Скважины

Для скважин создан базовый класс well (bs_bos_core/include/calc_well.h)

Скважина содержит:

1. Список перфораций
    1. Список первичных перфораций
    2. Список вторичных перфораций (например, добавляемых трещиной и т.п.)
2. Количество открытых перфораций
3. ~~Ассоциативный массив, для быстрого поиска перфораций по номеру блока~~
4. Данные дебета (?) скважины (каждый дебет содержит нагнетательную и добывающую части)
    1. oil_rate
    2. water_rate
    3. gas_rate
    4. liquid_rate (oil_rate + water_rate)
5. Суммарные данные дебета и дебет в поверхностных условиях
6. Координаты и другие данные, задаваемые в schedule файле:
    1. coord
    2. bhp_depth
    3. wefac (exploitation_factor)
    4. name
7. Контроллер скважины (well_controller_)
8. Состояние (well_state_)
9. Объекты для расчета:
    1. Давление в скважине (calc_well_pressure_)
    2. Плотности в скважине (calc_rho_)
    3. Плотности в перфорациях (calc_perf_density_)
    4. BHP в перфорациях (calc_perf_bhp_)
10. Список well_facilities (например, трещины должны быть реализованы через интерфейс well_facility_iface).

В скважине расчитываются данные. Часть этих данных хранится в ww_value и bw_value.

Функция *process* является основной расчетной функцией. Она состоит из четырех частей:

1. Проверка флага is_shut
2. Вызов pre_process для каждого well_facility_iface
3. Вызов process_impl, в котором реализован расчет дебетов и производных.
4. Вызов post_process для каждого well_facility_iface

Функция *clear_data* используется для очистки данных. В случае перегрузки в наследниках функция из базового класса должна быть обязательно вызвана.

Для заполнения якобиана используются функции *fill_jacobian*:

1. *fill_jacobian* - функция обхода соединений. Сейчас реализация перенесена в default_well, чтобы избежать некоторых накладных расходов во время вычислений.

Для заполнения правой части используется *fill_rhs*. Как правило ее перегрузка не требуется.

Функция *fill_rows* используется для построения структуры якобиана (только заполнение строк, колонки и значения заполняются в *fill_jacobian*).

Функция *restore_solution* - должна быть определена в конкретной реализации. Для стандартных скажин обновляет bhp по скважине на основе полученного решения.

Функция *custom_init* предназначен для дополнительной инициализации скважина после обработки событий на текущем timestep(сейчас использоваться не должна, вместо нее должна использоваться *pre_large_step*).

Функции *pre_large_step*, *pre_small_step*, *pre_newton_step* могут быть перегружены в конкретных . Используются для предварительной настройки скважин, инициализации данных и/или сохранения состояния скважины.

Функции *restart_small_step*, *restart_newton_step* - парные для *pre_small_step*, *pre_newton_step*. Если в последних было сохранение состояния, то в *restart_** должно быть восстановление этих состояний.

Перфорации (соединения)

Для перфораций создан класс connection (bs_bos_core/include/well_connection.h)

Данный класс содержит:

1. Координаты и другие данные, задаваемые в schedule файле:
    1. Координаты
    2. Статус соединения (status_, открыта или закрыта)
    3. Диаметр, КН, скин
    4. perforation_factor (*Transmissibility factor for the connection*)
    5. Номер блока, в котором располагается перфорация
    6. Направление скважины
    7. Тип соединения (?, сейчас не используется)
    8. Номер сегмента (по умолчанию 0)
    9. Глубину перфорации
2. head_term (?)
3. Текущий bhp в перфорации (cur_bhp)
4. Плотность в перфорации (density)
5. Давление в скважине (bulkp)

Во время расчета данных в скважине, часть данных хранится в перфорации. Доступ к этим данным осуществляется с помощью функций доступа:

1. get_rw_value
2. get_wr_value
3. get_rr_value
4. get_ps_value
5. get_rate_value

Функция compute_factors используется для расчета perforation_factor (см. выше).

Контроль скважин

Контроль скважин осуществляется классом well_controller (bs_bos_core/include/well_controller.h)

Данный класс содержит:

1. Ограничения для скважины (rate)
2. Ограничение на bhp (bhp_)
3. Историческое значение bhp (bhp_history_, не используется на данный момент)
4. Тип нагнетаемого флюида (injection_type_, сейчас только вода)
5. Состояние контроллера (нагнетание/добыча, управление по BHP/rate, тип добычи).
6. Сохраненные состояния

Следующие функции используются для установки данных при обработке schedule файла:

1. *set_rate*
2. *set_bhp*
3. *set_bhp_history*
4. *set_main_control*
5. *set_injection_type*

Функция *switch_to_bhp* используется для явного переключения контроля на BHP. Функция *check* проверяет ограничения у скважины и переключает контроль. Если было переключение контроля возвращает **true**.

Фабрики

Объекты скважин (calc_well и наследники), контролей (well_controller), перфораций (wells::connection) создаются с помощью фабрик.

Объекты фабрик хранятся в классе reservoir (bs_bos_core/include/reservoir.h) и могут быть заменены для создания объектов конкретного типа.

Фабрика скважин и перфораций - well_factory (bs_bos_core/include/calc_well.h)

1. *create_well* - создание скважины
2. *create_connection* - создание перфорации

Фабрика контроллеров - well_controller_factory (bs_bos_core/include/well_controller.h)

1. *create_controller* - создание контроллера

Как сделать свою реализацию скважин

1. Создать класс наследник для
    1. well
    2. wells::connection
    3. well_factory
    4. well_controller_factory

1. В наследнике класса well необходимо:
    1. Выделить (и/или определить) место для хранения расчетных данных
    2. Реализовать *process_impl*, *clear_data*, *restore_solution*
    3. Если необходимо реализовать *custom_init* (*pre_large_step*)
    4. Если необходимо реализовать *pre_small_step*, *pre_newton_step*, *restart_small_step*, *restart_newton_step*
2. В наследнике connection необходимо:
    1. Выделить (и/или определить) место для хранения расчетных данных
    2. Реализовать *get_rw_value*, *get_wr_value*, *get_rr_value*, *get_ps_value*, *get_rate_value*
3. В наследниках well_factory, well_controller_factory переопределить соответствующие функции

Расчет данных

После окончания расчета данных и до заполнения якобиана данные можно получить следующим образом:

1. connection::get_rate_value - вернет данные по дебету
2. connection::get_rr_value - вернет блок данных (N_PHASES * N_PHASES):

	Sg	So	P
water			
gas			
oil			

   Данные по Sw могут быть получены с помощью функции connection::get_ps_value, но по окончанию расчета данных столбец по So уже содержит данные по Sw.

3. connection::get_rw_value/get_wr_value (?)

Wefac (perforation_factor, см. выше) должен быть применен только в данным по дебету.

PS. Технические детали

В c++ и питоне well_factory и well_controller_factory доступны через объект reservoir. Пример доступа в питоне:

```
import bs
r = bs.bs_bos_core.reservoir_d ()
r.well_factory
r.well_controller_factory

#inherit well_rate_control_factory
f = bs.bs_bos_core.well_rate_control_factory_wrapper_di
class factory (f) :
    def __init__ (self) :
        f.__init__ (self)
    def create_control (self, type, is_bhp, is_prod, calc_model) :
        return None

#set new factory in reservoir
factory_ = factory ()
r.well_controller_factory.set_rate_control_factory (factory_)
```

Документ может содержать неточности или не полностью покрывать функционал скважин.