МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования

«ТЮМЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ

Кафедра прикладной и технической физики

|  |  |
| --- | --- |
|  | Заведующий кафедрой  канд. техн. наук  Б.В. Григорьев |

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

бакалавра

ТЕМА РАБОТЫ

Разработка программного комплекса для оптимизации проектных решений по разработке нефтяного месторождения

16.03.01 Техническая физика

|  |  |  |
| --- | --- | --- |
| Выполнил (а) работу  студент 4 курса  очной формы обучения | Колбеко Александр Борисович | |
|  |  |  |
| Руководитель  *Канд. Техн. Наук* | Гильмиев Денис Рустамович | |

Тюмень

2020 год

АННОТАЦИЯ

К выпускной квалификационной работе на тему "Разработка программного комплекса для оптимизации проектных решений по разработке нефтяного месторождения".

Выполнил: студент группы 39ТФ162 Колбеко Александр Борисович.

Научный руководитель: кандидат физико-технических наук, старший преподаватель кафедры прикладной и технической физики Гильмиев Денис Рустамович.

Выпускная квалификационная работа содержит 48 страниц, 7 рисунков, приложения с листингом программного решения.

Ключевые слова: имитационное моделирование, двухфазная фильтрация, Python, уравнение материального баланса, NumPy, объектно-ориентированное программирование.

Объект исследования: залежь углеводородов.

Предмет исследования: изучение, моделирование процесса двухфазной фильтрации, разработка имитационной модели, включающей в себя генератор вариантов разработки, сопряженный с последующей экономической оценкой.

Результаты работы: реализована имитационная модель на языке программирования Python, произведено сравнение получаемых результатов с аналитическим решение Баклея-Леверетта.

Практическая значимость: Разработанный метод и имитационная модель доказали свою работоспособность и могут быть использованы для дальнейших исследований.

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 4](#_Toc41929015)

[1 ОБЗОР И АНАЛИЗ ЛИТЕРАТУРЫ 7](#_Toc41929016)

[1.1 ОБЗОР СУЩЕСТВУЮЩИХ РЕШЕНИЙ 7](#_Toc41929017)

[1.2 ЦЕЛИ И ЗАДАЧИ РАБОТЫ 9](#_Toc41929018)

[1.3 ФУНКЦИЯ И ЗАДАЧА БАКЛЕЯ-ЛЕВЕРЕТТА 9](#_Toc41929019)

[2 МЕТОД МАТЕРИАЛЬНОГО БАЛАНСА ПРИ АНАЛИЗЕ И ПРОГНОЗИРОВАНИИ ПОКАЗАТЕЛЕЙ СИСТЕМ РАЗРАБОТКИ 13](#_Toc41929020)

[2.1 ДИСКРЕТИЗАЦИЯ 15](#_Toc41929021)

[2.2 СОПОСТАВЛЕНИЕ ФРОНТА ВЫТЕСНЕНИЯ 19](#_Toc41929022)

[3 ОБЗОР ПРОГРАММНОЙ РЕАЛИЗАЦИИ 21](#_Toc41929023)

[3.1 СХЕМА РАБОТЫ ПРОЕКТА 21](#_Toc41929024)

[3.2 ОСНОВНЫЕ КОМПОНЕНТНЫ МОДЕЛИ 24](#_Toc41929025)

[3.3 АНАЛИЗ ПОЛУЧАЕМЫХ РЕЗУЛЬТАТОВ 26](#_Toc41929026)

[4 ВЫВОДЫ ПО РАБОТЕ 29](#_Toc41929027)

# ВВЕДЕНИЕ

Нефтегазодобыча – это огромная промышленная отрасль, история которой насчитывает уже более полутора веков. В этой сфере вращаются миллиарды долларов и работают сотни тысяч людей. Экономика Российской Федерации и Тюменской области в частности во многом зависит от добычи полезных ископаемых, коими изобилует наша земля. Первое упоминание о разведке нефтяных месторождений в Западной Сибири датируется 1903 годом, но, не смотря на довольно солидный возраст этой сферы деятельности, нефтедобывающие компании сравнительно недавно стали прибегать к наукоемким методам исследования и разработки. И до сих пор никто не может с точностью сказать, что именно происходит на глубине нескольких километров под землей. Тем не менее, проводится ряд исследований: самого геологического пласта, пластовых флюидов. Строится гидродинамическая модель. Эта модель строится, опираясь на законы подземной гидромеханики, которая служит основой для разработки нефтяных и газовых месторождений.

Подземная гидромеханика – это наука, изучающая особый вид движения жидкостей, газов и их смеси в пористой среде горных пород. Этот вид движения называется фильтрацией. Фильтрационное движение флюидов имеет ряд особенностей, часть которых очень непросто бывает описать математически, и еще сложнее предать результатам физический смысл, так как законы, на которые мы опираемся, имеют в большинстве своем эмпирическое обоснование, в частности – закон Дарси и другие.

Пластовое давление является одной из важнейших характеристик залежи углеводородных ресурсов, определяющей энергетическую составляющую пласта. В процессе эксплуатации месторождения огромная роль отводится на достоверное и оперативное прогнозирование текущего пластового давления в областях работы действующего фонда скважин. Без этой информации невозможно спрогнозировать дебеты скважин в сколько-то отдаленном будущем, а без достоверной информации о продуктивности работы действующего фонда не представляется возможным формирование рентабельного плана разработки месторождения УВ.

Пластовое давление формируется за счет гидростатического давления, расширения или сжатия флюидов, изменения их массы, а также изменения объема порового или трещиноватого пространства. Различают начальное (до вскрытия подземного резервуара или не нарушенное техногенными процессами) и текущее (динамическое) пластовое давления. От текущего давления очень сильно зависят физико-химические параметры флюидов, насыщающих пласт. Они представляют собой сложные многокомпонентные смеси из нефти, газа, воды и конденсата, находящиеся в равновесии, очень чувствительные к условиям, в которых находятся. В зависимости от газового фактора (отношения количества газа, растворенного в нефти в кубических метрах приведенного к стандартным условиям, к количеству этой же нефти, выраженной в кубических метрах или тоннах приведенной к стандартным условиям), флюид имеет определенное предельное значения давления, при котором газ начнет выделяться из нефти, данный процесс сопровождается увеличением вязкости нефти и формированием кластеров пузырьков газа, закупоривающих поровое пространство и препятствующих движению нефти к забою скважины. Именно поэтому такая большая роль отведена мониторингу текущего состояния разработки месторождений, организации системы поддержания пластового давления.

Современный подход к моделированию сопряжен с множеством трудностей. На данный момент процесс создания вариантов разработки не автоматизирован, требует большого количества времени профильных специалистов. Так же, ввиду уточнения информации относительно объекта разработки, имеет место изменение карт ННТ, пористостей и проницаемостей, вследствие чего приходится изменять ранее сформированные сетки скважин и производить повторные расчеты уровней добычи. Так же для Российских нефтегазовых кампаний достаточно чувствительным фактором является не стабильная политическая ситуация на мировой арене, в рамках которой используемое программное обеспечение может подвергаться санкциям, ввиду чего может быть потерян доступ к его дальнейшему использованию, что, в свою очередь, может парализовать на какое-то время существующие бизнес процессы и стать причиной убытков. Уже сейчас нефтегазовые кампании переходят на собственные реализации гидродинамических симуляторов, например, "Техсхема" Сургутнефтегаза или "РН-КИМ" Роснефти, дабы минимизировать связанные с санкциями риски.

Исходя из вышесказанного, можно заключить, что использование полноценных коммерческих гидродинамических симуляторов, представленных на рынке (tNavigator, Tempest, Eclipse), для оценки всех вариантов разработки является невозможным ввиду их дороговизны, долгого времени расчета и ограниченности числа лицензий.

# 1 ОБЗОР И АНАЛИЗ ЛИТЕРАТУРЫ

## 1.1 ОБЗОР СУЩЕСТВУЮЩИХ РЕШЕНИЙ

В рамках выполнения дипломной работы мной были изучены технические описания таких коммерческих гидродинамических симуляторов, как Tempest от компании Roxar, tNavigator от компании Rock Flow Dynamics и Eclipse от Schlumberger. Они представляют собой пакеты программного обеспечения для моделирования процесса фильтрации. Для построения модели каждый из них требует следующие входные данные:

1. Координаты устьев скважин, альтитуды, инклинометрия, эти данные используются для создания траекторий скважин в модели.
2. Координаты пластопересечений.
3. Стратиграфические разбивки, рассчитанные геологом в проекте – используются в качестве основы при формировании структурного каркаса гидродинамической модели.
4. Кривые ГИС – используются для корреляционных построений, выделения литотипов, оценки характера насыщения и ФЕС, фациального анализа, привязки данных сейсморазведки. Результаты интерпретации ГИС (РИГИС) используются при построении 3D модели для распространения свойств – построения кубов фильтрационно-емкостных свойств (ФЕС).
5. Отбивки флюидных контактов в скважинах – используются для построения карт флюидных контактов и геометризации залежей. Интервалы перфорации, результаты испытаний и работы скважин, гидродинамического каротажа используются для обоснования и корректировки положения флюидных контактов.
6. Даты бурения и ввода скважин в добычу (под закачку), карты накопленных отборов и закачки – используются при отборе скважин с неискаженным влиянием разработки величинами начальной насыщенности.
7. Сейсмические данные. Структурные карты и поверхности нарушений по данным сейсморазведки, бурения и других методов используются для формирования структурного каркаса. Карты или кубы сейсмических атрибутов используются для распространения ФЕС в межскважинном пространстве.
8. Уравнения петрофизических зависимостей, средние и граничные значения свойства коллектора, кривые капиллярного давления.
9. Количественные (определения Кп, Кпр, Кв) и качественные (описания) исследования керна. Применяются при настройке данных ГИС для последующей массовой интерпретации, а также при создании концептуальной модели.
10. Общие и геологические данные, к ним можно отнести 2D карты эффективных толщин и нефтенасыщенных толщин, данные отчета запасов и прочее.

Каждый рассмотренный симулятор использует наиболее распространенную модель фильтрации трехфазной системы, черная нефть. Реализованная версия модели позволяет использовать при расчете двойные пористость-проницаемость, неструктурированные сетки, разломы, выклинивания, закачку полимеров и щелочей, ГРП, гистерезис, диффузия, адсорбцию, десорбцию, разные виды аквифера, трассерные исследования, а также поддержку расчетов на нескольких графических ускорителях или кластере.

На этапе ознакомления с техническими описаниями и руководствами пользователя стало понятно, что пытаться создать в одиночку подобный симулятор в столько короткие сроки практически невозможно, поэтому было решено идти по альтернативному, и, надо сказать, достаточно популярному пути – создание прокси модели.

## 1.2 ЦЕЛИ И ЗАДАЧИ РАБОТЫ

Целью данной работы является создание имитационной модели, включающей в себя гидродинамический симулятор, генератор и оценщик вариантов разработки. Подобный подход к проектированию разработки не был реализован ни в одном из известных мне коммерческих программных пакетов, и, на мой взгляд, при должно уровне реализации, позволит изменить и оптимизировать существующие бизнес процессы в нефтегазовых кампаниях.

В рамках работы были выделены следующие задачи:

* Получение математической модели двумерной двухфазной фильтрации на основе уравнения материального баланса.
* Написание гидродинамического симулятора на основе полученной математической модели.
* Проектирование препроцессора, выполняющего роль генератора варианта разработки.
* Проектирование постпроцессора, выполняющего экономическую оценку варианта разработки на основании данных, полученных из гидродинамической модели.

## 1.3 ФУНКЦИЯ И ЗАДАЧА БАКЛЕЯ-ЛЕВЕРЕТТА

Говоря об задаче двухфазной фильтрации, нельзя не упомянуть о классической в теории вытеснения модели Баклея-Леверетта. В рамках данной модели принято рассматривать некоторый участок порового пространства коллектора, заполненного двумя жидкостями с четкой границей раздела. Прежде, чем продолжить, нам следует ввести некоторые условные обозначения:

* Р-давление;
* , - вязкость воды и нефти соответственно;
* ,-
* - относительная проницаемость;

, ,,=сonst;

=; =;

* Индексы «О»-нефть (Oil); «w»-вода (Water);

«абс»-абсолютное значение; «отн»-относительная величина.

* , - насыщенность соответствующей фазы

+ ,

* t- время;
* ,- ;
* *U*  - скорость фильтрации системы ;
* f(S) - функция Баклея-Леверетта ;
* m- пористость коллектора;

Физическое описание строится на законе Дарси. Запишем закон для обеих фаз:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

Почленно разделим (1) на (2) и получим:

|  |  |  |
| --- | --- | --- |
|  |  |  |

Далее, запишем уравнение неразрывности для каждой фазы:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

Сложим (4) и (5):

|  |  |  |
| --- | --- | --- |
|  |  |  |

С учетом того, что + , имеем:

|  |  |  |
| --- | --- | --- |
|  |  |  |

= const = U - скорость фильтрации жидкости. Запишем закон Дарси для жидкости:

|  |  |  |
| --- | --- | --- |
|  |  |  |

Выразим градиент давления:

|  |  |  |
| --- | --- | --- |
|  |  |  |

Теперь подставим полученное выражение (9) в (1), получим:

|  |  |  |
| --- | --- | --- |
|  |  |  |

Отношение принято обозначать как f(s) – функция Баклея-Леверетта. Её физический смысл заключается в следующем: она равна объемной доли воды в суммарном потоке фаз. Типичный вид функции и её производной изображены на рисунке 1.

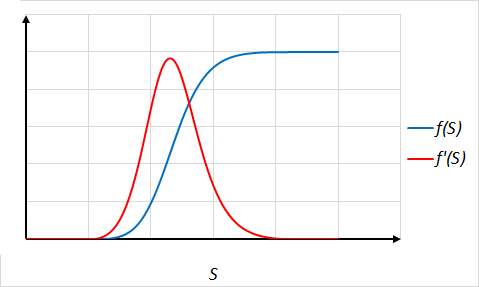


Рис. 1. Функции Баклея-Леверетта и ее производной

Подставим выражение (10) в (4) и получим уравнения Баклея-Леверетта:

|  |  |  |
| --- | --- | --- |
|  |  |  |

Для решения задачи Баклея-Левертта нам необходимо знать вид функции Баклея-Леверетта f(S), которая в свою очередь, как было показано выше, выражается через фазовые проницаемости образца для воды kв(S) и нефти kн(S). Такие зависимости находят эмпирически. В ходе эксперимента можно получить совокупность точек, которые будут показывать значения относительных фазовых проницаемостей для определенных значений водонасыщенности.

Зная функцию Баклея-Леверетта можно решить саму задачу и в результате получить функциональную зависимость насыщения от координат и времени S(x, t).

# 2 МЕТОД МАТЕРИАЛЬНОГО БАЛАНСА ПРИ АНАЛИЗЕ И ПРОГНОЗИРОВАНИИ ПОКАЗАТЕЛЕЙ СИСТЕМ РАЗРАБОТКИ

Уравнение материального баланса базируется на законе сохранения массы. На практике оно может быть применено для определения режима разработки, оценке запасов, прогнозирования технологических параметров в процессе эксплуатации месторождения. Именно оно легло в основу симулятора, реализованного в работе, для решения задачи двумерной двухфазной фильтрации.

Залежь представляется совокупностью гидродинамически связанных блоков. Для каждого из блоков решается уравнение материального баланса, получаемое из следующей системы:

|  |  |  |
| --- | --- | --- |
|  |  |  |

Условные обозначения:

* Объемный коэффициент газа Bg
* Объемный коэффициент нефти Bo
* Объемный коэффициент системы Bt
* Объемный коэффициент воды Bw
* Сжимаемость пор cf и воды cw
* Начальный объем газовой шапки G
* Накопленная добыча газа Gp, растворенного газа Gps, газа из газовой шапки Gpc
* Средняя насыщенность связанной воды Swc
* Накопленный приток из законтурной области We, накопленная добыча/закачка i-й фазы Qi/Winj
* Начальные балансовые запасы нефти N, накопленная добыча нефти Np
* Среднее пластовое давление p, начальное пластовое давление pl
* Накопленный газонефтяной фактор Rp
* Газосодержание R.

В рамках задачи рассматривается синтетическая залежь, не содержащая газ. Тогда в уравнении материального баланса не будет второго слагаемого слева от равно (расширение газовой шапки) и слагаемого третьего справа от равно (объем свободного газа в пласте). Тогда уравнение примет вид:

|  |  |  |
| --- | --- | --- |
|  |  |  |

Произведем группировку слагаемых относительно Δp:

|  |  |  |
| --- | --- | --- |
|  |  |  |

Для удобства записи произведем замену на , параметр сжимаемости системы, получим:

|  |  |  |
| --- | --- | --- |
|  |  |  |

Где Δp – разница начального и текущего давления в j блоке. Влияние аквифера учитывается с помощью водоносного пласта Фетковича, который представляет собой дополнительные расчетные ячейки, соединенные со всеми приграничными ячейками. Слагаемое Qi представляет собой сумму накопленных перетоков фазы по доступным направлениям для блока, в соответствии с законом Дарси.

## 2.1 ДИСКРЕТИЗАЦИЯ

Т.к. блоки гидродинамически связаны и в начальный момент времени пластовое давление везде одинаково, то система находится в равновесии:

|  |  |  |
| --- | --- | --- |
|  |  |  |

Под подразумевается выражение (15), полученной в прошлом разделе. В него скрыто входят давления по четырем соседних ячейкам (т.к. имеет общая граница, через которую осуществляется фильтрация в соответствии с законом дарси, слагаемое Qi) и, непосредственно, давление в самой ячейке, т.е. имеем дело с функцией нескольких переменных. В данной записи n – временной индекс, k-номер блока, приближение давления из которого берется.

Вследствие производственной деятельности в k-м блоке происходит отбор массы, из-за чего начинает убывать пластовое давление. Ввиду появления разницы давлений между блоками начинается фильтрация жидкости, подчиняющаяся закону Дарси, также вызывающая в них изменение давления. Для решения данной системы используется многомерный метод секущих, общая форма записи которого имеет вид:

|  |  |  |
| --- | --- | --- |
|  |  |  |

Перенесем первое слагаемое в правую часть уравнения, заменяя производные конечными разностями назад, получим:

|  |  |  |
| --- | --- | --- |
|  |  |  |

Также, для большей лаконичности и краткости записи произведем следующую замену:

|  |  |  |
| --- | --- | --- |
|  |  |  |

Где индекс к показывает, для какой ячейки считается уравнение материального баланса, а j ячейку, по изменению давления в которой берется производная. Раскроем скобки, совершив замену:

|  |  |  |
| --- | --- | --- |
|  |  |  |

Оставим слагаемые с в левой части уравнения, а направо относительно знака равенства, после проделанных действий получим:

|  |  |  |
| --- | --- | --- |
|  |  |  |

Заменим свободный член уравнения в правой части на :

|  |  |  |
| --- | --- | --- |
|  |  |  |

Таким образом, мы пришли к матрице вида . Коэффициенты у давлений, которые относятся к ячейкам, не имеющих общих граней с k-й ячейкой равны 0, т.к. функция материального баланса относительно них является константой.

Полученные корни переписываются в , а прошлые в . На одной из итераций разность становится меньше заранее заданной delta, происходит фиксация технологических параметров работы скважин и потоков жидкости по ячейкам.

Далее, определяется фазовый состав потока жидкости. Для этого в рамках моей работы используется функция Баклея-Леверетта. Для его использования требуются аналитические зависимости относительных фазовых проницаемостей компонент жидкости представлены на рисунке 2.

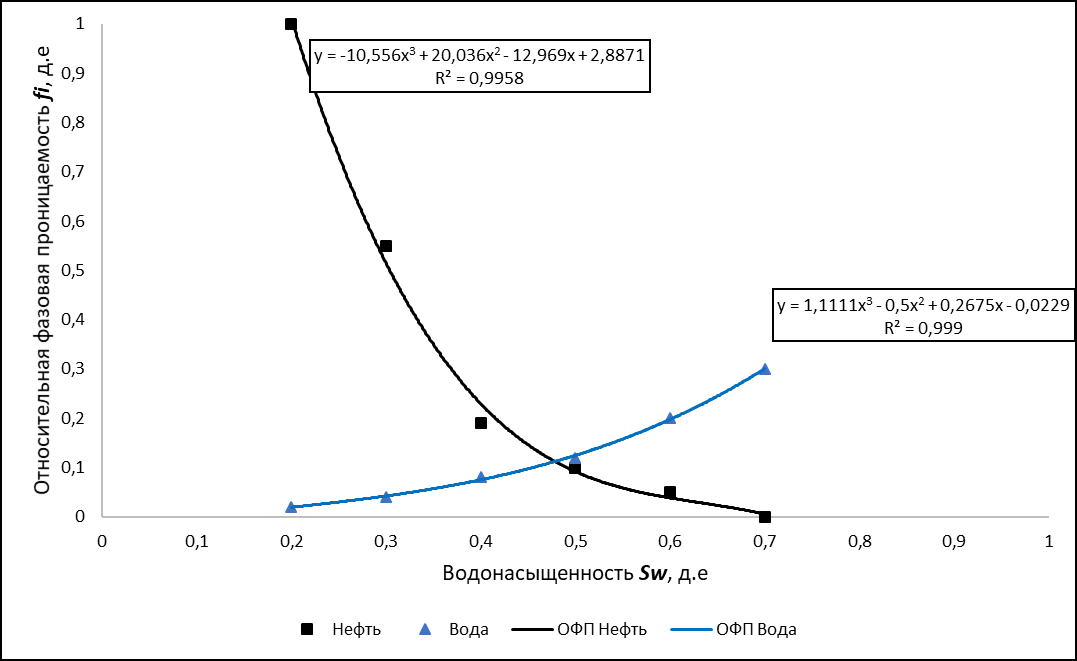


Рис. 2. Используемые ОФП

Альтернативно прямому заданию относительных фазовых проницаемостей с использование данных реального образца, так же можно было использовать одну из известных аппроксимаций, позволяющих связать значение насыщения и фазовой проницаемости.

После получения устойчивого решения по жидкости, в соответствии с функцией Баклея-Леверетта ( ), определяется фазовый состав потока между блоками. Значения проницаемостей вычисляются в соответствии с аналитическими зависимостями относительных фазовых проницаемостей, против потока.

## 2.2 СОПОСТАВЛЕНИЕ ФРОНТА ВЫТЕСНЕНИЯ

Для проверки реализованного алгоритма проводилась серия тестов, целью которых было получение фронтов вытеснения, которые в дальнейшем сопоставлялись с решением одномерной задачи Баклея-Леверетта.

Будем рассматривать вытеснение в прямолинейном потоке в горизонтальном образце, который представляет собой однородную изотропную среду. То есть будем считать, что пористость m и абсолютная проницаемость k являются постоянными величинами. Так же сделаем следующие приближения.

1. Поперечное сечение образца W мало настолько, что можно считать давление P и водонасыщенность S неизменяющимися по сечению.

2. Давление в водяной и нефтяной фазах одинаковы, то есть полагаем, что капиллярное давление пренебрежимо мало.

3. Фазы являются несжимаемыми, несмешивающимися.

4. Температура образца и флюида при вытеснении не изменяется (изотермическое приближение).

Первоначально образец полностью насыщен нефтяной фазой, далее через один конец образца x=0 закачивается вода, нефть и вода начинают свое движение, происходит образование водонефтяного фронта – зоны совместного движения флюидов.

Решение задачи Баклея-Леверетта методом Крупнова заключается в использовании семейства разносных дивергентных схем:

|  |  |  |
| --- | --- | --- |
|  |  |  |

С необходимыми условиями для сходимости:

|  |  |  |
| --- | --- | --- |
|  |  |  |

Используя рекомендуемые Ю.П. Красновым значения *β’*=0 и 0,925 ≤ *β* ≤ 0,975, соотношение (23) примет упрощенный вид:

|  |  |  |
| --- | --- | --- |
|  |  |  |

В рамках сопоставления инициализировалась расчетная область, представляющая собой ряд из ячеек, обладающих начальным насыщением нефти. Параметры сессии:

* l – ребро ячейки 0,5 метров;
* , - вязкость воды и нефти, соответственно 1 и 10 сП;
* Q – объемный расход жидкости 0,000015 м3/сек;
* ʋ - скорость фильтрации 0,00006 м/сек;
* S – площадь фильтрации 0,25 м2;
* t1, t2, t3 – время расчета 33, 47, 100 часов соответственно;

Результаты сопоставлении представлены на рисунке 3.

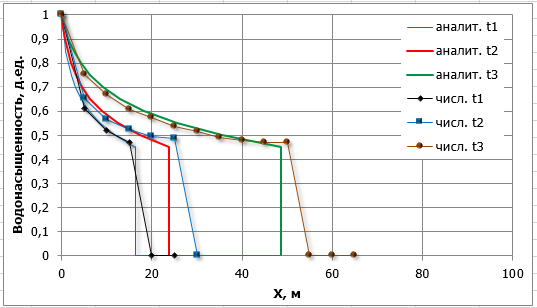


Рис. 3. Сопоставление фронта вытеснения

Отклонение пространственной координаты между аналитическим и численным решением обусловлено параметры численной сетки. Относительная погрешность значения насыщения водой колеблется от 3 до 7,5%, что позволяет говорить об удовлетворительной точности получаемого решения.

Т.к. результаты численного одномерного решения соотносятся с аналитическим решением задачи Баклея-Леверетта, то реализованный алгоритм также применим и к двумерной постановке задачи.

# 3 ОБЗОР ПРОГРАММНОЙ РЕАЛИЗАЦИИ

Проект разработан в объектно-ориентированном стиле, содержит 7 управляющих процедур, 38 функций. Присутствует реализация таких паттернов, как factory, observer, singleton. В рамках проекта так же были использованы open source библиотеки Numpy, Scipy и matplotpib.

## 3.1 СХЕМА РАБОТЫ ПРОЕКТА

Проект представляет собой совокупность взаимосвязанных процедур, объединенных одной целью – нахождение наиболее оптимального варианта разработки залежи углеводородов. На этапе проектирования было решено придерживаться объектно-ориентированного стиля программирования, разделять процедуры и классы по группа решаемых ими задач. Подобный подход в очередной раз доказал свою жизнеспособность на этапе разработки, т.к. в ходе работы несколько раз изменялась математическая модель, ввиду чего вносились значительные изменения в теле программы. Было значительно проще заниматься модернизацией отдельных объектов и их вызовов, чем если бы весь проект был выполнен в процедурном стиле.

Можно выделить 3 основных процесса:

* генерация и последующая оценка варианта разработки;
* инициализации расчетной области с объектами, заданными генератором;
* моделирование фильтрации;

К нереализованному функционалу следует отнести выделение в отдельных блок настройщик свойств области. На данный момент свойства коллектора, начальные насыщения просто задаются конфигурацией выделенных переменных в том же файле, где и находится объектная модель расчетной области.

Для лучшего понимания отдельных элементов алгоритма, обратимся к его упрощенному графическому представлению, рисунку 4.

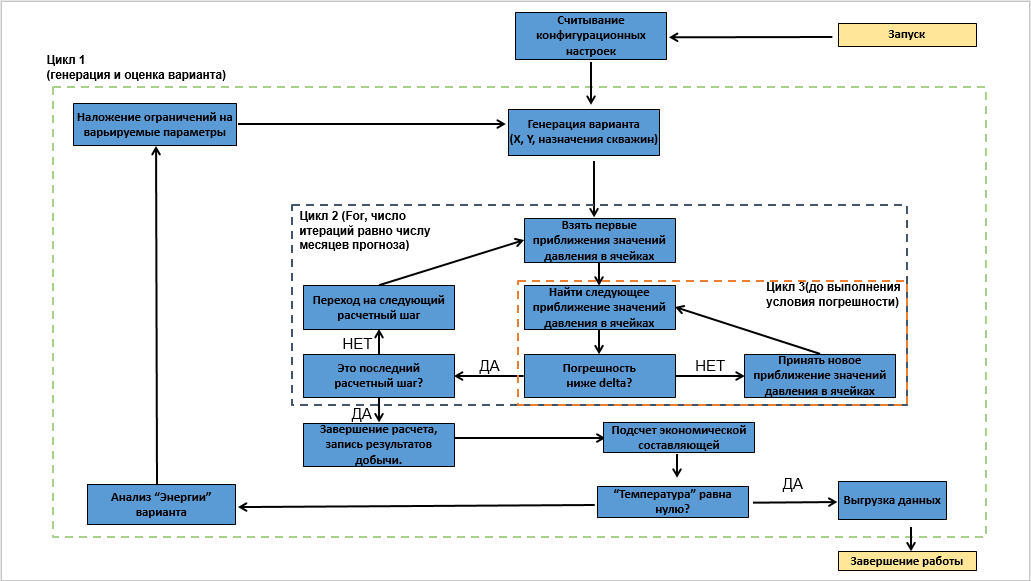


Рис. 4. Упрощенная схема проекта

В конфигурационном файле указываются целевые коэффициенты охвата скважинами, цены на капитальные вложения при разработке, временной период на прогноз, стоимость продажи тонны сырья.

Также на этапе подготовки необходимо указать параметры залежи и насыщающих флюидов: зависимости относительных фазовых проницаемостей, физико-химические свойства, геометрические характеристики, шаг расчетной сетки, пористость-проницаемость, начальное пластовое давление. В текущий момент разработанный алгоритм технически не ограничен формой контура объекта, однако ввиду отсутствия полноценного интерфейса для настройки параметров залежи, все тестовые расчеты выполнялись на синтетических прямоугольных залежах.

В соответствии с заданными параметрами начинается перебор возможных отношений, добывающих/нагнетательных скважин, их расположения. При генерации вариантов формируется массив кортежей, каждый из которых содержит 3 параметра: координата X, координата Y и назначение скважины. Для избегания ситуации расположения скважин слишком близко к друг другу, соседние со взятой скважиной точки также "выкалываются", в соответствии с шагом сетки. Количество скважин определяется через заданные коэффициенты охвата, а именно:

где S – площадь залежи в м2, а kохв – заданный коэффициент охвата.

Далее, сгенерированный массив кортежей отправляется в процедуру симулятора, где объекты-скважины помещаются в соответствии со своими координатами и назначениями. После окончания прогноза расчетная сетка передается обратно в процедуру-генератор, где из объектов-скважин достаются технологические показатели для последующей оценки экономической целесообразности варианта.

На данный момент реализована оценка по приведенной окупаемости разработки, которая осуществляется следующим образом:

Qi – накопленная добыча по i-й скважине в тоннах;

с – стоимость продажи сырья, рублей за тонну;

nj – число скважин соответствующего типа;

P – капитальные затраты на ввод в эксплуатацию скважины соответствующего типа (реализованы наклоно-направленные добывающие скважины, нагнетательные, и нагнетательные с отработкой).

Варианты с наилучшим показателем окупаемости сохраняются в памяти, по достижении числа итераций лучший из сохранившихся вариантов отправляется в модуль для визуализации. Можно произвести оценку энергетической составляющей, увидев поверхность давления, а также параметры каждой из скважин. Таким образом можно получать наиболее перспективные варианты, целесообразность оценки которых на коммерческом симуляторе имеет за собой крепкое обоснование.

## 3.2 ОСНОВНЫЕ КОМПОНЕНТНЫ МОДЕЛИ

В рамках расчетной схемы, для нового варианта разработки инициализируется объект класса Grid, который является одиночным и предоставляет доступ к этому объекту из других процедур проекта. Главное поле класса – матрица ячеек. Индексация происходит в соответствии с параметрами моделируемого месторождения. Так же класс Grid выступает в роли фабрики для объектов класса GridsCell, каждый образец данного класса ставится в соответствие индексу матрицы, что в дальнейшем предоставляет удобный и быстрый доступ к ним. Помимо матрицы у объекта Grid есть поля списка скважин, номерной индексации ячеек. После завершения инициализации объекта расчетной сетки, объект передается вычислительной процедуре.

Вычислительный алгоритм оперирует над объектами ячеек GridsCell, содержащихся в матрице объекта Grid. У объектов-ячеек есть переменные класса и экземпляра. Классовые переменные имеют одинаковое значение для каждого экземпляра, к ним относятся:

* Параметр сжимаемости системы;
* Вязкости флюидов;
* Пористость;
* Сжимаемости флюидов;
* Начальной пластовое давление;
* Начальное насыщение нефтью, определяется геометрическим методом;
* Мощность пласта;
* Геометрические характеристики ячеек;
* Значение абсолютной проницаемостью;

Зависимости относительных фазовых проницаемостей так же закладываются в метод класса ячеек.

Благодаря классовым переменным, удается избежать излишнего использования оперативной памяти для хранения идентичных значений для каждого экземпляра. Поля экземпляра класса содержат уникальные значения соответствующих величин, принадлежащих конкретному объекту, к ним относятся следующие параметры:

* Словари давлений;
* Поля насыщений соответствующей фазы по месяцам;
* Словари накопленных и фиктивных накопленных потоков фазы по направлению;
* Ссылка на объект скважины, если в данной ячейке она должна содержаться, в соответствии с вариантом разработки;
* Номер ячейки;
* Словарь соседних ячеек по направлениям;

Объекты скважин является представителями класса GridsWell, которые так же имею свои поля. К классовым полям относятся скин-фактор и радиус скважины. В текущей реализации скин-фактор скважины является константой, равной -3, однако в реальности данный параметр является динамическим и отражает в себе дополнительное фильтрационное сопротивление, возникающее вследствие несовершенства вскрытия пласта, загрязнения зоны забоя, разгазирования или сжатия скелета горной породы. Полями экземпляра являются:

* Назначение скважины
* Координаты
* Словари накопленная добыча жидкости, воды и нефти у добывающих скважин.
* Словари накопленной текущей закачки у нагнетательных скважин.

Описанная математическая модель реализована с помощью взаимодействия данных объектов. По достижении заданного расчетного месяца объект Grid возвращается в процедуру-генератор, где показатели добычи скважин переводятся в денежный эквивалент, происходит сравнение текущего и лучшего из рассмотренных вариантов. Если текущий вариант более выгоден экономически, то ему присваивается статус лучшего. Данная процедура повторяется до достижения заданного числа итераций.

## 3.3 АНАЛИЗ ПОЛУЧАЕМЫХ РЕЗУЛЬТАТОВ

В данном подразделе я бы хотел продемонстрировать некоторые результаты, получаемые алгоритмом.

По завершении работы, имеем информацию об экономической составляющей каждой их расчетных сессий. Данные будут представлены в виде точечного графика зависимости приведенной стоимости варианта от числа добывающих скважин, рисунок 5.

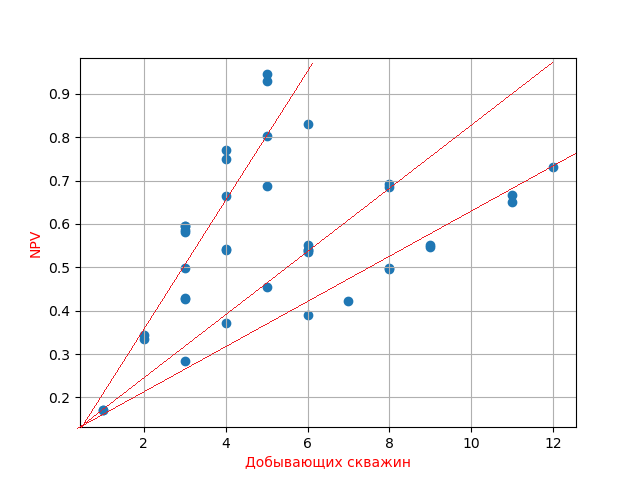


Рис. 5. Приведенная стоимость по вариантам

Расчеты проводились на синтетической модели залежи с площадью 1 км2, прогнозировались первые 12 месяцев разработки.Как говорилось ранее, при генерации варианта алгоритм опирается на закладываемые на этапе генерации параметра охвата скважин. Из графика видно, точки расположились тремя группами. Каждая группа точек относится к своему коэффициенту охвата, соответственно 400, 300 и 250 м2. Наибольшей экономической эффективностью обладают варианты с охватом в 400 м2. Лучший из них содержит пять добывающих скважин и одну нагнетательную. Поверхность давления на момент конца 12 месяца представлен на рисунке 6.

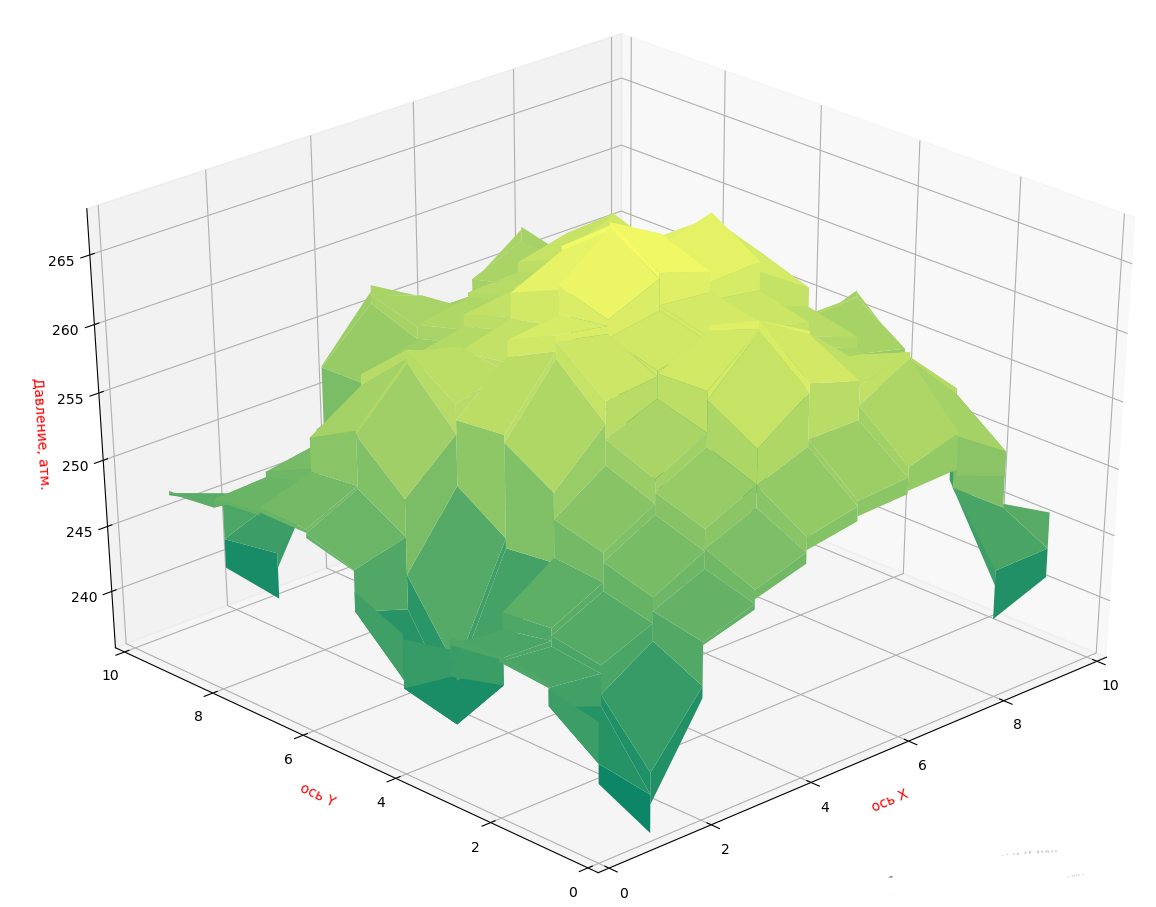


Рис. 6. Поверхность давления

В центральной области расположена нагнетательная скважина, среднее давление на уровне начального пластового, порядка 250 атмосфер, по периметру находятся добывающие скважины, за 12 месяцев работы, просадившие давление на 15-20 атмосфер. Средний дебит нефти за период работы 13 м3/сутки. Карта остаточного насыщения нефтью представлена на рисунке 7.

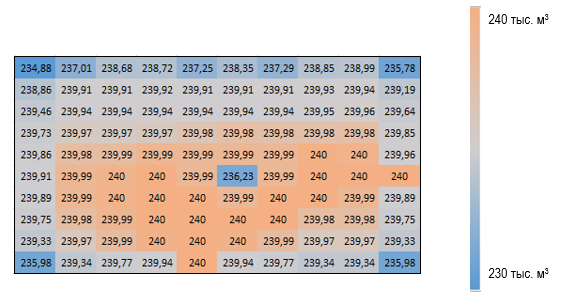


Рис.7. Остаточных насыщения нефтью

За охваченный прогнозом период было извлечено 23,43 тысяч м3нефти, 65 тысяч м3 жидкости. Забойные давления на добывающих скважина 170 атмосфер, на нагнетательной 300 атмосфер.

В рамках синтетического расчета было оценено 30 различных вариантов по 6, 12, 16 скважин соответственно. На выполнение прогноза по всем вариантам ушло менее двух минут. Наилучший из рассмотренных вариантов соотносится с практическим подходом площадного расположения скважин, является чем-то средним между пятиточечной и семиточечной системой.

# 4 ВЫВОДЫ ПО РАБОТЕ

Разработанный алгоритм показал свою применимость для решения задачи подготовки вариантов разработки месторождения. Результаты, получаемые при использовании, соотносятся с существующими подходами расположения скважин. Объем данных позволяет судить о целесообразности того или иного расположений скважин, при этом такая оценка не требовательна по времени и обладает достаточной математической обоснованностью. Конечно, данный подход не позволит полностью отказаться от использования коммерческих гидродинамических симуляторов при проектировании месторождений, но существенно ускорит наиболее затратный с точки зрения времени этап формирования вариантов, отправляемых на расчет, увеличит их обоснованность.

Дальнейшее развитие предложенной концепции позволит коренным образом изменить существующие бизнес процессы современных департаментов разработки месторождений.

# ЗАКЛЮЧЕНИЕ

В процессе написания выпускной квалификационной работы решены следующие задачи:

1. Разработана математическая модель двумерной двухфазной фильтрации на основе уравнения материального баланса.
2. Выполнена дискретизация неявной схемы математической модели двумерной двухфазной фильтрации по оценке пластового давления, явной, по оценке насыщения.
3. Разработан алгоритм и программный комплекс, позволяющий генерировать и оценивать варианты разработки.
4. Выполнена сравнительная оценка точности результатов с известными аналитическими и численными решениями.

Полученную методику в последующем можно усовершенствовать, например, реализовав модель Black Oil, или оптимизировав процесс генерации вариантов. В том числе применить данный алгоритм для реальных месторождений, уже введенных в разработку, и проанализировать результаты относительно сформированной сетки скважин.

Применение подобного подхода на практике позволит высвободить людские ресурсы нефтегазовых кампаний на иные задачи, минимизировать человеческий фактор и связанные с ним ошибки при решении задачи формирования вариантов разработки.

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Андерсон Д. Вычислительная гидромеханика и теплообмен / Д. Андерсон, Дж. Таннехилл, Р. Плетчер. – Москва: Мир, 1990. – 384 с.
2. Башкирцева Н. Ю. Высоковязкие нефти и природные нефти / Н. Ю. Башкирцева // Вестник Казанского технологического университета. – 2014.
3. Крупнов Ю. П. Применение дивергентных схем для задачи Баклея-Леверетта / Ю. П. Крупнов // Доклады БГУИР. – 2013 – № 4 (74). – С. 93–95.
4. Лебедев А. С. Практикум по численному решению уравнений в частных производных / А. С. Лебедев, С. Г. Черный. – Новосибирск: Издательство Новосибирского государственного университета, 2000. – 136 с.
5. S.Patankar Numerical heat transfer and fluid flow/ Hemisphere Publishing Corporation – New York, 1980.
6. Чарный И. А. Подземная гидрогазодинамика / И. А. Чарный. – Москва: Государственное научно-техническое издательство нефтяной и горно-топливной литературы, 1963. – 396 с.
7. Шокин Ю. И. Метод дифференциального приближения: применение к газовой динамике / Ю. И. Шокин, Н. Н. Яненко. – Новосибирск: Наука, 1985. – 357 с.
8. Corey A. T. The interrelation between gas and oil relative permeabilities [Электронный ресурс] / A. T. Corey. – URL: https://www.discovery-group.com/pdfs/Corey\_1954.pdf.
9. Naar J. Imbibition relative permeability in unconsolidated porous media [Электронный ресурс] / J. Naar, R. J. Wygal, J. H. Henderson. – URL: https://www.onepetro.org/journal-paper/SPE-213-PA.
10. Stone H. L. Probability model for estimating three-phase relative permeability [Электронный ресурс] / H. L. Stone. – URL: https://www.onepetro.org/journal-paper/SPE-2116-PA.
11. Баранов В.Е. Разработка коллектора / В.Е. Баранов, М.Р. Камартдинов, Т.Г. Кузьмин – Разработка коллектора учебное пособие, 2006.
12. Котяхов Ф. И. Физика нефтяных и газовых коллекторов. М.: "Недра", 1977, 287 с.
13. Иванов К.Ф., Суриков С. В. Механика жидкости и газа. Конспект лекций для студентов механических и энергетических специальностей. Часть 1. - Одесса: ОГПУ, 1995, - 119с.
14. Каневская Р.Д. Математическое моделирование гидродинамических процессов разработки месторождений углеводородов. - М. - Ижевск: Институт компьютерных исследований, 2002, 140 с.
15. Азиз Х, Сеттари Э. Математическое моделирование пластовых систем: Пер. М.: Недра, 1982 - 407 с. (оригинал:Aziz K., Settari A. PetroleumReservoirSimulation. London: AppliedSciencePublisherLtd., 1979.)
16. Баренблатт Г.И., Ентов В. М., Рыжик В. М. Движение жидкостей и газов в природных пластах. - М.: Недра, 1984. - 211 с.
17. Басниев К. С., Дмитриев Н. М., Розенберг Г. Д. Нефтегазовая гидромеханика: Учебное пособие для вузов. - М. - Ижевск: Институт компьютерных исследований. 2005, 544с.
18. Басниев К. С., Кочина И. Н., Максимов В. М. Подземная гидромеханика: Учебник для вузов. - М.: Недра 1993, 416с.
19. Пирвердяк А. М. Физика и гидравлика нефтяного пласта. - М.: "Недра", 1982, 192 с.
20. Пыхачев Г. Б., Исаев Р. Г. Подземная гидравлика. Учебное пособие. М.: "Недра", 1972, 360 с.

ПРИЛОЖЕНИЯ

## Листинг программы

Приложение 1

Процедура генератора варианта

import copy  
import random  
import math  
from project.calculation import MaterialBalance  
from project.visualisation import drawing  
  
  
def temperature\_k(Tini, k):  
 return Tini/(math.log(1+k))  
  
def income\_calculation(wells\_list, last\_mouth, price):  
 amount = len(wells\_list)  
 accumulated\_production = 0  
 for well in wells\_list:  
 if well.destiny == 'extract':  
 accumulated\_production += well.accumulated\_oil\_production[last\_mouth-1]  
 calc\_npv = (accumulated\_production \* 1000 \* 0.85 \* price)/(amount \* nns\_cost)  
 return calc\_npv  
  
Best\_npv = 0  
Best\_variant = None  
  
oil\_price = 23000 # руб/т  
nns\_cost = 19153500 # цена бурения ННС рублей  
Tini = 10  
Tmin = 1  
"# обязательные параметры "  
fate = ["extract", "inject"]  
Nx = 10  
Ny = 10  
months = int(input("введите число месяцев для прогноза:")) # расчетный шаг в месяцах  
DesignVariant = [(int(Ny/2), int(Nx/2), "extract")]  
  
"# задаю предполагаемый охват у скважины"  
coverage = [400]  
wells\_amount = []  
for item in coverage:  
 wells\_amount.append((Nx\*Ny\*(100\*\*2)//(item\*\*2)))  
smallest\_amount = min(wells\_amount)  
wells\_amount.append(smallest\_amount + 1)  
wells\_amount.append(smallest\_amount - 1)  
  
  
  
Temperature = Tini  
economic\_list = []  
result = None  
  
for wells in wells\_amount:  
 k = 1  
 while k < 10:  
 extract\_amount = 0  
 DesignVariant = []  
 taken\_coords = []  
 compited\_iterations = 0  
 while compited\_iterations < wells:  
 x = random.randint(0, Nx-1)  
 y = random.randint(0, Ny-1)  
 if (y, x) not in taken\_coords:  
 if compited\_iterations > 0:  
 destiny = random.choice(fate)  
 if destiny == "extract":  
 extract\_amount +=1  
 else:  
 destiny = "extract"  
 extract\_amount += 1  
 DesignVariant.append((int(y), int(x), destiny))  
 for dy in range(-1, 2):  
 for dx in range(-1, 2):  
 taken\_coords.append((y + dy, x + dx))  
 taken\_coords = list(set(taken\_coords))  
 compited\_iterations += 1  
  
 print(f"добывающих {extract\_amount} из {wells}")  
 print(DesignVariant)  
  
  
 result = MaterialBalance.MB\_calculation(Nx, Ny, DesignVariant, months)  
 NPV = income\_calculation(result.wells\_list, months, oil\_price)  
 prodaction\_to\_injection = 0  
 for well in result.wells\_list:  
 if well.destiny is "extract":  
 prodaction\_to\_injection += 1  
 prodaction\_to\_injection = prodaction\_to\_injection  
 economic\_list.append((prodaction\_to\_injection, NPV))  
 if NPV > Best\_npv:  
 Best\_variant = result  
 print(NPV)  
  
 Temperature = temperature\_k(Tini, k)  
 k += 1  
  
 drawing.draw\_pressure\_graph(Nx, Ny, result, months)  
 drawing.draw\_NPV\_grapg(economic\_list)  
 drawing.draw\_oil\_saturation(Nx, Ny, result, months)  
print(f"итераций {k}")

Приложение 2

Процедура материального баланса

import sys  
from project.ObjectModels import Grid  
from project.calculation import Leverett  
from project.calculation import BuckleyLeverett  
import numpy as np  
import time  
import copy  
import math  
from project.visualisation import drawing  
  
  
def MB\_calculation(Nx, Ny, DesignVariant, calculation\_steps):  
 def \_material\_balance(x, y, a\_matrix, b\_matrix, scenario=None):  
 def \_calculate\_material\_balance\_fluid(step, direction = None, coordinates = None):  
 *"""This function calculate material balance* ***:param*** *step: moment of time* ***:param*** *direction: if this parameter given, should take one pressure from last step* ***:param*** *coordinates: index for neighbour cells* ***:return****: material balance value  
 """* pressures\_copy = copy.deepcopy(neighbours\_pressure)  
  
 if direction:  
 x = coordinates[1]  
 y = coordinates[0]  
 pressures\_copy[direction] = CellsBox.matrix [y, x].get\_prev\_pressure\_fluid(step)  
  
 "# если в ячейке есть скважина, проверяю ее назначение и пересчитываю накопленные параметры"  
 injection = 0  
 production = 0  
 if cell.well\_presence:  
 well = cell.well\_presence  
 pressure\_well = cell.well\_presence.well\_pressure  
 if pressure\_well < pressures\_copy["itself"]:  
 injection = well.get\_accumulated\_injection(step)  
 production = well.fluid\_production(cell, step)  
 elif pressure\_well >= pressures\_copy["itself"]:  
 injection = well.water\_injection(cell, step)  
 production = well.get\_accumulated\_fluid\_production(step)  
  
 "# записываю давление в текущей ячейке в переменную, чтобы дальше удобно считать поток по соседям"  
 cell\_pressure = pressures\_copy.pop("itself") # cell\_pressure давление в ячейке, для которой считается Матбаланс  
 summary\_flow = 0  
 for flow\_to, pressure in pressures\_copy.items():  
 summary\_flow += cell.calculate\_flow\_fluid(flow\_to, pressure, cell\_pressure, CellsBox, step)  
 ini\_fluid = Grid.GridsElements.GridsCell.beginningFluid  
 ce = Grid.GridsElements.GridsCell.ce  
  
 MatBal = Grid.GridsElements.GridsCell.beginningPressure - cell\_pressure - ((production + injection + summary\_flow)/  
 (ini\_fluid \* ce))  
 return MatBal  
  
 cell = CellsBox.matrix[y, x]  
 neighbours\_pressure = {"itself": None, "west": None, "north": None, "east": None, "south": None}  
 "# давления из ячеек соседнего окружения, если соседняя ячейка за контуром, то беру начальное пластовое"  
 for direction, coordinates in cell.neighbours.items():  
 if coordinates and scenario == "fluid":  
 neighbours\_pressure[direction] = CellsBox.matrix[coordinates[0], coordinates[1]].get\_pressure\_fluid(step)  
 else: # если входим в это условие, то значит соседняя ячейка это аквифер  
 neighbours\_pressure[direction] = Grid.GridsElements.GridsCell.beginningPressure  
  
 "# заполнение матриц а, матрица в вписывается значение мб с текущими давлениями, чтобы после снова не считать "  
 for direction, coordinates in cell.neighbours.items():  
 for\_which = cell.cell\_number # ячейка, для которой считаем мб  
 if coordinates and scenario == "fluid":  
 by\_which = CellsBox.matrix[coordinates[0], coordinates[1]].cell\_number # ячейка, из которой берем приближение давления  
 mb\_current\_pressure = \_calculate\_material\_balance\_fluid(step) # пересчет мб с текущими давлениями  
 b\_matrix\_fluid[for\_which] = -mb\_current\_pressure  
 mb\_dif\_pressure = \_calculate\_material\_balance\_fluid(step, direction, coordinates) # пересчет мб с давлением прошлого шага  
 delta\_pressure = CellsBox.matrix[coordinates[0], coordinates[1]].delta\_pressure\_fluid(step)  
 a\_matrix[for\_which, by\_which] = (mb\_current\_pressure - mb\_dif\_pressure)/delta\_pressure# считаю коэффициент a  
  
 timeBefore = time.time()  
 CellsBox = Grid.Grid(Nx, Ny, DesignVariant)  
 timeAfter = time.time()  
  
 a\_matrix\_fluid = np.zeros((Ny \* Nx, Nx \* Ny), dtype="float32") # a\_matrix{y,x] где у - номер ячейки, для которой считаем производную, x - по которой  
 b\_matrix\_fluid = np.zeros((Ny \* Nx), dtype="float32") # один b для каждой ячейки  
  
 for step in range(calculation\_steps):  
 do\_iter = 0  
 while True:  
 "# прохожусь по ячейкам, передаю координаты текущей ячейки в матбаланс"  
 for y in range(Ny):  
 for x in range(Nx):  
 \_material\_balance(x, y, a\_matrix\_fluid, b\_matrix\_fluid, scenario="fluid")  
  
 for for\_which in range(Nx\*Ny):  
 for by\_which in range(Nx\*Ny):  
 fluid\_pressure\_in\_by\_which = CellsBox.cells\_numbers[by\_which].get\_pressure\_fluid(step)  
 b\_matrix\_fluid[for\_which] += a\_matrix\_fluid[for\_which, by\_which] \* fluid\_pressure\_in\_by\_which  
  
 pressure\_fluid\_roots = np.linalg.solve(a\_matrix\_fluid, b\_matrix\_fluid)  
  
 pressure\_accuracy = True  
 for cell in range(Nx\*Ny):  
 fluid\_accuracy = (abs(CellsBox.cells\_numbers[cell].get\_pressure\_fluid(step) - pressure\_fluid\_roots[cell]) > 0.1)  
 if fluid\_accuracy:  
 pressure\_accuracy = False  
 break  
  
 # выполнится если сошлось давление по жидкости  
 for cell in range(Nx \* Ny):  
 CellsBox.cells\_numbers[cell].new\_approach(pressure\_fluid\_roots[cell], step, pressure\_accuracy)  
  
 if pressure\_accuracy:  
 pressure\_oil\_roots = []  
 for cell in range(Nx\*Ny):  
 water\_saturation = CellsBox.cells\_numbers[cell].water\_fund[step-1]/CellsBox.cells\_numbers[cell].fluid\_fund[step]  
 pressure\_oil\_roots.append(pressure\_fluid\_roots[cell] - Leverett.JFunction(water\_saturation))  
 CellsBox.cells\_numbers[cell].save\_pressures\_oil(pressure\_oil\_roots[cell], step)  
  
 # ниже и до конца отвечает за изменение запасов фаз в ячейках  
 for cell in range(Nx\*Ny):  
 element = CellsBox.cells\_numbers[cell]  
 oil\_flow = 0  
 #water\_flow = 0  
  
 for direction, coordinates in element.neighbours.items():  
 if direction != "itself" and coordinates is not None:  
 flow\_for\_direction = element.fluid\_flow\_fict[direction]  
 this\_cell\_pressure = pressure\_fluid\_roots[cell]  
 neighbour\_number = CellsBox.matrix[coordinates[0]][coordinates[1]].cell\_number  
 another\_cells\_pressure = pressure\_fluid\_roots[neighbour\_number]  
 if this\_cell\_pressure >= another\_cells\_pressure:  
 if element.get\_oil\_permeability(step) == 0:  
 oil\_perm = 0  
 water\_perm = 1  
 else:  
 water\_perm = CellsBox.cells\_numbers[neighbour\_number].get\_water\_permeability(step)  
 oil\_perm = CellsBox.cells\_numbers[neighbour\_number].get\_oil\_permeability(step)  
 else:  
 if CellsBox.cells\_numbers[neighbour\_number].get\_oil\_permeability(step) == 0:  
 oil\_perm = 0  
 water\_perm = 1  
 else:  
 water\_perm = element.get\_water\_permeability(step)  
 oil\_perm = element.get\_oil\_permeability(step)  
  
 water\_share = BuckleyLeverett.BuckleyLeverett(water\_perm, oil\_perm) # доля воды в потоке  
 oil\_flow += -1 \* (flow\_for\_direction \* (1-water\_share) ) # домножение на -1, т.к поток считается отрицательным, если направлен ВНУТРЬ элемента.  
  
 oil\_production = 0  
 if element.well\_presence:  
 well = CellsBox.cells\_numbers[cell].well\_presence  
 if well.destiny == "extract":  
 fluid\_production = well.fict\_fluid\_production  
 water\_perm = element.get\_water\_permeability(step)  
 oil\_perm = element.get\_oil\_permeability(step)  
 water\_share = BuckleyLeverett.BuckleyLeverett(water\_perm, oil\_perm)  
 oil\_production = fluid\_production \* (1 - water\_share)  
 well.save\_production(step, oil\_production, "oil")  
  
 if oil\_flow > element.beginningOil - element.oil\_fund[step-1] + oil\_production:  
 oil\_flow =0  
  
 element.oil\_fund[step] = element.oil\_fund[step - 1] - oil\_production + oil\_flow  
 fluid\_in\_cell = element.fluid\_fund[step]  
 element.water\_fund[step] = fluid\_in\_cell \* (1 - element.oil\_fund[step]/fluid\_in\_cell)  
  
 if pressure\_accuracy:  
 break  
 do\_iter += 1  
 print("calculating ...")  
  
  
 "# давления на последний месяц"  
 print(f"давление по нефти")  
 for i in range(Ny):  
 for j in range(Nx):  
 for\_print = str(CellsBox.matrix[i, j].layer\_pressure\_oil[step])  
 print(for\_print[0:6], end="|")  
 print(f"\n{'-'\*Nx\*Nx}")  
  
 print(f"давление по воде")  
 for i in range(Ny):  
 for j in range(Nx):  
 for\_print = str(CellsBox.matrix[i, j].layer\_pressure\_fluid[step])  
 print(for\_print[0:6], end="|")  
 print(f"\n{'-'\*Nx\*Nx}")  
  
 print(f"тыс.м3 жидкости в ячейке")  
 for i in range(Ny):  
 for j in range(Nx):  
 for\_print = str(CellsBox.matrix[i, j].fluid\_fund[step])  
 print(for\_print[0:6], end="|")  
 print(f"\n{'-' \* Nx \* Nx}")  
  
 print(f'тип хранилища ячеек {type(CellsBox)}')  
 print(f'время на инициализацию сетки: {timeAfter-timeBefore} секунд')  
 return CellsBox  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 Nx = 10  
 Ny = 10  
 DesignVariant = [(0, 0, "extract"), (0, 4, "extract"), (0, int(9), "extract"), (int(9), 0, "extract"),  
 (0, int(1), "extract"), (0, int(6), "extract"), (int(5), int(5), "inject")]  
 months = 12  
 calculatedObject = MB\_calculation(Nx, Ny, DesignVariant, months)  
 for mounth in range(months):  
 print(f"нысыщение блоков водой на {mounth} месяц")  
 for i in range(Ny):  
 for j in range(Nx):  
 for\_print = str(calculatedObject.matrix[i, j].water\_fund[mounth]/calculatedObject.matrix[i,j].oil\_fund[mounth])  
 print(for\_print, end="|")  
 print(f"\n{'-' \* Nx \* Ny }")  
  
 drawing.draw\_oil\_saturation(Nx, Ny, calculatedObject, months)

from project.ObjectModels import GridsElements  
  
  
def BuckleyLeverett(water\_permeability, oil\_permeability):  
 mu\_oil = GridsElements.GridsCell.mu\_oil  
 mu\_water = GridsElements.GridsCell.mu\_water  
 return water\_permeability/(water\_permeability + (mu\_water/mu\_oil)\*oil\_permeability)

Приложение 3

Объектная модель

import numpy as np  
import project.ObjectModels.GridsElements as GridsElements  
  
  
class Grid:  
  
 def \_\_init\_\_(self, Nx, Ny, DesignVariant = None):  
 *"""This function create Numpy array and add in each cell element of Grid* ***:param*** *Nx: amount of cells along the X axis* ***:param*** *Ny: amount of cells along the Y axis* ***:param*** *DesignVariant: tuple with wells coordinates and purpose* ***:return****: generated numeric field  
 """* def \_neighbour\_identification(x, y):  
 *"""This function define cell's neighbours* ***:param*** *x: cell's x coordinate* ***:param*** *y: cell's y coordinate* ***:return****: dict containing neighbours  
 """* neighbours = {"itself": (y, x), "west": None, "north": None, "east": None, "south": None}  
 if x != 0:  
 neighbours["west"] = (y, x - 1)  
 if x != Nx - 1:  
 neighbours["east"] = (y, x + 1)  
 if y != 0:  
 neighbours["north"] = (y - 1, x)  
 if y != Ny - 1:  
 neighbours["south"] = (y + 1, x)  
  
 return neighbours  
  
 def \_create\_matrix(self):  
 matrix = np.zeros((Ny, Nx), dtype=type(GridsElements.GridsCell))  
 number = 0  
 for y in range(Ny):  
 for x in range(Nx):  
 neighbours = \_neighbour\_identification(x, y)  
 matrix[y, x] = GridsElements.GridsCell(neighbours=neighbours, cell\_number= number)  
 self.cells\_numbers[number] = matrix[y, x]  
 number += 1  
  
 for i in range(len(DesignVariant)): # список кортежей-вариантов (y,x, назначение)  
 variant = DesignVariant[i]  
 x = variant[1]  
 y = variant[0]  
 well\_type = variant[2]  
 matrix[y, x].well\_presence = GridsElements.GridsWell(destiny=well\_type,  
 coordinate\_x=x,  
 coordinate\_y=y,  
 well\_number=i)  
 self.wells\_list.append(matrix[y, x].well\_presence)  
 return matrix  
 self.wells\_list = []  
 self.cells\_numbers = {}  
 self.Nx = Nx  
 self.Ny = Ny  
 self.matrix = \_create\_matrix(self)

import numpy  
import math  
  
class GridsCell:  
 *"""  
 в классовых переменных указаные общие параметры для ячеек:  
 -----------  
 beginningPressure - Начальное пластовое давление (атм)  
 beginningOil - начальные запасы нефти (тыс.м3.)  
 CellSize - размер ребра ячейки (м)  
 CellHeight - мощность ННТ в ячейке (м)  
 """* ce = 0.0000149 # сжимаемость системы, аналитическая функция в матмодели (1/атм)  
 mu\_oil = 8 # сП  
 mu\_water = 1 # сП  
 mu\_fluid = 5.5 # посмотреть как пересчитывать в зависимости от содержания компонент  
 porosity = 0.4  
 Boil = 1.126  
 Bw = 1  
 beginningPressure = 250.0 #начальное пластовое давление (атм)  
 beginningOilSaturation = 0.6 #доля нефти в жидкости  
 CellSize = 100.0 #размер ребра ячейки (м)  
 CellHeight = 3.0 #мощность ННТ в ячейке (м)  
 beginningFluid = ((CellSize \*\* 3) \* porosity)/1000 # начальные запасы жидкости тыс м3  
 beginningOil = beginningFluid\*beginningOilSaturation # начальные запасы нефти тыс м3  
 beginningWater = beginningFluid - beginningOil # начальные запасы воды тыс м3  
 absolute\_permeability = 50 # абсолютная проницаемость в мД  
  
 def \_\_init\_\_(self, well\_presence = None, neighbours=None, cell\_number=None):  
 self.layer\_pressure\_water = {} # пластовое давление в ячейке (атмосферы) на конец месяца  
 self.layer\_pressure\_fluid = {}  
 self.layer\_pressure\_oil = {}  
 self.oil\_fund = {}  
 self.water\_fund = {}  
 self.fluid\_fund = {}  
 self.layer\_pressure\_water\_prev = GridsCell.beginningPressure + 1 # пластовое давление в ячейке (атмосферы) на конец месяца  
 self.layer\_pressure\_oil\_prev = GridsCell.beginningPressure + 1  
 self.layer\_pressure\_fluid\_prev = GridsCell.beginningPressure + 1  
 self.water\_flow\_accumulated = {"west": 0, "north": 0, "east": 0, "south": 0} # словарь перетоков, по ключу хранится накопленный по данному направлению переток  
 self.water\_flow\_fict = {"west": 0, "north": 0, "east": 0, "south": 0} # словарь фиктивных перетоков на i шаге, после схождения эти значения суммируются в flow\_accumulated  
 self.oil\_flow\_accumulated = {"west": 0, "north": 0, "east": 0, "south": 0}  
 self.oil\_flow\_fict = {"west": 0, "north": 0, "east": 0, "south": 0}  
 self.fluid\_flow\_accumulated = {"west": 0, "north": 0, "east": 0, "south": 0}  
 self.fluid\_flow\_fict = {"west": 0, "north": 0, "east": 0, "south": 0}  
 self.well\_presence = well\_presence # есть ли в ячейке скважина, если есть нужно хранить ссылку на объект скважины  
 self.oil\_saturation = float() # текущая нефтенасыщенность на конец месяца  
 self.neighbours = neighbours # словарь соседей  
 self.cell\_number = cell\_number  
 self.absolute\_permeability = GridsCell.absolute\_permeability  
 self.oil\_fund[-1] = GridsCell.beginningOil  
 self.water\_fund[-1] = GridsCell.beginningWater  
 self.fluid\_fund[-1] = GridsCell.beginningFluid  
 self.layer\_pressure\_water[0] = GridsCell.beginningPressure  
 self.layer\_pressure\_oil[0] = GridsCell.beginningPressure  
 self.layer\_pressure\_fluid[0] = GridsCell.beginningPressure  
  
 def get\_accumulated\_fluid\_flow(self, step, direction):  
 if step == 0:  
 return 0  
 else:  
 return self.fluid\_flow\_accumulated[direction]  
  
 def get\_accumulated\_water\_flow(self, step, direction):  
 if step == 0:  
 return 0  
 else:  
 return self.water\_flow\_accumulated[direction]  
  
 def get\_accumulated\_oil\_flow(self, step, direction):  
 if step == 0:  
 return 0  
 else:  
 return self.oil\_flow\_accumulated[direction]  
  
 def delta\_pressure\_fluid(self, step):  
 return self.layer\_pressure\_fluid[step] - self.layer\_pressure\_fluid\_prev  
  
 def get\_pressure\_fluid(self, step):  
 return self.layer\_pressure\_fluid[step]  
  
 def get\_prev\_pressure\_fluid(self, step):  
 return self.layer\_pressure\_fluid\_prev  
  
 def delta\_pressure\_oil(self, step):  
 return self.layer\_pressure\_oil[step] - self.layer\_pressure\_oil\_prev  
  
 def get\_pressure\_oil(self, step):  
 return self.layer\_pressure\_oil[step]  
  
 def get\_prev\_pressure\_oil(self, step):  
 return self.layer\_pressure\_oil\_prev  
  
 def get\_fluid\_permeability(self):  
 return self.absolute\_permeability  
  
 def get\_water\_permeability(self, step):  
 Sw = self.water\_fund[step-1] / self.fluid\_fund[step-1]  
 if Sw < 0.2:  
 Sw = 0.2  
 elif Sw > 0.7:  
 Sw = 0.7  
 RPP = 0.4286\*Sw  
 #RPP = (14.358\*(Sw\*\*3) - 15.464\*(Sw\*\*2) + 5.5374\*Sw - 0.6512) # relative phase permeability  
 return self.absolute\_permeability \* RPP  
  
 def get\_oil\_permeability(self, step):  
 Sw = self.water\_fund[step-1] / self.fluid\_fund[step-1]  
  
 if 0 <= Sw < 0.7:  
 RPP = 7.1429 \* (Sw \*\* 4) - 10.952 \* (Sw \*\* 3) + 6.7143 \* (Sw \*\* 2) - 3.2119 \* Sw + 1  
 elif Sw < 0:  
 Sw = 0  
 RPP = 7.1429 \* (Sw \*\* 4) - 10.952 \* (Sw \*\* 3) + 6.7143 \* (Sw \*\* 2) - 3.2119 \* Sw + 1  
 elif Sw > 0.7:  
 RPP = 0  
 else:  
 RPP = 0  
  
 #RPP = 0.5357\*(Sw\*\*4) - 2.1071 \* (Sw\*\*3) + 3.4232\*(Sw\*\*2) - 2.8518\*Sw + 1  
 #RPP = (440.02\*(Sw\*\*4) - 859.22\*(Sw\*\*3) + 628.34\*(Sw\*\*2) - 204.27\*So + 24.955) # relative phase permeability  
 return self.absolute\_permeability \* RPP  
  
 def save\_pressures\_oil(self, new\_oil\_press, step):  
 self.layer\_pressure\_oil\_prev = self.layer\_pressure\_oil[step]  
 self.layer\_pressure\_oil[step + 1] = new\_oil\_press  
  
 def calculate\_flow\_fluid(self, direction, another\_cell\_pressure, this\_cell\_pressure, grid, step):  
 *"""This function calculate fluid flow* ***:param*** *direction: flow direction (west, north, east, south)* ***:param*** *another\_cell\_pressure: pressure in that direction* ***:param*** *this\_cell\_pressure: pressure in this cell* ***:param*** *grid: numeric field, given for taking pressure in another cell* ***:param*** *step: calculation step (month)* ***:return****: fixes in entity and return value of flow in thousand meters\*\*3  
 """* permeability = self.get\_fluid\_permeability()  
 if self.neighbours[direction] and (this\_cell\_pressure >= another\_cell\_pressure):  
 neighbour\_x = self.neighbours[direction][1]  
 neighbour\_y = self.neighbours[direction][0]  
 permeability = grid.matrix[neighbour\_y, neighbour\_x].get\_fluid\_permeability()  
 contact\_space = GridsCell.CellHeight \* 1.5 # GridsCell.CellSize  
 delta\_pressure = this\_cell\_pressure - another\_cell\_pressure  
 acc\_flow = self.get\_accumulated\_fluid\_flow(step, direction) # накопленный по направлению поток  
 self.fluid\_flow\_fict[direction] = ((permeability \* contact\_space \* delta\_pressure)  
 / (GridsCell.mu\_fluid \* GridsCell.CellSize)) \* 0.03 + acc\_flow  
 return self.fluid\_flow\_fict[direction]  
   
 def calculate\_flow\_water(self, direction, another\_cell\_pressure, this\_cell\_pressure, grid, step):  
 *"""This function calculate water flow using upstream permeability* ***:param*** *direction: flow direction (west, north, east, south)* ***:param*** *another\_cell\_pressure: pressure in that direction* ***:param*** *this\_cell\_pressure: pressure in this cell* ***:param*** *grid: numeric field, given for taking pressure in another cell* ***:param*** *step: calculation step (month)* ***:return****: fixes in entity and return value of flow in thousand meters\*\*3  
 """* permeability = self.get\_water\_permeability(step)  
 if self.neighbours[direction] and (this\_cell\_pressure >= another\_cell\_pressure):  
 neighbour\_x = self.neighbours[direction][1]  
 neighbour\_y = self.neighbours[direction][0]  
 permeability = grid.matrix[neighbour\_y, neighbour\_x].get\_water\_permeability()  
 contact\_space = GridsCell.CellHeight \* 1.5 # GridsCell.CellSize  
 delta\_pressure = this\_cell\_pressure - another\_cell\_pressure  
 self.water\_flow\_fict[direction] = ((permeability \* contact\_space \* delta\_pressure)  
 / (GridsCell.mu\_water \* GridsCell.CellSize)) \* 0.03  
 return self.water\_flow\_fict[direction]  
  
 def calculate\_flow\_oil(self, direction, another\_cell\_pressure, this\_cell\_pressure, grid, step):  
 *"""This function calculate oil flow using upstream permeability* ***:param*** *direction: flow direction (west, north, east, south)* ***:param*** *another\_cell\_pressure: pressure in that direction* ***:param*** *this\_cell\_pressure: pressure in this cell* ***:param*** *grid: numeric field, given for taking pressure in another cell* ***:param*** *step: calculation step (month)* ***:return****: fixes in entity and return value of flow in thousand meters\*\*3  
 """* permeability = self.get\_oil\_permeability(step)  
 if self.neighbours[direction]:  
 if self.neighbours[direction] and (this\_cell\_pressure >= another\_cell\_pressure):  
 neighbour\_x = self.neighbours[direction][1]  
 neighbour\_y = self.neighbours[direction][0]  
 permeability = grid.matrix[neighbour\_y, neighbour\_x].get\_oil\_permeability()  
 contact\_space = GridsCell.CellHeight \* 1.5 # GridsCell.CellSize  
 delta\_pressure = this\_cell\_pressure - another\_cell\_pressure  
 self.oil\_flow\_fict[direction] = ((permeability \* contact\_space \* delta\_pressure)  
 / (GridsCell.mu\_oil \* GridsCell.CellSize)) \* 0.03 # 0.3 => \* 30 дней / 1000  
 return self.oil\_flow\_fict[direction]  
 else:  
 return 0  
  
 def new\_approach(self, new\_fluid\_pres, step, accuracy): # если accuracy True, то следующий шаг по времени, в противном случае только замена давлений  
 if not accuracy:  
 self.layer\_pressure\_fluid\_prev = self.layer\_pressure\_fluid[step]  
 self.layer\_pressure\_fluid[step] = new\_fluid\_pres  
  
 else:  
 self.layer\_pressure\_fluid\_prev = new\_fluid\_pres + 1  
 self.layer\_pressure\_fluid[step + 1] = new\_fluid\_pres  
  
 directions = self.fluid\_flow\_accumulated.keys()  
 fluid\_summ = 0  
  
 for direction in directions:  
 self.fluid\_flow\_accumulated[direction] = self.fluid\_flow\_fict[direction]  
 fluid\_summ += self.fluid\_flow\_fict[direction]  
 if self.well\_presence:  
 well = self.well\_presence  
 well.new\_approach(step) # этим действием вызывается запись накопленных показателей по скважине  
 fluid\_summ += well.fict\_fluid\_production + well.fict\_water\_injection  
  
 self.fluid\_fund[step] = self.fluid\_fund[step - 1] - fluid\_summ  
  
  
class GridsWell:  
  
 Rw = 0.15 # радиус ствола скважины, м  
 Skin = -3 # скин фактор  
  
 def \_\_init\_\_(self, coordinate\_x=None, coordinate\_y=None, well\_number=None, destiny=None):  
  
 def radios\_calc():  
 return (0.28 \* (2 \*(GridsCell.CellSize\*\*2)\*\*0.5))/(2 \* ((GridsCell.absolute\_permeability \* pow(10, -9)) \*\* 0.25))  
  
 self.Rb = radios\_calc()  
 self.well\_number = well\_number  
 self.destiny = destiny  
 self.coordinate\_x = coordinate\_x  
 self.coordinate\_y = coordinate\_y  
 self.accumulated\_water\_injection = {} # накопленная скважиной закачка на текущий момент  
 self.accumulated\_water\_production = {} # накопленная скважиной добыча на текущий момент  
 self.accumulated\_fluid\_production = {}  
 self.accumulated\_oil\_production = {}  
 self.fict\_water\_injection = 0  
 self.fict\_water\_production = 0  
 self.fict\_fluid\_production = 0  
 self.fict\_oil\_production = 0  
  
 if self.destiny == "inject":  
 self.well\_pressure = GridsCell.beginningPressure + 50  
 elif self.destiny == "extract":  
 self.well\_pressure = GridsCell.beginningPressure - 80  
  
 def get\_accumulated\_injection(self, step):  
 if step == 0:  
 return 0  
 else:  
 return self.accumulated\_water\_injection[step-1]  
  
 def get\_accumulated\_water\_production(self, step):  
 if step == 0:  
 return 0  
 else:  
 return self.accumulated\_water\_production[step-1]  
  
 def get\_accumulated\_fluid\_production(self, step):  
 if step == 0:  
 return 0  
 else:  
 return self.accumulated\_fluid\_production[step-1]  
  
 def get\_accumulated\_oil\_production(self, step):  
 if step == 0:  
 return 0  
 else:  
 return self.accumulated\_oil\_production[step-1]  
  
 def water\_injection(self, cell, step):  
 *"""This function return accumulated water injection considering current mouth production* ***:param*** *cell: cell entity, which contain this well* ***:param*** *step: current month* ***:return****: fictitious water injection  
 """* layer\_pressure = cell.get\_pressure\_fluid(step)  
 delta\_pressure = layer\_pressure - self.well\_pressure  
 self.fict\_water\_injection = ((cell.get\_water\_permeability(step) \* GridsCell.CellHeight \* delta\_pressure)/  
 (18.41 \* GridsCell.mu\_water \* (math.log((self.Rb / self.Rw)) - 0.75 + self.Skin)) \* 0.03  
 + self.get\_accumulated\_injection(step))  
 return self.fict\_water\_injection  
  
 def water\_production(self, cell, step):  
 *"""This function return accumulated water production considering current mouth production* ***:param*** *cell: cell entity, which contain this well* ***:param*** *step: current month* ***:return****: fictitious water production  
 """* layer\_pressure = cell.get\_pressure\_water(step)  
 delta\_pressure = layer\_pressure - self.well\_pressure  
 self.fict\_water\_production = ((cell.get\_water\_permeability(step) \* GridsCell.CellHeight \* delta\_pressure)/  
 (18.41 \* GridsCell.mu\_water \* (math.log((self.Rb/self.Rw)) - 0.75 + self.Skin))\*0.03  
 + self.get\_accumulated\_water\_production(step))  
 return self.fict\_water\_production  
   
 def fluid\_production(self, cell, step):  
 *"""This function return accumulated fluid production considering current mouth production* ***:param*** *cell: cell entity, which contain this well* ***:param*** *step: current month* ***:return****: fictitious fluid production  
 """* layer\_pressure = cell.get\_pressure\_fluid(step)  
 delta\_pressure = layer\_pressure - self.well\_pressure  
 self.fict\_fluid\_production = ((cell.get\_fluid\_permeability() \* GridsCell.CellHeight \* delta\_pressure)/  
 (18.41 \* GridsCell.mu\_fluid \* (math.log((self.Rb/self.Rw)) - 0.75 + self.Skin))\*0.03  
 + self.get\_accumulated\_fluid\_production(step))  
 return self.fict\_fluid\_production  
  
 def oil\_production(self, cell, step):  
 *"""This function return accumulated oil production considering current mouth production* ***:param*** *cell: cell entity, which contain this well* ***:param*** *step: current month* ***:return****: fictitious oil production  
 """* layer\_pressure = cell.get\_pressure\_oil(step)  
 delta\_pressure = layer\_pressure - self.well\_pressure  
 self.fict\_oil\_production = ((cell.get\_oil\_permeability(step) \* GridsCell.CellHeight \* delta\_pressure)/  
 (18.41 \* GridsCell.mu\_oil \* (math.log((self.Rb/self.Rw)) - 0.75 + self.Skin))\*0.03  
 + self.get\_accumulated\_oil\_production(step))  
 return self.fict\_oil\_production  
  
 def new\_approach(self, step):  
 *""" this function fix well job for month* ***:param*** *step: calculation month* ***:return****: nothing  
 """* self.accumulated\_fluid\_production[step] = self.fict\_fluid\_production  
 self.accumulated\_water\_injection[step] = self.fict\_water\_injection  
  
 def save\_production(self, step, production, phase):  
 if self.destiny is "extract":  
 if phase == "oil":  
 self.accumulated\_oil\_production[step] = production  
 elif phase == "water":  
 self.accumulated\_water\_production[step] = production  
 else:  
 self.accumulated\_water\_injection[step] = production

Приложение 4

Визуализация

import matplotlib.pyplot as plot  
from matplotlib import cm  
import numpy  
from mpl\_toolkits.mplot3d import Axes3D  
from scipy.ndimage import zoom  
  
  
def draw\_pressure\_graph(Nx, Ny, myData, month):  
 zoom\_power = 2  
 x = numpy.arange(0, Nx, 1)  
 y = numpy.arange(0, Ny, 1)  
 xgrid, ygrid = numpy.meshgrid(x, y)  
 xgrid = zoom(xgrid, zoom\_power )  
 ygrid = zoom(ygrid, zoom\_power)  
 zgrid = numpy.zeros((Ny, Nx))  
 for i in range(Ny):  
 for j in range(Nx):  
 zgrid[i,j] = myData.matrix[i, j].layer\_pressure\_fluid[month]  
 zgrid = zoom(zgrid, zoom\_power)  
 fiqure = plot.figure()  
 axes = Axes3D(fiqure)  
 axes.plot\_surface(xgrid, ygrid, zgrid, cmap="summer", antialiased=True)  
 axes.set\_xlabel("ось X", color="red")  
 axes.set\_ylabel("ось Y", color="red")  
 axes.set\_zlabel("Давление, атм.", color="red")  
 plot.xlim(0, Nx)  
 plot.ylim(0, Ny)  
  
 plot.show()  
  
def draw\_NPV\_grapg(NPV\_list):  
 fiqure = plot.figure()  
 wells\_amount = []  
 NPV\_per\_variant = []  
 axes2 = fiqure.add\_subplot()  
 for wells, npv in NPV\_list:  
 wells\_amount.append(wells)  
 NPV\_per\_variant.append(npv)  
 wells\_amount = numpy.array(wells\_amount)  
 NPV\_per\_variant = numpy.array(NPV\_per\_variant)  
 axes2.scatter(wells\_amount, NPV\_per\_variant)  
 axes2.set\_xlabel("Добывающих скважин", color="red")  
 axes2.set\_ylabel("NPV", color="red")  
 axes2.grid()  
  
 plot.show()  
  
  
def draw\_oil\_saturation(Nx, Ny, myData, month):  
 x = numpy.arange(0, Nx, 1)  
 y = numpy.arange(0, Ny, 1)  
 xgrid, ygrid = numpy.meshgrid(x, y)  
 zgrid = numpy.zeros((Ny, Nx))  
 for i in range(Ny):  
 for j in range(Nx):  
 zgrid[i, j] = myData.matrix[i, j].oil\_fund[month-1]  
 fiqure = plot.figure()  
 axes = plot.imshow(zgrid)  
 plot.colorbar()  
 axes.set\_cmap("Blues\_r")  
  
 plot.show()