УДК 004.942

С.В. Рогачев[1]

ВЫЧИСЛЕНИЕ КРАЕВОЙ И УГЛОВОЙ СПЕЦИАЛЬНЫХ ФУНКЦИЙ ВОЛНОВЫХ КАТАСТРОФ В КОМПЛЕКСЕ ПРОГРАММ WAVECAT

В статье рассматривается создание двух программных модулей для системы численного расчета специальных функций волновых катастроф Wavecat. Показан процесс формализации вычисления СВК на языке программирования ANSI C с использованием библиотек программного комплекса. Построены новые сечения для СВК, соответствующих особенностям дифференцируемых отображений F_4 и $A_1A_2A_1A_1$. Предложены улучшения для комплекса программ, необходимость которых была продиктована спецификой разработанных программных модулей.

Ключевые слова: *теория катастроф, излучение, волны, комплекс программ, численные методы.*

S.V. Rogachev

COMPUTING OF EDGE AND CORNER SPECIAL FUNCTIONS OF WAVE CATASTROPHES IN WAVECAT COMPUTING SOFTWARE

The paper describes the development of two program modules for numerical system for computing the special functions of wave catastrophes – "Wavecat". It shows the process needed to formalize computing of two special functions on the ANSI C programming language with usage of program libraries provided by the described software. A set of new cross-sections was painted by the system for two catastrophes F_4 and $A_1A_2A_1A_1$. A number of improvements for the software complex were proposed because of the necessity shown during development of new program modules.

Keywords: *theory of catastrophes, radiation, waves, computing software, numeric methods.*

Введение

Wavecat – информационная система, посвященная волновой теории катастроф (см. [1–4]), а также одноименный комплекс программ, созданный с целью вычисления специальных функций волновых катастроф (СВК) без использования какого-либо стороннего программного обеспечения [5; 6; 7]. В ряде статей рассматривались архитектурные особенности этого программного обеспечения (например [8; 9]) и построения (графики спецфункций) [10], совершенные с его помощью. Цель настоящей статьи – демонстрация подходов к созданию вычислительных модулей описываемой системы, осуществляющих численное нахождение значений СВК краевой (F_4) и угловой ($A_1A_2A_1A_1$) катастроф.

Напомним, что краевые катастрофы связаны с фокусировками, в которых участвуют вторичные лучевые поля (в терминах геометрической теории дифракции), возбужденные первичным полем на кромках тел. Для угловых катастроф характерны вторичные поля, образованные в точках излома кромки [11]. При вычислении значений СВК, необходимых при построении равномерных асимптотик [12], с использованием метода ОДУ [13], это приводит к использованию значений других СВК и их производных в системах уравнений и начальных значениях.

Так, особенность F_4, подразумевающая каустики ГО и краевых лучей, в соответствующей системе ОДУ и начальных условиях содержит функцию Эйри, ее первую производную и функцию Френеля. Катастрофа $A_1A_2A_1A_1$, имея еще более сложную структуру, задается системой уравнений со значениями СВК B_3, ее первыми и вторыми производными. Результаты вычисления B_3 используются и при задании начальных значений спецфункции.

[1] Аспирант АНО ВО «Российский новый университет».

При реализации программного модуля для вычисления названных СВК их специфика выливается в необходимость получения промежуточных данных для других спецфункций в каждой рассматриваемой точке $\vec{S}(\lambda, a)$. Необходимость такой организации вычислений несколько осложняет решение в текущей системе без ее доработки, но не делает их невозможными. Тем не менее, существует ряд улучшений, которые будут описаны в дальнейшем.

Особенность F_4

Перейдем к СВК особенности F_4. СВК этой особенности определяется осциллирующим интегралом [11]:

$$V(\lambda_1, \lambda_2, \lambda_3) = \\ = \int_0^{+\infty} dz \int_{-\infty}^{+\infty} \exp\left(i\left(kz^2 + x^3 + \lambda_1 x + \lambda_2 z + \lambda_3 zx\right)\right) dx. \quad (1)$$

Опустив определение понятий фундаментального вектора, а также механизм перехода к системе ОДУ (детали можно почерпнуть в соответствующих публикациях), приведем систему (2), используемую для вычисления СВК:

$$\frac{dV}{dt} = \lambda_3^1 \frac{\partial V}{\partial \lambda_3(t)}, \\ \frac{dV^1}{dt} = \lambda_3^1 \frac{\partial V^1}{\partial \lambda_3(t)}. \quad (2)$$

Для вычисления соответствующих первой и второй производных используются выражения (3):

$$U_{02} = -\frac{k}{2}\left(i\left(\lambda_2 V - i\lambda_3 V^1\right) + A_i^+(\lambda_1)\right), \\ \frac{\partial V}{\partial \lambda_3} = k\frac{1}{2}\left(i\frac{\partial A_i^+(\lambda_1)}{\partial \lambda_1} - \lambda_2 V^1 + \frac{i}{3}\lambda_3\left(\lambda_{1V} - i\lambda_3 U_{02}\right)\right) \equiv U_{01}, \\ U_{22} = \frac{k}{2i}\left(1 + \lambda_2 U_{02} + \lambda_3 U_{01}\right), \quad (3) \\ \frac{\partial V^1}{\partial \lambda_3} = -\frac{i}{3}\left(\lambda_1 U_{02} - i\lambda_3 U_{22}\right).$$

Начальные условия в точке $\vec{S}^0 = \left(\lambda_1^1, \lambda_2^1, 0\right)$:

$$V(S^0) = A_i^+(\lambda_1^1) \times F_r^k(\lambda_2^1), \\ V^1(S^0) = \frac{\partial A_i^+(\lambda_1^1)}{\partial \lambda_1} \times F_r^k(\lambda_2^1). \quad (4)$$

Эта система ОДУ относительно параметра t может быть разрешена с использованием одной из известных разностных схем, например, метода Рунге – Кутты, который применен в программном модуле.

Создадим новый программный модуль для системы Wavecat на языке ANSI C. Напомним, что описываемый комплекс программ имеет в комплекте собственные языковые средства (WaveLang) [14], но этот специализированный язык пока приспособлен лишь для написания модулей, соответствующих особенностям основных катастроф.

Подключим необходимые заголовочные файлы:

```
#include <kernel/integration/cmplx_runge_kutta.h>
#include <kernel/core/point_array.h>
#include <math.h>
```

Таким образом, становится возможным использовать метод Рунге – Кутты с комплексной арифметикой и сохранять результаты вычислений в массиве точек.

Выполним определение необходимых констант, они будут использоваться в дальнейшем в качестве индексов в хранилищах параметров СВК, компонентов фундаментального вектора и временного хранилища вычислительного контекста.

```
enum parameters {
   LAMBDA_1 = 0,
   LAMBDA_2,
   LAMBDA_3,
   K
};
enum components {
   V = 0,
   V1
};
enum storage_entries {
   Ai = 0,
   Aid
};
```

Для того чтобы вычислительное ядро могло автоматически связывать параметры с их значениями, требуется определить массив символьных имен:

```
static char *par_names[] = {«l1», «l2», «l3», «k», NULL};
```

Система ОДУ, приведенная выше, полностью отражена в следующей функции:

```
void cmplx_catastrophe_Fsub4_function(const catastrophe_t *const catastrophe,const double t, const double complex *y,double complex *const f)
{
```

```
    double l1, l2, l3;
    double complex U01, U02, U22, U13;

    l1 = PARAM(LAMBDA_1) * t;
    l2 = PARAM(LAMBDA_2) * t;
    l3 = PARAM(LAMBDA_3) * t;

    U02 = 0.5 * (-PARAM(K)) * (I *
      (l2 * y[V] - I * l3 * y[V1]) +
      STORAGE_COMPLEX(Ai));

    U01 = 0.5 * PARAM(K) * (I *
      STORAGE_COMPLEX(Aid) - l2 *
      y[V1] + I/3.0 * l3 *(l1 * y[V]
      - I * l3 * U02));

    U22 = 0.5 * PARAM(K) / I * (1.0 +
      l2 * U02 + l3 * U01);

    U13 = -I/3.0 * (l1 * U02 - I * l3
      * U22);

    f[V] = PARAM(LAMBDA_3) * U01;
    f[V1] = PARAM(LAMBDA_3) * U13;
}
```

Вычислительному ядру требуется не только функция (в терминах императивного программирования, а не в математическом смысле) с описанием системы уравнений, но и процедура, осуществляющая координацию вычислений для конкретной СВК. Такая процедура приведена ниже:

```
static void calculate(catastrophe_t
       *const   catastrophe,const
       unsigned  int  i,  const
       unsigned int j)
{
    cmplx_equation_t *equation;
    point_array_t *point_array;

    double module;
    double phase;

    double complex F2;

    const double g13 = 2.678938534;
    const double g23 = 1.354117939;

    double divsqrt3 = 1.0 / sqrt(3.0);

    assert(catastrophe);

    equation = catastrophe->equation;
    point_array   =   catastrophe-
    >point_array;

    assert(equation);
    assert(point_array);

    equation->initial_vector[V]
    = 0.5 * sqrt(M_PI) * cexp(I *
    PARAM(K) * M_PI / 4.0);

    equation_set_function(equation,
    cmplx_catastrophe_Frenaile_
    function);
    cmplx_runge_kutta(0.0,     1.0,
    0.01, catastrophe);
    F2    =    equation->resulting_
    vector[V];

    equation->initial_vector[V]   =
    divsqrt3 * g13;
    equation->initial_vector[V1]  =
    -divsqrt3 * g23;
    equation_set_function(equation,
    cmplx_catastrophe_Airy_
    function);
    cmplx_runge_kutta(0.0,     1.0,
    0.01, catastrophe);

    STORAGE_COMPLEX(Ai) = equation-
    >resulting_vector[V];
    STORAGE_COMPLEX(Aid) = equation-
    >resulting_vector[V1];

    equation->initial_vector[V]   =
    STORAGE_COMPLEX(Ai) * F2;
    equation->initial_vector[V1]  =
    STORAGE_COMPLEX(Aid) * F2;
    equation_set_function(equation,
    cmplx_catastrophe_Fsub4_
    function);
    cmplx_runge_kutta(0.0,     1.0,
    0.008, catastrophe);

    module    =    cabs(equation-
    >resulting_vector[V]);
    phase  =  (180.0  /  M_PI)  *
    carg(equation->resulting_
    vector[V]);

    point_array->array[i][j].module
    = module;
    point_array->array[i][j].phase =
    phase;
}
```

Интересно, что обычно вычислительное ядро создает лишь один объект задания-катастрофы типа catastrophe_t, который используется при вычислениях конкретной СВК и обладает особенными характеристиками, свойственными этой спецфункции. Например, каждой СВК ставится в соответствие объект типа equation_t, описывающий систему ОДУ и связанное с ней состояние. Таким образом, система изначально не предназначалась для случаев, когда требуется получить ряд промежуточных результатов – значений других СВК и их производных в той же точке. В примере этот недостаток исправляется

временной модификацией объекта equation_t, путем вызова equation_set_function(), куда в качестве аргумента передаются библиотечные ОДУ, описывающие функцию Эйри и интеграл Френеля. Полученные значения функций записываются для временного хранения в хранилище вычислительного контекста и в локальные переменные.

Стоит обратить внимание, что рассчитанное значение функции Эйри не может быть сохранено в глобальной переменной, поскольку вычислительное ядро способно прозрачно выполнять декомпозицию задачи на несколько независимых параллельных [15], что приводит к порче сохраненного значения ввиду конкурентного доступа. Таким образом, значение должно быть как-то связано с текущим вычислительным контекстом. Для доступа к таким значениям в Wavecat существует семейство макросов STORAGE_COMPLEX/STORAGE_REAL. Поскольку промежуточный результат для интеграла Френеля не используется в главной системе уравнений, соответствующее значение сохраняется в локальной переменной.

Для комплексного значения СВК в данной точке вычисляются модуль и фаза, которые записываются в соответствующую позицию массива результатов.

Для того чтобы новая СВК стала доступной вычислительному ядру, вычислительный модуль требуется описать стандартным для Wavecat образом:

```
static catastrophe_desc_t cmplx_
catastrophe_Fsub4_desc = {
   .type = CT_COMPLEX,
   .sym_name = «Fsub4»,
   .fabric = catastrophe_fabric,
   .num_parameters = 4,
   .par_names = par_names,
   .equation.cmplx = cmplx_
    catastrophe_Fsub4_function,
   .num_equations = 2,
   .calculate = calculate
};
```

Также модулю необходима функция-конструктор:

```
static int cmplx_catastrophe_Fsub4_
     init(void) __attribute__
     ((constructor));
static int cmplx_catastrophe_Fsub4_
     init(void)
{
   fprintf(stderr, «Catastrophe
   Fsub4 (complex)
   initialization.\n»);
   register_catastrophe_
   desc(&cmplx_catastrophe_Fsub4_
   desc);
}
```

После сборки модуля и перезапуска сервиса вычислительного ядра можно выполнить первые построения. Для начала посчитаем спецфункцию для случая, когда каустики ГО и краевых лучей совпадают $(\lambda_3 = 0)$. Поскольку подобные построения уже публиковались, это позволит проверить корректность созданного вычислительного модуля.

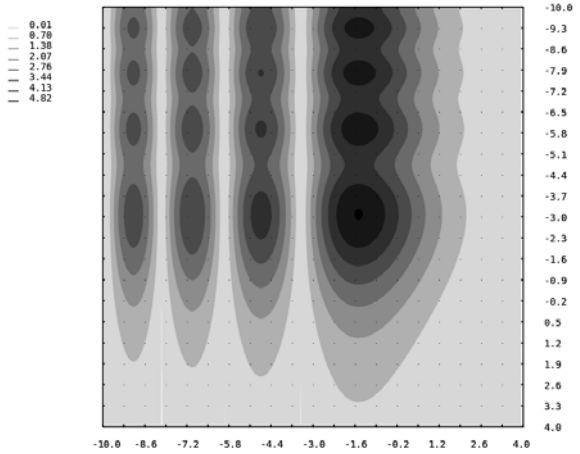

Рис. 1. Амплитуда краевой СВК F_4. Горизонтальная ось: $\lambda_1 = [-10, 4]$, вертикальная ось: $\lambda_2 = [-10, 4]$, $\lambda_3 = 0$

Как можно заметить, результат построений соответствует действительности (см. [11]).

Выйдем из плоскости $(\lambda_3 = 0)$, чтобы увидеть фокусировки первичных и вторичных полей:

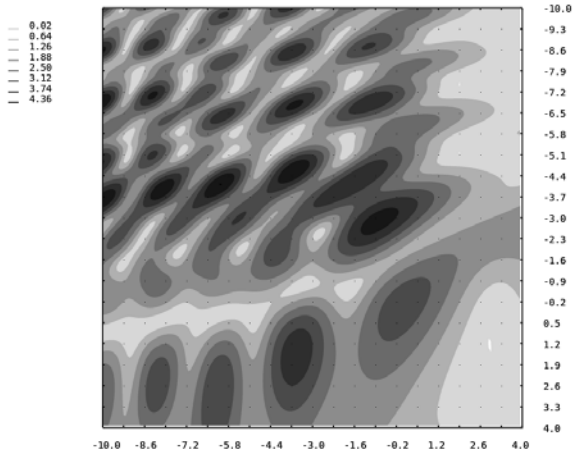

Рис. 2. Амплитуда краевой СВК F_4. Горизонтальная ось: $\lambda_1 = [-10, 4]$, вертикальная ось: $\lambda_2 = [-10, 4]$, $\lambda_3 = -3$

Полученное изображение соответствует случаю, когда $\lambda_3 = -3$.

Хотя, возможно, это имеет малую практическую ценность, Wavecat позволяет выполнить построения иных сечений, в том числе фиксируя значения параметров λ_1 и λ_2.

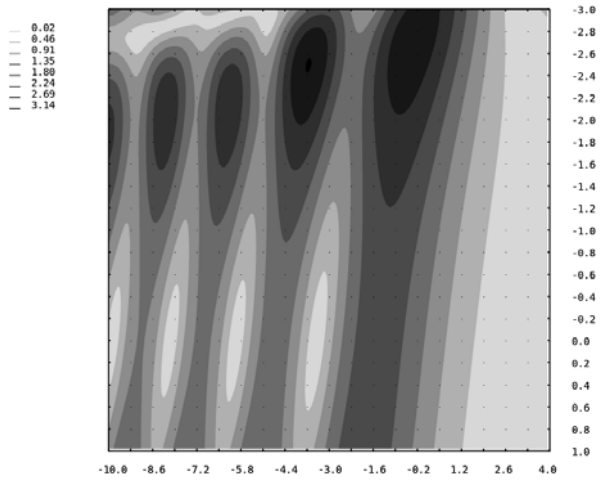

Рис. 3. Амплитуда краевой СВК F_4.
Горизонтальная ось: $\lambda_1 = [-10, 4]$, $\lambda_2 = 0$,
вертикальная ось: $\lambda_3 = [-3, 1]$

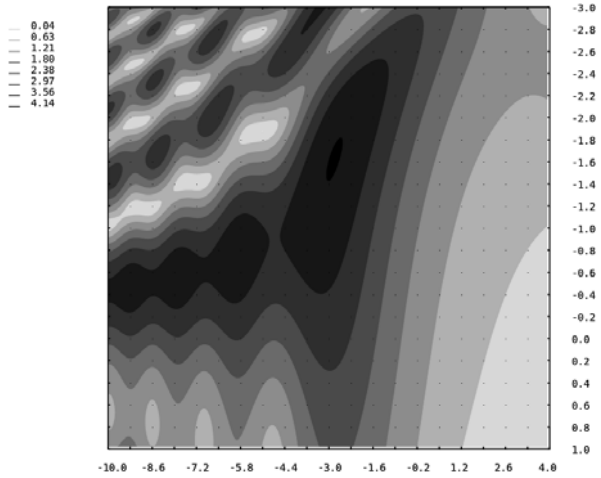

Рис. 4. Амплитуда краевой СВК F_4. $\lambda_1 = 0$,
горизонтальная ось: $\lambda_2 = [-10, 4]$,
вертикальная ось: $\lambda_3 = [-3, 1]$

Особенность $A_1A_2A_1A_1$

Получив ряд построений, перейдем к созданию программного модуля для вычисления СВК особенности $A_1A_2A_1A_1$.

Следующая СВК соответствует угловой особенности $A_1A_2A_1A_1$. Определение специальной функции представлено интегралом [11]:

$$V_{A_1A_2A_1A_1}(\lambda_1,\lambda_2,\lambda_3,a) = \int_0^{+\infty} dz \int_{-\infty}^{+\infty} \exp\Big(i\big(k_2y^3 + \\ + azy + k_1z^2 + \lambda_1 y + \lambda_2 y^2 + \lambda_3 z\big)\Big)dy. \quad (5)$$

Если в качестве начального вектора выбрать $\vec{S}^0 = (\lambda_1,\lambda_2,\lambda_3,0)$, система ОДУ будет иметь вид (6):

$$\frac{dV}{dt} = a^1\frac{\partial V}{\partial a}, \\ \frac{dV^1}{dt} = a^1\frac{\partial V^1}{\partial a}. \quad (6)$$

Определения частных производных в правых частях уравнений – следующие:

$$\frac{\partial V}{\partial a} = \frac{k_1}{2}\left(i\frac{\partial V_{B_3}(\lambda_1,\lambda_2)}{\partial\lambda_1} - \lambda_3V^1 + iaU_{11}\right), \\ \frac{\partial V^1}{\partial a} = \frac{k_1}{2}\left(i\frac{\partial^2 V_{B_3}(\lambda_1,\lambda_2)}{\partial\lambda_1^{\,2}} - \lambda_3U_{11} + iaU_{111}\right). \quad (7)$$

Для вычисления членов U_{11} и U_{111} воспользуемся следующими выражениями:

$$U_{11} = -\frac{k_2}{3}\Big(iV_{B_2}(\lambda_3) - \lambda_1V + 2i\lambda_2V^1 + iaU_{03}\Big), \\ U_{03} = i\frac{k_1}{2}\Big(iV_{B_3}(\lambda_1,\lambda_2) - \lambda_3V + iaV^1\Big), \quad (8) \\ U_{111} = -\frac{k_2}{2}\left(-V - \lambda_1V^1 + 2i\lambda_2U_{11} - a\frac{\partial V}{\partial a}\right).$$

Опустим определения для функций, соответствующих краевой особенности B_3 и интеграла Френеля.

Начальные значения в точке $\vec{S}^0 = (\lambda_1,\lambda_2,\lambda_3,0)$ выражаются через интеграл Френеля, СВК B_3 и одну из ее первых производных (9):

$$V(\vec{S}^0) = V_{B_2}(\lambda_3)V_{B_3}(\lambda_1,\lambda_2), \\ V^1(\vec{S}^0) = V_{B_2}(\lambda_3)\frac{\partial V_{B_3}(\lambda_1,\lambda_2)}{\partial\lambda_1}. \quad (9)$$

Поскольку для вычисления второй производной $\frac{\partial V^1}{\partial a}$ требуется знать вторую производную $\frac{\partial^2 V_{B_3}(\lambda_1,\lambda_2)}{\partial\lambda_1^{\,2}}$, воспользуемся выражением из системы ОДУ, определяющей СВК соответствующей особенности:

$$\frac{\partial^2 V_{B_3}(\lambda_1,\lambda_2)}{\partial\lambda_1^{\,2}} = \frac{k_2}{3}\Big(\lambda_1V - 2i\lambda_2V^1 - i\Big). \quad (10)$$

От математической записи перейдем к реа-

лизации программного модуля для системы Wavecat.

Определим необходимые перечисления, которые будут использоваться в дальнейшем в качестве индексов:

```
enum components {
   V = 0,
   V1
};

enum parameters {
   LAMBDA_1 = 0,
   LAMBDA_2,
   LAMBDA_3,
   ALPHA,
   K_1,
   K_2
};

enum storage_items {
   VB2L3 = 0, /* Bsub2(l3) */
   VB3L1L2, /* Bsub3(l1, l2) */
   DVB3L1L2, /* Bsub3'(l1, l2) */
   DDVB3L1L2 /* Bsub3''(l1, l2) */
};
```

Непосредственно процедура, используемая при решении системы ОДУ, может быть записана так:

```
void   cmplx_catastrophe_Asub1Asub2
       Asub1Asub1_function(const
       catastrophe_t        *const
       catastrophe,const    double
       t, const double complex *y,
       double complex *const f)
{
   double l1, l2, l3, a;
   double complex U11, U03, U111;
   double complex V0a, V1a;

   l1 = PARAM(LAMBDA_1);
   l2 = PARAM(LAMBDA_2);
   l3 = PARAM(LAMBDA_3);
   a = PARAM(ALPHA) * t; /* Check
   it! */

U03=(I*PARAM(K_1)/2.0)*(I*STORAGE_
       COMPLEX(VB3L1L2) - l3*y[V]
       + I*a*y[V1]);
U11 = (-PARAM(K_2)/3.0) *(I*STORAGE_
       COMPLEX(VB2L3) - l1*y[V] +
       2.0*I*l2*y[V1] + I*a*U03);

V0a = (PARAM(K_1)/2.0) *(I*STORAGE_
       COMPLEX(DVB3L1L2) - l3*y[V1]
       + I*a*U11);

U111 = (-PARAM(K_2)/2.0) *(-y[V]
       - l1*y[V1] + 2*I*l2*U11 -
       a*V0a);
V1a = (PARAM(K_1)/2.0) *(I*STORAGE_
       COMPLEX(DDVB3L1L2) - l3*U11
       + I*a*U111);

   f[V] = V0a*PARAM(ALPHA);
   f[V1] = V1a*PARAM(ALPHA);
}
```

Процедура, управляющая вычислениями, может быть выражена на ANSI C следующим образом:

```
static void calculate(catastrophe_t
       *const catastrophe, const
       unsigned int i, const
       unsigned int j)
{
   cmplx_equation_t *equation;
   point_array_t *point_array;

   double module;
   double phase;

   const double sqrt_pi = sqrt(M_
   PI);
   const double g13 = 2.678938534;
   const double g23 = 1.354117939;

   assert(catastrophe);

   equation = catastrophe->equation;
   point_array    =    catastrophe-
   >point_array;

   assert(equation);
   assert(point_array);

   /* Precalculate Bsub2(l3) */
   equation->initial_vector[V]    =
        0.5 * sqrt_pi * cexp(I *
        PARAM(K_1) * M_PI / 4.0);
   equation_set_function(equation,
   Bsub2_function);
  cmplx_runge_kutta(0.0, 1.0, 0.01,
  catastrophe);
   STORAGE_COMPLEX(VB2L3)         =
   equation->resulting_vector[V];

   /* Precalculate Bsub3 */
   equation->initial_vector[V]    =
        (1.0 / 3.0) * g13 * cexp(I
        * PARAM(K_2) * M_PI / 6.0);
   equation->initial_vector[V1]   =
        (I / 3.0) * g23 * cexp(I *
        PARAM(K_2) * M_PI / 3.0);
   equation_set_function(equation,
        Bsub3_function);
```

```
    cmplx_runge_kutta(0.0,      1.0,
        0.01, catastrophe);
    STORAGE_COMPLEX(VB3L1L2)        =
        equation->resulting_
        vector[V];
    STORAGE_COMPLEX(DVB3L1L2)       =
        equation->resulting_
        vector[V1];
    STORAGE_COMPLEX(DDVB3L1L2)
        =  (PARAM(K_2)  /  3.)  *
        (PARAM(LAMBDA_1) * equation-
        >resulting_vector[V]     -
        2  *  I  *  PARAM(LAMBDA_2)
        *     equation->resulting_
        vector[V1] - I);
    equation->initial_vector[V]     =
        STORAGE_COMPLEX(VB2L3)    *
        STORAGE_COMPLEX(VB3L1L2);
    equation->initial_vector[V1]    =
        STORAGE_COMPLEX(VB2L3)    *
        STORAGE_COMPLEX(DVB3L1L2);
    equation_set_function(equation,
        cmplx_catastrophe_
        Asub1Asub2Asub1Asub1_
        function);
    cmplx_runge_kutta(0.0,      1.0,
        0.01, catastrophe);
    module      =      cabs(equation-
        >resulting_vector[V]);
    phase   =   (180.0   /   M_PI)   *
        carg(equation->resulting_
        vector[V]);
    point_array->array[i][j].module
        = module;
    point_array->array[i][j].phase =
        phase;
}
```

Для того чтобы модуль стал доступен системе, его требуется описать специальной структурой данных:

```
static catastrophe_desc_t
    cmplx_catastrophe_
    Asub1Asub2Asub1Asub1_desc = {
    .type = CT_COMPLEX,
    .sym_name                       =
    «Asub1Asub2Asub1Asub1»,
    .fabric = catastrophe_fabric,
    .num_parameters = 6,
    .par_names = par_names,
    .equation.cmplx                 =
    cmplx_catastrophe_
    Asub1Asub2Asub1Asub1_function,
    .num_equations = 2,
    .calculate = calculate
};
```

Выполним несколько построений с использованием нового модуля.

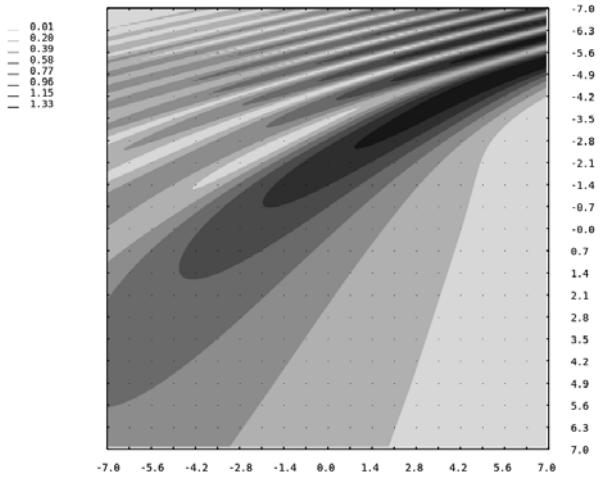

Рис. 5. Угловая СВК $A_1A_2A_1A_1$.
Горизонтальная ось: $\lambda_1 = [-7, 7]$,
вертикальная ось: $\lambda_2 = [-7, 7]$, $\lambda_3 = 0$,
$a = 1$, $k_1 = k_2 = 1$

График полученной особенности похож на результат, ранее опубликованный для СВК B_3. Предположительно, выведение особенности из гиперплоскости, определяемой парой $\lambda_3 = 0$, $a = 1$, приведет к небольшому изменению графика, поскольку перестанут сливаться вторичные поля краевых и угловых лучевых семейств.

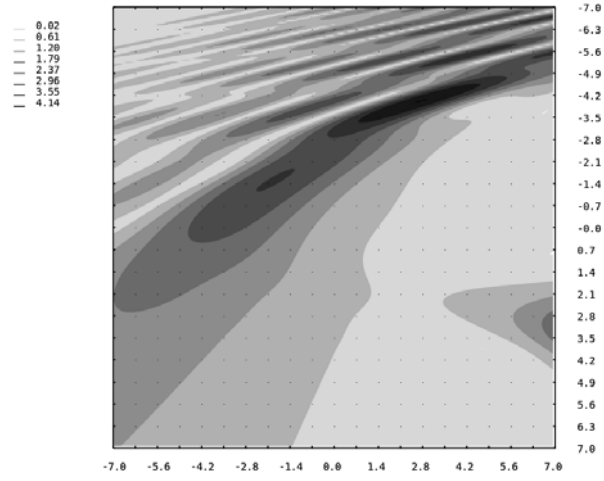

Рис. 6. Угловая СВК $A_1A_2A_1A_1$.
Горизонтальная ось: $\lambda_1 = [-7, 7]$,
вертикальная ось: $\lambda_2 = [-7, 7]$, $\lambda_3 = -3$,
$a = 2$, $k_1 = k_2 = 1$

Важное отличие от разобранной СВК F_4 – необходимость вычисления второй производной СВК B_3, что выполняется следующим образом:

```
STORAGE_COMPLEX(DDVB3L1L2)
     = (PARAM(K_2) / 3.) *
     (PARAM(LAMBDA_1) * equation-
     >resulting_vector[V] - 2 * I
     * ARAM(LAMBDA_2) * equation-
     >resulting_vector[V1] - I);
```

Необходимость выполнения этого кода в управляющей процедуре выглядит неуместно, но поскольку функции, ответственные за представление ОДУ для процедуры численного интегрирования, не сохраняют результатов для вторых производных, такие вычисления приходится повторять дополнительно.

Как и в случае с краевой СВК F_4, можно заметить, что в программном модуле приходилось применять хитрость, чтобы уложить задачу расчета нескольких СВК в один объект задания – вычислительное ядро изначально проектировалось с учетом СВК основных особенностей. Такой обходной путь не только некрасив, но и малоэффективен, поскольку, таким образом, не используется возможность кеширования (сохранения для будущего использования) результатов, прозрачно предоставляемая вычислительным ядром каждому модулю.

Заключение

С учетом вышеизложенного становится необходимым преобразовать изначальную архитектуру, чтобы учитывать необходимость вычисления различных СВК и их производных в одной и той же точке. Один из вариантов решения – заведение массива объектов типа catastrophe_t для каждого задания. При этом всегда есть главный объект катастрофы, соответствующей, собственно, СВК, которую требуется вычислить. Другие же, подчиненные, объекты используются для получения промежуточных значений. Взаимосвязь объектов может задаваться в дескрипторе особенности, как и связь СВК с параметрами. Прототип подобного решения уже опробован, и дальнейшая работа будет связана с его стабилизацией и добавлением в общую кодовую базу.

За счет подобного представления можно добиться высокой степени переиспользования кода модулей и корневой части вычислительного ядра (параллелизм и кеширование).

Для решения проблемы с получением значений производных высоких порядков достаточно сохранять их наравне со значениями фундаментального вектора в качестве результата решения систем ОДУ.

В статье были представлены новые сечения особенностей F_4 и $A_1A_2A_1A_1$, в том числе при фиксированном первом параметре, в процессе подготовки иллюстраций для публикации были обнаружены ошибки в подпрограмме построения графиков линий равного уровня, которые были устранены.

Исходный код программного комплекса, за исключением языковых средств (временно), опубликован в проекте github по адресу: https://github.com/richiefreedom/wavecat. Последние изменения могут отсутствовать в репозитарии на момент публикации, но в скором времени они будут также загружены.

Литература

1. Дорохина Т.В., Крюковский А.С., Лукин Д.С. Информационная система «Волновые катастрофы в радиофизике, акустике и квантовой механике» // Электромагнитные волны и электронные системы. – 2007. – Т. 12. – № 8. – С. 71–75.

2. Дорохина Т.В., Крюковский А.С., Малышенко А.Б. Разработка численных алгоритмов расчета и визуализации волновых катастроф // Вестник Российского нового университета. – 2008. – Выпуск 3. Управление, вычислительная техника и информатика. – С. 25–48.

3. Дорохина Т.В., Крюковский А.С., Лукин Д.С., Палкин Е.А., Растягаев Д.В. Математическое моделирование волновых полей типа катастроф с помощью специализированной информационно-справочной системы // Труды Российского научно-технического общества радиотехники, электроники и связи им. А.С. Попова. Серия: научная сессия, посвященная Дню радио, 17–18.05.2006 г. – Выпуск LXI. – 2006. – С. 287–289.

4. Дорохина Т.В., Крюковский А.С., Лукин Д.С. Принципы проектирования и построения информационно-справочной системы волновой теории катастроф // Цивилизация знаний: российские реалии : труды Восьмой Всероссийской научной конференции. Москва, 20–21 апреля 2007. – М. : РосНОУ, 2007. – Ч. I. – С. 158–164.

5. Крюковский А.С., Рогачев С.В. Система расчета и визуализации специальных функций волновых катастроф // Электромагнитные волны и электронные системы. – 2013. – Т. 18. – № 8. – С. 010–017.

6. Рогачев С.В., Крюковский А.С. Специализированная система расчёта и визуализации

специальных функций волновых катастроф // Цивилизация знаний: проблемы и смыслы образования : труды Четырнадцатой Международной научной конференции. – М. : РосНОУ, 2013. – С. 171–174.

7. Растягаев Д.В., Рогачев С.В. Специализированная система моделирования специальных функций волновых катастроф // Цивилизация знаний: российские реалии : труды Пятнадцатой Международной научной конференции : в 2 частях. – 2014. – С. 439–444.

8. Крюковский А.С., Рогачев С.В. Архитектура программного комплекса расчёта специальных функций волновых катастроф // Вестник Российского нового университета. – 2014. – Выпуск 4. Управление, вычислительная техника и информатика. – С. 13–20.

9. Крюковский А.С., Рогачев С.В. Разработка специализированной системы расчета функций волновых катастроф // Сверхширокополосные сигналы в радиолокации, связи и акустике : материалы IV Всероссийской научной конференции. Муромский институт (филиал) Федерального государственного бюджетного образовательного учреждения высшего профессионального образования «Владимирский государственный университет имени Александра Григорьевича и Николая Григорьевича Столетовых». – Муром, 2013. – С. 139–144.

10. Крюковский А.С., Рогачев С.В. Вычисление специальных функций волновых катастроф в среде численного моделирования “WaveCat” // T-Comm: Телекоммуникации и транспорт. – 2015. – Т. 9. – № 4. – С. 54–59.

11. Крюковский А.С. Равномерная асимптотическая теория краевых и угловых волновых катастроф : монография. – М. : РосНОУ, 2013. – 368 с.

12. Крюковский А.С., Лукин Д.С. Теория расчета эталонных фокальных и дифракционных электромагнитных полей на основе специальных функций волновых катастроф // Радиотехника и электроника. – 2003. – Т. 48. – № 8. – С. 912–921.

13. Крюковский А.С. Метод обыкновенных дифференциальных уравнений для расчета специальных функций волновых катастроф (СВК) // Дифракция и распространение электромагнитных и акустических волн. – 1992. – С. 29–48.

14. Рогачев С.В. Проектирование предметно ориентированного языка для расчета специальных функций волновых катастроф // Вестник Российского нового университета. – 2013. – Выпуск 4. Управление, вычислительная техника и информатика. – С. 40–46.

15. Рогачев С.В. Параллельный расчет специальных функций волновых катастроф // Сверхширокополосные сигналы в радиолокации, связи и акустике : материалы V Всероссийской научной конференции. – 2015. – С. 77–86.