



Python в ПК tNavigator

2019

Python в tNavigator

Python – высокоуровневый, интерпретируемый язык программирования, ориентированный на повышение производительности разработчика и читаемости кода

Python – язык программирования, который позволяет работать быстро и осуществлять интеграцию систем более эффективно

- В tNavigator Python лежит в основе инструментов:
 - Калькулятор графиков в модулях:
 - GUI симулятора
 - Адаптация и Оптимизация
 - Дизайнер Моделей
 - Workflow в модулях:
 - Дизайнер Геологии
 - Дизайнер Моделей
 - Адаптация и Оптимизация

Калькулятор графиков

- Предназначен для создания дополнительных графиков
- Дополнительный инструмент анализа моделей
- Синтаксис и логика языка Python
- Библиотека функций для работы с графиками
- Импорт библиотек
- Доступен в:



GUI симулятора



АНМ



Дизайнер моделей

Объекты:	WGVT	WGVIR	WGPI	WGITH	WGIT	WGIRWEF
Well	WGIRT	WGIRH	WGIR	WECONWGR	WBULKP	WHCON
Group	WWORH	WWOR	WOWRH	WOWR	WOVIT	WOVIR
Conn	WOPI	WOITH	WOIT	WOIRWEF	WOIRH	WOIR
Fip	WBP	WSTATH	WPIO	WOVPT	WOVPR	WOPTH
Field	WOPT	WOPRWEF	WOPRT	WOPRH	WOPR	WOPP
	WOMT	WOMR	WBPO	WCPT_1	WCPR_1	WCOMT_1
	WCOMR_1	WCMT_1	WCMPR_1	WCMT_1	WCMT_1	WCGMT_1
	WCGMR_1	WOGRH	WOGR	WGORH	WGOR	WWMPT_3
	WWMPT_3	WWMIT_3	WWMIR_3	WCPT_3	WCPR_3	WODN

Окно калькулятора

Управление
списком

Список
скриптов

Выбор типа
объекта
для доступа
к векторам

Калькулятор графиков

Скрипты: + X [иконка] [иконка]

Редактор кода: Вычислить

Функции:

- Объект скважина
- Объект группа
- Объект инт.перфорации
- Объект модель
- Объект врем. шаг
- Объект график
- Объект ОТЧ.РЕГ.
- Глобальные функции

Единицы:
Безразмерная величина ()

Автоматический экспорт Вид (дерево/табл. парам.)

Объекты:

- Well
- Group
- Conn
- Fip
- Field

WGVIT	WGVIR	WGPI	WGITH	WGIT	WGIRWEF
WGIRT	WGIRH	WGIR	WECONWGR	WBULKP	WMCON
WWORH	WWOR	WOWRH	WOWR	WOVIT	WOVIR
WOPI	WOITH	WOIT	WOIRWEF	WOIRH	WOIR
WBP	WSTATH	WPIO	WOVPT	WVPR	WOPTH
WOPT	WOPRWEF	WOPRT	WOPRH	WOPR	WOPP
WOMT	WOMR	WBPO	WCPT_1	WCPR_1	WCOMT_1
WCOMR_1	WCMT_1	WCMPR_1	WCMIT_1	WCMIR_1	WCGMT_1
WCGMR_1	WGRH	WGR	WGORH	WGOR	WWMPT_3
WWMPR_3	WWMIT_3	WWMIR_3	WCPT_3	WCPR_3	WODN

Выбор
функций

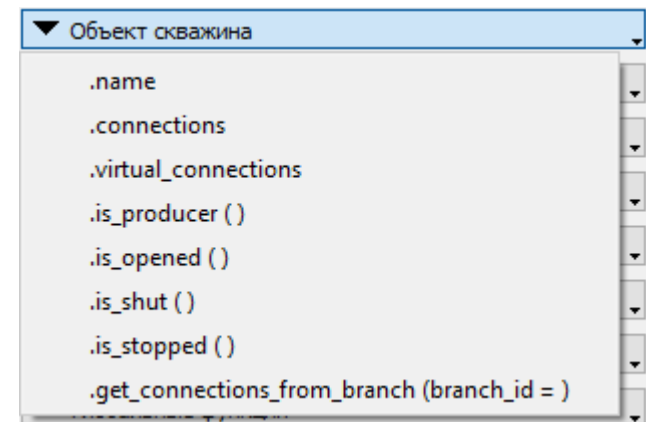
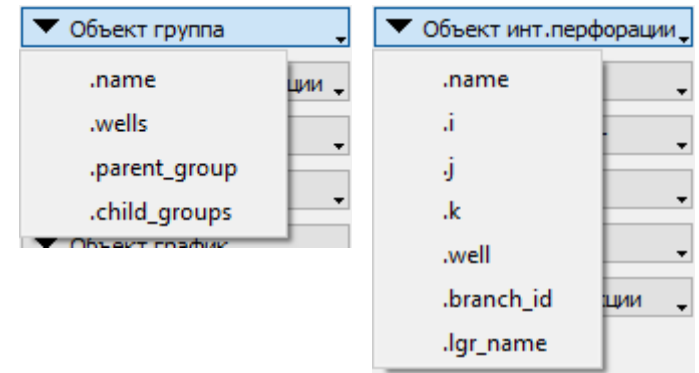
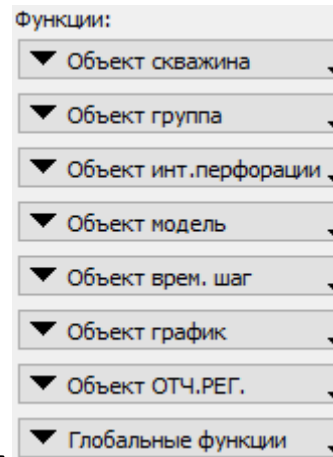
Список
доступных
единиц
измерения

Список доступных векторов
для выбранного типа объекта

Объекты в калькуляторе

➤ Типы объектов

- График
- Модель
- Временной шаг
- Скважина
- Группа
- Ячейка перфорации
- Отчетный регион
- Месторождение

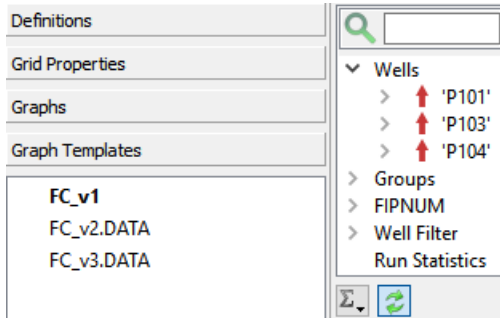


- Каждый объект имеет определенный набор свойств
- Функции для работы с объектами и получения доступа к свойствам объектов

Доступ к векторам и графикам

- Вектора - многомерные массивы
- Массивы индексируются объектами в зависимости от типа вектора:
 - **Скважины** – модель, временной шаг, скважина
 - **Группа** – модель, временной шаг, группа
 - **Ячейка перфорации** – модель, временной шаг, ячейка перфорации
 - **Отчетный регион** – модель, временной шаг, отчетный регион
 - **Месторождение** – модель, временной шаг
- Пример:
wopr[m,w,t] – значение дебита нефти по скважине **w** в модели **m** в момент времени **t**.
- Порядок задания объектов не важен. **wopr[m,w,t]** аналогично **wopr[w,m,t]**
- Если задать не все объекты, то получается соответствующее подмножество:
wopr[m,w] возвращает график дебита нефти по скважине **w** в модели **m** по всем временным шагам.

Доступ к векторам и графикам



Модель	Скважина	Отчетные шаги					
		01.08.2018	01.01.2019	01.01.2020	01.01.2021	01.01.2022	01.01.2023
FC_v1	P101	0.00	3638.63	9342.79	13009.09	15436.22	17075.52
	P103	0.00	2308.29	5854.74	8060.11	9494.40	10450.70
	P104	0.00	486.28	1088.51	1410.33	1609.24	1740.39
FC_v2.DATA	P101	0.00	3304.40	8312.86	11310.21	13161.12	14328.44
	P103	0.00	2635.50	6586.91	8918.24	10347.68	11243.56
	P104	0.00	1359.49	3276.61	4359.74	5008.73	5411.30
FC_v3.DATA	P101	0.00	3015.50	7588.70	10320.34	12007.80	13073.29
	P103	0.00	1831.86	4468.16	5984.49	6903.13	7476.74
	P104	0.00	2453.47	6047.06	8185.98	9505.09	10336.08

Выражение:

Результат:

m=FC_v2.DATA, w=P104, t=01.01.2023
wopt[m,w,t]

5411.30

m=FC_v1, w=P101
wopt[m,w]

01.08.2018	01.01.2019	01.01.2020	01.01.2021	01.01.2022	01.01.2023
0.00	3638.63	9342.79	13009.09	15436.22	17075.52

w=P103
wopt[w]

Модель	Отчетные шаги					
	01.08.2018	01.01.2019	01.01.2020	01.01.2021	01.01.2022	01.01.2023
FC_v1	0.00	2308.29	5854.74	8060.11	9494.40	10450.70
FC_v2	0.00	2635.50	6586.91	8918.24	10347.68	11243.56
FC_v3	0.00	1831.86	4468.16	5984.49	6903.13	7476.74

Примеры использования

Пример 1

Модель многопластовой залежи с совместным фондом скважин.

Необходимо разделить добычу по пластам и вывести соответствующие графики.

В модели 3 пласта. Диапазоны слоев:

- **1 пласт – [1;10]**
- **2 пласт – [11;25]**
- **3 пласт – [26;35]**

Пример 1

#Задание диапазона слоев

```
zones=[10, 25, 35]
```

#Цикл по всем зонам

```
for i in range(len(zones)):
```

#Создание графиков

```
layer_liq_rate=graph (type = 'well', default_value =0)
```

```
layer_oil_rate=graph (type = 'well', default_value =0)
```

```
if i==0:
```

```
    k1=1
```

```
    k2=zones[i]
```

```
else:
```

```
    k1=zones[i-1]+1
```

```
    k2=zones[i]
```

#Расчет векторов

```
for w in get_all_wells ( ):
```

```
    for c in w.connections:
```

```
        if (c.k>=k1) and (c.k<=k2):
```

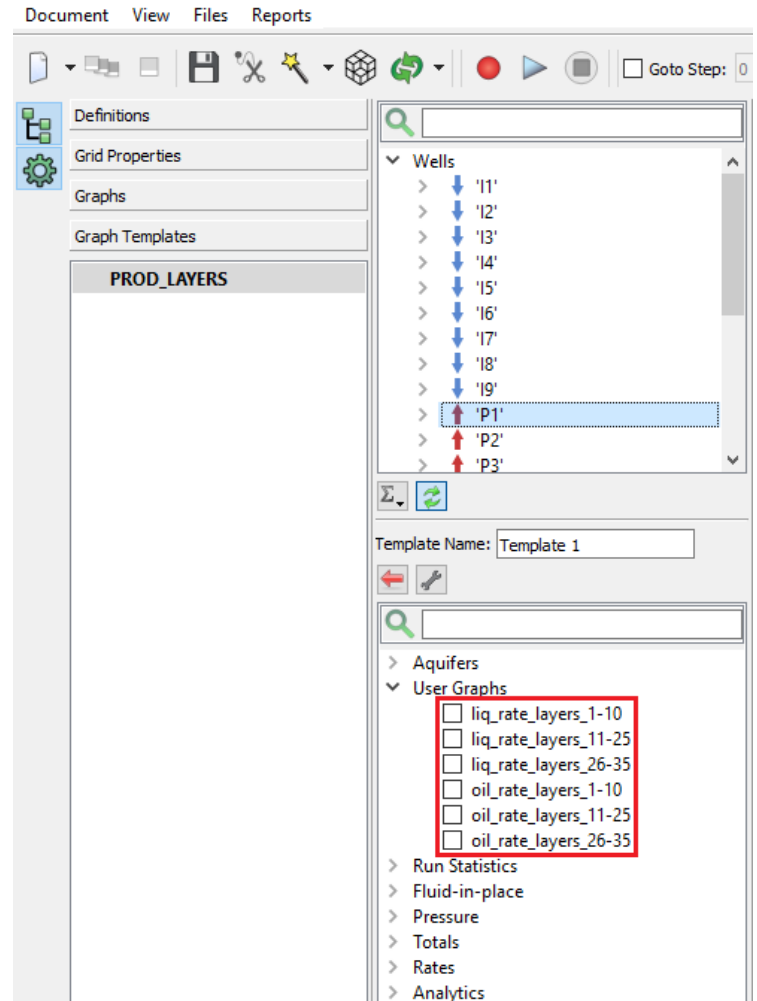
```
            layer_liq_rate[w]+=c.lpr[c]
```

```
            layer_oil_rate[w]+=c.opr[c]
```

#Экспорт векторов

```
export(layer_liq_rate,name='liq_rate_layers_'+str(k1)+'-'+str(k2),units="liquid_surface_rate")
```

```
export(layer_oil_rate,name='oil_rate_layers_'+str(k1)+'-'+str(k2),units="liquid_surface_rate")
```



Пример 2

Необходимо получить список скважин, которые находятся в работе на заданный момент времени.

```

c1=0
c2=0
#Выбор временного шага
for t in get_all_timesteps ( ):
    if t.name=="01.07.2011":
        c1=1
#Цикл по всем моделям и скважинам
    for m in get_all_models ( ):
        for w in get_all_wells ( ):
#Проверка статуса. Если скважина доб. или нагн.
            if wstat[m,w,t]==1 or wstat[m,w,t]==2:
#Печать имени скважины
                print(w.name)
                c2+=1
#Проверка заданной даты
if c1==0:
    print("No such date in the model")
#Проверка наличия скважин на заданную дату
elif c2==0:
    print("No open wells on date "+str(t.name))

```

Редактор кода:
Вычислить

```

c1=0
c2=0
for t in get_all_timesteps ( ):
    if t.name=="01.07.2011":
        c1=1
        for m in get_all_models ( ):
            for w in get_all_wells ( ):
                if wstat[m,w,t]==1 or wstat[m,w,t]==2:
                    print(w.name)
                    c2+=1
if c1==0:
    print("No such date in the model")
elif c2==0:
    print("No open wells on date "+str(t.name))

```

```

P1
P2
I1
P3
P4
I2

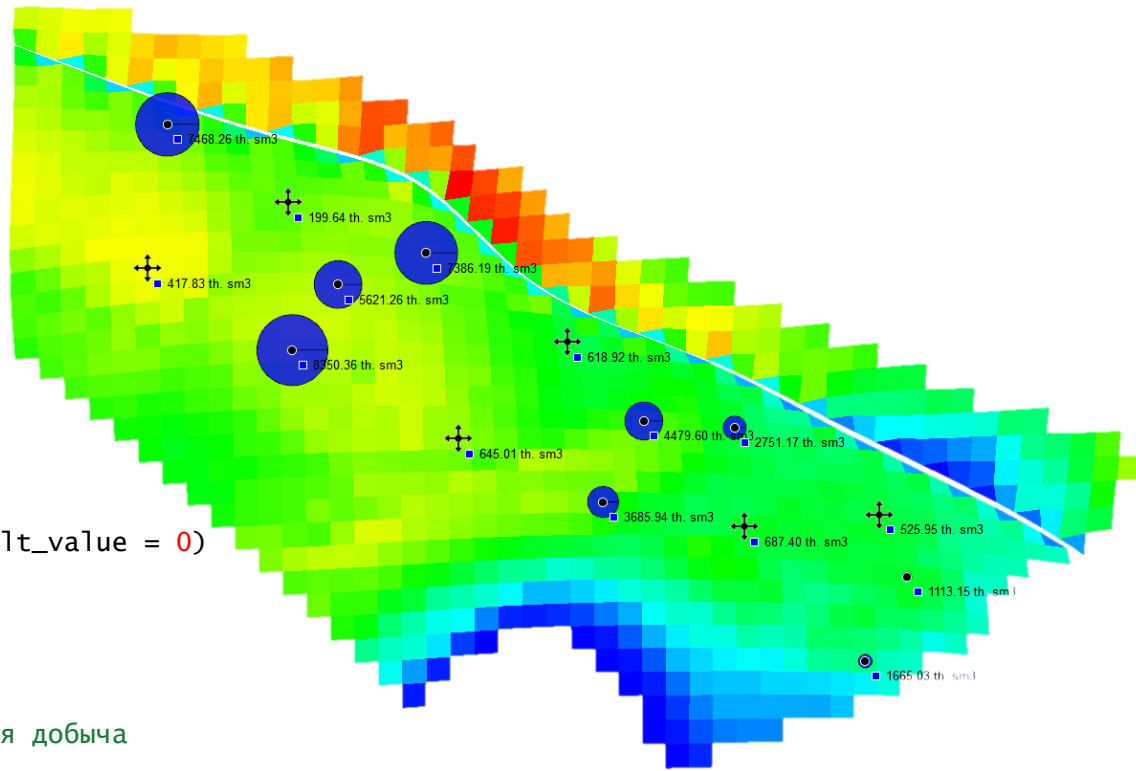
Complete!

```

Пример 3

На круговых диаграммах необходимо отобразить добычу газа скважинами в первые 2 года их работы

```
#Импорт библиотеки datetime
import datetime
#Создание нового графика
gas_prod_2_years=graph (type = 'well', default_value = 0)
#Цикл по всем моделям
for m in get_all_models ( ):
#Цикл по всем скважинам
    for w in get_all_wells ( ):
        start=0
#Поиск временного шага, с которого начинается добыча
        for t in get_all_timesteps ( ):
            if start==0:
                if wgpt[m,w,t]>0:
                    tend=tprev.to_datetime ( )
#Определение временного шага через 2 года с момента начала добычи
                    t_end=get_timestep_from_datetime (datetime.date(tend.year+2,tend.month,tend.day), mode = 'nearest')
                    start=1
                    tprev=t
            if start==1:
#Заполнение графика единым значением
                for t in get_all_timesteps ( ):
                    gas_prod_2_years[m,w,t]=wgpt[m,w,t_end]
#Экспорт графика
export (gas_prod_2_years, name = 'Gas Total for the first 2 years', units = "gas_surface_volume")
```

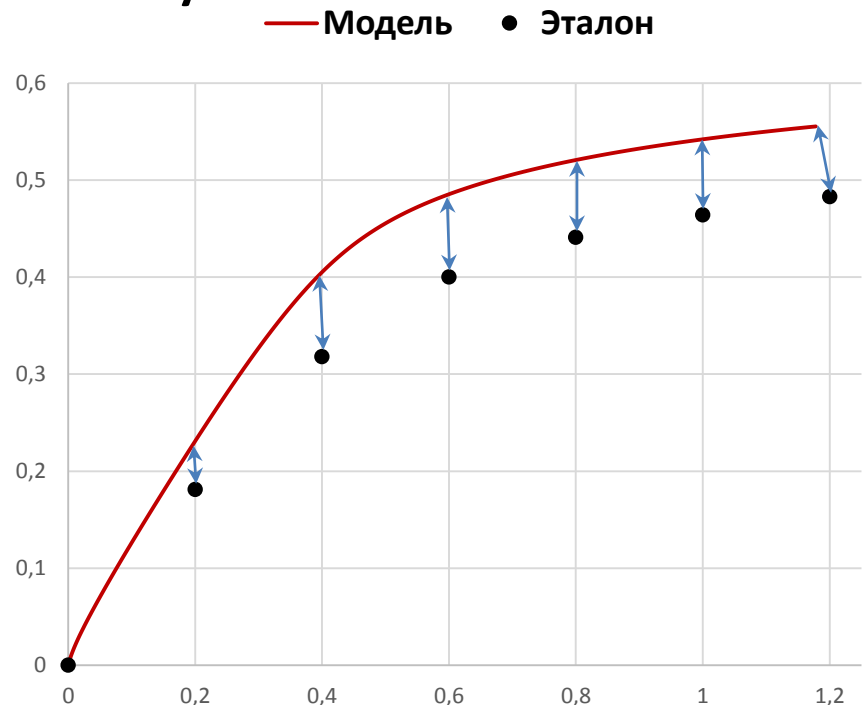


Пример 4

Необходимо настроить модель по результатам испытаний разведочных скважин и характеристике вытеснения объекта-аналога.

Для настройки модели на характеристику вытеснения ЦФ должна рассчитываться как сумма квадратов отклонений характеристики «КИН-прокачка» от заданной по объекту-аналогу.

Характеристика задана в виде таблицы. Отклонения нужно рассчитывать между ближайшими точками (КИН, прокачка) объекта аналога и (КИН, прокачка) модели, т.к. значения прокачки в таблице могут не совпадать с расчетными.



Расчет ЦФ

```
#Задаем значение запасов и порового объема
oir = 3371236
pv = 5425100
#Расчет векторов КИН и прокачка
kin = FOPT / oir
prok = FWIT / pv

#Имеющуюся эталонную таблицу необходимо представить в виде двух массивов
ethalon_kin = [0.181, 0.318, 0.4, 0.441, 0.464, 0.483]
ethalon_prok = [0.2, 0.4, 0.6, 0.8, 1.0, 1.2]

#Создается график objective, который по сути и будет являться итоговой целевой функцией
objective = graph (type = 'field', default_value = 0)

#Цикл по всем моделям проекта
for m in get_all_models ():
    #Цикл по всем элементам эталонной характеристики
    for i in range (len (ethalon_kin)):
        et_kin = ethalon_kin[i]
        et_prok = ethalon_prok[i]
        min_diff = None
        min_t = None
    #Цикл по всем временным шагам. Производится поиск ближайшей точки к i-ой эталонной по значению прокачки
    for t in get_all_timesteps ():
        curr_prok = float (prok[m, t])
        curr_diff = abs (curr_prok - et_prok)
        if min_diff == None or curr_diff < min_diff:
            min_diff = curr_diff
            min_t = t
    #Добавляем квадрат отклонения для текущей точки к значению целевой функции
    objective[m] += (float (kin[m, min_t]) - et_kin) ** 2 + (float (prok[m, min_t]) - et_prok) ** 2

#Для первого шага задаем значение 0
first_step=[*get_all_timesteps ()][0]
objective[first_step] = 0
#Экспорт графика
export (objective, name = 'Objective function', units = "no")
```

Адаптация модели

➤ Задать целевую функцию:

1. Тип функции – оптимизация на прогнозе
2. Объект – пользовательский график
 - a. Параметр – созданный график Objective function
 - b. Тип – минимизация
3. Шаги – с нулевого по последний

➤ Создать эксперимент Дифференциальная эволюция

➤ Запустить расчеты

Истогр. диаграмма Составная диаграмма Анализ Mds Cdf Прокси-модели Целевые функции

Пользовательская целевая функция

Тип функции: Оптимизация на прогнозе

Добавить слагаемое Добавить несколько слагаемых Удалить слагаемое

Слагаемое	Тип объекта	Параметр	Вес
1	Польз. график по месторожд.	Objective function	1

Объекты

Тип: Польз. график по месторожд.

Всего добывающие
Всего нагнетательные

Весовой параметр: Равно Вычислить

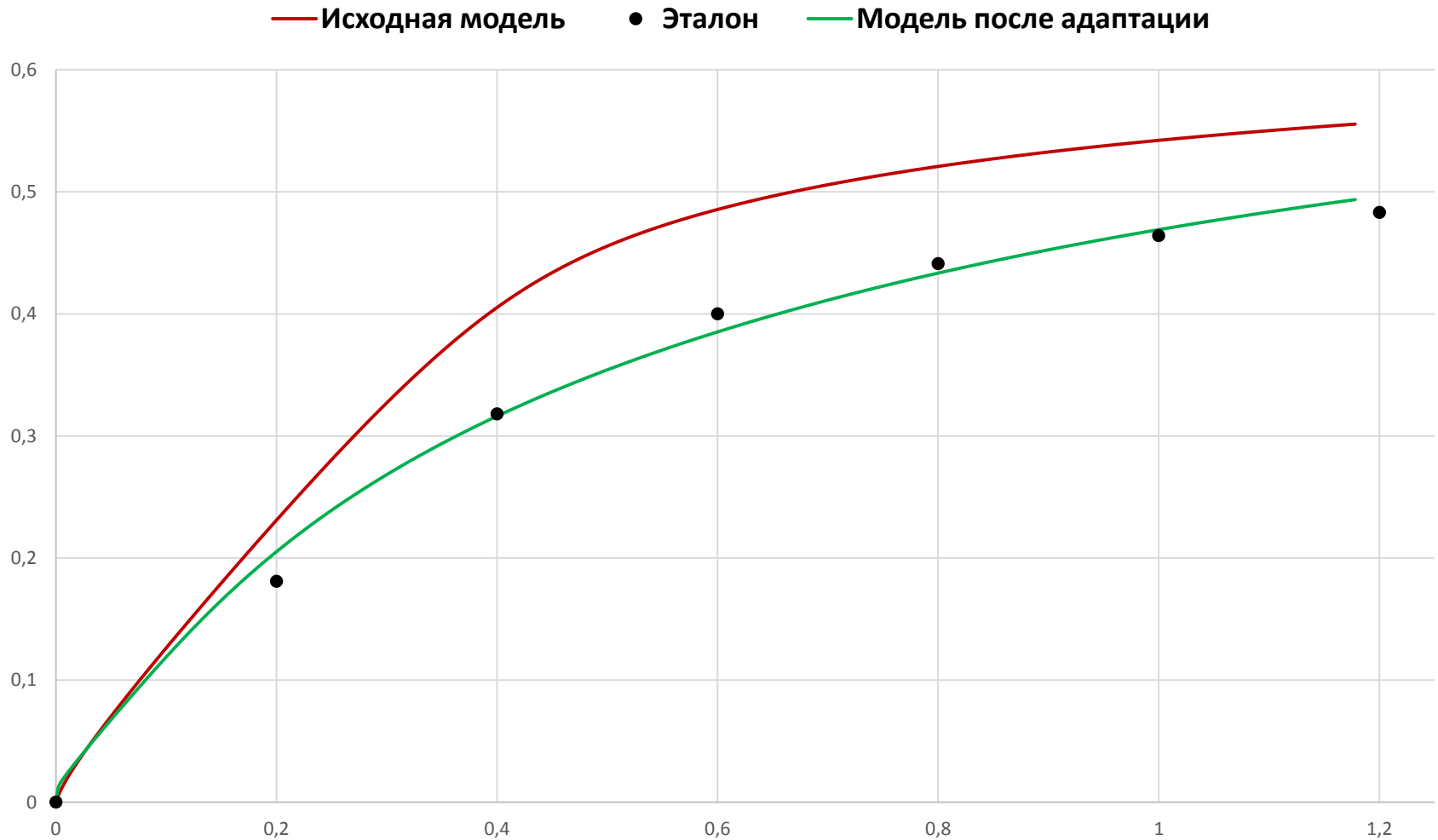
Параметр	Тип	Точность дебита, %
<input checked="" type="checkbox"/> Objective function	Минимизировать накопленное	

С вр.шага: 0 01.01.2010

До вр.шага: 600 01.01.2060

Ok

Результат



Пример 5

Необходимо средствами модуля АНМ настроить модель на трассерные исследования

➤ Исходные данные:

- Добыча трассера по скважинам на конкретную дату

➤ Пример модели:

- 11x11x10 ячеек
- 4 добывающих скважины
- 1 нагнетательная скважина
- Для настройки на добычу трассера
изменяется множитель на проницаемость
в точках скважины (для межскважинного
пространства множитель интерполируется)
- Для работы с вектором добычи трассера создана UDQ:

UDQ

```
DEFINE WUTRTOT WPTTR1 /
```

```
/
```

Задание модификаторов

Определение модификаторов: множитель на проницаемость по скважинам

```
DEFINES
'PERMMULT_P1'      1      0.1      10      /
'PERMMULT_P2'      1      0.1      10      /
'PERMMULT_P3'      1      0.1      10      /
'PERMMULT_P4'      1      0.1      10      /
/
```

Использование модификаторов в арифметике

```
ARITHMETIC
--Создание куба множителей путем интерполяции
ARRMPERMSRC=-999.25
ARRMPERMSRC=IF(BLOCK(50,50)==1,@PERMMULT_P1@,ARRMPERMSRC)  --WELL P1
ARRMPERMSRC=IF(BLOCK(1050,50)==1,@PERMMULT_P2@,ARRMPERMSRC)  --WELL P2
ARRMPERMSRC=IF(BLOCK(1050,1050)==1,@PERMMULT_P3@,ARRMPERMSRC)  --WELL P3
ARRMPERMSRC=IF(BLOCK(50,1050)==1,@PERMMULT_P4@,ARRMPERMSRC)  --WELL P4
ARRPERMMULT=interpolate_m1_idw(ARRMPERMSRC,-999.25,1,3)
--Модификация куба проницаемости
PERMX=PERMX*ARRPERMMULT
PERMY=PERMX
PERMZ=PERMX/10
/
```

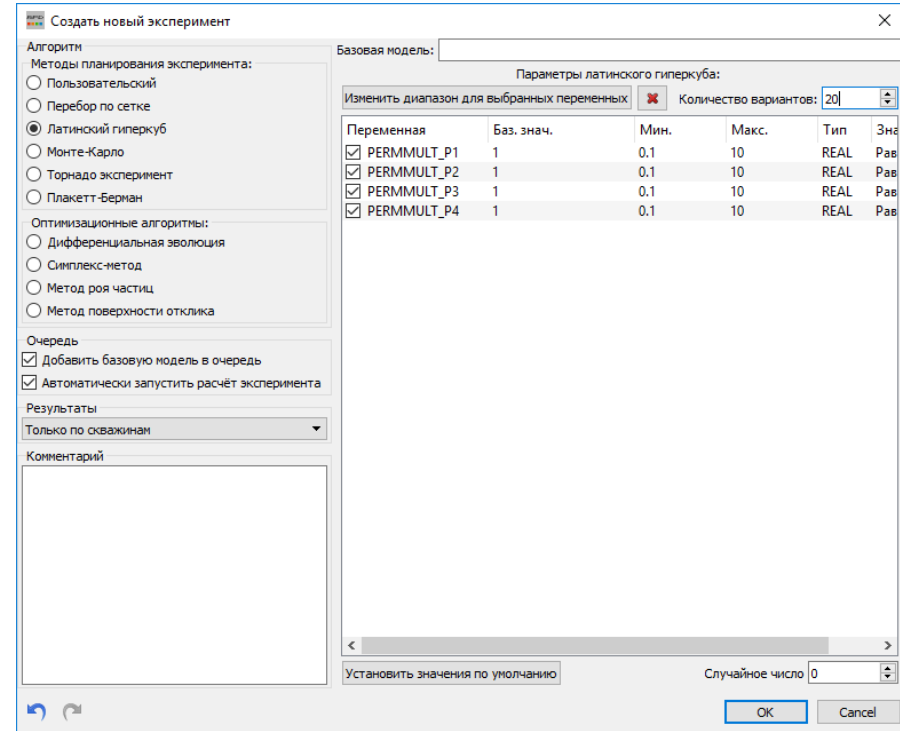
Создание проекта

➤ Открыть файл модели и запустить эксперимент Латинский Гиперкуб

- 20 расчетов
- Диапазон множителей на проницаемость: [0.1;10]

➤ Добавить замеры добычи трассера в проект:

- Используется калькулятор графиков (см. следующий слайд)
- Создается график накопленной добычи трассера – расчетные значения + исторические точки
- Создается целевая функция для адаптации модели



Дата замера	Номер скв.	Добыча трассера, м ³
01.01.2018	P1	78
	P2	193
	P3	812
	P4	1020

Создание проекта

➤ Открыть калькулятор графиков и добавить скрипт

```

import copy
import math
#Замеры добычи трассера вводятся в виде двух массивов одинаковой длины:
#wnam – имя скважины, tracer_totals – накопленная добыча
wnam=['P1', 'P2', 'P3', 'P4']
tracer_totals=[78,193,812,1020]
objective=graph (type = 'field', default_value = 0)
#Определяется историческая модель
history_model = [*get_all_models ()][0]
#Создание вектора TR1TOT – накопленная добыча трассера, где wutrtot – UDQ
TR1TOT=copy.copy (wutrtot)
for m in get_all_models ( ):
    for t in get_all_timesteps ( ):
#На дату 01.01.2018
        if t.name == "01.01.2018":
#Цикл по всем фактическим замерам
            for i in range (len (wnam)):
                w=get_well_by_name(wnam[i])
#Расчет целевой функции
                objective[m]+=100*math.fabs(tracer_totals[i]-wutrtot[t,w,m])/tracer_totals[i]
#Добавление исторических точек на график TR1TOT
                TR1TOT[history_model,w,t]=tracer_totals[i]

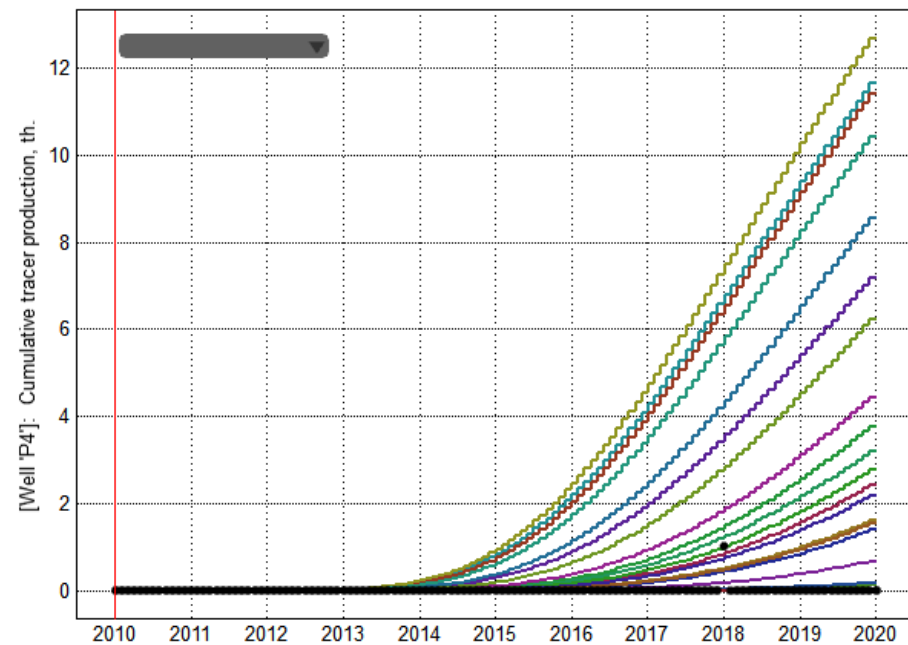
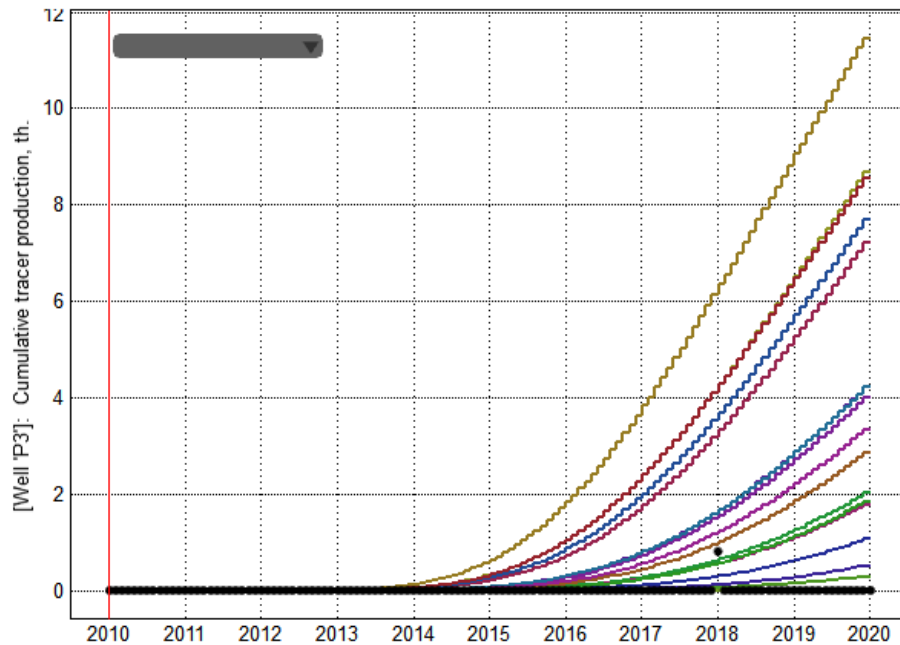
first_step=[*get_all_timesteps ()][0]
objective[first_step] = 0

#Экспорт целевой функции и графика накопленной добычи трассера
export (objective, name = 'OBJFUNC', units = 'no')
export (TR1TOT, name = 'Cumulative tracer production', units = 'liquid_surface_rate')

```

Создание проекта

- **Графики накопленной добычи трассера с историей:**
 - Расчеты «покрывают» исторические точки
 - Можно переходить к автоматической адаптации



Адаптация модели

➤ Задать целевую функцию:

1. Тип функции – оптимизация на прогнозе
2. Объект – пользовательский график
 - a. Параметр – созданный график OBJFUNC
 - b. Тип – минимизация
3. Шаги – с нулевого по последний

➤ Создать эксперимент Дифференциальная эволюция

➤ Запустить расчеты

Целевые функции

Пользовательская целевая функция

Тип функции: Оптимизация на прогнозе

Добавить слагаемое Добавить несколько слагаемых Удалить слагаемое

Слагаемое	Тип объекта	Параметр	Вес
1	Польз. граф...	OBJFUNC	1

Объекты

Тип: Польз. график по месторожд.

Все добывающие
Все нагнетательные

Весовой параметр: Равно Вычислить

Параметр	Тип	Точность дебита, %
<input checked="" type="checkbox"/> OBJFUNC	Минимизировать накопленное	

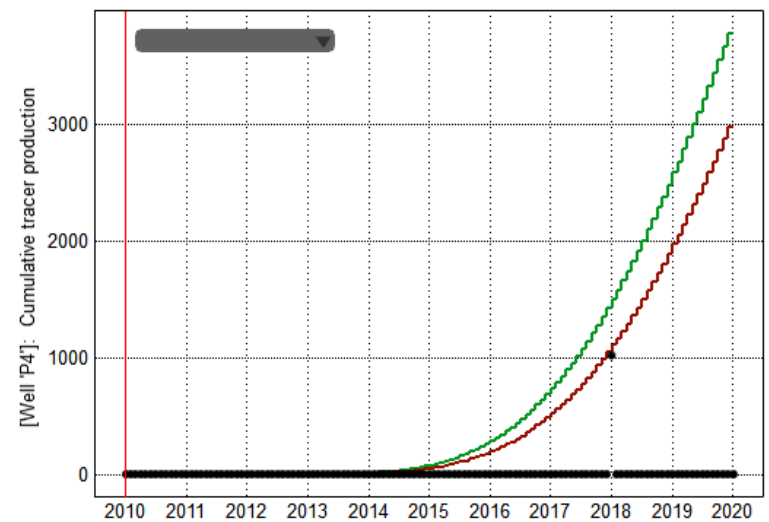
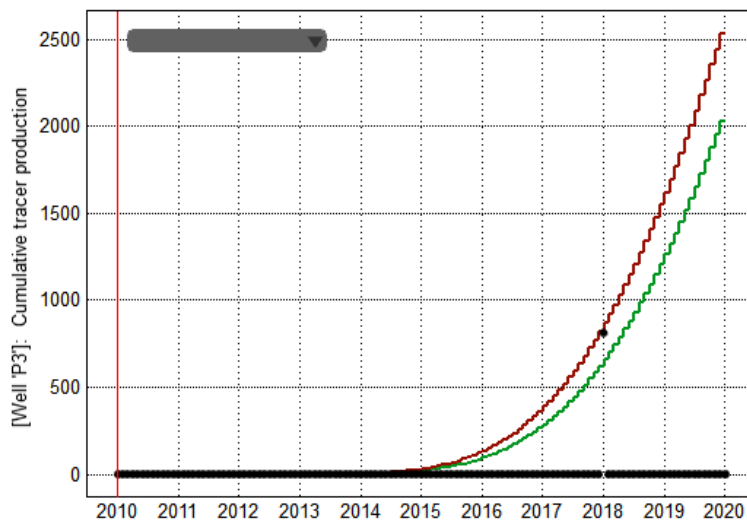
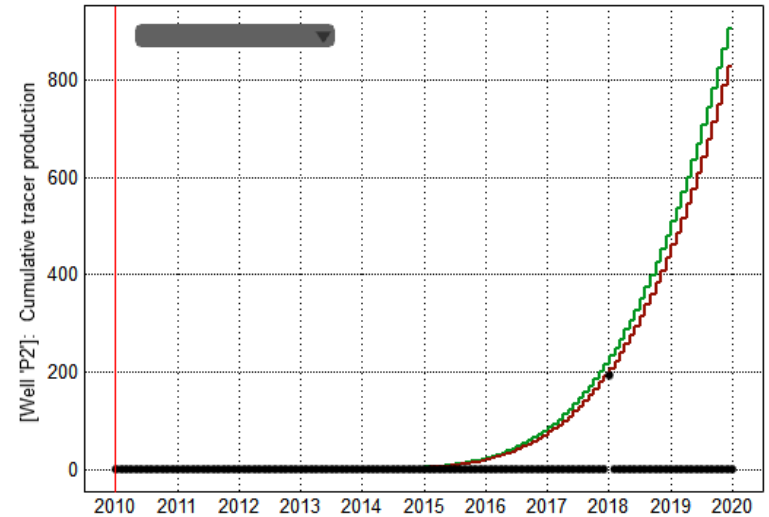
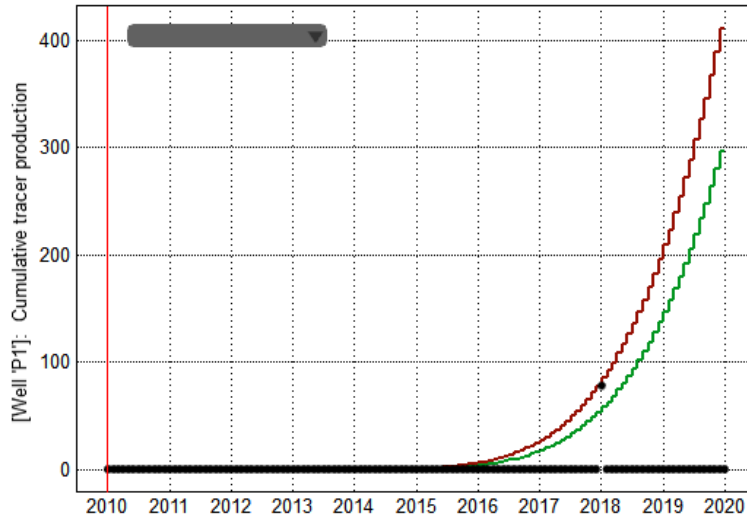
С вр.шага: 0 01.01.2010

До вр.шага: 120 01.01.2020

Ok

Результат

➤ Сравнение результатов базового расчета и расчета с наилучшим качеством:



- Базовый расчет

- Расчет с наилучшим качеством

Пример 6

Расчет экономических параметров:

- **Расчет параметров, необходимых для экспорта в Excel**
 - **Добыча нефти в тоннах**
 - **Добыча жидкости в тоннах**
 - **Фонд скважин (действующие и бездействующие)**
- **Экспорт рассчитанных параметров в Excel**
- **Пересчет экономических параметров в Excel на основе экспортированных данных**
- **Импорт результатов расчета Excel обратно в калькулятор**
- **Создание графиков для визуализации в tNavigator**

**Алгоритм можно использовать в
задачах оптимизации и оценки рисков**

Спасибо за внимание!

Rock Flow Dynamics
www.rfdyn.ru
tnavigator@rfdyn.ru