Р10-2004-202

Ч. Торок*, Н. Д. Дикусар

MS .NET-КОМПОНЕНТЫ ДЛЯ КУСОЧНО-КУБИЧЕСКОЙ АППРОКСИМАЦИИ

*Технический университет г. Кошице, Словакия

Торок Ч., Дикусар Н. Д. Р10-2004-202
MS .NET-компоненты для кусочно-кубической аппроксимации

Представлены компоненты MS Visual C# для алгоритма кусочно-кубической аппроксимации в режиме автоматического слежения (APCA — autotracking piecewise cubic approximation): библиотека классов и Windows-приложения. Алгоритмы объектно-ориентированных классов основаны на недавно предложенном новом численном подходе. Для выполнения алгоритма APCA и конструирования аппроксимант на основе данных создано Windows-приложение, основанное на компонентах классов. Приводится структура классов и описываются правила использования приложения и обработки данных и аппроксимант в нем. Эффективность нового подхода демонстрируется как для расчетных данных, так и для реальных измерений.

Работа выполнена в Лаборатории информационных технологий ОИЯИ.

Сообщение Объединенного института ядерных исследований. Дубна, 2004

Török Cs., Dikoussar N. D. P10-2004-202
MS .NET Components for Piecewise Cubic Approximation

The paper presents MS Visual C# components for autotracking piecewise cubic approximation (APCA): a class library and a Windows application. The algorithms of the object-oriented classes are based on a recently proposed new numerical approach. To perform APCA analysis and construct data approximants, a Windows application was built based on the class components. The paper shows the structure of the classes and how to use the application and handle data and their approximants. The efficiency of the new approach is demonstrated on simulated data and real measurements.

The investigation has been performed at the Laboratory of Information Technologies, JINR.

Communication of the Joint Institute for Nuclear Research. Dubna, 2004

ВВЕДЕНИЕ

Анализ зависимости между переменными является одной из главных задач технических и научных исследований. Характер зависимости и точность данных часто определяют инструменты и методы, доступные исследователям.

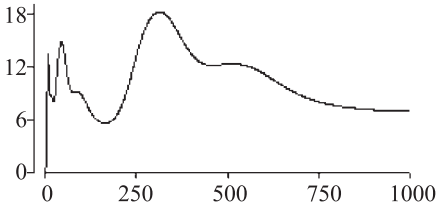

Рис. 1. Результаты расчета теплоемкости электронов для молекулы D-ацетона

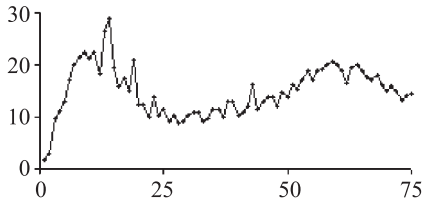

Рис. 2. Сечения для соударения π-мезонов

На рис. 1, 2 показаны две различные ситуации. На рис. 1 изображена сложная зависимость точек почти без ошибок, полученная в результате расчетов [1]. На рис. 2 представлена зависимость, на которой без труда можно увидеть два пика, однако данные сильно искажены шумом [2]. В первом случае нет необходимости использовать регрессионный анализ, тогда как во втором едва можно подобрать явную функцию, которая могла бы выразить зависимость на всем интервале. В этом случае можно использовать сплайны, однако число узлов в таком сплайне будет слишком велико и это число желательно уменьшить тем или иным способом.

В работе [3] мы предложили новый подход (алгоритм LOCUSD) к анализу сложных зависимостей, искаженных относительно небольшим шумом. Предложенный там алгоритм реализован в виде Widows-приложения *кусочно-кубической аппроксимации в режиме слежения* — APCA (**a**utotracking **p**iecewise **c**ubic **a**pproximation), позволяющего разбивать интервал/кривую на подынтервалы/сегменты различной длины, находить на каждом участке локальные кубические оценки (этап I) и получать глобальную кусочно-кубическую аппроксиманту (этап II). *Обнаружение точек стыковки* сегментов (*узлов*) в режиме автоматического слежения и *рекуррентное* вычисление параметров

составляют два важных свойства предложенного метода, использующего специальную кубическую модель аппроксиманты (TPS-модель) [3, 4].

В разд. 1 дано краткое описание компонентов APCA и формул, заложенных в алгоритме, который они реализуют. В следующем разделе описана объектно-ориентированная библиотека компонент на языке MS Visual C#. Третий раздел содержит описание Windows-приложения для выполнения APCA, анализа и формирования данных аппроксимант, построенных с помощью системы компонентов. В разд. 4 приводятся примеры результатов расчетов, выполненных с помощью компонентов APCA и Windows-приложения. В конце приводится итоговое описание пространства имен и сигнатур конструкторов и методов для классов SegmentsAll и Segment.

1. APCA

APCA основана на модели аппроксиманты со свободным параметром θ

$$f \approx \Pi + \theta Q$$

для кубического приближения функции, заданной формулой или таблицей. В APCA эта модель реализуется в два этапа. На первом этапе находятся подынтервалы и «кубические сегменты», а на втором — кусочно-кубическая аппроксиманта в целом. Прежде чем привести основные формулы каждого из этапов, дадим краткое описание кубической модели.

Модель. Для кусочно-кубического приближения кривой, заданной упорядоченным массивом данных

$$M : \{[x_n, f_n]\}_{n=\overline{1,N}},\ f_n = f(x_n),\ x_n < x_{n+1},\ 4 \ll N, \tag{1}$$

мы используем кубическую модель, полученную в рамках прямых и обратных четырехточечных или дискретных проективных преобразований (DPT) [4]. Набор из четырех точек

$$q : \{f(\alpha), f(\beta), f(0), f(\tau)\},\ q \in M,\ \alpha = x_\alpha - x_0,\ \beta = x_\beta - x_0,\ \tau = x - x_0$$

представляет основной объект в DPT-методике, где x_α, x_β, x_0 и x — любые значения из $\{x_n\}$, $x_\alpha \neq x_\beta \neq x_0 \neq x$, $n = 1, 2, \ldots, N$. Из трех точек в наборе q формируется 3D-вектор $\mathbf{r}_0 = [f(\alpha),\ f(\beta),\ f(0)]^T \in M$, который мы будем называть репером.

Для данного репера $\mathbf{r}_0$ введем две операции (прямую (DPT) и обратную (IDPT)).

DPT:

$$D\left(f(\tau);\ \mathbf{r}_0\right) \equiv Df(x) = p_1 f(\alpha) + p_2 f(\beta) + p_3 f(\tau),$$

где

$$p_1 = \frac{\beta\tau}{(\alpha-\tau)(\alpha-\beta)},\ p_2 = \frac{\alpha\tau}{(\beta-\tau)(\beta-\alpha)},$$
$$p_3 = \frac{\alpha\beta}{(\tau-\alpha)(\tau-\beta)},\quad p_1+p_2+p_3 = 1.$$

IDPT:

$$D^{-1}Df(\tau) = d_1 f(\alpha) + d_2 f(\beta) + d_3 Df(\tau), \tag{2}$$
$$d_1 = \frac{\tau(\tau-\beta)}{\alpha(\beta-\alpha)},\ d_2 = \frac{\tau(\tau-\alpha)}{\beta(\alpha-\beta)},\ d_3 = \frac{(\tau-\alpha)(\tau-\beta)}{\alpha\beta},$$
$$d_1+d_2+d_3 = 1.$$

В частности,

$$D(c_0 + c_1\tau + c_2\tau^2 + \theta\tau^3; \mathbf{r}_0) = c_0 + \theta\alpha\beta\tau. \tag{3}$$

На основе (2) и (3) легко получить кубическую модель аппроксиманты для f

$$\underbrace{D^{-1}Df(\tau)}_{f} \approx \underbrace{d_1 f(\alpha) + d_2 f(\beta) + d_3 f(0)}_{\Pi} + \underbrace{d_3\theta\alpha\beta\tau}_{\theta Q},\quad \tau\in[a,b],$$

или

$$C(\tau;\, \mathbf{a}, \mathbf{r}_0, \theta) = \Pi(\tau;\, \mathbf{a}, \mathbf{r}_0) + \theta Q(\tau;\, \mathbf{a}), \tag{4}$$

где $\mathbf{a} = (\alpha, \beta, x_0)$, $Q(\tau;\, \mathbf{a}) = \tau(\tau-\alpha)(\tau-\beta)$, $\Pi(\tau;\, \mathbf{a}, \mathbf{r}_0) = (\mathbf{r}_0, \mathbf{d})$, и $\mathbf{d} = [d_1, d_2, d_3]^T$.

Отметим, что параболы Π и Q обладают следующими свойствами:

$$\Pi(\alpha) = f(\alpha),\quad \Pi(\beta) = f(\beta),\quad \Pi(0) = f(0)$$

и

$$Q(\alpha) = Q(\beta) = Q(0) = 0.$$

Как мы видели, реперные точки $[\alpha, f(\alpha)]$, $[\beta, f(\beta)]$ и $[x_0, f(x_0)]$ играют ключевую роль в четырехточечном подходе. В АРСА α и β ассоциируются с левым и правым концом данного подынтервала, соответственно, а x_0 используется как начало локальной системы координат, в которой заданы α и β (см. [3, 4]).

Пусть данные (1) представляют отсчеты кривой $f(x)$. Будем искать интервалы $[a_k, b_k] \equiv [a, b]_k$, $k = 1, 2, \ldots$, на которых модель (4) аппроксимирует кривую кусками кубических сегментов с заданной точностью. При известном репере $\mathbf{r}_0$ модель (4) зависит только от неизвестного параметра θ, и для нахождения k-й актуальной аппроксиманты нам требуется найти граничную точку $[b_k, f(b_k)]$ интервала и оценку $\hat{\theta}_k$. Наш алгоритм ищет $[b_k, f(b_k)]$ (правую границу) и определяет оценку $\hat{\theta}_k$.

Для обнаружения точки останова (breakpoint) в k-м сегменте формируются векторы $\mathbf{r}_{0n} = [f_{-1}, f_{n+s}, f_0]$, $n = n_k, n_k + 1, \ldots$, где $f_* = f(x_*)$, s — регулирующий параметр (в данной версии не используется) и вычисляются следующие величины: оценка $\hat{\theta}_n$ в n-й точке (точечная оценка θ), пошаговое усреднение свободного параметра для m точек $\hat{\theta}_m$ и невязка в n-й точке

$$\delta_n = f_n - \Pi(\tau_n;\, \mathbf{a}_n, \mathbf{r}_{0n}) - \hat{\theta}_m Q(\tau_n;\, \mathbf{a}_n).$$

Критерием обнаружения правого конца интервала служит условие $|\delta_n| > \delta$, где δ — заданное число (параметр настройки). При выполнении этого неравенства точка становится концевой ($b_k = x_n$), иначе она включается в сегмент ($n = n_k + 1$), и цикл повторяется.

Наша цель заключается в разбиении целого интервала $\langle x_1, x_N \rangle$ на подынтервалы так, чтобы на каждом из них соответствующие сегменты функции приближались кубическими параболами. В АРСА эта задача решается в два этапа.

Этап 1. Результатами первого этапа являются:

— K подынтервалов $\langle x_1^k, x_{n_k}^k \rangle$, $k = \overline{1, K}$, где $x_1^1 = x_1$, $x_{n_K}^K = x_N$ и $n_k \geqslant 4$ (каждый подынтервал должен включать не менее четырех точек),

— K оценок $\bar{\theta}^k$ свободного параметра θ и $\sum\limits_{k=1}^{K} (n_k - 3)$ локальных точечных оценок $\hat{y}_*^k$, где

$$\left.\begin{aligned}
&\bar{\theta}^k = \bar{\theta}_{n_k-3}^k, \quad \bar{\theta}_m^k = \frac{m-1}{m}\bar{\theta}_{m-1}^k + \frac{\hat{\theta}_m^k}{m},\\
&\text{или } \hat{\theta}_m^k = \hat{\theta}_{m-1}^k + g_m[f_m^k - \Pi_m^k - \hat{\theta}_{m-1}^k Q_m^k],\\
&\text{где } \ g_m = Q_m^k / \sum_{i=1}^{m} (Q_i^k)^2,\\
&\bar{\theta}_0^k = 0, \quad \hat{\theta}_m^k = \frac{f_{m+2}^k - \Pi_{m+2}^k}{Q_{m+2}^k},\\
&\hat{y}_{m+2}^k = \Pi_{m+2}^k + \bar{\theta}_m^k Q_{m+2}^k,
\end{aligned}\right\} \ m = \overline{1,\, n_k - 3}, \quad \overline{k = 1, K},$$

а Π_{m+2}^k, Q_{m+2}^k вычисляются на основе реперных точек $[x_1^k, f_1^k]$, $[x_2^k, f_2^k]$ (первые две точки подынтервала) и текущей точки $[x_{m+3}^k, f_{m+3}^k]$ (правый конец подынтервала), см. [3]. Необходимо заметить, что оценки реперных координат не вычисляются, так как из свойств Π и Q мы имеем $\hat{y}_i^k \equiv f_i^k$, $i = 1, 2, n_k$.

Поиск $K \geqslant 1$ подынтервалов выполняется по следующей схеме: $k = 1$, шаг А: создается новый подынтервал с первой четверкой точек из (1) (координатами первой точки берутся $x_1^k \equiv x_1$, $f_1^k \equiv f_1$); третья берется за

первую текущую точку на k-м сегменте ($m = 1$), а остальные три точки используются как реперные;

- $m = m + 1$, шаг B: берется следующая подвижная точка с индексом $m + 2$, а первая, вторая и $(m + 2 + s)$-я ($s = 1$) используются в качестве реперных точек для вычисления оценки ординаты $\hat{y}^k_{m+2} = \Pi^k_{m+2} + \bar{\theta}^k_m Q^k_{m+2}$;
- если $|f^k_{m+2} - \hat{y}^k_{m+2}| < \delta$, то переходим к следующей точке (шаг B), иначе текущий подынтервал заканчивается с $n_k = m + 2$ точками (без последней подвижной точки), и начинается новый подынтервал ($k = k + 1$, шаг A) с первой точкой $[x^k_1 \equiv x^{k-1}_{m+2}, f^k_1 \equiv f^{k-1}_{m+2}]$ (левый конец нового сегмента приравнивается правому концу предыдущего).

Этап 2. Обнаружение подынтервалов составляет основное назначение АРСА. По результатам первого этапа можно строить различные интегральные оценки. Мы опишем коротко один из вариантов, в котором используются первые производные на концах подынтервалов $[a_k, b_k]$ для получения интегральной оценки θ k-го сегмента (индекс у a_k, b_k опущен):

$$\hat{\theta}^k = \frac{1}{\gamma^2}\left(\widehat{f}'_a + \widehat{f}'_b - \frac{2}{\gamma}\Delta_{ba}\right),$$

где $\widehat{f}'_a = \frac{1}{ab\gamma}(a^2\Delta_{ba} - \gamma^2(\Delta_{a0} - \hat{\theta}_k a^2 b))$, $\widehat{f}'_\beta = \frac{1}{ab\gamma}(a^2\Delta_{ba} - \gamma^2(\Delta_{b0} - \hat{\theta}_k a b^2))$, $\Delta_{ba} = f_b - f_a$, $\Delta_{a0} = f_a - f_0$, $\Delta_{b0} = f_b - f_0$, $\gamma = b - a$ (детали см. в [2]).

В результате кубические аппроксиманты представляются в виде

$$\hat{f}^k(\tau) = \Pi^k(\tau; a_k, b_k) + \hat{\theta}^k Q^k(\tau; a_k, b_k), \quad \tau = x - x_0, \quad k = \overline{1, K}, \quad (5)$$

где Π^k, Q^k вычисляются через реперные точки, образованные из граничных точек, и точки, расположенные в середине сегмента. Заметим, что в точках стыковки сегментов аппроксиманты совпадают.

2. ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРИЛОЖЕНИЕ АРСА

В .NET термином компонент определяется любая часть программного кода, созданного для работы в виде составной части общего приложения, которую можно независимо распространять и многократно использовать в различных приложениях. Таким образом, компонент представляет собой дискретную единицу программного обеспечения [5]. Нами предусмотрены

два типа компонентов: использующие графический интерфейс пользователя (GUI) и не использующие GUI. Создание компонентов основано на библиотеке LinAlg [6], которая состоит из набора типов и классов, допускающих векторное программирование, и включает широкий диапазон численных, статистических и графических методов.

Точки данных (Data points), собранные в сегмент точки (Segment) и набор всех сегментов на целом интервале (SegmentsAll) представляют три главных объекта в нашем методе. На их основе мы разработали объектно-ориентированное приложение APCA в MS Visual C# с тремя классами/компонентами: `Point4, Segment, SegmentsAll`. Благодаря такой архитектуре мы получаем легкий доступ к данному сегменту и информацию о нем. Это свойство играет ключевую роль при компонентном расширении, связанном с развитием и улучшением APCA в будущем. Хотя LinAlg позволяет легко отображать, рисовать или сохранять результаты и диагностику, для упрощения этих задач мы создали четвертый класс `FormShowPlotSave`.

Платформа .NET использует концепцию пространства имен для группирования типов в логические категории со связанными функциональными возможностями. Прежде чем описать классы и перечисления пространства имен DikTor.APCA, мы коротко изложим правила использования компонентов APCA.

Краткое введение в APCA. Обработка данных на основе списка имен компонентов APCA является простой и может быть реализована различными способами. В следующем примере требуется дополнить данные только одним конкретным параметром δ:

```
string fullPath = @"F:\Csaba\Data\AirLines.txt"; // файл содер-
жит данные
double delta = 0.008;
SegmentsAll segS = new SegmentsAll(fullPath, delta);
```

Раз мы имеем объект `SegmentsAll`, то мы можем отобразить, напечатать или сохранить его соответствующие члены или использовать методы `ShowInWF, PlotInWF` и `FileAppend` векторных и матричных объектов из библиотеки LinAlg:

`segS.TabAPCACoefsTit.ShowInWF(true);` (см. рис. 3)

или выдать на экран с помощью `FormShowPlotSave` форму объекта:

```
FormShowPlotSave formSave;
formSave = new FormShowPlotSave(segS);
formSave.ShowDialog(); (см. рис. 4).
```

Для вычисления оценки кривой и рисования ее графика достаточно написать:

```
segS.Stage2(ApcaOutType.Estimation2);
segS.YEst2.PlotInWF(segS.X, true); (см. рис. 5).
```

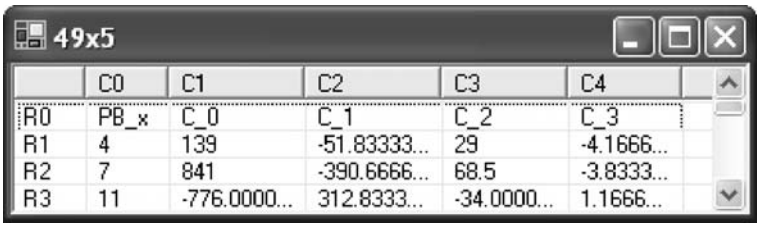

Рис. 3. Правые концы сегментов и четыре коэффициента для первых трех кубических полиномов из найденных 48

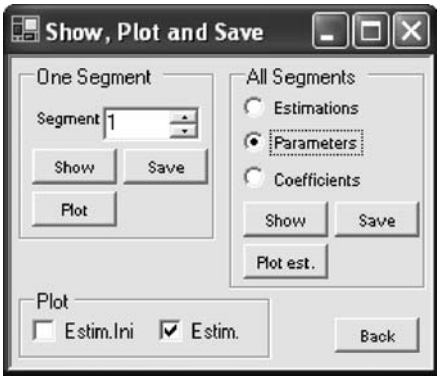

Рис. 4. Диалог FormShowPlotSave

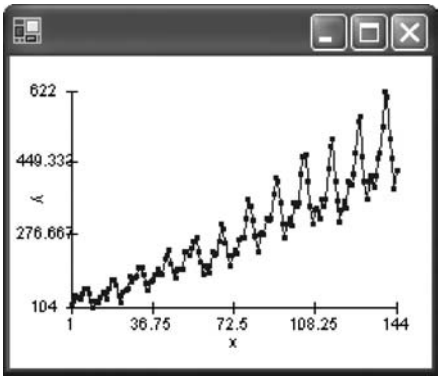

Рис. 5. APCA-оценки продаж авиабилетов

Ниже приводится более подробное описание четырех классов (в Приложении содержатся сигнатуры для конструкторов и методов классов `SegmentsAll` и `Segment` и их описание) и трех перечислений.

Класс `SegmentsAll`. Этот класс является основным в списке имен APCA. Он имеет четыре конструктора (см. Приложение), которые требуют данных и некоторых параметров, таких как значение δ. В конструкторы вы передаете либо имя текстового файла, содержащего данные в одной (Y) или двух колонках (X,Y), разделенных ограничителем, или сами векторы X,Y.

Каждый конструктор делит весь интервал на сегменты и формирует матрицы параметров и коэффициентов APCA. (`Stage1`). Как видно на следующем примере (`private stage1`) метода, после формирования сегмента нельзя уже включать дополнительную точку (так как кубический полином уже будет недостаточен для вычисления кривизны сегмента). Точка добавляется в `ArrayList`, после чего начинается новый сегмент (заметим, что в C# за наименьший индекс берется 0, а не 1):

```
private void stage1() // создание сегментов:
{
   int m = 3;
   bool isAdded;
     _segmentsArLi = new ArrayList();
   Segment seg = new Segment(this, 0, _delta, 1, _rHow, _tHow);
   while (m < _x.Length - 1)
     {
        isAdded = seg.IsAddedThisPoint(m);
        if(isAdded)
              m++;
        else
        {
        seg.NewA0BCentered(); // переопределение Alpha, P0, Beta
        _segmentsArLi.Add(seg);
        seg = new Segment(this, seg.PB.IndexGlobal, _delta, 1,
                     _rHow, _tHow);
        m = seg.PA.IndexGlobal + 3;
        }
      }
    seg.NewA0BCentered(); // переопределение Alpha, P0, Beta
    _segmentsArLi.Add(seg);
}
```

Напомним, что вычисление аппроксиманты и диагностики выполняется на втором этапе (`Stage2`). Чтобы добиться этого, вы должны установить `ApcaOutType`-перечисление или в методе `Stage2` объект `SegmentsAll`:

```
segS.Stage2(ApcaOutType.All);
segS.Residuals.PlotInWF(segS.X, true);
```

или в подходящем конструкторе `SegmentsAll`. Для настройки качества аппроксимации вы можете также установить одно из двух перечислений `ReperHow` и `ThetaHow` параметров. `ReperHow` задает способ выбора второй координаты в трех реперных точках, а `ThetaHow` определяет правило вычисления оценки θ:

```
ReperHow rHow = ReperHow.NoAver;
ThetaHow thHow = ThetaHow.Average;
segS = new SegmentsAll(fullPath, '\t', delta, rHow, thHow,
ApcaOutType.All);
segS.YEst2.ShowInWF(true);
segS.YEst2.PlotInWF(segS.X, true);
```

Класс `Segment`. Основная задача конструктора `Segment` состоит в установке атрибутов объекта `Segment`, таких как три реперные точки PA, PB и P0 (левый и правый конец сегмента и точка x_0 соответственно). Аппроксиманты и диагностика конструктором не вычисляются.

Сегмент содержит не менее чем четыре точки, и его главным членом является `IsAddedThisPoint`. В отличие от `SegmentsAll`, объект `Segment` используется редко. Массив `AllSegments ArrayList` в `SegmentsAll` содержит объекты `Segment`. Использование класса `Segment` иллюстрируется следующими кодовыми строками:

```
Segment seg;
seg = new Segment(segS, 11, delta, rHow, thHow); // используется
редко
seg.YAOB.ShowInWF(true);
seg = (Segment)segS.AllSegments[0]; // возвращает первый сегмент
seg.YEst2.PlotInWF(true);
for(int i=0; i<segS.Length; i++) // отображает оценку для каждо-
го сегмента
{
      seg = (Segment)segS.AllSegments[i];
      seg.YEst2.ShowInWF(true);
}
```

Структура `Point4`. Сегмент состоит из упорядоченных точек, представленных в виде структур `Point4`. В дополнение к таким глобальным переменным, как `GrandParent` (объект `SegmentsAll`), `IndexGlobal` (отнесенных к целому интервалу) и `IndexLocal` предлагается также функция `Shift`:

```
public Point4 Shift(int i),
```

которая возвращает структуру `Point4` (точку) с левой или правой стороны относительно данной точки, в зависимости от того, является ли индекс i отрицательным или положительным, соответственно.

Диалоговая форма `FormShowPlotSave`. В диалоговой форме `FormShowPlotSave` вы можете отобразить, распечатать или сохранить оценки, вычисленные на данном сегменте или на целом интервале (`SegmentsAll`), а также показать и сохранить таблицы параметров и коэффициентов для всего интервала. Для того чтобы использовать этот класс, достаточно его инициировать и использовать `ShowDialog`-метод (см. пункт о кратком введении).

Перечисления `ReperHow`, `ThetaHow` и `ApcaOutType`. `ReperHow` имеет два перечисления, которые определяют, требуется или нет подстройка по реперным ординатам: Average (используется при усреднении реперной ординаты по трем соседним ординатам для сглаживания шума) и NoAver (реперная ордината не модифицируется).

`ThetaHow` имеет два перечисления, определяющие способ оценивания свободного параметра θ: `Average` (пошаговое усреднение параметра θ) и

`RobMon` (рекуррентное вычисление оценки θ на основе рекурсивного МНК по типу процедуры Роббинса–Монро).

`ApcaOutType` имеет два перечисления, определяющих выдачу результата вычислений APCA на втором этапе: `All` (вычисление `YEst2` и диагностик) и `Estimation2` (вычисление только `YEst2`).

3. WINDOWS-ПРИЛОЖЕНИЕ

Чтобы сделать анализ в APCA более удобным, нами также разработано Windows-приложение [7]. В APCA реализован новый метод, и Windows-приложение позволяет не только использовать APCA для обработки реальных данных, но и выполнять тщательное их исследование.

В этом разделе дается краткое описание пользовательского графического интерфейса этого приложения, в котором показано, как можно прочитать или сгенерировать данные, вычислить оценки, подобрать параметры и использовать диагностики (вычисление производных, невязки и т.п.).

На рис. 6 показан графический интерфейс пользователя (GUI) Windows-приложения. Далее мы опишем три части GUI: *Data*, *Run APCA* and *Outputs*.

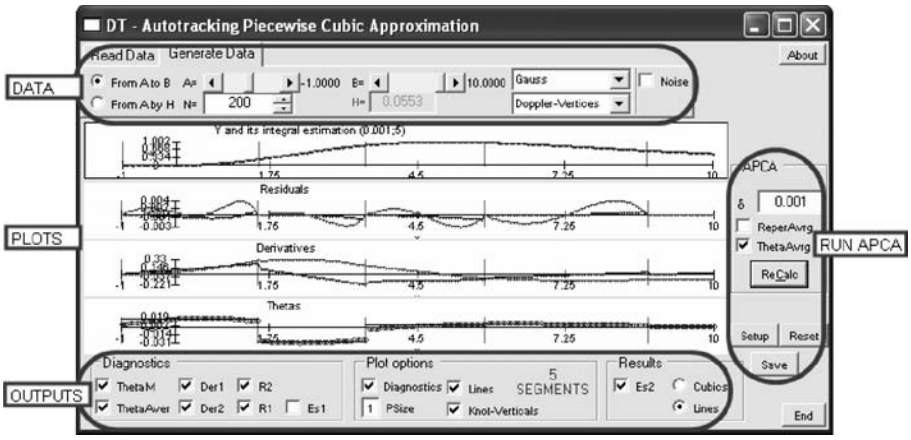

Рис. 6. Графический интерфейс пользователя Windows-приложения

Начнем с центральной части, связанной с графическим представлением данных и аппроксимант (`Y,YEst2` и `YEst2`), остатков (`R2, R1`), производных (`Der1, Der2`) и свободных параметров.

Data. В *TabControl* с двумя *TabPages* можно прочитать данные, подготовленные в виде текстовых файлов, или сгенерировать данные с аддитивными ошибками или без ошибок (см. рис. 7):

Рис. 7. Установки для чтения и генерации данных

1) файл с расширением *.txt должен содержать один или два столбца Y или X, Y соответственно (TAB delimited);

2) генерирование N точек от A до B: H = (B-A)/N;

3) генерирование N точек от A с шагом H;

4) выборка из N точек генерируется на основе указанной функции;

5) добавление гауссова шума к точкам;

6) если значение Seed отсутствует, тогда с помощью клавиши Enter (или нажатием кнопки ReCalc) генерируется выборка с новыми ошибками и APCA находит новые сегменты и аппроксиманту;

7) отношение сигнал/шум (SNR). APCA дает хорошие результаты для данных с небольшими ошибками (SNR >= 40) в зависимости от сложности кривой;

8) корень квадратный из дисперсии ошибки.

Run APCA. В APCA group box нажатием кнопки Run запускаются в работу обе стадии (Stage1 и Stage2) с установленными параметрами (рис. 8):

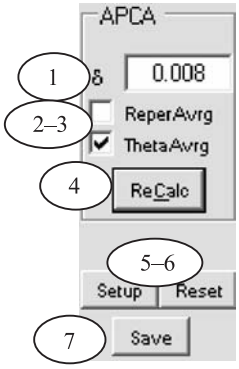

Рис. 8. Работа с APCA

1) δ является основным настроечным параметром в APCA. Чем меньше δ, тем выше точность аппроксимации и короче сегменты. Для бóльших ошибок используются бóльшие значения δ (чтобы получить более длинные сегменты);

2) усреднение реперных ординат по трем соседним точкам (локальное сглаживание для зашумленных данных);

3) последовательное усреднение свободного параметра θ (рекуррентное вычисление оценки θ на основе рекурсивного МНК по типу процедуры Роббинса–Монро);

4) повторное вычисление APCA при установке нового значения параметра δ;

5) изменение других установок при генерировании данных (открывает новое окно);

6) перезагрузка приложения;

7) сохранение параметров сегмента или сегментов и вывод графика аппроксимант (открывает описанную выше диалоговую форму `FormShowPlotSave`).

Outputs. В группу Outputs (рис. 9) включены опции для выдачи графиков оценок и различных диагностик:

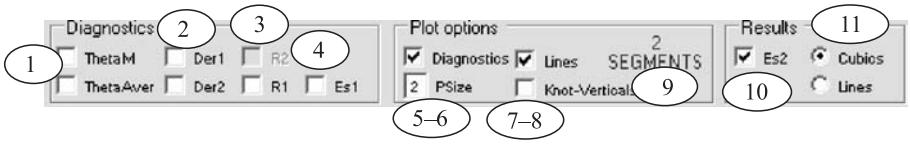

Рис. 9. APCA-оценки и диагностики

1) вывод графиков значений оценок $\hat{\theta}_m$ для текущей четверки точек и усредненной на всем сегменте оценки $\hat{\theta}^k$;

2) выдача графиков первой и второй производных, полученных по `YEst2`;

3) график остатков `R1=Y-Est1`. Для выдачи графика `R2=Y-Est2` надо выбрать Lines на панели Results;

4) график оценки `YEst1`, полученной по четырем точкам (локальная оценка);

5) отображение панели диагностики (Diagnostics);

6) установка размера точки на графиках;

7) соединение точек (`Y`) на графиках прямыми отрезками;

8) индикация обнаруженных узлов (концевые точки сегментов) вертикалями;

9) отображение числа обнаруженных кубических сегментов;

10) график кусочно-кубической аппроксиманты `YEst2` (глобальная оценка);

11) соединение точек на графике `YEst2` кубическими дугами или отрезками.

4. ПРИМЕРЫ РАБОТЫ WINDOWS-ПРИЛОЖЕНИЯ APCA

В предыдущих разделах мы изложили новый алгоритм для кусочно-кубического приближения одномерных действительных функций на промежутке и

его реализацию в виде объектно-ориентированного приложения. Однако наибольший интерес вызывают результаты, получаемые предложенным методом.

В этом разделе приведены примеры, иллюстрирующие работу APCA. Наряду с полученными аппроксимантами мы также приводим некоторые характеристики, такие как остатки (невязки), оценки первой и второй производных глобальной аппроксиманты, а также поведение оценки свободного параметра θ, играющего ключевую роль в предложенном методе. Значения δ и число обнаруженных сегментов показаны в верхней части рисунков.

Первый из трех примеров является простым, однако он демонстрирует качественные характеристики APCA. На рис. 10 показаны графики функции $\cos x$ (точки) и ее APCA-оценка `YEst2` (сплошная линия) на заданном интервале при $\delta = 0{,}005$, а также разбиение кривой на кусочно-кубические сегменты (вертикали). Рис. 11 иллюстрирует отклонения локальной оценки YEst1 (жирная линия) и `YEst2`. Заметим, что, как и следовало ожидать, интегральные *точечные* оценки немного больше локальных. На рис. 12 видно, что хотя первая производная выглядит почти гладкой, поведение второй производной (толстая линия) в точках стыковки выглядит хуже. Рис. 13 демонстрирует изменение в каждой точке `ThetaM` $\hat{\theta}_m^k$ и на каждом сегменте `ThetaAverage` $\bar{\theta}^k$.

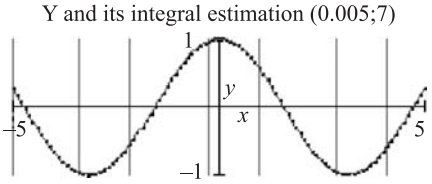

Рис. 10. Косинус и его кусочно-кубическая аппроксиманта

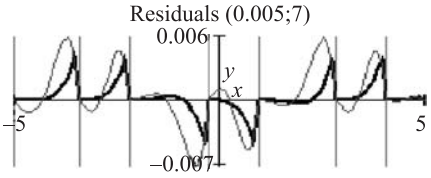

Рис. 11. Остатки для локальной и глобальной аппроксимант

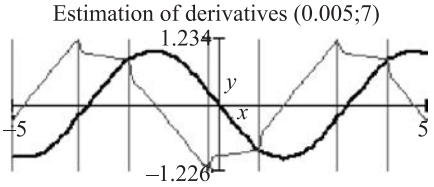

Рис. 12. Первая и вторая производные

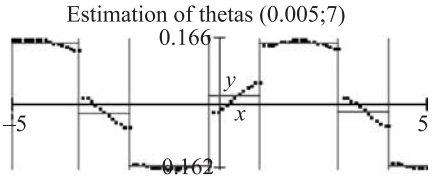

Рис. 13. Оценки свободного параметра θ

Данные второго примера были получены при численном моделировании теплоемкости электронов (ETC) для молекулы D-ацетона [1], и они практически не содержат больших ошибок. На рис. 14 показаны графики данных и

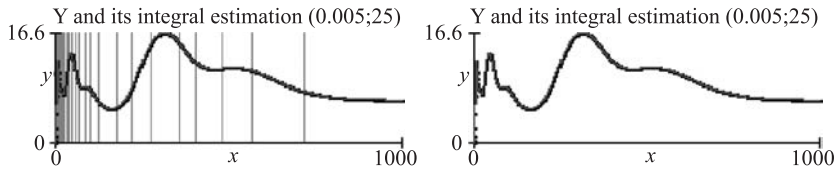

Рис. 14. ЕТС и их аппроксимация

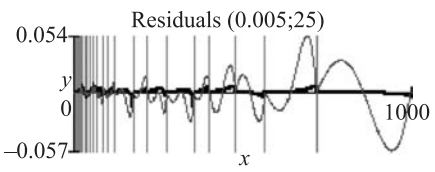

Рис. 15. Локальные и интервальные остатки

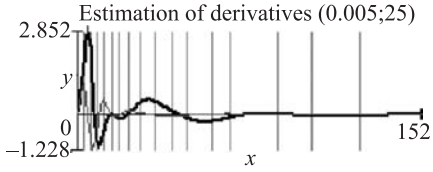

Рис. 16. Оценки первой и второй производных

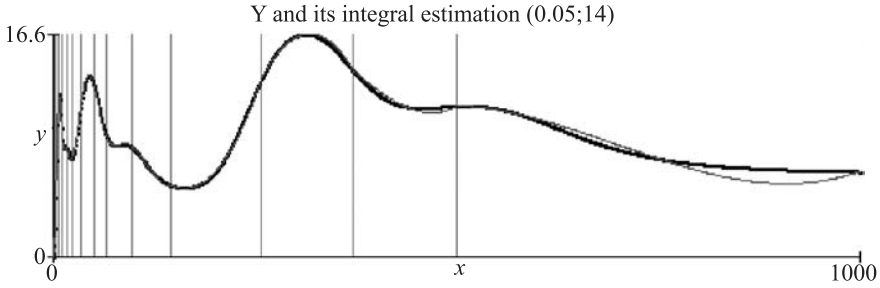

Рис. 17. ЕТС-аппроксиманта для $\delta = 0{,}05$

их АРСА аппроксимации, на рис. 15 — невязки, а на рис. 16 — оценки производных. Качество автоматически обнаруженных 25 аппроксимант соответствует выбору значения параметра ($\delta = 0{,}005$) и вполне удовлетворительное. В левой части, где график более динамичен, подынтервалы оказались более короткие. Чтобы проиллюстрировать роль параметра аппроксимации δ, на рис. 17 показаны результаты работы АРСА для $\delta = 0{,}05$. Мы видим, что увеличение δ в 10 раз почти вдвое уменьшило число сегментов (14), однако качество аппроксимации также ухудшилось.

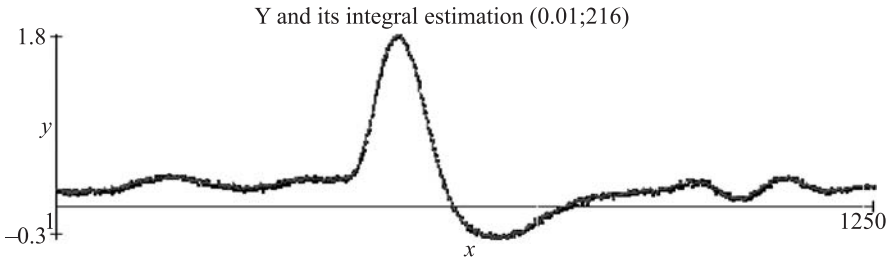

Рис. 18. Аппроксимация данных с осциллографа (1250 точек)

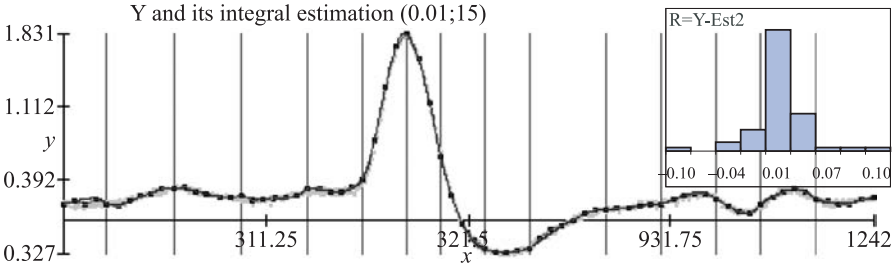

Рис. 19. Кусочно-кубическая аппроксиманта для данных с осциллографа и гистограмма остатков (на основе каждой 17-й точки)

Последний пример (рис. 18, 19) показывает результаты аппроксимации в режиме слежения с использованием реальных данных, полученных с цифрового осциллографа. В отличие от предыдущих примеров, эти данные слегка зашумлены, поэтому значения в реперных точках усреднены по трем ближайшим точкам, а оценка `ThetaAverage` вычислялась по процедуре типа Роббинса–Монро. Рис. 19 показывает аппроксиманты, (опция Cubics), вычисленные на основе каждого 17-го измерения (в обоих случаях $\delta = 0{,}01$). Хотя качество приближения существенно не изменилось, число сегментов существенно уменьшилось (с 216 до 15).

ЗАКЛЮЧЕНИЕ

Сформулируем основные выводы из приведенных результатов:

— получена приемлемая гладкость оценки первой производной для табулированной функции, представленной данными с малыми ошибками;

— меньшее значение параметра δ дает больше число сегментов и улучшает точность аппроксимации;

— для данных с шумом увеличение значения δ целесообразно;

— цель состоит в подборе такого значения δ, которое обеспечит желаемое качество аппроксимации и приемлемое число сегментов.

Цель авторов состояла в реализации алгоритмов недавно предложенного метода кусочно-кубической аппроксимации с использованием нового объектно-ориентированного языка MS Visual C#. Компоненты библиотеки позволяют разрабатывать Windows-приложения и создавать конструкции APCA-аппроксиманты без особых усилий. Дальнейшее развитие этих работ предполагается вести в направлении аппроксимации поверхностей и обработки изображений.

ПРИЛОЖЕНИЕ

В Приложении мы приводим резюме для классов `SegmentsAll` и `Segment`, список имен, сигнатуры конструкторов и методов.

Класс SegmentsAll

`SegmentsAll` — основной класс APCA. Он имеет четыре конструктора.

Конструкторы:

- `public SegmentsAll(string fullPath, double delta)` — разделяет весь интервал на сегменты, основанные на чтении вектора(ов) `Y`, `(X,Y)` данных из текстового файла и значения параметра, а также вычисляет APCA параметры и коэффициенты (Stage1); использует `ReperHow.NoAver`, `ThetaHow.Average`.

- `public SegmentsAll(string fullPath, char delimiter, double delta, ReperHow rHow, ThetaHow thHow, ApcaOutType outType)` — разделяет весь интервал на сегменты, основанные на чтении вектора(ов) `Y`, `(X,Y)` данных из текстового файла и значения параметра, а также вычисляет APCA-параметры и коэффициенты (Stage1); последний параметр определяет то, что вычислено на втором этапе (оценки и/или диагностики).

- `public SegmentsAll(VectorD y, VectorD x, double delta)` — разделяет весь интервал на сегменты, основанные на векторах и параметрах и обеспечивает вычисления параметров и коэффициентов (stage1); использует `ReperHow.NoAver, ThetaHow.Average`.

- `public SegmentsAll(VectorD y, VectorD x, double delta, ReperHow rHow, ThetaHow thHow, ApcaOutType outType)` — разделяет весь интервал на сегменты, основанные на векторах и параметрах и обеспечивает вычисления параметров и коэффициентов (Stage1); последний параметр определяет, что вычисляется на втором этапе (Stage2) (оценки и/или диагностика).

Открытые члены
Функции:

- `LinAlg.VectorD Approx_TabApcaParams(LinAlg.VectorD X)` — вычисляет кубическую аппроксиманту для всего вектора `X` на основе `TabApcaParams`.
- `double Approx_TabApcaParams(double x, LinAlg.MatrixD a)` — вычисляет кубическую аппроксиманту для значения `x` на основе данного a (`=TabApcaParams`).
- `double Approx_TabApcaParams(double x)` — вычисляет кубическую аппроксиманту для значения `x` на основе `TabApcaParams`. Эта функция неэффективна при последовательных вызовах, где лучше использовать `Approx_TabApcaParams(VectorD X)`.
- `LinAlg.VectorD Approx_TabCubApprCoefs(LinAlg.VectorD X)` — вычисляет кубическую аппроксиманту для целого вектора `X` на основе `TabAPCACoefs`.
- `double Approx_TabCubApprCoefs(double x, LinAlg.MatrixD a)` — вычисляет кубическую аппроксиманту для значения `x` на основе данного a (`= TabAPCACoefs`).
- `double Approx_TabCubApprCoefs(double x)` — вычисляет кубическую аппроксиманту для значения `x` на основе `TabAPCACoefs`. Эта функция неэффективна при последовательных вызовах, где лучше использовать `Approx_TabCubApprCoefs (VectorD X)`.
- `public void Stage2(ApcaOutType outType)` — вычисляет `EstY2` или `EstY2` и необходимые производные, если `ApcaOutType` равно `Estimation2` или `All` соответственно.

Свойства

- `System.Collections.ArrayList AllSegments` — возвращает сегменты в виде массива `ArrayList`.

- `int Count` — возвращает число всех найденных сегментов (см. также `Length`).
- `LinAlg.VectorD EstD1Y` — возвращает оценку первой производной для `y`.
- `LinAlg.VectorD EstD2Y` — возвращает оценку второй производной для `y`.
- `int[] IndsAlpha` — возвращает индекс левого конца (для всех сегментов).
- `int[] IndsBeta` — возвращает индекс правого конца (для всех сегментов).
- `int Length` — возвращает число всех точек (см. также `Count`).
- `LinAlg.VectorD Residuals1` — возвращает остатки локальной оценки `Y:R1=Y-YEst1` (Stage2).
- `LinAlg.VectorD Residuals2` — возвращает остатки локальной (второй) оценки `Y`, вычисленной на основе `Approx_TabApcaParams: R2=Y-YEst2` (Stage2).
- `LinAlg.MatrixD TabAPCACoefs` — возвращает для каждого сегмента (= строка): xb и «кубические» коэффициенты `c0`, `c1`, `c2`, `c3` (= столбцы).
- `LinAlg.MatrixS TabAPCACoefsTit` — `TabAPCACoefs` со строкой заголовка.
- `LinAlg.MatrixD TabApcaParams` — возвращает для каждого сегмента (= строка): `xa`, `x0`, `xb`, `ya`, `y0`, `yb`, `theta` (= столбцы). В последнем сегменте `x`a не подстроен (`xa` может быть меньше предпоследнего `xb`).
- `LinAlg.MatrixS TabApcaParamsTit` — `TabApcaParams` со строкой заголовка.
- `LinAlg.VectorD ThetaAvarage` — возвращает рекурсивную оценку $\overline{\theta}^k$ (основной параметр) для каждого сегмента (они затем корректируются на втором этапе).
- `LinAlg.VectorD ThetaM` — возвращает оценку $\hat{\theta}_m^k$ для каждой точки.

- `int ThisNbOfPointsWasNeeded` — в последнем сегменте одна или две точки могут быть заимствованы (при необходимости) из предпоследнего сегмента, после чего первая координата последнего сегмента будет меньше чем первая координата предыдущего.
- `LinAlg.VectorD X` — возвращает первую координату из входного массива.
- `LinAlg.MatrixD XYEst12` — возвращает первую и вторую координаты `Y`, `YEst1` и `YEst2`.
- `LinAlg.VectorD Y` — возвращает вторую координату точек из входного массива.
- `LinAlg.VectorD YEst1` — возвращает локальные оценки `Y` (Stage1).
- `LinAlg.VectorD YEst2` — возвращает локальные оценки `Y` (Stage2).

Класс Segment
Класс `Segment` имеет только один конструктор.

- `public Segment(SegmentsAll parent, int ia, double delta, ReperHow rHow, ThetaHow thHow)` — устанавливает реперные точки `PA`, `PB`, `P0` и другие атрибуты сегмента, такие как `Alpha, Beta, Parent` и `how` параметры.

Открытые члены

Методы:

- Новый метод аппроксимации может быть использован другими аппроксимационными методами, такими как нейронные сети и вейвлеты. Это дает не только новый инструмент, но открывает новые возможности для них [8]. Дальнейшее развитие метода будет связано с обобщением APCA для аппроксимации многомерных поверхностей `bool IsAddedThisPoint(int im)` — `True`, если подвижная, текущая точка (с индексом `im`) включается во временный сегмент;
- `void NewA0BCentered()` — центрирует `x0` и настраивает `alpha` и `beta`;
- `void Stage2(DikTor.APCA.ApcaOutType outType)` — вычисляет для данного сегмента значения EstY2, или EstY2 и необходимые производные, если `ApcaOutType` равно `Estimation2` или `All` соответственно.

Свойства:

- `double Alpha` — возвращает расстояние (отрицательное) между левым концом сегмента и `x0 (=xa-x0)`;
- `double Beta` — возвращает расстояние между правым концом сегмента и `x0 (=xb-x0)`;
- `LinAlg.VectorD D1F` — возвращает первую производную `Y`;
- `LinAlg.VectorD D2F` — возвращает вторую производную `Y`;
- `int Length` — возвращает число точек в сегменте;
- `DikTor.APCA.SegmentsAll Parent` — возвращает объект `SegmentsAll`, содержащий данный сегмент;
- `DikTor.APCA.Point4 P0` — возвращает точку `[x0,y0]` (она может быть в центре сегмента);

  `DikTor.APCA.Point4 PA` — возвращает левый конец сегмента;
- `DikTor.APCA.Point4 PB` — возвращает правый конец сегмента;
- `DikTor.APCA.ReperHow ReperHow` — сообщает, усреднять ли вторые координаты реперных точек по соседним точкам;
- `double ThetaAverage` — возвращает $\bar{\theta}^k$ для данного сегмента, используя рекурсивное усреднение.
- `DikTor.APCA.ThetaHow ThetaHow` — сообщает, как вычисляется `theta` для данного сегмента (только чтение);
- `LinAlg.VectorD ThetaM` — возвращает $\hat{\theta}_m^k$ для каждой точки данного сегмента;
- `int ThisNbOfPointsWasNeeded` — если имеются только две или три точки слева сегмента, тогда две или одна точки должны быть заимствованы у предыдущего сегмента соответственно (в этом случае последнее `alpha_x` будет меньшим чем предыдущее `beta_x`);
- `LinAlg.VectorD X` — возвращает первые координаты точек сегмента;
- `LinAlg.VectorD XAOB` — возвращает первые координаты реперных точек;

- `LinAlg.MatrixD XYEst12` — возвращает первою и вторую координаты точек, `YEst1` и `YEst2` (для точек данного сегмента);
- `LinAlg.MatrixS XYEst12Tit` — `XYEst12` со строкой заголовка;
- `LinAlg.VectorD Y` — возвращает вторые координаты точек для сегментов;
- `LinAlg.VectorD YAOB` — вторые координаты реперных точек;
- `LinAlg.VectorD YEst1` — возвращает локальные оценки `Y` для данного сегмента (stage1).
- `LinAlg.VectorD YEst2` — возвращает интегральную оценку `Y` для данного сегмента (stage2).

Статические функции `Par, W1_3 DPar*, DDPar*, Q, DQ*` и `DDQ*` используются при вычислениях модели (* при коррекции значения `theta` на втором этапе):

- `static public double Par(double t, double a, double b, double Fa, double Fb, double F0`
- `static public double Q(double t, double a, double b)`
- `static public double W1(double t, double a, double b)`
- `static public double W2(double t, double a, double b)`
- `static public double W3(double t, double a, double b)`
- `static public double DPar(double t, double a, double b, double Fa, double Fb, double F0)`
- `static public double DDPar(double a, double b, double Fa, double Fb, double F0)`
- `static public double DQ(double t, double a, double b)`
- `static public double DDQ(double t, double a, double b)`

ЛИТЕРАТУРА

1. *Костенко Б. Ф., Прибыш Я.* Теплоемкость электронов в плазме, образующейся при схлопывании кавитационного пузырька в D-ацетоне. Сообщение ОИЯИ Р11-2004-139. Дубна, 2004.

2. *Hagiwara K. et al.* // Phys. Rev. D. 2002. V. 66. P. 010001;

   http://pdg.lbl.gov/2002/contents_plots.html

3. *Дикусар Н. Д., Торок Ч.* Кусочно-кубическая аппроксимация в режиме автоматического слежения. Сообщение ОИЯИ Р11-2004-187. Дубна, 2004.

4. *Дикусар Н. Д.* Кусочно-кубическое приближение и сглаживание кривых в режиме адаптации. Сообщение ОИЯИ Р10-99-168, Дубна, 1999.

5. *Гуннерсон Э.* Введение в C#. С.-Пб.: Издательский дом «Питер», 2001.

6. *Török Cs.* Visualization and data analysis in the MS .NET Framework. JINR Commun. E10-2004-136. Dubna, 2004;

   http://www.jinr.ru/publish/Preprints/2004/136(e10-2004-136).pdf.

7. *Glynn J. et al.* Professional Windows GUI programming using C#. Wrox, 2002.

8. *Kepič T., Török Cs., Dikoussar N. D.* Wavelet compression, International Seminar of Computing Statistics, 2004, Bratislava (to be published).

Получено 21 декабря 2004 г.

Редактор *М. И. Зарубина*

Подписано в печать 28.02.2005.
Формат 60 × 90/16. Бумага офсетная. Печать офсетная.
Усл. печ. л. 1,38. Уч.-изд. л. 1,66. Тираж 290 экз. Заказ № 54808.

Издательский отдел Объединенного института ядерных исследований
141980, г. Дубна, Московская обл., ул. Жолио-Кюри, 6.
E-mail: publish@pds.jinr.ru
www.jinr.ru/publish/