e, como argumento, recebe o nome do arquivo de entrada, do inputGerado e do padrão.

Então, ela instancia Entrada e usa o método "leia" para ler a entrada e getEntrada para obtê-la em um dicionário. O item da chave "material" é passado para webScrapping a fim de baixar as tapes ausentes. A saída webScrapping é passada para uma instância de Tapes que, usando procurarTape, obtém-se o dicionário com os elementos como chave e o nome das tapes como item.

Em seguida, ela instancia Parametros e passa como argumento a entrada por meio de receberParametros. ObterMats retorna o dicionário gerado por Tapes e, extraíndo somente os itens, passamos a lista de tapes para setMatb. GetParametros retorna o dicionário com todos os parâmetros preenchidos.

Por fim, ela cria um objeto do tipo ParametrosAdmin e passa o nome do padrão, os parâmetros e o nome do inputGerado.

Antes de retornar a lista de tapes, main passa o dicionário para renomear que usando a mesma lógica que Parametros, copia os arquivos ENDF com nome de tape20+n, ficando, por exemplo, tape20 e tape21 se houverem dois arquivos ENDF usados, e retorna seus nomes em uma lista. No fim, main retorna tanto a lista quanto o nome do inputGerado.

Rodar: é responsável por passar os comandos para o console. Recebe como argumento o arquivoEntrada, o caminho, as tapes, o output e njoy (que são as chaves do arquivo json). Em seguida, cria a pasta no caminho e copia o executável do njoy para ela, as tapes e o arquivoEntrada para ela. Rodar, então, exclui a saída anterior do njoy, caso exista, e executa o njoy usando o comando:

sudo ./njoy21 < arquivoEntrada –o outputNome

macro_seccao: é responsável por instanciar a classe MacroSeccao que calcula a seção macroscópica total apartir da seção total microscópica obtida na saída do njoy. Recebe como argumento o nome da saída, os ingredientes e elementos, além do caminho do programa. Usando calcularMacroscopicaTotal para fazer os cálculos e returnMacroTotal para obtê-los numa lista e escrevê-la em um txt na pasta do programa.