ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ   
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

###### Московский институт электроники и математики

Евтеева Марина Леонидовна

**КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ ПРОЦЕССОВ ФОРМОВКИ СВЕРХПЛАСТИЧНЫХ МАТЕРИАЛОВ**

Междисциплинарная курсовая работа

студента образовательной программы   
 «Прикладная математика»,

группы БПМ-143

|  |  |
| --- | --- |
| \_\_\_\_\_\_\_\_\_\_\_\_\_  Подпись студента  М. Л. Евтеева | Руководитель  Доцент Московский институт электроники и математики им. А.Н. Тихонова / Департамент прикладной математики  С.А. Аксенов |
| Москва 2016 г. | |

# Аннотация

Работа посвящена реализации алгоритмов контактного взаимодействия при решении краевых задач механики деформируемого твердого тела, возникающих при моделировании процессов обработки материалов давлением. Разработаны программные компоненты, позволяющие вносить в матрицу жесткости граничные условия, соответствующие трению деформируемого тела о поверхность абсолютно твердого объекта. Для верификации реализованных функций была проведена серия тестовых расчетов по моделированию процессов газовой формовки сверхпластичных материалов.

# Abstract

The purpose of this report is to show the results of developing add-ons that allow to solve problems that includes friction between finite element mesh and tool boundaries during metal forming. Implementation is based on boundary value problem in solid mechanics. To verify accuracy of the changes there are presented several numerical simulations: free bulging, superplastic forming of a spherical shell.

Оглавление

[Аннотация 2](#_Toc482467700)

[Abstract 2](#_Toc482467701)

[Введение 4](#_Toc482467702)

[Основная часть 6](#_Toc482467703)

[1.1 Постановка задачи формоизменения 6](#_Toc482467704)

[1.2 Метод конечных элементов 7](#_Toc482467705)

[1.3 Особенности формирования матрицы жесткости 9](#_Toc482467706)

[1.4 Контактное взаимодействие с инструментами. Реализация трения скольжения. 11](#_Toc482467707)

[1.5 Некоторые расчеты с реализованным контактным взаимодействием 12](#_Toc482467708)

[Заключение 16](#_Toc482467709)

[Список литературы 17](#_Toc482467710)

[Приложение 18](#_Toc482467711)

[1.1 Код для проверки принадлежности точки фигуре 18](#_Toc482467712)

[1.2 Получение проекции точки на инструменте и нахождение угла 18](#_Toc482467713)

[1.3 Внесение граничных условий в матрицу жесткости 19](#_Toc482467714)

# Введение

Обработка материалов давлением (ОМД) - одна из важнейших отраслей машиностроения, переживающая очередной этап своего развития, вызванный широким внедрением компьютерных технологий. Современные методы математического и компьютерного моделирования позволяют на стадии проектирования технологического процесса проводить оценку и оптимизацию энергозатратности производства, анализировать особенности течения материала при реализации технологических операций, выбирать и настраивать технологические режимы, обеспечивающие требуемое качество конечной продукции. Широкое использование таких моделей привело к распространению коммерческих программных пакетов, позволяющих решать краевые задачи механики сплошной среды, моделирующие технологические процессы ОМД. Для решения краевых задач используется метод конечных элементов (далее - МКЭ) эффективность которого подтверждена многолетним опытом его применения.

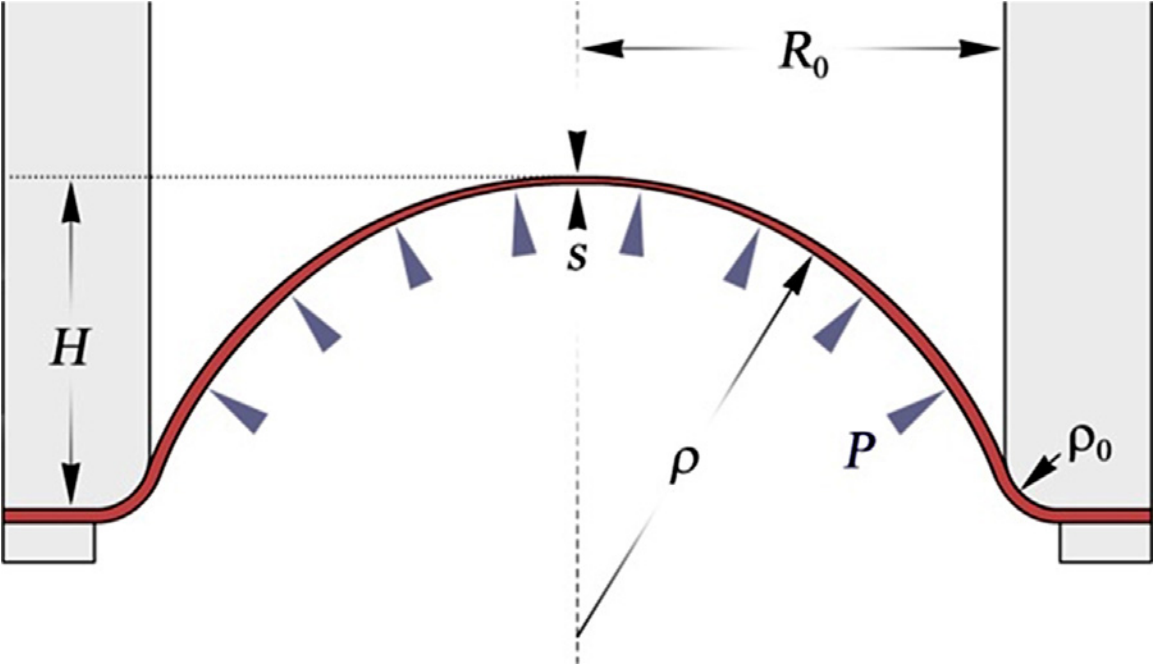
Использование компьютерного моделирования при проектировании технологических процессов ОМД позволяет максимально эффективно использовать особенности деформируемых материалов, таких, например, как сверхпластичность. Применение эффекта сверхпластичности в процессах газовой формовки позволяет производить за одну технологическую операцию оболочечные изделия сложной геометрической формы, которые не могут быть получены с помощью классического горячего формоизменения. Основной особенностью процессов сверхпластичной формовки является необходимость применения специфических для каждого конкретного изделия режимов давления. Расчет режима давления осуществляют с помощью компьютерного моделирования таким образом, чтобы скорость деформации в заготовке не выходила из узкого диапазона сверхпластичности. Адекватность результатов расчета, в первую очередь определяется корректностью задания уравнений состояния материала и граничных условий задачи.

Целью работы является разработка программных модулей, реализующих алгоритмы внесения граничных условий, соответствующих условиям трения скольжения между заготовкой и штампом, при решении задачи формоизменения с помощью МКЭ.

# Основная часть

## Постановка задачи формоизменения

Пусть заготовка – твердое тело – в начальный момент времени покоится, зажатая между цилиндрической матрицей и прижимом при температуре сверхпластичности. Далее в рабочую полость подается газ, по соответствующему режиму давления, обеспечивающему постоянную скорость деформации. Процесс изображен на рис. 1



*Рис.1 Схема формовки в цилиндрическую матрицу*

Матрица характеризуется радиусом и радиусом скругления , заготовка – толщиной , в процессе формовки также собственным радиусом и высотой в каждый момент времени. Также известно давление .

При такой постановке задачи на разных частях граничной поверхности заданы различные граничные условия:

1. Граничные условия первого рода или кинематические условия соответствуют случаю, когда на поверхности задается вектор перемещений или скоростей , где – единичные вектора, направленные вдоль осей координат
2. Граничные условия второго рода или силовые условия задаются в том случае, когда на поверхности действуют поверхностные силы. Тогда вектор поверхностных сил на элементарной площадке задается таким образом: , а сами граничные условия , где – компоненты тензора напряжений.
3. Смешанные граничные условия, описывающие случай, когда на поверхности задаются значения и кинематических, и динамических величин, а также возможно задание взаимосвязи между ними. Принцип задания таков, что в каждом из направлений вдоль осей координат задается либо кинематическое, либо динамическое граничное условие.

В работе реализовывался программный модуль, позволяющий использовать смешанные граничные условия для более правильного описания поведения заготовки при взаимодействии с матрицей.

## Метод конечных элементов

Метод конечных элементов (далее - МКЭ) довольно часто используется для решения задач прикладной физики, т.к. позволяет перейти от сложной непрерывной среды к более простой дискретной. Рассматриваемая в работе область применения МКЭ – решение задачи механики деформируемого твердого тела, моделирование этого процесса.

Идея метода заключается в том, чтобы перейти от непрерывной искомой функции к конечному числу ее значений, которые определены в узлах сетки [1]. Сетка строится разбиением области на достаточно малые элементы, соединенные в конечном числе узлов, расположенных на их границе. Скорости перемещения внутри элемента определяются через скорости перемещения узлов, при помощи некоторой аппроксимирующей функции. Ее вид напрямую зависит от выбранных элементов, поэтому принято выбирать наиболее простые геометрические фигуры: треугольник или прямоугольник для плоской задачи, для осесимметричной – треугольный или прямоугольный в сечении тор и для объемной задачи – параллелепипед или тетраэдр.

Получение скорости перемещения в любой точке внутри элемента выражается уравнением (2) в матричной форме, где – вектор скоростей произвольной точки, – матрица функций формы (положения) и – скорости перемещения узловых точек рассматриваемого элемента

()

А основными в МКЭ являются матричное уравнение жесткости элемента () и глобальное для всей системы) (в общем виде)

()

()

, где – матрица жесткости элемента, – вектор узловых перемещений элемента, – внутренние силы, действующие на элемент e, замененные на эквивалентные узловые силы. и – внешние распределенные поверхностные и массовые силы, приведенные к эквивалентным узловым силам, – глобальная матрица жесткости, заполняющаяся по принципу ,

– вектор заданных внешних узловых сил с глобальной нумерацией

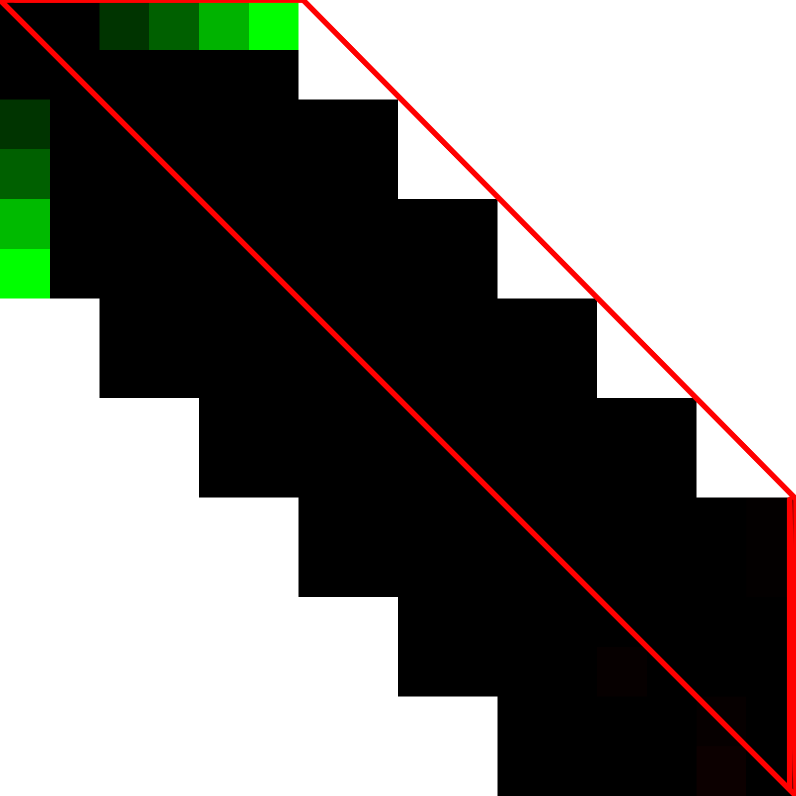
, а – глобальные векторы узловых сил, эквивалентных распределенным поверхностным и массовым силам, заполняются аналогично .

Таким образом, задача сводится к поиску скоростей узлов и распределению их на элементы, дабы смоделировать поведение всего тела.

## Особенности формирования матрицы жесткости

Для формирования матрицы жесткости в работе использовался программный код, реализующий алгоритмы, описанные в [1]. Хранение матрицы жесткости реализовано с учетом ее свойств: симметричности и ленточности.

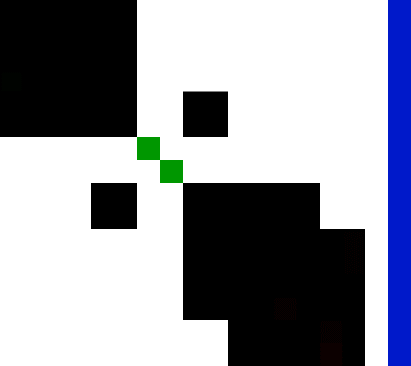
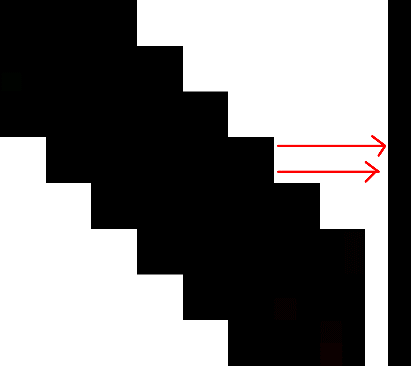
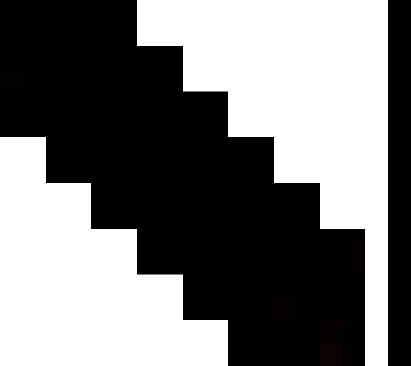
Рис. 2 Глобальная матрица жесткости



На Рис.2 схематично изображена глобальная матрица жесткости для 6 элементов. Белым цветом изображены ячейки, содержащие нули. В первой строке и первом столбце ячейки одного цвета содержат одинаковые значения, поскольку матрица симметричная, программа хранит и работает только с верхней половиной ленты, как отмечено на картинке линией (диагональные элементы включены).

Программно это реализовано через шаблонный класс CBandMatrix, хранящий в себе количество строк, ширину ленты и сами значения, лежащие в шаблонном классе CCustomArray, представляющий из себя одномерный массив. Для простоты реализовано индексация в массиве как в матрице, поэтому проблем с обращением к нужному элементу не возникает.

До добавления трения скольжения были только возможности задавать определенные граничные условия первого рода - запрета перемещения по нужной степени свободы некоторого узла - реализуя, например, заделку.



*А) Б) В)*

*Рис.3 Схема внесения кинематических граничных условий. А) исходная матрица жесткости и правая часть. Б) изменение правой части В) матрица жесткости и правая часть после внесения граничных условий*

На Рис. 3 схематично изображено, как внесение граничных условий 1 рода выглядит на глобальной матрице жесткости и правой части. Алгоритмически для k-го узла это происходит так:

1. В векторе правой части (действующие силы) установить на k-е место граничные условия, получающиеся из действующих сил
2. В глобальной матрице жесткости элемент
3. Пройти по всему вектору правой части , , где n – число строк матрицы
4. Занулить k-ую строку и столбец, оставив только единицу на диагонали
5. После выполнения пунктов 1-4 получился запрет перемещения по оси OX. Повторить для k+1 строки, чтобы получить запрет на перемещение по оси OY.

В общем случае, после внесения всех граничных условий, нам остается только решить СЛАУ , где ­ – искомый вектор скоростей узлов.

В программном коде решение СЛАУ осуществляется методом Гаусса: приведением матрицы к ступенчатому виду и нахождением решений обратным ходом.

## Контактное взаимодействие с инструментами. Реализация трения скольжения.

В работе была реализована возможность задавать коэффициент трения на каждом из инструментов во время создания нового проекта. Эти коэффициенты будут использованы для расчета сил трения между узлами конечноэлементной сетки и линиями инструмента.

В ходе самих расчетов на каждом шаге осуществляется проверка принадлежности граничных узлов инструменту. Это было реализовано по классическому методу проверки принадлежности точки фигуре: из точки выпускается луч и производится подсчет пересечений этого луча с отрезками, образующими стороны фигуры. Этот метод работает и для невыпуклых фигур без самопересечений. Листинг реализации этого метода приведен в Приложении. Если узел принадлежит инструменту, то на соответствующей линии инструмента проверяется коэффициент трения. Если пользователь задал его равным единице, то считается, что для узла задано граничное условие типа «прилипание», в следствие чего он движется вместе с инструментом. Если же коэффициент трения меньше единицы, то производится расчет силы трения.

Для того, чтобы понять направление силы трения и движения узла, необходимо получить касательную к инструменту. Для этого нужно спроецировать узел на инструмент и найти угол между проекцией и ближайшим узлом инструмента. Это и будет угол наклона искомой касательной. И вдоль этой касательной позже зададается симметрия по оси OX повернутой на угол . Если инструмент находится в движении, то необходимо спроецировать эту скорость в повернутой системе координат, иначе будет потеряно это внешнее воздействие. Это преобразование можно получить по формуле . Тогда движение узла в повернутой системе координат задается по оси OY и по оси OX.

Ровно так и вносятся граничные условия на данном узле в вектор правой части. Поскольку в ней все происходит в повернутой системе координат, необходимо повернуть и соответствующие строки и столбцы глобальной матрицы жесткости. При этом элементы поворачиваются два раза. Для самих столбцов и строк производится стандартное умножение на матрицу поворота , а для элементов на пересечении считается по формулам (10)

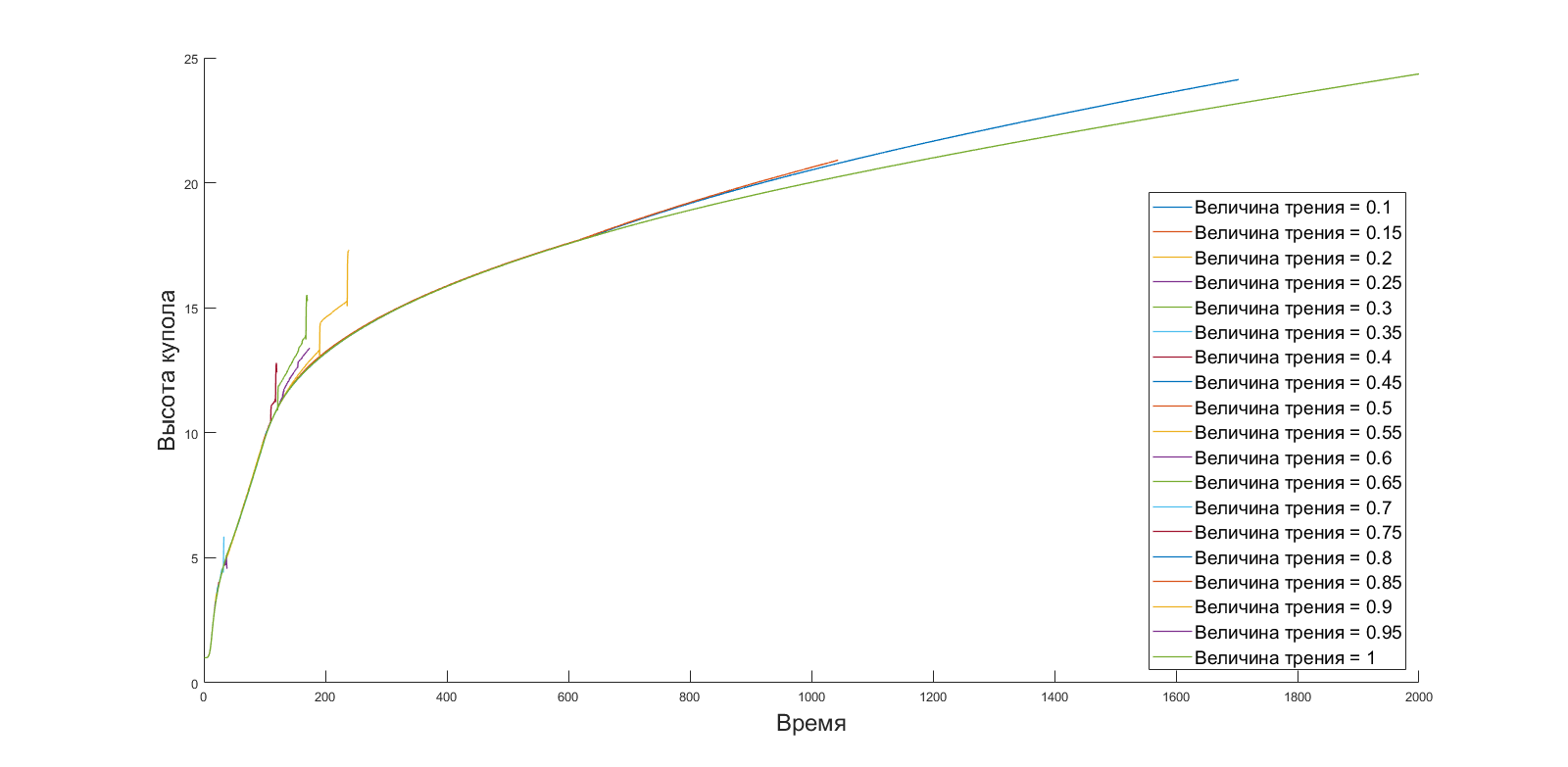
()

И так для всех узлов конечноэлементной сетки. После этого задается симметрия по оси OX, методом, описанным в параграфе 1.2 и система решается по методу Гаусса. После этого правая часть поворачивается назад на .

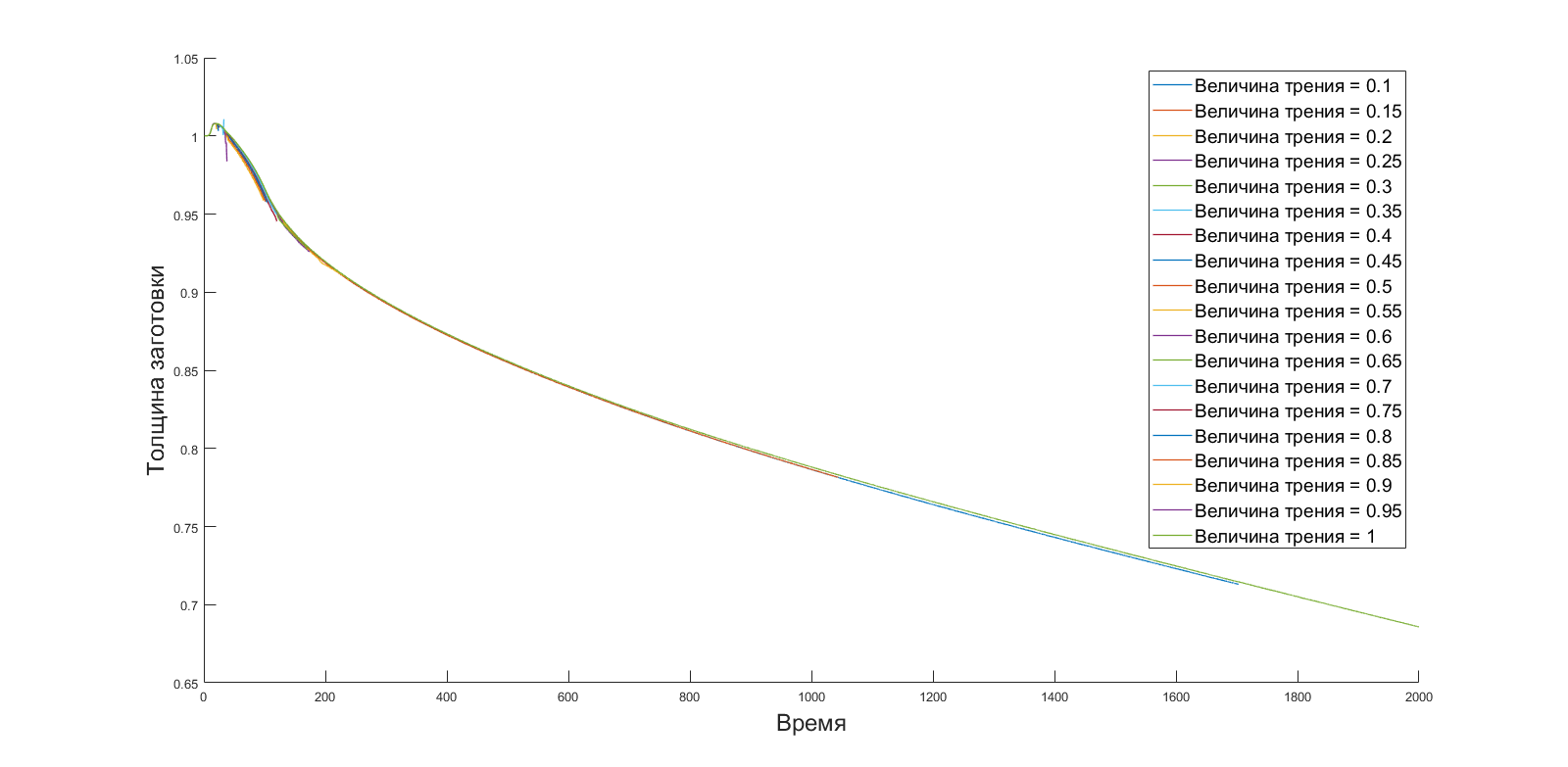
## Некоторые расчеты с реализованным контактным взаимодействием

Первой была проведена серия расчетов по формовке в цилиндрическую матрицу, варьируя коэффициент трения. Кроме этого ничего не изменялось: толщина заготовки 1 мм, длина 50 мм, давление 0.4 МПа, радиус скругления 5 мм, свойства материала K = 155.7, m = 0.265 [2], формовка длилась либо 2000 секунд, либо до разрыва.

На рис. 4 и 5 изображены графики высоты купола от времени и толщины купола от времени для разных коэффициентов трения. Их величины не так сильно различаются, но очень заметен тот факт, что рост коэффициента трения вызывает более ранний разрыв заготовки.

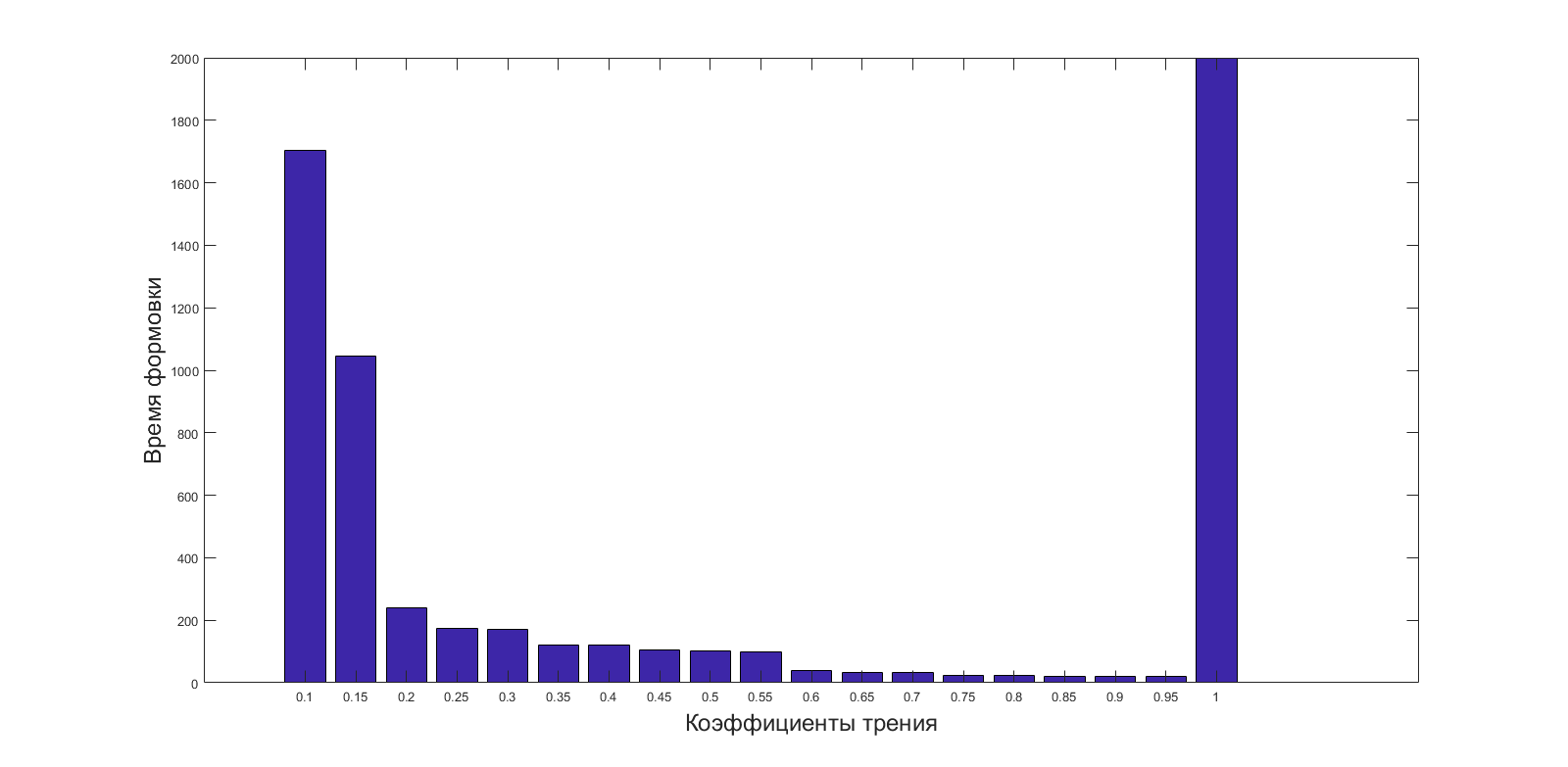


*Рис. 4 Зависимость высоты купола от времени для разных коэффициентов трения*



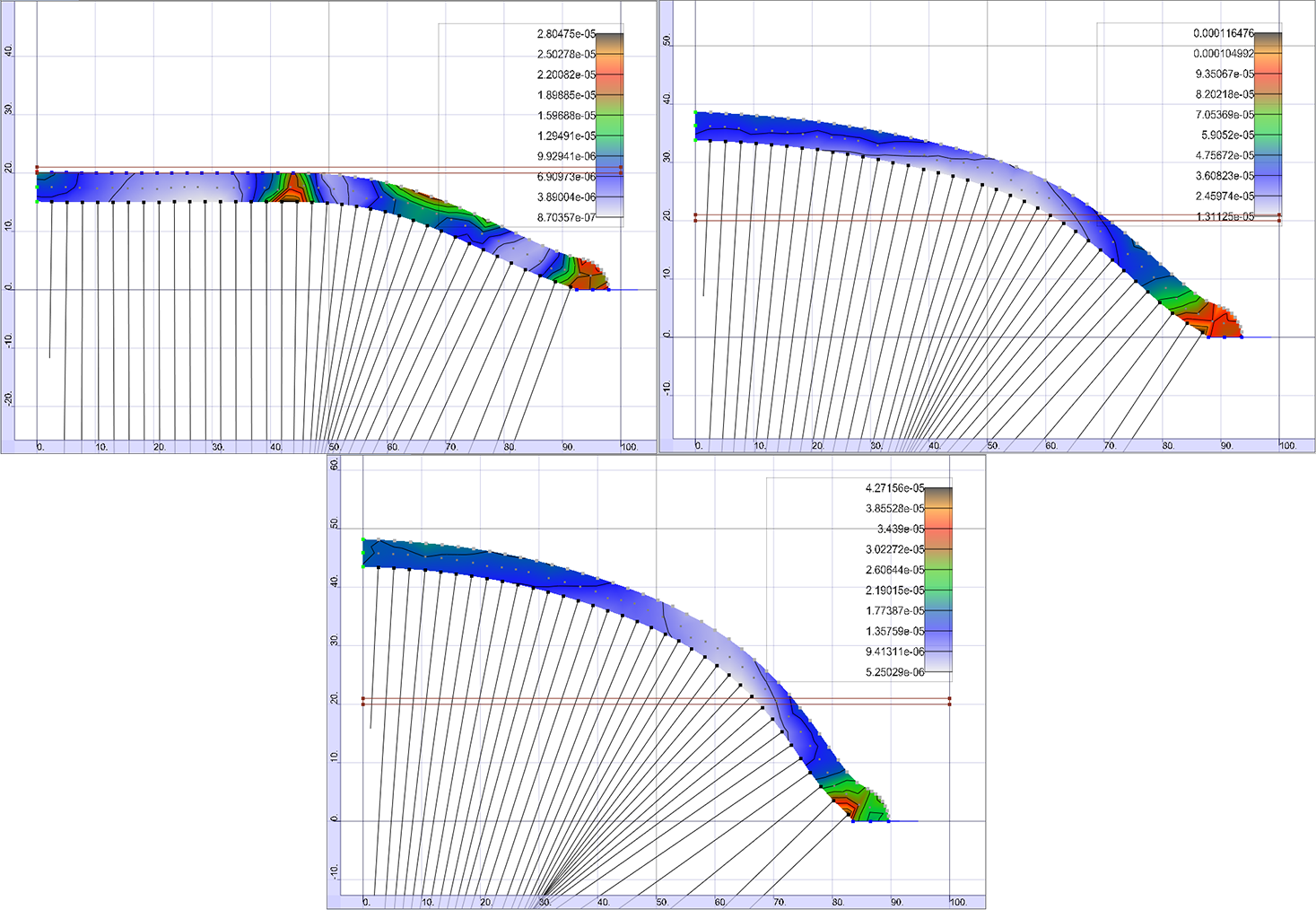
*Рис. 5 Зависимость толщины заготовки от времени для разных коэффициентов трения*

На рис. 6 наглядная зависимость времени до разрыва заготовки от коэффициента трения. Коэффициент трения равный 1 подразумевает «прилипание» узлов конечноэлементной сетки к матрице

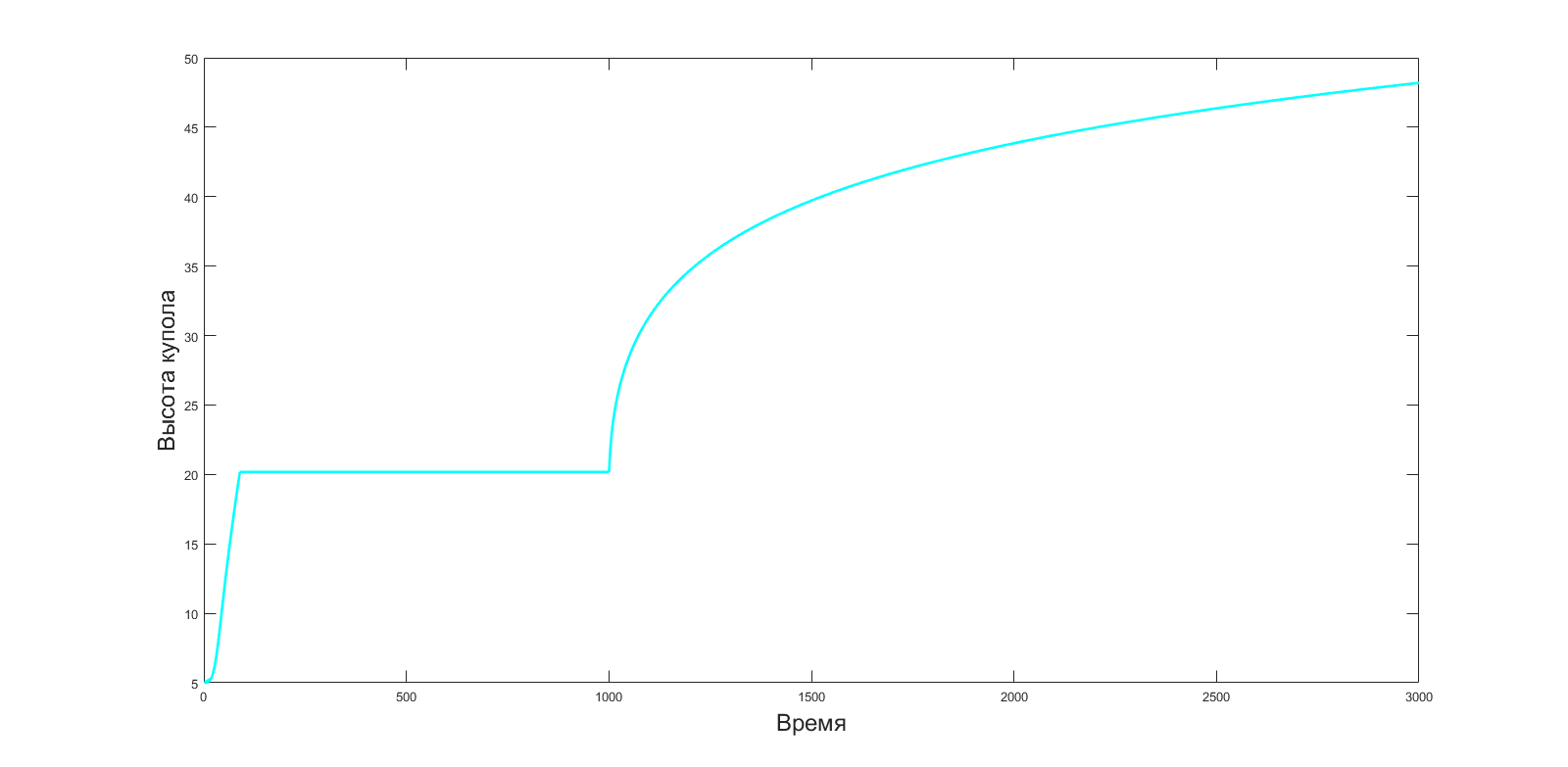


*Рис. 6 График зависимости времени формовки от коэффициентов трения*

Еще одна реализованная разновидность формовки – свободная формовка [3]. Это один из способов получения целой металлической сферы, когда диффузионной сваркой соединяют две заготовки по периметру и между ними подают газ, позволяя ему распирать их изнутри. Данный тип формовки проводится в 2 этапа: сперва с ограничительным инструментом, затем он убирается, и формовка продолжается. На Рис.7 показана реализация этого процесса в программе. На первой части изображена формовка до инструмента, после заданного момента инструмент перестает действовать на заготовку, и формовка продолжается до заданного времени.



*Рис. 7 Этапы свободной формовки*

И соответствующий график высоты купола от времени изображен на рис. 7. В данной задаче толщина заготовки 5 мм, длина 100 мм, давление 0.4 Мпа, свойства материала K = 450, m = 0.4, коэффициент трения = 0.15

*Рис. 7. График H(t) для свободной формовки*

# Заключение

Был реализован программный модуль, реализующий алгоритмы внесения граничных условий, соответствующих условиям трения скольжения между заготовкой и штампом при решении задачи формоизменения с помощью метода конечных элементов – внесение в матрицу жесткости граничных условий. Также была проведена серия тестовых расчетов по моделированию сверхпластичной формовки. По результатам расчетов можно заметить зависимость времени, в которое происходит разрыв заготовки, от коэффициента трения между матрицей и листом.

# Список литературы

|  |  |
| --- | --- |
| [1] | Чумаченко Е.Н., Логашина И.В., Математическое моделирование течение металла при прокатке, Москва: Московский институт электроники и математики, 2005. |
| [2] | S.A. Aksenov, E.N. Chumachenko, A.V. Kolesnikov, S.A. Osipov, «Determination of optimal gas forming conditions from free bulging,» *Journal of Materials Processing Technology 217,* pp. 158-164, 2014. |
| [3] | A.A. Kruglov, F.U. Enikeev, R.Ya. Lutfullin, «Superplastic forming of a spherical shell out a welded envelope,» *Materials Science and Engineering A323,* pp. 416-426, 2002. |

# Приложение

## Код для проверки принадлежности точки фигуре

Math::C2DPoint DP1 = pcontour->GetCache()[0];

bool bLeft = DP1.x < p.x ? true : false;

size\_t r = pcontour->GetCache().size();

int intrsct = 1;

if (p == DP1) return 0;

while (r--)

{

Math::C2DPoint DP2 = DP1;

DP1 = pcontour->GetCache()[r];

intrsct \*= IsIntersection(DP1, DP2, p);

}

return intrsct;

int C2DOutline::IsIntersection(const Math::C2DPoint& a, const Math::C2DPoint& b, const Math::C2DPoint& middle) {

double ax = a.x - middle.x;

double ay = a.y - middle.y;

double bx = b.x - middle.x;

double by = b.y - middle.y;

//лежит ли отрезок по одну сторону от луча

if (ay \* by > 0)

return 1;

int s = sgn(ax \* by - ay \* bx);

if (s == 0)

{

//совпадение с осью

if (ax \* bx <= 0)

return 0;

return 1;

}

//пересечение отрезка лучом, знак зависит от того, с какой стороны точка

if (ay < 0)

return -s;

if (by < 0)

return s;

return 1;

}

## Получение проекции точки на инструменте и нахождение угла

int nNode = m\_shape.GetClosestNode(pMesh->GetBorderNode(nBoundaryNode), dist); //получаем ближайший узел

if (nNode == -1) return false;

DBL closep = m\_shape.GetNode(nNode)->GetPoint();

Math::C2DPoint minim, clstnd;

//находим все кривые с этим узлом

for (size\_t i = 0; i < m\_shape.GetCurveCount(); i++)

{

C2DCurve \*pCur = m\_shape.GetCurve(i);

if (pCur->GetStart() == nNode || pCur->GetEnd() == nNode)

{

//находим ближайшую точку на кривой и сравниваем с предыдущей

int p = pCur->GetClosestPoint(pMesh->GetBorderNode(nBoundaryNode), minim);

//int p\_1 = pCur->GetClosestPoint(pMesh->GetBorderNode(nBoundaryNode2), minim);

if (p == -1) return false;

// DBL m = pMesh->GetBorderNode(nBoundaryNode).Len(minim);

//n1 = m\_shape.GetNode((pCur->GetStart() == nNode ? pCur->GetEnd() : pCur->GetStart()))->GetPoint();

if (dist > pMesh->GetBorderNode(nBoundaryNode).Len(minim))

{

dist = pMesh->GetBorderNode(nBoundaryNode).Len(minim);// получаем расстояние от точки Заготовки до Инструмента

// dist\_1 = pMesh->GetBorderNode(nBoundaryNode2).Len(minim);

clstnd = minim;

closep = dist;

}

}

}

DBL testangle = m\_shape.GetNode(nNode)->GetPoint().x - clstnd.x ? atan((m\_shape.GetNode(nNode)->GetPoint().y - clstnd.y) / (m\_shape.GetNode(nNode)->GetPoint().x - clstnd.x)) : 1.5708;

## Внесение граничных условий в матрицу жесткости

case C2DBCAtom::symX:

{

m\_rp[k] += bc.getQx();

m\_rp[k + 1] = bc.getQy();

m\_matr.cell(k + 1, k + 1) = 1;

for (size\_t j = k + 2; j < m\_matr.band(); j++)

{

m\_rp[j] -= m\_matr.cell(k + 1, j) \* m\_rp[k + 1];

m\_matr.cell(k + 1, j) = 0;

}

for (size\_t i = 0; i < k + 1; i++)

{

m\_rp[i] -= m\_matr.cell(i, k + 1) \* m\_rp[k + 1];

m\_matr.cell(i, k + 1) = 0;

}

break;

}