УДК 519.711

В ПОИСКАХ МИНИМАЛЬНОЙ РЕАЛИЗАЦИИ НЕЛИНЕЙНОЙ ДИНАМИЧЕСКОЙ СИСТЕМЫ (ЧАСТЬ 2: АЛГОРИТМ)

О.Ю. Копысов
Декартов Научен Център
Болгария, 9002, гр. Варна, ж.к. Чайка, бл.185, ап.13
E-mail: iainstitute@inbox.ru

Ключевые слова: объект, модель, экспериментальные данные, класс моделей, ЛСМодель с невязкой, вспомогательная ЛСМодель, идентификация, общее решение задачи идентификации, сингулярное разложение, метод смешанных наименьших и тотальных наименьших квадратов.

Аннотация: В докладе предложен итерационный алгоритм получения общего решения задачи идентификации неизвестных параметров и, уже на основе полученного общего решения, вычисления минимальной реализации и состояния нелинейного динамического объекта.

1. Введение

Исходный класс ЛСМоделей очень широк, и строить общий алгоритм для всех случаев просто не рентабельно. Он будет считать значительно дольше локальных алгоритмов, хотя все локальные алгоритмы мало чем отличаются друг от друга. Ниже приводится алгоритм на языке **MATLAB** для примера из первой части доклада. Все математические обозначения в этом докладе полностью совпадают с первой частью. Аналогичные обозначения на языке **MATLAB** легко узнаваемы.

В качестве численного метода решения задач Коши взят метод, основанный на формулах Рунге—Кутта четвертого и пятого порядка точности, который предложил Фельберг, подпрограмма называется `ode45` (*строки листинга* 19, 21, 49, 51).

Для ее использования необходим отдельный файл с именем, например, VSPU2019_Fi (для вычисления матрицы $\Phi(t)$) содержащий функцию, которая тоже называется `VSPU2019_Fi`, и вычисляет производные решаемой задачи Коши. Для вычисления производных матриц $R_k(t)$ используется еще и кубическая сплайн аппроксимация (подпрограмма `spline`), что позволяет работать с заданной точностью `opt` (*строка* 6) и с неравномерными отсчетами измеряемых сигналов.

Необходимая информация передается в функции через общие переменные (`global t y w C`) (*см. полный текст алгоритма*).

Задачи Коши решаются для третьих столбцов, например, `Fi_3` матрицы $\Phi(t)$. Подпрограмма `ode45` в результате дает массив размерности `m×3`, а для дальнейших вычислений необходимо `3×m`. Поэтому после каждого применения `ode45` результат транспонируется, например, `Fi_3=Fi_3';` (*строка* 19), так дальнейшие операторы

больше похожи на математические формулы. Далее предыдущие столбцы вычисляются умножением на матрицу C (*строки* 20, 50).

Матрица C (*строка* 18) задает порядок вспомогательной модели, здесь первый столбец, как и в примере из первой части, нулевой. Алгоритм допускает любые значения первого столбца, что может определять дополнительную фильтрацию измеряемых сигналов.

2. Итерационный алгоритм идентификации параметров и состояния

ДАНО:

```
t=0 : pi/25 : 2*pi;    m=size(t,2);                          %12
w0=sin(t)+cos(3*t); y0=-cos(t);  dy0=sin(t);                 %13
                                                             %14
Iteration=0; iter=0;                                         %15
 delta_y=0.33*sin(10*pi*t);        %  delta_y=0*t;           %16
 y=y0+delta_y;     w=w0;                                     %17
 C=[ 0 1 0; 0 0 1; 0 0 0];                                   %18
```

ШАГ 0: Решить задачи Коши для матрицы $\Phi(t)$

```
[tFi,Fi_3]=ode45(@VSPU2019_Fi,t,[0;0;1],opt);  Fi_3=Fi_3';   %19
   Fi_2=C*Fi_3;  Fi_1=C*Fi_2;                                %20
```

и тех матриц $R_k(t)$, функции $f_k(t,y,w)$ которых известны точно

```
[tR3,R3_3]=ode45(@VSPU2019_R3,t,[0;0;0],opt);  R3_3=R3_3';   %21
```

Сформировать матрицу P, вычислить ее QR-разложение и матрицу `T1` обратную к T

```
P=[Fi_1(1,:)' Fi_2(1,:)' Fi_3(1,:)' R3_3(1,:)'];             %22
[Q,T]=qr(P); T1=T(1:4,1:4)^(-1);                             %23
```

ШАГ 1: Решить задачи Коши для матриц $R_k(t)$, функции $f_k(t,y,w)$ которых известны неточно

```
 [tR1,R1_3]=ode45(@VSPU2019_R1,t,[0;0;0],opt); R1_3=R1_3';   %49
    R1_2=C*R1_3; R1_1=C*R1_2;                                %50
 [tR2,R2_3]=ode45(@VSPU2019_R2,t,[0;0;0],opt); R2_3=R2_3';   %51
```

ШАГ 2: Решить задачу идентификации посредством вспомогательной ЛСМодели с невязкой

```
 R=[R1_1(1,:)' R1_2(1,:)' R1_3(1,:)' R2_3(1,:)'];            %52
                                                             %53
 R1R2=Q'*R; [U,S,V]=svd(R1R2(5:m,:));                        %54
 v=V(1,3)*V(:,4)-V(1,4)*V(:,3); alpha=v/v(2);                %55
                                                             %56
 beta=-T1*R1R2(1:4,:)*alpha;                                 %57
 z0=beta(1:3);  a=[alpha;beta(4)];                           %58
```

ШАГ 3: Вычислить $\boldsymbol{z}(t)$, $\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{z}(t)$ и после анализа, на их основе вычислить компоненты вектора состояния исходной ЛСМодели

```
z=Fi_1*z0(1)+Fi_2*z0(2)+Fi_3*z0(3)...                          %60
   +R1_1*a(1)+R1_2*a(2)+R1_3*a(3)+R2_3*a(4)+R3_3*a(5);         %61
y=-z(3,:)/a(2);                                                %63
                                                               %64
dz=C*z+a(1:3)*y+[0;0;a(4)]*y.^3+[0;0;a(5)]*w;                  %62
dy=-dz(3,:)/a(2);                                              %65
```

ПЕРЕЙТИ К ШАГУ 1 и продолжить вычисления на основе вновь вычисленных компонент вектора состояния исходной ЛСМодели.

```
it=input('Делать ли ещё итерацию? (ДА - Enter, НЕТ - 0) ');  %44
```

Если же невязка $e(t) = z_1(t)$ удовлетворяет критерию идентификации, ЗАКОНЧИТЬ ВЫЧИСЛЕНИЯ.

Комментарий к строкам 63 *и* 65.

Напомню, что вспомогательный вектор состояния $\boldsymbol{z}(t)$ имеет вид:

$$z_1(t) = e(t),$$
$$z_2(t) = z_1^{(1)}(t) - c_1 z_1(t) - a_{11} y(t),$$
$$z_3(t) = z_2^{(1)}(t) - c_2 z_1(t) - a_{12} y(t).$$

Невязка $e(t)$ – это порождение неточных измерений, то есть приблизительно $0(t)$. $z_1^{(1)}(t) \approx 0(t)$ тоже, ведь брать в расчет производную от ошибки, большая ошибка, лучше положить ее равной нулю. a_{11}=0, следовательно, $z_2(t) \approx 0(t)$ и $z_2^{(1)}(t) \approx 0(t)$ тоже. Отсюда из третьей строки $y(t) \approx -z_3(t)/a_{12}$ и, следовательно, $dy/dt \approx -\frac{d}{dt} z_3(t)/a_{12}$.

Комментарий к окончанию вычислений.

Если алгоритм сходится, то норма невязки $e(t)$=$z_1(t)$ должна уменьшаться. Для анализа на экран выводится Чебышева норма $\max_t |z_s(t)|$ каждой компоненты вспомогательного вектора $\boldsymbol{z}(t)$:

```
z_max=[max(abs(z(1,:))) max(abs(z(2,:))) max(abs(z(3,:)))]   %69
```

Ниже приведена распечатка параметров, начальных условий и норм каждой компоненты вспомогательного вектора, а также сингулярных чисел для трех итераций:

```
parameters =
           0  1.0000e+000  3.5177e+000 -4.0000e+000 -1.0002e+000
           0  1.0000e+000  2.9729e+000 -3.9064e+000 -9.9987e-001
           0  1.0000e+000  2.9826e+000 -3.9649e+000 -1.0001e+000

z_0 =
-5.2643e-007  3.6604e-002  6.6347e-001
-6.9542e-004  6.2432e-003  9.7826e-001
-3.6197e-004 -4.5834e-003  1.0460e+000

z_max =
```

```
 1.1999e-003  3.6604e-002  1.3391e+000
 6.9542e-004  1.0007e-002  1.0543e+000
 7.2996e-004  5.0346e-003  1.0460e+000

singular_value =
 5.4468e+000  1.7376e-001  6.3547e-002  8.6610e-004
 5.2467e+000  1.7476e-001  1.4809e-002  3.3866e-004
 5.2316e+000  1.7469e-001  7.8653e-003  2.9792e-004
```

Норма невязки (*первый столбец* `z_max`) увеличилась в третьей итерации, значит, вычисления нужно закончить. Однако во втором столбце, норма $\max_t |z_2(t)|$ еще уменьшается, а $z_2(t)$, как и $z_1(t)$, должно идти к $0(t)$ и можно сделать еще пару итераций. Оба последних сингулярных числа (*третий и четвертый столбцы* `singular_value`) пока убывают, что подтверждает наши предположения о размерности общего решения.

Сходимость функций $y(t)$, $y^3(t)$, $\frac{d}{dt}y(t)$ (цветные толстые линии) к их теоретическим значениям $-cos(t)$, $-cos^3(t)$, $sin(t)$ (черные тонкие линии) можно увидеть на графиках (см. рис.1).

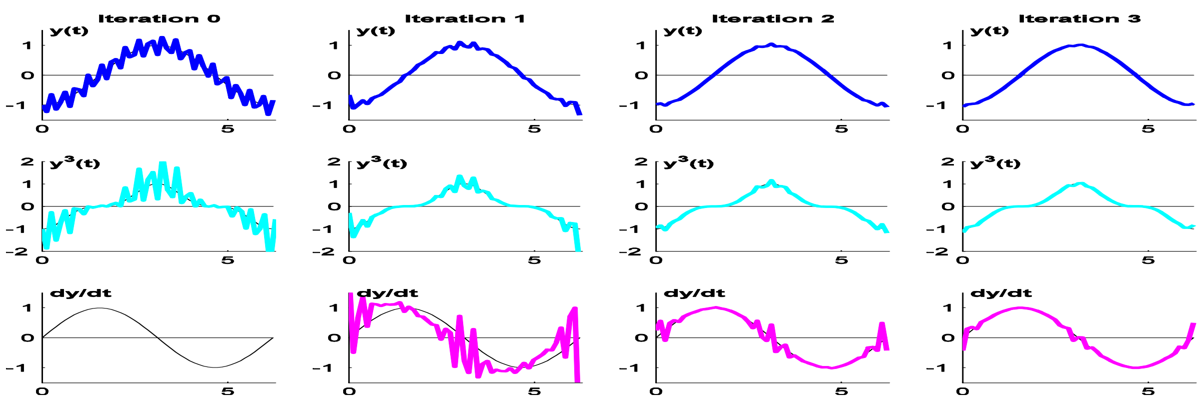

Рис. 1. Графики функций $y(t)$, $y^3(t)$, $\frac{d}{dt}y(t)$ для трех итераций.

3. Полный текст алгоритма

```
%%% S=3, K=3 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%        a(1)*ddy+a(2)*dy+a(3)*y+a(4)*y^3+a(5)*w=0,      %%%
%%%           w=sin(t)+cos(3t), y=-cos(t)+delta_y          %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear; clf; format short e;
opt=odeset('RelTol', 2^(-21)); %Точность решателя диф.ур.
LW=2;                          %Ширина линий на графиках
SPcol=4;            %Максимальное число столбцов графиков

global  t y w C

t=0 : pi/25 : 2*pi;    m=size(t,2);
w0=sin(t)+cos(3*t); y0=-cos(t);  dy0=sin(t);

Iteration=0; iter=0;
 delta_y=0.33*sin(10*pi*t);         %  delta_y=0*t;
 y=y0+delta_y;     w=w0;
```

```
 C=[ 0 1 0; 0 0 1; 0 0 0];                                              %18
[tFi,Fi_3]=ode45(@VSPU2019_Fi,t,[0;0;1],opt);  Fi_3=Fi_3';              %19
   Fi_2=C*Fi_3;  Fi_1=C*Fi_2;                                           %20
[tR3,R3_3]=ode45(@VSPU2019_R3,t,[0;0;0],opt);  R3_3=R3_3';              %21
P=[Fi_1(1,:)' Fi_2(1,:)' Fi_3(1,:)' R3_3(1,:)'];                        %22
[Q,T]=qr(P); T1=T(1:4,1:4)^(-1);                                        %23
while 1,                                                                %24
%%%%%%%%%%%%%%%%%%%%   ВЫВОД  ГРАФИКОВ   %%%%%%%%%%%%%%%%%%%%% 25
 iter=iter+1; if iter>SPcol, iter=1; end;                               %26
 subplot(3,SPcol,iter); cla; hold on;                                   %27
 title(['\bfIteration ',int2str(Iteration)]);                           %28
plot(t,0*t,'k', t,y0,'k'); plot(t,y,'b','LineWidth',LW);                %29
 text(.2,1.5,'\bfy(t)');xlim([t(1) t(m)]);ylim([-1.5 1.5]);             %30
 hold off;                                                              %31
 subplot(3,SPcol,SPcol+iter); cla; hold on;                             %32
plot(t,0*t,'k',t,y0.^3,'k');plot(t,y.^3,'c','LineWidth',LW);            %33
 text(.2,2,'\bfy^3(t)');xlim([t(1) t(m)]);ylim([-2 2]);                 %34
 hold off;                                                              %35
 subplot(3,SPcol,2*SPcol+iter); cla; hold on;                           %36
plot(t,0*t,'k', t,dy0,'k');                                             %37
if Iteration~=0,  plot(t,dy,'m','LineWidth',LW); end                    %38
 text(.2,1.5,'\bfdy/dt');xlim([t(1) t(m)]);ylim([-1.5 1.5]);            %39
 hold off;                                                              %40
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% 41
                                                                        %42
if Iteration>0,                                                         %43
it=input('Делать ли ещё итерацию? (ДА - Enter, НЕТ - 0) ');             %44
if it==0, break; end;                                                   %45
end                                                                     %46
Iteration=Iteration+1                                                   %47
                                                                        %48
 [tR1,R1_3]=ode45(@VSPU2019_R1,t,[0;0;0],opt); R1_3=R1_3';              %49
    R1_2=C*R1_3; R1_1=C*R1_2;                                           %50
 [tR2,R2_3]=ode45(@VSPU2019_R2,t,[0;0;0],opt); R2_3=R2_3';              %51
 R=[R1_1(1,:)' R1_2(1,:)' R1_3(1,:)' R2_3(1,:)'];                       %52
                                                                        %53
 R1R2=Q'*R; [U,S,V]=svd(R1R2(5:m,:));                                   %54
 v=V(1,3)*V(:,4)-V(1,4)*V(:,3); alpha=v/v(2);                           %55
                                                                        %56
 beta=-T1*R1R2(1:4,:)*alpha;                                            %57
 z0=beta(1:3);  a=[alpha;beta(4)];                                      %58
                                                                        %59
 z=Fi_1*z0(1)+Fi_2*z0(2)+Fi_3*z0(3)...                                  %60
    +R1_1*a(1)+R1_2*a(2)+R1_3*a(3)+R2_3*a(4)+R3_3*a(5);                 %61
 y=-z(3,:)/a(2);                                                        %63
                                                                        %64
 dz=C*z+a(1:3)*y+[0;0;a(4)]*y.^3+[0;0;a(5)]*w;                          %62
 dy=-dz(3,:)/a(2);                                                      %65
 %%%%%%%%%%%%%%%%%%%%%% ВЫВОД  НА  ЭКРАН  %%%%%%%%%%%%%%%%%%%%% 66
 parameters=a'                                                          %67
 z_0=z0'                                                                %68
 z_max=[max(abs(z(1,:))) max(abs(z(2,:))) max(abs(z(3,:)))]             %69
 singular_value=diag(S)'                                                %70
end;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% 71
```

Файл с именем VSPU2019_Fi для вычисления $\Phi(t)$

```
function[dFi]=VSPU2019_Fi(t_fi,Fi)
global  t y w C
dFi = C*Fi;
```

Файл с именем VSPU2019_R1 для вычисления третьего столбца $\boldsymbol{r}_{1,3}(t)$ матрицы $R_1(t)$

```
function[dR]=VSPU2019_R1(t_r1,R)
global  t y w C;
dR = C*R + [0; 0; spline(t,y,t_r1)];
```

Файл с именем VSPU2019_R2 для вычисления третьего столбца $\boldsymbol{r}_{2,3}(t)$ матрицы $R_2(t)$

```
function[dR]=VSPU2019_R2(t_r2,R)
global  t y w C;
dR = C*R + [0; 0; spline(t,y,t_r2)^3];
```

Файл с именем VSPU2019_R3 для вычисления третьего столбца $\boldsymbol{r}_{3,3}(t)$ матрицы $R_3(t)$

```
function[dR]=VSPU2019_R3(t_r3,R)
global  t y w C
dR = C*R + [0; 0; spline(t,w,t_r3)];
```