Параллельное построение множества достижимости высокоманевренного летательного аппарата методом «мультифиниша»

Е.М. Воронов, А.П. Карпенко, В.А. Федин

Рассматривается задача построения области достижимости динамической системы. Исследуется подход к решению этой задачи на основе многократного интегрирования модельной системы обыкновенных дифференциальных уравнений при различных управлениях. Предлагается метод балансировки загрузки многопроцессорной вычислительной системы при параллельном решении задачи на основе указанного подхода. Описывается реализация этого метода средствами коммуникационной библиотеки MPI. Приводятся результаты параллельного решения задачи для системы, описывающей динамику высокоманевренного летательного аппарата.

1. Введение

Во многих прикладных областях, использующих модели исследуемых объектов в виде обыкновенных дифференциальных уравнений (ОДУ), возникает задача построения области достижимости соответствующей динамической системы. Например, такая задача возникает при решении проблемы траекторной безопасности летального аппарата [1], а также при решении близкой задачи о посадке вертолета на подвижный носитель [2]. Важной особенностью задачи построения области достижимости 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.

Дискретную аппроксимацию области достижимости динамической системы можно построить путем многократного интегрирования модельной системы ОДУ при различных управлениях (метод мультифиниша). В практических задачах типичной является ситуация, когда основные вычислительные затраты при построении области достижимости методом мультифиниша обусловлены затратами на интегрирование указанной системы ОДУ. Поэтому ускорить вычисления можно за счет параллельного интегрирования этой системы ОДУ.

К параллельным методам интегрирования ОДУ относятся блочные методы, а также методы на основе геометрической схемы декомпозиции системы ОДУ.

Методы первого класса основаны на использовании одношаговых и многошаговых блочных формул интегрирования ОДУ, например, формулы Адамса-Башфорта [3]. В вычислительной практике обычно используются не более чем 4-х точечные блочные методы. Поэтому потенциальный параллелизм этой группы методов невелик.

Идея методов второго класса состоит в разбиении модельной системы ОДУ на подсистемы, количество которых равно количеству используемых процессоров вычислительной системы, и интегрировании каждой из этих подсистем ОДУ на своем процессоре. В силу, как правило, относительно невысокой размерности вектора фазовых координат исследуемой динамической системы данные методы также принципиально не могут обеспечить значительное ускорение.

В работе полагается, что, используемая многопроцессорная вычислительная система (МВС) представляет собой *однородную* MIMD-систему с общей памятью [4]. Поскольку метод мультифиниша использует интегрирование совокупности систем модельных ОДУ, естественным в данном случае является распараллеливание на основе разделения этой совокупности на наборы, количество которых равно количеству процессоров в используемой МВС. В работе рассматривается проблема балансировки загрузки МВС при распараллеливании метода мультифинаша по указанной схеме, получены оценки эффективности параллельных вычислений. В качестве примера рассмотрена задача построения области достижимости для летального аппарата, описываемого системой ОДУ шестого порядка.

2. Постановка задачи

Рассмотрим динамическую систему

$$\begin{cases} \dot{x}_1 = f_1(t, x_1, ..., x_n, u_1, ..., u_m),\ x_1(0) = x_1^0, \\ ... \\ \dot{x}_n = f_n(t, x_1, ..., x_n, u_1, ..., u_m),\ x_n(0) = x_n^0, \end{cases} \tag{1}$$

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

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

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

Для компактности записи используем векторную форму системы (1)

$$\dot{X} = F(t, X, U),\ X(0) = X^0, \tag{3}$$

где $F = F(t, X, U) = (f_1(t, X, U), f_2(t, X, U), ..., f_n(t, X, U))^T$ – n-мерная вектор-функция правой части системы ОДУ (1).

Среди фазовых переменных $x_1, x_2, ..., x_n$ выделим $\nu \le n$ переменных. Не ограничивая общности, положим, что эти переменные образуют ν-мерный вектор $Y = Y(t) = (x_1(t), x_2(t), ..., x_\nu(t))^T$.

Областью достижимости $D_Y = D_Y(T, X^0)$ системы (3) назовем множество всех возможных значений вектора $Y(T)$, которые достигаются на решениях системы (3) при начальных условиях X^0 и ограничениях (2).

Рассмотрим следующую *прямую задачу*: при заданном векторе начальных условий X^0 и конечном времени t_T построить область достижимости D_Y системы (3). Рассмотрим также *обратную задачу*: при заданных векторе начальных условий X^0, конечном времени T и точке $Y(T)$, принадлежащей области достижимости D_Y, найти управление $U \in D_U$, переводящее систему (3) в эту точку.

Для решения прямой задачи, очевидно, достаточно построить границу Γ_Y области достижимости D_Y. В некоторых случаях 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 (3) на эту границу [1]. В таком случае прямая задача сводится к построению границы Γ_Y.

Сделаем следующие допущения. Во всех случаях при интегрировании системы ОДУ (3) используется алгоритм с постоянным шагом интегрирования $\Delta t = \frac{T}{K}$, где $K = K(T)$ – количество шагов интегрирования. Для каждого из шагов интегрирования требуется l вычислений значения функции $F(t, X, U)$, вычислительная сложность которой (количество арифметических операций, необходимых для однократного вычисления значения этой функции) не зависит от аргументов t, X, U и равна C_F. Вычислительная сложность затрат на реализацию алгоритма интегрирования на одном шаге интегрирования равна $C_I = C_I(l)$.

Покроем множество D_U некоторой сеткой с узлами $U_1, U_2, ..., U_M$, где M – общее количество узлов сетки. Поставим в соответствие системе (3) совокупность M систем ОДУ с указанными управлениями:

$$\begin{cases} \dot{X}_1 = F(t, X_1, U_1), X_1(0) = X^0, \\ \quad\quad ... \\ \dot{X}_M = F(t, X_M, U_M), X_M(0) = X^0. \end{cases} \quad (4)$$

Тогда схему приближенного решения прямой задачи методом мультифиниша можно представить в следующем виде:

1) Путем интегрирования совокупности систем ОДУ (4) находим множество точек $\{Y_i(T), i \in [1:M]\}$, представляющее собой дискретную аппроксимацию области D_Y.

2) Запоминаем полученные наборы значений $(U_i, Y_i(T))$, $i \in [1:M]$.

3) Во множестве $\{Y_i(T), i \in [1:M]\}$ находим граничные точки $\{Z_j(T), j \in [1:\varsigma]\}$, представляющие собой дискретную аппроксимацию границы Γ_Y области достижимости.

4) Запоминаем соответствующие наборы значений $(U_j, Z_j(T))$, $j \in [1:\varsigma]$.

На основе полученных точек $\{Z_j(T), j \in [1:\varsigma]\}$ может быть построена непрерывная аппроксимация $\tilde{\Gamma}_Y$ границы Γ_Y.

3. Метод мультифиниша

3.1. Общая схема распараллеливания вычислений

Схема распараллеливания вычислений имеет следующий вид:

1) процессор P_i, $i \in [1:N]$ получает от *host* процессора векторы $U_{(i-1)\mu+1}, U_{(i-1)\mu+2}, ..., U_{i\mu}$;

2) процессор P_i, интегрирует при указанных управлениях систему ОДУ (3) – находит множество точек $\{Y_j(T), j \in [(i-1)\mu+1, i\mu]\}$;

3) процессор P_i передает координаты этих точек *host* -процессору.

Здесь N – количество процессоров в используемой МВС; $\mu = \left\lceil \frac{M}{N} \right\rceil$, где $\lceil v \rceil$ – символ ближайшего целого большего *v*.

При построении границы Γ_Y области достижимости D_Y возникает проблема балансировки загрузки МВС. Рассмотрим для примера случай, когда 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,

$$u_1^\Gamma(t, t^S) = \begin{cases} +1, & t \in [0, t^S], \\ -1, & t \in (t^S, T], \end{cases} \quad u_2^\Gamma(t) = u_3^\Gamma(t) = ... = u_m^\Gamma(t) = const,$$

где $t^S \in [0,T]$ – момент времени, когда происходит переключение управления $u_1^\Gamma(t)$ [1]. Далее рассмотрим два способа распределения точек переключения по процессорам для подобного управления.

3.2. Равномерная декомпозиция точек переключения

Покроем интервал $[0,T]$ равномерной сеткой с шагом $\Delta t^S = \frac{T}{M} \le \Delta t$ и узлами $t_i^S, i \in [1:M]$. Положим, что шаг Δt^S кратен шагу Δt, так что $\frac{\Delta t^S}{\Delta t} = \frac{K}{M} = r$, где $r \ge 1$ – целое число. При этом множество управлений $U_1^\Gamma, U_2^\Gamma, ..., U_M^\Gamma$ естественно сформировать в виде $U_i^\Gamma = \left(u_1^\Gamma(t, t_i^S),\ u_2^\Gamma,\ u_3^\Gamma,\ ...,\ u_m^\Gamma\right)^T$, $i \in [1:M]$. Таким образом, в данном случае схема распараллеливания имеет вид, представленный на Рис. 1.

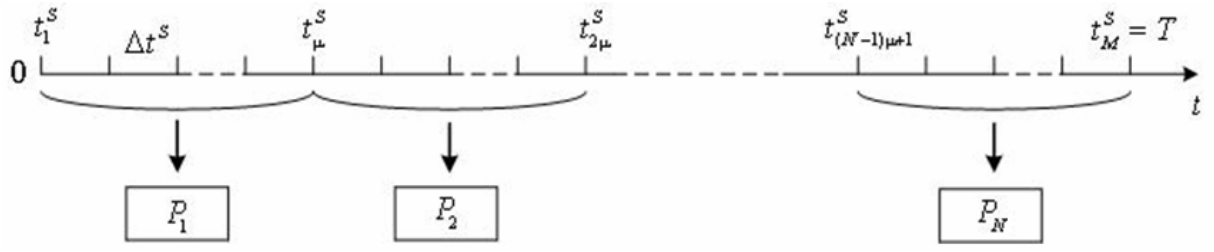

Рис. 1. Схема распараллеливания при равномерной декомпозиции точек переключения

В соответствии с этой схемой на процессоре P_1 выполняется интегрирование системы ОДУ (3) при управлениях $U_1^\Gamma, ..., U_\mu^\Gamma$, на процессоре P_2 – при управлениях $U_{\mu+1}^\Gamma, ..., U_{2\mu}^\Gamma$ и т.д. до процессора P_N, который выполняет интегрирование при управлениях $U_{(N-1)\mu+1}^\Gamma, ..., U_M^\Gamma$.

Вычисления на процессоре P_i организуем следующим образом.

Шаг 0 (этап «разгона»). Исходя из начальных условий X^0, выполняем интегрирование системы (3) при управлении $(1, u_2^\Gamma, u_3^\Gamma, ..., u_m^\Gamma)$ от момента времени 0 до момента времени $t_{i\mu-1}^S$ и запоминает значения компонентов векторов $X(t_{(i-1)\mu+1}^S)$, $X(t_{(i-1)\mu+2}^S)$, …, $X(t_{i\mu-1}^S)$.

Шаг 1. Исходя из начальных условий $X(t_{(i-1)\mu+1}^S)$, выполняем интегрирование системы (3) при управлении $(-1, u_2^\Gamma, u_3^\Gamma, ..., u_m^\Gamma)$ от момента времени $t_{(i-1)\mu+1}^S$ до момента времени T.

Шаг 2. Исходя из начальных условий $X(t_{(i-1)\mu+2}^S)$, выполняем интегрирование системы (3) при управлении $(-1, u_2^\Gamma, u_3^\Gamma, ..., u_m^\Gamma)$ от момента времени $t_{(i-1)\mu+2}^S$ до момента времени T.

…

Шаг μ. Исходя из начальных условий $X(t_{i\mu-1}^S)$, выполняем интегрирование системы (3) при управлении $(-1, u_2^\Gamma, u_3^\Gamma, ..., u_m^\Gamma)$ от момента времени $t_{i\mu-1}^S$ до момента времени T.

В рассмотренной схеме каждый из последующих процессоров повторяет шаг 0 своего предыдущего процессора и только затем выполняет интегрирование системы (3) на «своем» интервале $[t_{(i-1)\mu}^S, t_{i\mu-1}^S]$. Если вычислительные затраты на интегрирование системы (3) велики, то может оказаться целесообразной более тонкая организация вычислений на этапе разгона – однократное интегрирование системы (3) на интервале $[0, T]$ при управлении $(1, u_2^\Gamma, u_3^\Gamma, ..., u_m^\Gamma)$ на одном процессоре и передача значений необходимых компонентов векторов $X(t_1^S)$, $X(t_2^S), \ldots, X(t_M^S)$ остальным процессорам МВС. Однако, если $\mu >> 1$, что можно считать типичной ситуацией, выигрыш от такой организации вычислений не может быть существенным.

Оценим ускорение рассмотренной схемы распараллеливания. В сделанных допущениях время выполнения процессором P_i этапа «разгона» равно

$$T_{i,0} = t_c r(i\mu - 1)(lC_F + C_I); \tag{5}$$

время выполнения *j*-го шага, где $j \in [1:\mu]$, можно оценить величиной

$$T_{i,j} = t_{cal} r(M - (i-1)\mu - j)(lC_F + C_I). \tag{6}$$

Таким образом, общее время решения задачи процессором P_i равно

$$T_i = T_{i,0} + \sum_{j=1}^{\mu} T_{i,j}. \tag{7}$$

Здесь и далее t_{cal} – время выполнения арифметической операции на одном процессоре используемой МВС.

Поскольку загрузка первого процессора P_1 в данном случае, очевидно, максимальна, время параллельного решения задачи определяется выражением

$$\tau_{NN} = T_1 = \sum_{j=1}^{\mu} t_{cal} r(M - j)(lC_F + C_I) \approx t_{cal} r(lC_F + C_I)\frac{M^2}{N},$$

где учтено, что $M >> \mu$. Аналогично, время последовательного решения задачи на одном процессоре оценим величиной

$$\tau_1 = t_{cal} r(lC_F + C_I)\sum_{i=1}^{M}(M - (i-1)) \approx \frac{1}{2} t_{cal} r(lC_F + C_I)M^2.$$

Отсюда имеем

Утверждение 1. Ускорение S параллельного метода на основе равномерной декомпозиции точек переключения управления $u_1^{\Gamma}(t)$ оценивается величиной $S \approx \frac{N}{2}$.

Из утверждения 1 следует, что равномерная декомпозиция точек переключения управления $u_1^{\Gamma}(t)$ не может обеспечить ускорение, превышающее 50% от максимального потенциально возможного ускорения, равного N.

Лучшей балансировки загрузки многопроцессорной вычислительной системы можно добиться за счет неравномерной декомпозиции точек переключения управления $u_1^{\Gamma}(t)$.

3.3. Неравномерная декомпозиция точек переключения

Положим, что процессор P_i выполняется интегрирование системы ОДУ (3) при управлениях $U_a^{\Gamma}, ..., U_b^{\Gamma}$, а процессор P_{i+1} – при управлениях $U_{b+1}^{\Gamma}, ..., U_c^{\Gamma}$, $0 \le a < b < c \le M$. Потребуем, чтобы времена решения задачи процессорами P_i, P_{i+1} были равны, т.е. чтобы выполнялось равенство $T_i = T_{i+1}$, $i \in [1 : N-1]$. Из выражений (5) – (7) следует, что для этого необходимо выполнение равенства $b + \sum_{j=a}^{b}(M - j) = c + \sum_{j=b}^{c}(M - j)$ или, что то же самое, равенства

$$b + \frac{2M - (b+a)}{2}(b-a) = c + \frac{2M - (c+b)}{2}(c-b). \qquad (8)$$

Отсюда вытекает квадратное уравнение для определения величины c при известных значениях величин a, b:

$$c^2 - 2(M+1)c + \left(a^2 - 2b^2 - 2Ma + 2(2M+1)b\right) = 0. \qquad (9)$$

Если для процессора P_1 задаться величинами $a = a_1 = 0$, $b = b_1$, то из выражения (9) легко найти последовательно управления, которые должны обрабатывать процессоры $P_2, P_3, ...$

Рассмотрим для примера случай $M = 1000$, $b_1 = 100$. Из (9) вытекает, что в данном случае можно обеспечить равномерную загрузку 6 процессоров (таблица 1). Отметим, что здесь количество точек переключения Δt_6^S вычислено по остаточному принципу и не удовлетворяет уравнению (9).

Оценим ускорение рассматриваемого метода. Положим, что в последовательном режиме задача решается по схеме, аналогичной рассмотренной выше (с этапом «разгона»). Тогда время последовательного решения задачи можно оценить величиной

$$\tau_1 = t_{cal} r(lC_F + C_I)\left(M - 1 + \sum_{i=1}^{M}(M - i)\right) = t_{cal} r(lC_F + C_I)(M-1)(1 + 0.5M).$$

Поскольку в данном случае вычислительная загрузка всех используемых процессоров (исключая, быть может, последний процессор) примерно одинакова, оценка времени параллельного решения задачи следует из выражения (8) и равна

$$\tau_{NN} = T_1 = t_{cal} r(lC_F + C_I)\left((M+1)b_1 - 0.5b_1^2\right).$$

Таблица 1. Пример неравномерной декомпозиции точек переключения

Процессор P_i	t_a^S	t_b^S	Δt_i^S
P_1	0	100	100
P_2	101	212	111
P_3	213	342	129
P_4	343	504	151
P_5	505	756	251
P_6	757	1000	243

Утверждение 2. Ускорение S параллельного метода на основе неравномерной декомпозиции точек переключения управления $u_1^{\Gamma}(t)$ оценивается величиной

$$S = \frac{(M-1)(1+0.5M)}{(M+1)b_1 - 0.5b_1^2} \approx \frac{M^2}{2Mb_1 - b_1^2}.$$

Из утверждения 2 следует, что для рассмотренного выше примера ускорение равно

$$S \approx \frac{1000^2}{2\times 1000\times 100 - 100^2} \approx 5.26.$$

Заметное отличие ускорения от максимально возможного (равного 6) объясняется тем, что загрузка последнего процессора P_6 далека от оптимальной загрузки.

Рассмотрим другой пример. Пусть $\Delta t_1^S = 88$. Тогда по формуле (9) получим $\Delta t_2^S = 98$, $\Delta t_3^S = 109$, $\Delta t_4^S = 129$, $\Delta t_5^S = 168$, $\Delta t_6^S = 405$. Поскольку при этом все шесть процессоров загружены практически равномерно, ускорение равно

$$S \approx \frac{1000^2}{2\times 1000\times 88 - 88^2} \approx 5.93.$$

Рекуррентное уравнение (8) не удается решить явным способом. Поэтому количество точек переключения $b_1 = \Delta t_1^S$, обрабатываемых процессором P_1, приходится подбирать таким образом, чтобы вычислительная загрузка всех процессоров, включая последний процессор, была примерно одинакова. При этом критерий окончания перебора удобно строить, исходя из того, что при несбалансированности загрузки процессора P_N корни уравнения (9) оказываются комплексными.

4. Пример

Рассмотрим летальный аппарат, уравнения движения центра масс которого в нормальной земной системе координат $0xyz$ описываются системой нелинейных дифференциальных уравнений (10), в которой v – скорость летательного аппарата, θ – угол наклона траектории, ψ – угол поворота траектории, y – высота летательного аппарата, n_t – тангенциальная перегрузка, n_n – нормальная перегрузка, γ_c – скоростной угол крена, g – ускорение свободного падения.

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

$$n_t^{\min} \le n_t \le n_t^{\max},\ n_t^{\min} = -1.6,\ n_t^{\max} = 0.6;\ |n_n| \le n_n^{\max},\ n_n^{\max} = 8;\ \gamma_c \le |\pi|.$$

$$\begin{cases} \dot{v} = g \cdot (n_t - \sin\theta), \\ \dot{\theta} = g/v \cdot (n_n \cdot \cos\gamma_c - \cos\theta), \\ \dot{\psi} = -g \cdot n_n \cdot \sin\gamma_c / v \cdot \cos\theta, \\ \dot{x} = v \cdot \cos\theta \cdot \cos\psi, \\ \dot{y} = v \cdot \sin\theta, \\ \dot{z} = -v \cdot \cos\theta \cdot \sin\psi. \end{cases} \quad (10)$$

В работе [1] показано, что дальняя, ближняя и боковая границы области достижимости системы (10) формируются управлениями, принадлежащими классу кусочно-постоянных управлений. Ограничимся рассмотрением дальней границы области достижимости. Структура управлений, формирующих эту границу, представлена на Рис. 2.

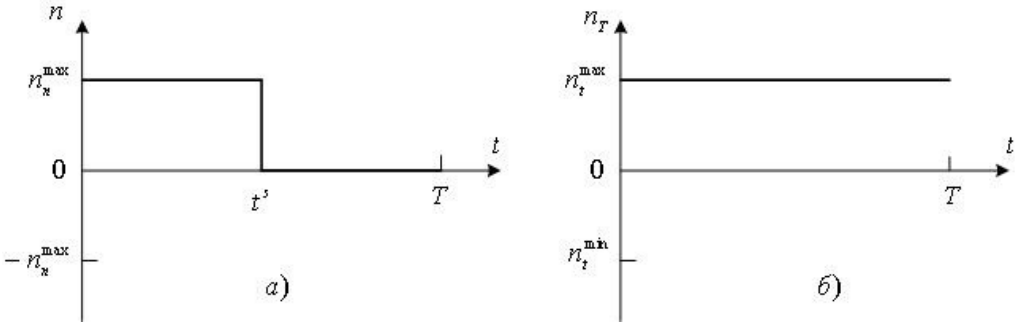

Рис. 2. Структура управлений, формирующих дальнюю границу области достижимости

Для экспериментального исследования эффективности метода мультифиниша разработана MPI-программа [4], реализующая этот метод. Эксперименты выполнены на виртуальном кластере, созданном с помощью программной системы VMware и функционирующим под управлением свободно распространяемой операционной системы Ubuntu. Результаты экспериментов иллюстрирует рис. 4, который показывает, что на числе процессоров от 2 до 10 достигается ускорение, близкое к расчетному.

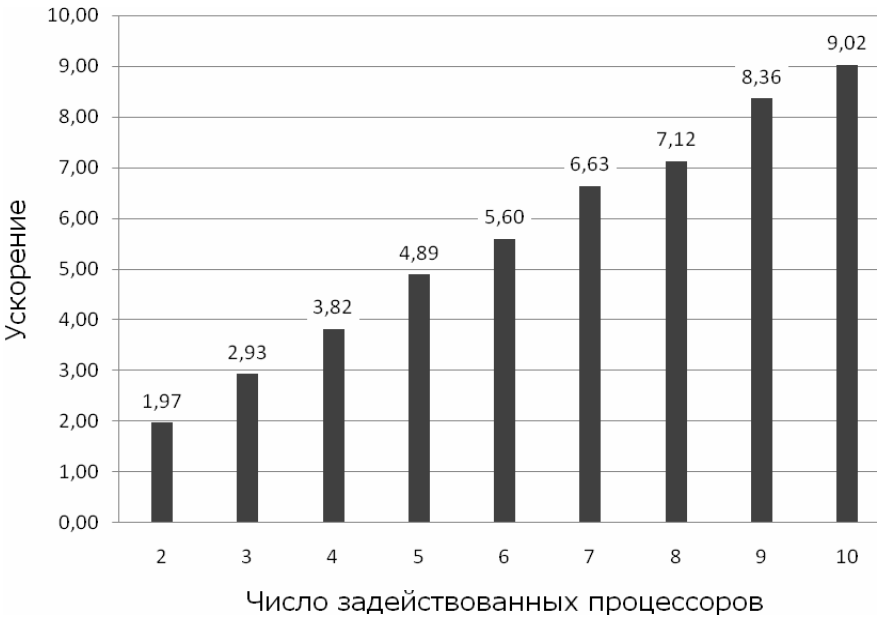

Рис. 4. Экспериментальные результаты

5. Заключение

В работе рассмотрен метод мультифиниша для построения области достижимости динамической системы. Предложен соответствующий параллельный алгоритм, ориентированный на вычислительные системы класса MIMD. Рассмотрены два метода балансировки загрузки этих вычислительных систем. Эффективность метода исследована аналитически, а также экспериментально на примере построения границ области достижимости высокоманевренного летательного аппарата. Исследование показало перспективность практического использования метода мультифиниша.

Авторы благодарят К.О. Вишневецкого за помощь в проведении вычислительных экспериментов.

Литература

1. Воронов Е.М., Карпунин А.А. Алгоритм оценки границ области достижимости летательного аппарата с учетом тяги // Вестник МГТУ. Сер. Приборостроение.- 2007.- №4(69).- С. 81-99.
2. Гурман В.И., Квоков В.И., Ухин М.Ю. Приближенные методы оптимизации управления летальным аппаратом // Автоматика и телемеханика.- 2008.- №4.- С. 191–201.
3. Claus Bendtsen. Parallel Numerical Algorithms for the Solution of Systems of Ordinary Differential Equations // PhD Dissertation, Institute of Mathematical Modeling Technical University of Denmark, June 1996.
4. Воеводин В.В., Воеводин Вл.,В. Параллельные вычисления.– СПб.: БХВ-Петербург, 2004.– 608 с.