

Создание графической оболочки пользователя в SIMULIA Abaqus на примере задачи прокатки.

Разработал: А. Кисловский

Под редакцией: С. Тропкина, Д. Нуштаева

Москва, 2021

Введение:

В данном пособии рассматривается процесс создания плагина (графической оболочки) для автоматизации задачи прокатки металлического листа в SIMULIA Abaqus. Графическая оболочка - это окно, позволяющее в интерактивном режиме собрать задачу для анализа и запустить расчёт (Рис. 1).

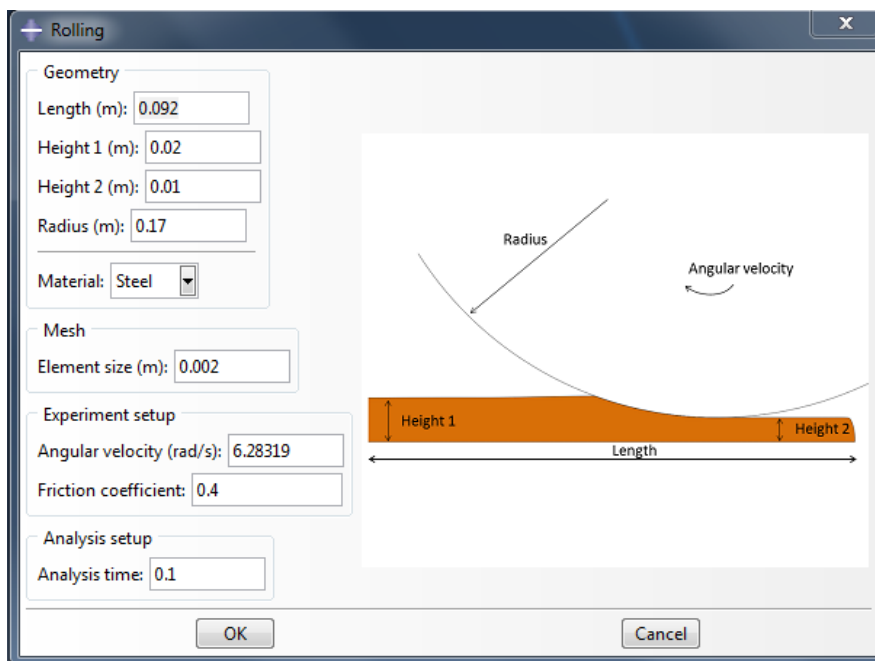


Рис. 1 Окно пользовательского интерфейса для задачи прокатки.

Пособие состоит из двух частей. В части 1 пошагово собирается задача прокатки стального листа (В настоящее время отсутствует детальное описание математической модели). Часть 2 посвящена параметризации задачи и созданию графической оболочки в программном комплексе SIMULIA Abaqus.

Графическая оболочка создаётся для программного кода, написанного на языке Python 2х, выполняющего сборку модели и запуск её на расчёт.

Самый простой способ создать Python код для **Plug-in** использовать **Macro Manager**. Макрос записывает действия пользователя в виде Python команд и воспроизводит их при запуске. Как было отмечено выше, плагин создаётся на основе Python функции. В программировании функции – это средства, позволяющие группировать наборы инструкций так, что в программе они могут

запускаться неоднократно. Функции могут вычислять некоторый результат и позволять указывать входные параметры, отличающиеся по своим значениям от вызова к вызову.

Предложенный способ не единственно возможный, но с точки зрения автора, один из самых простых.

Для успешного выполнения заданий пособия знание синтаксиса языка Python является желательным, но не обязательным условием. Моменты, связанные с языком программирования, целенаправленно опущены, а все правки текста программного кода сопровождаются скриншотами.

Ориентировочное время выполнения задания – 4 часа.

С любыми пожеланиями и замечаниями, обращайтесь по адресу: simulia_support@tesis.com.ru.

Часть 1 (Создание модели)

В данном разделе будет собрана модель прокатки стального листа. Для упрощения дальнейшей работы с материалами пособия, рекомендуем придерживаться последовательности выполнения предложенных действий. На выходе мы получим python скрипт – макрос, с которым будем работать в части 2 настоящего пособия.

Активируем запись макроса для задачи:

File -> Macro Manager... -> Create -> Введите имя макроса Steel_rolling и директорию Home -> Continue... -> Появится окно Record Macro идентифицирующие, что Abaqus CAE начал записывать совершаемые вами команды. Не закрывайте окно Record Macro до специального указания. (Рис. 2)

Создадим модель стальной плиты (Рис. 3 слева):

Parts -> Create...-> Name: plate, Modeling Space: 2D Planar, Type: Deformable, Base Feature: Shell, Approximate size: 0.5. -> Continue... -> Начертите прямоугольник с высотой равной 0.02 (м) и длиной 0,092 (м), для этого введите координаты левого угла (-0.092 0.0), правого (0.0 0.02) -> Done. Такое определение важно для дальнейшего анализа.

Создадим модель прокатного ролика (Рис. 3 справа):

Parts -> Create... -> Name: Roller, Modeling Space: 2D Planar, Type: Analytical rigid, Base Feature: Wire, Approximate size: 2.0 -> Continue... -> Начертите четверть дуги окружности радиусом 0,17 (м) -> Done.

Tools -> Reference Point -> разместите её в центре созданной окружности.

Создадим материал (Рис. 4):

Materials -> Name: Steel; Material Behaviors: Mechanical – Elasticity – Elastic, Data: Young's Modulus = $1.5e11$ (Па), Poisson's Ratio = 0.3, General – Density = 7850 (кг/м³), Mechanical – Plasticity – Plastic Data: Yield Stress = [1.6872e8, 2.1933e8, 2.7202e8, 3.0853e8, 3.3737e8, 3.6158e8, 3.8265e8, 4.0142e8, 4.1842e8, 4.3401e8, 4.4845e8] (Па), Plastic Strain = [0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0].

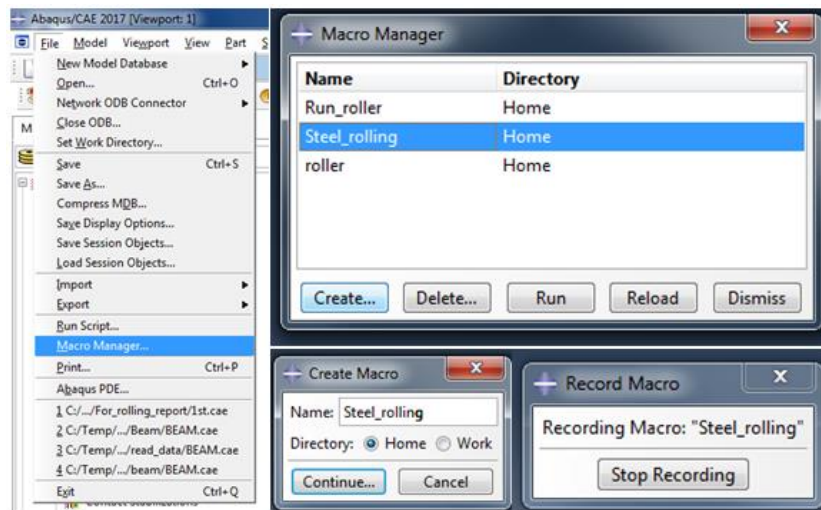


Рис. 2 Создание макроса.

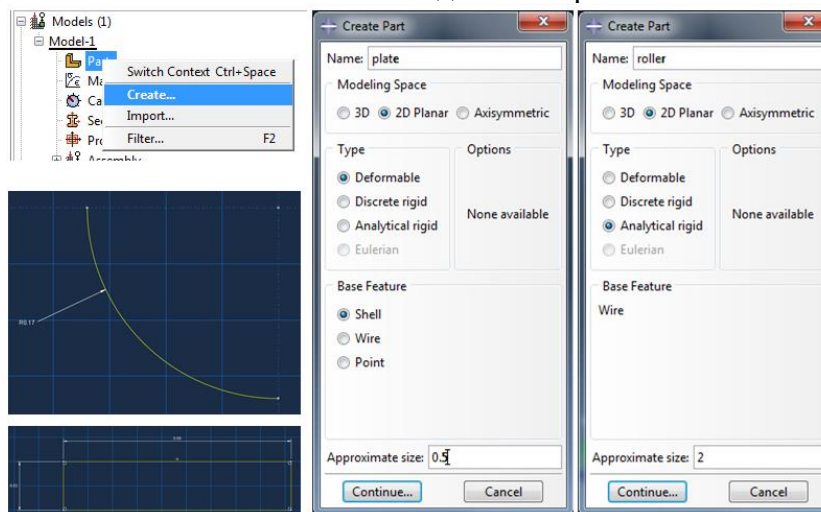


Рис. 3 Создание геометрии стальной плиты и прокатного ролика.

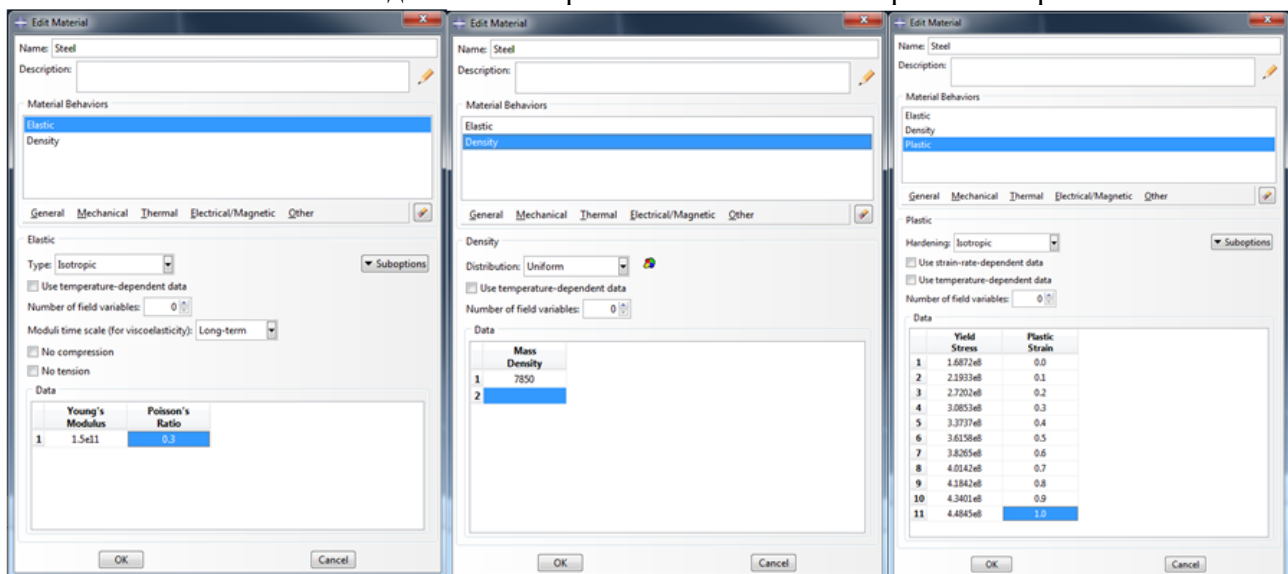


Рис. 4 Определение свойств материала плиты.

Создадим сечение и присвоим его стальной плите (Рис. 5):

Sections -> Name: Steel, Category: Solid, Type: Homogeneous -> Continue... -> Activate Plane stress/strain thickness: 1 -> ОК.

Parts -> plate -> Section Assignments -> Выберите деталь -> Done -> Section: Steel -> ОК.

Соберём модель (Рис. 6):

Assembly -> Instances -> Parts: Roller, plate -> ОК. Переместите правый нижний угол плиты в начало координат. В начальной конфигурации ролик касается плиты точкой отстоящей от одного из его концов на 20% его длины. Создадим эту точку. Tools -> Datum -> Type: Point, Method: Enter parameter -> Выделите грань катка -> Normalized edge parameter (0.2) -> Подтвердить действия. Сопрягите верхний правый угол пластины и созданную точку на поверхности катка.

Создадим удобные для дальнейшей работы сеты:

Assembly -> Sets = [bottom (нижняя грань плиты), plate (вся плита целиком), refPt (Опорная точка катка)].

И поверхности:

Assembly -> Surfaces = [roller (грань катка со стороны плиты), topPlate (верхняя и правая грани плиты)].

Задание свойств анализа (Рис. 7):

Steps -> Name: Single Pass Rolling, Procedure type: General Dynamic, Explicit -> Continue... -> Time period: 0.1 -> Во вкладке Mass scaling выберете Use scaling definitions below -> Create -> Application, Region: Set = plate, Type: Scale by factor = 1600 -> ОК -> убедитесь в том, что окно Edit Step выглядит, как на рисунке 8 -> ОК.

В результате анализа кроме нагружено-деформируемого состояния плиты интересно получить силу и момент действующую на ось ролика (Рис. 9).

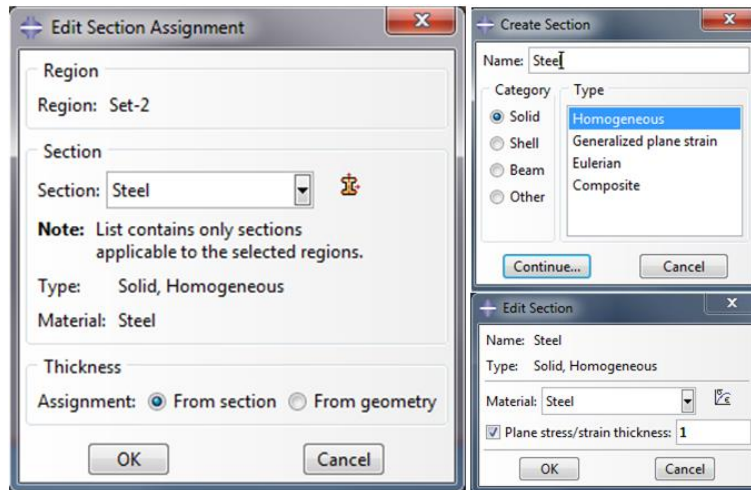


Рис. 5 Задание свойств сечений.

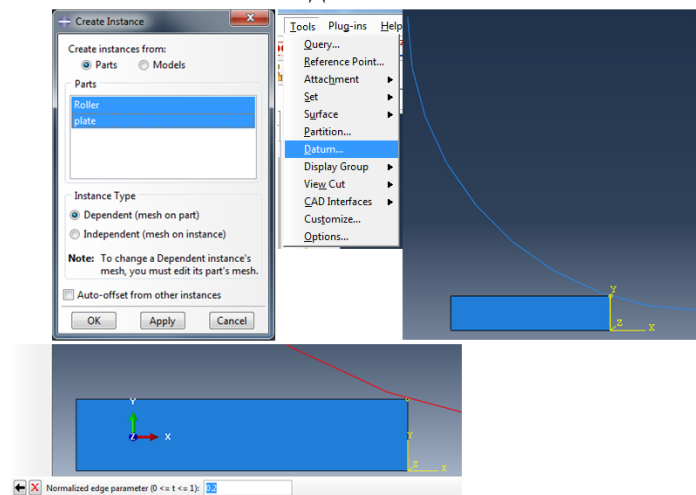


Рис. 6 Создание сборки модели.

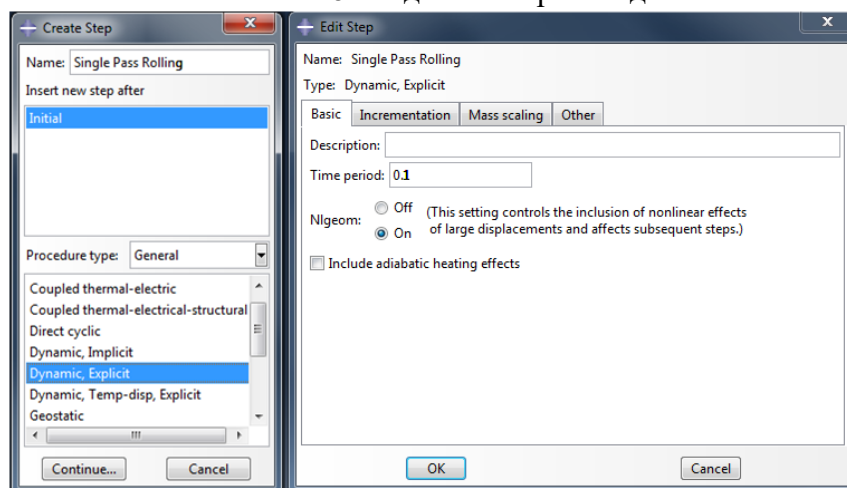


Рис. 7 Создание шага анализа.

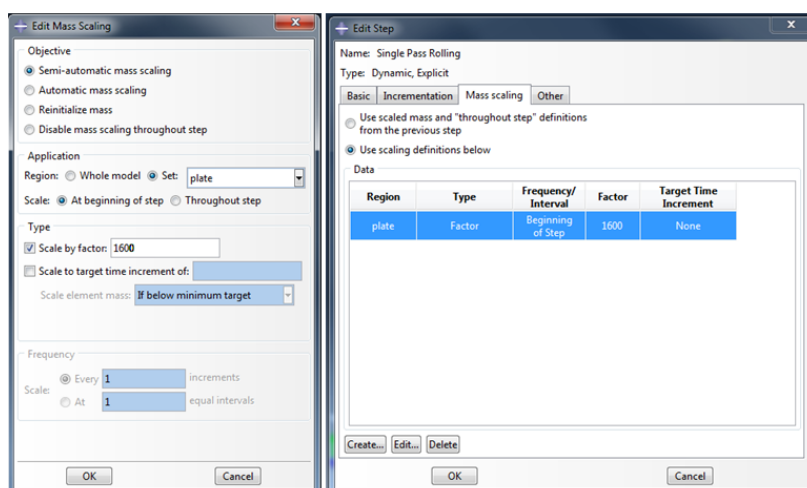


Рис. 8 Задание свойств анализа.

History Output Requests -> Step: Single Pass Rolling -> Domain: Set = refPt, Interval = 100 -> Forces/Reactions -> RT, RM. (Рис. 9 слева)

Активируем адаптацию сетки (Рис. 9 справа):

Other -> ALE Adaptive Mesh Domain -> Manager -> Edit -> Use the ALE adaptive mesh domain below, Region -> указатель мыши (выбрать из графического окна) -> Sets (внизу графического окна) -> plate -> Continue... -> Remeshing sweeps per increment = 3. -> OK -> Dismiss.

Определим контакт (Рис. 10):

Interaction Properties -> Name: Friction, Type: Contact -> Continue... -> Mechanical -> Tangential Behavior -> Friction formulation: Penalty -> Friction Coeff = 0.4.

Interactions -> Name: Friction; Surface-to-surface contact (Explicit) -> Continue... -> Surfaces... -> roller -> Continue... -> Surface -> topPlate -> Continue... -> OK.

Граничные условия (Рис. 11):

Каток вращается с угловой скоростью 2π рад/с. Скорость пластины в начальный момент времени равна $2\pi * 0.17 = 1.01587$ м/с.

BCs -> Name: bottom, Step: Initial, Displacement/Rotation -> Continue... -> bottom -> Continue... -> $U_2 = 0$.

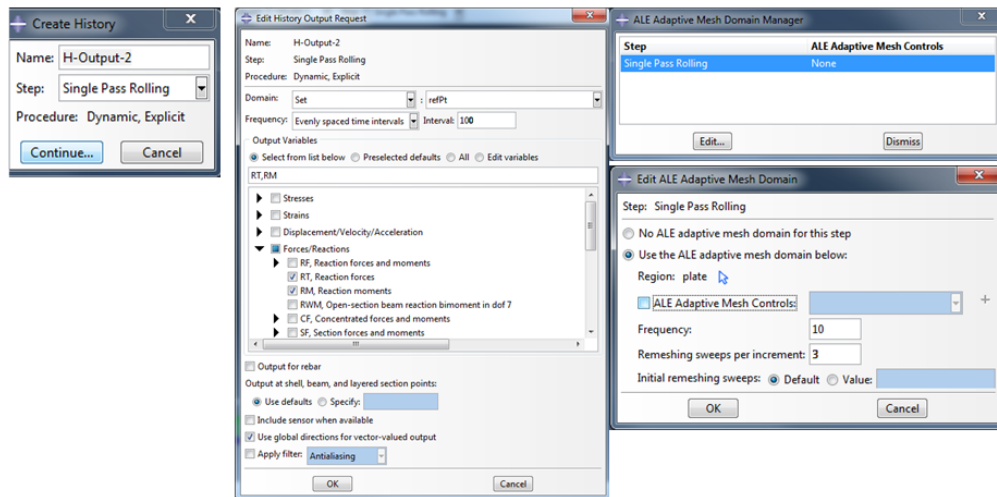


Рис. 9 Выбор вычисляемых переменных и активация адаптации сетки.

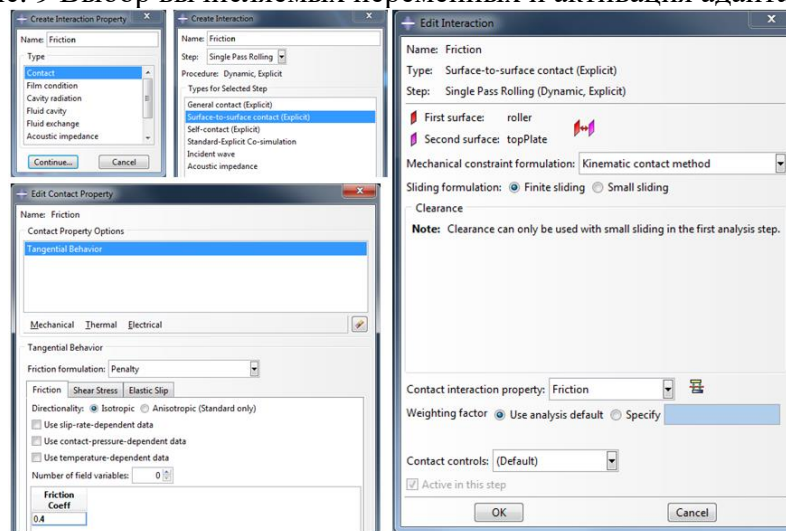


Рис. 10 Определение контакта.

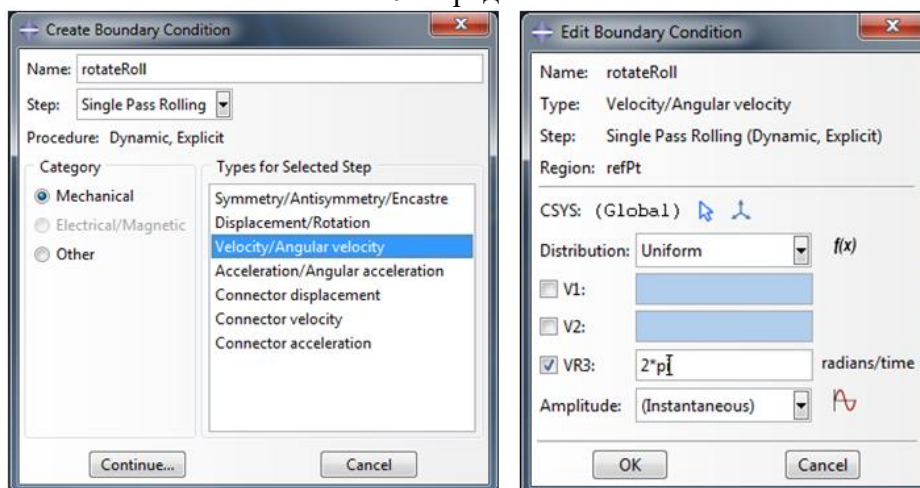


Рис. 11 Задание граничных условий.

BCs -> Name: fixRoll, Step: Initial, Displacement/Rotation -> Continue... -> refPt
-> Continue... -> $U1 = U2 = 0$.

BCs -> Name: rotateRoll, Step: Single Pass Rolling, Velocity/Angular velocity ->
Continue... -> refPt -> Continue... -> $VR3 = 2*\pi$.

Зададим начальную скорость плиты (Рис. 12):

Predefined Fields -> Name: initVel; Step: Initial; Category: Mechanical; Types for
Selected Step: Velocity -> Continue... -> Sets -> Выберите set: plate -> Continue... ->
V1: 1.01587 -> ОК.

Сетка (Рис. 13):

Для данной задачи будут использоваться элементы CPE4R из библиотеки
Abaqus/Explicit.

Plate (в дереве модели) -> Mesh (Empty) -> Mesh -> Element Type -> Element
Library: Explicit; Family: Plane Strain -> ОК -> Seed -> Part -> Approximate global size:
0.002 -> ОК -> Mesh -> Part -> Yes.

Part: Roller (см. Рис !) -> Seed Part -> Approximate global size: 0.02 -> ОК.

Определение анализа (Рис. 14):

Jobs -> Name: rolling -> Continue... -> ОК -> Job Manager -> Submit.

В окне Record Macro нажмите на кнопку **Stop Recording**.

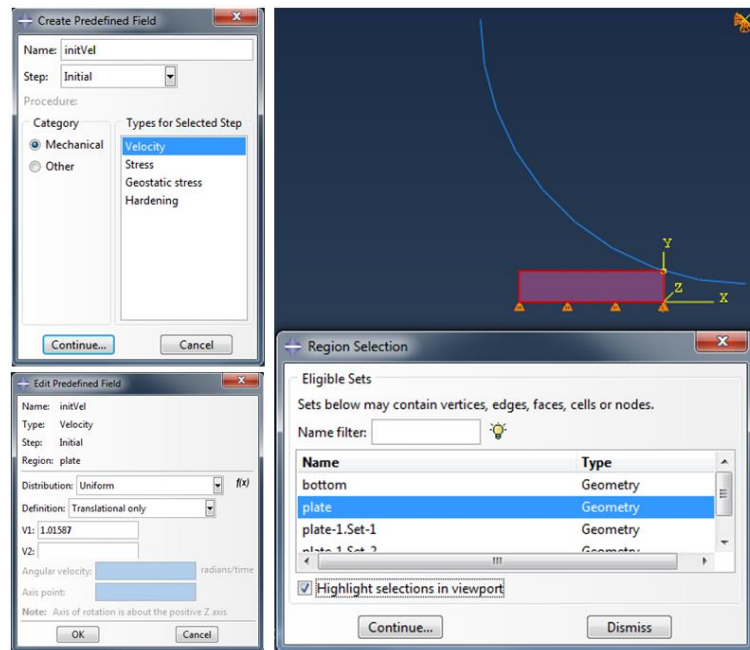


Рис. 12 Задание начальных условий.

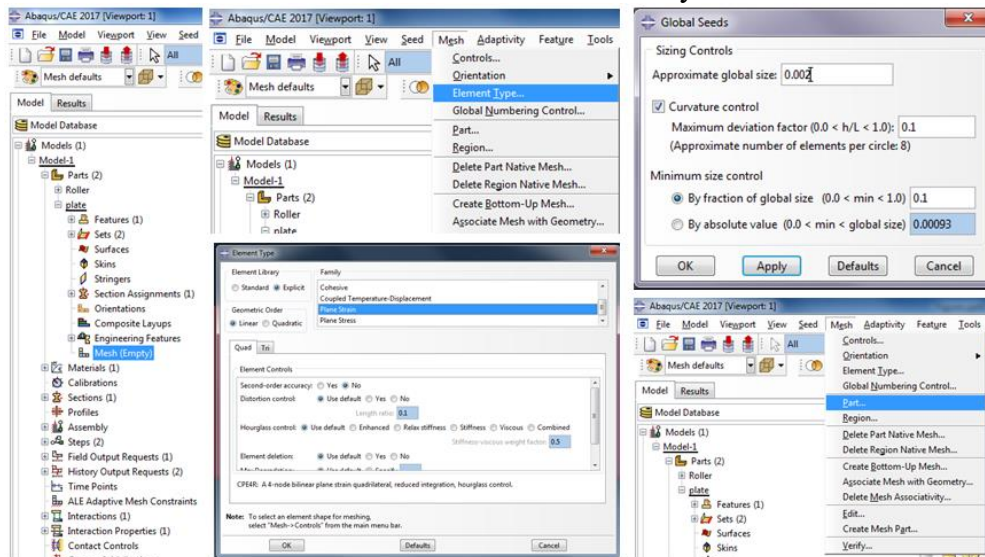


Рис. 13 Создание конечно-элементной сетки.

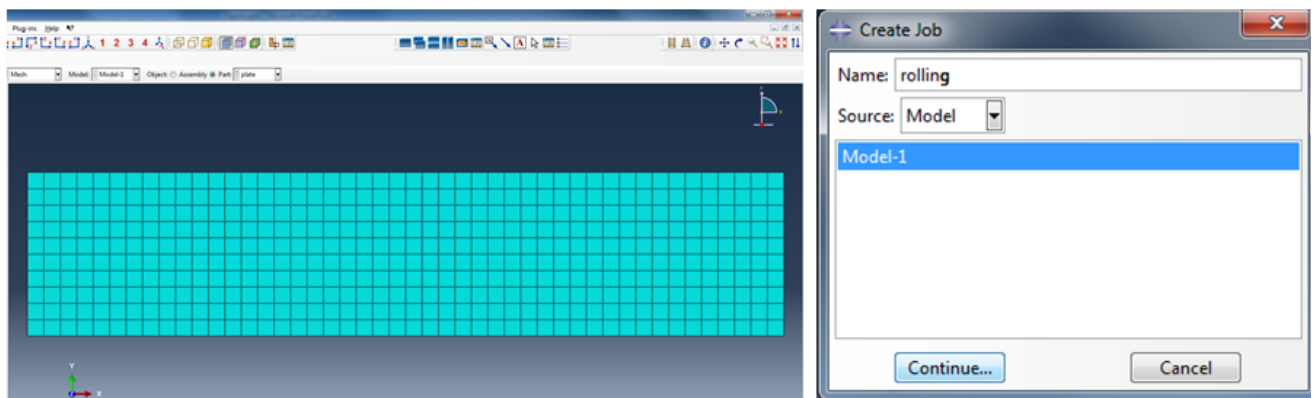


Рис. 14 Расчётная сетка и создание процедуры анализа.

Часть 2. Python. Создание диалогового окна.

В данном разделе рассмотрен процесс создания графической оболочки на основе python скрипта, записанного в первой части настоящего пособия. Выполняя инструкции, особое внимание стоит уделять форматированию вашего кода, чтобы избежать дальнейших ошибок. Всегда сверяйте ваш код, с приведенными примерами программного кода.

После того, как был записан файл макроса, можно переходить к созданию диалогового окна с помощью инструмента RSG Dialog Builder встроенного в ПК SIMULIA Abaqus.

Данный процесс условно делится на две части:

1. Подготовка python функции.
2. Создание графического интерфейса пользователя.

Перейдите в Home directory. Обычно это C:\Users\PC_name. Чтобы не повредить исходный файл abaqusMacros.py содержащий информацию о проделанных действиях в части 1, сделайте его копию с именем rolling.py, переместите его в вашу рабочую директорию, после чего откройте файл с помощью редактора кода встроенного в Abaqus (File → Abaqus PDE... → File → Open → rolling.py), или бесплатным текстовым редактором Notepad++.

Инструкция “def Steel_rolling(): “ в языке Python (Рис. 15) создает объект функции и связывает его с именем Steel_rolling. В строке заголовка в круглых скобках после имени перечисляются подающиеся ей на вход аргументы или параметры. Пустые скобки означают, что функция не принимает на вход никаких значений, а сама функция, в нашем случае, выполняет инструкции, прописанные в теле функции, соответствующие построению и запуску на расчёт задачи проката стальной пластины.

В данном разделе будут внесены изменения в тело функции. Код будет модифицирован так, чтобы функции на вход подавались основные параметры/объекты задачи и на основе данной функции создать графический интерфейс plug-in в Abaqus CAE (Рис. 16).

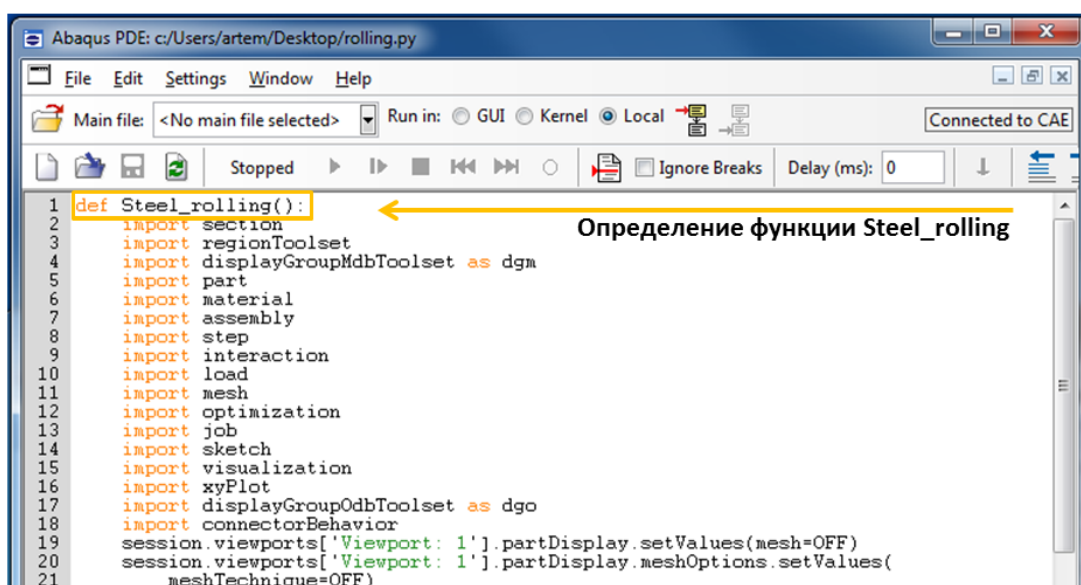


Рис. 15 Окно Abaqus PDE. Определение функции Steel_rolling.

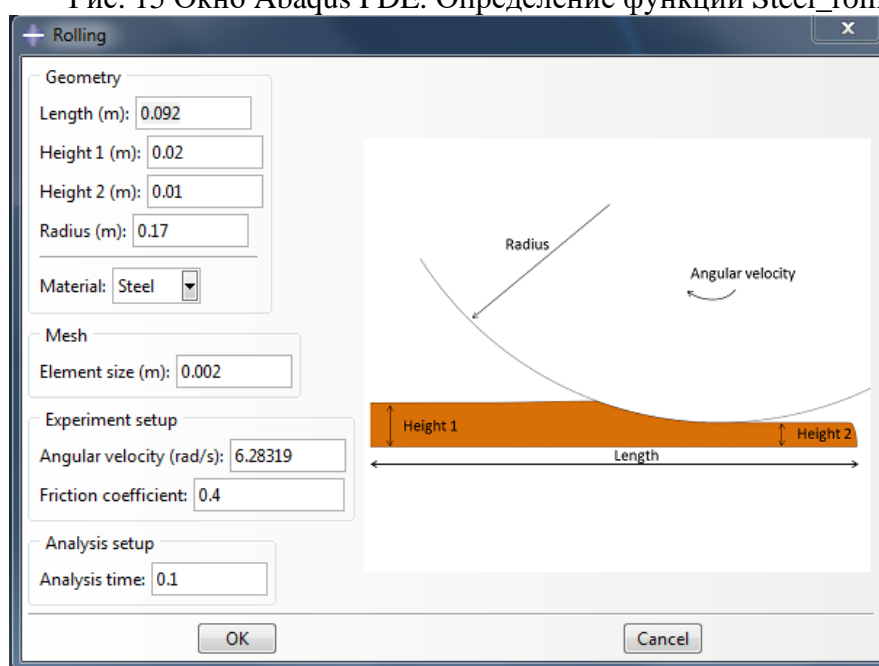


Рис. 16 Окно пользовательского интерфейса для задачи прокатки.

Изменение Python скрипта:

Откройте файл rolling.py с помощью Abaqus PDE... (File → Abaqus PDE-> File -> Open -> rolling.py). Для того чтобы

Для корректной работы функции необходимо подгрузить библиотеки для работы со средой Abaqus. Добавьте три строчки в начало открытого файла rolling.py, подгружающие классы Abaqus (Рис. 17).

```
from abaqus import *
from abaqusConstants import *
import __main__
```

Начало файла должно выглядеть, как показано на Рис. 17.

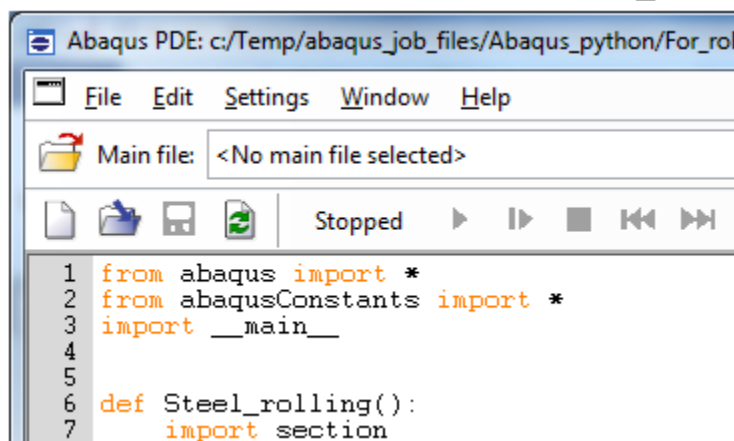


Рис. 17 Заголовок файла rolling.py

На данном шаге нам необходимо провести параметризацию задачи, на вход функция должна получать значения переменных указанных в таблице ниже. Каждая переменная должна быть ассоциирована с её объектным именем внутри функции.

Переменные	Keyword
Длина (Length)	L
Толщина (Height1)	H
Толщина после прокатки (Height2)	h
Радиус валика (Radius)	R
Модель материала (Material)	NameMaterial
Размер конечного элемента (Element size)	seedsize
Угловая скорость валика (Angular velocity)	Omega
Коэффициент трения (Friction coefficient)	mu
Время анализа (Analysis time)	an_time

Таблица 1. Соответствие переменных и имен объектов.

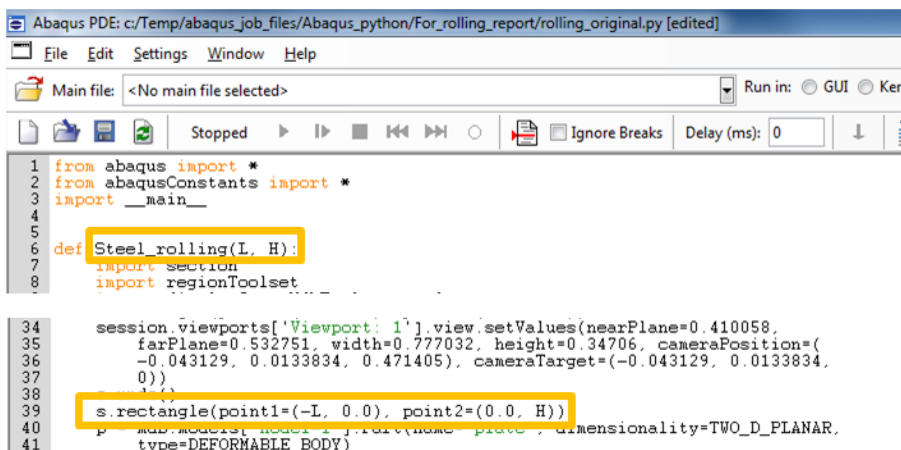
Чтобы упростить последующее редактирование файла создайте несколько пустых строчек таким образом, чтобы 39 строчка на рис. 18 совпадала с 39 строчкой в вашем файле. Если все действия, описанные в Части 1 были проделаны в строгом соответствии с данным пособием, то номера строк должны совпадать. Совпадение строк не является обязательным условием работоспособности кода, а делается для удобства дальнейшей навигации по файлу.

Для начала сопоставим длине, толщине стальной пластины объекты с именами L и H соответственно. Для этого замените строку 6 (Рис. 18) на

Steel_rolling(L, H):

а 39 строка должна включать данные объекты

s.rectangle(point1=(-L, 0.0), point2=(0.0, H)).



```

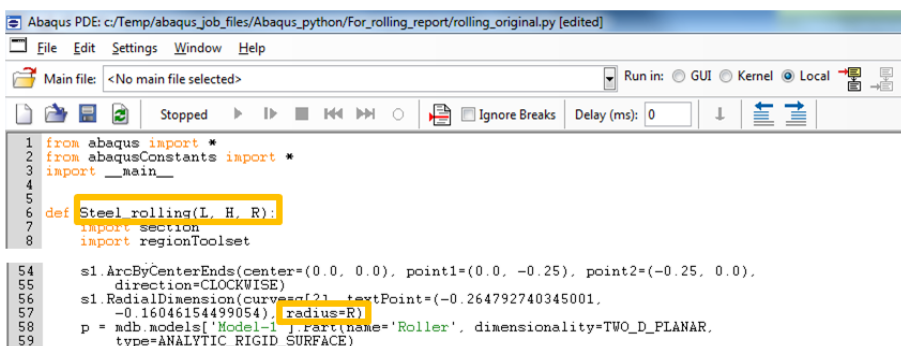
1 from abaqus import *
2 from abaqusConstants import *
3 import __main__
4
5
6 def Steel_rolling(L, H):
7     import section
8     import regionToolset
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34 session.viewports['Viewport: 1'].view.setValues(nearPlane=0.410058,
35     farPlane=0.532751, width=0.777032, height=0.34706, cameraPosition=(
36     -0.043129, 0.0133834, 0.471405), cameraTarget=(-0.043129, 0.0133834,
37     0))
38
39 s.rectangle(point1=(-L, 0.0), point2=(0.0, H))
40
41 p = mdb.models['Model-1'].Part(name='plate', dimensionality=TWO_D_PLANAR,
    type=DEFORMABLE_BODY)

```

Рис. 18 Добавление переменных L, H в функцию.

Объект с именем R соответствует радиусу валика, добавьте его в список переменных, передаваемых в функцию (Рис. 18), а команда на создание дуги окружности должна выглядеть следующим образом:

s1.RadialDimension(curve=g[2], textPoint=(-0.264792740345001, -0.16046154499054), radius=R)



```

1 from abaqus import *
2 from abaqusConstants import *
3 import __main__
4
5
6 def Steel_rolling(L, H, R):
7     import section
8     import regionToolset
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36 s1.RadialDimension(curve=g[2], textPoint=(-0.264792740345001,
37     -0.16046154499054), radius=R)
38
39
40
41 p = mdb.models['Model-1'].Part(name='Roller', dimensionality=TWO_D_PLANAR,
    type=ANALYTIC_RIGID_SURFACE)

```

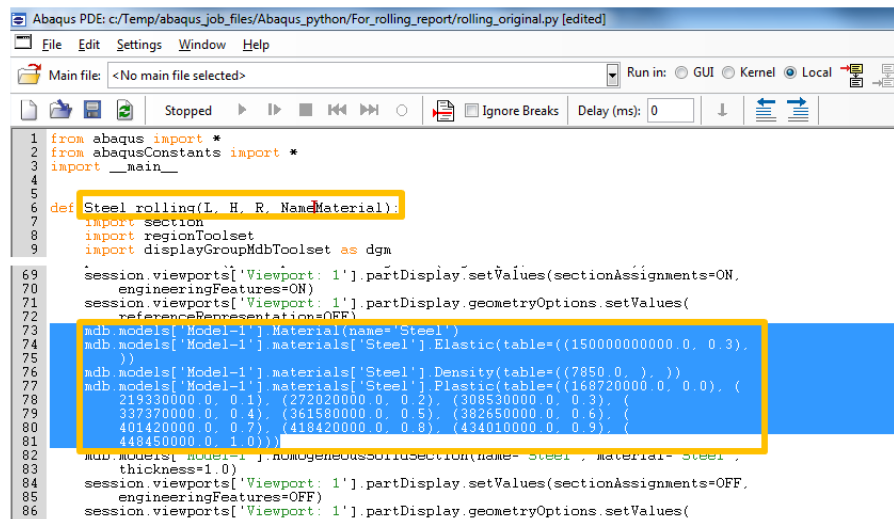
Рис. 18 Добавление переменной R в функцию.

Создадим три разных материала стали, на данном этапе различающиеся только названиями. Для этого, функция будет принимать на вход объект NameMaterial, который необходимо добавить в список переменных. В зависимости от имени переменной «Steel1», «Steel2» или «Steel3», свойства материалов могут

различаться. Присвоение свойств в зависимости от имени материала реализуется с помощью конструкции `if`. За создание материала и присвоение ему механических свойств отвечает участок кода, показанный на рисунке 20. Модифицируйте тело функции так, чтобы создание материала происходило так, как показано на рисунке 21. Определение материала должно выглядеть следующим образом:

```
if NameMaterial == 'Steel1':  
    mdb.models['Model-1'].Material(name=NameMaterial)  
    mdb.models['Model-1'].materials[NameMaterial].Elastic(table=((15000000000.0, 0.3),  
    ))  
    mdb.models['Model-1'].materials[NameMaterial].Density(table=((7850.0, ), ))  
    mdb.models['Model-1'].materials[NameMaterial].Plastic(table=((168720000.0, 0.0), (  
        219330000.0, 0.1), (272020000.0, 0.2), (308530000.0, 0.3), (  
        337370000.0, 0.4), (361580000.0, 0.5), (382650000.0, 0.6), (  
        401420000.0, 0.7), (418420000.0, 0.8), (434010000.0, 0.9), (  
        448450000.0, 1.0)))  
if NameMaterial == 'Steel2':  
    mdb.models['Model-1'].Material(name=NameMaterial)  
    mdb.models['Model-1'].materials[NameMaterial].Elastic(table=((15000000000.0, 0.3),  
    ))  
    mdb.models['Model-1'].materials[NameMaterial].Density(table=((7850.0, ), ))  
    mdb.models['Model-1'].materials[NameMaterial].Plastic(table=((168720000.0, 0.0), (  
        219330000.0, 0.1), (272020000.0, 0.2), (308530000.0, 0.3), (  
        337370000.0, 0.4), (361580000.0, 0.5), (382650000.0, 0.6), (  
        401420000.0, 0.7), (418420000.0, 0.8), (434010000.0, 0.9), (  
        448450000.0, 1.0)))  
if NameMaterial == 'Steel3':  
    mdb.models['Model-1'].Material(name=NameMaterial)  
    mdb.models['Model-1'].materials[NameMaterial].Elastic(table=((15000000000.0, 0.3),  
    ))  
    mdb.models['Model-1'].materials[NameMaterial].Density(table=((7850.0, ), ))  
    mdb.models['Model-1'].materials[NameMaterial].Plastic(table=((168720000.0, 0.0), (  
        219330000.0, 0.1), (272020000.0, 0.2), (308530000.0, 0.3), (  
        337370000.0, 0.4), (361580000.0, 0.5), (382650000.0, 0.6), (  
        401420000.0, 0.7), (418420000.0, 0.8), (434010000.0, 0.9), (  
        448450000.0, 1.0)))  
mdb.models['Model-1'].HomogeneousSolidSection(name='Steel', material=NameMaterial,  
    thickness=1.0)
```

Очень внимательно отнеситесь к отступам. Проверьте, что перед конструкцией `if`, нет пробелов, а внутренние инструкции имеют отступ равный 1 табуляции (4 пробела).

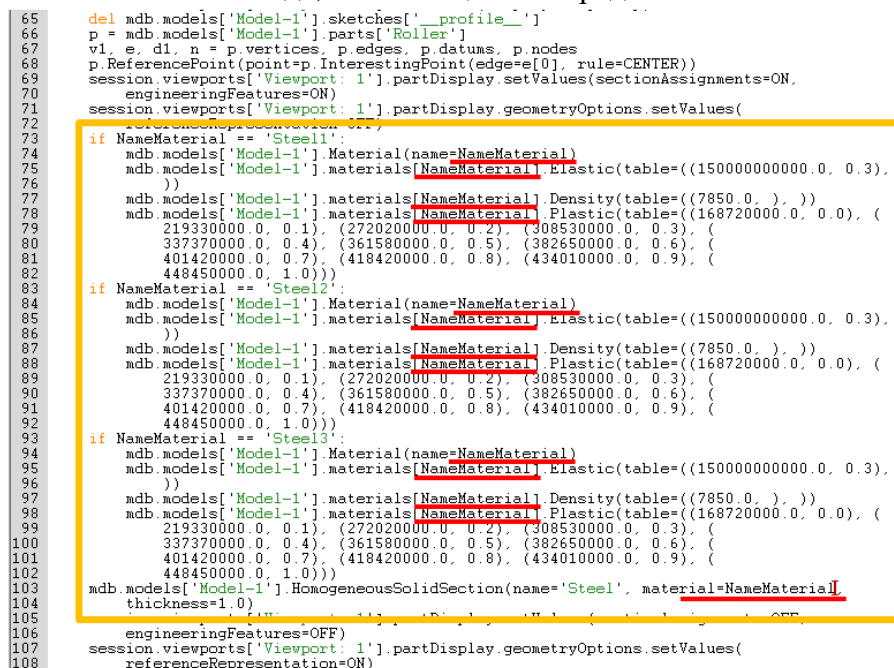


```

1 from abaqus import *
2 from abaqusConstants import *
3 import __main__
4
5
6 def Steel_rolling(L, H, R, NameMaterial):
7     import section
8     import regionToolset
9     import displayGroupHdbToolset as dgm
10
11 session.viewports['Viewport: 1'].partDisplay.setValues(sectionAssignments=ON,
12 engineeringFeatures=ON)
13 session.viewports['Viewport: 1'].partDisplay.geometryOptions.setValues(
14 referenceRepresentation=OFF)
15
16 mdb.models['Model-1'].Material(name='Steel')
17 mdb.models['Model-1'].materials['Steel'].Elastic(table=((150000000000.0, 0.3),
18 ))
19 mdb.models['Model-1'].materials['Steel'].Density(table=((7850.0, ), ))
20 mdb.models['Model-1'].materials['Steel'].Plastic(table=((168720000.0, 0.0), (
21 219330000.0, 0.1), (272020000.0, 0.2), (308530000.0, 0.3), (
22 337370000.0, 0.4), (361580000.0, 0.5), (382650000.0, 0.6), (
23 401420000.0, 0.7), (418420000.0, 0.8), (434010000.0, 0.9), (
24 448450000.0, 1.0)))
25
26 mdb.models['Model-1'].HomogeneousSolidSection(name='Steel', material='Steel',
27 thickness=1.0)
28 session.viewports['Viewport: 1'].partDisplay.setValues(sectionAssignments=OFF,
29 engineeringFeatures=OFF)
30 session.viewports['Viewport: 1'].partDisplay.geometryOptions.setValues(

```

Рис. 20 Участок кода, отвечающий за определение свойств материала.



```

65 del mdb.models['Model-1'].sketches['__profile__']
66 p = mdb.models['Model-1'].parts['Roller']
67 v1, e, d1, n = p.vertices, p.edges, p.datums, p.nodes
68 p.ReferencePoint(point=p.InterestingPoint(edge=e[0], rule=CENTER))
69 session.viewports['Viewport: 1'].partDisplay.setValues(sectionAssignments=ON,
70 engineeringFeatures=ON)
71 session.viewports['Viewport: 1'].partDisplay.geometryOptions.setValues(
72 referenceRepresentation=OFF)
73
74 if NameMaterial == 'Steel1':
75     mdb.models['Model-1'].Material(name=NameMaterial)
76     mdb.models['Model-1'].materials[NameMaterial].Elastic(table=((150000000000.0, 0.3),
77 ))
78     mdb.models['Model-1'].materials[NameMaterial].Density(table=((7850.0, ), ))
79     mdb.models['Model-1'].materials[NameMaterial].Plastic(table=((168720000.0, 0.0), (
80 219330000.0, 0.1), (272020000.0, 0.2), (308530000.0, 0.3), (
81 337370000.0, 0.4), (361580000.0, 0.5), (382650000.0, 0.6), (
82 401420000.0, 0.7), (418420000.0, 0.8), (434010000.0, 0.9), (
83 448450000.0, 1.0)))
84
85 if NameMaterial == 'Steel2':
86     mdb.models['Model-1'].Material(name=NameMaterial)
87     mdb.models['Model-1'].materials[NameMaterial].Elastic(table=((150000000000.0, 0.3),
88 ))
89     mdb.models['Model-1'].materials[NameMaterial].Density(table=((7850.0, ), ))
90     mdb.models['Model-1'].materials[NameMaterial].Plastic(table=((168720000.0, 0.0), (
91 219330000.0, 0.1), (272020000.0, 0.2), (308530000.0, 0.3), (
92 337370000.0, 0.4), (361580000.0, 0.5), (382650000.0, 0.6), (
93 401420000.0, 0.7), (418420000.0, 0.8), (434010000.0, 0.9), (
94 448450000.0, 1.0)))
95
96 if NameMaterial == 'Steel3':
97     mdb.models['Model-1'].Material(name=NameMaterial)
98     mdb.models['Model-1'].materials[NameMaterial].Elastic(table=((150000000000.0, 0.3),
99 ))
100     mdb.models['Model-1'].materials[NameMaterial].Density(table=((7850.0, ), ))
101     mdb.models['Model-1'].materials[NameMaterial].Plastic(table=((168720000.0, 0.0), (
102 219330000.0, 0.1), (272020000.0, 0.2), (308530000.0, 0.3), (
103 337370000.0, 0.4), (361580000.0, 0.5), (382650000.0, 0.6), (
104 401420000.0, 0.7), (418420000.0, 0.8), (434010000.0, 0.9), (
105 448450000.0, 1.0)))
106
107 mdb.models['Model-1'].HomogeneousSolidSection(name='Steel', material=NameMaterial,
108 thickness=1.0)
109
110 session.viewports['Viewport: 1'].partDisplay.setValues(sectionAssignments=OFF,
111 engineeringFeatures=OFF)
112 session.viewports['Viewport: 1'].partDisplay.geometryOptions.setValues(
113 referenceRepresentation=ON)

```

Рис. 21 Модифицированный участок кода, отвечающий за определения материала.
Для того чтобы модель была собрана верно, необходимо определить

взаимное положение стальной плиты относительно ролика. Правый нижний угол плиты находится в начале координат (0, 0). Для того чтобы получить нужную толщину детали на выходе равную h , центр ролика должен иметь координаты $(\sqrt{R^2 - (h + R - H)^2 + \varepsilon}, (h + R))$, как показано на рисунке 22. Параметр $\varepsilon = 10^{-5}$ нужен для того, чтобы гарантировать, что поверхность валика не пересекает верхней поверхности стальной плиты.

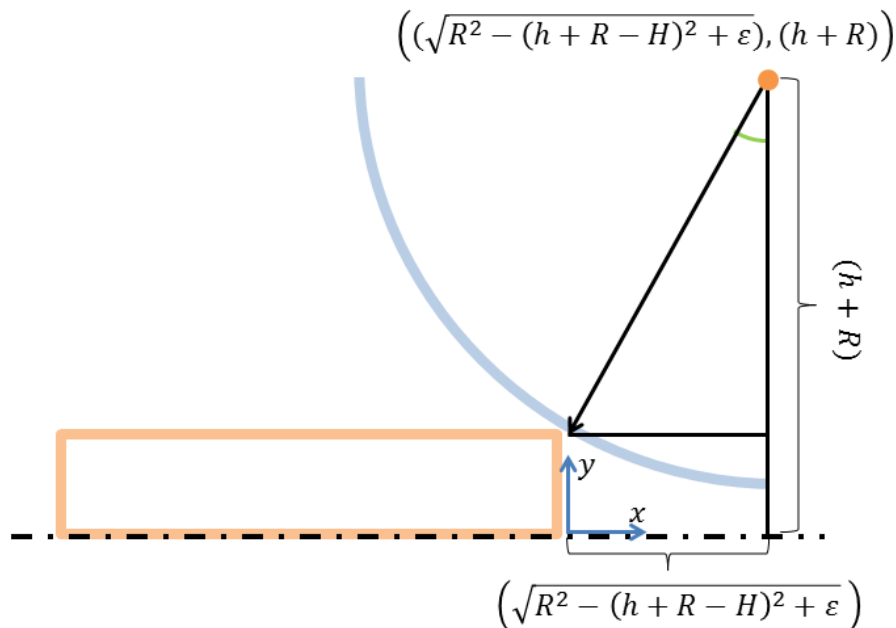
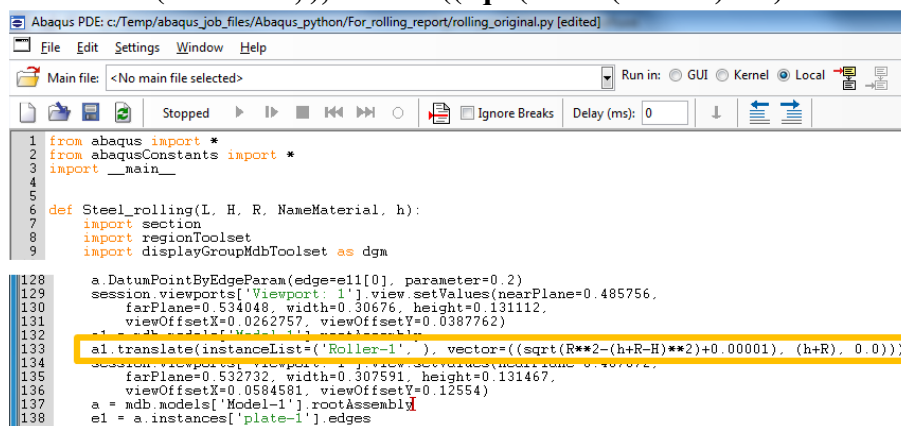


Рис. 22 Схематичное изображение взаимного расположения элементов.

Передайте в функцию объект h и измените строку, выделенную на рис. 23

a1.translate(instanceList=('Roller-1',), vector=((sqrt(R2-(h+R-H)**2)+0.00001), (h+R), 0.0))**



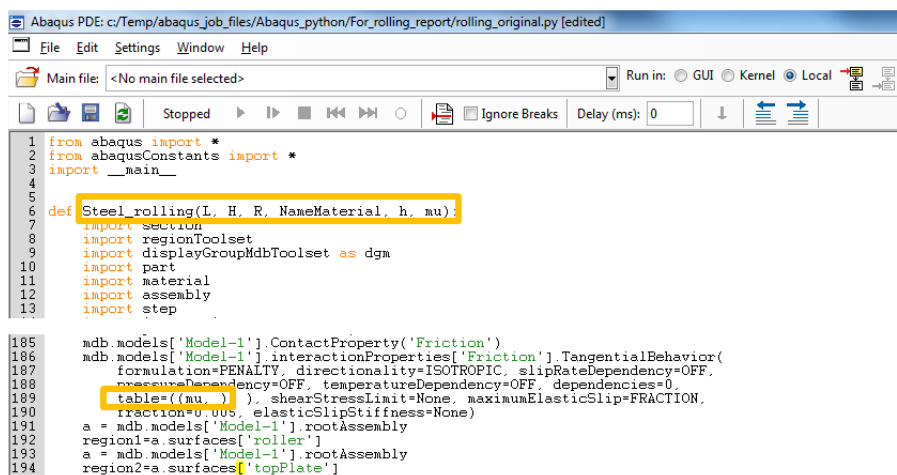
```

1 from abaqus import *
2 from abaqusConstants import *
3 import __main__
4
5
6 def Steel_rolling(L, H, R, NameMaterial, h):
7     import section
8     import regionToolset
9     import displayGroupMdbToolset as dgm
10
11
128 a.DatumPointByEdgeParam(edge=e1[0], parameter=0.2)
129 session.viewports['Viewport: 1'].view.setValues(nearPlane=0.485756,
130 farPlane=0.534048, width=0.30676, height=0.131112,
131 viewOffsetX=0.0262757, viewOffsetY=0.0387762)
132
133 a1.translate(instanceList=('Roller-1', ), vector=((sqrt(R**2-(h+R-H)**2)+0.00001), (h+R), 0.0))
134
135 session.viewports['Viewport: 1'].view.setValues(nearPlane=0.485756,
136 farPlane=0.532732, width=0.307591, height=0.131467,
137 viewOffsetX=0.0584581, viewOffsetY=0.12554)
138 a = mdb.models['Model-1'].rootAssembly
139 e1 = a.instances['plate-1'].edges
    
```

Рис. 23 Измененный участок кода, отвечающий за взаимное положение элементов.

Передайте в функцию объект μ , отвечающий за коэффициент трения и измените строку, выделенную на рисунке 24.

mdb.models['Model-1'].interactionProperties['Friction'].TangentialBehavior(formulation=PENALTY, directionality=ISOTROPIC, slipRateDependency=OFF, pressureDependency=OFF, temperatureDependency=OFF, dependencies=0, table=((mu,),), shearStressLimit=None, maximumElasticSlip=FRACTION, fraction=0.005, elasticSlipStiffness=None)



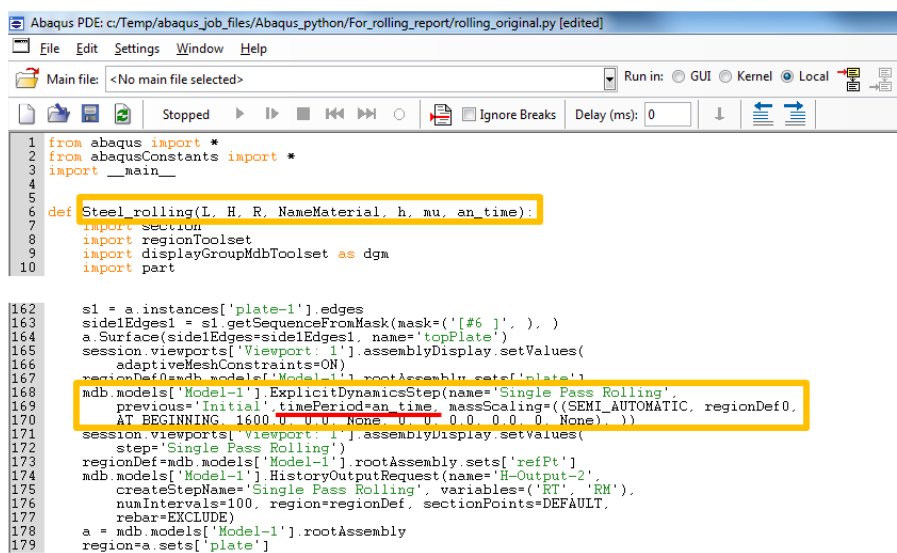
```

1 from abaqus import *
2 from abaqusConstants import *
3 import __main__
4
5
6 def Steel_rolling(L, H, R, NameMaterial, h, mu):
7     import section
8     import regionToolset
9     import displayGroupMdbToolset as dgm
10    import part
11    import material
12    import assembly
13    import step
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185 mdb.models['Model-1'].ContactProperty('Friction')
186 mdb.models['Model-1'].interactionProperties['Friction'].TangentialBehavior(
187     formulation=PENALTY, directionality=ISOTROPIC, slipRateDependency=OFF,
188     shearStressLimit=None, maximumElasticSlip=FRACTION,
189     table=(mu, mu), shearStressLimit=None, maximumElasticSlip=FRACTION,
190     fraction=0.005, elasticSlipStiffness=None)
191 a = mdb.models['Model-1'].rootAssembly
192 region1=a-surfaces['roller']
193 a = mdb.models['Model-1'].rootAssembly
194 region2=a-surfaces['topPlate']

```

Рис. 24 Определение коэффициента трения.

Возможно, для некоторых конфигураций задачи потребуется увеличить время анализа, для этого передайте в функцию объект an_time и выделенном участке кода (Рис. 25) после previous='Initial', вставьте строку **timePeriod=an_time**.



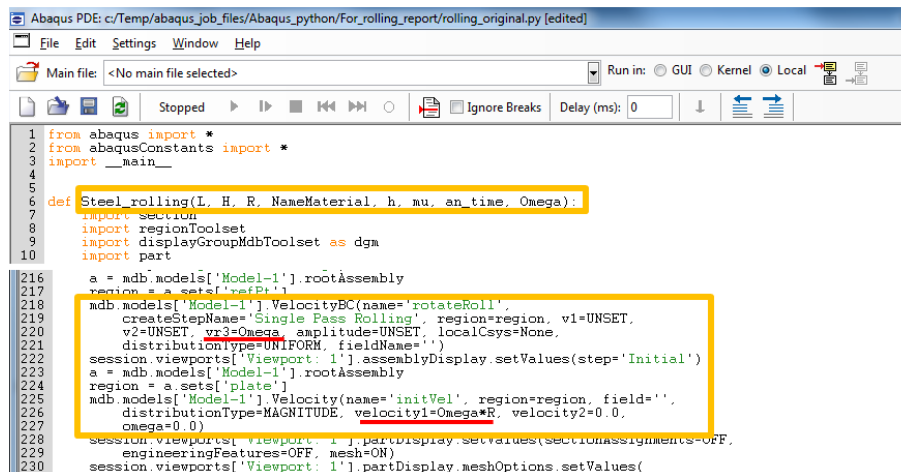
```

1 from abaqus import *
2 from abaqusConstants import *
3 import __main__
4
5
6 def Steel_rolling(L, H, R, NameMaterial, h, mu, an_time):
7     import section
8     import regionToolset
9     import displayGroupMdbToolset as dgm
10    import part
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67 s1 = a.instances['plate-1'].edges
68 side1Edges1 = s1.getSequenceFromMask(mask=('[#6 ]', .))
69 a.Surface(side1Edges=side1Edges1, name='topPlate')
70 session.viewports['Viewport: 1'].assemblyDisplay.setValues(
71     adaptiveMeshConstraints=ON)
72 regionDef=mdb.models['Model-1'].rootAssembly.sets['plate']
73 mdb.models['Model-1'].ExplicitDynamicsStep(name='Single Pass Rolling',
74     previous='Initial', timePeriod=an_time, massScaling=(SEMI_AUTOMATIC, regionDef,
75     AT BEGINNING, 1600, 0, 0, 0, None, 0, 0, 0, 0, 0, 0, None))
76 session.viewports['Viewport: 1'].assemblyDisplay.setValues(
77     step='Single Pass Rolling')
78 regionDef=mdb.models['Model-1'].rootAssembly.sets['refPt']
79 mdb.models['Model-1'].HistoryOutputRequest(name='H-Output-2',
80     createStepName='Single Pass Rolling', variables=('RM', 'RM'),
81     numIntervals=100, region=regionDef, sectionPoints=DEFAULT,
82     rebar=EXCLUDE)
83 a = mdb.models['Model-1'].rootAssembly
84 region=a.sets['plate']
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179

```

Рис. 25 Определение времени анализа.

Угловая скорость вращения валика – объект с именем Omega, который необходимо передать функции Steel_rolling на вход (Рис. 26). Измените значения переменных vr3 на Omega (**vr3=Omega**), отвечающую за скорость вала, и velocity1 на Omega*R (**velocity1=Omega*R**), отвечающую за скорость стальной плиты в начальный момент времени.



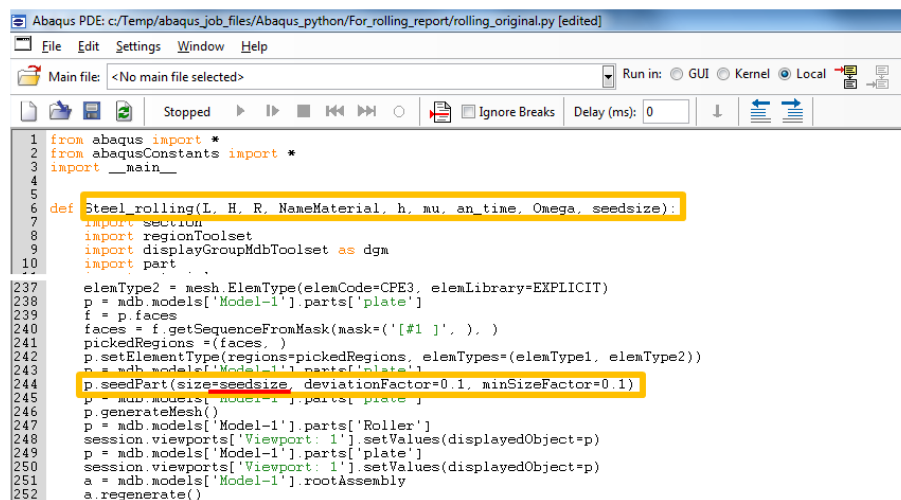
```

1 from abaqus import *
2 from abaqusConstants import *
3 import __main__
4
5
6 def Steel_rolling(L, H, R, NameMaterial, h, mu, an_time, Omega):
7     import section
8     import regionToolset
9     import displayGroupMdbToolset as dgm
10    import part
11
12    a = mdb.models['Model-1'].rootAssembly
13    region = a.sets['refPt']
14    mdb.models['Model-1'].VelocityBC(name='rotateRoll',
15    createStepName='Single Pass Rolling', region=region, v1=UNSET,
16    v2=UNSET, v3=UNSET, amplitude=UNSET, localCsys=None,
17    distributionType=UNIFORM, fieldName='')
18    session.viewports['Viewport: 1'].assemblyDisplay.setValues(step='Initial')
19    a = mdb.models['Model-1'].rootAssembly
20    region = a.sets['plate']
21    mdb.models['Model-1'].Velocity(name='initVel', region=region, field='',
22    distributionType=MAGNITUDE, velocity1=Omega*R, velocity2=0.0,
23    omega=0.0)
24    session.viewports['Viewport: 1'].partDisplay.setValues(sectionAssignments=OFF,
25    engineeringFeatures=OFF, mesh=ON)
26    session.viewports['Viewport: 1'].partDisplay.meshOptions.setValues(

```

Рис. 26 Определение угловой скорости валика и начальной скорости плиты.

Значение переменной `seeds` отвечает за размер конечных элементов, на которые разбивается стальная пластина. Передайте объект `seeds` в функцию. И измените значение параметра `size`, в выделенном на рисунке 27 участке кода на `size=seeds`.



```

1 from abaqus import *
2 from abaqusConstants import *
3 import __main__
4
5
6 def Steel_rolling(L, H, R, NameMaterial, h, mu, an_time, Omega, seeds):
7     import section
8     import regionToolset
9     import displayGroupMdbToolset as dgm
10    import part
11
12    elemType2 = mesh.ElemType(elemCode=CPE3, elemLibrary=EXPLICIT)
13    p = mdb.models['Model-1'].parts['plate']
14    f = p.faces
15    faces = f.getSequenceFromMask(mask=('[#1 ]', ), )
16    pickedRegions = (faces, )
17    p.setElementType(regions=pickedRegions, elemTypes=(elemType1, elemType2))
18    a = mdb.models['Model-1'].rootAssembly
19    p.seedPart(size=seeds, deviationFactor=0.1, minSizeFactor=0.1)
20    p = mdb.models['Model-1'].parts['plate']
21    p.generateMesh()
22    p = mdb.models['Model-1'].parts['Roller']
23    session.viewports['Viewport: 1'].setValues(displayedObject=p)
24    p = mdb.models['Model-1'].parts['plate']
25    session.viewports['Viewport: 1'].setValues(displayedObject=p)
26    a = mdb.models['Model-1'].rootAssembly
27    a.regenerate()

```

Рис. 27 Определение размера конечных элементов.

Проверьте отступы в файле и сохраните файл (File -> Save).

Создание графического окна в RSG Dialog Builder.

Plu-ins-> Abaqus -> RSG Dialog Builder.

Во вкладке GUI в графе Dialog Box Title введите **Rolling**. С помощью инструмента Horizontal Frame создайте горизонтальную группу объектов. Выделив Horizontal Frame дважды создайте вертикальную группу объектов, используя инструмент Vertical Frame, так как показано на рис. 28.

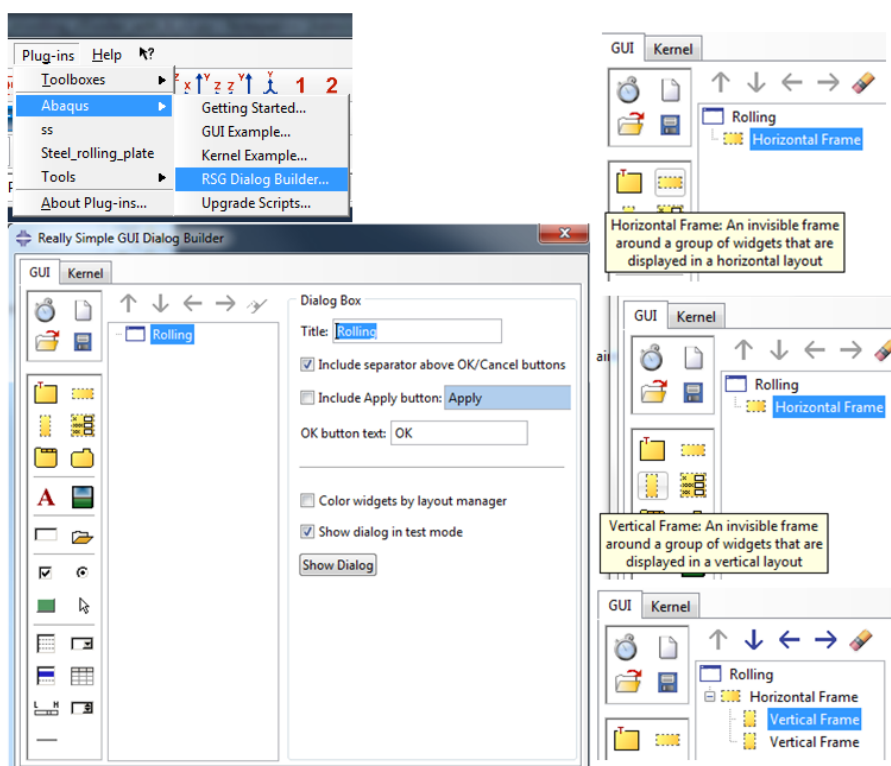


Рис. 28 Интерфейс RSG Dialog Builder.

Выделив первый Vertical Frame, создайте Group Box, с помощью одноименного инструмента. В графе Title введите **Geometry** (Рис. 29).

Выделив объект Geometry создайте четыре текстовых поля для ввода используя инструмент Text Field.

Отредактируйте первое текстовое поле. **Text: Length (m); ; Type: Float; keyword: L ; Default: 0.092.**

Все изменения, производимые в Really Simple GUI Dialog Builder можно отслеживать непосредственно в небольшом окошке, которое появляется автоматически (Рис. 29).

Второе текстовое окно отвечает за ввод толщины стального листа (Рис. 30).

Text: Height1 (m): ; Type: Float; keyword: H ; Default: 0.02.

Третье - за ввод толщины стального листа после прокатки. **Text: Height2 (m): ; Type: Float; keyword: h ; Default: 0.015.**

Четвертое - за радиус прокатного ролика. **Text: Radius (m): ; Type: Float; keyword: R ; Default: 0.17.**

Добавьте разделитель с помощью инструмента Separator.

И создайте Combo Box (Рис. 31). **Type: Standard; Text: Material: ; Keyword: NameMaterial; Default: Steel1;** Выбрав Item измените **Text: Steel1**, второй item -> **Text: Steel2**, третий -> **Text: Steel3**.

Добавим Group Box, Mesh. Выделите Первый Vertical Frame и нажмите на Group Box. Назовите его Mesh. Используя “стрелки” перенесите его ниже Group Box, Geometry (Рис. 32). Создайте текстовое поле и отформатируйте его. **Text: Element size (m): ; Type: Float; keyword: seedsizes ; Default: 0.002.**

Создайте Group Box с именем Experiment setup, переместите его под модуль Mesh. Внутри данного модуля создайте два текстовых поля и отформатируйте их. Первое: **Text: Angular velocity (rad/s): ; Type: Float; keyword: Omega ; Default: 6.28319.** Второе: **Text: Friction coefficient: ; Type: Float; keyword: mu ; Default: 0.4.**

Создайте Group Box с именем Analysis setup, переместите его под модуль Experiment setup (Рис. 29). Создайте одно текстовое поле. **Text: Analysis time: ; Type: Float; keyword: an_time ; Default: 0.1.**

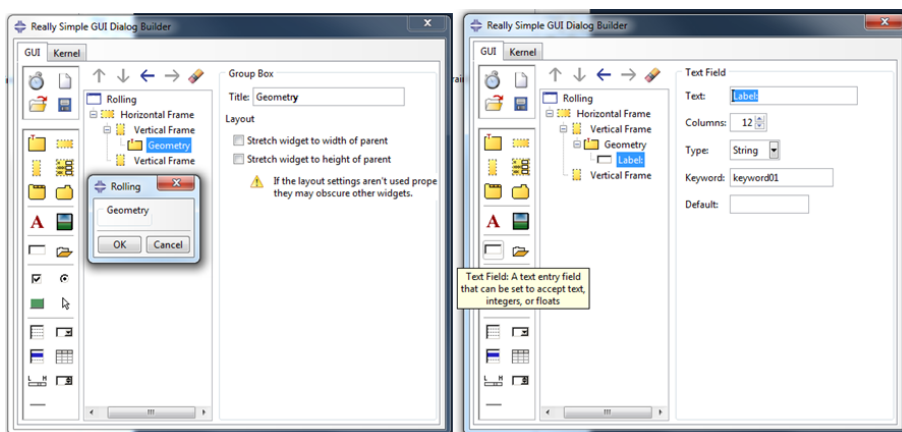


Рис. 29 Интерфейс RSG Dialog Builder.

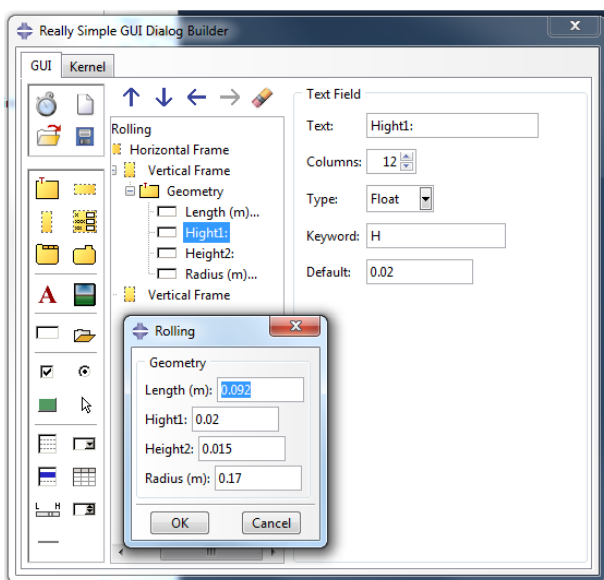


Рис. 30 Создание модуля Geometry.

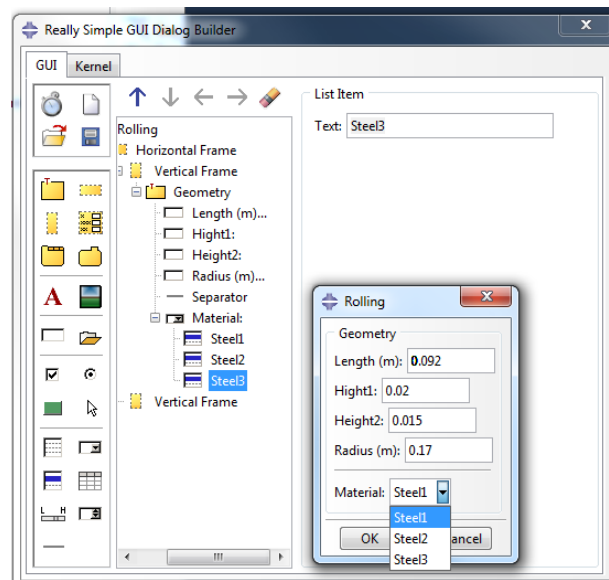


Рис. 31 Создание модуля Material.

Добавим поясняющую картинку к данной задаче. Выберите второй Vertical Frame и создайте Icon с помощью одноименного инструмента. В появившемся окне укажите полный путь к изображению. В результате диалоговое окно будет выглядеть, как показано на рисунке 34.

Объединим созданную нами python функцию с графической оболочкой. Перед этим в рабочей директории создайте копию файла rolling.py, чтобы избежать его потери. Перейдите во вкладку Kernel (Рис. 35). В окне Module: выберите инструмент Load a kernel module file from the File Selection dialog и выберите функцию rolling.py. В графе Function: выберите созданную нами функцию Steel_rolling.

Перейдите обратно во вкладку GUI и сохраните плагин с помощью инструмента Save your dialog box as a plug-in.

В качестве Directory name и Menu button name введите **Steel_rolling** (Рис. 36 слева). Выберите место для сохранения плагина в модуле Location. В нашем случае это Home directory и нажмите ОК. После чего, Abaqus покажет место, куда были сохранены данные созданного плагина и попросит перезапустить программу, для того чтобы воспользоваться созданным плагином.

Перезапустите Abaqus. Запустите созданный плагин. Plug-ins -> Steel_rolling (Рис. 36 справа). Подтвердите ввод параметров нажатием клавиши ОК и убедитесь в том, что модель была создана и отправлена на расчёт. При возникновении ошибки, Abaqus укажет на номер строки в файле кода, которую необходимо исправить.

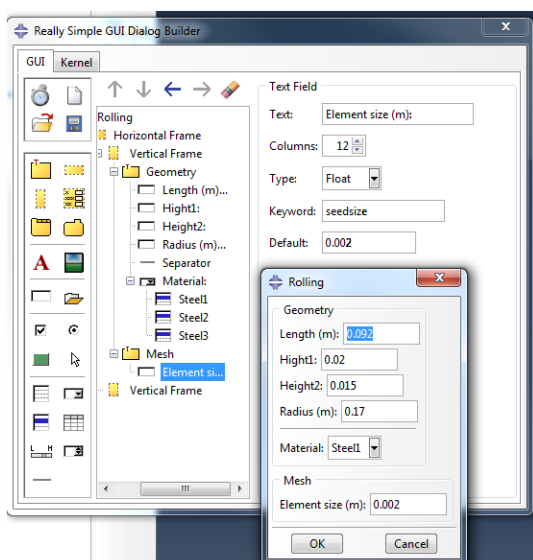


Рис. 32 Создание модуля Mesh.

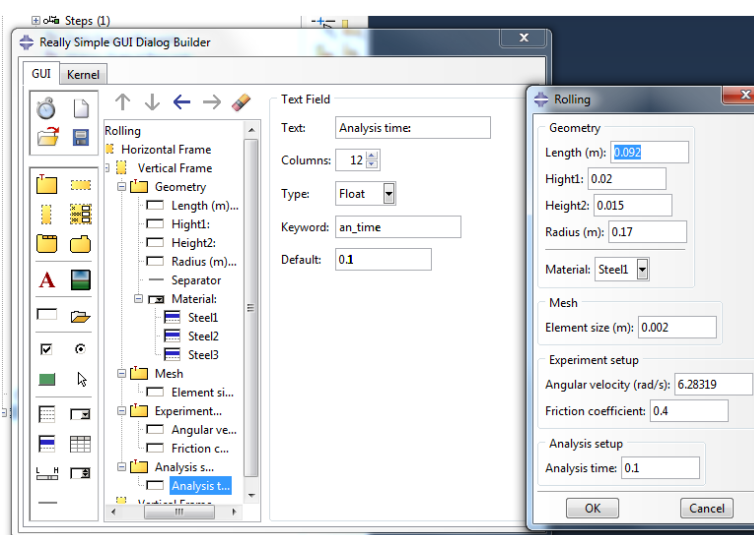


Рис. 33 Создание модуля Analysis time.

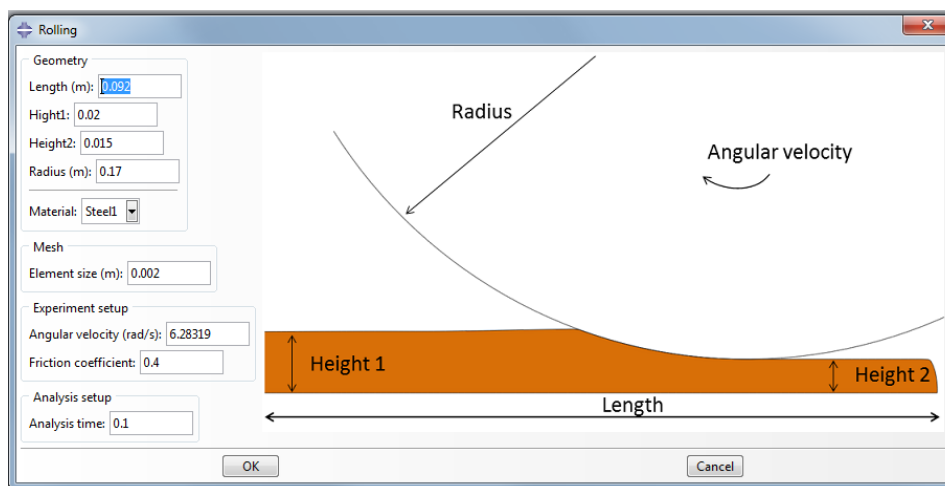


Рис. 34 Окно созданного графического пользовательского интерфейса.

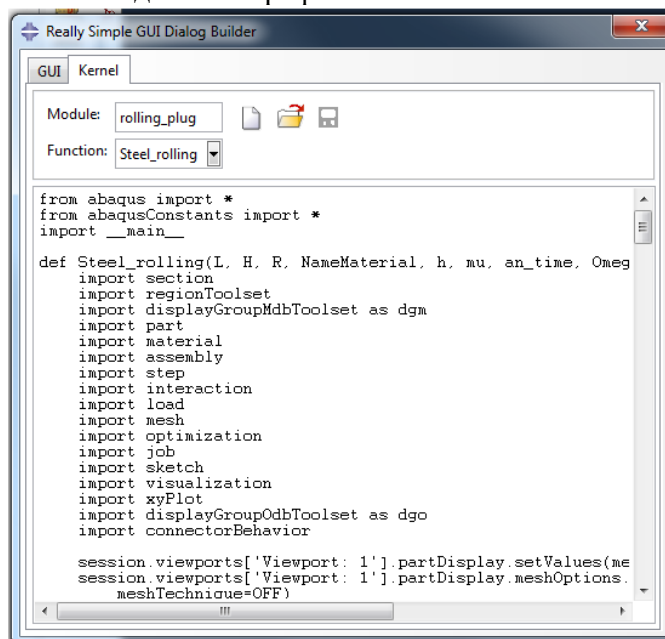


Рис. 35 Вкладка Kernel. RSG Dialog Builder.

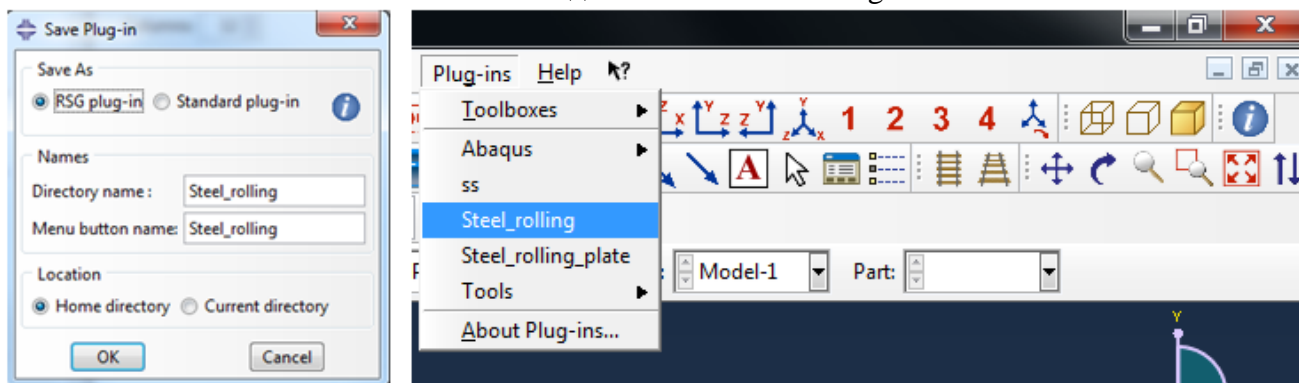


Рис. 36 Сохранение плагина и его последующее использование.

Заключение:

В данном пособии рассмотрен процесс создания графического интерфейса пользователя в программном комплексе SIMULIA Abaqus. Данный подход является универсальным и позволяет создавать интерактивные оболочки для задач любой сложности. Для более глубокого понимания Python инструкций, рекомендуем обратиться к Abaqus Scripting User's Guide. Использование Python в инженерных задачах может существенно сократить время анализа путём автоматизации рутинных операций, уменьшить вероятность совершения ошибок пользователем и выполнять оптимизацию.

Python – свободный объектно-ориентированный язык программирования интегрированный в SIMULIA Abaqus. На официальном сайте python.org можно найти исчерпывающее описание языка, документацию и примеры. Существует две версии языка Python 2.x и Python 3.x, Abaqus работает с версией 2.