ISSN: 2658–5782

Номер 3–4

2020

МНОГОФАЗНЫЕ СИСТЕМЫ

mfs.uimech.org

ИМех
им. Р.Р. Мавлютова

ISSN 2658–5782 Том 15 (2020), № 3–4, с. 183–200

Многофазные системы

http://mfs.uimech.org/mfs2020.3.129
DOI: 10.21662/mfs2020.3.129
УДК 539.371, 004.432.2, 004.942

Получена: 14.10.2020
Принята: 17.12.2020

Применение свободных программ FreeFem++/Gmsh и FreeCAD/CalculiX для моделирования статических задач упругости[1]

Насибуллаев И.Ш.

Институт механики им. Р.Р. Мавлютова УФИЦ РАН, Уфа

В работе рассматриваются этапы компьютерного численного моделирования инженерных задач и способы повышения точности моделирования; приводится краткий обзор свободных программ моделирования задач упругости методом конечных элементов, а также тенденции развития свободных систем автоматизированного проектирования и инженерного анализа. Для успешного проведения инженерного исследования необходимо подобрать удобный инструмент, учитывающий все особенности решаемой задачи. На основе решения тестовой статической задачи линейной упругости продемонстрировано два подхода к инженерному моделированию. Для первого подхода необходимы навыки программирования — полный цикл моделирования был написан на языке программирования пакета *FreeFem++*. Дополнительно показан способ создания расчетной сетки в программе *Gmsh* с последующим использованием в программе *FreeFem++*. Во втором подходе полный цикл моделирования проводится через интерфейс программы *FreeCAD* со встроенным решателем *CalculiX* и не требуется навыков программирования. Также предложен способ параметризации задачи с использованием встроенного в *FreeCAD* интерпретатора языка *Python*. Проведено сравнение результатов моделирования, полученных с использованием обоих подходов, для объекта, к которому приложено внешнее воздействие, определяемое граничными условиями Дирихле или Неймана, а также проанализированы два вида закрепления объекта: жесткая заделка и ограничение плоскостью с нулевым трением. Проведен анализ использования вычислительных ресурсов различными прямыми и итерационными методами. В рамках рассмотренной тестовой задачи статической линейной упругости наиболее оптимальным методом в *FreeFem++* является итерационный метод сопряженных градиентов *CG* как по времени вычислений, так и по используемому объему памяти. Наибольшую скорость вычислений дает итерационный метод *Cholesky* с обуславливанием неполным разложением Холецкого в программе *CalculiX*.

Ключевые слова: статическая задача упругости, свободное инженерное программное обеспечение, FreeFem++, Gmsh, FreeCAD, CalculiX, прямые и итерационные методы решения СЛАУ

1. Введение

В настоящее время существует множество программ компьютерного инженерного анализа задач упругости. Среди коммерческого ПО можно выделить *Ansys Mechanical* [1], *Simulia Abaqus* [2], *LS-DYNA* [3]. Данные программы позволяют провести полный цикл исследования: создание геометрии и расчетной сетки исследуемого объекта, моделирование с помощью различных методов и анализ полученных результатов. Использование этих программ в промышленности позволяет значительно сократить время и стоимость разработки конечного продукта за счет замены части натурных экспериментов компьютерным моделированием.

При проведении научных и инженерных исследований предпочтение отдается свободному ПО прежде всего из-за высокой стоимости коммерческих продуктов (бесплатные студенческие версии коммерческих продуктов имеют ограничение на количество узлов в модели и подходят

[1]Работа выполнена за счет средств государственного задания No 0246-2018-007.

больше для обучения, чем для исследования). Большое разнообразие свободных пакетов моделирования методом конечных элементов (МКЭ, FEM, finite element method) позволяет выбрать наиболее подходящий для целей исследования пакет. В настоящей работе приводится небольшой обзор свободного ПО для решения статических задач упругости и предлагается два подхода для проведения полного цикла исследования, а также показан пример использования различных пакетов в рамках одного исследования.

Компьютерное моделирование в пакетах разделяется на три этапа: подготовка модели, моделирование и обработка результатов. С каждым этапом связаны свои программы или программные модули:

1. *Препроцессор* (preprocessor) позволяет подготовить геометрию объекта, создать конечно-элементную (КЭ) сетку по геометрии используя, например, триангуляцию Делоне [4], задать модель материала, определить уравнения и граничные условия.

2. *Решатель* (solver) из системы дифференциальных уравнений в частных производных, записанных на узлах расчетной сетки, формирует систему линейных алгебраических уравнений (СЛАУ) и решает ее прямым или итерационным методом. Нелинейные уравнения решаются с помощью специальных численных алгоритмов. Например, в методе Ньютона [5] для нелинейной системы уравнений $F(\mathbf{u}) = 0$ сначала находится приближенное решение линеаризованной задачи $\mathbf{u}_0$, а затем итерационным методом решается СЛАУ $F(\mathbf{u} + \delta\mathbf{u}) = F(\mathbf{u}) + \delta\mathbf{u}J = 0$ и определяется поправка решения $\delta\mathbf{u}_{i-1} = -F(\mathbf{u}_{i-1})/J$, где J — якобиан $F(\mathbf{u})$; i — номер итерации. Решение $\mathbf{u}$ обновляется на каждом шаге по времени: $\mathbf{u}_i = \mathbf{u}_{i-1} + \delta\mathbf{u}_{i-1}$. То есть решение нелинейных задач также сводится к решению СЛАУ.

3. *Постпроцессор* (postprocessor) позволяет визуализировать результаты моделирования и провести обработку полученных данных.

В задачах с большим количеством степеней свободы необходимо подобрать оптимальный метод решении СЛАУ $Ax = b$ [6]. Прямые методы решения СЛАУ преобразуют исходную матрицу A к более простой форме (например, диагональной или треугольной), что позволяет получить точное решение x. Время, необходимое для решения системы, зависит от количества уравнений k, например, в методе Гаусса [7] производится k^3 вычислительных операций. Также для преобразования матрицы используется большой объем оперативной памяти. Таким образом, прямые методы подходят для решения задач с небольшим количеством степеней свободы (на современном персональном компьютере $k < 10^6$). Итерационные методы используют оперативную память более экономично, но скорость сходимости сильно зависит от числа обусловленности матрицы СЛАУ. Скорость сходимости можно повысить предобуславливанием матрицы [7]: умножая СЛАУ на матрицу P получим $PAx = Pb$. Новая система уравнений имеет то же решение, что и исходная, но сходимость итерационного решения изменится в зависимости от свойств матрицы P. Для матриц с различными особенностями (разреженная, симметричная, положительно определенная и т.д.) нужно выбирать методы, учитывающие эти особенности.

В МКЭ переменные определяются дискретным набором значений в узлах расчетной сетки, а значение в произвольной точке сетки определяется с помощью аппроксимации полиномами заданной степени. Для корректного моделирования необходимо подобрать тип КЭ и плотность узлов расчетной сетки и провести серию расчетов, пока искомые физические переменные не достигнут насыщения. Точность расчетов можно повысить двумя способами: увеличением количества узлов (уменьшением среднего расстояния между узлами) с фиксированным порядком аппроксимирующих полиномов (*h-метод*) или увеличением порядка аппроксимирующего полинома на фиксированных узлах сетки (*p-метод*) [8]. В *h-методе* точность результатов моделирования повышается за счет увеличения количества КЭ (с уменьшением их характерных размеров) без изменения типа. Для широкого спектра задач физики *p-метод* дает более быструю скорость сходимости (по крайней мере двукратную) по сравнению с *h-методом* [9]. Комбинация этих двух методов позволяет значительно увеличить скорость сходимости (в некоторых задачах экспоненциально) [10]. В современном подходе эффективность расчетной схемы повышается использованием КЭ высокого порядка с применением неравномерных адаптивных сеток, где плотность узлов пропорциональна градиентам переменных.

В последние годы в развитии свободных систем автоматизированного проектирования (*CAD*, computer-aided design) наметилась тенденция на включение в рамках одной программы дополнительных модулей, позволяющих сократить количество пакетов, необходимых для проведения полного цикла исследования. Приведем список программ, в которых к модулям пре/постпроцессоров были добавлены решатели различных физических

уравнений, включая задачи упругости (т.е. переход в сторону систем инженерного анализа, *CAE*, computer-aided engineering):

- *OneLAB* [11] объединяет генератор расчетных сеток и постпроцессор *Gmsh* [12] с решателем *GetDP* [13] (моделирование на сетках со смешанными элементами задач электромагнетизма, теплопереноса, акустики, упругости, решение уравнений в частных производных общего вида).

- *FreeCAD* [14] — программа параметрического трехмерного моделирования, которая позволяет проводить КЭ анализ структурных задач с помощью решателя *Z88* [15], а также гидродинамических задач и задач упругости с помощью решателей *CalculiX* [16] и *ElmerFEM* [17] и визуализировать результаты моделирования.

- *Salome Meca* — платформа для пре/постпроцессинга *Salome* [18] с интегрированным решателем термоупругих задач *Code Aster* [19].

- Генератор расчетных сеток *Netgen* содержит Python-интерфейс для решателей гидродинамических задач и задач упругости и электромагнетизма *Netgen/NGSolve* [20]

В данной работе, для демонстрации полного цикла исследования, приводится пример решения статической задачи линейной упругости с использованием двух подходов. В первом подходе для генерации расчетной сетки используется программа *Gmsh*, а само моделирование и анализ результатов проводятся с помощью программы, написанной на языке программирования *FreeFem++* [21]. Данный подход требует определенных навыков программирования.

FreeFem++ представляется собой подобный Си++ язык программирования для решения уравнений в частных производных, записанных в вариационной форме МКЭ на одно-, двух- и трехмерных расчетных сетках. *FreeFem++* имеет инструменты для препроцессинга (построение геометрии, генерация расчетной сетки, параметризация компьютерной модели); для решения систем алгебраических уравнений (прямые и итерационные методы LU, CG, Crout, Cholesky, GMRES, UMFPACK) и динамического перестроения и адаптации расчетных сеток; для постпроцессинга (визуализания расчетной сетки и КЭ переменных). Развитие компьютерного моделирования в *FreeFem++* осуществляется преимущественно за счет разработки новых алгоритмов и программного кода пользователями. В работе [22] была предложена модель течения жидкости через динамически изменяемое гидросопротивление. Полная трехмерная модель взаимодействия жидкости и упругой границы с анализом влияния граничных условий Дирихле и Неймана на деформацию границы представлена в работе [23]. Полное трехмерное моделирование регулирования течения жидкости деформацией упругой эластичной трубки пьезоэлементом показано в [24]. Интегрирование уравнений в *FreeFem++* реализовано для декартовой системы координат. В работе [25] предлагается подход для решения модели деформации гиперупругого материала в жидкости в осесимметричной системе координат, а в [26] представлены осесимметричные модели течения жидкости и деформации трубки в модели микронасоса, прокачивающего жидкость за счет несимметричной в осевом направлении динамической деформации трубки системой круговых пьезоэлементов. Особый интерес представляют работы [27] и [28], где предлагается расширение функциональности пакета для решения контактных задач упругости. В [29] предложен алгоритм «безопасной» перестройки расчетной сетки при динамическом изменении геометрии рассматриваемой системы, предотвращающий появление конечных элементов с отрицательными объемами.

Второй подход основан на использовании решателя *CalculiX*, интегрированного в пакет трехмерного моделирования *FreeCAD*, и не требует опыта в программировании. Для полноты описания программы приводятся примеры использования интерфейса программы и параметрической оптимизации построения модели с помощью Python-скрипта (оба примера являются полностью функционально эквивалентными и независимыми).

CalculiX представляет собой две программы — решатель CalculiX CrunchiX (cxx) [30], с помощью которого проводится моделирование командного файла формата *ABAQUS*, и пре/постпроцессора CalculiX GraphiX (cgx) [31], позволяющего проводить подготовку модели, визуализировать и анализировать результаты моделирования. Дистрибутив пакета содержит пример начала 90-х гг. трехмерного моделирования термоупругой и модальной (определение собственных резонансных частот) задачи ротора турбокомпрессора реактивного двигателя [32]. В работе [33] представлена параметрическая оптимизация трехмерной конструкции, состоящей из алюминиевой структуры в виде пчелиных сот, заключенной между двумя пластинами из стекловолокна, а также решается статическая задача упругости на определение деформаций в алюминиевой конструкции. Целью работы являлось определение оптимальной конструкции

с минимальным весом, при котором деформации не превышают заданных значений. В работе [34] представлена связанная динамическая задача по определению вибраций и напряжений на лопатках ротора (решалась в *CalculiX*), вызываемых аэродинамическим воздействием (которое рассчитывалось в стороннем ПО).

В настоящей работе в качестве пре/постпроцессора предлагается использовать *FreeCAD*, поскольку данная программа имеет удобный интерфейс и значительный инструментарий по построению геометрии, генерации расчетной сетки и анализу результатов моделирования, что позволяет значительно сократить время разработки модели.

Проводится сравнение различных прямых и итерационных методов решения СЛАУ:

- Прямые методы в *FreeFem++*: *LU* [6] (LU-decomposition; LU-разложение; несимметричная матрица), *Crout* [35] (Crout matrix decomposition; алгоритм Краута; симметричная матрица), *Cholesky* [7] (Cholesky decomposition; разложение Холецкого; симметричная и положительно определенная матрица), *UMFPACK* [36] (Unsymmetric MultiFrontal method; разреженная матрица);

- Итерационные методы в *FreeFem++*: *CG* [6] (conjugate gradient method; метод сопряженных градиентов; разреженная, симметричная и положительно определенная матрица), *GMRES* [37] (generalized minimal residual method; обобщенный метод минимальных невязок; разреженная матрица);

- Прямой метод в *CalculiX*: SPOOLES (SParse Object Oriented Linear Equations Solver; решатель для уравнений с разреженными матрицами [38]);

- Итерационные методы в *CalculiX*: внутренний итерационный метод, основанный на алгоритме, описанном в [39]; *ITERATIVE SCALING* использует предобуславливание масштабированием диагональных элементов матрицы СЛАУ; *ITERATIVE CHOLESKY* — предобуславливание с неполным разложением Холецкого.

2. Уравнения линейных деформаций в декартовых координатах

Геометрия тестовой задачи показана на рис. 1. Упругий цилиндр радиуса R и высотой H закреплен на нижнем основании Γ_d, а на поверхность Γ_u приложена нагрузка.

Упругие деформации твердого тела описываются уравнением движения (второй закон Ньютона), связывающим динамическое изменение вектора перемещений $\mathbf{u}$ под действием внутренних (описываемых тензором напряжений σ) и объемных сил $\mathbf{f}$ [40]:

$$\rho \frac{\partial^2 \mathbf{u}}{\partial t^2} = \nabla \sigma + \mathbf{f},$$

где ρ — плотность; t — время.

Статическое условие равновесия (после приложения силы прошло достаточно времени для установления поля деформации) в отсутствии объемных сил имеет вид:

$$\nabla \sigma = 0. \tag{1}$$

В изотропном теле тензор напряжений σ зависит от тензора деформаций ε и вектора перемещений $\mathbf{u}$ следующим образом:

$$\sigma = 2\mu\varepsilon + \lambda(\nabla \mathbf{u})\mathbf{I}, \tag{2}$$

где $\mathbf{I}$ — единичный тензор; параметры Ламе

$$\mu = \frac{E}{2(1+\nu)}, \quad \lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}$$

зависят от модуля Юнга E и коэффициента Пуассона ν.

Подставляя (2) в уравнение (1) и исключая тензор деформаций ε согласно определению

$$\varepsilon = \frac{1}{2}(\nabla \mathbf{u} + (\nabla \mathbf{u})^T),$$

получим уравнение равновесия, содержащее только вектор перемещений (уравнения Навье–Коши):

$$\nabla \left[\mu(\nabla \mathbf{u} + \nabla \mathbf{u}^T) + \lambda(\nabla \mathbf{u})\mathbf{I} \right] = 0. \tag{3}$$

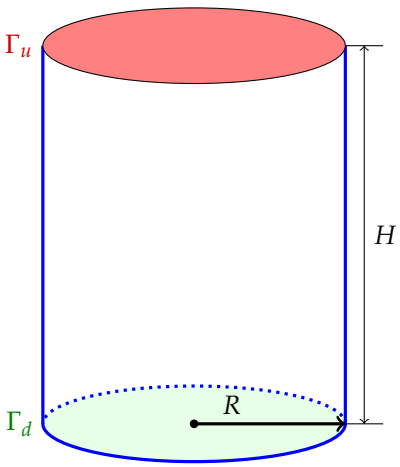

Рис. 1. Геометрия тестовой задачи

Граничные условия на верхней поверхности Γ_u могут быть заданы в виде условий Дирихле (перемещение на величину $\mathbf{u}_p$)

$$\vec{u}(\Gamma_u) = \mathbf{u}_p \tag{4}$$

или Неймана (на поверхность Γ_u приложено давление p и условием равновесия является баланс соответствующего напряжения σ_{ii})

$$\mu\sigma_{ii} = -p, \tag{5}$$

где знак «$-$» означает, что сила, создающая давление, направлена против нормали поверхности Γ_u.

Закрепление на нижней поверхности Γ_d можно задать двумя способами:
жесткая заделка

$$\vec{u}(\Gamma_d) = (0,0,0) \tag{6}$$

или ограничение плоскостью с нулевой силой трения

$$u_z(\Gamma_d) = 0. \tag{7}$$

Уравнение (3) записано в сильной (strong) формулировке. В МКЭ решаются уравнения в вариационной (variational, weak) формулировке. Умножим уравнение (1) на вектор пробных функций $\mathbf{v}$ и интегрируем по объему V. Согласно теореме Грина [41]:

$$\int\limits_V (\nabla\cdot\sigma)\mathbf{v}\,dV = \int\limits_\Gamma \frac{\partial\sigma}{\partial\mathbf{n}}\mathbf{v}\,d\Gamma - \int\limits_V \sigma\cdot\nabla\mathbf{v}\,dV. \tag{8}$$

Интеграл по поверхности Γ можно определить интегрированием граничных условий Неймана (5):

$$\int\limits_\Gamma \mu\sigma_{ii}\mathbf{v}\,d\Gamma = \int\limits_\Gamma \mu\frac{\partial\sigma}{\partial\mathbf{n}}\mathbf{v}\,d\Gamma = \int\limits_{\Gamma_u}(-p)\mathbf{v}\,d\Gamma. \tag{9}$$

Интегрирование уравнения (1) с учетом правила (8) и граничных условий Неймана (9) дает вариационную форму уравнения равновесия [21]:

$$\int\limits_V 2\mu\varepsilon(\mathbf{u}) : \varepsilon(\mathbf{v}) + \lambda(\nabla\mathbf{u})\cdot\nabla\mathbf{v})\,dV + \\ + \int\limits_{\Gamma_u}(-p)\mathbf{v}\,d\Gamma = 0, \tag{10}$$

где символ «:» означает тензорное скалярное произведение, т. е. $a : b = \sum_{i,j} a_{ij}b_{ij}$.

По известным значениям поля деформаций можно вычислить эквивалентные напряжения фон Мизеса:

$$\sigma_v^2 = \frac{1}{2}\Big[(\sigma_{xx}-\sigma_{yy})^2 + (\sigma_{yy}-\sigma_{zz})^2 + \\ (\sigma_{zz}-\sigma_{xx})^2 + 6(\sigma_{xy}^2+\sigma_{yz}^2+\sigma_{xz}^2)\Big], \tag{11}$$

где компоненты тензора напряжений имеют вид:

$$\sigma_{ii} = \lambda\left(\frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} + \frac{\partial u_z}{\partial z}\right) + 2\mu\frac{\partial u_i}{\partial x_i}, \\ \sigma_{ij} = \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right), \quad i \neq j. \tag{12}$$

3. Создание расчетной сетки в Gmsh

Gmsh — программа для параметрического построения геометрии объекта (в 1D, 2D или 3D) и генерации расчетной сетки с указанием необходимых свойств (плотность сетки, порядок и тип КЭ).

Для создания геометрии объекта предусмотрены два способа: с помощью интерфейса программы или с помощью тестового документа с командами в формате geo. Оба способа можно комбинировать (например создать геометрию в интерфейсе, затем отредактировать файл с заданием параметров). Рассмотрим второй способ, для того чтобы показать, как строится параметрическая геометрия.

В текстовом редакторе создадим файл cylinder.geo. Параметры определяются в виде переменных. Определим характерный размер расчетной сетки, радиус цилиндра и его высоту:

```
chl=5.0e-3;
r=2.5e-2;
h=5.0e-2;
```

Определим четыре точки основания цилиндра: центр; крайние правую, верхнюю, левую и нижнюю точки. Точки (линии, поверхности) задаются в стандартном порядке — против часовой стрелки, если создается замкнутый объект, и по часовой стрелке, если область объекта лежит вне замкнутого контура (например, если замкнутая область описывает полость в объекте). В качестве параметров указываются координаты x, y, z и характерный размер сетки:

```
Point(1) = {0,0,0,chl};
Point(2) = {r,0,0,chl};
Point(3) = {0,r,0,chl};
Point(4) = {-r,0,0,chl};
Point(5) = {0,-r,0,chl};
```

В *Gmsh* окружность состоит из нескольких дуг, каждая из которых имеет угловой размер, не превышающий $\pi/2$. Составим окружность из 4 дуг размером $\pi/2$ с тремя параметрами-точками — начало дуги, центр окружности, конец дуги:

```
Circle(1) = {2,1,3};
Circle(2) = {3,1,4};
Circle(3) = {4,1,5};
Circle(4) = {5,1,2};
```

Собираем части в окружность и составляем поверхность

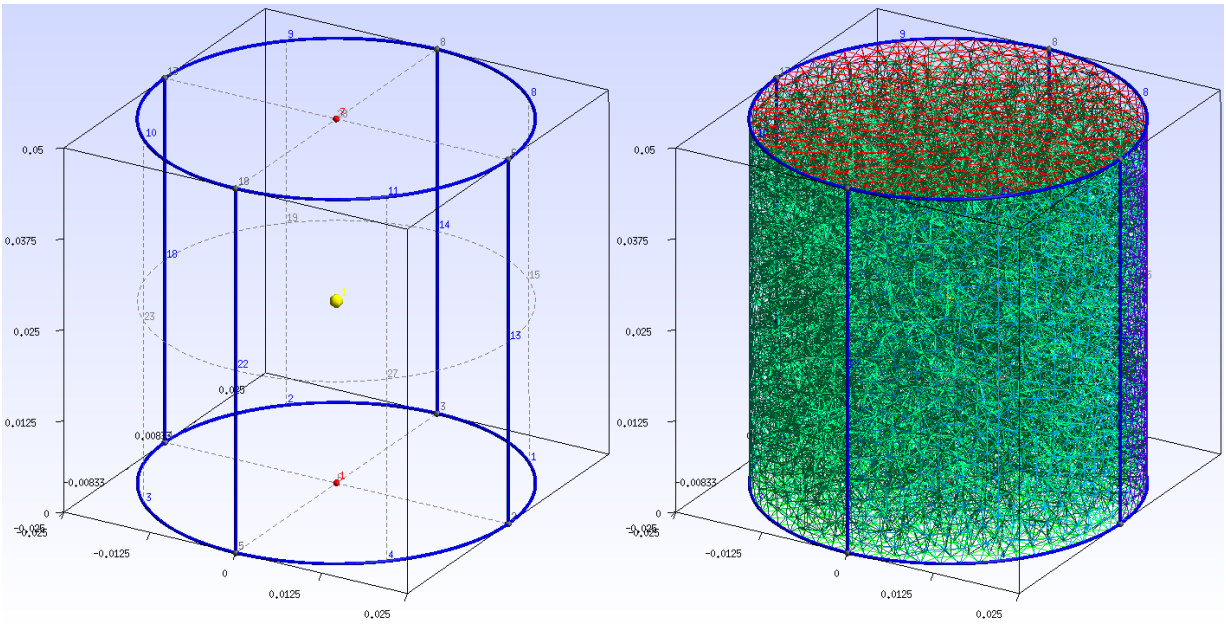

Рис. 2. Геометрия цилиндра и его расчетная сетка, построенные в *Gmsh*

```
Line Loop(5) = {1,2,3,4};
Plane Surface(6) = {5};
```

Для создания цилиндра вытягиваем круглое основание вдоль оси z на величину высоты цилиндра h:

```
cyl[] = Extrude {0,0,h} {Surface{6};};
```

Для того чтобы в программе моделирования задавать граничные условия, необходимо установить метки для всех поверхностей и объема объекта. Для этого сохраним текстовый документ, откроем его в *Gmsh*, в настройках геометрии зададим отображение номеров для поверхностей и объемов. Ориентируясь по этим номерам, создадим физические группы для основания цилиндра, его верхней и боковой поверхностей и всего объема. При создании расчетной сетки все узлы, сегменты и поверхность будут иметь метку физической группы, к которой они принадлежат. Добавляем определение физических групп в файл и перезагружаем его в *Gmsh*.

```
Physical Surface("cyldown") = {6};
Physical Surface("cylup") = {28};
Physical Surface("cylside") = {15,19,23,27};
Physical Volume("cylvol") = {cyl[]};
```

Если в системе установлена библиотека *OpenCASCADE*, то для файла геометрии достаточно использовать библиотечную функцию построения цилиндра и задать граничные поверхности и само тело следующим образом:

```
SetFactory("OpenCASCADE");
Cylinder(1) = {0,0,0,0,0,0.05,0.025,2*Pi};
Physical Volume("1", 1) = {1};
Physical Surface("1", 1) = {1};
Physical Surface("2", 2) = {2};
Physical Surface("3", 3) = {3};
```

На рис. 2 слева показана полученная геометрия с визуализацией всех меток для узлов, сегментов и поверхностей.

Для более точной настройки сетки указываем минимальный и максимальный размеры элементов (Options/Mesh/General/Min/Max element size) $\min = 1.e-5, \max = 2.65e-5$ и их порядок (первый или второй). Для создания трехмерной сетки во вкладке Mesh в списке команд выбираем элемент 3D и сохраняем сетку в формате msh (без записи дополнительной информации, только данные, связанные с физическими группами). На рис. 2 справа показана построенная расчетная сетка.

4. Моделирование в FreeFem++

Приведем пример построения трехмерной модели статического сжатия цилиндра в *FreeFem++* с использованием линейной упругой модели. Программа написанная на языке *FreeFem++* представляет собой текстовый файл с расширением edp.

FreeFem++ может работать с внешней расчетной сеткой в формате *Gmsh*. Подключаем библиотеку для работы с трехмерными расчетными сетками, библиотеку для работы с форматом *Gmsh* и загружаем ранее созданный файл с сеткой:

```
load "msh3"
load "Gmsh"
mesh3 ThE = Gmshload3("cylinder.msh");
```

Расчетную сетку для простой двумерной или трехмерной геометрии можно сделать средствами *FreeFem++*. Принцип построения геометрии следующий: задаются границы в параметрическом виде $x = x(t)$, $y = y(t)$, $z = z(t)$ с указанием метки границы (label); границы, составляющие замкнутый контур, образуют поверхность, а поверхности можно собрать в объемный объект. Задаем параметры геометрии (масштаб, радиус и высоту цилиндра) и модели (перемещение для граничных условий Дирихле и давление для граничных условий Неймана):

```
load "msh3"
real scale = 1.e-3;
real radius = 25*scale;
real height = 50*scale;
real displacementZ = 0.5*scale;
real pressureZ = 1.0e4;
```

Создаем окружность радиуса R с помощью параметрического уравнения окружности:

```
border bC(t=0,2*pi){
 x=radius*sin(t); y=radius*cos(t); label=0;};
```

Плотность расчетной сетки определяется количеством узлов на границе. Зададим количество узлов вдоль окружности цилиндра и вдоль его высоты:

```
int NBEC = 50, NBEZ = 20;
```

Строим двумерную сетку в границах окружности с указанием количества узлов на границе. Поскольку параметрическое уравнение окружности задает границу в направлении по часовой стрелки, количество узлов указываем со знаком «−», чтобы инвертировать направление:

```
mesh mC = buildmesh(bC(-NBEC));
```

Задаем номера (метки) для верхней, боковой и нижней поверхностей цилиндра (по этим меткам далее будут заданы граничные условия):

```
int[int] refTop = [0, 2], refBottom = [0, 3],
         refSide = [0, 1];
```

Вытягиваем двумерную сетку на величину H для создания трехмерной сетки цилиндра с метками на каждой поверхности:

```
mesh3 ThE = buildlayers(mC, NBEZ,
 zbound=[0, height], labelmid=refSide,
 labelup = refTop, labeldown = refBottom);
```

Выбирается один из двух приведенных выше способов задания сетки. Далее посмотрим, как выглядит сетка (будут показаны только граничные элементы) и выведем в консоль информацию по параметрам сетки (количество узлов, граничных/треугольных элементов и объемных элементов/тетраэдров). Информация по сетке позволяет провести проверку импортированной сетки и значения должны совпадать с информацией в *Gmsh* (Tools/Statistics):

```
plot(ThE);
cout << "Vertex, Border, Tetrahedra: "
  << ThE.nv << ", " << ThE.nbe << ", "
  << ThE.nt << endl;
```

Сечение расчетной сетки плоскостью $z = \text{const}$ представляет собой многоугольник, вписанный в окружность, что дает погрешность в величине объема V_m цилиндра. Оценить эту погрешность γ_V относительно реального объема V_r можно следующим образом:

$$V_r = \pi R^2 H,\ V_m = \int\limits_V dV,\ \gamma_V = |1 - V_m / V_r|. \quad (13)$$

Соответствующий код:

```
real vr = pi*radius^2*height;
real vm = int3d(ThE)(1.0);
cout << "Volume: real = " << vr << ", mesh = "
  << vm << ", err = " << 100.0*abs(1.0-vm/vr)
  << "%" << endl;
```

Аналогично можно оценить изменение объема сетки после деформации, например, для проверки условия несжимаемости материала.

Зададим свойства материала (модуль Юнга и коэффициент Пуассона) и рассчитаем коэффициенты Ламе:

```
real E = 1.0E6, sigma = 0.49;
real mu = E/(2*(1+sigma)),
     lambda = E*sigma/((1+sigma)*(1-2*sigma));
```

Точность решения зависит от порядка полиномов аппроксимирующих конечно-элементные переменные. Можно задать элементы первого

```
fespace Vh(ThE,[P1,P1,P1]);
```

или второго порядка

```
fespace Vh(ThE,[P2,P2,P2]);
```

Описываем компоненты вектора деформаций и соответствующих пробных функций:

```
Vh [d1,d2,d3] = [0,0,0], [vv1,vv2,vv3];
```

Определим макросы для тензора деформаций и дивергенции (макрос обязательно должен заканчиваться символами комментария //):

```
real sqrt2=sqrt(2.0);
macro epsilon(u1,u2,u3)
  [dx(u1),dy(u2),dz(u3),
  (dz(u2)+dy(u3))/sqrt2,
  (dz(u1)+dx(u3))/sqrt2,
  (dy(u1)+dx(u2))/sqrt2]
  // EOM
macro div(u1,u2,u3)
  ( dx(u1)+dy(u2)+dz(u3) )
  // EOM
```

Перейдем к решению уравнения (10) с помощью команды *solve* выбранным методом (*solver=[LU, CG, Crout, Cholesky, GMRES, UMFPACK]*); для итерационных методов указывается абсолютная погрешность *eps*. Для трехмерных интегралов используется команда *int3d* (объемные интегралы в (10)) с двумя аргументами: расчетная сетка и подинтегральное выражение. Символ одинарной кавычки означает операцию транспонирования. Код для решения задачи линейной упругости (10) имеет вид [21]:

```
solve Lame([d1,d2,d3],[vv1,vv2,vv3],
  solver=CG,eps=1.e-8)=
  int3d(ThE)(
    lambda*div(d1,d2,d3)*div(vv1,vv2,vv3)
    +2.0*mu*( epsilon(d1,d2,d3)'*
       epsilon(vv1,vv2,vv3) ))
```

Зададим граничные условия на нижней поверхности Γ_d в виде (6)

```
        + on(3,d1=0,d2=0,d3=0)
```

или в виде (7)

```
        + on(3,d3=0)
```

и граничные условия на верхней поверхности Γ_u в виде условий Дирихле (4)

```
        + on(2,d3=-displacementZ);
```

или Неймана (5) (команда *int2d* используется для задания поверхностного интеграла в (10)):

```
        +int2d(ThE,2)(pressureZ*vv3);
```

Если выбран метод *UMFPACK*, то для решения задачи, использующей более 2 ГБ памяти, необходимо подключить 64-разрядную версию:

```
load "UMFPACK64"
```

Поскольку деформации малы, для визуализации введем коэффициент для масштабирования деформаций и создадим новую, деформированную сетку:

```
real coef = 20.0;
mesh3 ThED = movemesh3(ThE, transfo=[
 x+coef*d1(x,y,z), y+coef*d2(x,y,z),
 z+coef*d3(x,y,z)]);
```

Выводим в консоль минимальное и максимальное значения компонент вектора деформаций и компоненты вектора деформаций на верхней поверхности:

```
cout << "Disp[mm]: min = " << d1[].min/scale
  << ", max = "  << d1[].max/scale << ", d1 = "
  << d1(radius, 0, height)/scale << ", d2 = "
  << d2(0, radius, height)/scale << ", d3 = "
  << d3(0, 0, height)/scale << endl;
```

Визуализация недеформированной и деформированной сеток:

```
plot(ThE, ThED);
```

Для трехмерной визуализации значений переменных в *FreeFem++* есть два способа. Первый — построение сетки сечения трехмерного объекта и вычисление на ней двумерных КЭ-переменных по значению трехмерной переменной. Покажем этот способ для сечения в плоскости $y = 0$ деформированного цилиндра:

```
mesh Thsec=square(NBEC,NBEZ,
  [(2.0*x-1.0)*radius,y*height]);
mesh ThsecD = movemesh(Thsec,
  [x+coef*d1(x,0,y),y+coef*d3(x,0,y)]);
fespace Xh(ThsecD,P2);
Xh secd3;
secd3=d3(x-coef*d1(x,0,y),0,
  y-coef*d3(x,0,y))/scale;
plot(Thsec,ThsecD);
real[int] viso(26);
for (int i = viso.n-1; i >= 0; i--)
  viso[viso.n-1-i] =
    i*d3(0,0,height)/scale/(viso.n-1);
plot(Thsec, ThsecD, secd3, viso=viso,
  nbiso=viso.n, fill=1, value=1);
```

Команда визуализации *plot* может сохранять изображение в виде графического файла формата *eps* опцией *ps* (работает только для двумерных объектов), также можно указать область визуализации опцией *bb*:

```
func bb = [[-1.1*radius,-0.1*height],
  [1.5*radius,1.1*height]];
plot(Thsec, ThsecD, secd3, viso=viso,
  nbiso=viso.n, fill=1, value=1,
  bb=bb, ps="f.eps");
```

Второй способ — визуализация трехмерной сетки с определенной на ней переменной с помощью подключаемого модуля *medit*:

```
load "medit"
medit("dz", ThED, d3/scale);
```

Команда *medit* открывает графическое окно, в котором можно настроить параметры визуализации сетки и значения переменной. Пока окно

открыто, выполнение кода *FreeFem++* приостанавливается, поэтому, для отображения промежуточных результатов моделирования, рекомендуется использовать команду *plot*.

Можно рассчитать компоненты тензора напряжений (12) и эквивалентные напряжения фон Мизеса (11):

```
fespace Eh(Thsec,P2);
Eh sxx = lambda*(dx(d1)+dy(d2)+dz(d3))+
  2*mu*dx(d1);
Eh syy = lambda*(dx(d1)+dy(d2)+dz(d3))+
  2*mu*dy(d2);
Eh szz = lambda*(dx(d1)+dy(d2)+dz(d3))+
  2*mu*dz(d3);
Eh sxy = mu*(dy(d1)+dx(d2));
Eh syz = mu*(dz(d2)+dy(d3));
Eh sxz = mu*(dz(d1)+dx(d3));
Eh sv = sqrt( (sxx-syy)^2+(syy-szz)^2+
  (szz-sxx)^2+6*(sxy^2+syz^2+sxz^2) )/sqrt2;
```

Во время разработки компьютерной модели удобно использовать встроенные функции *FreeFem++* для профилирования кода. Задаем две переменные

```
real cputime1=0, cputime2=0;
```

Время выполнения фрагментов кода (в секундах) определяется следующей конструкцией:

```
cputime1 = clock();
// here code for profiling
cputime2 = clock();
cout << "Time = " << cputime2-cputime1 << endl;
```

Код программы запускается в консоли (ОС Linux) или в командной оболочке (ОС Windows):

```
FreeFem++ cylinder.edp
```

Результаты моделирования для граничных условий (6) и (7) показаны на рис. 3.

5. Моделирование в FreeCAD/CalculiX

При определении геометрических и физических единиц нужно учитывать, что в *FreeCAD* и *CalculiX* используется инженерная система единиц по умолчанию: длина [мм], масса [кг], время [с]. Поэтому в интерфейсе, где размерность не указывается в явном виде, необходимо учитывать систему единиц по умолчанию.

Рассмотрим порядок подготовки модели в *FreeCAD*. Для удобства все инструменты разделены на модули: PartDesign (геометрия объекта), FEM (моделирование методом конечных элементов).

После выбора модуля Part Design создаем цилиндр (создать аддитивный примитив→аддитивный цилиндр) с радиусом $R = 25$ mm и высотой $H = 50$ mm (в интерфейсе программы размерность указывается латинскими символами).

В модуле FEM сначала создаем новый анализ (model→analysis), выделяем цилиндр и добавляем для него свойства материала (model→material→Material for Solid, достаточно задать модуль Юнга 1 MPa и коэффициент Пуассона 0.49). Задаем граничные условия: фиксированные на нижней грани (model→mechanical constraints→constraint fixed) и условия Дирихле (constraint displacement) или Неймана (constant pressure) на верхней. Создаем расчетную сетку для объекта (mesh→create FEM mesh from a shape by Gmsh) с указанием порядка КЭ (первый или второй) и плотности сетки (максимальный/минимальный размер КЭ min = 1 mm, max = 2.65 mm).

Для запуска решателя (CalculiXccxTools) задаем рабочую папку, сохраняем *.inp файл и запускаем расчет. Результаты расчета можно посмотреть в *CalculiX_static_results* (z Dispacement, scale=20).

Отметим, что в процессе моделирования в рабочей папке создаются два файла: командный файл модели в формате Abaqus “FEMMeshGmsh.inp” и файл результатов моделирования в формате Abaqus “FEMMeshGmsh.frd”. Первый может быть использован для запуска модели в *CalculiX/cxx* в командной строке

```
ccx FEMMeshGmsh
```

а второй — для анализа результатов в постпроцессоре *CalculiX/cgx*

```
cgx FEMMeshGmsh.frd
```

т. е. *FreeCAD* может использоваться в качестве препроцессора, а сам анализ модели можно проводить непосредственно в *CalculiX*. По умолчанию *FreeCAD* в командном файле указывает метод решения СЛАУ *SPOOLES*. Чтобы выбрать итерационный метод нужно в командном файле заменить строку (с помощью любого текстового редактора)

```
*STATIC
```

на строку

```
*STATIC, SOLVER=ITERATIVE SCALING
```

или

```
*STATIC, SOLVER=ITERATIVE CHOLESKY
```

В варианте *SCALING* используется предобуславливание с масштабированием диагональных элементов матрицы СЛАУ, а в *CHOLESKY* — неполное разложение Холецкого.

Деформация цилиндра, полученная в *FreeCAD/CalculiX*, с граничными условиями (6) и (7) показана на рис. 4.

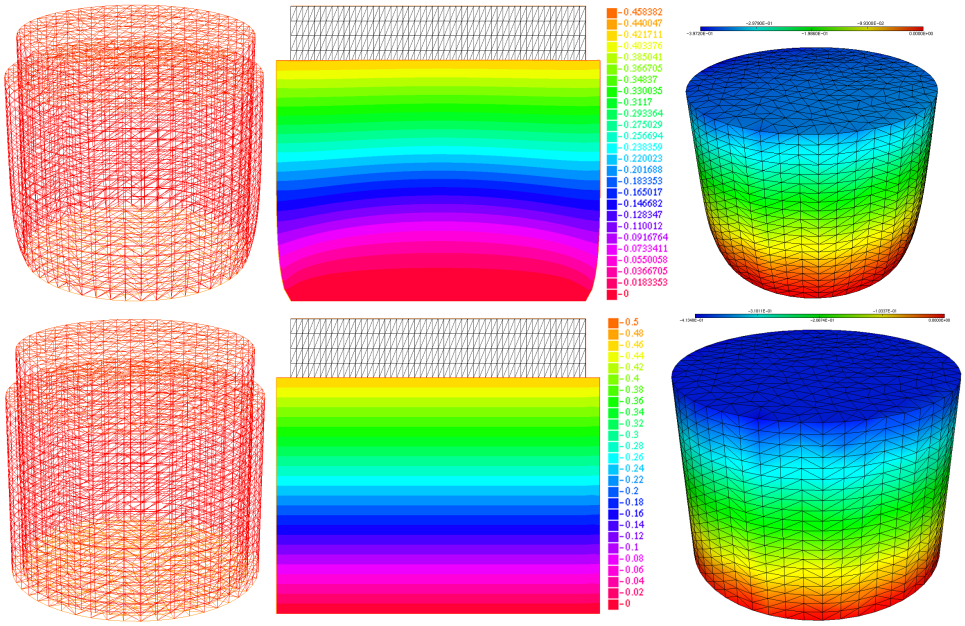

Рис. 3. Деформация цилиндра (слева), сечение $y = 0$ (по центру) и визуализация в *medit* (справа), рассчитанные в *FreeFem++* с граничными условиями (6) (вверху) и (7) (внизу). Деформация увеличена в 20 раз

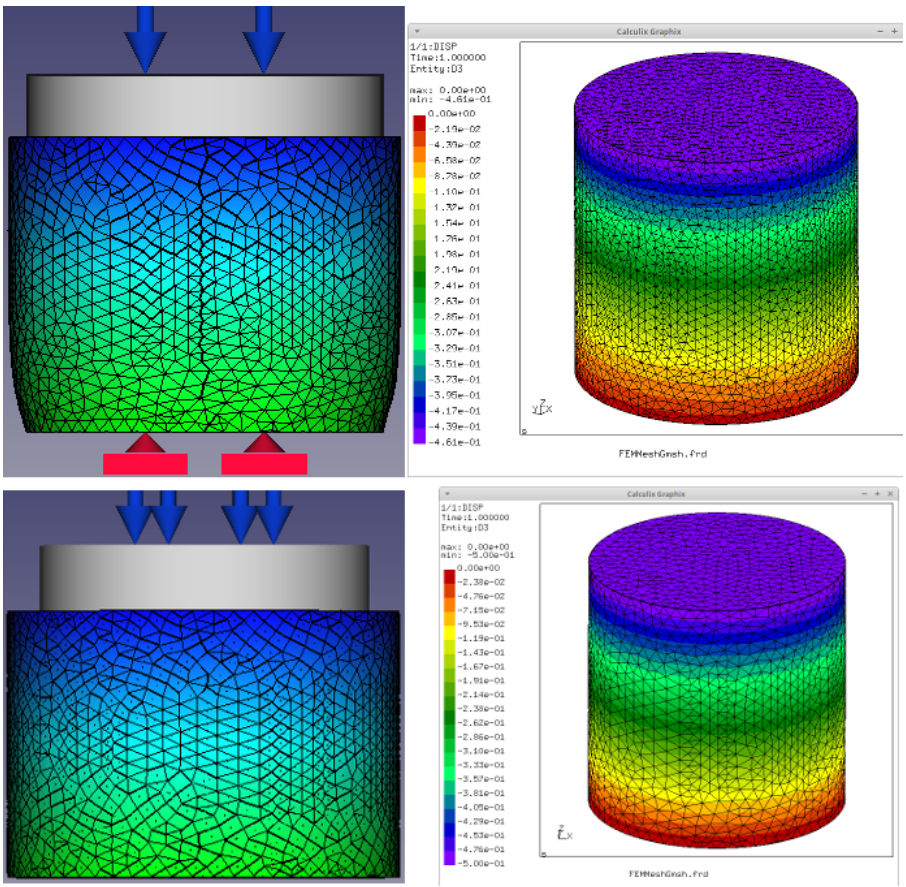

Рис. 4. Деформация цилиндра, полученная в *FreeCAD/CalculiX*, с граничными условиями (6) (сверху) и (7) (снизу). Визуализация результатов в *FreeCAD* (слева) и *CalculiX-cgx* (справа). Деформация в *FreeCAD* увеличена в 20 раз

Полученную модель можно редактировать в самой среде *FreeCAD*. В некоторых случаях, например, если необходимо провести большую серию расчетов с изменением геометрических и/или физических параметров модели, то более эффективным будет использование встроенного интерпретатора языка *Python*. Скрипты можно запускать через интегрированную в интерфейс *Python*-консоль. Продемонстрируем пример параметрического скрипта для рассматриваемой модели (пример основан на скрипте без параметризации [42]).

Создаем новый проект и определяем все параметры модели (радиус и высоту цилиндра; модуль Юнга и коэффициент Пуассона материала; плотность расчетной сетки; перемещение и давление на верхней поверхности цилиндра; рабочую папку):

```
doc = App.newDocument("Cyl")
r = 25
h = 50
ym = "1.0 MPa"
pr = "0.49"
mms = 2.65
zd = -0.5
cp = 0.01
wdir = "~/calculix"
```

Создаем цилиндр заданного радиуса и высоты и отображаем его в интерфейсе:

```
import Part
cylObj = doc.addObject('Part::Cylinder',\
'Cylinder')
cylObj.Radius = r
cylObj.Height = h
import FreeCADGui
FreeCADGui.ActiveDocument.activeView().\
viewAxonometric()
FreeCADGui.SendMsgToActiveView("ViewFit")
```

Задаем тип анализа модели:

```
import ObjectsFem
analysis = ObjectsFem.makeAnalysis(doc,\
"Analysis")
solver = ObjectsFem.\
makeSolverCalculiXCcxTools(doc, "CalculiX")
solver.GeometricalNonlinearity = 'linear'
solver.ThermoMechSteadyState = True
solver.MatrixSolverType = 'default'
solver.IterationsControlParameterTimeUse=False
analysis.addObject(solver)
```

Добавляем материал с заданными параметрами:

```
matObj=ObjectsFem.makeMaterialSolid(doc,\
"SolidMaterial")
mat = matObj.Material
mat['Name'] = "Silicone"
mat['YoungsModulus'] = ym
mat['PoissonRatio'] = pr
matObj.Material = mat
analysis.addObject(matObj)
```

Зафиксируем основание цилиндра (6):

```
fixed=ObjectsFem.makeConstraintFixed(doc,\
"FemConstraintFixed")
fixed.References = [(doc.Cylinder, "Face3")]
analysis.addObject(fixed)
```

Задаем на верхней грани цилиндра граничные условия Дирихле (4):

```
disp = ObjectsFem.makeConstraintPressure(doc,\
"FemConstraintDisplacement")
disp.References = [(doc.Cylinder,"Face2")]
disp.zDisplacement = zd
analysis.addObject(disp)
```

или Неймана (5):

```
pressure = ObjectsFem.makeConstraintPressure(\
doc, "FemConstraintPressure")
pressure.References = [(doc.Cylinder, "Face2")]
pressure.Pressure = cp
analysis.addObject(pressure)
```

Создаем расчетную сетку с элементами второго порядка генератором *NetGen*:

```
mesh = doc.addObject(\
'Fem::FemMeshShapeNetgenObject',\
'FEMMeshNetgen')
mesh.Shape = doc.Cylinder
mesh.MaxSize = mms
mesh.Fineness = "Moderate"
mesh.Optimize = True
mesh.SecondOrder = True
```

или генератором *Gmsh*:

```
femObj=ObjectsFem.makeMeshGmsh(doc,\
cylObj.Name + "_Mesh")
femObj.Part = doc.Cylinder
femObj.ElementOrder = u"2nd"
femObj.CharacteristicLengthMax = '2.65 mm'
femObj.CharacteristicLengthMin = '1 mm'
doc.recompute()
from femmesh.Gmshtools import GmshTools as gt
mesh = gt(femObj)
```

Добавляем созданную расчетную сетку в анализ, настраиваем решатель и запускаем расчет деформаций (в указанной рабочей папке создается файл модели и файл с результатами моделирования):

```
analysis.addObject(mesh)
from femtools import ccxtools
fea = ccxtools.FemToolsCcx()
fea.update_objects()
fea.setup_working_dir(wdir)
fea.setup_ccx()
fea.purge_results()
fea.write_inp_file()
fea.ccx_run()
```

Загружаем результаты из рабочей папки и показываем недеформированную и деформированную (с 20-кратным масштабированием деформаций) сетки:

```
fea.load_results()
for m in analysis.Group:
  if m.isDerivedFrom('Fem::FemResultObject'):
    result = m
    break
mesh.ViewObject.setNodeDisplacementByVectors(\
result.NodeNumbers, result.DisplacementVectors)
mesh.ViewObject.applyDisplacement(20)
```

Отметим основное отличие программирования в *FreeFem++* от *FreeCAD* — в *FreeFem++* программируется полная задача с геометрией, параметрами модели, заданием уравнений в явном виде и их решением, обработкой и сохранением полученных результатов; в *FreeCAD* скрипты автоматизируют настройку модели через интерфейс программы (значительно сокращая время при модификации модели за счет возможности параметризации задачи).

6. Анализ результатов

Проведем анализ тестовой задачи со сплошным цилиндром радиуса R и высотой H и с ограничением перемещения у основания вдоль оси Oz (7). На верхнюю плоскость цилиндра приложено давление p. Под действием давления цилиндр сжимается в осевом направлении на величину ΔH_a, которую можно оценить из определения для модуля Юнга E:

$$E = \frac{p}{\Delta H_a / H} \Rightarrow \Delta H_a = \frac{pH}{E}. \qquad (14)$$

Для значений $R = 2.5$ см, $H = 5$ см, $E = 1$ МПа, $p = 10$ кПа получим $\Delta H_a = -0.5$ мм. ΔH_a зависит от приложенного давления p линейно.

Во время деформации цилиндр растягивается в радиальном направлении и его радиус увеличивается на величину ΔR_a, которую можно вычислить из определения для коэффициента Пуассона:

$$\nu = -\frac{\Delta R_a}{R}\frac{H}{\Delta H_a} \Rightarrow \Delta R_a = -\nu\frac{\Delta H_a}{H}R. \qquad (15)$$

Для $H = 5$ см, $R = 2.5$ см, величины ΔH_a, вычисленной по формуле (14) и значения коэффициента Пуассона $\nu = 0.49$ получим $\Delta R = 0.1225$ мм.

При численном моделировании в программе *FreeFem++* с граничными условиями (7), для расчетной сетки, состоящей из 25920 объемных элементов второго порядка, погрешность величины объема цилиндра относительно его точного значения, определяемого по формуле (13), составила $\gamma_V = 0.29\%$, а погрешности величин осевого сжатия ΔH и радиального растяжения ΔR относительно аналитических решений (14) и (15) составили $\gamma_H = 3 \cdot 10^{-7}\%$ и $\gamma_R = 0.47\%$. С увеличением числа КЭ погрешности снижаются: например, при восьмикратном увеличении количества КЭ погрешности составили $\gamma_V = 0.066\%$, $\gamma_H = 3 \cdot 10^{-7}\%$ (погрешность округления) и $\gamma_R = 0.053\%$, что является подтверждением корректности построенной численной модели.

Рассмотрим жесткое закрепление на основании: $\mathbf{u} = 0$ (6) — сила трения скольжения достаточна для предотвращения скольжения основания цилиндра по поверхности. В этом случае площадь основания останется неизменной, а увеличение площади верхней поверхности цилиндра будет пропорциональна приложенному давлению. Следовательно, величина вертикального сжатия ΔH_f будет меньше ΔH_a. Поскольку уравнения (3) линейные, H_f зависит от величины давления p линейно.

В табл. 1 приведены следующие параметры моделирования и результаты: тип конечных элементов (КЭ) и их порядок (П), количество узлов m_v и объемных элементов (тетраэдров) m_t расчетной сетки, количество решаемых уравнений в СЛАУ k, метод решения СЛАУ (прямые методы обозначены прямым шрифтом, а итерационные — наклонным), величина вертикальной деформации ΔH на верхнем ребре цилиндра и ее относительная погрешность γ, процессорное время генерации расчетной сетки T_m и время расчета задачи упругости T_s, занимаемый объем оперативной памяти и файла подкачки (если он использовался), номер расчета n. Все расчеты проводились на одном ядре процессора. Затраченное на выполнение расчета время определялось в *FreeFem++* с помощью профилирования кода (команда *clock*), а в *CalculiX* — консольной командой *time*. Используемый объем ОЗУ и файла подкачки определялся с помощью программы *gnome-system-monitor*.

Первые 17 расчетов проводились в ОС CAELinux-2020 на процессоре Core 2 Duo-E8500 (два ядра по 3.16 ГГц, кэш 6 МБ) с 4 ГБ оперативной памяти и файлом подкачки, размещенным на твердотельном жестком диске, в *FreeFem++* ($n = 1 - 10$) и *CalculiX* ($n = 11 - 17$). Выбор параметров генерации сетки произведен таким образом, чтобы сетки были близки по количеству объемных элементов и размеру СЛАУ (15165 уравнений для КЭ первого порядка и 112545 уравнений — для второго). Остальные расчеты ($n = [18 - 33]$) проводились в ОС Ubuntu 14.04 на процессоре Intel Core i7-3770K (4 ядра и 8 потоков по 3.5 ГГц, кэш 8 МБ) с 16 ГБ ОЗУ и файлом подкачки, размещенном на жестком диске SATA-6Gb. Максимальная абсолютная погрешность для итерационных методов в *FreeFem++* задавалась равной $2 \cdot 10^{-7}$. В *CalculiX* максимальная абсолютная погрешность

Таблица 1. Результаты моделирования тестовой статической задачи линейной упругости с использованием различных методов решения СЛАУ в программах *FreeFem++* и *CalculiX*.

Программа	КЭ, П	m_v, m_t, k	Метод	ΔH, мм	γ, %	T_m, с	T_s, с	ОЗУ+ФП	n
FreeFem++	P1, 1	5082, 25920, 15246	LU	−0.439048	4.9	0.019	**24.6**	**237.4** МБ	1
			Crout				11.9	121.1 МБ	2
			Cholesky				8.7	121.1 МБ	3
			UMFPACK				5.64	144.9 МБ	4
			CG				**3.0**	**8.7** МБ	5
			GMRES				11.2	18 МБ	6
	P2, 2	5082, 25920, 112545	UMFPACK	−0.458382	0.71		**330**	**3.5 + 4** ГБ	7
			CG				**86.1**	**65.4** МБ	8
			GMRES				477	161.3 МБ	9
	P1, 1	38663, 214080, 115989	*CG*	−0.451112	2.3	0.15	46	55 МБ	10
CalculiX	C3D4, 1	5458, 25876, 15967	SPOOLES	−0.443498	3.9	2.5	**3.9**	**68.5** МБ	11
			Scaling	−0.443501	3.9		1.5	**13.25** МБ	12
			Cholesky	−0.443974	3.8		**1.3**	15.5 МБ	13
	C3D10, 2	39187, 25855, 116071	SPOOLES	−0.460443	0.26	4.4	**66.8**	**990** МБ	14
			Scaling	−0.460444	0.26		50.1	**92, 9** МБ	15
			Cholesky	−0.460735	0.2		**26.7**	127.5 МБ	16
	C3D4, 1	37871, 204843, 108861	*Cholesky*	−0.454901	1.5	8.8	15	115.7 МБ	17
FreeFem++	P2, 2	5082, 25920, 112545	UMFPACK	−0.458382	0.7	0.01	77.7	7.5 ГБ	18
			GMRES				271	161 МБ	19
			CG				46.9	65 МБ	20
		38663, 214080, 891567	*CG*	−0.461349	0.062	0.085	782	516 МБ	21
		124745, 708840, 2913801		−0.461561	0.017	0.3	3990	1.6 ГБ	22
		294111, 1694400, 6917043		−0.461638	**0**	0.6	13173	3.9 ГБ	23
CalculiX	C3D10, 2	39187, 25855, 226581	SPOOLES	−0.460443	0.26	4.4	47.4	990 МБ	24
			Scaling	−0.460444	0.26		30	92, 9 МБ	25
			Cholesky	−0.460700	0.2		15.5	127.5 МБ	26
		77960, 52887, 226581	SPOOLES	−0.460738	0.19	8	186	2.5 ГБ	27
			Cholesky	−0.460769	0.19		52	258.5 МБ	28
		146068, 101434, 426972	SPOOLES	−0.460998	0.14	14.4	644	5.8 ГБ	29
			Cholesky	−0.460977	0.14		112	495.5 МБ	30
		290182, 204844, 1851447	SPOOLES	−0.461233	0.088	30.3	2485	14 + 1 ГБ	31
			Cholesky	−0.461517	0.026		310	971 МБ	32
		627812, 448982, 4427019	*Cholesky*	−0.461687	0.011	66.4	764	2.1 ГБ	33

γ_m определяется самой программой и составляет $2 \cdot 10^{-7}$ для метода *Scaling* и $10^{-6}, \ldots, 5 \cdot 10^{-6}$ для метода *Cholesky* (для большего k меньшее γ_m). Расчеты $n = [21 - 23]$ и $n = [27 - 33]$ проведены для определения насыщения величины максимальной деформации ΔH при увеличении m_v. Относительная погрешность γ определялась относительно расчета $n = 23$ на сетке с $m_v = 294111$ узлами. Относительная погрешность величины объема цилиндра определялась по формуле (13) и для расчетов $n = [20 - 23]$ составила $\gamma_V = [0.26\%, 0.066\%, 0.029\%, 0.016\%]$. Прямые методы *LU*, *Crout* и *Cholesky* в *FreeFem++* удалось запустить только для КЭ первого порядка. Ускорение расчетов за счет более производительного процессора составило $\sim 70\%$.

Итерационные методы решения СЛАУ для рассматриваемой задачи продемонстрировали более

высокую скорость вычислений по сравнению с прямыми методами. Это связано с тем, что решалась трехмерная задача и даже для элементов первого порядка количество уравнений в СЛАУ k значительно. В *FreeFem++* метод сопряженных градиентов *CG* $n = [5, 8]$ показал более быструю скорость вычислений с использованием меньшего объема памяти по сравнению с другими методами. В *CalculiX* самую высокую скорость вычислений показал итерационный метод *Cholesky* с обуславливанием неполным разложением Холецкого ($n = [13, 15]$). Итерационный метод *Scaling* использовал на 30% меньше объема памяти из-за более простого обуславливания, чем метод *Cholesky*, а скорость вычислений была почти в 2 раза меньше, чем в *Cholesky* ($n = [15, 16]$). Метод *Cholesky* показал двукратную скорость вычислений по сравнению с *CG* для КЭ первого порядка ($n = [13, 5]$) и трехкратную скорость для элементов второго порядка ($n = [16, 8]$). Это может быть связано с тем, что итерационные методы в *FreeFem++* используют обуславливание только через подключение соответствующих пользовательских функций, а в проведенных расчетах обуславливание не использовалось.

На тестовой задаче из прямых методов *SPOOLES* ($n = 11$) показал лучшее быстродействие (в 1.6) и использовал меньший объем памяти (в 7.6 раз) относительно *UMFPACK* $n = 4$. Падение скорости вычислений для $n = 7$ в 5 раз по сравнению с $n = 14$ связано с использованием в $n = 7$ файла подкачки.

Использование элементов первого порядка приводит к высокой погрешности. В $n = 10$ и $n = 17$ расчетная сетка выбиралась таким образом, чтобы число уравнений k было близким к числу уравнений для элементов второго порядка ($n = [8, 16]$). Увеличение количества узлов в 7 раз ($n = [5, 10]$ и $n = [13, 17]$) уменьшило погрешность в 2 раза. При использовании КЭ второго порядка для того же количества уравнений погрешности уменьшаются в 7 раз ($n = [5, 8]$) или в 20 раз ($n = [13, 16]$). То есть при одинаковых вычислительных затратах точность вычислений для элементов второго порядка на грубой сетке выше, чем для элементов первого порядка на более плотной сетке.

Погрешность вычислений γ в пакете *CalculiX* и *FreeFem++* для элементов второго порядка для тестовых расчетов $n = 8$ и $n = 16$ составляет 0.7% и 0.2%. Элементы C3D4/C3D10 [43] и элементы P1/P2 [44] схожи (см. рис. 5): геометрия, количество узлов, порядок аппроксимирующих полиномов и положение промежуточных узлов в КЭ второго порядка (в середине соответствующего ребра) совпадают в обеих программах. Согласно документации,

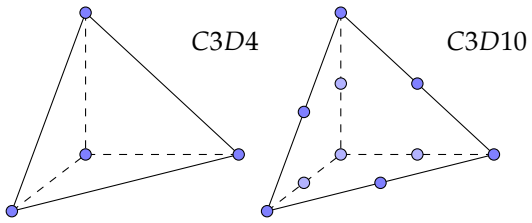

Рис. 5. Геометрия и расположение узлов (круги) в конечных элементах $C3D4$ и $C3D10$

в численном интегрировании КЭ-функции используется три точки интегрирования в *FreeFem++* и четыре точки в *CalculiX*.

7. Заключение

В работе приведен краткий обзор свободных программ численного моделирования статических задач упругости методом КЭ. Приведены примеры построения модели с использованием *FreeFem++*/Gmsh и *FreeCAD*/*CalculiX* (с использованием интерфейса программы и с использованием параметрического скрипта на языке *Python*). Проведен анализ использования вычислительных ресурсов различными прямыми и итерационными методами. В рамках рассмотренной тестовой задачи статической линейной упругости наиболее оптимальным методом в *FreeFem++* является итерационный метод сопряженных градиентов *CG* как по времени вычислений, так и по используемому объему памяти. Наибольшую скорость вычислений дает итерационный метод *Cholesky* с обуславливанием неполным разложением Холецкого в программе *CalculiX*.

На основе проведенного анализа вычислительных возможностей программ, их функциональности и удобства использования, обозначим особенности, которые могут повлиять на выбор программы численного моделирования. Особенности *FreeFem++*:

- необходимы навыки программирования;
- имеются функции пре/постпроцессора (создание 2D/3D геометрии и расчетной сетки; загрузка сетки, созданной в *Gmsh*; визуализация результатов расчетов);
- имеющиеся прямые методы используют больше вычислительных ресурсов по сравнению с *CalculiX*;
- имеются итерационные методы с возможностью подключения пользовательских предобуславливателей;

- позволяет создавать, перестраивать, адаптировать расчетные сетки в ходе выполнения кода, проводить полную параметризацию задачи, запускать серии расчетов по наборам параметров, проводить обработку результатов вычислений.

Особенности *CalculiX*:

- навыки программирования не требуются;
- *FreeCAD/CalculiX* позволяет провести полный цикл исследования в рамках одной программы, а также параметризацию задачи на языке программирования *Python*;
- в качестве отдельной программы содержит пре/постпроцессор (создание геометрии и расчетной сетки по командному файлу, визуализация данных по файлу результатов расчетов);
- имеет быстрый и экономичный к вычислительным ресурсам (в сравнении с прямыми методами *FreeFem++*) прямой метод *SPOOLES*;
- имеет быстрые итерационные методы с предобуславливанием.

Список литературы

[1] Ansys Mechanical — finite element analysis (FEA) software. https://www.ansys.com/products/structures/ansys-mechanical (дата обращения: 22.11.2020 г.)

[2] Simulia Abaqus. https://www.3ds.com/products-services/simulia/ (дата обращения: 22.11.2020 г.)

[3] LS-DYNA. http://www.lstc.com/ (дата обращения: 22.11.2020 г.)

[4] Delaunay B. Sur la sphère vide. A la mèmoire de Georges Voronoi // Bulletin de l'Académie des Sciences de l'URSS, Classe des Sciences Mathématiques et Naturelles. 1934. V. 6. P. 793–800. http://www.mathnet.ru/php/archive.phtml?wshow=paper&jrnid=im&paperid=4937&option_lang=eng

[5] Амосов А.А., Дубинский Ю.А., Копченова Н.В. Вычислительные методы для инженеров. М.: Высшая школа. 1994. 544 с.

[6] Chen K. Matrix Preconditioning Techniques and Applications, Cambridge University Press. 2005. 592 pp. https://www.cambridge.org/core/journals/journal-of-fluid-mechanics/article/abs/matrix-preconditioning-techniques-and-applications-by-k-chen-cambridge-university-press-2005-592-pp-isbn-0521-83828-2-55/BC30A72326AF23ED5D852AD2D0B145D2

[7] Golub G.H., Van Loan Ch.F. Matrix Computations. Fourth Edition. The Johns Hopkins University Press. 2013. 780 pp.

[8] Babuška I., Szabó B.A.. Katz I.N.. The p-version of the finite element method // SIAM Journal on Numerical Analysis. 1981. V. 18. P. 515–545.

[9] Babuška I., Szabó B.A. On the Rates of Convergence of the Finite Element Method // International Journal for Numerical Methods in Engineering. 1982. V. 18. P. 323–341,

[10] Babuška I., Dorr M.R. Error estimates for the combined h- and p- versions of the finite element method // Numer. Math. 1981. V. 7. P. 257—277. DOI: 10.1007/BF01398256

[11] ONELAB — Open Numerical Engineering LABoratory. http://onelab.info/ (дата обращения: 22.11.2020 г.)

[12] Geuzaine C., Remacle J.-F. Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities // International Journal for Numerical Methods in Engineering. 2009. Vol. 79, No. 11. P. 1309–1331. http://gmsh.info/doc/preprints/gmsh_paper_preprint.pdf (дата обращения: 22.11.2020 г.)

[13] GetDP — A General Environment for the Treatment of Discrete Problems. http://getdp.info/ (дата обращения: 22.11.2020 г.)

[14] FreeCAD. https://www.freecadweb.org/ (дата обращения: 22.11.2020 г.)

[15] Rieg F., Hackenschmidt R., Alber-Laukant B. Finite Element Analysis for Engineers. Basics and Practical Applications with Z88Aurora. Hanser. 2014. 719 pp.

[16] CalculiX — A Free Software Three-Dimensional Structural Finite Element Program. http://www.calculix.de/ (дата обращения: 22.11.2020 г.)

[17] Lyly M., Ruokolainen J., Järvinen E. ELMER - A finite element solver for multiphysics // CSC-report on scientific computing. 1999–2000. P. 156–159. https://www.csc.fi/documents/49902/86943/cscreport.pdf (дата обращения: 22.11.2020 г.)

[18] SALOME — the open source integration platform for numerical simulation. https://www.salome-platform.org/ (дата обращения: 22.11.2020 г.)

[19] Salome Meca and Code Aster. https://www.code-aster.org/ (дата обращения: 22.11.2020 г.)

[20] Schöberl J. NETGEN — An advancing front 2D/3D-mesh generator based on abstract rules // Computing and Visualization in Science. 1997. V. 1, No. 1. P. 41–52 DOI: 10.1007/s007910050004

[21] Hecht F. New development in FreeFem++ // Journal of Numerical Mathematics. 2012. V. 20, No. 3–4. Pp. 251–265. DOI: 10.1515/jnum-2012-0013

[22] Насибуллаев И.Ш., Насибуллаева Э.Ш. Течение жидкости через гидросопротивление с динамически изменяемой геометрией // Труды Института механики им. Р.Р. Мавлютова Уфимского научного центра РАН. 2017. Т. 12, № 1. С. 59–66. DOI: 10.21662/uim2017.1.009

[23] Насибуллаев И.Ш., Насибуллаева Э.Ш., Даринцев О.В. Изучение течения жидкости через деформируемый пьезоэлементом канал // Многофазные системы. 2018. Т. 13, № 3. С. 1–10. DOI: 10.21662/mfs2018.3.001

[24] Насибуллаев И.Ш., Насибуллаева Э.Ш., Даринцев О.В. Моделирование течения жидкости через деформируемый пьезоэлементом эластичный микроканал системы охлаждение микрозахвата // Мехатроника, автоматизация, управление. 2019. Т. 20, № 12. С. 740–750. DOI: 10.17587/mau.20.740-750

[25] Chiang Ch.-Yu, Pironneau O., Sheu T., Thiriet M. Numerical Study of a 3D Eulerian Monolithic Formulation for Incompressible Fluid-Structures Systems // Fluids. 2017. V. 2, No. 2. P. 34. DOI: 10.3390/fluids2020034

[26] Nasibullayev I.Sh., Darintsev O.V., Nasibullaeva E.Sh., Bogdanov D.R. Piezoelectric Micropumps for Microrobotics: Operating Modes Simulating and Analysis of the Main Parameters of the Fluid Flow Generation // Proceedings of 15th international conference on electromechanics and robotics "Zavalishin's reading" (Eds. by V. Shishlakov, A. Ronzhin), Smart Innovation, Systems and Technologies, 2021. Vol. 187, Chapter 43.
DOI: 10.1007/978-981-15-5580-0_43

[27] Pantz O. Treatment of contact between finite deformable bodies using FreeFem++. CMAP Ecole Polytechnique. 2011.
https://www.ljll.math.upmc.fr/hecht/ftp/ff++days/2011/Pantz.pdf (дата обращения: 22.11.2020 г.)

[28] Houssein H., Garnotel S., Hecht F. Frictionless contact problem for hyperelastic materials with interior point optimizer. 2019. hal-02355429
https://hal.archives-ouvertes.fr/hal-02355429/document (дата обращения: 22.11.2020 г.)

[29] Насибуллаев И.Ш., Даринцев О.В. Двумерная динамическая модель взаимодействия жидкости и пьезоэлектрического привода с поперечным изгибом в плоском канале // Многофазные системы. 2019. Т. 14, № 4. С. 220–232.
DOI: 10.21662/mfs2019.4.029

[30] Dhondt G. The Finite Element Method for Three-Dimensional Thermomechanical Applications. Wiley. 2004.

[31] Wittig K. CalculiX USER'S MANUAL- CalculiX GraphiX, Version 2.17.1. 2020.
http://www.dhondt.de/cgx_2.17.1.pdf (дата обращения: 22.11.2020 г.)

[32] Funke A., Wittig K. An investigation of a small jet engine.
http://www.calculix.de/ (дата обращения: 22.11.2020 г.)

[33] Abdulaziz A., Hedaya M., Mccrory J., Holford K., Elsabbagh A. Parametric Study of Honeycomb Composite Structure Using Open Source Finite Element Software. 2019.

[34] Vanti F., Agnolucci A., Pinelli L., Arnone A. An integrated numerical procedurefor flutter and forced response assessmentof turbomachinery blade-rows // Proceedings of 13th European Conference on Turbomachinery Fluid dynamics and ThermodynamicsETC13, April 8-12, 2019; Lausanne, Switzerland.

[35] Press W.H. Numerical Recipes 3rd Edition: The Art of Scientific Computing. Cambridge University Press. 2007. P. 50–52.

[36] Davis T.A. Algorithm 832: UMFPACK V4.3-an unsymmetric-pattern multifrontal method // ACM Trans. Math. Softw. 2004. V. 30, No. 2. P. 196–199.
DOI: 10.1145/992200.992206

[37] Saad Y., Schultz M.H. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems // SIAM J. Sci. Stat. Comput. 1986. V. 7. P. 856–869.
DOI: 10.1137/0907058

[38] Ashcraft C., Grimes R. SPOOLES: An Object-Oriented Sparse Matrix Library. PPSC. 1999.
https://static.aminer.org/pdf/PDF/000/548/747/spooles_an_object_oriented_sparse_matrix_library.pdf (дата обращения: 22.11.2020 г.)

[39] Schwarz H.R. FORTRAN-Programme zur Methode der finiten Elemente. Teubner. 1981.
DOI: 10.1002/zamm.19830631220

[40] Ландау Л., Лифшиц Е.М. Теоретическая физика. Т. 7. Теория упругости. М.: Наука. 1987. 248 с.

[41] Тихонов А.Н., Самарский А.А. Уравнения математической физики. М.: Наука. 1977. 735 с.

[42] FEM CalculiX Cantilever 3D.
https://wiki.freecadweb.org/FEM_Tutorial_Python/en (дата обращения: 22.11.2020 г.)

[43] Zienkiewicz O.C., Taylor R.L. The finite element method. McGraw-Hill Book Company. 1989.

[44] FreeFEM Documentation.
https://doc.freefem.org/_static/pdf/FreeFEM-doc-v3.pdf (дата обращения: 22.11.2020 г.)

ISSN 2658–5782 15 (2020), **3–4**, 183–200

Multiphase Systems

http://mfs.uimech.org/mfs2020.3.129
DOI: 10.21662/mfs2020.3.129

Received: 14.10.2020
Accepted: 17.12.2020

Application of free software FreeFem++/Gmsh and FreeCAD/CalculiX for simulation of static elasticity problems

Nasibullayev I.Sh.

Mavlyutov Institute of Mechanics UFRC RAS, Ufa, Russia

The paper discusses the stages of computer numerical simulation of engineering problems and ways to improve the accuracy of simulation; provides a brief overview of free software for simulation elasticity problems by the finite element method, as well as trends in the development of free CAD and CAE software. For a successful engineering study, it is necessary to choose a convenient tool that takes into account all the features of the problem being solved. Based on the solution of a test static problem of linear elasticity, two approaches to engineering modeling were demonstrated. The first approach requires programming skills - the full modeling cycle was written in the programming language of the *FreeFem++* software. Additionally, the method mesh generating in the *Gmsh* program with subsequent use in the *FreeFem++* program is shown. In the second approach, the full cycle of modeling is carried out through the interface of the *FreeCAD* program with the built-in *CalculiX* solver, which does not require programming skills. A way to parameterize the task using the *Python* interpreter built into *FreeCAD* is also proposed. The simulation results obtained using both approaches are compared for an object to which an external action is applied, determined by the Dirichlet or Neumann boundary conditions, and two types of object fastening are analyzed: rigid embedding and limitation by a plane with zero friction. The analysis of the use of computing resources by various direct and iterative methods is carried out. Within the framework of the considered test problem of static linear elasticity, the most optimal method in *FreeFem++* is the iterative method of conjugate gradients CG both in terms of computation time and in terms of the memory used. The highest speed of calculations is provided by the Cholesky iterative method with conditioning by the incomplete Cholesky expansion in the *CalculiX* program.

Keywords: static elasticity, free engineering software, FreeFem++, Gmsh, FreeCAD, CalculiX, direct and iterative solution methods of the system of linear equations

References

[1] Ansys Mechanical — finite element analysis (FEA) software. https://www.ansys.com/products/structures/ansys-mechanical (accessed: 22.11.2020).

[2] Simulia Abaqus. https://www.3ds.com/products-services/simulia/ (accessed: 22.11.2020).

[3] LS-DYNA. http://www.lstc.com/ (accessed: 22.11.2020).

[4] Delaunay B. Sur la sphère vide. A la mèmoire de Georges Voronoi // Bulletin de l'Académie des Sciences de l'URSS, Classe des Sciences Mathématiques et Naturelles. 1934. V. 6. P. 793–800. http://www.mathnet.ru/php/archive.phtml?wshow=paper&jrnid=im&paperid=4937&option_lang=eng

[5] Amosov A.A., Dubinsky Yu.A., Kopchenova N.V. [Computational methods for engineers] Vychislitel'nyye metody dlya inzhenerov. M: Vysshaya shkola. 1994. P. 544 (in Russian).

[6] Chen K. Matrix Preconditioning Techniques and Applications, Cambridge University Press. 2005. 592 pp. https://www.cambridge.org/core/journals/journal-of-fluid-mechanics/article/abs/matrix-preconditioning-techniques-and-applications-by-k-chen-cambridge-university-press-2005-592-pp-isbn-0521-83828-2-55/BC30A72326AF23ED5D852AD2D0B145D2

[7] Golub G.H., Van Loan Ch.F. Matrix Computations. Fourth Edition. The Johns Hopkins University Press. 2013. 780 pp.

[8] Babuška I., Szabó B.A.. Katz I.N.. The p-version of the finite element method // SIAM Journal on Numerical Analysis. 1981. V. 18. P. 515–545.

[9] Babuška I., Szabó B.A. On the Rates of Convergence of the Finite Element Method // International Journal for Numerical Methods in Engineering. 1982. V. 18. P. 323–341.

[10] Babuška I., Dorr M.R. Error estimates for the combined h- and p- versions of the finite element method // Numer. Math. 1981. V. 7. P. 257—277. DOI: 10.1007/BF01398256

[11] ONELAB — Open Numerical Engineering LABoratory. http://onelab.info/ (accessed: 22.11.2020).

[12] Geuzaine C., Remacle J.-F. Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. International Journal for Numerical Methods in Engineering. 2009. Vol. 79, No. 11. P. 1309–1331. http://gmsh.info/doc/preprints/gmsh_paper_preprint.pdf (accessed: 22.11.2020).

[13] GetDP — A General Environment for the Treatment of Discrete Problems. http://getdp.info/ (accessed: 22.11.2020).

[14] FreeCAD. https://www.freecadweb.org/ (accessed: 22.11.2020).

[15] Rieg F., Hackenschmidt R., Alber-Laukant B.. Finite Element Analysis for Engineers. Basics and Practical Applications with Z88Aurora. Hanser. 2014. 719 pp.

[16] CalculiX — A Free Software Three-Dimensional Structural Finite Element Program. http://www.calculix.de/ (accessed: 22.11.2020).

[17] Lyly M., Ruokolainen J., Järvinen E. ELMER - A finite element solver for multiphysics // CSC-report on scientific computing. 1999–2000. P. 156–159. https://www.csc.fi/documents/49902/86943/cscreport.pdf (accessed: 22.11.2020).

[18] SALOME — the open source integration platform for numerical simulation. https://www.salome-platform.org/ (accessed: 22.11.2020).

[19] Salome Meca and Code Aster. https://www.code-aster.org/ (accessed: 22.11.2020).

[20] Schöberl J. NETGEN - An advancing front 2D/3D-mesh generator based on abstract rules // Computing and Visualization in Science. 1997. V. 1, No. 1. P. 41–52 DOI: 10.1007/s007910050004

[21] Hecht F. New development in FreeFem++ // Journal of Numerical Mathematics. 2012. V. 20, No. 3–4. Pp. 251–265. DOI: 10.1515/jnum-2012-0013

[22] Nasibullayev I.Sh., Nasibullaeva E.Sh. [Fluid flow through hydraulic resistance with dynamically changing geometry] Techeniye zhidkosti cherez gidrosoprotivleniye s dinamicheski izmenyayemoy geometriyey. *Transactions of the Institute of Mechanics named after R.R. Mavlyutov, Ufa Scientific Center, Russian Academy of Scien* [Trudy Instituta mehaniki im. R.R. Mavlyutova]ces, Ufimskiy Nauchnyi Centr RAN]. 2017. V. 12, N. 1. P. 59–66 (in Russian). DOI: 10.21662/uim2017.1.009

[23] Nasibullayev I.Sh., Nasibullaeva E.Sh., Darintsev O.V., [Study of fluidflow through a channel deformed by piezoelement] Izucheniye techeniya zhidkosti cherez deformiruyemyy p'yezoelementom kanal. *Multiphase Systems* [Mnogofaznyye sistemy]. 2018. V. 13, No. 3. Pp. 1–10 (in Russian). DOI: 10.21662/mfs2018.3.001

[24] Nasibullayev I.Sh., Nasibullaeva E.Sh., Darintsev O.V., [Simulation of fluid flow through a elastic microchannel deformed by a piezoelement in microgrip cooling systems] Modelirovaniye techeniya zhidkosti cherez deformiruyemyy p'yezoelementom elastichnyy mikrokanal sistemy okhlazhdeniye mikrozakhvata. *Mekhatronika, Avtomatizatsiya, Upravlenie.* 2019. V. 20, No. 12. Pp. 740–750 (In Russian). DOI: doi:10.17587/mau.20.740-750

[25] Chiang Ch.-Yu, Pironneau O., Sheu T., Thiriet M. Numerical Study of a 3D Eulerian Monolithic Formulation for Incompressible Fluid-Structures Systems // Fluids. 2017. V. 2, No. 2. P. 34. DOI: 10.3390/fluids2020034

[26] Nasibullayev I.Sh., Darintsev O.V., Nasibullaeva E.Sh., Bogdanov D.R. Piezoelectric Micropumps for Microrobotics: Operating Modes Simulating and Analysis of the Main Parameters of the Fluid Flow Generation // Proceedings of 15th international conference on electromechanics and robotics "Zavalishin's reading" (Eds. by V. Shishlakov, A. Ronzhin), Smart Innovation, Systems and Technologies, 2021. Vol. 187, Chapter 43. DOI: 10.1007/978-981-15-5580-0_43

[27] Pantz O. Treatment of contact between finite deformable bodies using FreeFem++. CMAP Ecole Polytechnique. 2011 https://www.ljll.math.upmc.fr/hecht/ftp/ff++days/2011/Pantz.pdf (accessed: 22.11.2020).

[28] Houssein H., Garnotel S., Hecht F. Frictionless contact problem for hyperelastic materials with interior point optimizer. 2019. hal-02355429 https://hal.archives-ouvertes.fr/hal-02355429/document (accessed: 22.11.2020).

[29] Nasibullayev I.Sh., Darintsev O.V., [Two-dimensional dynamic model of the interaction of a fluid and a piezoelectric bending actuator in a plane channel] Dvumernaya dinamicheskaya model' vzaimodeystviya zhidkosti i p'yezoelektricheskogo privoda s poperechnym izgibom v ploskom kanale. *Multiphase Systems* [Mnogofaznyye sistemy]. 2019. V. 14, No. 4. Pp. 220–232 (in Russian). DOI: 10.21662/mfs2019.4.029

[30] Dhondt G. The Finite Element Method for Three-Dimensional Thermomechanical Applications. Wiley. 2004.

[31] Wittig K. CalculiX USER'S MANUAL — CalculiX GraphiX, Version 2.17.1. 2020. http://www.dhondt.de/cgx_2.17.1.pdf (accessed: 22.11.2020).

[32] Funke A., Wittig K. An investigation of a small jet engine. http://www.calculix.de/ (accessed: 22.11.2020).

[33] Abdulaziz A., Hedaya M., Mccrory J., Holford K., Elsabbagh A. Parametric Study of Honeycomb Composite Structure Using Open Source Finite Element Software. 2019.

[34] Vanti F., Agnolucci A., Pinelli L., Arnone A. An integrated numerical procedurefor flutter and forced response assessmentof turbomachinery blade-rows // Proceedings of 13th European Conference on Turbomachinery Fluid dynamics and ThermodynamicsETC13, April 8-12, 2019; Lausanne, Switzerland.

[35] Press W.H. Numerical Recipes 3rd Edition: The Art of Scientific Computing. Cambridge University Press. 2007. P. 50–52.

[36] Davis T.A. Algorithm 832: UMFPACK V4.3-an unsymmetric-pattern multifrontal method // ACM Trans. Math. Softw. 2004. V. 30, No. 2. P. 196–199. DOI: 10.1145/992200.992206

[37] Saad Y., Schultz M.H. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems // SIAM J. Sci. Stat. Comput. 1986. V. 7. P. 856–869. DOI: 10.1137/0907058

[38] Ashcraft C., Grimes R. SPOOLES: An Object-Oriented Sparse Matrix Library. PPSC. 1999. https://static.aminer.org/pdf/PDF/000/548/747/spooles_an_object_oriented_sparse_matrix_library.pdf (accessed: 22.11.2020).

[39] Schwarz H.R. FORTRAN-Programme zur Methode der finiten Elemente. Teubner. 1981. DOI: 10.1002/zamm.19830631220

[40] Landau L.D., Lifshitz E.M. [Theoretical physics. V. 7. Theory of Elasticity] Teoreticheskaya fizika. T. 7. Teoriya uprugosti. M.: Nauka. 2003. P. 259. (in Russian).

[41] Tikhonov A.N., Samarsky A.A. [Equations of mathematical physics] Uravneniya matematicheskoy fiziki. M.: Nauka. 1977. P. 735 (in Russian).

[42] FEM CalculiX Cantilever 3D. https://wiki.freecadweb.org/FEM_Tutorial_Python/en (accessed: 22.11.2020).

[43] Zienkiewicz O.C., Taylor R.L. The finite element method. McGraw-Hill Book Company. 1989.

[44] FreeFEM Documentation. https://doc.freefem.org/_static/pdf/FreeFEM-doc-v3.pdf (accessed: 22.11.2020).