**BC/NW 2016 № 1 (28):4.1**

**ИССЛЕДОВАНИЕ СХОДИМОСТИ МЕТОДА ТАНГЕНСОВ ПРИ РАСЧЕТЕ ПРОИЗВОДИТЕЛЬНОСТИ ВЫЧИСЛИТЕЛЬНЫХ СЕТЕЙ**

Абросимов Л.И., Сокин Д.В.

**Введение**

Как правило, современные сети имеют довольно сложные структуры и характеризуются множеством различных параметров и характеристик, которые следует учитывать при их проектировании. При построении моделей наибольшее внимание уделяется тем параметрам и характеристикам, которые непосредственно интересуют исследователя-проектировщика.

В рамках используемого метода контуров [1], обеспечивающего расчет производительности вычислительных систем, одним из наиважнейших этапов разработки является определение временных характеристик, позволяющих оценить качество функционирования всей системы в целом.

Каждый узел сети в рамках метода контуров рассматривается как одноканальная система массового обслуживания с неограниченной очередью ().

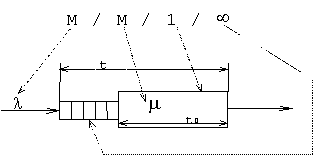


Рисунок 1. Узел как система

Основными параметрами сообщений при этом являются интенсивность поступления сообщений () и интенсивность обслуживания сообщений узлом (). А основными характеристиками узлов, как систем массового обслуживания, являются: время обслуживания сообщений узлом, время задержки сообщений в системе (т.е. в узле и очереди на обслуживание), коэффициент загрузки узла, количество сообщений в системе обслуживания и количество сообщений в очереди на обслуживание.

Основные параметры сообщений:

* – интенсивность поступления сообщений (усредненное количество сообщений, поступающих на вход узла в единицу времени);
* – интенсивность обслуживания сообщений узлом (усредненное количество сообщений, поступающих на вход узла в единицу времени).

Основные характеристики узлов, как систем массового обслуживания:

* – время обслуживания сообщений узлом;
* – время задержки сообщений в системе (в узле и очереди на обслуживание);
* – коэффициент загрузки узла ;
* (формула Литтла) – количество сообщений в системе обслуживания (в узле и очереди на обслуживание);
* – количество сообщений в очереди на обслуживание.

В зависимости от типа трафика (разомкнутый или замкнутый), проходящего через участок сети, при использовании метода контуров для вычисления его временных характеристик составляются линейные или нелинейные уравнения баланса.

Решение данных уравнений позволяет получить необходимые данные, для вычисления временных характеристик на последующих этапах метода контуров. При этом ключевую роль будут играть методы решения данных уравнений, а так же их сходимость, зависящая от исходных данных.

**Постановка задачи**

Основной задачей данной работы является исследование сходимости алгоритма решения систем нелинейных уравнений баланса. Составление и решение таких систем позволяет получить искомые значения базовых интенсивностей поступления сообщений-заявок для каждого отдельного контура замкнутой сети.

Основу составления каждого нелинейного уравнения составляет условие, что для установившегося процесса движения сообщений одного контура , количество сообщений в контуре не изменяется и равно сумме математических ожиданий сообщений во всех узлах сети рассматриваемого контура. Тогда для каждого отдельного контура можно записать равенство (1):

, (1)

где – заданное количество сообщений в контуре – математическое ожидание количества сообщений в узле , контура .

Для определения математического ожидания количества сообщений для каждого отдельного узла определенного контура , в системе обслуживания, используется формула Литтла (2):

*,* (2)

где – время задержки сообщений в системе (в узле и очереди на обслуживание).

При этом для уменьшения погрешности при расчете характеристик замкнутых систем необходимо использовать коэффициент ограниченности очереди (3):

,(3)

А при учете, что в рамках задачи при составлении нелинейных уравнений все определенного контура принимаются равными, т.е. , формула (1) преобразуется к виду (4):

, (4)

что позволит нам составить нелинейные уравнения в канонической форме (5):

, (5)

Решением системы таких нелинейных уравнений будут значения , при которых .

В качестве метода решения таких систем был выбран метод Тангенсов, предложенный профессором Абросимовым Л. И. [1], который, в отличие от метода дихотомии, обеспечивает гарантированное решение нелинейных уравнений при достаточно большой загрузке сети.

**Алгоритм метода тангенсов**

Алгоритм данного метода включает два основных этапа вычислений: пошаговую процедуру расчета каждого отдельного нелинейного уравнения и итеративную процедуру решения системы нелинейных уравнений.

На каждом цикле итеративной процедуры выполняется последовательное нахождение всех пошаговым методом тангенсов. Таким образом, достигается повышенная достоверность получаемого результата.

Самым ответственным является этап задания начальных условий, когда задаются исходные значения . Базовым значением , гарантирующим сходимость алгоритма, является точка , в которой . В рамках исходного алгоритма предлагается использовать формулу (6), позволяющую ускорить сходимость и уменьшить количество операций:

, (6)

где – базовое значение интенсивности поступающих сообщений контура на 0-й итерации; – интенсивность обработки сообщений узлом ; – количество контуров, проходящих через узел .

Уравнение (6) выполняется для всех контуров , после чего происходит решение первого уравнения пошаговым методом.

При вычислении пошаговой процедуры для первого уравнения, вначале задается значение интенсивности входного потока текущего контура , при котором , а все остальные значения фиксируются (, для всех не равных номеру текущего контура). Затем находится значение тангенса наклона прямой, численно равного коэффициенту крутизны , определяемого отношением граничных значений к (рисунок 2):

, (7)

где коэффициент при введен с целью обеспечения сходимости пошаговой процедуры.

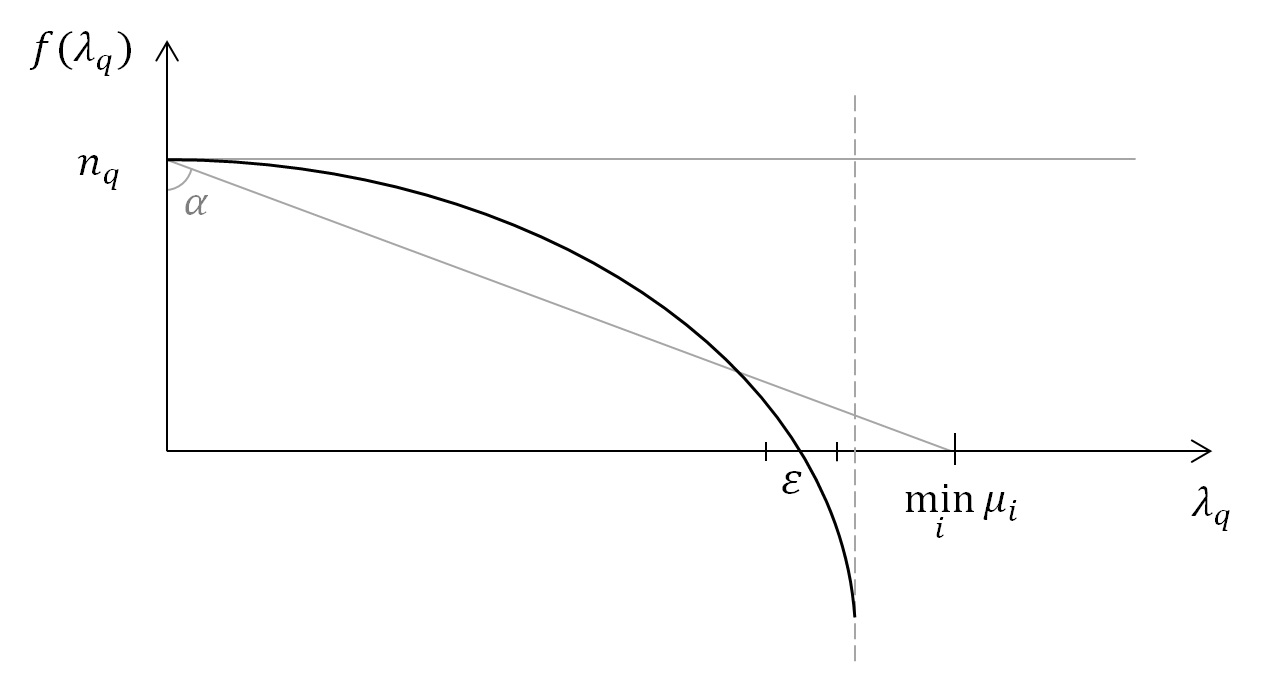


Рисунок 2. Вычисление первого тангенса

После этого начинается сам циклический процесс определения , посредством соотношений:

(8)

, (9)

где – значение приращения .

В результате вычислений возможна ситуация выхода из зоны допустимых решений, признаком чего является значение . В этом случае будет необходимо применять коррекцию :

|  |  |
| --- | --- |
| * если , то , а ; | (10) |
| * если , то . |

Сходимость пошаговой процедуры для одного контура изображена на рисунке 3.

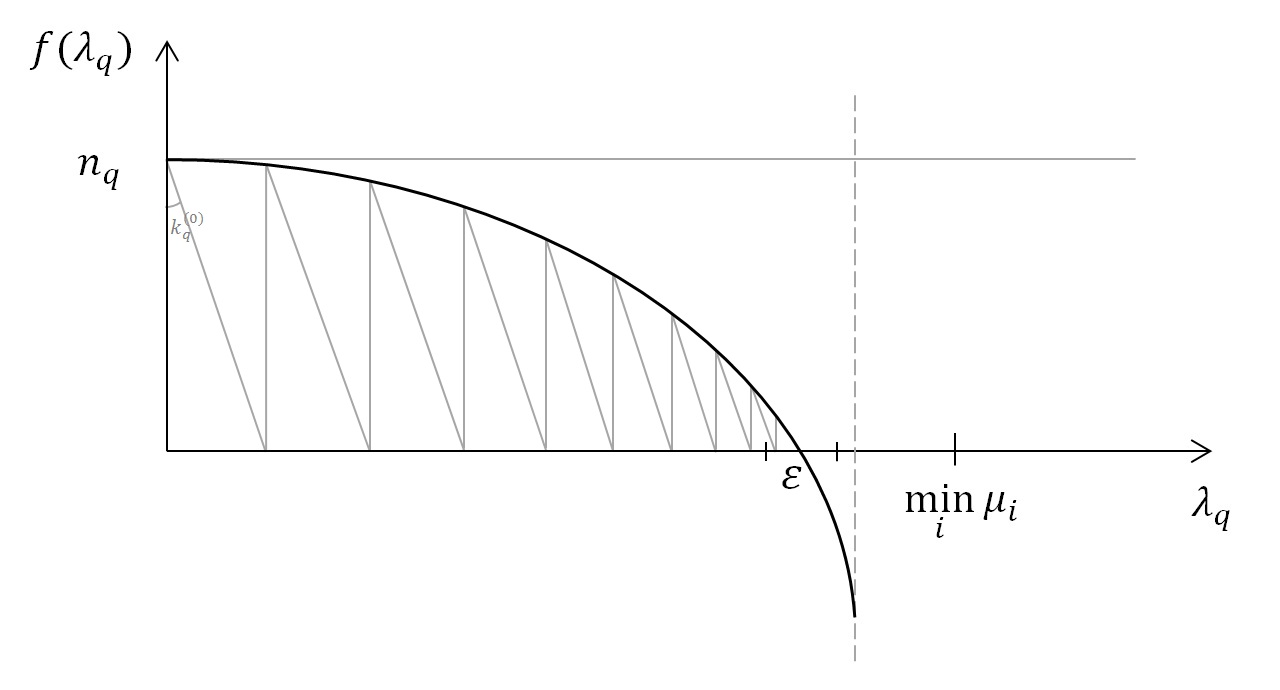


Рисунок 3. Сходимость пошаговой процедуры

Циклический пошаговый процесс продолжается до тех пор, пока на   
каком-то шаге не будет выполнено условие окончания:

(11)

где – заданная точность вычислений.

Выполнение условия (11) будет означать окончание пошаговой процедуры для текущего контура, в конце которой мы получим значение , где – номер текущей итерации, а – номер последнего шага, на котором было выполнено условие (11).

Далее аналогичным образом находятся значения других уравнений, в результате чего мы получим множество значений первой итерации ().

После этого начинается второй этап, на котором при решении каждого уравнения на каждой последующей итерации определяется значение интенсивности соответствующей итерации, при этом значения остальных интенсивностей рассматриваются как фиксированные, численно равные значениям последних итераций этих интенсивностей.

Следует отметить, что базовые значения задаются единожды в начале алгоритма, и при последующих итерациях вместо них будут выступать значения, полученные на последней итерации.

Итерационная процедура продолжается до тех пор, пока не будет выполнено условие окончания цикла:

**Исследование сходимости алгоритма метода тангенсов**

Рассмотрим следующий пример. Дана простая вычислительная сеть с тремя узлами и двумя замкнутыми контурами (рисунок 4). Контур обслуживают узлы 1 и 3, а контур обслуживают узлы 2 и 3. Для рассматриваемой вычислительной сети составляется система из двух нелинейных уравнений, и задаются исходные данные. Необходимо рассчитать интенсивности поступления сообщений для рассматриваемой системы.

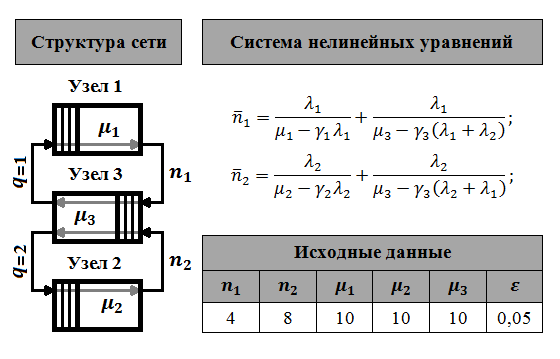


Рисунок 4. Исходные данные примера.

Результаты пошаговой процедуры для первой итерации контура 1, при , представлены в таблице 1.

Таблица 1. Результаты пошаговой процедуры для 1 итерации контура 1, при .

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Шаг** | **0** | **1** | **2** | **3** | **4** | **5** | **6** |
|  | 0 | 2,5000 | 4,3077 | 4,5759 | 4,4418 | 4,4516 | 4,4467 |
|  | 4 | 2,8923 | 0,4291 | -0,4410 | 0,0315 | -0,0004 | 0,0156 |

По этим результатам был построен график функции «» (рис. 5), где в качестве ограничений выступают значения «» по первому слагаемому (красная граница) и «» по второму слагаемому (оранжевая граница). Основной границей является минимальное между ними (), т.е. значение «», представляющее границу для .

Рисунок 5. График функции «», построенный по данным таб. 1.

По результатам выполнения алгоритма метода тангенсов были получены значения, представленные в таблице 2.

Таблица 2. Результаты вычислений нелинейных уравнений баланса.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Итерация** | **0** | **1** | **2** | **3** | **4** | **5** | **6** |
|  | 5,0000 | 4,4467 | 4,1028 | 3,8134 | 3,6916 | 3,5655 | 3,5277 |
|  | 5,0000 | 5,4926 | 5,8603 | 6,0585 | 6,1328 | 6,2041 | 6,2244 |

На основании результатов был нарисован график сходимости алгоритма (рис. 6).

Рисунок 6. Иллюстрация сходимости алгоритма по данным таб. 2.

Таким образом, можно убедиться, что алгоритм сходится, и вычисленные значения могут быть использованы в последующих этапах метода контуров, для расчета временных характеристик.

Для того, чтобы построить более полный график сходимости, иллюстрирующий границы для значений вычисляемых интенсивностей поступающих сообщений, расчеты были проведены повторно в два этапа со значением начальных .

При первом этапе сначала вычислялось значение для первого контура, а затем для второго (на всех итерациях). Во время второго этапа порядок был противоположным. Таким образом, были получены значения для обоих этапов, представленные в таблице 5.

Таблица 3. Результаты вычислений нелинейных уравнений баланса.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Контур 1** | | | | | | | | | | | | | | |
|  | 7,4205 | 5,7363 | 4,8038 | 4,2469 | 3,9115 | 3,7321 | 3,5778 | 3,5314 | 3,2455 | 3,1507 | 2,9208 | 2,5253 | 1,7619 | 0 |
|  | 0 | 3,0264 | 4,4664 | 5,2730 | 5,6846 | 5,9947 | 6,1087 | 6,1974 | 6,6284 | 6,7905 | 7,0973 | 7,6366 | 8,6203 | 10 |
| **Контур 2** | | | | | | | | | | | | | | |
|  | 10 | 7,4205 | 5,7363 | 4,8038 | 4,2469 | 3,9115 | 3,7321 | 3,5778 | 3,5314 | 3,1507 | 2,9208 | 2,5253 | 1,7619 | 0 |
|  | 0 | 3,0264 | 4,4664 | 5,2730 | 5,6846 | 5,9947 | 6,1087 | 6,1974 | 6,2225 | 6,6284 | 6,7905 | 7,0973 | 7,6366 | 8,6203 |

По результатам, представленным в данной таблице, был построен график сходимости, представленный на рисунке 7.

Рисунок 7. Развернутый график сходимости алгоритма по данным таб. 3.

В данном случае алгоритм сходится и на уровне решения отдельных уравнений и на уровне решения всей системы в целом. Но возможны случаи, когда при определенных значениях исходных данных алгоритм не будет сходиться.

В качестве примера рассмотрим случай, когда значение будет выходить за логическую границу по на пошаговой процедуре для отдельного нелинейного уравнения. В таком случае будет происходить разрыв функции и получаемое значение будет неверным, хотя алгоритм может продолжить поиск решения (рисунок 8).

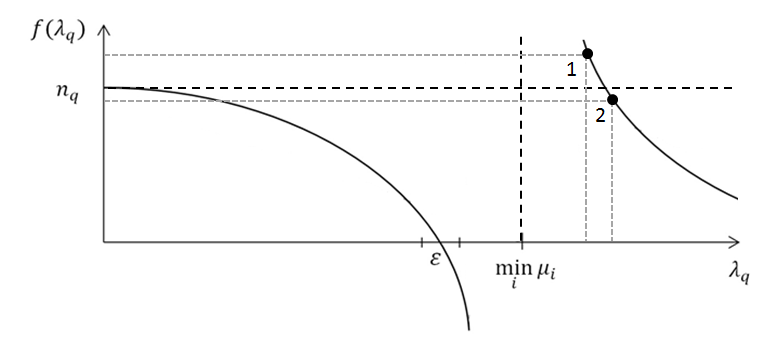


Рисунок 8. Случаи не сходимости алгоритма.

Такой исход связан с тем, что одно или несколько слагаемых нелинейного уравнения могут принять отрицательные значения при определенном , но при этом в результате значение функции будет положительным, что удовлетворяет изначальному условию .

Для решения этой проблемы можно было бы ввести дополнительное ограничение , но это не избавит нас от проблемы полностью, т.к. даже при этом ограничении возможен случай, когда выйдет за пределы допустимых значений, но при этом условия будут выполняться.

Чтобы полностью исключить эту проблему, достаточно ввести условие не отрицательности всех слагаемых уравнения, и тогда алгоритм всегда будет сходиться на уровне пошаговой процедуры.

Т.е. значение функции должно удовлетворять условию , а все его слагаемые не должны быть отрицательными, и при этом должно выполняться условие .

Но помимо этого так же возможен случай на уровне решения всей системы в целом. Для обеспечения сходимости на уровне итерационной процедуры, была использована формула, для определения порядка вычислений, т.к. при выборе в качестве начального контура, со значением большим, возможен случай расходимости алгоритма.

Использование данных ограничений для обеих процедур расчета гарантирует сходимость алгоритма при различных значениях исходных данных сети.

**Разработанная программа**

Для ускорений расчетов в процессе исследования была написана консольная программа, блок-схема алгоритма которой изображена на рисунке 9. Блок-схемы функций, используемых программой, приведены на рисунках 10 и 11.

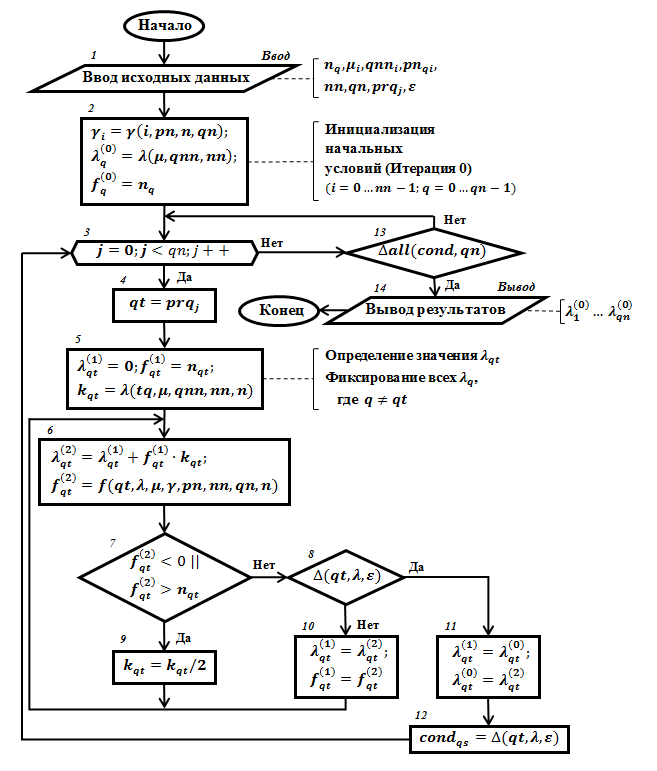


Рисунок 9. Блок-схема программы

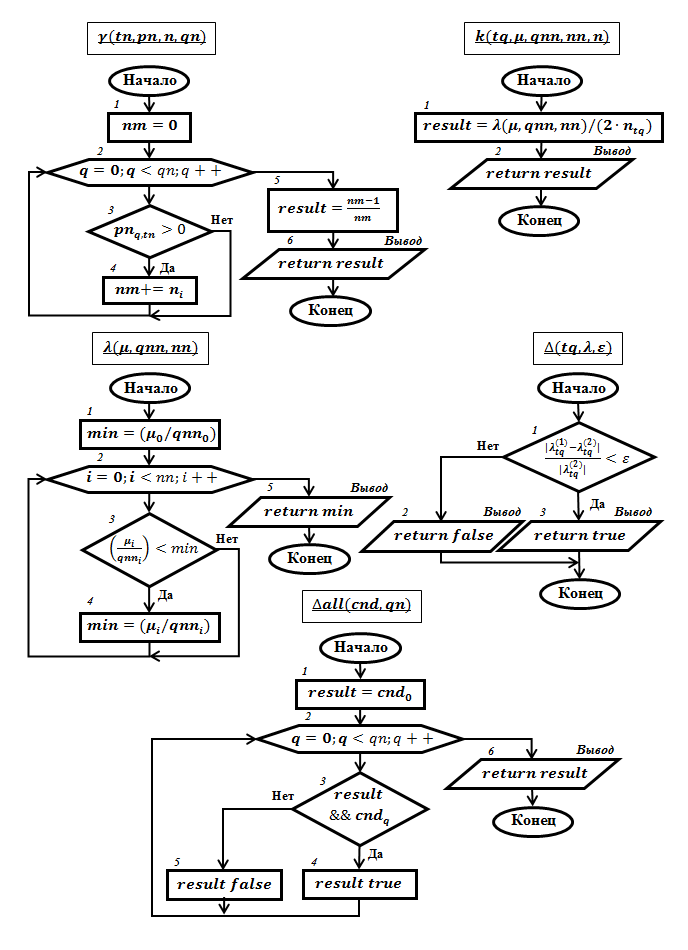


Рисунок 10. Блок-схемы функций программы (Часть 1).

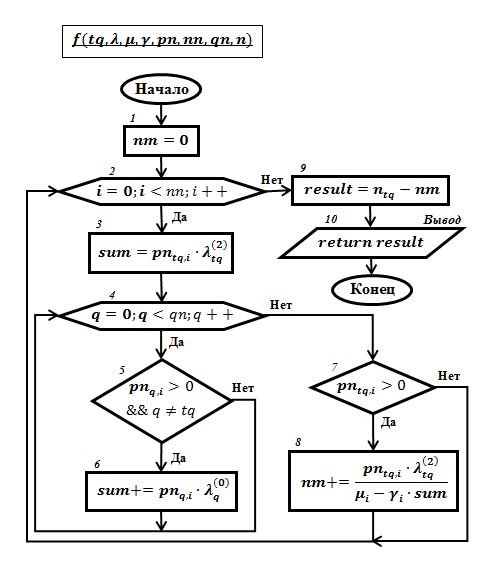


Рисунок 11. Блок-схема функции (часть 2).

В отличие от описанного ранее алгоритма метода тангенсов, алгоритм программы имеет некоторые отличительные особенности. В частности, составление нелинейных уравнений происходит внутри программы исходя из исходных данных, после чего сразу решается. Кроме того вместо нумерации шагов и итераций была введена система буферов, позволяющая хранить три различных значения. Так же здесь не указаны некоторые функции, как например функция инициализации исходных данных () и функция задания начальных условий (), а так же функции вывода результатов, и т.п. Программа писалась на языке C/C++ в программной среде   
Microsoft Visual Studio 2011, под 32-х битную версию Windows 7.

***Расшифровки и пояснения***

Исходные данные**:**

Вычисляемые данные**:**

* результат проверки условия на окончание цикла для контура

**Исследование производительности программы**

Замеры производительности проводились на следующей вычислительной системе (таблица 4):

Таблица 4. Характеристики вычислительной системы.

|  |  |
| --- | --- |
| **Тип системы** | Нетбук |
| **Модель системы** | Samsung N150 Plus (Black) |
| **Центральный процессор** | Intel Atom CPU N450 |
| **Частота ЦП** | 1.66 GHz 1.67 GHz (1666 MHz) |
| **Количество ядер** | 2 |
| **Кэш L1 кода** | 32 КБ |
| **Кэш L1 данных** | 24 КБ |
| **Кэш L2** | 512 КБ |
| **Системная плата** | Samsung N150P |
| **Жесткий диск** | 232 ГБ |
| **Файл подкачки** | 2110 МБ |
| **Виртуальная память** | 4137 МБ |
| **Оперативная память** | DIMM1: Samsung M4 70T2864FB3-CF7 (DDR2 SDRAM, 1 GB, 400 МГц, 64 bit) |
| **Операционная система** | Windows 7 SP1, 32 bit |
| **Частота высокоточного счетчика** | 1623554 тактов/сек |

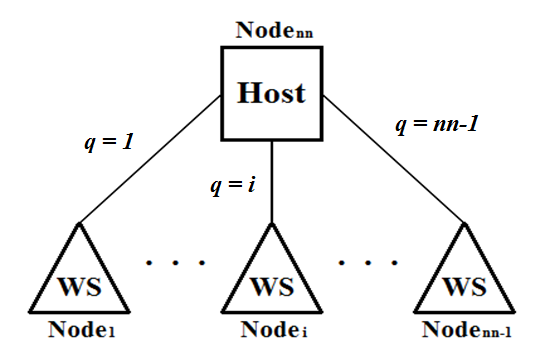
Исходные данные для примеров задавались по следующему алгоритму:

* , где ;
* , где ;
* ;

где: – количество узлов в сети; – количество контуров сети; – количество сообщений в контуре ; – интенсивность обработки сообщений узлом ; – интенсивность обработки сообщений сервером.

Вычисления нелинейных уравнений в каждой итерации проводились по возрастанию номера контура, то есть по порядку.

Структура сети для всех примеров выбиралась следующим образом: для каждого примера было задано множество узлов, среди которых один (последний по номеру) является общим хостом (сервером). Так же было задано множество контуров, каждый из которых проходит лишь через два узла: через хост, и узел, номер которого совпадает с номером контура. Пример структуры сети, состоящей из узлов, изображен на рисунке 12.

****  
Рисунок 12. Структура сети.

Таким образом, видно, что в данном случае все исходные данные зависят от количества узлов ().

Замеры проводились в два этапа: с выводом промежуточных результатов, и без вывода промежуточных результатов. Во время второго этапа так же замерялся объем памяти, выделенный для программы, переменных и массивов. Для каждого примера обоих этапов было проведено по десять замеров для получения среднего значения времени выполнения алгоритма. Инициализация значений исходных данных при этом не учитывалась, как и вывод результатов вычислений.

При замерах времени выполнения алгоритма на первом этапе, было использовано семь примеров, исходные данные которых задавались в соответствии с правилом, описанном выше, и имеющие структуру, аналогичную той, что изображена на рисунке 12. Для самих замеров использовалась функция «clock()», возвращающая количество тактов (импульсов), прошедших с момента запуска программы. Разница полученных значений этой функции перед вычислениями и после них позволяет получить количество тактов, потраченных на вычисление алгоритма. Для получения результата в секундах, полученное значение делилось на значение макроса «CLOCKS\_PER\_SEC», возвращающего количество тактов в секунду. Результаты замеров в миллисекундах записаны в таблице 5.

Таблица 5. Результаты замеров времени выполнения алгоритма (Этап 1).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№** | **qn** | **nn** | **Замеры времени выполнения алгоритма, мс** | | | | | | | | | | |
| **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **Среднее** |
| 1 | 2 | 3 | 920 | 936 | 904 | 1232 | 1138 | 1201 | 949 | 1170 | 1348 | 1064 | 1086 |
| 2 | 3 | 4 | 1528 | 1326 | 1450 | 1482 | 1498 | 1529 | 1404 | 1482 | 1388 | 1544 | 1463 |
| 3 | 4 | 5 | 1685 | 1748 | 1653 | 1731 | 1810 | 1716 | 1794 | 1762 | 1934 | 1607 | 1744 |
| 4 | 5 | 6 | 2044 | 2293 | 1996 | 1965 | 1872 | 1966 | 1935 | 2317 | 2199 | 2059 | 2065 |
| 5 | 10 | 11 | 3026 | 3105 | 2870 | 2855 | 3058 | 3323 | 2948 | 2932 | 2824 | 2964 | 2991 |
| 6 | 15 | 16 | 4228 | 4257 | 4353 | 4430 | 4461 | 4119 | 4212 | 4478 | 4524 | 4243 | 4331 |
| 7 | 20 | 21 | 5476 | 5862 | 5731 | 5873 | 5803 | 5772 | 5506 | 5912 | 5850 | 5835 | 5762 |

где № - номер примера; qn – количество контуров; nn – количество узлов.

По полученным результатам был построен график зависимости времени выполнения алгоритма от количества контуров в сети (рисунок 13), который так же справедлив и для зависимости времени выполнения алгоритма от количества узлов в сети (с учетом, что в сети количество узлов на один больше (сервер), чем контуров).

Рисунок 13. Зависимость времени выполнения алгоритма от количества контуров (Этап 1).

Во время второго этапа время, затрачиваемое на вывод промежуточных данных, не учитывалось, поэтому при малом количестве узлов/контуров время вычисления получалось крайне низким, вплоть до микросекунд.   
Из-за этого функция, примененная на первом этапе, не подходила, так как не могла отобразить результат нужной точности (в микросекундах).

Для решения этой проблемы при замерах была использована функция «QueryPerformanceCounter()», связанная со счетчиком высокого разрешения и возвращающая количество импульсов, прошедших с момента запуска системы. Поскольку результат так же возвращается в тактах (тиках/импульсах), затрачиваемое время на вычисления определяется аналогично первому этапу, за тем исключением, что вместо «CLOCKS\_PER\_SEC» используется функция «QueryPerformanceFrequency()», возвращающая частоту высокоточного счетчика высокого разрешения (если он существует) в импульсах в секунду. Результаты замеров были записаны в таблицу 6.

Таблица 6. Результаты замеров времени выполнения алгоритма (Этап 2).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№** | **qn** | **nn** | **Замеры времени выполнения алгоритма, мс** | | | | | | | | | | |
| **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **Среднее** |
| 1 | 2 | 3 | 0,109 | 0,115 | 0,125 | 0,099 | 0,109 | 0,108 | 0,112 | 0,119 | 0,113 | 0,113 | 0,112 |
| 2 | 3 | 4 | 0,169 | 0,158 | 0,161 | 0,138 | 0,277 | 0,155 | 0,164 | 0,162 | 0,159 | 0,157 | 0,170 |
| 3 | 4 | 5 | 0,215 | 0,208 | 0,212 | 0,219 | 0,252 | 0,209 | 0,213 | 0,218 | 0,227 | 0,208 | 0,218 |
| 4 | 5 | 6 | 0,289 | 0,284 | 0,313 | 0,306 | 0,297 | 0,285 | 0,283 | 0,346 | 0,311 | 0,293 | 0,301 |
| 5 | 10 | 11 | 0,718 | 0,766 | 0,685 | 0,621 | 0,739 | 0,720 | 0,654 | 0,713 | 0,779 | 0,724 | 0,712 |
| 6 | 15 | 16 | 1,570 | 1,582 | 1,563 | 1,656 | 1,609 | 1,595 | 1,593 | 1,602 | 1,641 | 1,614 | 1,603 |
| 7 | 20 | 21 | 3,081 | 3,132 | 2,759 | 2,448 | 3,034 | 2,858 | 3,183 | 3,064 | 3,032 | 3,539 | 3,013 |
| 8 | 30 | 31 | 4,661 | 4,559 | 3,748 | 4,403 | 2,986 | 2,725 | 4,618 | 4,535 | 2,904 | 5,584 | 4,072 |
| 9 | 40 | 41 | 5,105 | 5,622 | 5,665 | 5,540 | 5,702 | 5,689 | 5,981 | 5,652 | 5,638 | 5,744 | 5,634 |
| 10 | 50 | 51 | 10,781 | 10,495 | 10,268 | 10,348 | 10,362 | 10,939 | 11,541 | 11,124 | 11,439 | 10,672 | 10,797 |
| 11 | 65 | 66 | 19,727 | 21,498 | 19,838 | 22,103 | 19,505 | 22,159 | 20,827 | 21,063 | 21,093 | 20,183 | 20,799 |
| 12 | 80 | 81 | 35,278 | 35,262 | 38,530 | 35,796 | 47,354 | 38,410 | 58,112 | 53,854 | 38,036 | 47,302 | 42,793 |
| 13 | 100 | 101 | 84,097 | 91,943 | 67,057 | 86,517 | 68,189 | 72,801 | 69,607 | 75,704 | 78,599 | 66,332 | 76,085 |
| 14 | 120 | 121 | 147,49 | 142,85 | 154,16 | 140,12 | 159,47 | 139,82 | 140,15 | 119,66 | 140,56 | 108,68 | 139,30 |
| 15 | 150 | 151 | 296,12 | 271,91 | 261,27 | 294,86 | 321,57 | 288,81 | 313,80 | 307,13 | 262,66 | 314,75 | 293,29 |
| 16 | 200 | 201 | 628,90 | 554,98 | 720,19 | 719,89 | 696,56 | 489,97 | 663,96 | 634,36 | 567,34 | 557,01 | 623,32 |
| 17 | 250 | 251 | 1388,8 | 1483,2 | 1558,3 | 1357,8 | 1428,7 | 1318,2 | 1505,3 | 1353,8 | 1324,3 | 1485,1 | 1420,4 |
| 18 | 300 | 301 | 3260,5 | 3513,8 | 3319,6 | 3302,2 | 3309,8 | 3308,8 | 3220,5 | 3322,1 | 3237,3 | 3276,6 | 3307,1 |
| 19 | 350 | 351 | 5443,0 | 5373,1 | 5305,6 | 5395,4 | 5408,6 | 5194,6 | 5273,8 | 5357,0 | 5237,7 | 5259,7 | 5324,8 |
| 20 | 400 | 401 | 9749,2 | 8164,7 | 8262,6 | 8182,5 | 8245,4 | 8524,7 | 8110,2 | 8234,3 | 8191,0 | 8262,5 | 8392,7 |
| 21 | 450 | 451 | 12011 | 11978 | 11927 | 12686 | 12068 | 11987 | 12210 | 12011 | 11895 | 11899 | 12067 |
| 22 | 500 | 501 | 18061 | 17032 | 16845 | 17186 | 17643 | 16864 | 16864 | 16932 | 16895 | 23031 | 17735 |
| 23 | 600 | 601 | 30577 | 31007 | 30384 | 30598 | 30801 | 30335 | 30519 | 30636 | 30388 | 30362 | 30561 |
| 24 | 700 | 701 | 50568 | 50608 | 50543 | 51158 | 50486 | 50899 | 50671 | 50696 | 50834 | 50721 | 50718 |
| 25 | 800 | 801 | 79544 | 79360 | 79738 | 79403 | 79867 | 80455 | 79873 | 79612 | 79916 | 80038 | 79781 |
| 26 | 900 | 901 | 119236 | 120557 | 121153 | 119478 | 118987 | 119457 | 119493 | 119894 | 120650 | 119301 | 119821 |
| 27 | 1000 | 1001 | 174458 | 173609 | 177158 | 175561 | 176984 | 174640 | 175728 | 175931 | 175149 | 174341 | 175356 |

По полученным результатам так же был построен график зависимости времени выполнения алгоритма от количества контуров в сети (рисунок 14).

Рисунок 14. Зависимость времени выполнения алгоритма от количества контуров (Этап 2).

Для определения ресурсов, необходимых алгоритму при вычислении различных исходных данных, были получены выделяемые ему объемы памяти, которые зафиксированы в таблице 7.

Таблица 7. Выделяемые объемы памяти и количество операций (Этап 2).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Пример** | **Объем выделяемой памяти, КБ** | | | |
| **Переменные**  **и массивы** | **Пиковый**  **рабочий**  **набор** | **Частный**  **рабочий**  **набор** | **Виртуальная**  **(Подкачка)** |
| 1 | 0,174 | 1520 | 232 | 296 |
| 2 | 0,251 | 1512 | 228 | 288 |
| 3 | 0,336 | 1516 | 232 | 292 |
| 4 | 0,429 | 1516 | 232 | 292 |
| 5 | 1,014 | 1516 | 232 | 296 |
| 6 | 1,799 | 1512 | 228 | 288 |
| 7 | 2,784 | 1516 | 228 | 292 |
| 8 | 5,354 | 1520 | 240 | 304 |
| 9 | 8,724 | 1528 | 244 | 304 |
| 10 | 12,894 | 1532 | 244 | 304 |
| 11 | 20,649 | 1544 | 260 | 320 |
| 12 | 30,204 | 1552 | 268 | 332 |
| 13 | 45,744 | 1576 | 292 | 352 |
| 14 | 64,484 | 1596 | 312 | 372 |
| 15 | 98,594 | 1636 | 352 | 416 |
| 16 | 171,444 | 1716 | 436 | 496 |
| 17 | 264,294 | 1820 | 536 | 600 |
| 18 | 377,144 | 1936 | 656 | 720 |
| 19 | 509,994 | 2076 | 792 | 852 |
| 20 | 662,844 | 2240 | 956 | 1016 |
| 21 | 835,694 | 2412 | 1128 | 1188 |
| 22 | 1028,544 | 2616 | 1324 | 1388 |
| 23 | 1474,244 | 3072 | 1776 | 1840 |
| 24 | 1999,944 | 3604 | 2308 | 2372 |
| 25 | 2605,644 | 4212 | 2924 | 2988 |
| 26 | 3291,344 | 4912 | 3616 | 3676 |
| 27 | 4057,044 | 5684 | 4384 | 4452 |

где:

* Переменные и массивы – объем выделяемой памяти для инициализации используемых в алгоритме переменных и массивов;
* Пиковый рабочий набор – максимальный объем памяти рабочего набора, использованный процессом. То есть занятое процессом количество памяти.
* Частный рабочий набор – показывает конкретный объем используемой процессом памяти, который данный процесс не может использовать совместно с другими процессами.
* Виртуальная (Подкачка) – объем виртуальной памяти (файла-подкачки), выделенной процессу.

Примечание: значение объема памяти для переменных и массивов было получено непосредственно в программе, путем замера размерности инициализированных переменных и массивов. Значения же для остальных трех столбцов были получены посредством “Диспетчера задач” Windows.

Под общим количеством шагов подразумевается суммарное значение операций пошаговой процедуры каждого контура и каждой итерации. Среднее количество шагов определялось как отношение общего количества шагов к произведению количества итераций и количества контуров, в результате чего были получены средние значения количества шагов для каждого контура.

По результатам таблицы 4 был построен график зависимости объема выделяемой памяти от количества контуров в сети (рисунок 15).

Рисунок 15. Зависимость объема выделяемой памяти от количества контуров (Этап 2).

где П – переменные и массивы; ПРН/ЧРН – пиковый/частный рабочие наборы; В – виртуальная память.

После вычислений была проведена полиномиальная аппроксимация для наиболее экономного по времени вычислений второго этапа, как для графика зависимости времени выполнения алгоритма от количества контуров в сети (рисунок 14), так и для графика зависимости объема выделенной памяти от количества контуров в сети (рисунок 15). В результате аппроксимации (рисунок 16 и 17) были получены аппроксимирующие функции. Для графика, изображенного на рисунке 14, результаты вычисления аппроксимирующих функций в степенях со второй по шестую, для значения количества контуров равного 700, записаны в таблице 8.

Таблица 8. Результаты вычисления аппроксимирующих функций для граф. 14.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **X** | **Степень** | | | | |
| **2** | **3** | **4** | **5** | **6** |
| 700 | 60909,7 | 65907,55 | 62487,5 | 57011,48 | 42406,28 |
| **Погрешность** | 10191,3 | 15189,15 | 11769,1 | 6293,076 | 8312,125 |

где: X – количество контуров; Погрешность – разница экспериментального значения и значения, полученного посредством аппроксимирующей функции.

Таким образом, видно, что наиболее близкой по значению является функция, полученная при аппроксимации в степени 5 (формула 12), так как она обладает наименьшей погрешностью.

(12)

График с аппроксимирующей линией функции (12), с прогнозом вперед на 200 периодов изображен на рисунке 16.

Рисунок 16. Полиномиальная аппроксимация степени 5 для количества времени выполнения алгоритма, с прогнозом на 200 периодов вперед.

Аналогичным образом были получены результаты вычисления аппроксимирующих функций для графика, изображенного на рисунке 15, которые записаны в таблице 9.

Таблица 9. Результаты вычисления аппроксимирующих функций для граф. 15.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **X** | **Степень** | | | | |
| **2** | **3** | **4** | **5** | **6** |
| 700 | 3587,84 | 3594,92 | 3617,09 | 3504,221 | 3499,359 |
| **Погрешность** | 16,16 | 9,08 | 13,09 | 99,779 | 104,6408 |

где: X – количество контуров; Погрешность – разница экспериментального значения и значения, полученного посредством аппроксимирующей функции.

Но в данном случае, в отличие от предыдущего, наиболее близкой по значению является функция, полученная при аппроксимации в степени 3 (формула 13).

(13)

График с аппроксимирующей линией функции (13), с прогнозом вперед на 200 периодов изображен на рисунке 17.

Рисунок 17. Полиномиальная аппроксимация степени 2 для объема  
выделенной памяти, с прогнозом на 200 периодов вперед.

Аппроксимация проводилась только по пиковому рабочему набору, т.к. остальные виды выделяемой памяти имеют идентичные кривые.

**Вывод:** При сравнении результатов замеров времени выполнения алгоритма (на первом и втором этапах) для первых семи примеров, можно заметить, что вывод промежуточных данных занимает довольно много времени, и при больших размерностях сети приводит к крайне длительному выполнению алгоритма. В связи с этим, при вычислении нелинейных уравнений для сетей больших размерностей, следует запускать выполнение алгоритма без вывода промежуточных данных, или с сохранением их в отдельный файл (что все же может занять какое-то дополнительное время). Стоит так же отметить, что производительность программы будет ещё зависеть и от структуры рассматриваемой сети (узлов и контуров), а так же от исходных данных. Кроме того была проведена аппроксимация графиков второго этапа, в результате чего были получены аппроксимирующие функции, позволяющие прогнозировать приближенные значения объема выделяемой памяти и времени выполнения алгоритма для заданного количества контуров.

**Литература**

1. **Абросимов Л.И.**

Базисные методы проектирования и анализа сетей ЭВМ: учебное пособие / Л.И. Абросимов. – М.: Университетская книга, 2015. – 248 с. – ISBN 978-5-98699-153-5.