# ПОСТРОЕНИЕ ОБЛАСТИ ДОСТИЖИМОСТИ ДИНАМИЧЕСКОЙ СИСТЕМЫ НА NVIDEA И AMD ГРАФИЧЕСКИХ ПРОЦЕССОРАХ

**Ф.А. Витюков, В.К. Домашнев, А.П. Карпенко, В.А. Федин**

Задача построения области достижимости динамической системы (ОДДС) возникает, например, при решении проблемы траекторной безопасности летательного аппарата [1], близкой задачи о посадке вертолета на подвижный носитель [2], планировании движения многосекционного манипулятора типа «хобот» [3]. Важной особенностью задачи построения ОДДС является то, что ее часто приходится решать в режиме реального времени.

Аналитическое построение области достижимости удается лишь в простейших случаях, не представляющих практического интереса. Поэтому для построения этой области приходится использовать численные методы [4]. Одним из таких методов является метод «мультифиниша», идея которого состоит в многократном интегрировании модельной системы обыкновенных дифференциальных уравнений (ОДУ) при различных управлениях.

Эффективность параллельной реализации метода мультифиниша на MIMD вычислительных системах рассмотрена в работе [5]. В данной работе рассматривается реализация метода мультифиниша на NVidea и AMD графических процессорых устройствах (ГПУ) [6], [7].

В качестве примера в работе рассмотрена задача построения ОДДС для высокоманевренного летательного аппарата, описываемого системой ОДУ шестого порядка [1].

## 1. Постановка задачи

Рассмотрим динамическую систему

$$\dot{X} = F(t, X, U), \quad X(0) = X^0, \tag{1}$$

где $F = F(t, X, U) = (f_1(t, X, U), f_2(t. X, U), \ldots, f_n(t, X, U))^T$ - $n$-мерная вектор-функция, $X = X(t) = (x_1(t), x_2(t), \ldots, x_n(t))^T$ - $n$-мерный вектор фазовых переменных системы, $U = U(t) = (u_1(t), u_2(t), \ldots, u_m(t))^T$ - $m$-мерный вектор управлений, $X^0 = (x_1^0, x_2^0, \ldots, x_n^0)^T$ - $n$-мерный вектор начальных условий, $t \in [0, T]$. На вектор фазовых переменных $X$ и вектор управления $U$ наложены ограничения

$$X \in D_X, \qquad U \in D_U \subset L_U[0, T], \tag{2}$$

где $L_U[0, T]$ - некоторое пространство $m$-мерных функций, определенных на интервале $[0, T]$, например, пространство функций, интегрируемых с квадратом на этом интервале.

Среди фазовых переменных $x_1, x_2, \ldots, x_n$ выделим $v \le n$ переменных. Не ограничивая общности, положим, что эти переменные образуют $v$ -мерный вектор

$$Y = Y(t) = (x_1(t), x_2(t), \ldots, x_v(t))^T.$$

Областью достижимости системы (1) называется множество $D_Y = D_Y(T, X^0)$ всех возможных значений вектора $Y(T)$, которые принимаются на решениях этой системы при начальных условиях $X^0$ и выполнении ограничений (2).

*D-задача*: при заданных векторе начальных условий $X^0$, конечном времени $t_T$ и ограничениях (2) построить область достижимости $D_Y$ системы (1).

В некоторых важных приложениях удается найти множество допустимых управлений $D_U^\Gamma \subseteq D_U$, принадлежащих классу управлений $L_U^\Gamma[0, T] \subseteq L_U[0, T]$, которые приводят систему (1) на границу $\Gamma_Y$ ее области достижимости [1]. В этом случае задача построения ОДДС сводится к построению границы $\Gamma_Y$.

$\Gamma$ *-задача*: при заданных векторе начальных условий $X^0$, конечном времени $t_T$, фазовых ограничениях $X \in D_X$ и ограничениях на управления $U \in D_U^\Gamma \subset L_U^\Gamma[0, T]$ построить границу $\Gamma_Y$ области достижимости системы (1).

Далее во всех случаях будем исходить из того, что при интегрировании системы ОДУ (1) используется алгоритм с постоянным шагом интегрирования $\Delta t = T / K$, где $K$ - число шагов интегрирования. Положим, кроме того, что вычислительная сложность $C_F$ функции $F(t, X, U)$ при всех значениях аргументов $t, X, U$ многократно превышает вычислительную сложность затрат на реализацию используемого алгоритма

интегрирования на одном шаге интегрирования. Отметим, что для задач, представляющих практический интерес, такая ситуация является типичной.

2. **Метод мультифиниша**

Рассмотрим прежде метод мультифиниша применительно к - $D$ задаче. Покроем множество $D_U$ некоторой сеткой с узлами $U_1, U_2, \ldots, U_M$, где $M$ - общее количество узлов сетки. Поставим в соответствие системе (1) совокупность $M$ систем ОДУ с указанными управлениями:

$$\begin{cases} \dot{X}_1 = F(t, X_1, U_1), X_1(0) = X^0, \\ \quad \ldots \\ \dot{X}_M = F(t, X_M, U_M), X_M(0) = X^0. \end{cases} \tag{3}$$

Тогда схему приближенного построения ОДДС (1) методом мультифиниша можно представить в виде последовательности следующих шагов:

- путем интегрирования совокупности систем ОДУ (3) находим множество точек $\{Y_i(T), i \in [1:M]\}$, представляющее собой дискретную аппроксимацию области $D_Y$;
- во множестве $\{Y_i(T), i \in [1:M]\}$ находим граничные точки $\{Z_j(T), j \in [1:\varsigma]\}$, представляющие собой дискретную аппроксимацию границы $\Gamma_Y$ области достижимости;
- на основе точек $\{Z_j(T), j \in [1:\varsigma]\}$ строим подходящую непрерывную аппроксимацию $\tilde{\Gamma}_Y$ границы $\Gamma_Y$.

Перейдем к рассмотрению $\Gamma$ -задачи. Пусть класс функций $L_U^\Gamma[0,T]$ представляет собой класс релейных функций с не более чем одной точкой переключения. Положим, что одна из границ множества достижимости формируется управлениями этого класса, в которых все компоненты вектора управления, кроме одного, постоянны, а один из компонентов имеет одну точку переключения. Пусть, например,

$$u_1^\Gamma(t, t^S) = \begin{cases} +1, & t \in [0, t^S], \\ -1, & t \in (t^S, T], \end{cases} \qquad u_i^\Gamma \in \{u_{i,j_i}^\Gamma, j_i = 1,2,\ldots,k_i\} \quad , \quad i \in [2:m],$$

где $t^S \in [0,T]$ - момент времени, когда происходит переключение управления $u_1^\Gamma(t)$, $u_{i,j_i}^\Gamma$ - заданные константы, $k_i \geq 1$ - число уровней управления $u_i^\Gamma(t)$ [1].

Покроем интервал $[0,T]$ равномерной сеткой с шагом $\Delta t^S = T/M \leq \Delta t$ и узлами $t_j^S$, $j \in [1:k_1]$. Положим, что шаг $\Delta t^S$ кратен шагу $\Delta t$, так что $\Delta t^S / \Delta t = K / M = r$, где $r \geq 1$ - целое число. Пусть также имеет место равенство $M = k_1 k_2 \ldots k_m$. При этом множество управлений $U_1^\Gamma, U_2^\Gamma, \ldots, U_M^\Gamma$ естественно сформировать в виде

$$U_I^\Gamma = \left(u_1^\Gamma(t, t_{j_1}^S), u_{2,j_2}^\Gamma, u_{3,j_3}^\Gamma, \ldots, u_{m,j_m}^\Gamma\right)^T, \; i_j \in [1:k_j], \tag{4}$$

где $I = (j_1, j_2, \ldots, j_m)$ - мульти индекс; $j_i \in [1:k_i]$, $i \in [1:m]$.

Таким образом, схему приближенного решения $\Gamma$ -задачи методом мультифиниша можно представить в виде последовательности следующих шагов:

- путем интегрирования совокупности систем ОДУ (3) находим множество точек $\{Y_I^\Gamma(T), I = (i, j_i), i \in [1:m], j_i \in [1:k_i]\}$, представляющее собой дискретную аппроксимацию границы $\Gamma_Y$ области $D_Y$;
- на основе указанных точек строим подходящую непрерывную аппроксимацию $\tilde{\Gamma}_Y$ границы $\Gamma_Y$.

3. **NVIDIA и AMD графические процессорные устройства**

В настоящее время ГПУ активно используются для высокопроизводительных вычислений. Причин этому несколько - относительно невысокая стоимость ГПУ, их высокие производительность и доступность, невысокое энергопотребление.

Существуют два основных производителя ГПУ для решения обще вычислительных задач - компании NVidia и AMD. ГПУ этих производителей отличаются в деталях, но имеют сходную архитектуру, относящуюся к типу SIMD (ОКМД).

Основными вычислительными устройствами ГПУ являются RISC потоковые процессоры, которые при вычислениях исполняют одну и ту же вычислительную программу (шейдер) над разными исходными данными, находящимися в их регистрах общего назначения, число которых может доходить до 128. Число потоковых процессоров $N$ в современных ГПУ может доходить до ~200.

Совокупность исходных данных, которые обрабатываются шейдером, удобно представлять в виде некоторой целочисленной решетки $\Omega=\{\varpi_i\}$, узлы которой $\varpi_i$ ассоциируются с этими данными [7]. Система управления ГПУ осуществляет балансировку загрузки ГПУ путем распределения узлов $\varpi_i$ по потоковым процессорам. Для достижения высокой эффективности решения задачи на ГПУ число узлов решетки должно многократно превосходить число потоковых процессоров в ГПУ. Исполнение шейдера на одном из потоковых процессоров ГПУ называется потоком.

Известной проблемой современных процессоров является дисбаланс между скоростью работы процессора и скоростью обмена с памятью. Для ГПУ эта проблема стоит еще более остро, поскольку задержка доступа потокового процессора к ОЗУ центрального процессора может составлять несколько сотен тактов. Для того чтобы снизить влияние латентности доступа потоков к глобальной памяти, в ГПУ аппаратно поддерживается многопоточность - если один из потоков блокируется по доступу к памяти, система управления ГПУ передает на исполнение данному потоковому процессору другой поток. Поскольку контекст каждого из потоков хранится только в регистровой памяти потокового процессора, переключение потоков осуществляется очень быстро.

С точки зрения программирования вычислительная система на основе графических процессорных устройств (ГПУ) представляет собой сложную иерархическую структуру, имеющую несколько адресных пространств и ряд специфических механизмов. До недавнего времени широкое использование ГПУ для решения общевычислительных задач сдерживалось отсутствием адекватных средств программирования. Ситуация кардинально изменилась с появлением библиотек программирования таких, как CUDA (CUnifiedDriverArchitecture) компании NVidia и CAL (ComputationAbstractionLayer) компании AMD [9], [10]. . Различия между архитектурами NVIDEA и AMD ГПУ, а также между библиотеками CUDA и CAL рассмотрены, например, в работе [8].

В работе использовалось ГПУ GeForce 880GT (Compute Capability 1.1) компании NVidea, включающее в себя массив из 14 мультипроцессоров, модуль разделяемой памяти емкостью 16 KB, независимую кэш память команд (8 KB), аналогичную память констант (64 KB). Каждый из мультипроцессоров состоит из восьми потоковых процессоров, включающих в себя 8 KB регистровой памяти.

Вторым ГПУ, используемым в работе, является ГПУ Radeon HD5750, принадлежащее семейству RV840 компании AMD. Основой этого ГПУ является массив из 10 SIMD ядер, каждое из которых содержит по 16 блоков суперскалярных потоковых процессоров. Общее количество потоковых процессоров в ГПУ Radeon HD5750 равно $10*16*5=800$ .

## 4. Отображение метода мультфиниша на архитектуру ГПУ

Ограничимся рассмотрением $\Gamma$ -задачи. В простейшем случае отображение метода мультифиниша для $\Gamma$ -задачи на архитектуру ГПУ можно выполнить по следующей схеме:

- сопоставляем узлу $\varpi_I$ решетки $\Omega$ управление $U_I^\Gamma$, где $I=(j_1,j_2,\ldots,j_m)$, $j_i\in[1:k_i]$, $i\in[1:m]$;
- шейдеру ставим в соответствие задачу интегрирования системы ОДУ (1) в узле $\varpi_I$ (т.е. при управлении $U_I$).
- Отметим, что в таким образом построенной $m$ -мерной решетку $\Omega$, $i$ -му измерению соответствует сетка с узлами $u^\Gamma_{i,j_i}$.

Обозначим построенный шейдер $Sh_1$. В сделанных выше предположениях, вычислительная сложность шейдера $Sh_1$, легко видеть, равна $C(Sh_1)=KC_F$, а общий объем вычислительной работы для решения $\Gamma$ -задачи с использованием данного отображения - равен $C(P_\Gamma)=MKC_F$.

Очевидным развитием приведенной схемы является схема, в которой шейдеру (обозначим его в этом случае $Sh_2$) ставится в соответствие не один, а $q$ узлов решетки $\Omega$. В этом случае, очевидно, $C(Sh_2)=qKC_F$ и $C(P)=MKC_F$. Здесь и далее полагается, что величины $q$, $M$ кратны, так что число потоков $M/q$ - целая величина.

Метод мультифиниша для $\Gamma$ -задачи может быть отображен на архитектуру ГПУ и по более эффективной схеме $Sh_3$ (схема равномерной декомпозиции узлов расчетной сетки [5]):

- разбиваем решетку $\Omega$ на $zk_2k_3\ldots k_m$ блоков, где $z$ - число подмножеств, на которое разбита первое измерение решетки $\Omega$, т.е. узлы $u^{\Gamma}_{i,j_i}$, $j_1 \in k_1$;
- шейдеру ставим в соответствие задачу интегрирования системы ОДУ (1) в одном блоке;
- решение шейдером этой задачи организуем следующим образом:
  - исходя из начальных условий $X^0$, выполняет интегрирование системы (1) при управлении $\left(1, u^{\Gamma}_{2,j_2}, u^{\Gamma}_{3,j_3}, \ldots, u^{\Gamma}_{m,j_m}\right)$ от момента времени $0$ до момента времени $t^S_{i+q-1}$ и сохраняем значения компонентов векторов $X(t^S_i) = X^0_i$, $X(t^S_{i+1}) = X^0_{i+1}$ ,..., $X(t^S_{i+q-1}) = X^0_{i+q-1}$;
  - исходя из начальных условий $X^0_i$, выполняем интегрирование системы (1) при управлении $\left(-1, u^{\Gamma}_{2,j_2}, u^{\Gamma}_{3,j_3}, \ldots, u^{\Gamma}_{m,j_m}\right)$ от момента времени $t^S_i$ до момента времени $T$;
  - исходя из начальных условий $X^0_{i+1}$, выполняем интегрирование системы (1) при управлении $\left(-1, u^{\Gamma}_{2,j_2}, u^{\Gamma}_{3,j_3}, \ldots, u^{\Gamma}_{m,j_m}\right)$ от момента времени $t^S_{i+1}$ до момента времени $T$;
  - исходя из начальных условий $X^0_{i+q-1}$, выполняем интегрирование системы (1) при управлении $\left(-1, u^{\Gamma}_{2,j_2}, u^{\Gamma}_{3,j_3}, \ldots, u^{\Gamma}_{m,j_m}\right)$ от момента времени $t^S_{i+q-1}$ до момента времени $T$.

  Вычислительная сложность шейдера $Sh_3$ оценивается величиной

$$C(Sh_3) = KC_F + r\sum_{i=i+1}^{i+q-1}(M-j)C_F = KC_F + r(q-1)\left(M-\left(i+\frac{q}{2}\right)\right)C_F,$$

где $q = k_1 / z$.

Шейдер $Sh_3$ порождает потоки, имеющие различную вычислительную сложность. Так, очевидно, поток, соответствующий узлам $\varpi_{1,j_2,j_3,\ldots,j_m}$,..., $\varpi_{q,j_2,j_3,\ldots,j_m}$ имеет сложность, многократно превышающую сложность потока, соответствующего узлам $\varpi_{k_1-q+1,j_2,j_3,\ldots,j_m}$,..., $\varpi_{k_1 j_2,j_3,\ldots,j_m}$. Примерно равную вычислительную сложность потоков можно обеспечить за счет шейдера $Sh_4$, использующего неравномерную декомпозицию узлов расчетной сетки [5].

5. **Пример**

Рассмотрим высокоманевренный летательный аппарат, уравнения движения центра масс которого в нормальной земной системе координат $0xyz$ описываются системой нелинейных дифференциальных уравнений

$$\begin{cases} \dot{v} = g\cdot(n_T - \sin\Theta), \\ \dot{\Theta} = g/v\,(n\cdot\cos\gamma_C - \cos\Theta), \\ \dot{\Psi} = -g\cdot n\cdot\sin\gamma_C / v\cdot\cos\Theta, \\ \dot{x} = v\cdot\cos\Theta\cdot\cos\Psi, \\ \dot{y} = v\cdot\sin\Theta, \\ \dot{z} = -v\cdot\cos\Theta\cdot\sin\Psi, \end{cases} \tag{5}$$

где $v$ – скорость летательного аппарата, $\Theta$ – угол наклона траектории, $\Psi$ – угол поворота траектории, $y$ – высота летательного аппарата, $n_T$ – тангенциальная перегрузка, $n$ – нормальная перегрузка, $\gamma_C$ – скоростной угол крена, $g$ – ускорение свободного падения [1].

Управлениями летательного аппарата являются тангенциальная перегрузка, нормальная перегрузка и скоростной угол крена, так что $U = (n_T, n, \gamma_C)^T$. На управления наложены ограничения

$$n_T^{\min} \le n_T \le n_T^{\max}, \quad n_T^{\min} = -1.6, \quad n_T^{\max} = 0.6;$$
$$|n| \le n^{\max}, \quad n^{\max} = 8; \qquad |\gamma_C| \le \pi.$$

В работе [1] показано, что дальняя, ближняя и боковая границы области достижимости системы (4) формируются управлениями, принадлежащими классу кусочно-постоянных управлений. Ограничимся

рассмотрением дальней границы области достижимости. Для каждого допустимого $\gamma_C(t)=const$ структура управлений, формирующих эту границу, представлена на рисунке 1 [1].

Таким образом, в данном случае $m=3$ и $U_I^\Gamma=\left(u_1^\Gamma(t,t^S_{j_1}),u^\Gamma_{2,j_2},u^\Gamma_{3,j_3}\right)^T$, где $u_1^\Gamma=n$, $u_2^\Gamma=\gamma_C$, $u_3^\Gamma=n_T$; $j_i\in[1:k_i]$, $i=1,2,3$, $k_3=1$.

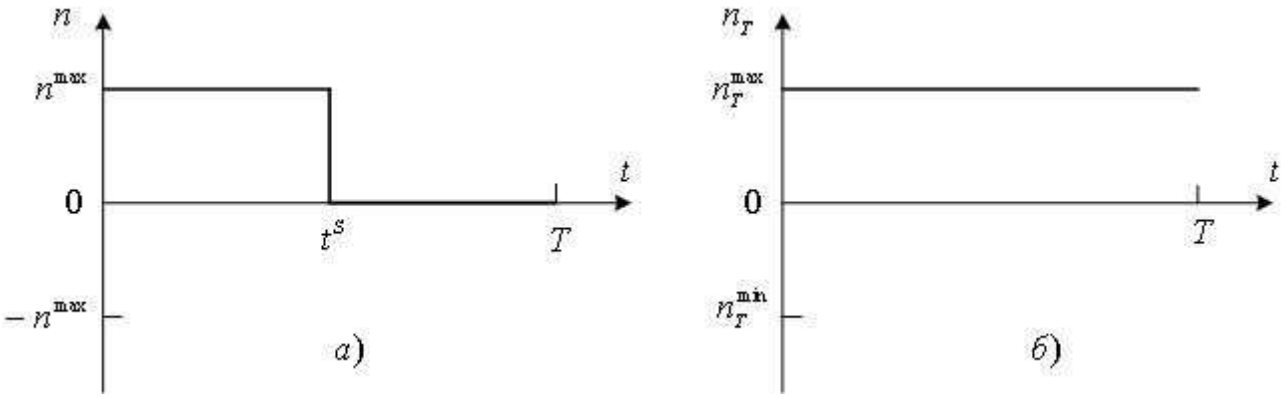


Рис. 1. Структура управлений, формирующих дальнюю границу области достижимости летательного аппарата

Исследование эффективности как ГПУ GeForce 880GT, так и ГПУ Radeon HD5750 при построении дальней границы области достижимости системы (5) выполнено для шейдеров $Sh_2$, $Sh_3$ и следующих значений параметров:

- $k_1=2048$; $k_2=128$; $k_3=1$;
- $q=1,2,4,8,16$.

В качестве меры эффективности использовано ускорение $S=\dfrac{T_{CPU}}{T_{GPU}}$, равное отношению времени последовательного решения задачи на центральном процессоре системы $T_{CPU}$ к времени ее параллельного решения на ГПУ $T_{GPU}$.

В качестве центрального процессора для ГПУ GeForce 880GT использовался процессор Intel Core 2Duo E4700/2,6 Гц/2 ГБ, а для ГПУ Radeon HD 5750 – процессор Intel Core 2Duo E4500/2,21 Гц/2 ГБ.

Эффективность шейдеров $Sh_2$, $Sh_3$ для ГПУ GeForce 880GT иллюстрирует рисунок 2. Рисунок показывает, что, как и следовало ожидать, шейдер $Sh_3$ обеспечивает значительно более высокое ускорение $S(Sh_3)$ по сравнению с ускорением $S(Sh_2)$, которое обеспечивает шейдер $Sh_2$. Ускорение $S(Sh_2)$ монотонно убывает с ростом величины $q$, что объясняется уменьшением при этом общего количества потоков и, как следствие, ухудшением балансировки загрузки ГПУ. Ускорение $S(Sh_3)$ имеет при $q=4$ максимум, равный ~300. Снижение ускорения при последующем росте величины $q$ также объясняется ухудшением балансировки загрузки ГПУ.

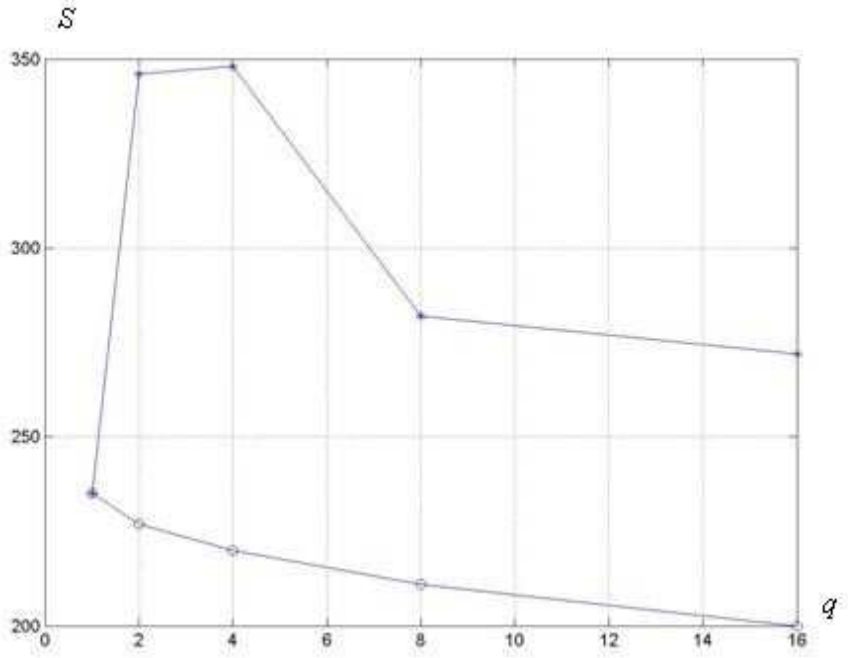


Рис. 2. Ускорение ГПУ GeForce 880GT (o - шейдер $Sh_2$; * - шейдер $Sh_3$)

Для ГПУ Radeon HD5750 эффективность шейдеров $Sh_2$ , $Sh_3$ иллюстрирует рисунок 3, аналогичный рисунку 2. Рисунок показывает, что шейдер $Sh_3$ обеспечивает более высокое ускорение, чем шейдер $Sh_2$ . В обоих случаях ускорения имеют при $q=2$ максимум, равный примерно 161 и 169 соответственно. С дальнейшим ростом величины $q$ оба ускорения монотонно убывают, что, как и для ГПУ GeForce 880GT, объясняется ухудшением балансировки загрузки ГПУ.

Отметим, что в силу использования разных центральных процессоров, сравнение между собой ускорений, которые обеспечивают указанные ГПУ, не вполне корректно.

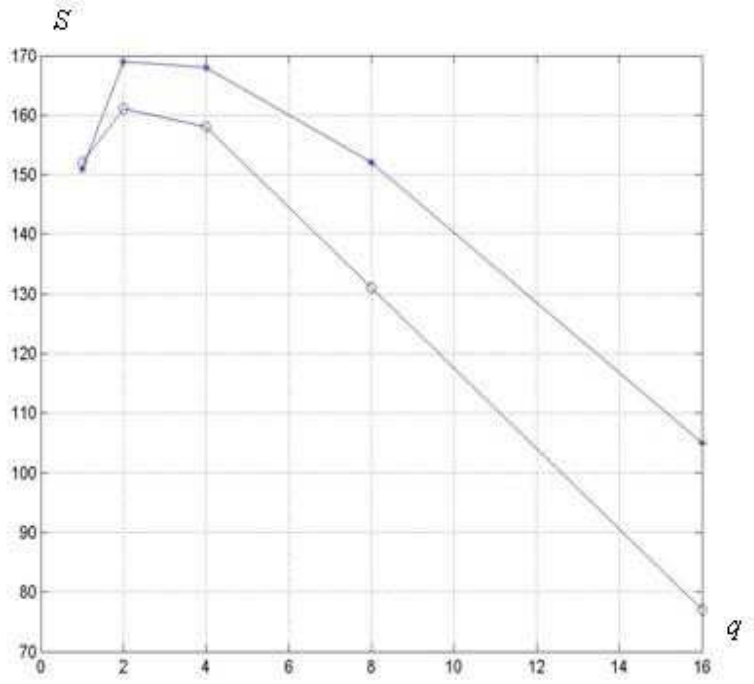


Рис. 3. Ускорение ГПУ Radeon HD5750 (o - шейдер $Sh_2$; * - шейдер $Sh_3$)

Пример дальней границы области достижимости для рассматриваемого летательного аппарата приведен на рисунке 4. Рисунок получен при $T=5$ с и следующем начальном положении летательного аппарата:

$v(0)=330{,}0$ м/с; $\Theta(0)=0$ ; $\Psi(0)=0$ ; $x(0)=0$ ; $y(0)=10000{,}0$ м; $z(0)=0$ .

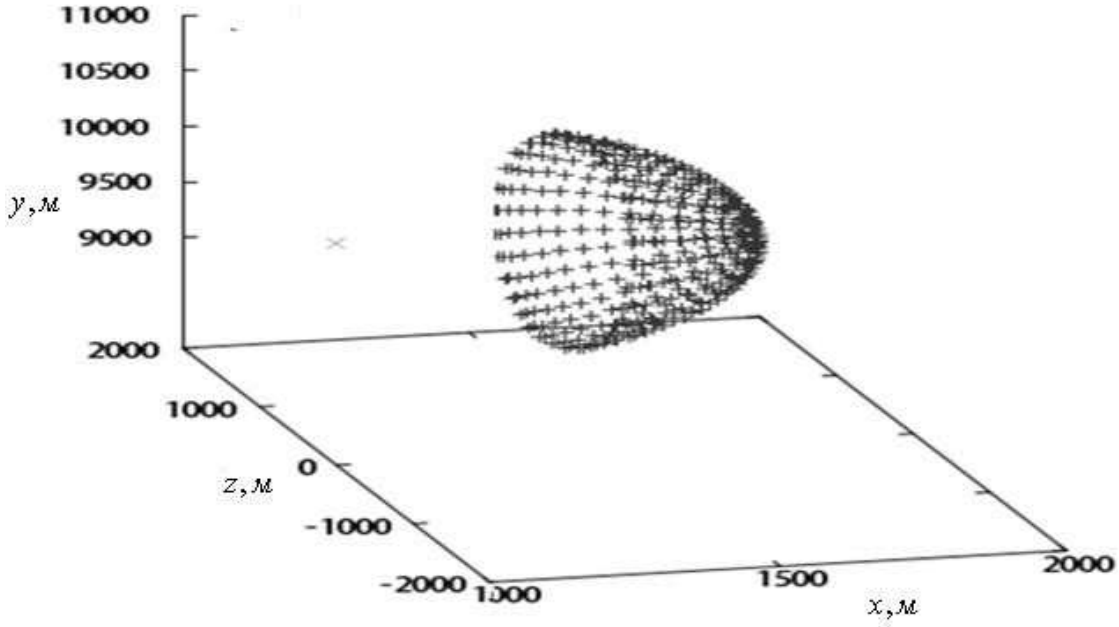


Рис. 4. Пример дальней границы области достижимости системы (5)

**Заключение**

В работе предложено несколько ориентированных на ГПУ схем распараллеливания метода мультифиниша при приближенном построении границы области достижимости динамической системы. Выполнено сравнительное исследование эффективности предложенных схем распараллеливания при построении границы области достижимости высокоманевренного летательного аппарата с помощью ГПУ GeForce880 GT компании NVidea, а также с помощью ГПУ Radeon HD5750 компании AMD.Проведенное исследование показало перспективность практической реализации рассмотренных схем метода мультифиниша на ГПУ.

В развитии работы предполагается реализация и исследование эффективности шейдера $Sh_4$, реализующего схему неравномерной равномерной декомпозиции узлов расчетной сетки. Кроме того, предполагается реализация метода мультифиниша на вычислительной системе, содержащей несколько ГПУ.

Работа выполнена в рамках аналитической ведомственной целевой программы «Развитие потенциала высшей школы (2009 – 2010 годы)», проект 2.1.2/1509

ЛИТЕРАТУРА:

1. Е.М. Воронов, А.А. Карпунин. Алгоритм оценки границ области достижимости летательного аппарата с учетом тяги // Вестник МГТУ. Сер. Приборостроение.- 2007.- №4(69).- С. 81-99.
2. В.И. Гурман, В.И. Квоков, М.Ю. Ухин. Приближенные методы оптимизации управления летательным аппаратом // Автоматика и телемеханика.- 2008.- №4.- С. 191–201.
3. С.М. Волкоморов, А.П. Карпенко. Оптимизация целевой конфигурации робота-манипулятора типа «хобот» // Наука и образование: электронное научно- техническое издание, 2010, №3, (http://technomag.edu.ru/doc/138076.html).
4. Е.М. Воронов, А.П. Карпенко, О.Г. Козлова, В.А. Федин. Численные методы построения области достижимости динамической системы // Наука и образование: электронное научно- техническое издание, 2010, №1. (http://technomag.edu.ru).
5. Е.М. Воронов, А.П. Карпенко, В.А. Федин. Параллельное построение множества достижимости высокоманевренного летательного аппарата методом «мультифиниша» [Электронный ресурс] // Параллельные вычислительные технологии (ПаВТ’2010): Труды международной научной конференции. – Челябинск: Издательский центр ЮУрГУ, 2010. – с. 113-120.
6. NVidia Tesla C1060. URL: (http://www.nvidia.com/object/product_tesla_c1060_us.html).
7. AMD FireStream 9270. URL: (http://ati.amd.com/technology/streamcomputing/product_irestream_9270.html).
8. А. Адинец, Вл.В Воеводин. Графический вызов суперкомпьютерам // Открытые системы. -2008. -№ 4. -С. 35–41.
9. Nvidia CUDA Programming Guide Version 3.0, 2010.
10. OpenCL Programming Guide 1.0.48, 2010.