УДК 519.85, 517.977.5

Прикладное применение интервального метода взрывов

Пановский В.Н.

Московский авиационный институт (национальный исследовательский университет), Волоколамское шоссе, 4, Москва, А-80, ГСП-3, 125993, Россия

e-mail: panovskiy.v@gmail.com

Аннотация

Представлено алгоритмическое и программное обеспечение разработанного интервального метода взрывов и его применение к задаче синтеза оптимального программного управления детерминированными динамическими системами. Решен ряд прикладных задач (ориентация и стабилизация спутника, задача о «мягкой посадке»), демонстрирующих применимость интервального метода взрывов.

Ключевые слова: интервальный анализ, глобальная оптимизация, метод взрывов, эвристический алгоритм, оптимальное управление, динамическая система.

Введение

В современной математике достаточно большое внимание уделяется решению задач глобальной оптимизации и синтеза оптимального управления динамическими системами. Эти задачи являются востребованными в ходе проектирования конструкций самолетов, вертолетов, космических аппаратов, когда возникает

необходимость в оптимизации характерных параметров и разработке систем управления как отдельными элементами конструкции, так и объектом в целом.

Существующие численные методы используют разнообразные подходы, но их использование связано с большими вычислительными затратами, излишними требованиями к постановке задачи, проблемами в достижении сходимости метода. Таким образом, разработка новых методов оптимизации, сочетающих в себе современные математические подходы, является актуальной.

Существующие интервальные методы поиска глобального минимума функции можно разделить на две группы в зависимости от решаемой задачи: методы условной и безусловной оптимизации.

К методам безусловной оптимизации можно отнести следующие методы:

- Интервальный адаптивный алгоритм или алгоритм Мура-Скелбоу [1]. Алгоритм оперирует с рабочим списком, в котором хранятся интервальные векторы, получающиеся в результате дробления исходного интервального вектора на более мелкие. Кроме этих параллелотопов в списке хранятся и нижние оценки областей значений целевой функции на них. На каждом шаге алгоритма из списка извлекается параллелотоп с наименьшей оценкой значения функции, производится его дробление, оценка получившихся интервальных векторов и занесение результатов обратно в список.
- Алгоритм Ичиды-Фуджи [1]. По сути является модификацией алгоритма Мура-Скелбоу, добавляя в него стадию модификации рабочего списка.

- Метод Дюсселя [2]. Основная идея данного метода заключается в следующем: избегать использования списков (взамен предлагается использовать рекурсивные процедуры).
- Метод Хансена [1], который имеет вариации, использующие градиент функции, понятие выпуклости и использование пробных точек. Оригинальный метод, так же как и методы Мура-Скелбоу и Ичиды-Фуджи, работает со списком, в котором хранятся интервальные векторы и оценка значения функции на них.
- Интервальный метод Ньютона [3], применяемый для решения систем уравнений, следующих из необходимых условий экстремума.
- Интервальный алгоритм «имитации отжига», разработанный С.П. Шарым [4]. В основе метода лежит классический (точечный) метод, который моделирует физические процессы отжига или кристаллизации.
- Метод случайного интервального дробления, разработанный С.П. Шарым [4]. Данный метод можно назвать модификацией интервального адаптивного алгоритма. Главное отличие в том, что интервальный вектор выбирается из рабочего списка случайным образом.
- Метод дробления графика, разработанный С.П. Шарым [5]. Данный метод тоже похож на интервальный адаптивный алгоритм, а основное новшество заключается в том, что метод ищет оценку минимального значения функции $f(x)$ за счет анализа совместности уравнения $f(x)-l=0$, где l – некоторое значение,

которое функция может принимать. С помощью такого анализа можно определить, больше или меньше, чем l, значение минимума функции.

- Интервальный эволюционный алгоритм, разработанный Н.В. Пановым и С.П. Шарым [6]. Данный метод похож на интервальный адаптивный алгоритм, модифицированный некоторыми понятиями эволюционных алгоритмов (в частности оператора мутации).

К методам условной оптимизации можно отнести следующие интервальные методы:

- Метод Хансена [3], который использует условия Фрица-Джона (необходимые условия экстремума) и численные методы решения систем. Использование данных условий позволяет оптимизировать целевую функцию при ограничениях обоих типов (и равенства, и неравенства).
- Метод Мура [7], который использует необходимые условия Куна-Таккера-Каруша. Использование данных условий позволяет оптимизировать целевую функцию при ограничениях типа неравенств.

Перечисленные методы имеют ряд недостатков, связанных прежде всего с требованием дифференцируемости и выпуклости целевой функции и функций, описывающих ограничения. Компьютерная реализация данных процедур является достаточно сложной и дополнительно увеличивает вычислительные затраты метода. Кроме того, большинство методов так или иначе оперируют с рабочим списком. Ведение и обработка этого списка также увеличивает вычислительную сложность

алгоритма, так как требуется проводить много операций сравнения, чтобы не нарушить упорядоченность элементов.

Интервальные алгоритмы были эффективно использованы для решения задач синтеза оптимального управления динамическими системами [8, 9].

Кроме этого, следует отметить, что крайне важно использовать и разрабатывать эвристические методы. Несмотря на отсутствие строгого обоснования, эти методы способы найти приемлемое решение задачи. Эвристические алгоритмы имеют следующие особенности: они не гарантируют нахождение лучшего решения; они не гарантируют нахождение решения, даже если оно существует («проскакивание решения»); они могут дать неверное решение в некоторых случаях. Однако существенным достоинством таких алгоритмов является их низкая вычислительная сложность, что позволяет их применять для решения задач повышенной трудности (например, задачи, принадлежащие классу NP-трудных). В совокупности с ключевыми особенностями интервального анализа (обработка множеств значений вместо отдельных точек, низкая требовательность к постановке задачи) разработка эвристических интервальных алгоритмов является перспективным направлением.

На данный момент существуют методы оптимизации, которые, так или иначе, используют эвристики, основывающиеся на проведении взрывов. Примером этому могут быть методы фейерверков и гранат (Fireworks Algorithm и Grenade Explosion Method) [10, 11]. В данной статье предлагается интервальный вариант методов

этого типа, содержащий новый набор правил и процедур детонации, с помощью которых проводится поиск.

Интервальный анализ. Основные понятия

Основной идеей интервального анализа является окружение вещественных чисел интервалами, а вещественных векторов – интервальными векторами, или параллелотопами [12-14]. Условимся в дальнейшем для обозначения интервала использовать строчные латинские буквы, заключенные в квадратные скобки ($[a],[b],[c],\ldots$) или, привычным представлением ($[a_l,a_r],[b_l,b_r],[c_l,c_r],\ldots$), для параллелотопа – то же обозначение, только начертание букв полужирное ($[\mathbf{a}],[\mathbf{b}],[\mathbf{c}],\ldots$) или, как прямое произведение интервалов ($[1;3]\times[2;3]\times\ldots\times[10;11]$).

Для произвольного интервала $[x]$ определены [13]: нижняя граница $\underline{[x]}=\sup\{\xi\in\Re\cup\{-\infty,\infty\}\mid\forall\zeta\in[x],\xi\le\zeta\}$, верхняя граница $\overline{[x]}=\inf\{\xi\in\Re\cup\{-\infty,\infty\}\mid\forall\zeta\in[x],\zeta\le\xi\}$, ширина (определена только для непустого интервала) $\omega([x])=\overline{[x]}-\underline{[x]}$, средняя точка (определена только для ограниченного и непустого интервала): $mid([x])=\frac{\underline{[x]}+\overline{[x]}}{2}$.

Те же параметры определены и для параллелотопов. Нижняя и верхняя границы и средняя точка становятся векторами, ширина же рассчитывается как максимум из ширин всех компонентов.

Интервальной оболочкой $[\mathrm{X}]$ множества $\mathrm{X}\subset\Re^n$ называется параллелотоп с наименьшей шириной, который содержит X, т.е., если множество берется в

квадратные скобки, это значит, что рассматривается интервальная оболочка этого множества.

Пусть $\circ$ - некоторая бинарная операция, тогда $[x]\circ[y]=[\{\xi_1\circ\xi_2 \mid \xi_1\in[x],\xi_2\in[y]\}]$; пусть f – некоторый унарный оператор, тогда $f([x])=[\{f(\xi)\mid\xi\in[x]\}]$.

Множество интервалов обозначается как $\mathrm{I}\Re$, интервальных векторов – как $\mathrm{I}\Re^n$. Пусть имеется некоторая функция f, действующая из $\Re^n$ в $\Re^m$. Функция $[f](\cdot)$ называется интервальной функцией включения [13] для f, действующей из $\mathrm{I}\Re^n$ в $\mathrm{I}\Re^m$, если $f([\mathbf{x}])=\{f(\xi)\mid\xi\in[\mathbf{x}]\}\subset[f]([\mathbf{x}]),\forall[\mathbf{x}]\in\mathrm{I}\Re^n$. Функция включения позволяет получить априорную оценку множества значений функции на параллелотопе. Оценкой прямого образа функции на некотором параллелотопе будем называть множество значений функции включения на этом параллелотопе.

Постановка задачи интервальной ε - минимизации

Пусть имеется целевая функция $f:\Re^n\to\Re$ и область поиска $[\mathbf{s}]$. Задача интервальной ε - минимизации формулируется следующим образом: найти такой параллелотоп $[\mathbf{p}]^*$, что

$$[\mathbf{p}]^*\subseteq[\mathbf{s}],\omega\left([\mathbf{p}]^*\right)\le\varepsilon,\forall[\mathbf{x}]\subseteq[\mathbf{s}],\omega([\mathbf{x}])\ge\varepsilon:\underline{[f]\left([\mathbf{p}]^*\right)}\le\overline{[f]([\mathbf{x}])}. \qquad (1)$$

Стратегия интервального метода взрывов

На первом этапе метода случайным образом на области поиска устанавливаются бомбы, каждая из которых ассоциируется с некоторым параллелотопом, описывающим ее местоположение. Кроме этого, бомбы сортируются по возрастанию нижней грани оценки прямого образа функции. В

алгоритме есть две итеративные части: глобального поиска и уточняющего, которые отличаются процедурой взрыва. На итеративных частях алгоритма происходит расчет мощностей бомб; реализация взрыва, в ходе которого каждая из бомб образует осколки, которые разлетаются в разные стороны; обновление списка бомб. Алгоритм заканчивает работу, когда превышено максимальное количество итераций. Из списка бомб выбирается та, которой соответствует наименьшая нижняя грань оценки прямого образа целевой функции.

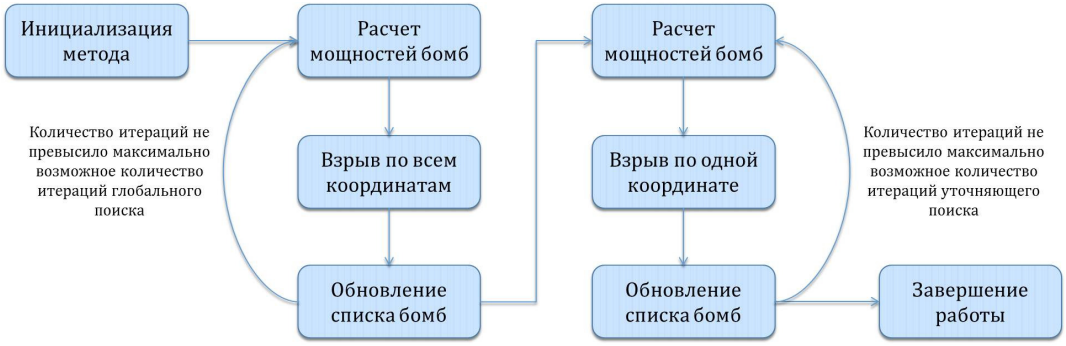

Рис. 1. Схема интервального метода взрывов

Пошаговое описание алгоритма

Шаг 1. Задать номер итерации алгоритма $t^g = 0$, максимальное количество итераций глобального и уточняющего поиска $t^g_{\max}$ и $t^c_{\max}$, максимальное количество бомб $B_{\max}$, вектор максимальных мощностей $r_{\max} \in \mathbb{R}^n$ и $[\mathbf{s}]$ – область поиска.

Шаг 2. Случайным образом создать множество бомб - параллелотопов $\{[\mathbf{b}]_i\}_{i=1}^{B_{\max}}$ на области поиска, где $[\mathbf{b}]_i = [\xi_1;\zeta_1]_i \times ... \times [\xi_n;\zeta_n]_i$, где ξ_j и ζ_j – равномерно распределенные на интервале $[s_j]$ случайные величины (если $\xi_j > \zeta_j$, то соответствующий интервал заменяется на интервал $[\zeta_j;\xi_j]$). Отсортировать по возрастанию нижней грани оценки прямого образа функции.

Шаг 3. Расчет мощностей бомб. Для каждой бомбы $[\boldsymbol{b}]_i$ рассчитать соответствующую мощность: $p^i = \dfrac{i-1}{B_{\max}-1} \cdot r_{\max}$, $i = 1,...,B_{\max}$.

Шаг 4. Взрыв (глобальный поиск). Во время этого этапа происходит образование двух осколков (операция бисекции параллелотопа – разделение вдоль компоненты, обладающей наибольшей шириной). Таким образом получаются две новых бомбы: $[\mathbf{b}]_i = \begin{pmatrix} \begin{pmatrix} [b_1]_i \\ \vdots \\ [b_k]_i \\ \vdots \\ [b_n]_i \end{pmatrix} + \begin{pmatrix} \xi\left(-p_1^i; p_1^i\right) \\ \vdots \\ \xi\left(0; p_k^i\right) \\ \vdots \\ \xi\left(-p_n^i; p_n^i\right) \end{pmatrix} \end{pmatrix} \cap [\mathbf{s}]$ и $[\mathbf{b}]_{B_{\max}+i} = \begin{pmatrix} \begin{pmatrix} [b_1]_i \\ \vdots \\ [b_k]_i \\ \vdots \\ [b_n]_i \end{pmatrix} + \begin{pmatrix} \xi\left(-p_1^i; p_1^i\right) \\ \vdots \\ \xi\left(-p_k^i; 0\right) \\ \vdots \\ \xi\left(-p_n^i; p_n^i\right) \end{pmatrix} \end{pmatrix} \cap [\mathbf{s}]$, где $\xi(a;b)$ – случайная величина, равномерно распределенная на интервале $[a;b]$.

Шаг 5. Обновление списка бомб. Так как на предыдущим этапе количество бомб удвоилось, лишние необходимо удалить. Для этого производится сортировка бомб по возрастанию нижней грани оценки прямого образа целевой функции. Далее просто удаляются последние $B_{\max}$ бомб.

Шаг 6. Увеличить t^g на единицу. Если $t^g < t^g_{\max}$, то перейти к шагу 3. В противном случае – к шагу 7.

Шаг 7. Положить $t^c = 0$.

IШаг 8. Расчет мощностей бомб. Для каждой бомбы $[\mathbf{b}]_i$ из списка $\left\{[\mathbf{b}]_i\right\}_{i=1}^{B_{\max}}$ рассчитать соответствующую мощность: $p^i = \dfrac{i-1}{B_{\max}-1} \cdot r_{\max}$, $i = 1,...,B_{\max}$.

Шаг 9. Взрыв (уточняющий поиск). В ходе данного этапа получаются две новые бомбы: $[\mathbf{b}]_i = \begin{pmatrix} \begin{pmatrix} [b_1]_i \\ \vdots \\ [b_k]_i \\ \vdots \\ [b_n]_i \end{pmatrix} + \begin{pmatrix} 0 \\ \vdots \\ \xi\left(0; p_k^i\right) \\ \vdots \\ 0 \end{pmatrix} \end{pmatrix} \cap [\mathbf{s}]$ и $[\mathbf{b}]_{B_{\max}+i} = \begin{pmatrix} \begin{pmatrix} [b_1]_i \\ \vdots \\ [b_k]_i \\ \vdots \\ [b_n]_i \end{pmatrix} + \begin{pmatrix} 0 \\ \vdots \\ \xi\left(-p_k^i; 0\right) \\ \vdots \\ 0 \end{pmatrix} \end{pmatrix} \cap [\mathbf{s}]$.

Шаг 10. Обновление списка бомб. Сортировка бомб по возрастанию нижней грани оценки прямого образа целевой функции. Удаляются последние $B_{\max}$ бомб.

Шаг 11. Увеличить t^c на единицу. Если $t^c < t^c_{\max}$, то перейти к шагу 8. В противном случае – к шагу 12.

Шаг 12. Выбрать из всех бомб ту, которой соответствует наименьшее значение нижней грани оценки прямого образа целевой функции.

Алгоритм нахождения оптимального программного управления дискретными детерминированными системами

Поведение модели объекта управления описывается разностным уравнением

$$x(t+1) = f(t, x(t), u(t)), \tag{2}$$

где t - дискретное время $t \in T = [0,1,..., N-1]$, число шагов N - задано, $x \in \Re^n$ - вектор состояния системы, $u \in [\mathbf{u}] \subset \Re^q$ - вектор управления, $[\mathbf{u}]$ - множество значений управления, представляющее собой параллелотоп, $f(t,x,u) = (f_1(t,x,u),..., f_n(t,x,u))^T$ - непрерывная вектор-функция. Начальное состояние системы (2) задано

$$x(0) = x_0 . \tag{3}$$

Конечное состояние $x(N)$ должно удовлетворять условиям:

$$\Gamma_i(x(N)) = 0, i = 1,..., l , \tag{4}$$

где $0 \le l \le n$, функции $\Gamma_i(x)$ - непрерывно дифференцируемы; система векторов $\left\{ \partial\Gamma_i(x(N)) \big/ \partial x_1 ,..., \partial\Gamma_i(x(N)) \big/ \partial x_n \right\}$, $i = 1,..., l$ линейно независима $\forall x(N) \in \Re^n$.

При управлении используется информация только о дискретном времени t.

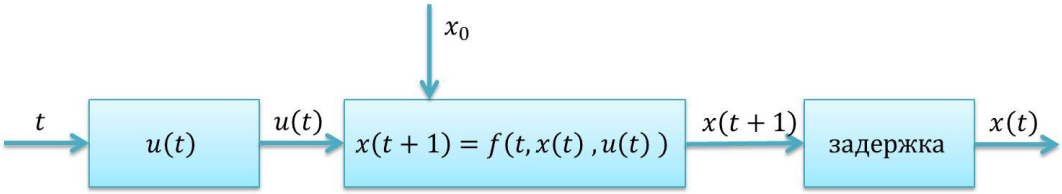

Рис. 2. Программное управление дискретной системой

Множество допустимых процессов $D(0, x_0)$ определяется как множество пар $d = (x(\cdot), u(\cdot))$, включающих траекторию $x(\cdot) = \{x_0, x(1), ..., x(N)\}$ и управление $u(\cdot) = \{u_0, u(1), ..., u(N-1)\}$, где $u(t) \in [\mathbf{u}], \forall t \in T$, удовлетворяющих (2), (3) и (4).

На множестве $D(0, x_0)$ определен функционал качества управления

$$I(d) = F(x(0), ..., x(N), u(0), ..., u(N-1)), \tag{5}$$

где $F(x(0), ..., x(N), u(0), ..., u(N-1))$ - заданная непрерывная функция.

Основная идея поиска оптимального управления заключается в преобразовании функционала качества управления (5). Для этого по разностному уравнению (2) строится система равенств:

$$\begin{cases} x(0) = x_0, \\ x(1) = f(0, x(0), u(0)), \\ \dots \\ x(N) = f(N-1, x(N-1), u(N-1)). \end{cases} \tag{6}$$

В полученной системе известны все величины, кроме $u(0), ..., u(N-1)$. Подставляя правые части (6) в (5), получаем преобразованный функционал качества, зависящий только от управлений:

$$\bar{I}(u_1(0), ..., u_q(0), ..., u_1(N-1), ..., u_q(N-1)) = F(u_1(0), ..., u_q(0), ..., u_1(N-1), ..., u_q(N-1)). \tag{7}$$

Кроме этого, добавим в полученный функционал (7) слагаемое, отвечающее за выполнение конечного условия (4):

$$\overline{\overline{I}}\left(u_1(0),...,u_q(0),...,u_1(N-1),...,u_q(N-1)\right)=F\left(u_1(0),...,u_q(0),...,u_1(N-1),...,u_q(N-1)\right)+$$

$$+\sum_{i=1}^{l}R_i\cdot h_\infty^0\left(\left[\Gamma_i\left(x(N)\right);\Gamma_i\left(x(N)\right)\right],[-\xi_i;\xi_i]\right) \quad (8)$$

где $h_\infty^0\left([a],[b_i]\right)=\inf\left\{r\,|\,[a]\subseteq[b]+[r]\cdot[-1;1],[r]=[r;r],r\geq 0\right\}$ - мера близости интервалов, R_i - величины штрафов, ξ_i - величины погрешности удовлетворения конечного условия, которые задаются пользователем.

Замечание: под параллелотопом $[\mathbf{u}]$ подразумевается интервальный вектор

$$\underbrace{\left[u_1(0)\right]\times...\times\left[u_q(0)\right]}_{[u(0)]}\times\underbrace{\left[u_1(1)\right]\times...\times\left[u_q(1)\right]}_{[u(1)]}\times...\times\underbrace{\left[u_1(N-1)\right]\times...\times\left[u_q(N-1)\right]}_{[u(N-1)]}.$$

Тогда искомую задачу можно сформулировать следующим образом: найти параллелотоп $[\mathbf{u}]^*$, являющийся решением задачи интервальной ε - минимизации функционала (8).

Алгоритм нахождения оптимального программного управления непрерывными детерминированными системами

Поведение модели объекта управления описывается дифференциальным уравнением

$$\dot{x}(t)=f\left(t,x(t),u(t)\right), \quad (9)$$

где t - непрерывное время $t\in T=[t_0,[t_{1l},t_{1r}]]$, - промежуток времени функционирования системы (интервал с подвижным правым концом), $x\in\Re^n$ - вектор состояния

системы, $u \in [\mathbf{u}] \subset \Re^q$ - вектор управления, $[\mathbf{u}]$ - множество значений управления, представляющее собой параллелотоп, $f(t,x,u) = (f_1(t,x,u),...,f_n(t,x,u))^T$ - непрерывно-дифференцируемая вектор-функция. Начальное состояние системы (9) задано

$$x(t_0) = x_0 . \tag{10}$$

Конечное состояние $x(t_1)$ и момент окончания функционирования системы t_1 должны удовлетворять условиям:

$$\Gamma_i(t_1, x(t_1)) = 0, i = 1,...,l , \tag{11}$$

где $0 \le l \le n+1$, функции $\Gamma_i(t_1, x(t_1))$ - непрерывно дифференцируемы; система векторов $\left\{ \partial\Gamma_i(t_1,x) / \partial x_1 ,..., \partial\Gamma_i(t_1,x) / \partial x_n \right\}$, $i = 1,...,l$ линейно независима $\forall (t_1, x(t_1)) \in [t_{1l}, t_{1r}] \times \Re^n$.

При управлении используется информация только о непрерывном времени t (применяется программное управление).

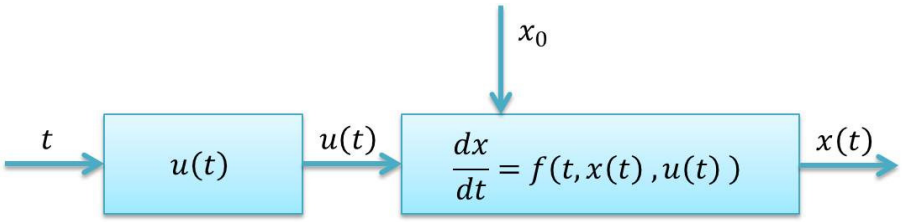

Рис. 3. Программное управление непрерывной системой

Множество допустимых процессов $D(t_0, x_0)$ определяется как множество троек $d = (t_1, x(\cdot), u(\cdot))$, включающих момент окончания функционирования системы t_1 траекторию $x(\cdot)$ и управление $u(\cdot)$, где $u(t) \in [\mathbf{u}], \forall t \in T$, удовлетворяющих (9), (10) и (11).

На множестве допустимых процессов определен функционал качества управления

$$I(d)=\int_{t_0}^{t_1} f^0\left(t,x(t),u(t)\right)dt+F\left(t_1,x(t_1)\right), \qquad (12)$$

где $f^0(t,x,u)$ и $F(t_1,x)$ - заданные непрерывные функции.

Требуется найти такую траекторию $\left(t_1^*,x^*(\cdot),u^*(\cdot)\right)$, которая бы минимизировала значение функционала (12).

Предлагается искать управление в виде кусочно-постоянной интервальной функции.

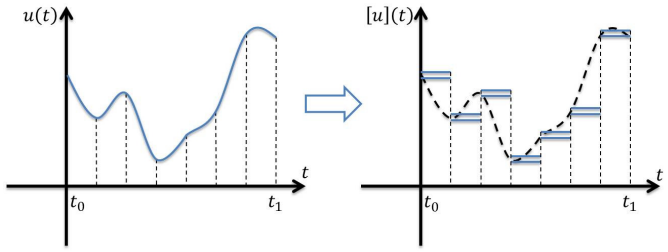

Рис. 4. Замена непрерывного управления кусочно-постоянным интервальным

Такая форма управления позволяет генерировать траектории, соответствующие управлениям, в виде интервалов. Поиск функции и момента окончания функционирования системы, минимизирующих функционал (12), предлагается проводить в несколько этапов: построение разностных уравнений по системе (9) с помощью формул Эйлера, Эйлера – Коши и Рунге – Кутты 3-го и 4-го порядков и численного интегрирования критерия (12), поиск оптимального управления полученной системы, восстановление полученного управления до кусочно-постоянного.

Замечание: управлению, представляемому функцией $[u](t)$, и моменту окончания функционирования системы можно однозначно сопоставить интервальный вектор $\underbrace{[u_1](\tau_0)\times...\times[u_q](\tau_0)}_{[u](\tau_0)} \times \underbrace{[u_1](\tau_1)\times...\times[u_q](\tau_1)}_{[u](\tau_1)} \times...\times$

$\times\underbrace{[u_1](\tau_{N-1})\times...\times[u_q](\tau_{N-1})}_{[u](\tau_{N-1})}\times[t_1]$, где $\tau_i = t_0 + \frac{t_1 - t_0}{N}\cdot i$, N - количество шагов дискретизации.

Программное обеспечение

На основе алгоритма автором разработано программное обеспечение для поиска глобального минимума функций. Среда разработки – Microsoft Visual Studio, язык программирования – C#.

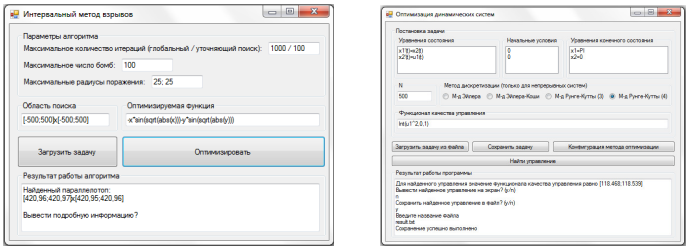

Рис. 5. Главные окна программ

С помощью данного программного обеспечения можно не только решать описанные ранее задачи, но и получать некоторую статистическую информацию, которая собирается по мере работы алгоритма (среднее расстояние, на которое разлетались осколки, траектории движения бомб и т.п.).

Решение прикладных задач

Задача об ориентации космического аппарата. Будем рассматривать плоский вариант задачи ориентации космического аппарата (КА): поворот КА производится лишь в одной плоскости за счет вращения маховика, установленного внутри КА. На

вал маховика подается вращательный момент, который сообщает ему угловое ускорение $\ddot{\theta}$. Таким образом, $J_M\ddot{\theta} = M$, где J_M - момент инерции маховика. Согласно закону сохранения кинетического момента КА: $\frac{d}{dt}\left(J_M\dot{\theta} + J_0\dot{\varphi}\right)$, где J_0 - момент инерции КА, φ - угол ориентации. Тогда из последнего равенства следует, что $J_M\ddot{\theta} = -J_0\ddot{\varphi} = M$. Величину, пропорциональную вращательному моменту M, взятую с обратным знаком, примем за сигнал управления $u = -\frac{M}{J_0}$.

Введем обозначения для составляющих вектора состояния: $x_1(t) = \varphi$, $x_2(t) = \dot{\varphi}$, тогда $\dot{x}_1(t) = x_2(t)$, $\dot{x}_2(t) = u(t)$.

Функционал качества управления: $I = \int_{t_0}^{t_1} u^2(t)dt$.

В начальный момент времени $t_0 = 0$: $x_1(t_0) = x_2(t_0) = 0$, в конечный момент времени $t_1 = 1$: $x_1(t_1) = \pi$, $x_2(t_1) = 0$.

Параметры точности и погрешности: $\varepsilon = 0.00001$, $R_1 = R_2 = 500$, $\xi_1 = \xi_2 = 0.01$.

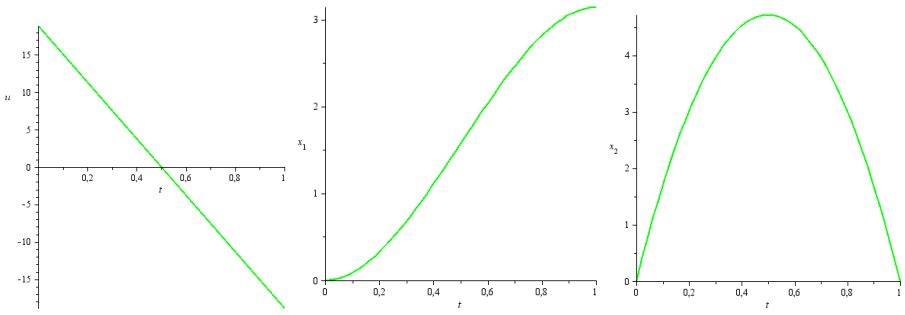

Рис. 6. Оптимальное управление и траектории, полученные с помощью принципа максимума ($I^* = 12 \cdot \pi \approx 118.435$)

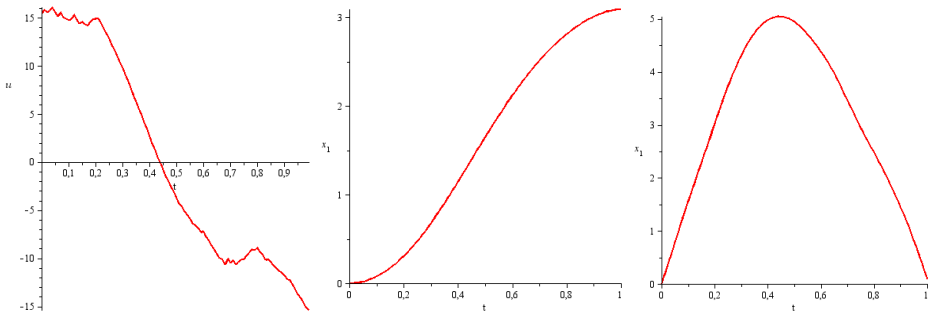

Рис. 7. Управление и траектории, полученные с помощью интервального метода взрывов ($I \in [118.445;118.478]$)

Задача о мягкой посадке. Уравнения, связывающие высоту h над поверхностью, вертикальную скорость v и массу m, имеют вид: $\dot{h} = v$, $\dot{v} = -g + \mu \cdot \frac{u}{m}$, $\dot{m} = -u$, где μ – скорость истечения газов относительно ракеты, $\mu \cdot \dot{m}$ - «тяга». Пусть в начальный момент времени ракета имела высоту h_0 и скорость v_0.

Функционал качества управления: $I = -\int_{t_0}^{t_1} u(t)dt$.

В начальный момент времени $t_0 = 0$: $m(t_0) = 500$, $h(t_0) = 100000$, $v(t_0) = 1000$ в конечный момент времени $t_1 \in T = [1000;10000]$: $h(t_1) = v(t_1) = 0$.

Параметры точности и погрешности: $\varepsilon = 0.00001$, $R_1 = R_2 = 10000$, $\xi_1 = 10$, $\xi_2 = 1$.

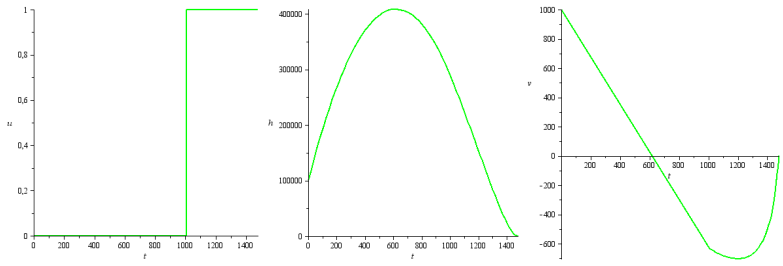

Рис. 8. Оптимальное управление и траектории ($I^* \approx 469.275$)

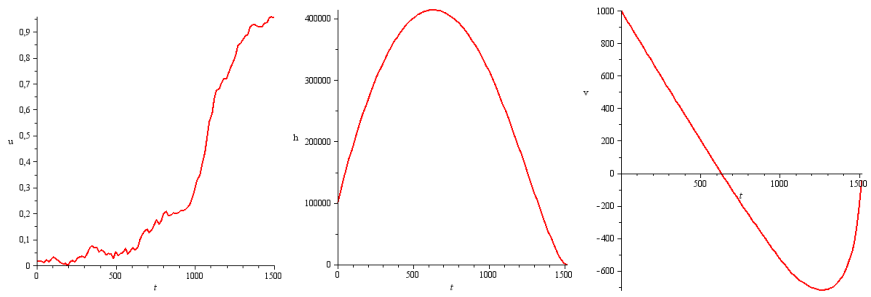

Рис. 9. Управление и траектории, полученные с помощью интервального метода взрывов ($I \in [470.124; 470.708]$)

Задача о стабилизации спутника. Рассматривается задача гашения вращательного движения спутника с помощью установленных на нем двигателей.

Система, описывающая движения твердого тела относительно центра инерции, после перехода к безразмерным переменным выглядит следующим образом: $\dot{p} = \frac{u_1}{6}$, $\dot{q} = u_2 - 0.2 \cdot r \cdot p$, $\dot{r} = 0.2 \cdot (u_3 + p \cdot q)$.

Функционал качества управления: $I = \int_{t_0}^{t_1} \left(|u_1(t)| + |u_2(t)| + |u_3(t)| \right) dt$.

В начальный момент времени $t_0 = 0$: $p(t_0) = 24$, $q(t_0) = 16$, $r(t_0) = 16$ в конечный момент времени $t_1 = 1$: $p(t_1) = q(t_1) = r(t_1) = 0$.

Параметры точности и погрешности: $\varepsilon = 0.00001$, $R_1 = R_2 = R_3 = 1000000$, $\xi_1 = \xi_2 = \xi_3 = 0.1$, $\xi_2 = 1$.

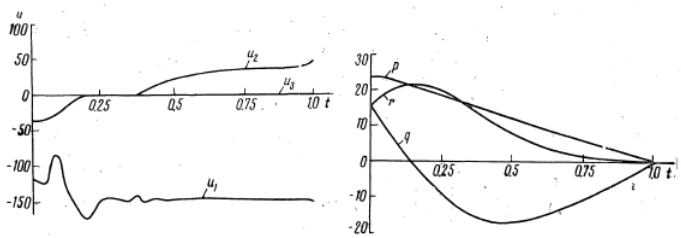

Рис. 10. Управление и траектории, полученные в [15] ($I^* \approx 169.42$)

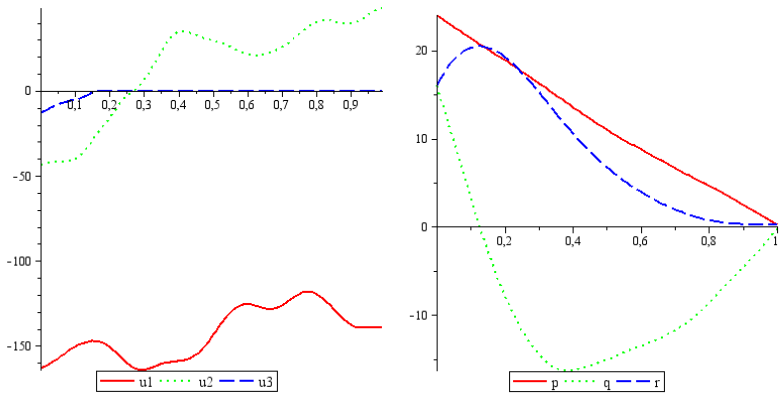

Рис. 11. Управление и траектории, полученные с помощью интервального метода взрывов ($I \in [173.61;173.75]$)

Рекомендации

Для повышения скорости сходимости метода можно предложить следующее:

- с каждой итерацией уменьшать вектор максимальных мощностей,
- заменить закон распределения случайной величины ξ на шаге взрывов на нормальный закон, математическое ожидание в котором с каждой итерацией сдвигалось бы в сторону нуля.

Введение данных правил позволит уменьшить радиус разлета осколков после взрыва, уменьшив вероятность того, что осколок «вылетит» из окрестности локального оптимума.

Заключение

Разработанный интервальный метод взрывов является простым (за счет небольшого количества параметров) и эффективным инструментом для решения задачи глобальной условной оптимизации. Данный алгоритм успешно

протестирован при решении прикладных задач поиска оптимального программного управления детерминированными системами.

Библиографический список

1. Ratschek H., Rokne J. New Computer Methods for Global Optimization. - Chichester: Horwood, 2007, 229 p.
2. Dussel R. Einschließung des Minimalpunktes einer streng konvexen Funktion auf einem n-dimensionalen Quader. - Karlsruhe, KIT, 1972, 165 p.
3. Hansen E. Global optimization using interval analysis. - New York: Marcel Dekker, 2004, 515 p.
4. Shary S.P. Randomized Algorithms in Interval Global Optimization // Numerical Analysis and Applications, 2008, v. 1, № 4, pp. 376-389.
5. Шарый С.П. Новый подход в интервальной глобальной оптимизации // Труды XII Байкальской международной конференции «Методы оптимизации и их приложения», 2001, т. 1, с. 289-295.
6. Панов Н.В., Шарый С.П. Интервальный эволюционный алгоритм для поиска глобального оптимума // Известия Алтайского государственного университета, 2011, т. 2, № 1(69), с. 108-113.
7. Moore R.E. Methods and applications of interval analysis. - Philadelphia: SIAM, 1979, 190 p.
8. Пановский В.Н. Интервальные алгоритмы нахождения оптимального программного управления детерминированными системами // Конкурс научно-технических работ и проектов «Молодежь и будущее авиации и космонавтики», 2012, с. 155.
9. Пановский В.Н. Применение аппарата интервального анализа для поиска глобального экстремума функций // Труды МАИ, 2012, № 51, с. 1-20.
10. Tan Y., Zhu Y. Fireworks Algorithm for Optimization // Advances in Swarm Intelligence. Lecture Notes in Computer Science, 2010, v. 6145, pp. 355-364.

11. Ahrari A., Shariat-Panahi M., Atai A.A. GEM: A Novel Evolutionary Optimization Method with Improved Neighborhood Search , *Applied Mathematics and Computation*, 2009, vol. 210, no. 2, pp. 376-386.

12. Moore R.E. Interval analysis. - Englewood Cliffs: Prentice Hall, 1966, 145 p.

13. Jaulin L., Kieffer M., Didrit O., Walter E. Applied Interval Analysis. - Berlin: Springer, 2001, 384 p.

14. Шарый С.П. Конечномерный интервальный анализ. – Новосибирск: XYZ, 2010, 606 с.

15. Крылов И.А. Численное решение задачи об оптимальной стабилизации спутника // Вычислительная математика и математическая физика, 1968, т. 8, № 1, с. 203 – 208.

16. Пантелеев А.В., Летова Т.А. Методы оптимизации. Практический курс. – М.: Логос, 2011, 425 с.

17. Пантелеев А.В., Метлицкая Д.В., Алешина Е.А. Методы глобальной оптимизации. Метаэвристические стратегии и алгоритмы. – М.: Изд-во Вузовская книга, 2013. 248 с.

18. Пантелеев А.В. Применение эволюционных методов глобальной оптимизации в задачах оптимального управления детерминированными системами. – М.: Изд-во МАИ, 2013. 160 с.