

Технология сбалансированной идентификации математических моделей (SvF): установка системы, подготовка файла-задания, примеры. (SvF_Linux_UserGuide_Rus_v.28)

Технология сбалансированной идентификации математических моделей (т.н. SvF-технология, от Simplicity vs Fitting) представляет собой перспективное направление прикладной математики, сочетающее методы структурного математического моделирования, оптимизации, адаптивной регуляризации и распределенных вычислений. Приводится описание принципов функционирования и практического применения SvF-технологии.

Введение

Технология сбалансированной идентификации математических моделей (**SvF** - Simplicity vs Fitting) предназначена для идентификации математических моделей на основе техник оптимизационного моделирования.

В основе технологии следующий взгляд на научное исследование:

- проекция объективной реальности в обозримый набор математических моделей,
- сравнение указанных моделей с целью выбрать ту, которая лучше других аппроксимирует (описывает) объективную реальность.

Построение математической модели можно разбить на этапы:

- формулировка целей исследования;
- построение общей (концептуальной) модели;
- анализ экспериментальных данных, включая планирование новых экспериментов;
- выбор (аналитической) математической модели и ее исследование;
- выбор конечномерной численной, аппроксимирующей, возможно бесконечномерную математическую модель (с неизвестными функциями непрерывного аргумента);
- идентификация (определение параметров и зависимостей) модели на экспериментальных данных;
- верификация построенной модели, оценка погрешностей;
- анализ построенной модели, определение ее области применимости,
- использование построенной модели, например, для прогноза поведения исследуемого явления или объекта.

Указанный план, вообще говоря, требует много времени, знаний в различных областях математики и навыков программирования. Особенно “технические” этапы, начиная с третьего.

Целью технологии является освободить, по возможности, исследователя от рутины (знания и навыки не отменяются). В идеале, ему достаточно: собрать и сгруппировать имеющиеся данные измерений; сформировать математическое описание (модель); определить значения неизвестных параметров и/или функциональных зависимостей (идентификация); построить графики и оценки погрешностей описания для выбранной модели. После завершения расчета по выбранной модели он анализирует результаты. Если они не устраивают исследователя, то он может привлечь дополнительные данные или модифицировать модель (добавить/сократить параметры, изменить состав уравнений и т.д.). Затем можно провести расчет по измененной модели и количественно сравнить результаты с тем, что было получено для предыдущей модели.

Имеется отдаленное сходство SvF-технологии с методами «глубокого обучения», с одним важным отличием: в SvF-технологии «обучаются» не нейронные сети, достаточно универсального типа, а конкретные, структурные модели (включающие, например, системы интегро-дифференциальных уравнений), учитывающие представления исследователя об «устройстве» и принципах поведения/функционирования изучаемого явления.

1. Требования к ПО

- OS Linux
- Python 3.7.4+ (для экономии места на дисках, рекомендуется использовать реализации интерпретатора языка и менеджера пакетов Python в системе Miniconda, <https://docs.conda.io/en/latest/miniconda.html>)
- numpy 1.6.*
- matplotlib 1.5.*
- Pyomo 5.6.*, <http://www.pyomo.org/>
- Решатель нелинейных задач математического программирования Ipopt 3.12.*, <https://github.com/coin-or/Ipopt>
- Для проведения масштабных расчётов желательно использование платформы Everest, <http://everest.distcomp.org/> в частности, приложения SSOP, <https://optmod.distcomp.org/apps/vladimirv/solve-set-opt-probs>, позволяющего одновременно решать набор оптимизационных задач на вычислительных ресурсах, подключённых к системе <https://optmod.distcomp.org>. Для этого нужно будет зарегистрироваться на сайте <https://everest.distcomp.org/> или <https://optmod.distcomp.org>.

2. Установка системы

За установочным файлом *SvF.zip* или за доступом к нему через *Gitlab* следует обратиться к Александру Соколову, alexander.v.sokolov@gmail.com.

1. Файл *SvF.zip* распакуйте в любой удобный для Вас каталог.
2. Файл *runSvF28.sh* из каталога *SvFp* скопируйте в */home/ВашеИмяПользователя/bin*. Затем откройте его в любом текстовом редакторе, найдите там строку «*python /home/sokol/SvF/Lib28/_START28.py*» и укажите правильное месторасположение файла *_START28.py* в соответствии с каталогом, выбранным в п. 1.
3. Для установки свободно доступных пакета Pyomo и решателя Ipopt либо используйте инструкции на сайтах, указанных в разделе 1, либо обратитесь за консультацией к А. Соколову.
4. Для использования внешних распределённых ресурсов необходимо зарегистрироваться на <https://everest.distcomp.org/> или <https://optmod.distcomp.org>.

Для тех, кто пользуется системой контроля версий Git можно скачать систему из общедоступного репозитория <https://github.com/distcomp/SvF>.

3. Принципы функционирования

Общая схема человеко-машинной технологии, реализующий сбалансированный метод идентификации SvF, показано на рисунке 1. Подробное описание технологии можно найти в следующих статьях

1. Sokolov A. V., Voloshinov V. V. *Model Selection by Balanced Identification: the Interplay of Optimization and Distributed Computing* // *Open Computer Science*, 2020, 10 — p. 283–295. <https://doi.org/10.1515/comp-2020-0116>

2. Соколов, А.В.; Волошинов, В.В. Выбор математической модели: баланс между сложностью и близостью к измерениям. *International Journal of Open Information Technologies*, 2018, 6(9) С. 33-41, <http://injoit.org/index.php/j1/article/view/612>

На «верхнем уровне» исследователь должен подготовить файлы данных и файл-задание, содержащий формальное описание модели и спецификации параметров управляющих проведением расчетов. Остальные действия по работе алгоритма (с привлечением внешних дополнительных вычислительных ресурсов или на компьютере исследователя) будут выполнены автоматически технологией SvF.

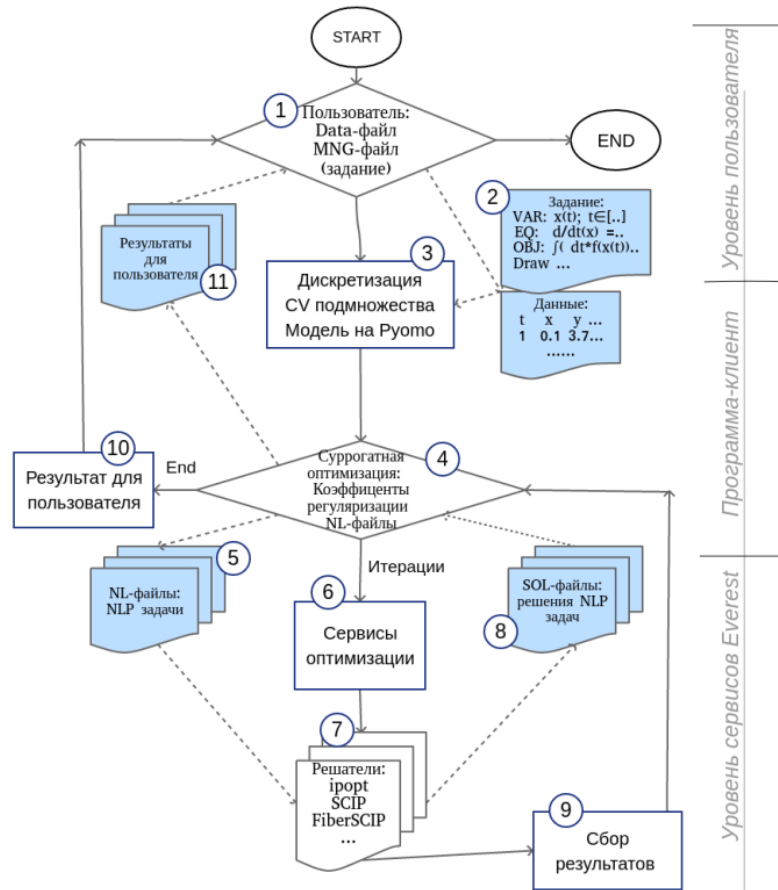


Рис. 1. Технологический цикл SVF-технологии.

Математическое описание и ссылки на файлы данных нужно формализовать в файле вычислительного задания в текстовом формате с расширением *.mng* или в формате Open/Libre Office *.odt*. Далее для краткости, файл вычислительного задания будем называть *mng-файл*.

Для запуска программы в каталоге, содержащем *mng* –файл, в командной строке наберите `$ bash runSvF28.sh`.

Если таких файлов несколько, система предложит осуществить выбор.

Возможен также запуск с явным указанием файла задания:
`$ bash runSvF28.sh ИмяФайла.`

Такой способ предпочтителен, если в каталоге содержится несколько файлов с расширением `.mng` или `.odt`.

В процессе обработки файла-задания система преобразует его содержание в программу на языке Python (StartModel.py+ Model.py). При этом для описания задач математического программирования будут использоваться средства пакета Pyomo (Python Optimization Modeling Objects), <http://www.pyomo.org/>. Исходные данные задач оптимизации в итоге преобразуются в т.н. NL-файлы (*.nl) в формате, принятом в одном из самых распространенных стандартов оптимизационного моделирования AMPL, <https://ampl.com/>. Именно эти файлы отправляются AMPL-совместимым решателям, например, COIN-Or Ipopt. Результаты решения возвращаются в виде т.н. SOL-файлов (*.sol), данные из которых извлекаются также с помощью Pyomo.

Программа выполняется с использованием назначенных вычислительных ресурсов, которые указываются средствами Everest API, <https://gitlab.com/everest/python-api/>, см. пример <https://github.com/distcomp/pyomo-everest/blob/master/ssop/demoUseSsop.py> (ищите `ssop_config.SSOPP_RESOURCES`).

Начальные значения параметров регуляризации считываются из файла, имя которого формируется из имени `mng`-файла заменой расширения на `.res` (далее – `res`-файл).

Если в каталоге имеются файлы решений (*.sol), то они используются в качестве начального приближения.

Результаты расчетов (найденные решения) сохраняются в виде графических (*.png) и текстовых (*.sol) файлов, значения найденных параметров регуляризации и найденные оценки погрешностей сохраняются в – `res`-файле.

4. Файл вычислительного задания

Текстовый `mng`-файл содержит полное формальное описание задачи идентификации неизвестных параметров и функциональных зависимостей математической модели. Этой информации достаточно для численного решения на основе технологии сбалансированной идентификации (SvF-технологии). Для этого используются специальные ключевые слова, определения (декларации) системы SvF, математические выражения в достаточно естественной нотации и (если это необходимо) операторы на языке Python.

С целью повышения наглядности математической задачи, в описания допускается использование некоторых специальных символов (см. Табл1)

Табл. 1. Специальные символы и их имена в программе на языке Python.

'J'	'\\int'
'E'	'\\in '
'V'	'sqrt'
'.'	'*'
'\"'	'apst'
'τ'	'tau'
'μ'	'muu'
'π'	'pi'
'α'	'alpha'
'σ'	'sigma'
'ξ'	'xi'
'Ω'	'Omega'

Порядок следования операторов - произвольный, однако определение объекта должно предшествовать ссылке на него. Нотация является позиционной (как в языке Python). Однако, ключевые слова, заканчивающиеся двоеточием (*Var*;, *Set*;, *EQ*; и др.) начинают блок описаний.

Идущие далее описания наследуют ключевое слово, если они не начинаются с первой позиции. Блок заканчивается на любом операторе (кроме '#'), записанном с первой позиции файла. Кроме того необходимо помнить, что описание целевой функции (или функций) запускает расчеты и следующий за ним код предназначен только для обработки результатов.

При написании параметров системы (параметров, управляющих расчетами) и ключевых слов не имеет значения регистр символов (можно использовать прописные и строчные буквы). Символ '\' означает продолжение текста на следующей строке.

Рекомендуется следующая структура mng-файла:

- Управляющие параметры (например, `ExitStep = 1e-6`);
- Источники данных (измерений) (например, `Select * from ABC.txt where ...`);
- Множества (сетки) дискретизации (например, `SET: t = [t_min, t_max, t_step]`);
- Функции-параметры (например, `PARAM: Q(t)`);
- Функции-неизвестные (например, `VAR: E(t); >=0`);
- Уравнения (например, `EQ: E = Cond (Q,T) ·VPD`)
- Целевая функция (функции) (например, `OBJ: x.Complexity (Penal[0]) + x.MSD()`)
- Обработка результатов (например, `Draw Q`)

Ниже в примерах мы будем постепенно вводить различные конструкции и примеры нотации.

5. Примеры постановки, формализации и решения различных задач.

5.1. Модель движения «Маятник с трением».

Подробно математическая постановка ряда задач выбор модели, описывающей данные, искусственно подготовленные при помощи формулы осциллятора и датчика псевдослучайных чисел, и последовательный анализ моделей приведены в работах (Sokolov, Voloshinov, 2020),(Соколов, Волошинов, 2018), указанных выше. Здесь мы ограничимся краткими постановками и подробным обсуждением их формализации – описанием файла задания. Соответствующие «цифровые» постановки и результаты расчетов находятся в каталогах *SvF/Examples/1-Oscillator* (дистрибутива SvF).

Форматы исходных данных

Исходные данные для идентификации задач находятся в файлах *Spring5.dat* и *Spring5.xlsx*. Фрагмент текстового файла *Spring5.dat*:

t	x_f	x	#SvFver_62_tbl
-1.0		0.0691932442765	-0.0113286164626
-0.825		0.223051530384	0.191218514986
-0.65		0.407965450633	0.379565282838
...			

Первая строка содержит названия столбцов (t x_f x) и символическую кодировку формата (#SvFver_62_tbl). Остальные строки содержат значения соответствующих переменных.

Аналогичные данные в формате MS Excell приведены в фрагменте файла *Spring5.xlsx*:

```

time      position      #END#
-1.0      -0.0113286164626
-0.825    0.191218514986
-0.65     0.379565282838
.          .
2.5       1.28494258671
#END#

```

Первая строка содержит названия столбцов, далее идут соответствующие значения. Комбинация #END# (необязательная) ограничивает размеры таблицы

0-x(t) (full, short, very_short) – функция одного переменного (сплайн-аппроксимация, непараметрическая регрессия).

Задача состоит в поиске функции $x(t)$, аппроксимирующей экспериментальные данные из файла *Spring5.dat*.

Файл-задание (*MNG-full.mng*) на поиск функции $x(t)$:

```

CVNumOfIter = 21          # максимальное кол-во итераций поиска весов регуляризации
CVstep       = 21          # кол-во подмножеств для процедуры кросс-валидации
Select x, t from ../Spring5.dat # считывание столбцов x, t из файла ../Spring5.dat
t_min = -1.0              # левая граница интервала
t_max = 2.5               # правая граница интервала
SET:  T = [t_min, t_max, 0.025] # множество T от t_min до t_max с шагом 0.025
VAR:  x(t); t ∈ T # t \in T # неизвестная функция, заданная на множестве T
OBJ:  x.MSD() + x.Complexity(Penal[0]) # целевая функция – смешанный критерий выбора x(t)
Draw  x                   # отображение найденной функции x(t)
EOF                          # конец файла, все что дальше опускается

```

Смысл операторов кратко поясняется в комментариях (после #).

При выполнении оператора *Select* данные из файла попадают в программу с именами x , t . При описании сеточной функции (декларации *SET*) первый параметр определяет минимальное значение равное t_{\min} , второй – максимальное значение t_{\max} , третий параметр определяет шаг сетки 0.025. Затем при описании неизвестной функции $x(t)$ (оператор *VAR:*) они связываются с такими же именами x , t . Такая связь позволяет рассчитать среднеквадратичное отклонение неизвестной от данных в целевой функции - $x.MSD()$ (оператор *OBJ:*). Другое слагаемое $x.Complexity(Penal[0])$ определяет взвешенный ($Penal[0]$) штраф за кривизну искомой кривой.

Возможна более короткая запись задания в файле *MNG-shot.mng* (красным цветом выделены изменения относительно предыдущей версии):

```

CVNumOfIter = 21          # максимальное кол-во итераций поиска весов регуляризации
CVstep       = 21          # кол-во подмножеств для процедуры кросс-валидации
Select * from ../Spring5.dat # считывание всех столбцов из файла ../Spring5.dat
SET:  t = [  , , 0.025] # множество t от min до max значений t из файла с шагом 0.025
VAR:  x(t)                 # неизвестная функция, заданная на множестве t
OBJ:  x.MSD() + x.Complexity(Penal[0]) # целевая функция – смешанный критерий выбора x(t)
Draw  x                   # отображение найденной функции x(t)
EOF                          # конец файла, все что дальше опускается

```

Знак $*$ в операторе *Select* означает считывание всех данных, в том числе и столбцов t и x . Множество, определяющее область определения обозначается t . Через это имя осуществляется связь со столбцом данных t , что позволяет опустить первые два аргумента (в

операторе *SET*), заменив их на минимальное и максимальное значения их всех значений одноименного столбца.

Еще более короткая запись вычислительного задания приведена в файле *MNG-very_shot.mng*:

```
Select * from ../Spring5.dat      # считывание всех столбцов из файла ../Spring5.dat
VAR:  x ( t )                    # неизвестная функция, заданная на множестве t
OBJ:  x.MSD() + x.Complexity(Penal[0]) # целевая функция – смешанный критерий выбора x(t)
Draw                                     # отображение всех функций
```

Если значения некоторых управляющих параметров не заданы, при выполнении задания они получают значения по умолчанию. Например, для параметра *CVstep*, определяющего число частей (подмножеств), на которые разбивается набор экспериментальных данных при перекрестной проверке (cross validation) это значение 7 частей. Другой опущенный параметр *CVNumOfiter* (максимальное кол-во итераций поиска весов регуляризации) примет значение по умолчанию 20. Заметим, что такое изменение может привести к некоторому изменению результата. Описание множества *t* происходит автоматически. Оператор *Draw* отображает все функции задачи. Наконец, *EOF* необязателен в конце файла.

1-differential_equation_1 – дифференциальное уравнение 1-ого порядка: правая часть полином.

Файл-задание (*MNG-dif-1.mng*) на поиск функций $x(t)$ и $f(t)$:

```
CVNumOfiter = 21      # максимальное кол-во итераций поиска весов регуляризации
CVstep      = 21      # кол-во подмножеств для процедуры кросс-валидации
Select  time AS t, position AS x from ../Spring5.xlsx # считывание данных из таблицы ../Spring5.xlsx
                                                # столбец time переименовывается в t, position - в x

SET:    t = [ , , 0.025]      # область определения функции x(t)
        X = [ 0, 2.2, 0.1 ]   # область значений (с запасом) функции x(t)
VAR:    x ( t )               # искомая функция
        f ( X ); PolyPow = 5  # искомая правая часть - полином 5-ой степени
EQ:     d/dt(x) = f(x)        # дифференциальное ур-ие
OBJ:    x.MSD() + f.Complexity ( Penal[0] ) # целевая функция – смешанный критерий выбора x(t)
Draw                                     # отображение всех функции: x(t) и f(x)
```

Две новые конструкции: неизвестная функция $f(x)$, представляемая полиномом 5-ой степени (*PolyPow* = 5) и конструкция *EQ*:, определяющая дифференциальное уравнение. Выражение d/d означает оператор дифференцирования, в данном случае дифференцирование функции x по t .

1G-differential_equation_1 – дифференциальное уравнение 1-ого порядка: правая часть сглаженная ломаная.

Файл-задание (*MNG-dif-1G.mng*) на поиск функций $x(t)$ и $f(t)$ отличается от рассмотренного выше одной строкой:

```
вместо записи  f ( X ); PolyPow = 5  # искомая правая часть - полином 5-ой степени
стоит запись   f ( X ); VarType = G  # искомая правая часть - сглаженная ломаная.
```

Сглаженная ломаная – специальная непрерывно-дифференцируемая конструкция, аппроксимирующая ломаную.

2-differential_equation_2 – дифференциальное уравнение 2-ого порядка.

подробным описание модели можно ознакомиться здесь:

<http://injoit.org/index.php/j1/article/view/612>

$$\frac{d^2x}{dt^2} = f\left(x, \frac{dx}{dt}\right), x \in C^2(t_{\min}, t_{\max}), f \in C^2(G), G \subset R^2$$

Файл-задание (MNG-dif-2.mng) на поиск функций $x(t)$ и $f(t)$:

```
CVNumOfIter = 21          # максимальное кол-во итераций поиска весов регуляризации
CVstep       = 21          # кол-во подмножеств для процедуры кросс-валидации
Select t, x from ../Spring5.dat # считывание данных
SET:  t = [ , , 0.025]      # область определения функции x(t)
      X = [ -0.1, 2.2, 0.1 ] # область значений (с запасом) функции x(t)
      V = [ -1, 1.5, 0.1 ]   # область значений (с запасом) функции v(t)
VAR:  x ( t )                # искомая функция
      v ( t )                # искомая функция
      f ( X, V ); PolyPow = 6 # искомая правая часть - полином 5-ой степени от x и t
EQ:   d2/dt2(x) == f(x,v)    # дифференциальное ур-ие 2-ого порядка
      v == d/dt(x)           # дифференциальное ур-ие 1-ого порядка
OBJ:   f.Complexity(Penal[0], Penal[1])/x.V.sigma2 + x.MSD() # критерий выбора x(t),v(t) и f(x,t)
Draw x                                     # отображение функции x(t)
Pl = Polyline (x, v, None, 'Trajectory') # ломаная, состоящая из точек {xi,vi}
Pl.Y[0] = Pl.Y[1]                        # прячем 0-ую точку
Pl.Y[-1] = Pl.Y[-2]                      # прячем последнюю точку
Draw f Trajectory;LC:red                 # отображение функции f(x,v) и траектории решения
```

Конструкция VAR: $f(X, V)$; PolyPow = 6 описывает неизвестную от двух переменных, заданную на множестве $X \times V$ полиномом 6-ой степени. Polyline (полилиния) – ломаная линия, задаваемая набором пар точек (x, v) . Последний оператор – отображение функции двух переменных (линиями уровня) и траектории системы в координатах x и v (линия красного цвета).

3-Oscillator_K_Mu_xr – Осциллятор с вязким трением (дифференциальное уравнение 2-ого порядка с неизвестными коэффициентами).

Использование формата .odt (и соответствующего редактора LibreOffice Writer) позволяет существенно улучшить наглядность представления задания:

- различные фонты, цвета, типы и жирности текста,
 - возможность использования математических символов, греческого алфавита и т.д.,
 - возможность набирать формулы в редакторе форм,
 - возможность добавлять в файл дополнительные (не влияющие на расчеты) элементы.
- Например, словесная или схематическая постановка задачи, результаты в виде графиков, найденные погрешности и т.д.

Ниже приведен фрагмент файла-задания (Oscillator_K_Mu_x.odt или Oscillator_K_Mu_xr_ChDir.odt) на поиск функции $x(t)$ и коэффициентов K , μ , x_r (уже описанные конструкции отмечены серыми комментариями):

```

*****
BoF-SvF                                     # Файл-задание обрабатывается с этой команды
# ChDir = ./3-Oscillator_K_Mu_xr           # Рабочий каталог перемещается в ./3-Oscillator_K_Mu_xr
CVNumOfIter = 50                           # максимальное кол-во итераций поиска весов регуляризации
CVstep      = 21                           # кол-во подмножеств для процедуры кросс-валидации
DIF1        = Central                       # использование центральной схемы аппроксимации производной
Select x,t from ../Spring5.dat             # считывание данных
GRID:  t ∈ [ -1., 2.5, 0.025 ]             # область определения функции x(t) и v(t)

Var:    x ( t )                             # искомая функция
        v ( t )                             # искомая функция
        K                                     # неизвестный параметр – жесткость пружины
        Δx                                  # неизвестный параметр - смещение точки крепления (Deltax)
        μ                                   # неизвестный параметр - вязкость среды (в формуле Стокса),
                                           # (в программе заменяется на mu)
# EQ: d2/dt2(x) == - K * (x - Δx) - μ*v     # обычная запись ур-ия – без редактора формул

EQ:    
$$\frac{d^2}{dt^2}(x) == -K * (x - \Delta x) - \mu * v$$
 # формулы набираются в редакторе формул

        
$$v == \frac{d}{dt}(x)$$
 # дифференциальное ур-ие 1-ого порядка
OBJ: x.Complexity ( Penal[0])/x.V.sigma2 + x.MSD() # критерий выбора
Draw                                       # отображение функций
EOF                                       # конец обработки задания
*****

```

Все, что находится до **BoF-SvF** и после **EOF**, опускается.

Иногда удобно хранить файл задание в каталоге отличном от каталога со всеми файлами задачи. Что бы изменить рабочий каталог используется команда ChDir.

Первую производную можно аппроксимировать по-разному, в том числи **DIF1 = Central**.

В тексте используются символы ϵ , Δ и μ . Естественно, в тексте программы они заменяются на буквенные выражения.

Формулы набираются в редакторе формул (хранение в формате LaTeX).

5.2. Мультипликативное представление функций.

Пример 5.2а. Модель транспирации $E=Ust(Q,VPD) \cdot VPD$

С подробным описанием можно ознакомиться в статье
https://www.matbio.org/2019/Sokolov_14_665.pdf

```

RunSolver L&S # Параллельные расчеты ведутся на удаленном сервере
py_max_iter = 15000 # Опция максимальное кол-во итераций для solver

```

```

CVNumOfIter = 21 # Максимальное кол-во итераций CV

```

ExitStep = 1e-6 # Контроль сходимости. Завершение по малости смещения
useNaN # useNaN – строки с отсутствующими данными считываются

Select ROWNUM AS t, I As Q, Ta As T, WPD As VPD, Enan As E, Pnan As P, Dat, NN, PRel, ERel \\
from ../DATA/Phot-5short.xlsx
MakeSets_byParam Dat 11 # CV procedure parameters – 11 subsets by Dat

Set: t = [0, , 1] # Time - number of point (first=0)

Param: VPD(t)

Q (t)

T (t)

Var: E (t)

EoD # Закрывает открытую таблицу. Здесь можно не закрывать.

Var: Cqv (Q , VPD); Q ∈ [, ,-50]; VPD ∈ [, ,-50]

EQ: E = Cqv (Q , VPD) · VPD · 0.0101495726495726

E(t) = Cqv (Q (t), VPD (t)) · VPD(t) · 0.0101495726495726

E(t) = Cqv (Q (t), VPD (t)) · VPD(t) · 0.0101495726495726; t ∈ t

Obj: Cqv.Complexity (Penal[0],Penal[1]) + E.MSDnan()

DrawTransp # транспонирует двумерные рисунки.

DrawVar # готовит графики только для функций-переменных

EOF

Здесь появляется новое ключевое слово *Param*:, которое открывает блок описания функций-параметров. Они участвуют в расчетах как заранее заданные функции и поэтому должны быть определены до начала расчетов. В данном примере VPD(t), Q(t) и T(t) считываются (на основе совпадения имен переменных и аргументов с именами столбцов) из таблицы *Phot-5short.xlsx*.

Ключевое слово EQ: (equation) задает связь $E = Cqv (Q , VPD) \cdot VPD \cdot 0.0101495726495726$ между функциями E, Cqv, Q и VPD. Это сокращенная запись – более подробная нотация находится в закомментированных строках ниже.

DrawVar – готовит графики только для функций-переменных.

5.4. Интегральные уравнения

Пример 5.4а. Математическая модель: fast nonlocal heat transport in magnetic fusion plasmas.

С подробным описание модели можно ознакомиться здесь: <http://arxiv.org/abs/1901.03789>

Здесь приводится формальная математическая постановка. Однако некоторые элементы описания содержат дополнительную информацию, необходимые для дальнейших преобразований, например, при описании множеств, шаг дискретизации.

Константы:

```

Fracmin = 0.5Fracmax = 10
r_step = 0.01
t_step = 0.005
t_inj = 6.994
tmin = tinj
tmax = 7.1
rmax = 0.44
rpp = 0.9
qmax = 3 # Только для 7 разряда

```

Множества:

```

Set:  t = [tmin, tmax, t_step]
      rm = [0, rmax, r_step]
      rp = [0, rpp, r_step]

```

неизвестные

```

Var:  P(rm, t);
#     Abs(rp);      >= 0
      Abs(rm);      >= 0
      Amebs;        >= 0
      qme;            >= 0                                #(9)
#     I(t);           >= 0
      Isr(t);         >= 0
      Fout(t);        >= 0.01; <= 0.2 * pi                #(7)
#     I0;             >= 0
      Isr0;           >= 0
      Fout0;          >= 0.01; <= 0.2 * pi                #(7o)
      AbsGTrm;      >= 0

```

известные

Парам: P(r_m, t)

уравнения

```

EQ:  Abs(rm) * (Isr(t) - Isr0) = P(rm, t)                #(1)
      Isr(t) = 2 * qme / (sqrt(4 * qme * Fout(t) + Amebs * 2 + Amebs))  # (4)
      Isr0 = 2 * qme / (sqrt(4 * qme * Fout0 + Amebs * 2 + Amebs))  # (4o)
      ∫rpp*22 * ∫0rmax Abs(r) * r * dr                # (5)
      Amebs ≥ ∫ * (Fracmin + 1                             # (5a)
      Amebs ≤ ∫ * (Fracmax + 1                             # (5b)
      Amebs**2 * (1+Fout(t)) <= qme                        # (6)
      Amebs**2 * (1+Fout0) <= qme                          # (6o)
      AbsGTrm = # (11)

```

```

#       $I_{sr}(t) = \sqrt{I(t)}$                                 #(S1)
#       $I_{sr0} = \sqrt{I_0}$                                 #(S1o)

 $1/W_F < F_{out}(t)/F_{out} 0 < W_F,$                                 #(12)
#Пробные значения
 $W_F = 10, 2$ 

```

Файл-задание

```

# ПАРАМЕТРЫ РАСЧЕТОВ:

CVstep      = 13
CVNumOfIter = 0
ExitStep     = 1e-7

# КОНСТАПТЫ:

##### 7
t_inj7      = 6.994
qmax7       = 5 #3
##### 8
#t_inj      = 7.497
#qmax       = 6
#####
Frac_min    = 0.5
Frac_max    = 10
W_F         = 10.2
r_step      = 0.02
t_step      = 0.005
t_min7      = t_inj7
t_int       = 0.05
r_max       = 0.44
r_pp        = 0.9

# МНОЖЕСТВА (СЕТКИ):

GRID: t = [0, t_int, t_step]
      r_m = [0, r_max, r_step]
      r_p = [0, r_pp, r_step]
Tbl = Select r As r_m, tm As t, Pi As P7, Pi As pP7 from ../27-002Pi_T(r,t)n(r)-21/Pi(r,tm).sol
Tbl.dat('t')[:] = t_min7

# НЕИЗВЕСТНЫЕ И ПАРАМЕТРЫ

Var: P7(r_m,t)
Param: pP7(r_m,t)
EOD

Var: A_bs(r_m);    >=0
      Ame_bs;      >=0
      qme7;         >=0; <= qmax7 # bounds = [0, qmax7]          #(9)
      Isr(t);       >=0
      F_out(t);     >=0.01; <= 0.2*pi # bounds = [0.01, .2*pi]   #(7)
      Isr0;         >=0
      F_out0;       >=0.01; <= 0.2*pi # bounds = [0.01, .2*pi]   #(7o)

```

```

Int_r_m;      >=0
AbsGTr_m;     >=0

# УРАВНЕНИЯ
EQ:  A_bs(r_m) · (Isr(t)- Isr0) = P7(r_m,t)           # (1)
     Isr(t) = 2·qme7/(√(4·qme7·F_out(t)+Ame_bs**2)+Ame_bs) # (4)
     Isr0 = 2·qme7/(√(4·qme7·F_out0 +Ame_bs**2)+Ame_bs) # (4o)
     Int_r_m = 2/(r_pp**2)·∫( 0, r_max, dr_m*A_bs(r_m)·r_m ) # (5)
     Ame_bs <= (1+Frac_max) · Int_r_m                 # (5a)
     Ame_bs >= (1+Frac_min) · Int_r_m                 # (5b)
     AbsGTr_m=(Ame_bs- Int_r_m)·r_pp**2/(r_pp**2-r_max**2) # 11
     Ame_bs**2 · (1+F_out(t)) <= qme7                # (6)
     Ame_bs**2 · (1+F_out0 ) <= qme7                 # (6o)
     F_out0 <= F_out(t) * W_F                        # 12
     F_out(t) <= F_out0 * W_F                       # 12

```

```

Obj:  A_bs.Complexity (Penal[0]) + Isr.Complexity (Penal[1]) \
      + F_out.Complexity (Penal[2]) + P7.MSD()

```

DrawTransp

Draw

EOF

Новые конструкции:

- 1) Tbl.dat('t')[:] -= t_min7 – оператор (Python): для столбца с именем 't' таблицы Tbl уменьшить все значения на t_min7
- 2) $\sqrt{4 \cdot qme7 \cdot F_out(t) + Ame_bs^{**2}}$ -> sqrt (4·qme7·F_out(t)+Ame_bs**2);
- 3) $\int(0, r_max, d(r_m) \cdot A_bs(r_m) \cdot r_m)$ – интеграл от 0 до r_max для выражения d(r_m)*A_bs(r_m)·r_m. Дифференциал d(r_m) – стоит на первом месте!

Целевая функция (Obj:) состоит из штрафов за кривизны трех функций (A_bs, Isr, F_out) и среднеквадратичного отклонения P7 от данных.

№№№№№№№№№№№№№№№№№№№№№№№№ - дальше не читать!

6. Параметры и операторы файла вычислительного задания (.mng)

Переменных с именами d и d2 лучше избегать – они зарезервированы как символы-дифференциалов.

6.1. Управляющие параметры

```
DIF1 = Central
RunSolver  Server # ServerParallel # LocalParallel #Local #
EOF
printL = 1
```

6.2. Опции сервера

6.3. SET: or GRID:

```
GRID:      t = [ -1.,  2.5, 0.025 ]
           gx = [ -0.1, 2.2, 0.1  ]
           gv = [ -1,   1.5, 0.1  ]
```

6.4. Select - некоторое небольшое подмножество языка SQL

```
Select r As r_m,tm As t, Pi As P_RTP, Pi As pP_RTP from ../27-Pe_Pi/Pi(r,tm).sol
Select r As r_m,tm As t, Pi As P_RTP, Pi As pP_RTP from ../27-Pe_Pi/Pi(r,tm).sol As Tbl
Tbl = Select r As r_m,tm As t, Pi As P_RTP, Pi As pP_RTP from ../27-Pe_Pi/Pi(r,tm).sol
      Если требуется работа с записями таблицы (Tbl), лучше использовать вариант (Tbl=)
Select x,t from Spring5.dat where x>=1 and t==100
Select * from Spring5.dat
ROWNUM - Поле номера записи по порядку
EoD - Закрывает открытую таблицу. Во избежание путаницы (имен столбцов с другими именами), лучше закрывать.
Select X,Y, num as Z from Les.xyz where Z==2
```

6.5. WriteSvFtbl

WriteSvFtbl "tmp.txt" – Сохранение таблицы в текстовый файл

6.6. CV

CVstep = 7 – разбивает множество данных на 7 подмножеств :

```
[ [0,7,14...], [1,8,15...] ... [6,13...] ]
```

MakeSets_byParam ParamName =Data Every =3 Margin =0

```
MakeSets_by Param Data 13 4
```

разбивает множество данных на 13 подмножеств на основе столбца Data

В обучающей последовательности убирает поля (по 4 до и после)

Важно при работе с временными рядами.

MakeSets_byParts Every =3 PartSize =96 Margin =4

```
MakeSets_byParts 3 96 4
```

По умолчанию Every=7, PartSize = 1; Margin=0

CVstep = 7 – разбивает множество данных на 7 подмножеств :

```
[ [0,7,14...], [1,8,15...] ... [6,13...] ]
```

6.7. VAR:

PolyPow - степень полинома

Var: x (t); t ∈ [, 2.5,0.025]; x ∈ [-6.,7]; # <=7; >= -6

v (t)

```

f ( gx, gv ); PolyPow = 6
var:  x ( t ); t ∈ t64; x ∈ [ -10.,10 ];
      x ( t ); DataFile = ../DATA/Spring5.dat
      x ( t ); Select x,t from ../DATA/Spring5.dat
d/d(t) - полный интервал автоматически уменьшается на шаг
VAR:  T(ρ,t); t ∈ Time; ρ ∈ Rp; T ∈ [0,delH2O]; PolyPow =5;
      E ( t ); t ∈ [0, t_max, 1]; >=0
DataFile = T.txt;
initialize = 1
Reg ( gWD ); PolyPow = 7; <= 1 - как EQ:

```

6.8. PARAM:

Param Ust (Q , T)
 # Если InFile не задан, считывается из файла по умолчанию.

6.9. EQ:

```

EQ:
d2/dt2(x(t)) == f(x(t),v(t)); #t <> 1
v(t) == d/dt(x(t))
v(t17) == d/dt17(x(t17)); t17 ∈ t; ; t17 <> 1
v(t17) == d/dt17(x(t17)); t17 ∈ [ -1., 2.5, 0.025 ]

if ..cond.. : d(PHqt(gQ,gT))/d(gQ) >= 0    => if not ( ..cond..) : return Constraint.Skip
fr (0, 2, f) - rectangle = f(0) + f(1) для шаг = 1

```

6.10. OBJ:

```

OBJ: f.Complexity ( [Penal[0], Penal[1]] )/x.V.sigma**2 + x.MSD()
ObjL: x.Complexity ( Penal[0] ) + x.MSD()
ObjU: x.MSDverif ( formula )
Obj: ∫ ( dt*x(t)**2 )
Obj: ∫ ( 0, 1, dt*x(t)**2 )      # dt on the first place !

```

6.11. Draw

```

Draw
DrawTransp - транспонирование картинки.
Draw fun11 L:1,2,4,8,16,32,64 – задание значений уровней
DrawVar – только переменные
DrawErr – оставшиеся ошибки
MarkerSize = 1

```

6.12. Операторы Python, использующие разработанные классы.

Работа с таблицами

Tbl.dat('t')[:] := t_min7 – оператор (Python): для столбца с именем 't' таблицы Tbl уменьшить все значения на t_min7

Работа с компонентами GIS

```

Pol = Polyline ([1,2,3],[4,5,6])
Pol = Polyline ('X','Y','Z')
Pol.Draw ( mask, 6, 1000 ) - Draw(self, mask, pixSize=1, Val=None):

```



```

mask.FloodFillReal ([6508870, 5875460], 1000,1000)
FloodFillReal (self, xy, BordVal,FillVal)
dh = NaN
ang = 2
H.Smoothing (dh, ang, mask,1000)
H.SaveGrid ( 'smooth2grad.asc' ) - эквивалентно H.SaveGrid ( 'smooth2grad.asc', 'N' )
TA = H.TiltAngle ( ) - sqrt ( **2 + **2 ) производных — модуль градиента
TA.Draw()
TA.SaveGrid ( 'after.asc' )
Xder = H.Make_1_Deriv()
Yder = H.Make_1_Deriv('y')

```

7. Организация файлов данных

.xlsx - первая строка содержит имена столбцов

.txt - первая строка содержит имена столбцов. Разделитель пробел или табуляция.

#END# - первый столбец xlsx - конец
#END# - в строке имен xlsx - конец

Дополнительно

Transp - Транспонировать при отрисовке Draw A.sol;Transp

SavePoints — сохранять точки в *.txt. По умолчанию False

FOR: CC in [RUS GBR FRA DEU ESP ITA]: - скобки необязательны, «:» - можно опустить
 EOFOR

Если имя res-файла не задано явно, оно формируется из png-файла, заменой расширения имени на res.

Файлы .sol содержат решения задачи (они же и начальные приближения) как сеточные функции одной или двух переменных.

Первая строка содержит служебную информацию :

имена переменных, имя функции, служебная информация (после знака `#`)

1D — содержит таблицу со столбцами: аргумент и функция

2D — содержит матрицу решения.

В первой строке сначала имя, соответствующее столбцам, затем — строкам.

Далее строка чисел, определяющая значения столбцов (первого аргумента)

Далее набор строк. В каждой строке: первое число — значение второго аргумента, далее значения функции.

Пример :

r	t	Pdf	#SvFver_62_mtr2
	0.0	5.0	10.0
100	50.3	99.5	140.5
102	49.5	96.5	135.6

определяет сеточную функцию двух переменных, например, Pdf(5.0, 102) = 96.5

Литература

Sokolov A. V., Voloshinov V. V. Model Selection by Balanced Identification: the Interplay of Optimization and Distributed Computing // Open Computer Science, 2020, 10 – p. 283–295.

<https://doi.org/10.1515/comp-2020-0116>

Соколов, А.В.; Волошинов, В.В. Выбор математической модели: баланс между сложностью и близостью к измерениям. International Journal of Open Information Technologies, 2018, 6(9) С. 33-41

<http://injoit.org/index.php/j1/article/view/612>

```
Param:    Position(x,y) = -1.2; #initialize = -1.2
Param:    S ( t ) = SS(t).sol #из файла считываем все (И Set)
           но сохраняем имена S и t
Param:    S ( t ); DataFrom = S(t).sol # берем только данные
Var:      S ( t )>0
Var:      S ( t ) \in [0,100]
ComplCycle
ComplCyc0E(self,bets) первая точка должна быть равна последней
```

```
RunMode = L&S    L+N # (NL-file local
```

```
tbl = joinTab (tblH,tbCs,tb)
```

```
tbl.dat('SPI')[:] = log (tbl.dat('SPI')[:])
```

Технология сбалансированной идентификации математических моделей (т.н. SvF-технология, от Simplicity vs Fitting) открывает перспективное направление прикладной математики, сочетающее методы структурного математического моделирования, оптимизации, адаптивной регуляризации и распределенных вычислений.

Недостающее (закомментированное) определение целевой функции (Obj: ...) будет добавлено автоматически.

```
useNaN                                     # use NaN values in experimental data
```

Оператор 'MakeSets_byParam Dat 11' задает разбиение множества данных (для перекрестной проверки) на 11 подмножеств по столбцу таблицы 'Dat'.

