ПРИМЕНЕНИЕ МЕТОДА КОНЕЧНЫХ ЭЛЕМЕНТОВ ДЛЯ РЕШЕНИЯ ПЛОСКОЙ ЗАДАЧИ ТЕОРИИ УПРУГОСТИ

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Васильев А.Е.

Жвакина А. В. – кандидат технических наук, доцент

В настоящее время, в виду постоянно растущей численности населения, как никогда ранее необходимо современное и ресурсоэффективное строительство. Такое строительство невозможно без точных компьютерных расчетов, которые поднимают проектную работу на новый уровень, повышая скорость и качество решения многих сложных инженерных задач, многие из которых ранее рассматривались лишь упрощенно. В разработанном программном продукте в основе расчетов использован метод конечных элементов.

Метод конечных элементов (МКЭ) это численный метод для решения задач математической физики и инженерии. Первоначально МКЭ появился в строительной механике на заре пятидесятых годов двадцатого века, но в последующее десятилетие метод получил широкое распространение, развился и начал применяться не только для расчета конструкций, но и для решения широкого ряда других инженерных задач.

Типичный расчет происходит в три этапа:

1) Разделение области, в которой ищется решение, на конечные элементы.
2) Определение аппроксимирующей функции каждого элемента. Коэффициенты аппроксимирующих функций могут быть выражены через вектор узловых значений искомой функции (при условии непрерывности в узлах).
3) Объединение уравнений в глобальную систему уравнений (так называемый ансамбль) для финального расчета.

Наряду с МКЭ были рассмотрены другие численные методы решения дифференциальных уравнений. В частности, проанализирован метод конечных разностей (МКР). В результата анализа выявлены преимущества МКЭ перед МКР:

1) Способность работать с областями с более сложной геометрией.
2) Решением является функция, где для каждой точки можно сразу вычислить значение.

Реализация МКЭ для решения плоской задачи выполнена на языке программирования C++, была использована библиотека Eigen (для простоты математических расчетов).

Объект изучения разделяется на большое число элементов, в нашем конкретном случае это будут треугольники. Иначе говоря, непрерывная среда заменяется на множество конечных элементов и узлов, образующих сетку. Перечень элементов вместе с описанием свойств материала, приложенных внешних сил и закреплений образует входные параметры нашей программы. Примем что объект изготовлен из стали с коэффициентом Пуассона v = 0.3, модулем Юнга E = 2000 МПа, этого достаточно для построения матрицы упругости.

После чтения входных данных производим расчет глобальной матрицы жесткости по формуле:

где K - матрица жесткости; δ - вектор перемещений; R - вектор внешних нагрузок. Сама глобальная матрица жесткости строится как комбинация матриц жесткости отдельных элеменов, полученных из формул сопротивления материалов.

Важным этапом является задание закреплений, ведь из механики следует, что, система без закреплений способна перемещаться в любых направлениях. Математически отсутствие закреплений приводит к отсутствию единственного решения у полученной системы линейных алгебраических уравнений (СЛАУ).

Сравнивалось несколько способов решения СЛАУ. В результате в программе использованы возможности библиотеки Eigen, а конкретно – SimplicalLDLT, который решает систему прямым методом. Эффективным решением так же может являться использование нейронных сетей для минимизации среднеквадратичной ошибки, этот подход характеризуется быстродействием за счет распараллеливания вычислений.

После вычисления общего вектора перемещений, определяются векторы перемещений для каждого отдельного конечного элемента, затем выполняется переход от перемещений к деформациям, и далее к напряжениям. После чего выполняется визуализация полученных напряжений в образце/конструкции, проводится дальнейший анализ и дается оценка применимости образца на конкретном реальном проекте, а также выполняются остальные необходимые действия.

На данный момент все программные комплексы для расчета строительных конструкций, существующие на рынке являются зарубежными разработками, причем очень дорогими, требовательными к ресурсам ЭВМ. Разработанная программа для решения плоской задачи позволяет повысить эффективность строительства за счет повышения скорости и качества необходимых инженерных расчетов.

Исследованиеподдержанопроектом CERES. Centers of Excellence for young RESearchers (Reg.no. 544137-TEMPUS-1-2013-SK-JPHES),

Список использованных источников:
1. Daryl L. Logan. A first course in the finite element method – Cengage Learning, 2011. – 752 с.
2. О.Зенкевич. Метод конечных элементов в технике – Москва, 1975. – 541 с.
3. Метод конечных элементов [Электронный ресурс], https://ru.wikipedia.org/wiki/Метод_конечных_элементов

ПОЛНОТЕКСТОВЫЙ ПОИСК В ANDROID ПРИЛОЖЕНИЯХ

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Вашило К. Р.

Хотеев А.Л. – канд. физ.-мат. наук, доцент

Рассмотрены ключевые аспекты организации полнотекстового поиска в приложениях ОС Android. Приведены результаты сравнения поиска в обычных таблицах SQLite и таблицах с настроенной поддержкой полнотекстового поиска.

Как только в мобильном приложении приходится хранить достаточно много информации, возникает проблема скорости поиска и доступа к интересующим пользователя данным. Типы приложений при этом могут быть совершенно различными, следует перечислить основные источники больших объемов текстовой информации: сообщения и переписка, текстовые документы, справочные материалы, книги. Следует учесть, что далее речь идет о поиске по документам, находящимся именно на телефоне (носимом устройстве Android) пользователя, где объем данных ограничен, как и мощности существующих инструментов. Если требуется работа на больших объѐмах данных и сложных запросах (единицы и десятки Гб текста одного документа/записи), следует реализовать собственную поисковую систему на стороне сервера.

Первые реализации полнотекстового поиска сканировали содержимое всех документов для нахождения заданной для поиска фразы. При этом время поиска было неприемлемо велико и зависело от размера базы данных. Следующим шагом было создание полнотекстового индекса - словаря, в котором перечислены все слова или термы текста и места, где они находятся в тексте.

В качестве базы данных на OS Android используется SQLite. Модули FTS3 и FTS4 расширяют SQLite и делают возможным создание специальных виртуальных таблиц, поддерживающих полнотекстовый индекс. Модули FTS1, FTS2 считаются устаревшими, и из-за известных проблем их использование нежелательно.

Сравнение использования FTS и запроса SQL. Добавим набор данных "Энроновская коллекцияe-mail— из более пятиста тысяч документов в таблицу FTS и обычную таблицу SQLite.

```
CREATE TABLE data1(body TEXT);
CREATE VIRTUAL TABLE data2 USING fts3(body TEXT);
```

Листинг 1 – Пример создания обычной таблицы SQLite и аналогичной таблицы FTS

Проанализируем результаты работы приведенных запросов к базе данных:

```
SELECT count(*) FROM data1 WHERE body MATCH 'android'; /* результат 22.5 секунды */
SELECT count(*) FROM data2 WHERE body LIKE '%android%'; /* результат 0.03 секунды
*/
```

Листинг 2 – Пример выполнения запросов обычной таблице SQLite и в таблице FTS

Оба поисковых запроса являются регистронезависимыми. На заполнение данными таблицы FTS3 ушло чуть больше времени: 30 и 25 минут. Однако следует отменить, что приведенные запросы не являются эквивалентными. Перечислим основные различия работы запросов:

1) Результаты запроса LIKE могут включать в себя слова "androidphobe" или —FlyAndroid‖. Запрос MATCH отдает результатом только строки, содержащие "android" как отдельное вхождение.

2) Для хранения таблицы FTS3 понадобилось на треть больше дискового пространства (~2 Гб).

С точки зрения разработчика таблицы FTS в целом схожи с обычными таблицами SQLite. К ним применимы те же команды INSERT, UPDATE, DELETE для добавления, редактирования и удаления данных. Для запроса данных используется SELECT. Чтобы сделать запрос, задействуя встроенный полнотекстовый индекс, используется оператор MATCH.

Различия между FTS3 и FTS4. FTS4 является усовершенствованным FTS3. FTS4 не поддерживается в