e-ISSN 1607-3274 Радіоелектроніка, інформатика, управління. 2021. № 2
p-ISSN 2313-688X Radio Electronics, Computer Science, Control. 2021. № 2

УДК 004.9:004.94:51

РЕАЛИЗАЦИЯ КОНЕЧНО-ЭЛЕМЕНТНОЙ БИБЛИОТЕКИ КЛАССОВ С ИСПОЛЬЗОВАНИЕМ ОБОБЩЕННОГО ПРОГРАММИРОВАНИЯ

Чопоров С. В. – д-р техн. наук, профессор, профессор кафедры программной инженерии Запорожского национального университета, Запорожье, Украина.

Игнатченко М. С. – аспирант кафедры программной инженерии Запорожского национального университета, Запорожье, Украина.

Кудин А. В. – канд. физ.-мат. наук, доцент кафедры программной инженерии Запорожского национального университета, Запорожье, Украина.

Кривохата А. Г. – канд. физ.-мат. наук, старший преподаватель кафедры программной инженерии Запорожского национального университета, Запорожье, Украина.

Гоменюк С. И. – д-р техн. наук, профессор, декан математического факультета, профессор кафедры программной инженерии Запорожского национального университета, Запорожье, Украина.

АННОТАЦИЯ

Актуальность. Для компьютерного моделирования сложных объектов и явлений различной природы на практике часто используют численный метод конечных элементов. Его программная реализация (особенно для исследования новых классов задач) является достаточно трудоемким процессом.

Высокая стоимость разработки программного обеспечения обусловливает актуальность разработки новых подходов к повышению эффективности программирования и сопровождения (в т.ч. добавление новых функций).

Цель. Цель работы – создание новой эффективной архитектуры программ конечно-элементного анализа проблем математической физики, позволяющей легко расширять их функциональность для решения новых классов задач.

Метод. Предложен метод разработки программ для конечно-элементного анализа с использованием обобщенного программирования, что дает возможность существенно упростить архитектуру программного обеспечения и сделать его более удобным для сопровождения и модификации за счет разделения алгоритмов и структур данных.

Предложена новая архитектура классов, реализующих конечно-элементный расчет, позволяющая легко расширять функциональность программ за счет добавления новых типов конечных элементов, методов решения систем линейных алгебраических уравнения, параллельных вычислений и т.д.

Результаты. Предложенный подход был программно реализован в виде библиотеки классов на языке C++. Проведен ряд вычислительных экспериментов, подтвердивших его работоспособность при решении практических задач.

Выводы. Разработанный подход можно использовать как для создания систем конечно-элементного анализа общего назначения с открытой архитектурой, так и для реализации специализированных программных пакетов, ориентированных на решение конкретных классов задач (механики разрушения, эластомеров, контактного взаимодействия и т.п.).

КЛЮЧЕВЫЕ СЛОВА: метод конечных элементов, конечный элемент, объектно-ориентированное программирование, обобщённое программирование, архитектура программного обеспечения.

АББРЕВИАТУРЫ

ПО – программное обеспечение;

МКЭ – метод конечных элементов;

КЭ – конечный элемент;

СЛАУ – система линейных алгебраических уравнений;

НОМЕНКЛАТУРА

u_i – компоненты вектора перемещений;

u_i^l – искомое значение функции $u_i(\mathbf{x})$ в l-ой вершине (узле) конечного элемента;

ε_{ij} – компоненты тензора деформации;

σ_{ij} – компоненты тензора напряжения;

X_i – компоненты вектора объемных нагрузок;

$\overline{X}_i$ – компоненты вектора поверхностных нагрузок;

Ω – область расчета;

Γ – поверхность области расчета;

$N_l(\mathbf{x})$ – функция формы КЭ;

M – количество узлов КЭ;

$\mathbf{D}$ – матрица упругости, содержащая характеристики материала;

$\mathbf{N}$ – вектор компонент функций формы;

$\mathbf{B}$ – вектор производных от функций формы;

$\mathbf{p}$ – вектор компонент объемной нагрузки;

$\mathbf{q}$ – вектор компонент поверхностной нагрузки;

$\mathbf{K}$ – матрица жесткости;

$\mathbf{F}$ – вектор компонент нагрузки.

ВВЕДЕНИЕ

Использование вычислительной техники для замены физических испытаний виртуальным компьютерным экспериментом позволяет существенно уменьшить издержки при проектировании и производстве сложных современных инженерно-технических систем.

Применение МКЭ является одним из наиболее эффективных при реализации компьютерного анализа. Таким образом, возникает необходимость в постоянном совершенствовании существующего и разработке нового ПО для компьютерного моделирования различных объектов и процессов.

DOI 10.15588/1607-3274-2021-2-17

Объектом исследования являлся процесс разработки ПО для математического моделирования и анализа сложных инженерно-технических систем.

Наиболее важным этапом проектирования является оценка прочности и долговечности создаваемой техники. Это приводит к необходимости анализа ее напряженно-деформированного состояния в процессе эксплуатации. В большинстве случаев аналитическое решение соответствующих задач не представляется возможным, поэтому на практике применяют различные численные методы, наиболее популярным среди которых в силу своей универсальности является МКЭ [1].

Для применения МКЭ в настоящее время разработано большое количество различных коммерческих программных систем, самыми известными среди которых являются Ansys [2], COMSOL [3], MSC Nastran [4] и др. [5].

Альтернативой им являются ПО с открытым исходным кодом, среди которого можно выделить FreeFEM [6], GetFEM [7], OpenCAD [8] и др. [5, 9, 10].

Сложность стоящих перед промышленностью задач требует постоянной разработки новых методов и алгоритмов решения различных классов задач (механики разрушения, гидро- и аэродинамики, контактных задач и т. д.), а особенно новых композитных материалов. Все это обусловливает необходимость создания такого ПО для конечно-элементного анализа, архитектура которого бы позволяла легко встраивать новые типы КЭ для учета особенностей того или иного конструкционного материала (например, эластомера); новые методы решения СЛАУ; поддержку параллельных вычислений и т. п.

Предметом исследования являлись методы разработки программ конечно-элементного анализа для создания удобной к модификации архитектуры соответствующего ПО.

Одним из возможных путей решения этой задачи является использование обобщенной парадигмы программирования, позволяющей отделить наиболее общие алгоритмы от соответствующих структур данных.

Целью работы являлось создание такой архитектуры ПО для конечно-элементного анализа, которая бы позволила сократить время на его разработку и сопровождение.

1 ПОСТАНОВКА ЗАДАЧИ

Анализ проектируемой конструкции на прочность и долговечность обычно сводится к моделированию ее напряженно-деформированного состояния в процессе эксплуатации, параметры которого в статике можно найти, минимизируя следующий энергетический функционал (вариационный принцип Лагранжа) [1]:

$$\int_{\Omega} \sigma_{ij}\varepsilon_{ij} d\Omega - \int_{\Omega} X_i u_i d\Omega - \int_{\Gamma} \overline{X}_i u_i d\Gamma = 0 . \qquad (1)$$

Одним из наиболее эффективных способов минимизации функционала (1) является использование МКЭ. Для его применения исходную область Ω разбивают на некоторое множество КЭ заданной формы Ω_k, $k = \overline{1, N}$. Аппроксимация искомой функции (например, компонент вектора перемещений) для заданного КЭ описывается следующим соотношением:

$$u_i(\mathbf{x}) = \sum_{l=1}^{M} u_i^l N_l(\mathbf{x}) . \qquad (2)$$

Как функции формы чаще всего используют полиномы, коэффициенты которых определяются из следующего соотношения:

$$N_l(x_m) = \begin{cases} 1, l = m, \\ 0, l \neq m. \end{cases}$$

Подставляя (2) в (1) с учетом соотношений Коши и обобщенного закона Гука, получим соотношение, которое в матричном виде можно записать так:

$$\int_{\Omega_k} \mathbf{B}^T \mathbf{D}\mathbf{B} d\Omega_k = \int_{\Omega_k} \mathbf{N}^T \mathbf{p} d\Omega_k + \int_{\Gamma_k} \mathbf{N}^T \mathbf{q} d\Gamma_k . \qquad (3)$$

Минимизация соотношения (3) при помощи, например, метода вариации произвольных постоянных позволяет окончательно получить СЛАУ, которую в матричной форме можно записать следующим образом:

$$\mathbf{K}\mathbf{u} = \mathbf{F}. \qquad (4)$$

Таким образом, в общем виде возникает задача создания такого ПО, которое бы автоматизировало все стадии решения задачи с помощью МКЭ: от разбиения исходной области на КЭ заданного типа, до решения СЛАУ вида (4). При этом существенный интерес представляет обеспечение возможности создания лаконичного кода и простоты изменения типов контейнеров и алгоритмов.

2 ОБЗОР ЛИТЕРАТУРЫ

Обобщенное программирование – это парадигма разработки ПО, базирующаяся на идее написания алгоритмов таким образом, чтобы они не зависели от типов обрабатываемых данных. Например, на языке программирования C++ обобщенную (шаблонную) процедуру поиска максимального значения среди двух объектов можно описать при помощи кодов:

```
template <typename T> T max(const T& lhs, const T& rhs)
{
  return lhs < rhs ? rhs : lhs;
}
```

Эта функция может быть использована для любых типов данных T, для которых определена операция сравнения «меньше».

DOI 10.15588/1607-3274-2021-2-17

Общую теорию использования обобщенного программирования описано в работе [11]. Методологию написания программ с использованием шаблонных классов на языке C++ рассмотрено в [12–14].

Одними из первых публикаций, посвященных использованию парадигмы обобщенного программирования при разработке систем конечно-элементного анализа, являются работы [15, 16]. В них исследованы самые общие принципы применения шаблонов при реализации базовых контейнерных классов и итераторов.

В [17] рассмотрена обобщенная реализация низкоуровневых матричных операций, используемых при создании ПО для МКЭ.

Принципы реализации шаблонных классов для параллельной реализации векторов большой длины предложены в [18].

В работе [19] приведены примеры использования элементов функционального и обобщенного программирования для реализации разных типов КЭ и базовых операций над ними: вычисление функций формы, их производных, а также различных вспомогательных операций над ними. Основной упор сделан на реализацию векторных вычислений с использованием соответствующих возможностей современных графических процессоров.

Повышение эффективности конечно-элементных вычислений при использовании обобщенного программирования обосновывается в [20].

Различные варианты обобщенной реализации структур данных и алгоритмов для описания КЭ, дискретных моделей, параллельных вычислений и т.п. приведены в [21, 22].

В большинстве известных работ обобщенное программирование применяется для создания низкоуровневых процедур, реализующих вспомогательные операции или структуры данных, необходимые для разработки конечно-элементного ПО.

Таким образом, разработка высокоуровневых подходов для реализации новых типов КЭ или алгоритмов расчета является актуальной.

3 МАТЕРИАЛЫ И МЕТОДЫ

Алгоритм применения МКЭ можно условно разделить на два основных этапа: 1) формирование локальных матриц жесткости, массы и сопротивления (демпфирования) для каждого КЭ и добавление их к соответствующим глобальным матрицам (ансамблирования); 2) учета краевых условий, формирования СЛАУ и ее решения.

Первый этап – формирование глобальных матриц может быть достаточно нетривиальной и длительной процедурой в случае использование специальных типов КЭ, например, оболочечных. Поэтому его программная реализация является важнейшей частью разработки любого ПО для конечно-элементного анализа. На практике требуется создание программ с такой архитектурой, которая бы позволяла сравнительно легкое добавление к уже существующей библиотеке новые типы КЭ, предназначенные для решения специальных классов задач (многослойных оболочек, композитов, контактного взаимодействия и т.д.) или оптимизированные по скорости вычислений.

Наиболее распространенными на практике формами КЭ являются треугольные и четырехугольные элементы в плоском случае, а также тетраэдральные и кубические элементы в трехмерном пространстве. Большинство автоматических генераторов сеток ориентированы на дискретизацию исходных геометрических областей на элементы именно таких типов.

Для описания формы наиболее распространенных на практике изопараметрических КЭ предлагается следующая иерархия классов, UML-диаграмма которой приведена на рис. 1.

Здесь базовый абстрактный класс Shape описывает наиболее общие свойства формы изопараметрического элемента, такие как количество его узлов, коэффициенты функции формы, параметры квадратур для численного интегрирования и т.п. Кроме того, данный класс содержит набор методов, предназначенных для вычисления коэффициентов функций формы, Якобиана, значений функций формы и ее производных в заданных точках и т.п. Большинство этих методов являются абстрактными и реализованы в производных от Shape классах.

Классы Shape1D2, Shape2D3 и Shape3D4 инкапсулируют наиболее распространенные линейные формы КЭ: стержень, треугольник и тетраэдр. Производные от них классы реализуют либо элементы более высоких степеней аппроксимации (Shape1D3 – квадратичный трехузловой стержень; Shape2D6 – квадратичный шестиузловой треугольник; Shape3D10 – квадратичный десятиузловой тетраэдр), либо элементы другой формы (Shape2D4 – четырехугольник; Shape3D8 – кубический элемент).

Соответствующую иерархию классов шаблонов, описывающих различные типы КЭ, можно представить следующим образом (рис. 2).

Здесь базовый абстрактный класс FE описывает наиболее общие свойства и методы КЭ: его форму, количество степеней свободы, упругие и физические характеристики, толщину, локальные матрицы жесткости, массы и сопротивления и т.п. В данном классе также задекларирован абстрактный метод генерации локальных матриц.

Производные от FE классы шаблонов содержат реализацию процедуры построения локальных матриц для конкретного типа элемента. В качестве параметра они принимают форму элемента. Например, класс FE2DP<Shape2D3> описывает конечный элемент пластины треугольной формы с линейной аппроксимацией, а FE2DP<Shape2D4> – элемент четырехугольной формы с билинейной аппроксимацией.

Такой способ описания классов, реализующих изопараметрические КЭ, позволяет, с одной стороны, отделить реализацию формы элемента от алгоритмов вычисления локальных матриц (численного интегрирования), а с другой – существенно упростить создание элементов новой формы и/или типа.

DOI 10.15588/1607-3274-2021-2-17

Второй этап применения МКЭ – решение СЛАУ также является достаточно длительной и ресурсоемкой процедурой, особенно при решении сверхбольших систем уравнений, возникающих, например, при создании новой аэрокосмической техники.

Совместно с МКЭ обычно используются два типа методов решения СЛАУ: прямые и итерационные. Прямые методы позволяют получить точное решение (без учета машинной погрешности), но обычно они требуют значительных объемов оперативной памяти компьютера и могут быть неэффективными при решении больших СЛАУ. Приближенные итерационные методы обычно требуют существенно меньше оперативной памяти, поэтому часто используются для решения систем уравнений высоких порядков, хотя время вычислений при этом может быть значительным. Кроме того, для многих существующих методов решения СЛАУ в настоящее время разрабатываются параллельные реализации для компьютерных систем как с общей, так и распределенной памятью.

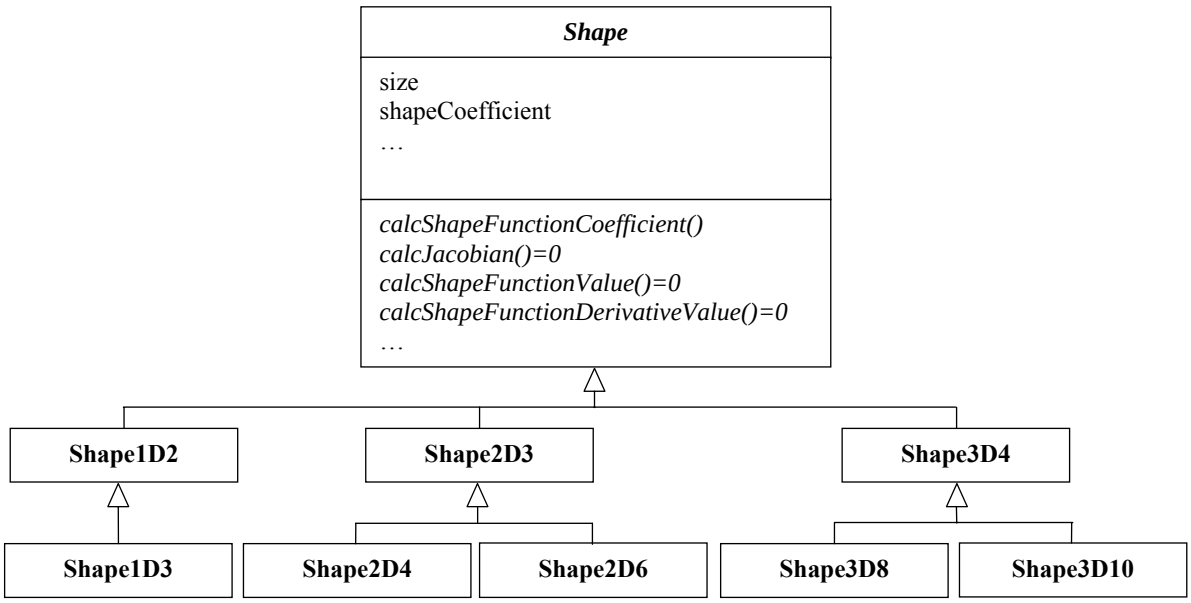

Рисунок 1 – Иерархия классов, описывающих форму конечного элемента

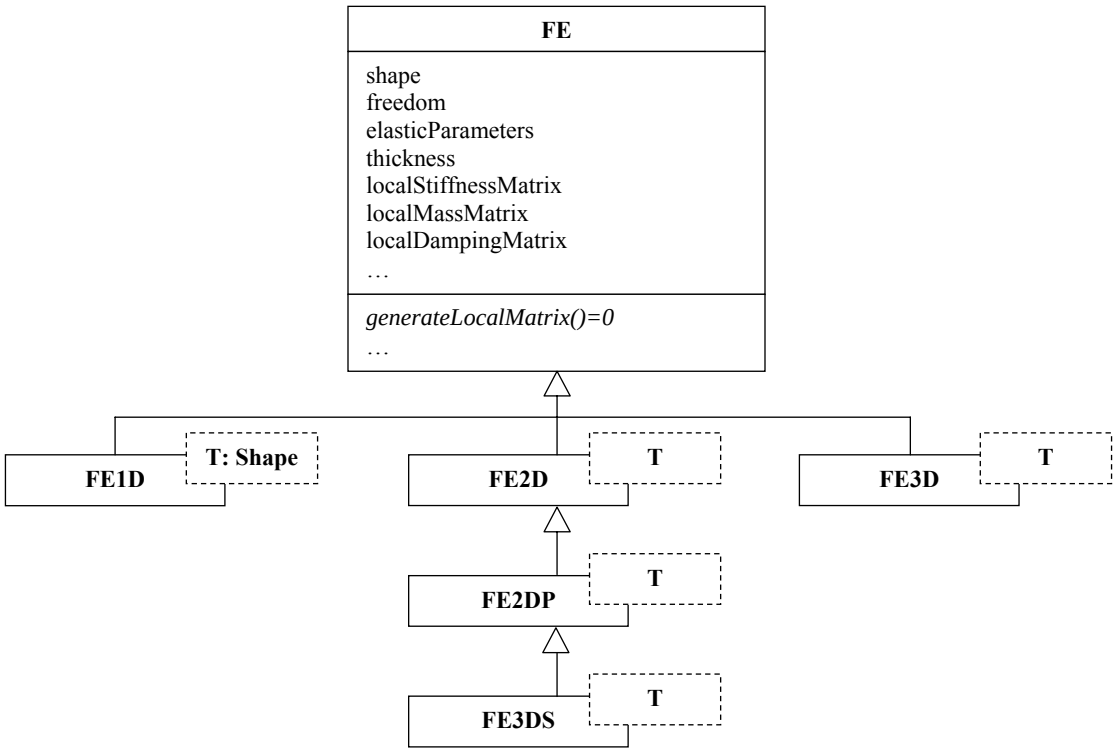

Рисунок 2 – Иерархия классов шаблонов, описывающих различные типы конечных элементов

DOI 10.15588/1607-3274-2021-2-17

Таким образом, возникает задача разработки такой реализации систем конечно-элементного анализа, в которой алгоритмы решения задачи с использованием МКЭ не зависели бы от метода решения СЛАУ. Для этого предлагается обобщенный подход, когда метод решения СЛАУ является параметром алгоритма расчета задачи с использованием МКЭ.

Для этого, в первую очередь, нужно создать абстрактный класс-решатель, инкапсулирующий все необходимые операции для формирования СЛАУ в МКЭ: добавление элемента в матрицу; добавление элемента в вектор столбец правой части; учета краевых условий и т.д.

Диаграмма иерархии классов, реализующих решатель СЛАУ, приведена на рис. 3.

Здесь параметром класса шаблона Solver<T> является обобщенный тип T, инстанцируемый как реализованный в каждом конкретном решателе контейнер для хранения матрицы коэффициентов СЛАУ.

Для использования классов шаблонов, производных от Solver<T>, предлагается следующая иерархическая структура, инкапсулирующая различные алгоритмы применения МКЭ (рис. 4).

Здесь абстрактный класс FEM описывает базовые для МКЭ атрибуты, такие как: параметры расчета; конечно-элементную сетку; конечный элемент, абстрактную процедуру расчета и т.п.

Производные от FEM классы шаблонов реализуют определенные виды расчетов (например, линейную статику) и в качестве параметра принимают шаблоны следующих классов: решателя СЛАУ, производного от Solver; конечного элемента, производного от FE; формы конечного элемента, производного от Shape.

4 ЭКСПЕРИМЕНТЫ

Для изучения возможностей предложенного обобщенного подхода к организации конечно-элементных классов на языке C++ была программно реализована библиотека QFEM [23], с помощью которой было выполнено численное решение ряда практических задач.

Вначале была рассмотрена тестовая задача о нахождении прогибов квадратной пластины с жестко защемленными краями, находящейся под действием равномерно распределенной поверхностной нагрузки. Были выбраны следующие параметры пластины: сторона – 1 м; толщина – 0,01 м; модуль Юнга – 203200 МПа; коэффициент Пуассона – 0,27; нагрузка – 0,05 МН/м^2.

В простейшем случае программа решения такой задачи с использованием библиотеки QFEM может быть реализована таким образом:

```
#include "fem/femstatic.h"
#include "analyse/analyse.h"
#include "solver/eigensolver.h"

// Инициализация системы сообщений
TMessenger *msg = new Tmessenger();
int main()
{
```

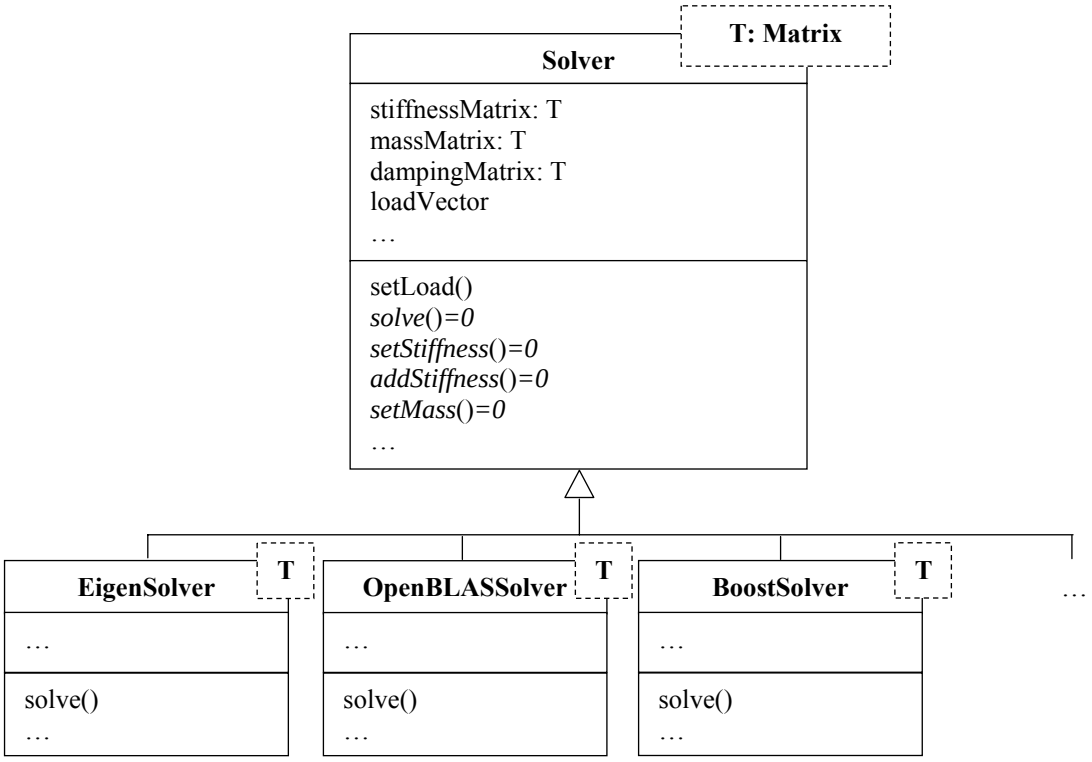

Рисунок 3 – Обобщенная реализация классов для решения СЛАУ

DOI 10.15588/1607-3274-2021-2-17

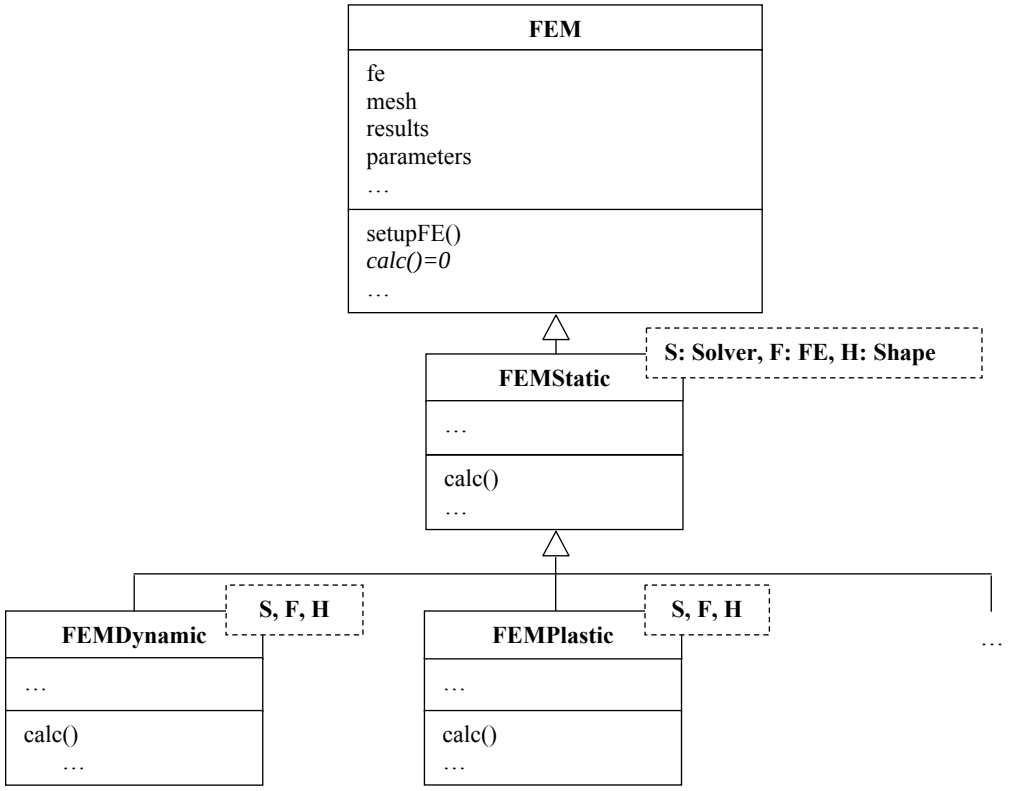

Рисунок 4 – Диаграмма классов шаблонов, реализующих различные алгоритмы МКЭ

```
// Конечно-элементная модель пластины
matrix<double> x = {{-0.5, -0.5}, {0, -0.5},
                    {0.5, -0.5}, {0.5, 0},
                    {0.5, 0.5}, {0, 0.5},
                    {-0.5, 0.5}, {-0.5, 0},
                    {0, 0}};
matrix<int> fe = {{0, 1, 8}, {0, 8, 7}, {1, 2, 8},
                  {2, 3, 8}, {3, 4, 8}, {4, 5, 8},
                  {5, 6, 8}, {6, 7, 8}},
            be = fe;
TMesh mesh(FEType::fe2d3p, x, fe, be);
TResults result;
TFEMParams params;
TFEMStatic<TEigenSolver, TShape2D3,
    TFE2DP> fem("Plate", &mesh, &result);

params.addYoungModulus(203200);
params.addPoissonRatio(0.27);
params.addThickness(0.01);
params.addBoundaryCondition(Direction::X |
  Direction::Y | Direction::Z, 0,
  "x == -0.5 or x == 0.5 or y == -0.5 or y == 0.5");
params.addPressureLoad(0.05);

fem.setParams(params);
fem.startProcess();

delete msg;
return 0;
}
```

В приведенном примере выполняется расчет пластины, состоящей из 8 линейных треугольных КЭ. Для расчета пластины, состоящей из четырехугольных КЭ, вышеприведенную программу нужно переделать следующим образом:

```
…
  matrix<int> fe = {{0, 1, 8, 7}, {1, 2, 3, 8},
                    {3, 4, 5, 8}, {5, 6, 7, 8}};
  TFEMStatic<TEigenSolver, TShape2D4,
      TFE2DP> fem("Plate", &mesh, &result);
…
```

Расчет данной задачи выполнялся также при использовании сетки, состоящей из 2173 узлов и 1046 КЭ в форме квадратичного (шестиузлового) треугольника, а также сетки из 40401 узла и 40000 четырехугольных билинейных элементов. Полученное распределение прогибов пластины приведено на рис. 5.

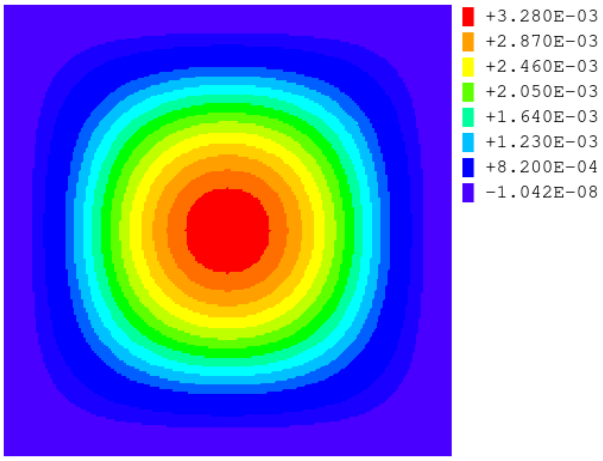

Рисунок 5 – Распределение прогибов по пластине

DOI 10.15588/1607-3274-2021-2-17

Расчетный максимальный прогиб пластины – 0.00343657 для шестиузловых треугольных элементов и 0,00327985 для четырехугольных элементов. Погрешность полученного численного результата по сравнению с имеющимся аналитическим решением [24] составляет 0,36%.

Следует отметить, что решение СЛАУ, возникающих при использовании МКЭ, является достаточно сложной задачей, т.к. в случае использования специальных типов КЭ могут возникать плохо обусловленные матрицы. На практике зачастую не всегда возможно заранее выбрать оптимальный метод решения СЛАУ, что приводит к необходимости проведения различных вычислительных экспериментов и подбору оптимального метода решения конкретной СЛАУ. Предложенный обобщенный подход к описанию класса, реализующего конечно-элементный расчет, где метод решения системы уравнений фактически является его параметром, позволяет существенно упростить задачу подбора оптимальных параметров решателя.

Так, например, при численном расчете вышеописанной задачи на компьютере с процессором AMD Ryzen 7 2700X Eight-Core Processor 3.70 GHz и объемом памяти 32 ГБ под управлением ОС Windows x64 при использовании сетки, состоящей из 40401 узлов и 40000 четырехугольных КЭ, время решения СЛАУ разными методами может значительно различаться (табл. 1).

Таким образом, обобщенная реализация решателя конечно-элементных классов позволяет без изменения структуры библиотеки легко добавлять новые реализации методов решения СЛАУ с последующей оптимизацией выбора метода решения для конкретной задачи.

5 РЕЗУЛЬТАТЫ

Для верификации предложенного подхода был выполнен численный анализ напряженно-деформированного состояния тонкостенной трубы с жестко защемленными краями, находящейся под действием внутреннего давления.

Задача решалась при следующих параметрах: радиус трубы – 1,99 м; длина – 4,014 м; толщина стенки – 0,0369 м; модуль Юнга – 203200 МПа; коэффициент Пуассона – 0,27; равномерно распределенная по внутренней поверхности нагрузка – 0,05 МН/м^2.

Результаты проведенных экспериментов при использовании различных типов КЭ приведены в табл. 2.

На рис. 6 приведено распределение радиальных прогибов по трубе с визуализацией деформирования конструкции под действием нагрузки.

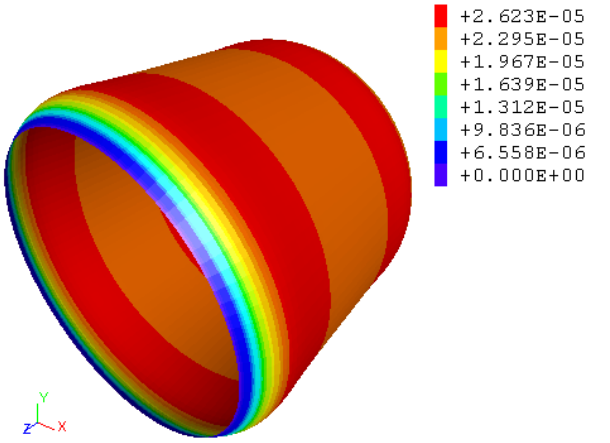

Рисунок 6 – Распределение прогибов по трубе

6 ОБСУЖДЕНИЕ

Как видно из табл. 2 использование различных типов КЭ для данной тестовой задачи дает расхождение между результатами не более 1,2%, что может свидетельствовать о достоверности полученного решения.

Предложенная структура классов шаблонов, описывающих КЭ, по сравнению с [19, 20, 22] позволяет существенно упростить процесс реализации новых типов элементов за счет разделения их формы и алгоритмов построения локальных матриц жесткости, массы и демпфирования.

По сравнению с [18, 21] предложенная структура классов шаблонов, разделяющая алгоритмы применения МКЭ и структуры данных для хранения разреженных матриц и решения соответствующих им СЛАУ, позволяет без изменения структуры ПО добавлять новые методы решения СЛАУ, что дает возможность, например, использовать параллельные компьютерные системы при решении задач высоких порядков.

Таблица 1 – Сравнение методов решения СЛАУ

Метод	Тип	Библиотека	Время расчета, с
Градиентный спуск	Итерационный	Eigen	1286
Метод Ланцоша	Итерационный	QFEM	870
LLT-факторизация	Прямой	Eigen	22
Прямое LLT-разложение	Прямой	Intel MKL	1

Таблица 2 – Результаты численных экспериментов

Тип КЭ	Количество узлов	Количество КЭ	Максимальный прогиб, м
Линейный тетраэдр	11845	34783	$2{,}63181\times10^{-5}$
Квадратичный тетраэдр	70317	34783	$2{,}61435\times10^{-5}$
Линейный оболочечный треугольник	28244	55984	$2{,}65075\times10^{-5}$
Билинейный оболочечный четырехугольник	5100	5000	$2{,}62280\times10^{-5}$
Квадратичный оболочечный треугольник	25752	12748	$2{,}62033\times10^{-5}$

DOI 10.15588/1607-3274-2021-2-17

Использование предложенной архитектуры ПО для конечно-элементного анализа по сравнению с [21], а также со стандартными объектно-ориентированными или функциональными подходами позволяет сократить затраты времени как на саму разработку программ, так и на последующее их сопровождение, а именно: добавление новых типов КЭ, методов решения СЛАУ, а также алгоритмов решения с помощью МКЭ новых классов задач.

При этом использование обобщенного подхода к разделению данных и алгоритмов позволяет уменьшить количество возникающих при разработке ошибок.

ВЫВОДЫ

Решена актуальная задача создания обобщенной архитектуры программ для конечно-элементного анализа, позволяющей уменьшить затраты времени на разработку ПО за счет разделения структур данных и алгоритмов их разработки, что дает возможность также снизить вероятность появления ошибок в программном коде.

Научная новизна полученных результатов состоит в том, что:

– впервые предложен обобщенный подход к разработке классов, реализующих КЭ, в которых форма элемента является параметром алгоритмов построения локальных матриц жесткости, массы и демпфирования, что позволяет значительно упростить процедуру добавления новых типов КЭ к уже имеющимся и уменьшить вероятность появления ошибок в их программной реализации;

– впервые предложен обобщенный подход к реализации процедуры решения СЛАУ, позволяющий за счет создания высокоуровневой обертки для различных методов решения систем уравнения унифицировать структуры данных и алгоритмы для хранения больших разреженных матриц, возникающих при применении МКЭ.

Таким образом, впервые предложена обобщенная архитектура ПО для конечно-элементного анализа, в которой тип и форма КЭ, а также метод решения СЛАУ являются параметрами алгоритмов решения задач с использованием МКЭ. Такой подход позволяет без изменения структуры ПО и алгоритмов расчета добавлять новые специализированные типы КЭ и методы решения СЛАУ, предназначенные как для решения новых классов задач, так и для использования конкретных возможностей вычислительной техники (параллельных вычисления, использования GPU и т.п.).

Практическая ценность полученных результатов состоит в том, что на языке C++ реализована библиотека классов шаблонов для решения различных задач механики (статика, динамика, пластичность), а также проведено экспериментальное исследование, подтвердившее работоспособность предложенных обобщенных подходов при решении ряда практических задач. Разработанное ПО может быть рекомендовано для решения актуальных задач проектирования современной техники.

Перспективы дальнейших исследований состоят в том, чтобы расширить возможности разработанной библиотеки классов для использования параллельных компьютерных систем как с общей, так и распределенной памятью.

БЛАГОДАРНОСТИ

Работа выполнена в рамках госбюджетной научно-исследовательской темы «Математическое и программное обеспечение автоматизированного проектирования аэрокосмической техники» (гос. рег. № 0118U000210) Запорожского национального университета.

ЛИТЕРАТУРА / ЛІТЕРАТУРА

1. Zienkiewicz O. C. The Finite Element Method: Its Basis and Fundamentals. Sixth edition / O. C. Zienkiewicz, R. L. Taylor, J. Z. Zhu. – Oxford : Butterworth-Heinemann, 2016. – 753 p. DOI: 10.1002/nme.1760
2. Madenci E. The Finite Element Method and Applications in Engineering Using ANSYS® / E. Madenci. I. Guven. – Boston : Springer, 2015. – 651 p. DOI: 10.1007/978-1-4899-7550-8
3. Zimmerman W. Multiphysics Modeling with Finite Element Methods / W. Zimmerman. – Singapore : World Scientific, 2006. – 432 p. DOI: 10.1142/6141
4. MSC Nastran – Multidisciplinary Structural Analysis. [Electronic resource] – Access mode: https://www.mscsoftware.com/product/msc-nastran
5. Top Finite Element Analysis (FEA) Software: List, Reviews, Comparison & Price | TEC. [Electronic resource] – Access mode: https://www3.technologyevaluation.com/sd/category/finite-element-analysis-fea
6. FreeFEM – An open-source PDE Solver using The Finite Element Method: [Electronic resource] – Access mode: https://freefem.org/
7. GetFEM Homepage – GetFEM. [Electronic resource] – Access mode: http://getfem.org/
8. OpenCAD The Programmers Solid 3D CAD Modeller. [Electronic resource] – Access mode: https://www.openscad.org/
9. The dial.II Finite Element Library. [Electronic resource] – Access mode: https://www.dealii.org/
10. Netgen/NGSolve. [Electronic resource] – Access mode: https://ngsolve.org/
11. Stepanov A. A. From Mathematics to Generic Programming / A. A. Stepanov, D. E. Rose. – Crawfordsville : Addison-Wesley, 2014. – 292 p.
12. Meyers S. Effective Modern C++: 42 Specific Ways to Improve Your Use of C++11 and C++14 / S. Meyers. – Sebastopol : O'Reilly, 2017. – 316 p.
13. Galowicz J. C++17 STL Cookbook / J. Galowicz. – Birmingham : Packt Publishing Ltd, 2017. – 504 p.
14. Vandevoorde D. C++ Templates: The Complete Guide / D. Vandevoorde, N. M. Josuttis, D. Gregor. – Boston : Addison-Wesley, 2017. – 788 p.
15. Bittencourt M. L. Using C++ templates to implement finite element classes / M. L. Bittencourt // Engineering Computation. – 2000. – Vol. 17(7). – P. 775–788. DOI: 10.1108/02644400010352243
16. Bangerth W. Using Modern Features of C++ for Adaptive Finite Element Methods: Dimension-Independent Programming in deal.II / W. Bangerth // Proceedings of the 16th IMACS

DOI 10.15588/1607-3274-2021-2-17

World Congress. – 2000. [Electronic resource] – Access mode: https://www.dealii.org/7.3.0/
17. FEMEngine: finite element method C++ code based on functional and template metaprogramming / [A. Gurin, A. Baykin, T. Polyansky et al] // IEEE Conference Proceedings. – 2019. – Vol. 2019. – P. 92–96. DOI: 10.1109/ISPRAS47671.2019.00020
18. Plagne L. Parallel expression template for large vectors / L. Plagne, F. Hülsemann // Conference: Workshop on Parallel/High-Performance Object-Oriented Scientific. – 2009. [Electronic resource] – Access mode: https://www.researchgate.net/publication/228965935_Parallel_expression_template_for_large_vectors DOI: 10.1145/1595655.1595663
19. Schöberl J. C++11 Implementation of Finite Elements in NGSolve / J. Schöberl // ASC Report. – 2014. – No. 30. [Electronic resource] – Access mode: https://www.asc.tuwien.ac.at/~schoeberl/wiki/publications/ngs-cpp11.pdf
20. Rupp K. Increased efficiency in finite element computations through template metaprogramming / K. Rupp // Conference: Proceedings of the 2010 Spring Simulation Multiconference. – 2010. [Electronic resource] – Access mode: https://www.iue.tuwien.ac.at/pdf/ib_2010/CP2010_Rupp_2.pdf DOI: 10.1145/1878537.1878633
21. Cirak F. Generic programming techniques for parallelizing and extending procedural finite element programs / F. Cirak, J. C. Cummings // Engineering with Computers. – 2008. [Electronic resource] – Access mode: https://www.researchgate.net/publication/220677624_Generic_programming_techniques_for_parallelizing_and_extending_procedural_finite_element_programs DOI: 10.1007/s00366-007-0058-x
22. Bastian P. Generic implementation of finite element methods in the Distributed and Unified Numerics Environment (DUNE) / P. Bastian, F. Heimann, S. Marnach // Kybernetika. – 2010. – Vol. 46. P. 294–315.
23. The Simple FEM Solver. [Electronic resource] – Access mode: https://github.com/SeregaGomen/QFEM
24. Timoshenko S. Theory of plates and Shells / S. Timoshenko, S. Woinowsky-Krieger. – Singapore : McGraw-Hill, Inc., 1964. – 594 p.

Статья поступила в редакцию 15.02.2021.
После доработки 09.04.2021.

УДК 004.9:004.94:51

РЕАЛІЗАЦІЯ СКІНЧЕННО-ЕЛЕМЕНТНОЇ БІБЛІОТЕКИ КЛАСІВ З ВИКОРИСТАННЯМ УЗАГАЛЬНЕНОГО ПРОГРАМУВАННЯ

Чопоров С. В. – д-р техн. наук, професор, професор кафедри програмної інженерії Запорізького національного університету, Запоріжжя, Україна.

Ігнатченко М. С. – аспірант кафедри програмної інженерії Запорізького національного університету, Запоріжжя, Україна.

Кудін О. В. – канд. фіз.-мат. наук, доцент кафедри програмної інженерії Запорізького національного університету, Запоріжжя, Україна.

Кривохата А. Г. – канд. фіз.-мат. наук, старший викладач кафедри програмної інженерії Запорізького національного університету, Запоріжжя, Україна.

Гоменюк С. І. – д-р техн. наук, професор, декан математичного факультету, професор кафедри програмної інженерії Запорізького національного університету, Запоріжжя, Україна.

АНОТАЦІЯ

Актуальність. Для комп'ютерного моделювання складних об'єктів і явищ різної природи на практиці часто застосовують чисельний метод скінченних елементів. Його програмна реалізація (особливо для дослідження нових класів задач) є досить трудомістким процесом.

Висока вартість розробки програмного забезпечення обумовлює актуальність розробки нових підходів до підвищення ефективності програмування й супроводу (у т.ч. додавання нових функцій).

Мета. Мета роботи – створення нової ефективної архітектури програм скінченно-елементного аналізу проблем математичної фізики, що дозволяє легко розширювати їх функціональність для розв'язання нових класів задач.

Метод. Запропоновано метод розробки програм для скінченно-елементного аналізу з використанням узагальненого програмування, що дає можливість істотно спростити архітектуру програмного забезпечення й зробити його більш зручним для супроводу і модифікації за рахунок розділення алгоритмів і структур даних.

Запропоновано нову архітектура класів, які реалізують скінченно-елементний розрахунок, що дозволяє легко розширювати функціональність програм за рахунок додавання нових типів скінченних елементів, методів розв'язання систем лінійних алгебраїчних рівняння, паралельних обчислень тощо.

Результати. Запропонований підхід було програмно реалізовано у вигляді бібліотеки класів на мові C++. Проведено ряд обчислювальних експериментів, які підтвердили його працездатність при вирішенні низки практичних задач.

Висновки. Розроблений підхід може бути використаний як для створення систем скінченно-елементного аналізу загального призначення з відкритою архітектурою, так і для реалізації спеціалізованих програмних пакетів, орієнтованих на розв'язання конкретних класів задач (механіки руйнування, еластомерів, контактної взаємодії і т.д.).

КЛЮЧОВІ СЛОВА: метод скінченних елементів, скінченний елемент, об'єктно-орієнтоване програмування, узагальнене програмування, архітектура програмного забезпечення.

UDC 004.9:004.94:51

IMPLEMENTATION OF A FINITE ELEMENT CLASS LIBRARY USING GENERALIZED PROGRAMMING

Choporov S. V. – Dr. Sc., Professor, Professor of the Software Engineering Department, Zaporizhzhia National University, Zaporizhzhia, Ukraine.

Ihnatchenko M. S. – Postgraduate student of the Software Engineering Department, Zaporizhzhia National University, Zaporizhzhia, Ukraine.

Kudin O. V. – PhD, Associate Professor of the Software Engineering Department, Zaporizhzhia National University, Zaporizhzhia, Ukraine.

DOI 10.15588/1607-3274-2021-2-17

Kryvokhata A. G. – PhD, Senior Lecturer of the Software Engineering Department, Zaporizhzhia National University, Zaporizhzhia, Ukraine.

Homeniuk S. I. – Dr. Sc., Professor, Dean of the Faculty of Mathematics, Professor of the Software Engineering Department, Zaporizhzhia National University, Zaporizhzhia, Ukraine.

ABSTRACT

Context. For computer modeling of complex objects and phenomena of various nature, in practice, the numerical finite element method is often used. Its software implementation (especially for the study of new classes of problems) is a rather laborious process.

The high cost of software development makes the development of new approaches to improving the efficiency of programming and maintenance (including the addition of new functions) urgent.

Objective. The aim of the work is to create a new effective architecture of programs for finite element analysis of problems in mathematical physics, which makes it easy to expand their functionality to solve new classes of problems.

Method. A method for developing programs for finite element analysis using generalized programming is proposed, which makes it possible to significantly simplify the architecture of the software and make it more convenient for maintenance and modification by separating algorithms and data structures.

A new architecture of classes that implement finite element calculation is proposed, which makes it possible to easily expand the functionality of programs by adding new types of finite elements, methods for solving systems of linear algebraic equations, parallel computations, etc.

Results. The proposed approach was implemented in software as a class library in C ++. A number of computational experiments have been carried out, which have confirmed its efficiency in solving practical problems.

Conclusions. The developed approach can be used both to create general-purpose finite element analysis systems with an open architecture, and to implement specialized software packages focused on solving specific classes of problems (fracture mechanics, elastomers, contact interaction, etc.).

KEYWORDS: finite element method, finite element, object-oriented programming, generic programming, software architecture.

REFERENCES

1. Zienkiewicz O. C., Taylor R. L., Zhu J. Z. The Finite Element Method: Its Basis and Fundamentals. Sixth edition. Oxford, Butterworth-Heinemann, 2016, 753 p. DOI: 10.1002/nme.1760
2. Madenci E., Guven I. The Finite Element Method and Applications in Engineering Using ANSYS®. Boston, Springer, 2015, 651 p. DOI: 10.1007/978-1-4899-7550-8
3. Zimmerman W. Multiphysics Modeling with Finite Element Methods. Singapore, World Scientific, 2006, 432 p. DOI: 10.1142/6141
4. MSC Nastran – Multidisciplinary Structural Analysis. [Electronic resource]. Access mode: https://www.mscsoftware.com/product/msc-nastran
5. Top Finite Element Analysis (FEA) Software: List, Reviews, Comparison & Price | TEC. [Electronic resource]. Access mode: https://www3.technologyevaluation.com/sd/category/finite-element-analysis-fea
6. FreeFEM – An open-source PDE Solver using The Finite Element Method: [Electronic resource]. Access mode: https://freefem.org/
7. GetFEM Homepage – GetFEM. [Electronic resource]. Access mode: http://getfem.org/
8. OpenCAD The Programmers Solid 3D CAD Modeller. [Electronic resource]. Access mode: https://www.openscad.org/
9. The dial.II Finite Element Library. [Electronic resource]. Access mode: https://www.dealii.org/
10. Netgen/NGSolve. [Electronic resource]. Access mode: https://ngsolve.org/
11. Stepanov A. A., Rose D. E. From Mathematics to Generic Programming. Crawfordsville, Addison-Wesley, 2014, 292 p.
12. Meyers S. Effective Modern C++: 42 Specific Ways to Improve Your Use of C++11 and C++14 / S. Sebastopol, O'Reilly, 2017, 316 p.
13. Galowicz J. C++17 STL Cookbook. Birmingham, Packt Publishing Ltd, 2017, 504 p.
14. Vandevoorde D., Josuttis N. M., Gregor D. C++ Templates: The Complete Guide. Boston, Addison-Wesley, 2017, 788 p.
15. Bittencourt M. L. Using C++ templates to implement finite element classes, *Engineering Computation*, 2000, Vol. 17(7), pp. 775–788. DOI: 10.1108/02644400010352243
16. Bangerth W. Using Modern Features of C++ for Adaptive Finite Element Methods: Dimension-Independent Programming in deal.II, *Proceedings of the 16th IMACS World Congress*, 2000, [Electronic resource], Access mode: https://www.dealii.org/7.3.0/
17. Gurin A. , Baykin A., Polyansky T. et al. FEMEngine: finite element method C++ code based on functional and template metaprogramming, *IEEE Conference Proceedings*, 2019, Vol. 2019, pp. 92–96. DOI: 10.1109/ISPRAS47671.2019.00020
18. Plagne L., Hülsemann F. Parallel expression template for large vectors, *Conference: Workshop on Parallel/High-Performance Object-Oriented Scientific*, 2009, [Electronic resource], Access mode: https://www.researchgate.net/publication/228965935_Parallel_expression_template_for_large_vectors DOI: 10.1145/1595655.1595663
19. Schöberl J. C++11 Implementation of Finite Elements in NGSolve, *ASC Report,* 2014, No. 30. [Electronic resource]. Access mode: https://www.asc.tuwien.ac.at/~schoeberl/wiki/publications/ngs-cpp11.pdf
20. Rupp K. Increased efficiency in finite element computations through template metaprogramming, *Conference: Proceedings of the 2010 Spring Simulation Multiconference*. 2010, [Electronic resource]. Access mode: https://www.iue.tuwien.ac.at/pdf/ib_2010/CP2010_Rupp_2.pdf DOI: 10.1145/1878537.1878633
21. Cirak F., Cummings J. C. Generic programming techniques for parallelizing and extending procedural finite element programs, *Engineering with Computers,* 2008. [Electronic resource]. Access mode: https://www.researchgate.net/publication/220677624_Generic_programming_techniques_for_parallelizing_and_extending_procedural_finite_element_programs DOI: 10.1007/s00366-007-0058-x
22. Bastian P., Heimann F., Marnach S. Generic implementation of finite element methods in the Distributed and Unified Numerics Environment (DUNE), *Kybernetika*, 2010, Vol. 46, pp. 294–315.
23. The Simple FEM Solver. [Electronic resource]. Access mode: https://github.com/SeregaGomen/QFEM
24. Timoshenko S., Woinowsky-Krieger S. Theory of plates and Shells. Singapore, McGraw-Hill, Inc., 1964, 594 p.

DOI 10.15588/1607-3274-2021-2-17