# Руководство пользователя SkyNet

Вы установили SkyNet. Если все шаги проделаны верно, то при выполнении

**from** SkyNet **import** **\***

не должно быть ошибок.

Затем нужно определить входные данные для процесса. Сначала определяются элементы, участвующие в процессе. Они описаны в файле “sunet”.

**with** open("sunet") **as** f:

nuclides **=** [l**.**strip() **for** l **in** f**.**readlines()]

Содержимое файла “sunet”:

“ n

p

d

t

he3…”

Для большинства представленных примеров они определяются именно так, с небольшими отличиями.

Далее нужно загрузить NuclideLibrary из файла.

nuclib **=** NuclideLibrary**.**CreateFromWinv("winvne\_v2.0.dat", nuclides)

или

…

Из этого файла подгружается только информация для элементов, указанных в переменной nuclides.

Теперь определяются настройки сети

opts **=** NetworkOptions()

opts**.**ConvergenceCriterion **=** NetworkConvergenceCriterion**.**Mass

opts**.**MassDeviationThreshold **=** 1.0E-10

opts**.**IsSelfHeating **=** do\_heat

opts**.**EnableScreening **=** do\_screen

opts**.**DisableStdoutOutput **=** True

## Описание параметров сети

NetworkOptions.ConvergenceCriterion:

Перечисляемый тип, определяющий критерий сходимости. Может быть:

* NetworkConvergenceCriterion**.**Mass – критерий определяется через массу

NetworkOptions**.**MassDeviationThreshold

Тип double. Порог отклонения массы

NetworkOptions**.**IsSelfHeating

Тип boolean. Указывает, учитывать ли процесс само-нагревания.

NetworkOptions**.**EnableScreening

Тип boolean. Включает пересчет со screening’ом.

NetworkOptions**.**DisableStdoutOutput

Тип boolean. Включает или отключает вывод в консоль.

NetworkOptions**.**MinDt

Тип double. Указывает минимальное значение времени

NetworkOptions**.**NSEEvolutionMinT9

Тип double. Указывает минимальное значение …(скорее всего температуры, позже утоню)

Для корректной работы Screening’а нужно загрузить данные

helm **=** HelmholtzEOS(SkyNetRoot **+** "/data/helm\_table.dat")

Если мы указываем opts**.**EnableScreening = false, то подгружать эта данные не нужно

## Библиотеки реакций

При проведении эволюции в процессе участвуют несколько реакций. У пользователя есть выбор, какие реакции учитывать. Для этого создается несколько переменных вида:

REACLIBReactionLibrary("reaclib",

ReactionType**.**Strong, do\_inv, LeptonMode**.**TreatAllAsDecayExceptLabelEC,

"Strong reactions", nuclib, opts, True, True), где

1. Первый аргумент – источник информации о реакции, имя файла. Тип String
2. Второй аргумент – тип реакции. Перечисляемый тип. Если в данном семействе только одна реакция – указывается только ReactionType.Strong. (Есть также ReactionType.Weak)
3. Третий аргумент – … . Тип Boolean. Используется только для сильных реакций.
4. Четвертый аргумент – Не очень ясная штука, во всех примерах одно и то же. Перечисляемый тип.
5. Пятый аргумент – наименование реакции. Тип String. Указывается любое уникальное название.
6. Шестой аргумент – переменная параметра сети. Тип NetworkOptions.
7. Седьмой и восьмой аргументы - всегда True

Затем все эти реакции сохраняются в массив reactionLibraries = [reaction1, reaction2, …].

Далее формируется процесс screening’а, а также сеть реакций

screen **=** SkyNetScreening(nuclib)

net **=** ReactionNetwork(nuclib, reactionLibraries, helm, screen, opts).

Сеть построена, осталось только загрузить начальные данные.

dat **=** np**.**loadtxt("traj\_skynet")

density\_vs\_time **=** PiecewiseLinearFunction(dat[:,0], dat[:,2], True)

temperature\_vs\_time **=** PiecewiseLinearFunction(dat[:,0], dat[:,1], True)

Данные подгружаются из файла traj\_skynet, в котором в первой колонке указано время, во второй – температура, в третьей – плотность.

Дополнительно для начала процесса эволюции нужно указать начальное (t0) и конечно время (tFinal), а также если мы используем эволюции через само-нагревание, то нужно задать начальную температуру Temp0.

Temp0 **=** dat[0,1]

t0 **=** dat[0,0] **+** 1.0e-20

**if** (do\_heat):

tfinal **=** 1.0E+03

**else**:

tfinal **=** 1.242595E+03

t0 задано таким образом, чтобы начальное время не могло было равно 0.

Загружаем начальное распределение распространенности элементов:

initY **=** np**.**loadtxt("init\_Y"),

где “int\_Y” – имя файла. Формат:

Количество строк аналогично файлу «sunet», 1 к 1 отражают распространенность того или иного элемента.

Другой способ задания Y это через

Y0 **=** np**.**zeros(nuclib**.**NumNuclides())

Y0[nuclib**.**NuclideIdsVsNames()["c12"]] **=** 0.5 **/** 12.0

Y0[nuclib**.**NuclideIdsVsNames()["o16"]] **=** 0.5 **/** 16.0

В этом случае мы задаем начальные данные только для двух элементов, остальные же = 0.

Далее запускается процесс эволюции. Если есть само-нагревание:

output **=** net**.**EvolveSelfHeatingWithInitialTemperature(initY, t0, tfinal,

Temp0, density\_vs\_time, pref)

Если нет:

output **=** net**.**Evolve(initY, t0, tfinal, temperature\_vs\_time, density\_vs\_time,

pref)

Где pref – название выходного файла.

Бывают ситуации когда у нас нет начального распределения элементов, но есть другие данные, такие как плотность, температура и ye (распространенность электоронов(?)). В таком случае можно построить вектор Y

nseResult **=** nse**.**CalcFromTemperatureAndDensity(temperature\_vs\_time(t0), rho, ye)

initY **=** nseResult**.**Y()

