



Munich Personal RePEc Archive

Parallel programming in MATLAB and its applications

Olenev, H.H.; Pechenkin, R.V. and Chernecov, A.M.

Dorodnicyn computing centre of the Russian academy of sciences

15. May 2007

Online at <http://mpa.ub.uni-muenchen.de/17796/>

MPRA Paper No. 17796, posted 10. October 2009 / 22:30

РОССИЙСКАЯ АКАДЕМИЯ НАУК
ВЫЧИСЛИТЕЛЬНЫЙ ЦЕНТР ИМ. А.А. ДОРОВНИЦЫНА

Н.Н. ОЛЕНЁВ, Р.В. ПЕЧЁНКИН, А.М. ЧЕРНЕЦОВ

**ПАРАЛЛЕЛЬНОЕ ПРОГРАММИРОВАНИЕ В
МАТЛАВ И ЕГО ПРИЛОЖЕНИЯ**



ВЫЧИСЛИТЕЛЬНЫЙ ЦЕНТР ИМ. А.А. ДОРОВНИЦЫНА РАН
МОСКВА ◊ 2007

УДК 512.643 + 519.86

Ответственный редактор
академик РАН А.А. Петров

Работа посвящена вопросам использования распределенных и параллельных вычислений в среде MATLAB. Изложена технология настройки кластера для использования MATLAB. Технология апробирована на стандартных задачах линейной алгебры. Предложен промышленный подход идентификации математических моделей сложных систем на основе параллельных средств MATLAB. Подход рассмотрен на примере простейшей динамической модели экономики современной России и позволяет освоить его большому числу потенциальных пользователей. Представлены результаты расчетов по двум возможным сценариям развития российской экономики.

Работа выполнена при частичной финансовой поддержке Российского фонда фундаментальных исследований (коды проектов 07-01-00563-а, 07-01-12032-офи), Российского гуманитарного научного фонда (код проекта 06-02-91821-а/G), программы Intel Education Project 2006-2007, программы Президиума РАН № 15, программы ОМН РАН № 3, по программе государственной поддержки ведущих научных школ (код проекта НШ-5379.2006.1).

Рецензенты Г.М. Михайлов,
В.С. Молоствов

Научное издание

©Вычислительный центр им. А.А. Дородницына
Российской академии наук, 2007

Оглавление

Введение	5
I. Администрирование и конфигурирование	10
1.1 Используемые термины и обозначения	10
1.2 Требования к аппаратному и программному обеспечению	15
1.3 Инсталляция и запуск MDCE	18
1.3.1 Работа в Windows	19
1.3.2 Работа в Unix	22
1.4 Запуск планировщика	23
1.4.1 Запуск в Windows	24
1.4.2 Запуск в Unix	24
1.5 Запуск рабочих процессов	25
1.6 Объекты-ссылки на системные процессы	28
1.7 Эквиваленты между функциями DCT и MPI	34
1.8 Использование сторонних планировщиков	35
1.9 Заключение	36
II. Программирование параллельных задач	38
2.1 Режим rmode, первая параллельная задача	39
2.2 Вычисление определенного интеграла	42

2.3	Объект "параллельная задача"	46
2.3.1	Объект "параллельное задание"	49
2.3.2	Матричное умножение	52
2.3.3	Решение СЛАУ методом Гаусса	57
III. Параллельные вычисления в моделировании экономики		67
3.1	Проблема идентификации внешних параметров модели экономики	69
3.2	Численная реализация задачи идентификации	85
3.2.1	Листинги программ	89
3.2.2	Результаты идентификации модели, их графическое представление	95
3.3	Сценарные расчеты с моделью	98
3.3.1	Базовый, он же пессимистический сценарий	99
3.3.2	Оптимистический сценарий	105
3.3.3	Сравнение, выводы	111
Заключение		113

Введение

Данная работа представляет собой первую попытку в виде научной монографии, доступной начинающему пользователю MATLAB, описать параллельную вычислительную технологию вычислений, реализованную компанией MathWorks с помощью двух взаимосвязанных пакетов расширений (приложений): MATLAB Distributed Computing toolbox [1] и MATLAB Distributed Computing Engine [2].

В настоящее время не только компания MathWorks обладает технологиями, позволяющими реализовывать параллельные вычисления. Следует упомянуть об аналогичных средствах, имеющихся в математических пакетах других фирм. Так, Maple, начиная с версии 9.5, имеет в своем составе toolbox HPC-Grid [3], который предлагает пользователю этой среды аналогичные возможности. Для пакета Mathematica имеется Parallel Computing Toolkit [4]. Также имеется независимый toolkit Pooch MPI [5], который работает только под операционной системой MacOS, что резко снижает его область применения. Часто применяемый пакет Mathcad на момент написания работы не имеет каких-либо средств, позволяющих выполнять распределенные вычисления.

Несмотря на наличие подобного рода инструментов в пакетах компьютерной алгебры Maple и Mathematica, среда

технических расчетов MATLAB является, по мнению авторов, более привычной и удобной для программирования, отладки и реализации параллельных алгоритмов.

За последние годы различными научными коллективами были созданы многочисленные пакеты расширений (Toolbox) MATLAB, реализующих какую-либо функциональность распределенных вычислений. С их обзором можно ознакомиться в [6]. Эти пакеты в своей реализации использовали различные способы (парадигмы):

- Использование модели передачи сообщений.
- Работа с общей памятью.
- Использование специальных процедур, с помощью которых производится обмен данных между последовательными сессиями MATLAB.
- Перекомпиляция кода MATLAB на C и создание параллельного кода уже для программ C.

В зависимости от реализации при использовании этих пакетов расширений требовалось иметь разное количество лицензий на запуск клиентских сессий MATLAB.

Пакеты расширений для параллельного программирования сторонними производителями начали разрабатываться в то время, когда необходимость в параллельном программировании для MATLAB не являлась критической. С тех пор в связи с необходимостью расчета более сложных задач эта необходимость резко возросла. Некоторые toolbox работали только под определенную операционную систему, чаще всего – под Unix. Поэтому использование единого подхода для всех операционных

систем стало необходимым. Большая часть разработанных пакетов уже не поддерживается. Особо следует отметить пакет MPITB [7], написанный в 2000 г. и получивший достаточно широкое распространение для кластерных систем. В нем реализовано применение функций `Message Passing Interface` – интерфейса передачи сообщений MPI – в программах MATLAB. Последняя на момент написания работы версия – для MATLAB 7.0.1 R14SP1 и библиотеки LAM MPI v. 7.1.1 [8] – датируется 2005 г. Основными его недостатками являются:

- Необходимость получения лицензии на каждое выполняемое ядро.
- Невозможность применения пакета для неоднородных вычислительных архитектур.
- Отсутствие официальной поддержки от компании `MathWorks`, постоянно развивающей систему MATLAB.
- Исключение работы пакета с операционными системами, отличающимися от ОС `Unix`.

Данный пакет был импортирован в среду `Windows` и реализацию MPICH2 [9]. Его дистрибутивы можно скачать с [10]. Последняя версия от 02.2006 работает с MATLAB v. 7.1.0.246 (R14) Service Pack 3, т.е. также с устаревшей версией MATLAB.

Учитывая все перечисленное выше, реализация подобных пакетов расширений уже самой фирмой `MathWorks` выглядит вполне логичным шагом.

Авторы ставили перед собой цель написать такую работу, которая могла бы значительно ускорить процесс освоения

технологии параллельного программирования читателем, который вообще не знаком ни с навыками параллельного программирования в системе MATLAB, ни в целом с системой.

В первом разделе описаны основные процедуры установки, настройки и конфигурирования кластерной части MATLAB Distributed Computing Engine. Этот раздел будет интересен в первую очередь системным администраторам и администраторам кластерных систем. В данном разделе авторы попытались представить информацию, содержащуюся "между строчек" в стандартной технической документации. Как показал опыт сотрудничества с другими исследовательскими группами, информации, представленной в этом разделе, достаточно для настройки высокопроизводительной вычислительной среды.

Во втором разделе описаны первоначальные сведения для создания параллельных программ с использованием средств библиотеки MPI и ее реализации на основе платформы MATLAB. Рассмотренные простые задачи позволяют пользователю убедиться в приросте производительности вычислений при увеличении числа процессоров и понять базовые принципы написания параллельных программ в MATLAB.

В третьем разделе авторы в качестве примера приложения параллельных вычислений в MATLAB для научных исследований рассматривают их применение в математическом моделировании экономических систем. Это приложение ни в коей мере не ограничивает применение параллельных вычислений в математическом моделировании сложных систем только данной областью науки. Во-первых, рассмотренные примеры легко могут быть обобщены на аналогичные задачи в других областях знаний. Во-вторых, экономические приложения к современной жизни интересны

для всех любознательных людей.

Рассмотрение приложений параллельных вычислений очень важно, поскольку в связи с развитием многопроцессорных и многоядерных [11] архитектур вычислительных систем в настоящее время параллельное программирование планируют применять повсеместно во всех областях науки, техники и бизнеса. В настоящее время фактическим стандартом параллельного программирования служит довольно сложный для первоначального изучения интерфейс передачи сообщений MPI – библиотека передачи сообщений, собрание функций, облегчающих коммуникацию (обмен данными и синхронизацию задач) между процессами параллельной программы с распределенной памятью [12, 13]. Качественные реализации MPI обеспечивают асинхронную коммуникацию, эффективное управление буфером сообщения, эффективные группы и богатые функциональные возможности. MPI включает большой набор коллективных операций коммуникации, виртуальных топологий и различных способов коммуникации и, кроме того, MPI поддерживает неоднородные сети.

Приложения, разработанные компанией MathWorks для создания параллельных и распределенных программ с использованием средств библиотеки MPI, и их реализация на платформе MATLAB упрощают практическое применение параллельных вычислений на многоядерных компьютерах, кластерах и GRID-системах.

Авторы выражают *искреннюю благодарность* департаменту MathWorks компании Softline за предоставленные дистрибутивы MATLAB и возможность ознакомиться и протестировать технологические возможности, заложенные в приложении MATLAB Distributing Computing Engine.

I. Администрирование и конфигурирование

Изучение настоящего раздела начинающим пользователем MATLAB даст возможность ощутить на собственном опыте преимущества решения практических задач с помощью распределенных вычислений на многопроцессорных, многоядерных или даже неоднородных архитектурах в сравнении с обычным (последовательным) вариантом решения этих задач при использовании одного компьютера.

1.1 Используемые термины и обозначения

Цель данного раздела — описать принципиальную схему работы двух Toolboxes MATLAB, которые реализуют технологию распределенных и параллельных вычислений (в дальнейшем, если не будет оговорено отдельно, под термином *распределенные вычисления* будут пониматься как параллельные, так и распределенные вычисления).

В 2005 г. для того, чтобы занять определенный сегмент рынка инженерного программного обеспечения, предназначенного для программирования распределенных задач,

компанией **MathWorks** были разработаны и выпущены на рынок два новых продукта под названием **Distributing Computing Toolbox (DCT)** и **MATLAB Distributing Computing Engine (MDCE)**. Эти два пакета расширений нельзя рассматривать отдельно друг от друга, и применяются они только в связке. Оба эти **toolbox** изначально были предназначены для решения одной задачи – увеличения производительности (сокращения времени счета) при использовании нескольких компьютеров, объединенных в сеть, вместо одного.

Для дальнейшего изложения потребуется дать несколько определений основным понятиям, которые встретятся нам на пути освоения параллельной технологии.

Client – сессия **MATLAB**, в которой определяются и из которой отправляются задачи. Это сессия **MATLAB**, в которой обычно работает разработчик– программист. Также распространено название **MATLAB-клиент**.

Client computer – компьютер, на котором запущен **MATLAB-клиент**.

Cluster – набор компьютеров, которые соединены в сеть с целью решать общую вычислительную задачу..

Head node – обычно это узел кластера, предназначенный для запуска **job manager** и **license manager**. Часто бывает полезным запускать все сервисные процессы для распределенных вычислений, не связанные с рабочими процессами, на одном компьютере.

Coarse-grained application – приложения, для которых время исполнения (счета) значительно больше, чем вре-

мя, требуемое для создания, отправки и получения данных расчетной задачи.

Distributed application – приложения, аналогичные coarse-grained application, запускаются независимо на различных узлах, возможно с различными входными параметрами. Не совершают обмен сообщениями, не имеют общих данных или точек синхронизации между узлами. Распределенные вычисления могут быть как coarse-grained, так и fine-grained (разветвленные задачи).

Distributed computing – вычисления, производимые с помощью распределенных приложений, запускаемых на нескольких узлах одновременно.

Job – полностью описанная вычислительная операция большой размерности для выполнения в MATLAB, состоящая из набора подзадач.

License Manager – компонент сервера лицензий, который отвечает за проверку и поддержку имеющихся лицензий на ядро/различные toolbox.

Task – подзадача, некоторый сегмент основной задачи. Именно подзадача отправляется на счет worker.

Mdce – служба, которая должна быть запущена на всех машинах, перед тем, как там будут запущены job manager или worker-процессы. Это основополагающая среда для запуска всех остальных процессов.

Job Manager – фирменный планировщик The MathWorks, это процесс, который образует систему очередей из задач (`job`) и распределяет подзадачи для `workers`. Под этим термином понимаются также планировщики, разрабатываемые сторонними фирмами производителями.

Worker – системный процесс MATLAB, который выполняет вычисление подзадач. Также распространенное название – MATLAB `worker` или `worker`-процесс.

Scheduler – планировщик. Основной функционал заключается в процессе планировки и управления очередями задач. В качестве планировщика может выступать MathWorks `job manager` или продукты сторонних фирм разработчиков.

По мере необходимости мы будем вводить дополнительные определения, пока введенных терминов должно быть достаточно, чтобы рассмотреть простейшую схему (см. 1.1), положенную в основу `Distributing Computing Toolbox`. При использовании промышленного кластера в качестве MATLAB `client` может выступать любой компьютер, подключенный к сети и имеющий доступ к кластеру, для `scheduler` обычно выделяют один из узлов кластера (в принципе, на этом же узле можно запустить и процесс `worker`, это не должно сильно сказаться на производительности), на всех остальных узлах кластера можно запустить по одному процессу. Запускать на одном узле несколько процессов имеет смысл только в случаях, когда узел многопроцессорный либо многоядерный.

Детали процесса запуска процессов `job manager` и `worker`, а также небольшие ухищрения (описанные в официальной документации [2]) мы рассмотрим, когда перейдем к описанию соответствующего этапа.

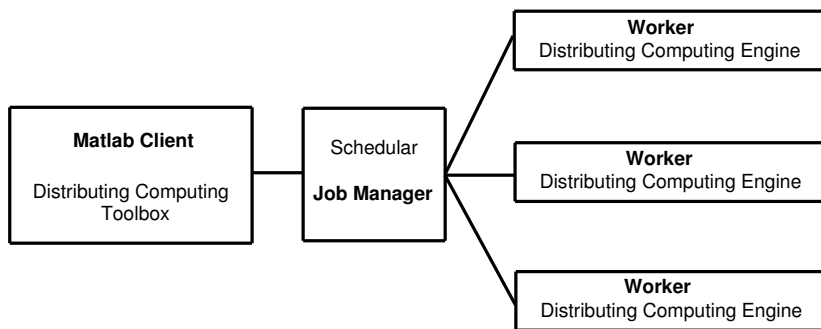


Рис. 1.1. Простейшая конфигурация, реализующая механизм распределенных вычислений MATLAB

Для первоначального знакомства с распределенной технологией — с объектами, их свойствами и методами, службами и сервисами — необязательно иметь в своем распоряжении промышленный кластер. Более того, для первоначального знакомства достаточно будет персонального компьютера, рабочей станции или ноутбука. Для того чтобы почувствовать прирост производительности, разумеется, необходим многопроцессорный комплекс, состоящий минимум из

двух компьютеров. В его роли может выступать как обычный многоядерный процессор, так и многопроцессорная система независимо от способа организации - кластер или SMP.

Для инсталляции и работы DST компьютер должен удовлетворять нескольким условиям. Во-первых, на нем возможно запустить MATLAB (разд. Требования к аппаратному и программному обеспечению). Во-вторых, накладываются определенные требования к сети (разд. Требования к сети).

1.2 Требования к аппаратному и программному обеспечению

Аппаратная платформа (общие требования):

- CD(DVD) дисковод (для инсталляции пакета).
- Для некоторых типов лицензий FlexLm Manager версии 10.8.0.1 (поставляется вместе с MATLAB).
- В случае использования Flexlm требуется применение протокола TCP/IP на всех платформах, использующих сервер лицензий.
- Для лицензий, использующих аппаратный ключ, требуется наличие USB-порта.

Минимальные требования к объему оперативной памяти и дисковому пространству изложены в таблице ниже.

Параметр	Минимум	Рекомендовано
Дисковое пространство	460 Мб	
Оперативная память	512 Мб	1 Гб

Минимальное дисковое пространство 460 Мб требуется для инсталляции ядра и справочной системы.

Для 32-битной архитектуры возможна работа на следующих микропроцессорах фирм Intel и AMD.

Фирма	Микропроцессор
Intel	Pentium III, Pentium 4, Pentium Xeon, Pentium M
AMD	Athlon, Athlon MP, XPAMD, Opteron

Для 64-битной архитектуры можно использовать микропроцессоры: у Intel: EM64T, у AMD: AMD64.

Операционная система:

- Для Windows - Windows 2000 SP3/XP SP1/2003/Vista.
- Для Linux: начиная с ядра версии 2.4.x и GLIBC 2.3.2.
- Для Solaris : Solaris 8,9,10 для SPARC/UltraSPARC.
- Для Macintosh: powerPC G4 - Mac OS X 10.4.7, intel - Mac OS X 10.4.8.

В данной работе применение **Solaris** и **Macintosh** не рассматривается по нескольким причинам:

- эти системы значительно менее распространены, чем **Windows** и **Linux**,
- для достижения поставленных целей — описания технологии распределенного программирования от компании **MathWorks** — вполне достаточно рассмотреть две системы,
- операции для **Solaris** аналогичны операциям для **Linux**.

Общие требования к аппаратной платформе и операционной системе определены. Это позволяет запускать ядро MATLAB. Для запуска `Distributing Computing Toolbox` накладываются также определенные требования к сети.

Требования к сети

- Распределенные вычислительные процессы должны быть способны определять друг друга по именам хостов, что требует наличия в сети службы разрешения имен (DNS, например). Кроме того, имя хоста, под которым компьютер виден остальным компьютерам сети, должно совпадать с именем хоста, под которым компьютер определяет сам себя.
- Имя хоста для каждого узла кластера должно ограничиваться IP-адресом, которому соответствует адрес одной из сетевых карт. Также поддерживаются машины с несколькими сетевыми картами.
- Рекомендуется минимум 5 ГБ дискового пространства для хранения рабочих файлов при использовании встроенного планировщика `JobManager`.
- Распределенные вычислительные процессы используют несколько TCP портов. Для узла, на котором запущено всего n `JobManager` и рабочих процессов, служба MDCE резервирует для собственного использования $5+n$ последовательных портов.
- Дополнительные TCP порты открываются для межпроцессного обмена между рабочими процессами во время выполнения параллельного задания.

- Чтобы запускать параллельные приложения, которые используют межпроцессный обмен, все узлы кластера, выполняющие приложение, должны иметь одинаковый размер слов и архитектуру команд процессора. Гетерогенные кластерные системы, на которых возможен запуск - Solaris и Macintosh. Системы Windows и Linux в версии MDCE 3.0 не поддерживаются.

Требования к MPI

MATLAB 2006b поставляется с реализацией MPI-2 [13] mpich2 v. 1.0.3. Вместо стандартной библиотеки можно подключить собственную реализацию. Для этого она должна удовлетворять следующим условиям:

- Быть собранной как динамическая библиотека.
- Поддерживать все функции MPI-1.
- Поддерживать пустые аргументы в MPI_Init, в соответствии с разд. 4.2 стандарта MPI-2.
- Иметь заголовочный файл mpi.h, полностью совместимый с mpich2.
- Для работы встроенного JobManager от MathWorks дополнительно требуется поддержка функций MPI-2 MPI_Open_port, MPI_Comm_accept, MPI_Comm_connect.

1.3 Инсталляция и запуск MDCE

Цель, которую мы ставим в данном разделе - это запустить на нашей рабочей станции службу, которая, в свою

очередь, позволит запустить процесс `job manager`. После запуска с его помощью рабочего процесса станет возможным на одном компьютере ознакомиться с основными свойствами и методами объектов.

Во-первых, необходимо удостовериться, что на вашем компьютере установлены соответствующие `toolbox`. Для этого, запустив `MATLAB`, в командном окне введите команду `>>ver`. В списке проинсталлированных `toolbox` обязательно должны быть следующие два `toolbox`:

```
Distributed Computing Toolbox
Version 3.0(R2006b),
MATLAB Distributed Computing Engine
Version 3.0(R2006b).
```

Остальные пакеты расширений в данный момент нас не интересуют.

В зависимости от того, как у вас установлен дистрибутив `MATLAB`, `C:\MATLAB\R2006B` или `C:\MATLABR2006B` или еще как-то, будем называть первую часть пути к директории `MATLAB` просто путь – `path`.

1.3.1 Работа в Windows

В каталоге `path\toolbox\distcomp\bin\` содержатся `*.bat` файлы, которые позволяют запускать служебные сервисы и процессы. Для запуска службы `mdce` пользователь должен либо в окне сеансе `cmd`, либо непосредственно в `MATLAB` запустить файл `mdce.bat install`.

Например, в приведенном примере, когда путь имел вид `C:\MATLAB\R2006B` или `C:\MATLABR2006B` запуск (инсталляция) службы `mdce` непосредственно из командной строки

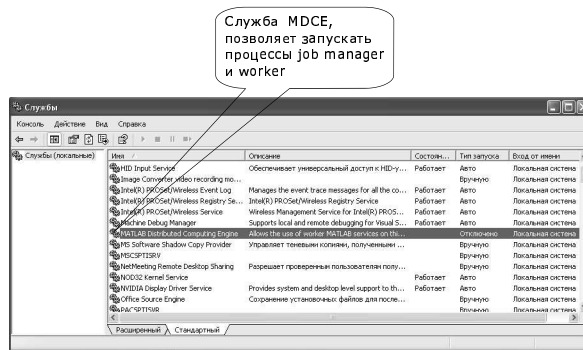


Рис. 1.2. Служба MDCE

MATLAB выглядела бы следующим образом:

```
!C:\MATLAB\R2006B\toolbox\distcomp\bin\mdce install
```

или

```
>>!C:\MATLAB\R2006B\toolbox\distcomp\bin\mdce install
```

знак ! указывает MATLAB, что выполняемая команда системная. В обоих случаях результат должен быть одним и тем же, в списке служб Windows должна появиться новая служба – MATLAB Distributed Computing Engine.

Аналогичный результат может быть получен с использованием функции `eval(command)`. В качестве параметра `command` необходимо передать строку, составив ее так, как если бы мы вводили команду в сеансе командной строки.

Корневой каталог установки MATLAB можно определить, например, следующим образом:

```
>>command1=matlabroot
command1 = C:\MATLAB15
```

Затем создадим переменную `command2`:

```
>> command2='\toolbox\distcomp\bin\'
```

Теперь склеим наши переменные:

```
>> command=['!' command1 command2 'mdce install'];
```

И передадим функции `eval` значение `command`

```
>> eval(command)
```

Аналогично будем поступать и в других случаях, когда требуется запустить другие `*.bat` файлы.

В том случае, если вам требуется получить быструю справку по какой-либо команде, достаточно ввести

```
>> help function_name
```

в этом случае будет выведен листинг help информации, расположенной в заголовке функции `command`, или ввести

```
>> doc function_name
```

в этом случае откроется страница помощи, на которой будет отображена более чем подробная информация по данной команде.

В том случае, если вы забыли, как точно называется требуемая функция, можно, набрав только первые буквы, нажать кнопку "Tab", и перед вами откроется ниспадающий список с функциями, начинающимися с введенных ранее букв. Теперь можно быстро,

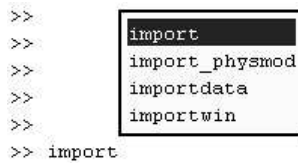


Рис. 1.3. Ниспадающая, быстрая подсказка

перемещаясь по списку, найти нужную команду (см. рис 1.3).

1.3.2 Работа в Unix

Инсталляция MDCE в Unix происходит практически аналогично с Windows. Настройки MDCE в отличие от Windows находятся в файле `mdce_def.sh`.

Лицензирование

Методы лицензирования в средах Windows и Unix отличаются. Если в Windows применяется пара лицензионный ключ + аппаратный ключ, то в Unix используется механизм лицензионных файлов и широко известный Flexlm Manager. В лицензионном файле, в частности, обязательно должны быть прописаны строки, относящиеся к Distributing Computing toolbox и MDCE, например:

```
INCREMENT Distrib_Computing_Toolbox MLM 16 10-feb-2007 0 \  
CDE6EOABF6779416F1F4 HOSTID=DEMO SN=DEMO  
# LicenseNo: DEMO HostID: <имя >  
INCREMENT TMW_Archive MLM 16 01-jan-0000 0 \  
CD86C100118CC4209A0F \  
VENDOR_STRING=908ff75f02aae4679ac61e7ccff HOSTID=DEMO SN=0
```

```
INCREMENT MATLAB_Distrib_Comp_Engine MLM 16 15-feb-2007 16 \  
9DE651D0F3651D1D608B HOSTID=ID=0 SN=DEMO
```

Инсталляция MDCE

Покажем последовательность действий для инсталляции mdce. Пусть MATLABROOT - корневой каталог, в который установлен MATLAB. Отметим, что при работе в конфигурации по умолчанию установка и запуск mdce требует прав суперпользователя root.

```
[root@broody bin]# cd $MATLABROOT/toolbox/distcomp/bin  
[root@broody bin]# ./mdce start  
Creating LOGBASE directory (/var/log/mdce).  
Creating CHECKPOINTBASE directory (/var/lib/mdce).  
Starting the MATLAB Distributed Computing Engine  
in the background.
```

Признаком нормальной работы службы mdce является следующее сообщение при запросе статуса:

```
[root@broody bin]# ./mdce status  
The MATLAB Distributed Computing Engine  
is running with PID 13592.  
Use nodestatus to obtain more information.
```

1.4 Запуск планировщика

Обратимся теперь ко второму этапу конфигурирования, а именно к процедуре запуска процесса Job Manager. С помощью файла `startjobmanager.bat`, который расположен в директории `MATLAB\toolbox\distcomp\bin\`, производится запуск системного процесса, который будет по сути нашим планировщиком. Запуск

можно произвести только в том случае, если служба `mdce` уже запущена.

Сначала рассмотрим использование встроенного планировщика от MathWorks - JobManager. Об использовании других планировщиков можно прочесть в разд. 1.7.

1.4.1 Запуск в Windows

Команде запуска `startjobmanager` можно передавать различные параметры с использованием ключей. Приведем описание только некоторых ключей. Полное описание пользователь может найти в стандартной документации.

-name – имя Job Manager, идентификатор. Если ключ `name` не задан, то процессу присвоится значение переменной `DEFAULT_JOB_MANAGER_NAME=default_jobmanager`, которое определено в файле `MATLAB\toolbox\distcomp\bin\mdce_def`. Пользователь может сам поменять значения переменных в этом файле на свое усмотрение.

-remotehost – имя удаленной рабочей станции, узла, на котором будет произведен запуск этого процесса. Обязательным требованием является запуск службы `mdce` на этой удаленной сессии. В качестве значения этого ключа можно передавать IP адрес компьютера (узла). Если ключ не используется, процесс будет запущен на локальной станции (с которой производится запуск).

-v – режим `verbose`, подробного листинга, во время запуска отображаются системные сообщения.

1.4.2 Запуск в Unix

При запуске в среде Unix синтаксис команды `startjobmanager` совпадает с запуском в Windows. В приведенном ниже примере

производится запуск JobManager с именем MyJobManager на локальной машине. Включен режим выдачи отладочной информации.

```
[root@broody bin]# ./startjobmanager -name MyJobManager -v
Contacting the mdce service on the host broody.ccas.ru
to start the job manager lookup process.
Started the job manager lookup process
on the host broody.ccas.ru.
Contacting the mdce service on the host broody.ccas.ru
to start a job manager process.
Started the job manager MyJobManager
on the host broody.ccas.ru.
```

1.5 Запуск рабочих процессов

Именно системные процессы, называемые workers или "удаленные сессии MATLAB", обеспечивают решение подзадач (task). Запуск worker аналогичен запуску процесса JobManager. С помощью файла `startworker.bat`, который так же, как и в предыдущем случае, запускается с использованием ключей

-name – имя MATLAB worker. Если ключ не задан, то имя по умолчанию это параметр `DEFAULT_WORKER_NAME` заданный в файле `mdce_def file`.

-jobmanager – значением этого ключа должно быть имя `jobmanager`, с которым будет ассоциирован (от которого будет получать подзадачи) запускаемый процесс worker. По умолчанию присваивается значение параметра `DEFAULT_JOB_MANAGER_NAME`, которое задано в файле `mdce_def file`.

-jobmanagerhost – IP адрес или имя компьютера на котором запущен процесс `jobmanager`.

-remotehost – имя удаленного компьютера (IP адрес) на котором должен быть запущен данный процесс `worker`.

-v – режим `verbose`, подробного листинга, во время запуска отображаются системные сообщения.

В нашем примере необходимо выполнить команду

```
>>!C:\MATLAB\R2006B\toolbox\distcomp\bin
\startworker -name worker1 -remotehost mylaptop
-jobmanager MyJobManager -jobmanagerhost mylaptop,
```

после чего будет запущен процесс с именем `worker1`.

После того как в сети (на кластере) запущен хотя бы один процесс `jobmanager`, можно использовать команду `nodestatus`. Данная команда позволяет получать информацию с различной детализацией о запущенных на конкретном узле (`node`):

- процессах,
- Job Manager,
- службах MDCE.

Синтаксис этой команды очень прост. Имеется два ключа для передачи параметров этой процедуре:

-remotehost – имя удаленного компьютера (IP адрес), на котором должен быть запущен данный процесс `worker`,

-infolevel – уровень информативности, 1 - минимальная информация, 3 - подробная.

Выполнив команду

```
>>!C:\MATLAB\R2006B\toolbox\distcomp\bin\nodestatus  
-remotehost mylaptop -infolevel 3
```

при одном запущенном `jobmanager` и одном процессе `worker`, ассоциированном с этим планировщиком, вы увидите следующее сообщение

```
Job manager  
lookup process:  
  
Job manager:  
Status Running  
Name MyJobManager  
Running on host mylaptop  
Number of workers 1  
  
Worker:  
Name worker1  
Running on host mylaptop  
Status Idle  
Job manager MyJobManager  
Connection with  
job manager Connected  
  
Summary:  
The mdce service  
on mylaptop manages  
processes:  
Job manager  
lookup processes 1  
Job managers 1  
Workers 1
```

Поменяв значение `infolevel` на 3, пользователь может увидеть более детальную информацию о процессах, которые запущены на узле. Выводимой информации достаточно, чтобы выявить некорректные моменты в работе кластера, например выявить запущенные рабочие процессы, не ассоциированные с `JobManager`.

Для того чтобы завершить создание нашего минимального кластера, нам необходимо запустить процесс `worker` на втором компьютере, который соединен через сеть с первым (служба `mdce` должна быть запущена и на этом компьютере тоже). Пусть имя второго `worker` будет `worker2`. Для запуска `worker2` на втором компьютере, нужно в команде `startworker` в качестве ключа `remotehost` указать имя второго компьютера (пусть имя компьютера будет `mydesktop`).

Теперь можно считать, что минимальный кластер создан.

1.6 Объекты-ссылки на системные процессы

Итак, как уже было сказано, наши процессы `jobmanager` и `worker` живут отдельно от клиентской сессии `MATLAB`. При остановленном клиентском приложении `MATLAB` эти процессы тем не менее продолжают существовать в системе.

Для управления этими процессами из клиентской сессии `MATLAB`, необходимо получить ссылку на эти системные процессы. В этом случае в рабочей области `MATLAB` мы получим переменную – объект распределенных вычислений – и с помощью ее свойств и методов сможем управлять этими объектами.

Ссылка на `jobmanager`

Для поиска ссылки на процесс `jobmanager` необходимо воспользоваться функцией `findResource` (данная функция принадлежит `MATLAB Distributing Computing toolbox`) в качестве параметров передать тип искомого процесса и имя узла.

```
jm = findResource('scheduler','type','jobmanager',...  
                 'Name','MyJobManager','LookupURL','mynote')
```

Результатом выполнения данной команды будет переменная `jm` в

текущей области MATLAB.

```
>> jm
    jm =
    distcomp.jobmanager
>>
```

Посмотреть свойства объекта можно, как и в обычном случае, воспользовавшись графическим режимом, дважды щелкнув курсором по объекту `jm` или воспользовавшись командой `get(jm)`, которая является предопределенной для всех классов системы MATLAB.

```
>>get(jm)
      Name: 'MyJobManager'
      Hostname: 'mynote'
      HostAddress: {0x1 cell}
      Jobs: [0x1 double]
      State: 'running'
      Configuration: ''
      NumberOfBusyWorkers: 0
      BusyWorkers: [0x1 double]
      NumberOfIdleWorkers: 1
      IdleWorkers: [1x1 distcomp.worker]
>>
```

Вообще говоря, методы, которыми обладает какой-либо объект, можно узнать с помощью команды `method`. Так, например, передав в качестве параметра объект `jm`, мы можем узнать, какими методами он обладает.

```
>> methods(jm)
Methods for class distcomp.jobmanager:
```

```
createJob          demote  pause   resume
createParallelJob findJob  promote
>>
```

Ссылка на рабочий процесс worker

Аналогично с помощью функции `findResource` мы можем найти ссылку на системный процесс исполнителя - `worker`.

```
>> worker = findResource('worker', 'LookupURL', 'mynote')
worker =
distcomp.worker

>> get(worker)
      Name: 'worker1'
      Hostname: 'mynote'
      HostAddress: {0x1 cell}
      State: 'running'
      JobManager: [1x1 distcomp.jobmanager]
      CurrentJob: []
      CurrentTask: []
      PreviousJob: []
      PreviousTask: []
>>
```

Вообще говоря, получить объект `worker` можно и с помощью объекта `jobmanager`. Так, у объекта (переменной) `jm` есть свойство `IdleWorkers` (число свободных рабочих процессов), значение которого в нашем случае `[1x1 distcomp.worker]`. Для того чтобы получить объект `worker`, обратимся через точку к соответствующему свойству

```
>> w=jm.IdleWorkers
```

```

w =
  distcomp.worker

>> get(w)
      Name: 'worker1'
      Hostname: 'mynote'
      HostAddress: {0x1 cell}
      State: 'running'
      JobManager: [1x1 distcomp.jobmanager]
      CurrentJob: []
      CurrentTask: []
      PreviousJob: []
      PreviousTask: []
>>

```

В данном случае переменная `w` будет равносильна переменной `worker` – обе эти переменные представляют собой ссылки на один и тот же системный процесс `worker`.

Следует заметить, что эти объекты – ссылки на системные процессы – обладают занимательным свойством, а именно: **размер этих переменных в байтах равен нулю.**

```

>> whos jm
      Name      Size      Bytes  Class
      jm         1x1                distcomp.jobmanager
Grand total is 1 element using 0 bytes

>> whos worker
      Name      Size      Bytes  Class
      worker    1x1                distcomp.worker
Grand total is 1 element using 0 bytes

```

Несмотря на то, что сам системный процесс `jobmanager` использует 512 Мб оперативной памяти, `JOB_MANAGER_MAXIMUM_`

MEMORY = 512 m - как это определено в файле `mdce_def.bat`, процессу `worker` доступно всего 60 Мб. Каким образом можно изменить этот параметр - размер процессов, можно узнать, воспользовавшись стандартной документацией [2]. В нашем сквозном примере эти параметры изменяться не будут.

В том случае, если требуется программно запустить несколько рабочих процессов на нескольких узлах, можно написать цикл, в котором будет происходить запуск соответствующего процесса `worker` на соответствующем узле. Причем текст команды будет формироваться (склеиваться) программно.

```
mroot=matlabroot;
% путь установки MATLAB
catalog='\toolbox\distcomp\bin\win32\';
% каталог, в котором установлен toolbox
mdcepath=[mroot catalog]
% соединяем две переменные (склеиваем строку)
nodes={['ip1'},{'ip2'},{'ip3'},{'ip4'},{'ip5'}];
% массив ячеек - каждая из которых IP адрес или имя узла
MyJobManager = 'MyJobManager';
% имя уже запущенного jobmanager
MyJMhost      = 'mynote';
% имя узла, на котором запущен jobmanager
%%
for i=1:length(nodes)
% цикл для запуска worker на соответствующем узле
name=['worker' num2str(i)];
% уникальное имя worker
command=['!' mdcepath 'startworker -name ' name...
        ' -remotehost ' nodes{i}...
        ' -jobmanager ' MyJobManager...
        ' -jobmanagerhost ' MyJMhost ' -v']
% склеили команду, которая запустит i-й процесс
```

```
eval(command)
% функция eval выполняет команду,
% которую ей передали в виде строки - входного параметра,
end
```

Результатом работы этого участка кода являются запущенные процессы на соответствующих узлах. Аналогичный код можно написать, например, и для команд `stopworker`, `nodestatus`.

Дисковые затраты

Служба MDCE, а также планировщик и рабочие процессы `worker`, генерируют различные рабочие файлы в процессе своей работы. Рабочие файлы бывают двух типов: файлы журнала и файлы контрольных точек (checkpoints).

ОС	Логи (LOGBASE)	Контрольные точки (CHECKBASE)
Windows	"TEMP\MDCE\log"	TEMP\MDCE\Checkpoint
Unix	/var/log/mdce	/var/lib/mdce

Еще раз отметим, что согласно данным производителя, для хранения файлов на жестком диске необходимо иметь не менее 5 Гбайт свободного места в рабочих каталогах. Также следует иметь в виду, что после сбоя работа рабочих процессов и `jobmanager` начнется с момента последней контрольной точки. Чтобы этого избежать, необходимо использовать при запуске ключ `-clean`.

В процессе работы размер рабочего каталога (`/var/lib/mdce` или `TEMP\MDCE\Checkpoint`) стремительно растет. Чтобы избежать переполнения диска, мы рекомендуем выполнять после каждого запуска уничтожение созданного задания через функцию `destroy(job)`.

1.7 Эквиваленты между функциями DCT и MPI

В приложении Distributed Computing toolbox (DCT) системы MATLAB версии 2006b имеется встроенный набор функций для передачи сообщений между процессами, основанный на интерфейсе передачи сообщений Message Passing Interface (MPI). Эквиваленты между функциями DCT системы MATLAB и библиотеки MPI [12] для языка программирования C/C++ приведены в табл. 1.1.

Таблица 1.1. Сравнение основных функций передачи сообщений в DCT MATLAB с соответствующими функциями в MPI

Функция/ переменная MATLAB	Функция/ переменная MPI в C/C++
mpiInit	MPI_Init(int *argc, char ***argv)
numlabs	MPI_Comm_size(MPI_COMM_WORLD, int size)
labindex	MPI_Comm_rank(MPI_COMM_WORLD, int rank)
LabBarrier()	MPI_Barrier(MPI_COMM_WORLD)
shared_data=... LabBroadcast(root,buffer)	MPI_Bcast(void* buffer, int count, MPI_Datatype datatype, int root, MPI_COMM_WORLD)
LabSend(buf,dest) LabSend(data,dest,tag)	MPI_Send(void *buf, int count, MPI_Datatype datatype, int dest, int tag, MPI_COMM_WORLD)
data=LabReceive(source) data=LabReceive('any',tag) data=LabReceive(source,tag) [data,source,tag]=LabResceive(.)	MPI_Recv(void *buf,int count, MPI_Datatype datatype,int source, int tag, MPI_COMM_WORLD, MPI_Status *status)
is_data_available=... LabProbe(source,tag)	MPI_Probe(int source,int tag, MPI_COMM_WORLD,MPI_Status *status)
received=labSendReceive(... labTo,labFrom,data) received = labSendReceive(... labTo,labFrom,data,tag)	MPI_Sendrecv(void *sendbuf, int sendcount, MPI_Datatype sendtype, int dest, int sendtag, void *recvbuf, int recvcount, MPI_Datatype recvtype, int source, int recvtag, MPI_COMM_WORD, MPI_Status *status)
mpiFinalize	MPI_Finalize()

Из табл. 1.1 можно заметить, что в отличие от MPI в при-

ложении DCT системы MATLAB пока не предусмотрена возможность работы с различными коммутаторами, разделяющими процессы на группы. Здесь имеется только один коммутатор, который включает все процессы. Соответствующий коммутатор в MPI называется MPI_COMM_WORLD.

Механизм тегирования – использование меток (тегов) для разбиения сообщений по типам – в MATLAB не является обязательным, однако он представлен целочисленными тегами в диапазоне от нуля до 32767. Если теги явно не указывать, то MATLAB самостоятельно присваивает всем рабочим процессам тег, равный нулю.

Функция `mpigateway` системы MATLAB дает возможность входа во внутреннюю функциональность MPI. Функция `mpiLibConf` позволяет локализовать реализацию MPI, которая будет использоваться. Функция `mpiSettings(option, varargin)` может использоваться для смены стандартных опций MATLAB для работы с коммуникациями MPI.

Отметим, что часто используемая в параллельном программировании операция `Sendrecv` была реализована только в MDCE версии 3.0. Также реализованы такие функции, как `MPI_Reduce` с операцией `MPI_SUM` и подобные ей.

За более подробной информацией можно обратиться к стандартной документации по `Distributed Computing Toolbox` [1].

1.8 Использование сторонних планировщиков

MATLAB `Distributing Computing Engine` поддерживает следующие планировщики: встроенный от MathWorks (`JobManager`), `Windows CCS` - планировщик от Microsoft для `Windows Server 2003 Cluster Edition`, `LSF`, `mpiexec` либо планировщик общего вида. Под планировщиком общего вида может выступать, например, планировщик системы `PBS` [14] или `Sun Grid Engine` [15]. Среди преимуществ встроенного планировщика от MathWorks можно выделить:

- поддержку гетерогенности,
- поддержку контрольных точек,
- обеспечение автоматической доставки данных в процессы (свойство `FileDependencies`).

Основным его и *весьма существенным* недостатком является невозможность совместного использования с существующими на многопроцессорном комплексе системами управления заданиями.

1.9 Заключение

Вообще говоря, перечисленные процедуры (запуск службы `mdce`, запуск процессов `jobmanager` и `worker`) должен выполнять администратор кластера, а не тот пользователь-программист, который будет заниматься программированием самих вычислительных задач. Для программиста не имеет значения архитектура кластера, ему не нужно знать имена рабочих процессов, имена узлов, на которых они запущены и т.д. Единственный объект, который нужен программисту-исследователю – это ссылка на объект планировщика – `JobManager`. Этого будет достаточно, чтобы создать объекты задач, подзадач, чтобы отправить задачу на счет.

Тем не менее материал, изложенный в данном разделе, сам по себе является достаточно интересным, чтобы его рассматривать отдельно.

Замечания

Следует заметить также, что при использовании распределенных вычислений необходимо обеспечить идентичное наличие на каждом исполнительном узле как всех необходимых функций из разных `toolbox`, так и пользовательских файлов. Что касается

`toolbox`, то лицензия на MDCE автоматически предполагает возможность установки всех возможных `toolbox` на узел.

Сделать это для пользовательских файлов можно тремя путями:

- либо пользователь самостоятельно рассылает все необходимые файлы в соответствующие места,
- либо пользователь располагает файлы в общей для сервера и узлов папке; в этом случае для **Windows** имеет смысл сделать общую папку, а для **Linux** использовать файловую систему **NFS**,
- либо пользователь использует механизм встроенного планировщика **JobManager** и **MATLAB** самостоятельно занимается распределением файлов (при отправке задания, **MATLAB** автоматически упаковывает рабочие файлы в **zip**-архив и распаковывает его в соответствующих сессиях рабочих процессов).

При реализации изложенных пунктов могут возникнуть разного рода проблемы, когда невозможно запустить тот или иной процесс. Для того чтобы избежать большинства этих проблем, требуется выполнение следующих условий:

- Пользователь имеет права администратора на том компьютере, на котором происходит запуск соответствующего процесса.
- Брандмауэр операционной системы должен быть либо отключен, либо следует произвести соответствующую ему настройку.
- В среде **Windows** желательно минимизировать количество антивирусных программ, а лучше всего их вообще отключить.

II. Программирование параллельных задач

После изучения настоящего раздела начинающий пользователь MATLAB сможет провести требуемые процедуры для создания распределенных и параллельных задач и понять, как описываются сами объекты – задачи.

Минимальное оборудование, необходимое для выполнения примеров из данного раздела – один компьютер с установленным на нем MATLAB Distributing Computing Toolbox, процесс jobmanager и один рабочий процесс. Этого будет достаточно, чтобы освоить технологию создания задачи, ее отправки на обсчет и возврата результатов.

В заключение раздела приведены примеры, которые требуют уже двух запущенных рабочих процессов. Эти два рабочих процесса могут быть, вообще говоря, запущены на одном и том же компьютере. Однако для того чтобы ощутить прирост производительности при использовании двух рабочих процессов, они должны быть запущены на разных компьютерах, либо в том случае, если у Вас многоядерный или многопроцессорный компьютер, вы можете запустить их на одной и той же машине, при этом в момент запуска автоматически, без дополнительных ключей, они будут запущены на разных ядрах или процессорах.

2.1 Режим `pmode`, первая параллельная задача

В MATLAB в настоящее время существует два подхода для решения параллельных задач. Первый подход основан непосредственно на процедуре отправки задания `jobmanager`, в инструкциях (`m` - файле) которой описана последовательность команд, которая будут выполняться рабочими процессами. В этом `m`-файле помимо основных команд MATLAB могут быть использованы функции MPI для коммуникаций между рабочими процессами. Второй подход для решения параллельных задач основан на режиме `pmode`. С помощью этого режима непосредственно из командного окна MATLAB становится возможным обращение к процессам `workers`, просмотр их локальных переменных, обмен данными между ними. В режиме `pmode` команды, вводимые в рабочем окне MATLAB, будут исполняться всеми рабочими процессами, ассоциированными с соответствующим `jobmanager`.

Режим `pmode`, по мнению авторов, следует использовать исключительно с двумя целями: как удобный пользовательский режим, предназначенный для первоначального знакомства с элементами параллельного программирования, для понимания многопроцессорной архитектуры и парадигмы параллельного программирования и как средство отладки параллельных программ.

В предыдущем разделе уже был рассмотрен один из способов получения ссылки на планировщика. Второй способ поиска ссылки на планировщика заключается в использовании свойства `'configuration'`, которое есть описание того планировщика, ссылку на который необходимо найти. Под описанием, имеются в виду следующие свойства: минимальное и максимальное количество рабочих процессов, `FileDependencies`, `LookupURL` и т.д. Пользователь может создать (описать) несколько свойств непосредственно в `m`-файле

`matlabroot/toolbox/distcomp/user/distcompUserConfig.m`, дав им соответствующие имена, и в дальнейшем осуществлять поиск ссылки на планировщик, используя только свойство `'configuration'`, а не имя самого планировщика и узла, на котором он запущен.

Команда `jm = findResource('scheduler','configuration','myconfig')` осуществляет поиск ссылки на системный процесс `jobmanager`, обладающего свойством `'myconfig'`.

Пользователь может включить режим `pmode`, введя команду:

```
>> pmode start conf numlabs
```

где `conf` - имя конкретной конфигурации планировщика, `numlabs` - количество рабочих процессов, которые должны быть запущены. Другие варианты включения режима `pmode` описаны в [1]. Итак, команда `pmode start myconfig 2` включает режим `pmode`, ассоциирует его с планировщиком, описанным в свойстве `myconfig`, при этом используя только два рабочих процесса. Следует отметить, что при исполнении этой команды ссылка на планировщик в рабочем пространстве `MATLAB` не появится.

После включения режима `pmode` обычный указатель на текущую строку `MATLAB` поменяет представление с привычного `">"` на `"P>>"`. Следующие две функции `MPI` будут рассмотрены в первую очередь:

labindex – уникальный идентификатор, порядковый номер рабочего процесса. `labindex` принимает значение от 1 до `n`, где `n` – общее число рабочих процессов, доступных в данной сессии (ассоциированных с данным планировщиком),

numlabs – общее число рабочих процессов, ассоциированных с данным планировщиком.

При вводе в командной строке команды `labindex` будет получен следующий ответ системы:

```
P>> labindex
1: ans =
1:      1
2: ans =
2:      2
P>>
```

В данном случае в рабочем окне отображен ответ системы так, как если бы команда `labindex` была введена (выполнена) непосредственно в каждом из процессов `workers`.

Введя команду `numlabs`, очевидно, мы получим следующий ответ системы:

```
P>> numlabs
1: ans =
1:      2
2: ans =
2:      2
P>>
```

Иными словами, находясь в любой из двух сессий рабочих процессов, можно определить, сколько всего процессов доступно в данный момент.

Создадим теперь переменную `a`, которая будет определена в сессиях (рабочих пространствах) всех процессов. Пусть переменная `a` будет принимать значение ID процесса, т.е. `labindex`.

```
P>> a=labindex
1: a =
1:      1
2: a =
2:      2
P>> whos
```

```

1:  Name      Size  Bytes  Class      Attributes
1:
1:  a         1x1      8  double
1:  ans       1x1      8  double
2:  Name      Size  Bytes  Class      Attributes
2:
2:  a         1x1      8  double
2:  ans       1x1      8  double
P>>

```

Командой `whos` выводится список всех переменных, которые определены в текущей сессии. В обоих процессах в данном примере определена только переменная `a`.

Следует заметить, что в сессии рабочих процессов доступны только функции тех `toolbox`, которые установлены на клиентской части (на том компьютере, где производится запуск параллельной задачи). Вообще говоря, в силу того, что любой рабочий процесс есть не что иное, как системный процесс `MATLAB`, ассоциированный с программой `MATLAB`, установленной на соответствующем узле, в его сессии доступны все функции всех пакетов расширений, в том числе и `SIMULINK`.

2.2 Вычисление определенного интеграла

Информации, представленной в предыдущем разделе, достаточно для решения первой "классической" параллельной задачи - вычисления значения числа π . Как известно, значение числа π равно значению определенного интеграла:

$$\int_0^1 \frac{4}{1+x^2} dx = \pi.$$

Чтобы решить эту задачу, проинтегрировать функцию

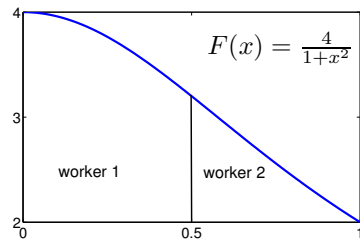


Рис. 2.1. Схематичный рисунок к задаче вычисления числа π

$F(x) = \frac{4}{1+x^2}$, следует воспользоваться тем свойством, что каждый из доступных рабочих процессов может интегрировать функцию $F(x)$ на своем участке интервала (схематически это проиллюстрировано на рис. 2.1).

Очевидно, что первым шагом решения задачи должно быть распространение функции $F(x)$ среди всех рабочих процессов. Вводим команду

```
P>> F = @(x) 4./(1 + x.^2)
1: F =
1:    @(x) 4./(1 + x.^2)
2: F =
2:    @(x) 4./(1 + x.^2)
P>>
```

Теперь функция $F(x)$ становится переменной, определенной в каждом рабочем процессе.

Все, что осталось сделать перед тем, как запустить задачу вычисления интеграла, – необходимо определить границы интегрирования для каждого рабочего процесса. Пусть **a** и **b** – переменные, которые определяют границы интегрирования для каждого рабочего процесса. Вводим команду

```
P>> a = (labindex - 1)/numlabs;  
P>> b = labindex/numlabs;
```

Эти переменные будут определены в сессиях каждого рабочего процесса, причем значения их различны.

```
P>> [a, b]  
1: ans =  
1:      0      0.5000  
2: ans =  
2: 0.5000  0.1000
```

Теперь, воспользовавшись функцией `quadl(F, a, b)`, в параметры которой передаются подынтегральная функция и граница интегрирования, вычислим значение определенного интеграла в каждом из рабочих процессов.

```
P>> myIntegral = quadl(F, a, b)  
1: myIntegral =  
1:      1.8546  
2: myIntegral =  
2:      1.2870
```

Теперь, после того как в каждой сессии определена переменная `myIntegral`, необходимо воспользоваться функцией `gplus` и произвести глобальное суммирование переменной `myIntegral` по всем процессам.

gplus – `gplus(x)` возвращает сумму значений переменной `x` по всем процессам. Результат тиражируется на всех сессиях процессов.

```
P>> piApprox = gplus(myIntegral)  
1: piApprox =
```

```
1:      3.1416
2: piApprox =
2:      3.1416
```

Если требуется, переменную, хранящуюся в сессии рабочего процесса, можно экспортировать в текущую (пользовательскую) сессию MATLAB. Предварительно нужно переключиться из режима `pmode` в обычный режим MATLAB с помощью команды `suspend`. Команда `lab2client` позволяет копировать определенную переменную из любого процесса в пользовательскую сессию.

```
P>> pmode suspend
>> pmode lab2client piApprox 1
```

Итогом рассмотренной задачи является переменная `piApprox`, определенная в локальной сессии MATLAB. Для сравнения полученного значения со значением константы `pi`, которое хранится в системе MATLAB, рассмотрим разность `piApprox - pi`

```
>> piApprox - pi
ans =
    2.4866e-010
```

Если пользователю после работы в локальной сессии необходимо снова переключиться в режим `pmode`, то это можно сделать, введя команду

```
>> pmode resume
P>>
```

Замечания

Суммируя выше сказанное в данном разделе, можно сделать следующие выводы:

- Режим `pmode` работает без ссылки на `jobmanager` в клиентской сессии.
- Существует возможность копировать переменные из клиентской сессии в сессии рабочих процессов и наоборот.
- Рабочие процессы могут выполнять все функции, лицензия на которые есть в клиентской части.
- Рассматривать режим `pmode` следует исключительно как средство для первоначального знакомства с элементами параллельного программирования в `MATLAB` и как отладочное средство.

2.3 Объект "параллельная задача"

В данном разделе будут описана технология создания параллельных программ с использованием `m`-файлов. Данная технология существенно отличается от режима `pmode` и позволяет пользователю при написании параллельных программ использовать редактор кода `MATLAB`. При этом подходе становится возможным программировать в клиентской сессии `MATLAB` параллельные программы практически любой сложности.

Перейдем непосредственно к описанию технологии создания параллельных задач. Одним из методов объекта планировщика (`jm`, `jobmanager`) является метод `createParallelJob`¹. Методом `createParallelJob` можно воспользоваться двумя способами: передав функции `createParallelJob` в качестве

¹методы любого объекта можно узнать, введя команду `methods(var)`, где `var` - переменная `MATLAB`.

входного параметра переменную-ссылку на объект планировщик `pjob=createParallelJob(jm)`, либо через операцию "точка", обратившись напрямую к этому методу `pjob=jm.createParallelJob`. Результаты выполнения обеих команд должны быть идентичными.

```
>> pjob = jm.createParallelJob
    pjob =
        distcomp.paralleljob
>>
```

Переменная `pjob` является ссылочной переменной и ссылается на определенный участок оперативной памяти, выделенной для системного процесса планировщика. По умолчанию в объекте `pjob` заполнены следующие свойства:

```
>> get(pjob)
        Name: 'Job1'
          ID: 1
    UserName: 'your_name'
         Tag: ''
        State: 'pending'
RestartWorker: 0
        Timeout: Inf
MaximumNumberOfWorkers: Inf
MinimumNumberOfWorkers: 1
        CreateTime: 'Mon Dec 18 15:20:36 MSK 2006'
        SubmitTime: ''
        StartTime: ''
        FinishTime: ''
          Tasks: [0x1 double]
FileDependencies: {0x1 cell}
PathDependencies: {0x1 cell}
```

```
JobData: []
Parent: [1x1 distcomp.jobmanager]
UserData: []
QueuedFcn: []
RunningFcn: []
FinishedFcn: []
Configuration: ''
```

Смысл большинства свойств понятен по названию, с детальным описанием всех свойств можно ознакомиться в [1]. Обратимся для начала к свойствам `MaximumNumberOfWorkers` и `MinimumNumberOfWorkers`.

В силу того, что задача будет отправляться на вычисление через планировщик `jm` (у которого в свойствах указано, сколькими занятыми и сколькими свободными рабочими процессами он обладает в данный момент времени), уже в самой задаче (в объекте `pjob`) должно быть указано, сколько свободных процессов требуется планировщику для начала вычислений, описанных в задаче `pjob`.

Значение свойств `MaximumNumberOfWorkers` и `MinimumNumberOfWorkers` есть положительное целое число. Установим его равным двум, т.е. вычисления в задаче `pjob` начнутся, как только у планировщика `jm` появятся два свободных рабочих процесса.

Записывать команды, которые устанавливают свойствам определенные значения, можно как в командном окне, так и непосредственно в `m`-файле, что является более удобным на наш взгляд. Напомним что с версии MATLAB R2006 редактор кода позволяет выделять ячейки с помощью двух символов `%`. Так, например, участок кода

```
%%
set(pjob,'MinimumNumberOfWorkers',2)
set(pjob,'MaximumNumberOfWorkers',2)
%%
```

устанавливает нужные нам значения свойств. Для проверки значения свойств можно воспользоваться командой `get` и вернуть значение свойства, например выполнив `get(pjob, 'MinimumNumberOfWorkers')`.

2.3.1 Объект "параллельное задание"

В данном разделе описывается, пожалуй, один из самых важных объектов в рассматриваемой нами архитектуре, а именно непосредственно параллельное задание. По сути параллельное задание есть не что иное, как `m`-файл, который будет исполняться одновременно всеми рабочими процессами. Передать `m`-файл напрямую рабочему процессу нельзя, для этой цели необходимо использовать объект - `Task` (параллельное задание). Объект `Task` - является дочерним объектом по отношению к объекту `distcomp.paralleljob` и может быть создан посредством метода `createTask`.

Создадим `m`-файл и опишем в нем функцию (процедуру), которая будет выполняться каждым процессом (по сути эта процедура состоит из команд, которые были введены при описании режима `pmode`).

При написании этого `m`-файла следует постоянно помнить о том, что данный `m`-файл будет выполняться одновременно несколькими процессами. Переменные, которые будут определены в этом `m`-файле, будут локальными для каждой сессии процесса, к ним не будет открыт доступ от других процессов без использования соответствующих функций MATLAB для передачи сообщений между процессами.

Следующий участок кода представляет собой `m`-файл (функцию), который состоит из уже описанных команд:

```

function piapprox=par_pi(F,a,b)
% инициализация
a=(labindex - 1)/numlabs; b=labindex/numlabs;
% вычисление интеграла
myInt=quadl(F,a,b);
% суммирование по всем workers
piapprox=gplus(myInt);

```

После того как написали m-файл, следует указать в свойстве `FileDependencies` объекта `pjob` имя файла, который будет выполняться рабочими процессами (вместо имени файла может быть указана и директория, если задача требует создания дополнительных m-файлов)

```

%%
set(pjob, 'FileDependencies', {'par_pi.m'});
%%

```

После этого создаем переменную, которая будет являться подзадачей. Во входных параметрах требуется указать переменную - параллельную задачу, количество выходных аргументов (в рассматриваемом примере - 1) и входные аргументы `{F,a,b}` (переменные `F,a,b` должны уже быть определены в локальной сессии MATLAB):

```

%%
obj = createTask(pjob,'par_pi',1,{F,a,b})
%%

```

Теперь все, что осталось сделать - это выполнить следующий участок кода:

```

%%
submit(pjob); waitForState(pjob);
%%

```

Здесь команда `submit` отправляет задачу `pjob` планировщику `jobmanager`. Задача начинает решаться рабочими процессами не сразу, а только тогда, когда свободными окажутся столько процессов, сколько указано в свойстве `MinimumNumberOfWorkers`. До этого момента задача находится в очереди. Как только количества свободных процессов оказалось достаточно, задача `pjob` начинает "считаться" (иными словами, рабочие процессы начинают выполнять один и тот же `m`-файл). В этот момент свойство `State` объекта `pjob` принимает значение `running`. Свойство меняет свое значение на `finished` только после того, как задача, описанная в `m`-файле, выполнится. Как только задача поменяла статус на `finished`, можно воспользоваться методом `getAllOutputArguments` и получить данные в локальную сессию `MATLAB`.

```
%%  
results = getAllOutputArguments(pjob)  
%%
```

Следует отдельно отметить, что при отправке задачи планировщик предварительно упаковывает в `zip`-архив файлы, указанные в свойстве `FileDependencies`, после чего отправляет его всем процессам, где распаковывает его в локальных сессиях, что гарантирует наличие исполняемого `m`-файла у всех рабочих процессов.

В целом `m`-файл, в котором описаны основные шаги для создания `m`-файла (программы на языке `MATLAB`, которая непосредственно отправляет задачу планировщику и возвращает результаты расчета) выглядит следующим образом.

```
%% Параллельная программа для вычисления числа pi  
jm = findResource('scheduler', 'type', 'jobmanager', ...
```

```

        'Name', 'MyJobManager')
%% Создание параллельной задачи
pjob = jm.createParallelJob;
%%
P=2;
set(pjob, 'MinimumNumberOfWorkers', P);
set(pjob, 'MaximumNumberOfWorkers', P);
%%
set(pjob, 'FileDependencies', {'par_pi.m'});
%%
F = @(x) 4./(1 + x.^2);
a=0;
b=1;
%% Создание параллельной подзадачи
obj = createTask(pjob, 'par_pi', 1, {F, a, b});
%%
submit(pjob); waitForState(pjob);
%%
results = getAllOutputArguments(pjob)
%%

```

В дальнейшем мы не будем столь подробно останавливаться на этой последовательности, а обратимся непосредственно к самому важному, на наш взгляд, моменту, а именно к написанию m-файлов с использованием функций передачи сообщений (тех m-файлов (функций на языке MATLAB), которые исполняются непосредственно рабочими процессами).

2.3.2 Матричное умножение

В данном разделе будет описана широко известная задача параллельного матричного умножения. Фактически будет описано, каким образом, используя несколько процессов, можно

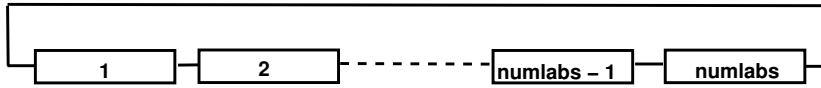


Рис. 2.2. Кольцо из numlabs процессоров

решить следующую задачу:

$$z = y + Ax, \quad A \in \mathbb{R}^{n \times n}, x, y, z \in \mathbb{R}^n. \quad (2.1)$$

Данная задача подробно описана в разд. 6.1. [16]. Здесь мы остановимся на основных моментах, связанных с ней. По сути, технология, предлагаемая при использовании **Distributing Computing toolbox** и **MATLAB Distributing Computing Engine**, есть система с распределенной памятью (так как каждый рабочий процесс обладает своим адресным пространством). Схематично эта система (топология) представлена на рис. 2.2.

На начальной стадии перед началом вычислений в каждой сессии (в каждом рабочем пространстве процесса) доступен соответствующий блок матрицы A , вектора x и вектора y .

$$\begin{bmatrix} z_1 \\ \vdots \\ z_p \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_p \end{bmatrix} + \begin{bmatrix} A_{11} & \cdots & A_{1p} \\ \vdots & & \vdots \\ A_{p1} & \cdots & A_{pp} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_p \end{bmatrix}$$

Вычисления начинаются с того, что каждый рабочий процесс отправляет доступный ему блок вектора x соседнему (по номеру ID) процессу. Следующим шагом является получение с помощью функции `labReceive` блока вектора x от соседнего рабочего процесса, который к этому моменту уже отправил его. После получения блока x происходит непосредственное вычисление вектора

$y_k = y_k + A_{p,\tau}x_\tau$, где τ – номер доступного блока вектора x (на каждом шаге, у каждого процесса рабочего процесса определена своя переменная τ). Схематично на примере трех процессов перемещение подвектора x_k представлено в таб. 2.1.

Таблица 2.1. Перемещение подвектора x_k для трех процессов

Шаг	worker1	worker2	worker3
1	x_3	x_1	x_2
2	x_2	x_3	x_1
3	x_1	x_2	x_3

Ниже приведен листинг программы (m-файла), выполняемой каждым рабочим процессом. В этой программе переменная τ определяет номер текущего доступного блока вектора x .

Функция параллельного матричного умножения

```
function z=par_multiplication(A,x,y)
% A - матрица
% x,y,z вектор-столбцы
% z=Ax+y;
q=mod(n,numlabs);% остаток от деления
if q==0
    r=n/numlabs;
    x=x((labindex-1)*r+1:labindex*r);
    y=y((labindex-1)*r+1:labindex*r);
    A=A((labindex-1)*r+1:labindex*r,:);
else % делится с остатком
    r=(n-q)/numlabs;
    if labindex==1
        x=x(1:r+q);
        y=y(1:r+q);
```

```

        A=A(1:r+q,:);
    else
        x=x((labindex-1)*r+1+q:labindex*r+q);
        y=y((labindex-1)*r+1+q:labindex*r+q);
        A=A((labindex-1)*r+1+q:labindex*r+q,:);
    end
end
for t=1:numlabs
    if labindex==numlabs
        labSend(x,1); % последний сразу отправил первому
    else
        labSend(x,labindex+1);
    % остальные отправляют своим соседям (справа)
    end
    if labindex==1
        x=labReceive(numlabs);
    % первый получил от последнего
    else
        x=labReceive(labindex-1);
    % остальные получают от своих соседей (слева)
    end
    tau=labindex-t;
    if tau<=0
        tau=tau+numlabs;
    end
    y=y+A(:,1+(tau-1)*r:tau*r)*x;
end
z=y;

```

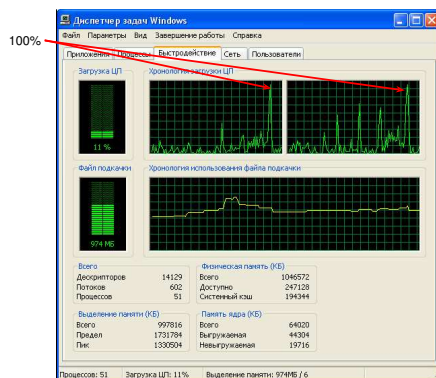


Рис. 2.3. Загрузка двухъядерного процессора при решении задачи (2.1), $N = 2000$

Вычислительные эксперименты

Как видно из рис. 2.3, при решении задачи (2.1) с помощью функции `par_multiplication`, оба ядра становятся загруженными на 100%. Данная задача не показывает на самом деле прирост производительности (при рассмотренном размере задачи, по крайней мере), но является показательной в плане понимания того, как можно применять распараллеливание задачи на несколько процессоров, даже для такой казалась бы простой задачи, как матричное умножение. Помимо этого, данная задача позволяет убедиться в том, что при использовании `Distributing Computing Toolbox` становится возможным достичь пиковой производительности всех используемых процессов (процессоров).

Вычислительная задача, параллельное решение которой дает прирост производительности, рассмотрена в разд. 2.3.3.

2.3.3 Решение СЛАУ методом Гаусса

Существует большое количество методов решения систем линейных алгебраических уравнений (СЛАУ), в данной работе рассмотрен метод решения СЛАУ, заключающийся в последовательном исключении (метод с выбором главного элемента не рассматривался).

Именно с помощью этой задачи экспериментально выявлена эффективность использования параллельного метода решения СЛАУ, реализованная с помощью средств **MATLAB Distributed Computing toolbox**. В качестве тестовых систем первоначально был использован ноутбук с двухядерным процессором Intel Duo Core T2300 1.66 ГГц. Для тестирования **MATLAB** в промышленном режиме, данная задача (решение СЛАУ) была запущена на кластере Вычислительного центра им. А.А. Дородницына Российской академии наук (ВЦ РАН) <http://www.ccas.ru>.

Кратко приведем описание использованного программного и аппаратного обеспечения. Система состоит из восьми двухпроцессорных узлов Intel Xeon/2.6 ГГц, соединенных высокоскоростным межсоединением Myrinet 2000 (пиковая скорость передачи 2 Гбит/с). Используемая операционная система: Redhat 9.0 (версия ядра 2.4). Более подробно с данными о кластере можно ознакомиться в [17, 18]. В процессе экспериментов выяснилось, что для ядра 2.4 версии **MATLAB 2006 (A, B)** имеются ошибки при реализации одной из встроенных библиотек. Для корректной работы потребовалось загрузить с помощью **Internet** и установить соответствующее обновление. Более подробно об этой ошибке можно узнать в **Internet** по адресу [19]. Насколько известно авторам, эта ошибка уже исправлена в версии **MATLAB 2007A**.

Опишем реализованный параллельный метод решения СЛАУ. Как известно, при последовательном решении системы с помощью метода последовательного исключения используется алгоритм, листинг которого приведен ниже. Данный алгоритм хорошо из-

вестен из курса линейной алгебры.

Функция последовательного исключения

```
function A=gauss(A)
[m n]=size(A);
for i=1:n
    A(i,:)=A(i,+)/A(i,i);
    for k=i+1:n
        A(k,:)=A(i,:)*A(k,i)-A(k,);
    end
end
```

Подход, лежащий в основе параллельного метода решения представленной задачи, заключается в том, что на начальной стадии (инициализации), вся матрица A расположена в рабочем пространстве первого процесса (назовем его **мастер**)². Первый шаг - первый процесс (**мастер**) отправляет равные слои матрицы A всем остальным процессам (назовем их **рабочими**). Выделение соответствующего слоя матрицы A реализуется с помощью функции `par_getslice`.

После отправки процесс **мастер** нормирует первый элемент в первой строке своей подматрицы и сразу же отправляет ее всем остальным. **Рабочие**, получив данную строку, начинают приводить к нулевому столбцу соответствующий столбец своей подматрицы. К тому моменту, как **мастер** завершил приводить

²Довольно часто при описании работы параллельных алгоритмов используются термины **мастер** и **рабочие**. Так, например, **мастером** называют рабочий процесс, который на стадии инициализации производит массовую рассылку заданий, а на стадии окончания работы алгоритма производит сбор данных от всех процессов, которые называют **рабочими**.

к верхнедиагональному виду свою подматрицу, **рабочие** также завершили получать от него соответствующие строки. После этого **рабочие** отправляют оставшиеся ненулевые подматрицы первому процессу **мастеру**. В итоге после первого шага в рабочем пространстве **мастера** хранится верхнедиагональная, прямоугольная подматрица и квадратная ненулевая подматрица (полученная из подматриц, присланных **рабочими**). Следующий шаг заключается в том, что **мастер** рассылает (делит) соответствующие подматрицы ненулевой квадратной матрицы всем процессам **рабочим**, включая и себя. Алгоритм продолжает работать до тех пор, пока размер отправляемого слоя квадратной матрицы не станет равным единице.

В результате после последнего шага алгоритма в рабочем пространстве **мастера** будет храниться верхнедиагональная матрица U с единичной диагональю. Именно эта матрица и будет считаться нами результатом работы параллельного алгоритма гауссового исключения, листинг которого приведен ниже.

Процедура выделения из матрицы определенного слоя

```
function [A q r]=par_getslice(A,labindex,numlabs)
[n n]=size(A); % размер матрицы
q=mod(n,numlabs);% остаток от деления
if q==0 % делится без остатка
    r=n/numlabs; A=A((labindex-1)*r+1:labindex*r,:);
else % делится с остатком
    r=(n-q)/numlabs;
    if labindex==1
        A=A(1:r+q,:);
    else
        A=A((labindex-1)*r+1+q:labindex*r+q,:);
    end
end
end
```

Описанный метод можно модифицировать таким образом, чтобы на конечном шаге алгоритма матрица U была распределена между всеми процессами, но данный вариант процедуры в данной работе не приводится.

Ниже приведен листинг параллельной программы гауссового исключения, которая выполняется каждым процессом.

Параллельная программа гауссового исключения

```
function U=par_gauss(A) [M N]=size(A);
r=0;s=0;
while r=1      s=s+1; % счетчик
    if labindex==1
        [m n]=size(A);      % размер матрицы
        % на второй итерации матрица A обновляется,
        % уменьшается в размере,
        % склеивается из подматриц, вычисленных рабочими
        qq=mod(m,numlabs); % остаток от деления
        for otherLab = 2:numlabs
            % отправка соответствующих слоев матрицы A
            labSend(qq,otherLab,6); % остаток от деления
            [Asend q r]=par_getslice(A,otherLab,numlabs);
            labSend(Asend,otherLab,3);
        end
        [A q r]=par_getslice(A,labindex,numlabs);
        % мастер получил свой блок матрицы A
        [m n]=size(A);
        for i=1:m
            % данный цикл приводит подматрицу в мастере
            % к верхнетриangularному виду
            A(i,:)=A(i,+)/A(i,i);
        % строка A(i,:) должна быть отправлена всем рабочим
```

```

        for otherLab = 2:numlabs
            labSend(A(i,:),otherLab,1);% строка
            labSend(i,otherLab,2);    % номер строки
        end
% получение нулевых элементов под элементом A(i,i)
        for k=i+1:m
            A(k,:)=A(i,:)*A(k,i)-A(k,:);
        end
    end
    % A - верхнетридиагональная подматрица,
    %     вычисленная мастером
    if s==1 % первый шаг
        U=A;
    else
        U=[U;zeros(m,N-n) A];
    end
    % склейка новой матрицы A, каждая подматрица
    % которой вычислена соответствующим рабочим
    A=[];
    for otherLab = 2:numlabs
        Arecive=labReceive(otherLab,4);
        A=[A;Arecive];
    end
    if (r==1 && qq==0)
        U=[U;zeros(1,N-r) 1];
    end
else
    A=labReceive(1,3); % подматрица
    q=labReceive(1,6); %
    [m n]=size(A);
    r=m;
    for i=1:m+q
        Arow=labReceive(1,1); %строка

```

```

        j=labReceive(1,2);      %индекс
        for k=1:m
            A(k,:)=Arow*A(k,j)-A(k,:);
        end
    end
end
A=A(:,j+1:n); % отправка мастеру подлежат
              % только ненулевые столбцы
labSend(A,1,4);
U=[];
end

```

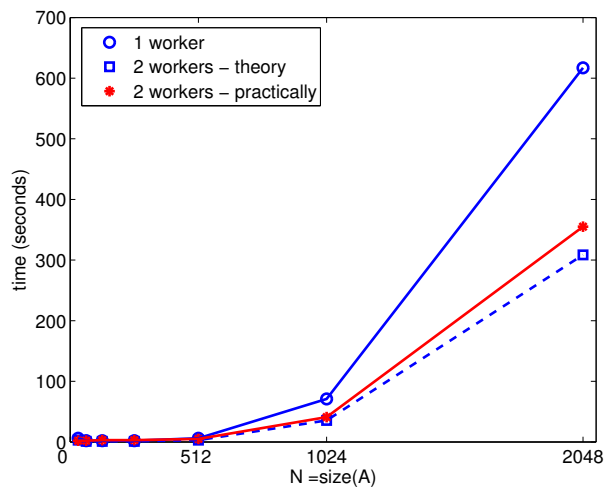


Рис. 2.4. Графики сравнительной эффективности при использовании одного и двух ядер процессора, график линейного ускорения (при отсутствии накладных расходов транспортировки)

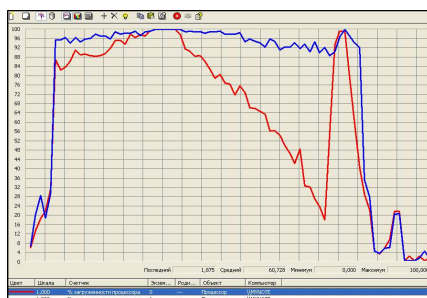


Рис. 2.5. Графики загрузки двух ядер процессора при решении задачи гауссового исключения с $N = 2048$ на двух рабочих процессах. Графики получены в консоли производительности

Вычислительные эксперименты

Для исследования эффективности параллельной реализации гауссового исключения были произведены запуски алгоритма параллельного последовательного исключения при различных размерах матрицы A . Для первого эксперимента был использован двухъядерный процессор (с двумя запущенными рабочими процессами), результаты тестирования алгоритма `par_gauss` представлены в виде графиков на рис. 2.4, 2.5.

Как и следовало ожидать, из-за наличия накладных расходов, связанных с пересылкой данных между рабочими процессами, линейного ускорения при использовании двух процессов достичь не удалось (см. маркеры \square и $*$ на рис. 2.4). Тем не менее прирост производительности при использовании двух рабочих процессов явно есть, о чем свидетельствует график с маркером $*$.

Рис. 2.5 представлен исключительно для иллюстрации того, что при использовании двух процессов оба ядра достигают пико-

вой загрузки. Спад интенсивности загрузки одного из ядер вполне объясним, так как с увеличением числа итераций уменьшается объем пересылаемых данных.

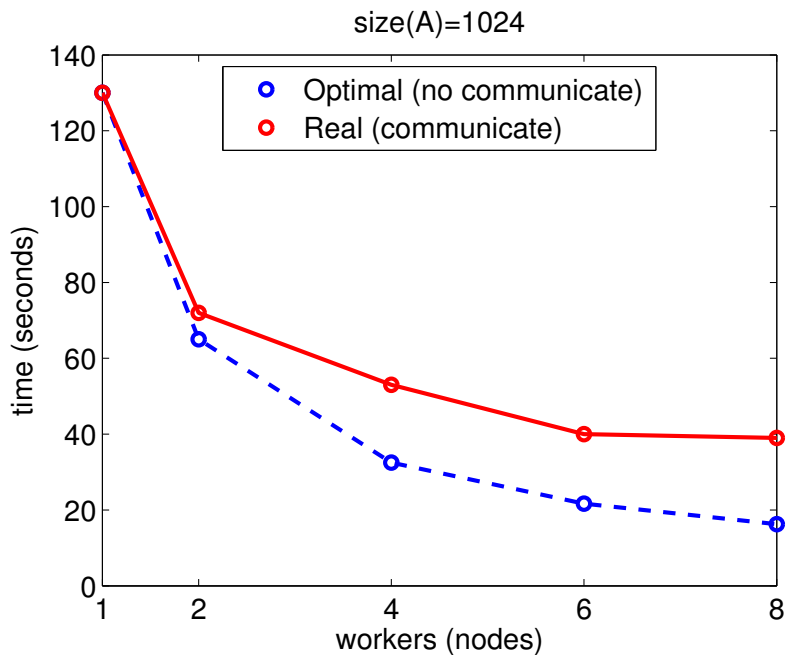


Рис. 2.6. Графики оптимального (без коммуникаций) и реального (с коммуникациями) времени решения задачи гауссового исключения на кластере ВЦ РАН, $N = 1024$

Дополнительно в качестве вычислительного теста был произведен запуск рассмотренной задачи при $N = 1024$ на кластере ВЦ РАН. Напомним, что при использовании встроенной реализации MPI обмен данными происходит с использованием протокола TCP/IP в обычной сети на базе FastEthernet 100 Мбит/с.

Результаты представлены на рис. 2.6-2.8.

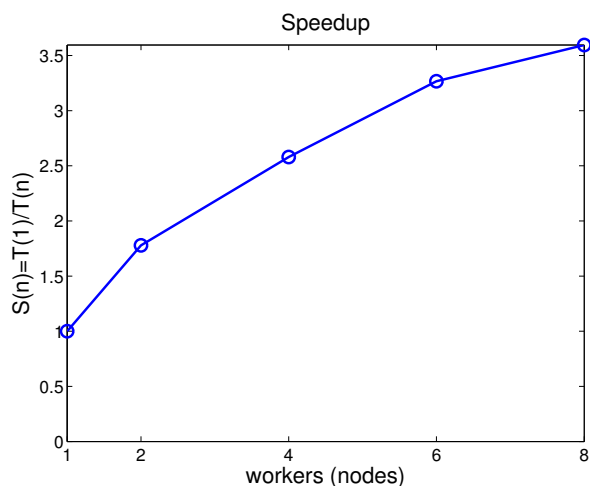


Рис. 2.7. Достигнутое ускорение

Анализ результатов

На рис. 2.7, 2.8 приведены результаты решения задачи на 2, 4, 6 и 8 процессорах соответственно. Анализируя график, видно, что при работе свыше 4 процессоров эффективность падает ниже 50 %, что для данной задачи является не лучшим показателем.

Это может быть связано со следующими причинами:

- плохая реализация библиотеки MPI с использованием протокола TCP/IP и сети на базе FastEthernet,
- декомпозиция задачи на слишком мелкие подзадачи,
- реализация части кода MATLAB с использованием Java, что резко повышает время расчета.

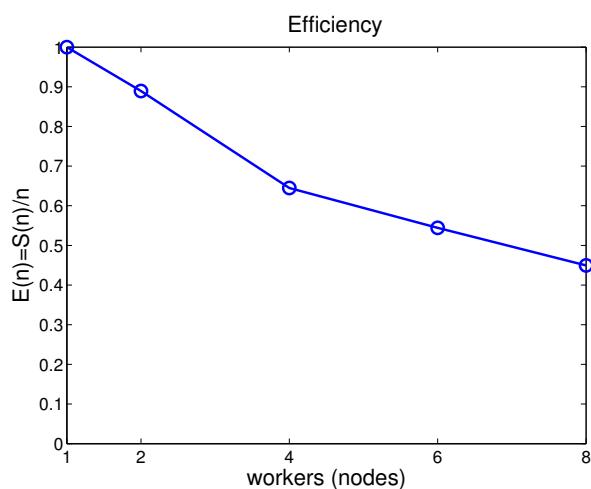


Рис. 2.8. Достигнутая эффективность

В заключение разд. II следует отметить, что технология параллельных вычислений, реализованная в **MATLAB Distributed Computing toolbox**, позволяет создавать не только параллельные программы, предназначенные для решения задач линейной алгебры, но и решать вычислительные задачи, требующие больших вычислительных мощностей, возникающие в различных областях науки и техники (имитационное моделирование методом Монте-Карло, оптимизация портфеля инвестиций, содержащего большое количество инструментов, идентификация параметров динамической системы, расчет глобальных биогеохимических циклов элементов (например, азота и углерода) для пространственно - распределенных моделей экосистем и т.д.).

Об одном из таких классов задач пойдет речь в следующем разделе.

III. Параллельные вычисления в моделировании экономики

Можно применять параллельные вычисления для решения традиционных задач математического моделирования сложных систем, например таких, как ускорение расчетов при проведении численных экспериментов с моделью экономики страны или региона, а можно и не применять, если в общем времени построения модели непосредственный расчет составляет относительно небольшое время и совершается редко. Нужно применять параллельные вычисления для решения таких задач, которые другим способом решить либо невозможно, либо чрезвычайно трудно. Это расширяет границы области научных исследований и приложений в бизнесе, в которых эффективно применять методы математического моделирования.

Начинающий пользователь MATLAB, освоивший из предыдущих глав технологию распределенных вычислений в MATLAB, после изучения данной главы сможет применить данную технологию для решения практических задач по определению параметров математических моделей сложных систем, возникающих в различных областях науки и бизнеса.

Идентификация сложной математической модели той или иной экономической системы, заключающаяся в определении внешних параметров этой модели, является одной из основных

задач, где параллельные вычисления позволяют находить решения, которые без применения параллельных вычислений было невозможно найти точно. Проблемы, возникающие при решении этой задачи, будут здесь рассмотрены на примере специально построенной простейшей эконометрической динамической модели современной российской экономики.

Идентификация модели заключается в определении ее внешних параметров на основе исторических данных. Часть параметров оценивают прямым образом на основе имеющихся статистических данных. Большую часть параметров оценивают косвенным образом, сравнивая расчетные временные ряды показателей модели с их статистическими аналогами.

Огромный возможный набор сочетаний значений параметров не позволял до появления высокопроизводительных параллельных архитектур точно определять эти значения. Использование высокопроизводительных параллельных вычислений благодаря достаточно полному перебору позволяет точно решить задачу идентификации параметров сложных математических моделей экономических систем. Дело в том, что в моделях экономики имеется немало параметров, которые не удастся найти напрямую из данных экономической статистики. В случае, когда данных статистики хватает, качество исходных статистических данных, как правило, таково, что их хватает только для определения интервалов, в которые попадают параметры модели. Кроме того, и начальные значения некоторых переменных модели часто точно неизвестны и поэтому должны рассматриваться как такого рода параметры.

Будем определять неизвестные параметры экономической модели косвенным образом, сравнивая рассчитанные с помощью модели выходные временные ряды ее переменных с доступными статистическими временными рядами для этих переменных в изучаемой экономической системе. Материалы исследования излагаются таким образом, чтобы их легко можно было обобщить для

использования в задачах идентификации более сложных моделей экономических систем и вообще в задачах идентификации математических моделей сложных процессов и систем.

3.1 Проблема идентификации внешних параметров модели экономики

Рассмотрим простейшую модель экономики страны, чтобы на ней продемонстрировать все те основные задачи, которые приходится решать исследователю при идентификации внешних параметров математических моделей экономики с помощью параллельных вычислений. Эта модель будет несколько отличаться от предложенной в 1928 г. Фрэнком Пламптоном Рамсеем, английским математиком, философом и экономистом, стандартной замкнутой простейшей модели экономики страны [20], варианты которой представлены во многих учебниках по математической экономике [21-24] при теоретическом анализе процессов распределения национального дохода на потребление и накопление. Учтем в модели внешнеторговый оборот страны и изменение относительных цен на составляющие основного макроэкономического баланса с помощью неких заданных функций, параметры которых определим из данных экономической статистики.

Такое эконометрическое расширение простейшей модели экономики дает возможность оценить ее внешние параметры (идентифицировать модель) по статистическим временным рядам макропоказателей, которые характеризуют изменение структуры использования валового внутреннего продукта в современной экономике России. Идентифицированную модель можно использовать не только для теоретического анализа процессов распределения, но и для сценарных прогнозных расчетов изменения экономических макропоказателей.

Пусть валовой внутренний продукт (ВВП) $Y(t)$ определяется

однородной производственной функцией объемов капитала (основных фондов страны) $K(t)$ и труда (среднегодового числа занятых в народном хозяйстве страны) $L(t)$ с постоянной эластичностью замещения³, (CES - функцией)

$$Y(t) = Y_0 \left[a \left(\frac{L(t)}{L_0} \right)^{-b} + (1 - a) \left(\frac{K(t)}{K_0} \right)^{-b} \right]^{-\frac{1}{b}}, \quad (3.1)$$

где начальные значения (нижний индекс 0 соответствует $t = 2000$ г.) и параметры обладают свойствами $Y_0 > 0$, $L_0 > 0$, $K_0 > 0$, $a \in (0, 1)$, $b > -1$.

Однородная производственная функция с постоянной эластичностью замещения успешно применялась при исследовании экономики СССР и США (см., например, [25]), ее можно применить для исследования экономики современной России. Для учета научно-технического прогресса в [25] применялась модификация производственной функции (3.1), в которую добавлялся множитель в виде показательной функции времени $\exp(\lambda t)$. Здесь мы рассматриваем короткий период времени и пока абстрагируемся от учета научно-технического прогресса.

Обычно параметры производственной функции определяют с помощью нелинейного метода наименьших квадратов по данным экономической статистики для временных рядов переменных, непосредственно входящих в производственную функцию.

Но, во-первых, такой подход полностью оправдан только при отсутствии модели, в которой используется данная производственная функция, поскольку все параметры модели должны быть согласованы. Найти параметры производственной функции, внутренне согласованные с поведением других переменных модели, весьма сложно в силу чрезвычайно большого

³Эластичность замещения труда и капитала ϵ для CES-функции (3.1) выражается через параметр b : $\epsilon = 1/(1 + b)$.

числа возможных вариантов. Однако использование высокопроизводительных вычислений на современной вычислительной технике позволяет найти параметры производственной функции, согласованные с поведением других переменных математической модели экономики.

Во-вторых, что более существенно, при рассмотрении современной российской экономики нужно иметь в виду, что значения второго фактора производственной функции, капитала $K(t)$, представляемые статистическими органами, вызывают большое сомнение (см., например, [26]).

Статистические данные по капиталу (основным фондам страны) практически не меняются от года к году, а их стоимость чрезвычайно завышена. Фактически эти данные относятся к основным фондам, созданным во времена СССР, а они в настоящее время в большой степени представляют собой "бесплатные" ресурсы, подобные некоторым природным ресурсам, которые еще можно использовать без оплаты (воздух, а в некоторых случаях вода и земля). На выпуск оказывает влияние только капитал, вовлеченный в процесс производства, имеющий объективную стоимость, некий "эффективный" капитал, который при идентификации модели будет **выражен в постоянных ценах 2000 г.** и который мы и пытаемся здесь оценить.

Такое представление об основных фондах (совокупном физическом капитале страны) соответствует общепринятому в экономической теории определению физического капитала, как величины, равной запасу произведенных товаров, участвующих в производстве товаров и услуг (см., например, [27, с.322]).

Здесь мы измеряем труд среднегодовым числом занятых в народном хозяйстве, что отличает нашу работу от [26], где труд измерялся общим числом отработанных часов для учета его сезонных колебаний от квартала к кварталу. На основе статистических данных (см. ниже табл. 3.1) будем полагать, что

труд $L(t)$ растет с постоянным темпом $\gamma > 0$.

$$\frac{dL}{dt} = \gamma L(t), \quad L(0) = L_0. \quad (3.2)$$

Будем считать, что капитал (эффективная стоимость производственных фондов) $K(t)$ меняется в силу обычно применяемого в макроэкономических моделях уравнения (см., например, [28])

$$\frac{dK}{dt} = J(t) - \mu K(t), \quad K(0) = K_0, \quad (3.3)$$

где μ — темп⁴ выбытия капитала, а $J(t)$ — скорость прироста нового капитала (инвестиции в основной капитал).

Пусть в каждый момент времени t выполняется основной макроэкономический продуктовый баланс в текущих ценах: сумма выпуска ВВП $p_Y(t)Y(t)$ и импорта $p_I(t)I(t)$ равна сумме конечного потребления населения, правительства и некоммерческих предприятий с добавлением чистого накопления богатств и прироста материальных запасов, $p_C C(t)$, инвестиций в основной капитал $p_J J(t)$ и экспорта $p_E E(t)$.

$$p_Y Y(t) + p_I I(t) = p_C C(t) + p_J J(t) + p_E E(t), \quad (3.4)$$

где через $p_Y(t)$, $p_I(t)$, $p_C(t)$, $p_J(t)$, $p_E(t)$ обозначены дефлятор ВВП и индексы цен на импорт, конечное потребление, инвестиции и экспорт, $p_Y(2000) = p_I(2000) = p_C(2000) = p_J(2000) = p_E(2000) = 1$. Поскольку нас интересуют значения величин

⁴Обычно этот темп отождествляют с темпом амортизации капитала. При рассмотрении эффективного капитала, реально вовлеченного в процесс производства в современной экономической ситуации в России, этот темп будет ниже темпа амортизации, поскольку в процесс производства вовлекается часть производственных фондов, доставшихся нынешней экономике от советских времен.

выпуска, инвестиций, экспорта и импорта, выраженные в постоянных ценах (ценах 2000 г.), то удобно перейти к продуктовому балансу, выраженному в индексах относительных цен

$$Y(t) + \pi_I(t)I(t) = Q(t) + \pi_J(t)J(t) + \pi_E(t)E(t), \quad (3.5)$$

где индексы относительных цен импорта, инвестиций и экспорта заданы отношениями

$$\pi_I(t) = \frac{p_I(t)}{p_Y(t)}, \quad \pi_J(t) = \frac{p_J(t)}{p_Y(t)}, \quad \pi_E(t) = \frac{p_E(t)}{p_Y(t)}, \quad (3.6)$$

а выражение, балансирующее равенство (3.5), дает величину

$$Q(t) = p_C(t) \frac{C(t)}{p_Y(t)}. \quad (3.7)$$

Для решения системы уравнений (3.1) - (3.7) нужно определить объемы инвестиций $J(t)$, экспорта $E(t)$ и импорта $I(t)$ в постоянных ценах 2000 г. Предполагаем здесь, что эти объемы определяются постоянными параметрами. Объем инвестиций в постоянных ценах $J(t)$ определяется долей σ текущей стоимости инвестиций в сумме текущих стоимостей выпуска и импорта:

$$\sigma = \frac{\pi_J(t)J(t)}{Y(t) + \pi_I(t)I(t)}. \quad (3.8)$$

Объем экспорта в постоянных ценах $E(t)$ определяется долей δ экспорта в выпуске (по их текущим стоимостям):

$$\delta = \frac{\pi_E(t)E(t)}{Y(t)}. \quad (3.9)$$

Объем импорта в постоянных ценах $I(t)$ определяется отношением ρ импорта к разнице ВВП и экспорта (по их текущим стоимостям):

$$\rho = \frac{\pi_I(t)I(t)}{Y(t) - \pi_E(t)E(t)}. \quad (3.10)$$

Из (3.8)-(3.10) видно, что объемы экспорта $E(t)$, импорта $I(t)$ и инвестиций $J(t)$ в постоянных ценах определяются параметрами δ, ρ, σ , относительными ценами $\pi_E(t), \pi_I(t), \pi_J(t)$ и выпуском (валовым внутренним продуктом) $Y(t)$.

$$\pi_E(t)E(t) = \delta Y(t), \quad (3.11)$$

$$\pi_I(t)I(t) = \rho(1 - \delta)Y(t), \quad (3.12)$$

$$\pi_J(t)J(t) = \sigma(1 + \rho(1 - \delta))Y(t). \quad (3.13)$$

Итак, для идентификации модели надо задать изменение внешних интенсивных параметров модели: трех относительных цен $\pi_E(t), \pi_I(t), \pi_J(t)$, - а также определить семь постоянных параметров $a, b, \gamma, \mu, \sigma, \delta, \rho$ и три начальных значения Y_0, K_0 , и L_0 таким образом, чтобы расчетные временные ряды макропоказателей (переменных модели) были близки к статистическим временным рядам соответствующих макропоказателей экономики России.

Таблица 3.1. Статистические данные

год	2000	2001	2002	2003	2004	2005	2006
$L(t)$	65.273	65.124	66.358	67.247	67.244	68.719	69.600
$\pi_E(t)$	1	0.84442	0.76610	0.72863	0.68475	0.69651	0.67010
$\pi_I(t)$	1	0.89204	0.82339	0.73075	0.59196	0.52193	0.45556
$\pi_J(t)$	1	1.02043	1.00752	0.97393	0.93350	0.88821	0.85997
$Y(t)$	7305.6	7676.9	8039.3	8625.8	9268.8	9817.6	10478.0
$I(t)$	1755.8	2084.1	2388.4	2811.2	3466.2	4055.4	4878.3
$J(t)$	1165.2	1265.7	1300.0	1462.2	1633.6	1807.2	2051.7
$E(t)$	3218.9	3354.1	3699.6	4162.0	4653.1	4950.9	5297.5
$Q(t)$	4677.3	5412.2	5861.9	6223.4	6609.5	6880.7	7386.3

Используемые статистические данные представлены в табл. 3.1. Источник данных: Федеральная служба государственной статистики РФ <http://www.gks.ru>. Составляющие ВВП приведены в постоянных ценах 2000 г., в млрд. руб. Индексы относительных цен получены расчетом на основе данных по изменению составляющих ВВП в текущих и постоянных ценах

и имплицитного дефлятора ВВП. Данные для величины $Q(t)$, определенной (3.7), рассчитаны из баланса (3.5).

На основе имеющихся в табл. 3.1 статистических данных можно непосредственно определить часть параметров модели, во всяком случае некоторые границы их изменения. По ходу такой идентификации параметров мы уточним и само описание модели.

Определим экспоненциальную линию тренда для числа занятых $L(t)$ согласно статистическим данным по занятости, представленным в табл. 3.1 (как среднеквадратическое отклонение расчетных и статистических значений). Ниже приведен листинг участка кода на языке MATLAB, в котором реализована подгонка методом наименьших квадратов статистических данных для числа занятых в экономике России в 2000-2006 гг. $L(t)$ к экспоненциальной функции.

Подгонка $L(t)$ экспоненциальной функцией

```
%% Подгонка экспоненциальной моделью
L=[65.273 65.124 66.358 67.247 67.244 68.719 69.6];
t=[2000 2001 2002 2003 2004 2005 2006];
tm2000=t-2000;
% exp1 - имя модели из библиотеки моделей cflibhelp
[Lfit,expgodness] = fit(tm2000',L','exp1');
% полученная подгонка используется в идентификации
Lestim = Lfit(tm2000);
Lfit
% характеристики подгонки
expgodness
% рисунок "Сравнение расчета со статистикой по труду L"
figure
plot(t,L,'ob',t,Lestim,'or','LineWidth',2); hold on
h = legend('L stat','L estim',2);title('L')
plot(t,L,'-b',t,Lestim,'-r','LineWidth',2);
```

В результате экспоненциальной подгонки в командном окне получим данные, представленные ниже, что дает следующее за ними соотношение для $L(t)$:

Результат подгонки параметров функции $L(t)$

```
Lfit =
  General model Exp1:
    Lfit(x) = a*exp(b*x)
  Coefficients (with 95% confidence bounds):
    a = 64.84 (64.14, 65.55)
    b = 0.01122 (0.008283, 0.01419)
  expgodness =
    sse: 0.8313
    rsquare: 0.9502
    dfe: 5
  adjrsquare: 0.9403
    rmse: 0.4077
```

$$L(t) = 64.84e^{0.01124(t-2000)}. \quad (3.14)$$

Для уравнения (3.2) соотношение (3.14) дает значение темпа роста занятого населения параметра $\gamma = 0.01124$ и значение начальной величины числа занятых $L_0 = L(2000) = 64.84$, при этом последнее несколько меньше статистического значения из табл. 3.1. Для дискретного с единичным шагом варианта уравнения (3.2): $L_{t+1} = L_t(1 + \hat{\gamma})$, - имеем $\hat{\gamma} = \exp^{0.01124} - 1 = 0.0113$, $L_0 = 64.84$. В дальнейшем при совокупной оценке параметров модели в дискретном времени можно, например, искать два данных параметра, исходя из доверительных границ в 95%: $L_0 \in [64.14, 65.55]$, $\hat{\gamma} = [0.00832, 0.01420]$.

Можно также найти линии тренда для индексов относительных цен $\pi_E(t)$, $\pi_I(t)$ и $\pi_J(t)$ согласно данным табл. 3.1 (как сред-

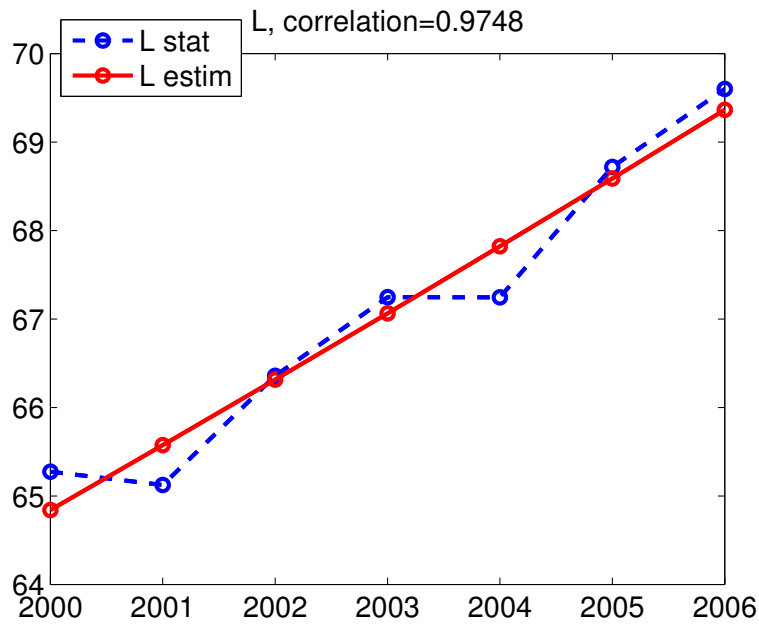


Рис. 3.1. График экспоненциальной подгонки труда $L(t)$ на основе данных табл. 3.1

неквадратическое отклонение расчетных и статистических значений), например по формулам

$$\pi_E(t) = a_E + (1 - a_E)e^{-b_E(t-2000)}, \quad (3.15)$$

$$\pi_I(t) = 1 - a_I(t - 2000)^2 e^{-b_I(t-2000)}, \quad (3.16)$$

$$\pi_J(t) = a_J + (1 - a_J)(1 + t - 2000)e^{-b_J(t-2000)}. \quad (3.17)$$

В этих уравнениях условия нормировки $\pi_E(2000) = \pi_I(2000) = \pi_J(2000) = 1$ соблюдаются автоматически. Фрагмент

кода для вычисления параметров функции $\pi_E(t)$ представлен ниже.

**Подгонка индекса относительных цен $\pi_E(t)$
функцией (3.15)**

```
%% Подгонка piE моделью x(1)+(1-x(1))*exp(-x(2)*(t-2000))
piE=[1 0.84442 0.76610 0.72863 0.68475 0.69651 0.67010];
t=[2000 2001 2002 2003 2004 2005 2006];
tm2000=t-2000;
ydata=piE;lb=[0 -inf]; ub=[inf inf];
[xE resnorm]= lsqcurvefit(@funE,[0 0],tm2000,ydata,lb,ub)
piEestim=funE(xE,tm2000);

figure
plot(t,piE,'ob',t,piEestim,'r-','Marker','s')
title('piE')
function piX=funE(x,xdata)
piX=x(1) + (1-x(1))*exp(-x(2)*xdata);

%%% Результат, отображаемый в командном окне %%%
>>
xE =
    0.6684    0.6142
resnorm =
    4.2293e-004
```

Необходимый код для вычисления параметров функций $\pi_I(t)$, $\pi_J(t)$ легко написать по аналогии, что начинающему пользователю MATLAB рекомендуется сделать самостоятельно. Ниже представлен код соответствующих этому коду вспомогательных функций на языке MATLAB.

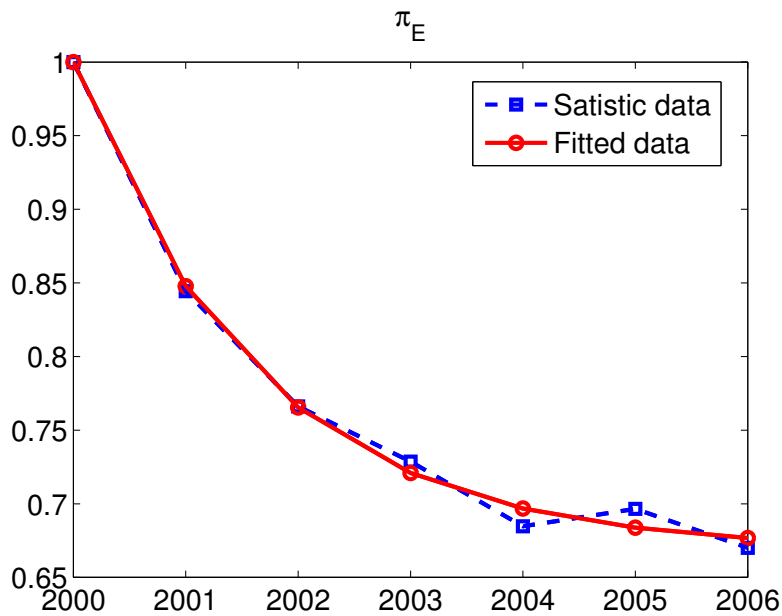


Рис. 3.2. Подгонка относительной цены π_E по данным табл. 3.1

**Вспомогательные функции для оценки индексов
относительных цен на импорт π_I и на инвестиции π_J**

```
%
function piX=funI(x,xdata)
piX=1-x(1) *xdata.*xdata.*exp(-x(2)*xdata);% x(1)>0
%
function piX=funJ(x,xdata)
piX=x(1) + (1-x(1))*(1+xdata).*exp(-x(2)*xdata);% x(1)>0
%
```

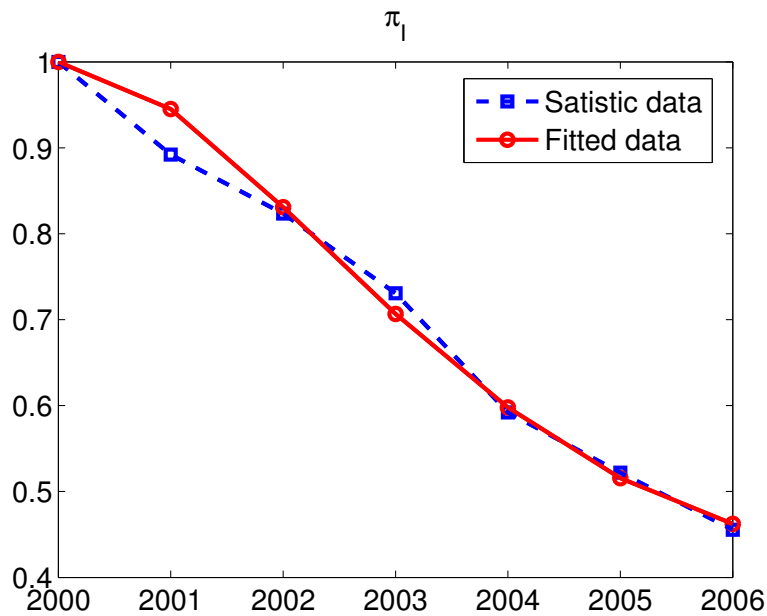


Рис. 3.3. Подгонка относительной цены π_I по данным табл. 3.1

Результаты подгонки относительных цен

$$\pi_E(t) = 0.6684 + 0.3316e^{-0.6142(t-2000)}, \quad (3.18)$$

$$\pi_I(t) = 1 - 0.0712(t - 2000)^2 e^{-0.2602(t-2000)}, \quad (3.19)$$

$$\pi_J(t) = 0.811 + 0.189(1 + t - 2000)e^{-0.5276(t-2000)} \quad (3.20)$$

представлены на рис. 3.2 - 3.4.

Среднее значение отношения объема инвестиций в основной капитал к сумме ВВП и импорта (3.8) с 2001 г. по 2006 г. остается практически постоянным: $\sigma = 0.1346 \pm 0.0026$ (первая цифра - среднее значение, вторая - среднеквадратичное отклонение), так

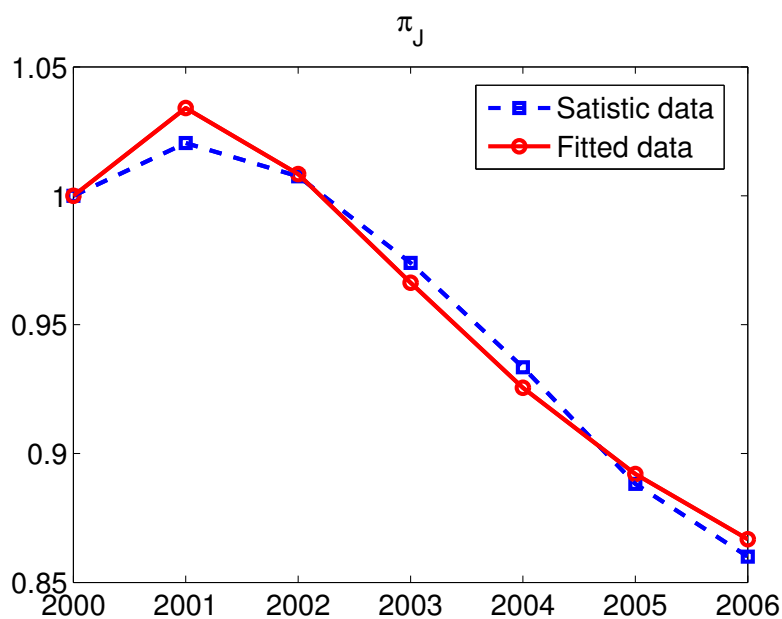


Рис. 3.4. Подгонка относительной цены π_J по данным табл. 3.1

что $\sigma \in [0.1320, 0.1372]$. Значение этой величины за 2000 г., $\sigma = 0.1286$, чуть-чуть не укладывается в этот интервал, что можно объяснить тем, что 2000 г. еще следует относить к переходному периоду экономики к новой структуре после дефолта 1998 г.

Для периода с 2001 г. по 2006 г. доля δ экспорта в ВВП (3.9) $\delta = 0.3511 \pm 0.0103$, а для переходного 2000 г. $\delta = 0.4406$. Если учитывать только период времени с 2001 г. по 2006 г., интервал для поиска наиболее близкого к статистике δ будет меньше: $\delta \in [0.3407, 0.3614]$.

То же самое касается параметра ρ отношения импорта к

остатку от ВВП, после вычета из него объема экспорта (3.10) с 2001 г. по 2006 г. $\rho = 0.3532 \pm 0.0264$, а для переходного для установления импорта 2000 г. $\rho = 0.4296$. Если учитывать только период времени с 2001 г. по 2006 г., интервал для поиска наиболее близкого к статистике ρ меньше: $\rho \in [0.3268, 0.3796]$.

Для точного определения пятнадцати параметров модели $a, b, K(0), \mu, L(0), \hat{\gamma}, a_E, b_E, a_I, b(I), a_J, b_J, \sigma, \delta, \rho$ можно использовать косвенный подход, сравнивая рассчитанные по модели экономики России выходные временные ряды ее переменных с доступными статистическими временными рядами этих переменных для 2000 - 2006 гг. Временные ряды считаются похожими, если они близки как функции времени (другими словами, между значениями временных рядов существует сильная, возможно нелинейная, связь). Поскольку длины статистических временных рядов, которым мы доверяем, составляют здесь небольшую величину в шесть значений, мы не будем здесь использовать индекс похожести, введенный на основе вейвлет-коэффициентов для сравнения нелинейных временных рядов в [29-31]. Вместо него будем использовать коэффициент корреляции Пирсона $D(X, Y)$, который является мерой силы и направленности линейной связи между сравниваемыми временными рядами X, Y , и чем он ближе к единице, тем более схоже поведение этих рядов. При этом следует учитывать, что инфляционная составляющая может увеличивать линейную связь рядов, поэтому при использовании коэффициента корреляции нужно сравнивать показатели в реальных величинах. Если длины сравниваемых временных рядов равны n , то имеем следующее выражение для коэффициента корреляции Пирсона:

$$D(X, Y) = \frac{n \left(\sum_{t=1}^n X_t Y_t \right) - \left(\sum_{t=1}^n X_t \right) \left(\sum_{t=1}^n Y_t \right)}{\sqrt{\left[n X_t X_t^T - \left(\sum_{t=1}^n X_t \right)^2 \right] \left[n Y_t Y_t^T - \left(\sum_{t=1}^n Y_t \right)^2 \right]}}, \quad (3.21)$$

где T – знак транспонирования.

Индекс Тейла $E(X, Y)$ измеряет несовпадение временных рядов X_t и Y_t и чем ближе он к нулю, тем ближе сравниваемые ряды [32]. Для удобства проведения расчетов со сверткой критериев в дальнейшем вместо индекса Тейла будем использовать коэффициент близости $U(X, Y) = 1 - E(X, Y)$:

$$U(X, Y) = 1 - \sqrt{\frac{\sum_{t=1}^n (X_t - Y_t)^2}{\sum_{t=1}^n X_t^2 + \sum_{t=1}^n Y_t^2}}. \quad (3.22)$$

Чем выше он (чем ближе он к единице), тем более близки ряды.

При сравнении экономических временных рядов используется индекс Тейла, а не среднеквадратическое отклонение. Это связано с тем, что экономические макропоказатели могут экспоненциально расти, например, на режиме сбалансированного роста, и в экономике этот режим считается вполне нормальным, хорошим.

Для однозначности выбора оптимального варианта можно использовать ту или иную свертку коэффициентов близости $U(X, Y)$ и корреляции $D(X, Y)$; например, если подгонка расчетных и статистических данных для всех макропоказателей имеет примерно равную важность, можно максимизировать среднегеометрическую величину всех коэффициентов.

В формальной записи

$$F(\vec{a}) \rightarrow \max_{\vec{a} \in A}, \quad (3.23)$$

где множество параметров задано на параллелепипеде

$$A = \{ \vec{a} \in R^N : a_i^- \leq a_i \leq a_i^+, 1 \leq i \leq N \}, \quad (3.24)$$

а функционал представляет собой среднегеометрическое всех кри-

териев близости и корреляции:

$$F = \sqrt[2m]{\prod_{j=1}^m D_j(\vec{a})U_j(\vec{a})}. \quad (3.25)$$

Здесь m - число макропоказателей; j - номер макропоказателя, $j = 1 \dots, m$.

При этом при переборе следует рассматривать только те варианты значений параметров, при которых коэффициенты близости и корреляции выше некоторых заданных положительных величин; например, в нашем случае можно использовать ограничения $D_j > 0.7$, $U_j > 0.85$, где $j = 1, \dots, m$.

В первом приближении при определении параметров модели можно для параметров σ, δ, ρ взять их средние значения и использовать выражения (3.14), (3.18)-(3.20) для задания внешних величин $L(t), \pi_E(t), \pi_I(t), \pi_J(t)$. Тогда вместо пятнадцати неопределенных параметров останется только четыре, $a, b, K(0), \mu$, которые можно найти простым перебором.

После определения этих четырех параметров для уточнения решения задачи идентификации можно вернуться к первоначальной постановке задачи, состоящей в определении пятнадцати параметров, интервалы изменения которых можно задать как небольшие отклонения от решения упрощенной задачи. Начинаящему пользователю **MATLAB**, после изучения численной реализации задачи идентификации для четырех параметров, будет полезно написать параллельную программу для первоначальной задачи в качестве упражнения, выбрав по три-четыре точки на интервале изменения каждого искомого параметра.

В случае сложных моделей, содержащих десятки и сотни неизвестных параметров можно использовать декомпозицию модели на отдельные блоки, задавая внешние для блока переменные временными рядами, взятыми из статистических данных, а при их отсутствии — из данных, полученных при расчете других блоков

модели. Такой подход даст возможность за разумное время определить независимые параметры благодаря параллельным вычислениям для перебора параметров модели на заданных интервалах их изменения.

3.2 Численная реализация задачи идентификации

Сложные математические модели экономических систем, как правило, рассчитывают по численной схеме с дискретным временем. Поскольку наша задача — показать основные проблемы и пути их решения в общем случае, мы и для расчета нашей простейшей модели будем применять дискретную схему. Более того, для простоты изложения мы шаг по времени примем равным единице. Тогда уравнение (3.3) в дискретном виде выглядят так:

$$K_{t+1} = (1 - \mu)K_t + J_t, K_0 = K_0^*. \quad (3.26)$$

Здесь дискретный момент времени $t = 0$ соответствует 2000 г., величина K_0^* равна искомому начальному запасу капитала, а параметр μ , если он положителен, определяет долю амортизируемого в течение года капитала. Заметим, что единичный шаг по времени удобен для сравнения рассчитываемых по модели показателей с их статистическими аналогами, представленными в табл. 3.1.

Для остальных уравнений дискретная и непрерывная записи совпадают (для отличия двух форм записи мы в дискретной записи момент времени указываем с помощью нижнего индекса, а численные значения начальных данных помечаем верхней звездочкой). Простые алгебраические преобразования уравнений (3.4) - (3.20) дают выражение всех макропоказателей простейшей модели в момент времени t — экспорта E_t , импорта I_t , инвестиций в основной капитал J_t , балансирующей величины конечного потребления домашних хозяйств и правительственных организаций Q_t — как величин, пропорциональных валовому внутреннему

продукту (выпуску) Y_t и деленных на соответствующие индексы относительных цен.

$$E_t = \frac{\delta Y_t}{\pi_t^E}, \quad (3.27)$$

$$I_t = \frac{\rho(1-\delta)Y_t}{\pi_t^I}, \quad (3.28)$$

$$J_t = \frac{\sigma(1+\rho(1-\delta))Y_t}{\pi_t^J}, \quad (3.29)$$

$$Q_t = ((1-\sigma)(1+\rho(1-\delta)) - \delta)Y_t. \quad (3.30)$$

Для положительности величины Q , в которую включено конечное потребление, требуется выполнение условия продуктивности системы.

$$1 - \delta - \frac{\sigma}{1 + \rho(1 - \sigma)} > 0. \quad (3.31)$$

Относительные индексы цен в (3.27) - (3.30) определяются соотношениями (3.18) - (3.20).

Теперь для упрощения работы с моделью перейдем в выражениях для труда L_t , капитала K_t и выпуска Y_t к относительным величинам l_t, k_t, y_t соответственно:

$$l_t = \frac{L_t}{L_0}, k_t = \frac{K_t}{K_0}, y_t = \frac{Y_t}{Y_0}. \quad (3.32)$$

Начальные значения всех этих величин равны единице: $l_0 = k_0 = y_0 = 1$. Тогда (3.1) и (3.32) дают

$$y_t = \left[a l_t^{-b} + (1-a) k_t^{-b} \right]^{-\frac{1}{b}}. \quad (3.33)$$

Из (3.14), (3.26), (3.32) получим

$$l_t = e^{\gamma t}, \quad l_0 = 1, \quad (3.34)$$

$$k_{t+1} = (1-\mu)k_t + \frac{\alpha\beta y_t}{\pi_t^J}, \quad k_0 = 1. \quad (3.35)$$

В (3.34) $\gamma = 0.01124$. В (3.35) введены обозначения

$$\alpha = \frac{Y_0}{K_0}, \quad \beta = \sigma(1 + \rho(1 - \delta)), \quad (3.36)$$

$$\pi_t^J = a^J + (1 - a^J)(1 + t)e^{-b^J t}, \quad \pi_0^J = 1, \quad (3.37)$$

где $a^J = 0.811$, $b^J = 0.5276$.

Наша основная задача — подобрать такой временной ряд для капитала, который наилучшим образом приближает временные ряды для макропоказателей, рассчитанных по модели, к их статистическим аналогам, представленным в табл. 3.1, поэтому численную реализацию идентификации (нахождения внешних параметров) модели мы и начинаем с варианта с наименьшим числом параметров. Мы предполагаем, что представленные ниже параметры фиксированы: $L_0 = 64.84$, $Y_0 = 7305.6$, $\gamma = 0.01124$, $\rho = 0.3532$, $\delta = 0.3511$, $\sigma = 0.1346$, $a^J = 0.811$, $b^J = 0.5276$. Тогда, в соответствии с (3.36) $\beta = 0.1569$.

Итак, имеем соотношения модели (3.33)-(3.35), которые при каждом заданном наборе параметров a, b, μ, α с помощью выражений (3.32)-(3.35) и (3.27)-(3.30) дают искомые временные ряды макропоказателей $Y_t, L_t, K_t, I_t, Q_t, J_t, E_t$. Для сравнения близости расчетных временных рядов указанных макропоказателей с их статистическими аналогами надо вычислить критерии корреляции (3.21) и близости (3.22) для выпуска, потребления, инвестиций, экспорта и импорта - Y_t, Q_t, J_t, E_t, I_t - за период 2001-2006 гг. и вычислить свертку этих критериев (3.25).

Возможный интервал изменения оцениваемых параметров: $a \in (0, 1)$, $b \in (-1, 2)$, $\mu \in (-0.2, 0.1)$, $\alpha \in (0, 3)$. Для поиска параметров с помощью параллельных вычислений надо взять сетку по каждому из интервалов, устроить перебор всех возможных сочетаний, распараллелить этот перебор на доступное число процессоров. На каждом из процессоров отбросить варианты, в которых коэффициенты корреляции и близости не превышают 0.4. Среди оставшихся вариантов выбрать вариант с наиболь-

шей сверткой (3.25), совокупным критерием $F(\vec{a})$, отправить его номер процессору-мастеру, вычислить самый большой критерий среди полученных наибольших у процессоров-рабочих, а затем для полученного оптимального варианта рассчитать все временные ряды макропоказателей модели по формулам (3.27)-(3.30), (3.32)-(3.35), вывести все графики, сравнивающие расчет со статистикой. Схема алгоритма приведена ниже.

Схема алгоритма

1. В цикле по сеткам для интервалов задания параметров задаются их значения через равный интервал (эти циклы и надо распараллелить для расчета разных групп наборов параметров в параллельных процессах, идущих на разных процессорах многопроцессорной вычислительной системы или разных ядрах многоядерной системы).
2. Для каждого набора параметров рассчитываются временные ряды всех макропоказателей L, Y, I, Q, J, E .
3. Для каждого макропоказателя, участвующего в сравнении расчетных и статистических данных, рассчитываются критерии (3.21)-(3.22) близости U и корреляции D между расчетными и статистическими временными рядами на промежутке времени 2001-2006 гг., для которого, как мы предполагаем, справедливы выдвинутые нами гипотезы о постоянности долей (3.8)-(3.10).
4. Рассчитываем свертку критериев (3.25).
5. В каждом процессе определяем лучший номер набора параметров по критерию (3.23).
6. На процессе-мастере выбираем лучший номер из лучших, полученных от рабочих процессов. Рассчитываем и распечатываем для него временные ряды всех показателей и значения всех критериев.

3.2.1 Листинги программ

Описанный алгоритм идентификации динамической системы реализован в виде основного файла сценария, в котором происходит инициализация переменных, первичная статистическая подгонка данных и кластерной части, набора m-файлов, исполняемых на кластере.

Основной m-файл, выполняемый на кластере, состоит из трех логических блоков:

1. Инициализация сетки для каждого рабочего процесса (см. код на языке MATLAB ниже).

Инициализация сетки

```
slice=(0.9-0.1)/numlabs; % участок для каждого
                        % рабочего уникален
a1=0.1+(labindex-1)*slice;% начало отрезка
a2=0.1+labindex*slice;   % и конец отрезка
                        % для каждого рабочего
                        % также уникальны
for a = linspace(a1,a2,slice) % верхний цикл
    index1=index1+1;
```

2. Непосредственно вложенный цикл, в котором происходит вычисление значения целевой функции F (см. ниже код на языке MATLAB).

Основной вложенный цикл

```
for a = linspace(a1,a2,slice)
    index1=index1+1;
    index2=0;
    for b = linspace(-0.9,-0.6,n)
```

```

index2=index2+1;
index3=0;
for alphaa=linspace(0.04,0.9,N)
    K0=Y0/alphaa;
    index3=index3+1;
    index4=0;
    for muu= linspace(-0.2,0.05,N)
        index4=index4+1;
        [Yestim Kestim]=...
            forecastYK(Y0, K0, Y0, K0, L0, a,...
                b, 1, alphaa, muu,...
                betaa, Lestim, piJestim);
        Jestim = betaa * Yestim ./ piJestim';
        Eestim = deltaa * Yestim ./ piEestim';
        Iestim = rho * (1 - deltaa) *...
            Yestim ./ piIestim';
        Qestim = ((1-sigmaa) * (1 + rho*...
            (1-deltaa))- deltaa) * Yestim;
        Ycor=pearson(Y',Yestim);
        Yomt=omth(Y',Yestim);
        Jcor=pearson(J',Jestim);
        Jomt=omth(J',Jestim);
        Yres=[Ycor Yomt Jcor Jomt];
        if sum(Yres<0.4)>=1
            F(index1,index2,index3,index4)=-1;
        else
            F(index1,index2,index3,index4)=...
                geomean(Yres);
        end
    end
end
end
end
end
end

```

Внутри цикла происходит вызов вспомогательных функций (функций для расчета коэффициента близости по индексу Тейла, коэффициента корреляции Пирсона и прогноза выпуска и капитала по уравнениям модели), которые доступны каждому рабочему процессу, в силу того что планировщик автоматически доставляет и распаковывает их в каждой сессии рабочего процесса. Код этих вспомогательных функций на языке MATLAB представлен ниже.

Функция близости рядов по индексу Тейла

```
function omth=omth(x,y)
% omth = One minus Theil's index
numerator=sum((x-y).^2);
denominator=sum(x.^2)+sum(y.^2);
omth=1-sqrt(numerator/denominator);
```

Функция корреляции Пирсона

```
function pearson=pearson(x,y)
n=numel(x);
numerator=n*(sum(x.*y))- sum(x)*sum(y);
denominator= sqrt((n*sum(x.^2)-sum(x)^2)*...
                (n*sum(y.^2)-sum(y)^2));
pearson=numerator/denominator;
```

Функция прогноза выпуска и капитала

```
function [Y K]=forecastYK(Yt, Kt, Y0, K0, L0,...
    a, b, c, alphaa, muu, betaa, L, PiJ)
```

```

if length(L)==length(PiJ)
    N=length(L);
else
    warning('In function forecastYK
            time-series L and PiJ have
            different lengths')
end
l=L/L0;
y=zeros(N,1); Y=zeros(N,1);
k=zeros(N,1); K=zeros(N,1);
y(1)=Yt/Y0; k(1)=Kt/K0; % Yt, Kt for forecast
for i=2:N
    k(i)=(1-muu)*k(i-1)+alphaa*betaa*y(i-1)/PiJ(i-1);
%   to avoid 1/0 error
    if b==0
%   Cobb-Douglas production function
        y(i) = (l(i)^a) * (k(i)^(1-a));
    else
%   CES production function geterogenity c
        y(i)=(a*l(i)^(-b)+(1-a)*k(i)^(-b))^(1-c/b);
    end
end
end
Y=y*Y0; K=k*K0;

```

3. Финализация – этот шаг реализован в виде небольшого участка кода, в котором все рабочие процессы отправляют свои расчетные данные, максимальное значение целевой функции F и значения параметров, на которых это значение достигнуто, процессу-мастеру. На последнем происходит выбор максимального из присланных значений, что и является конечным результатом, — т.е. выходным значением функции (см. код на языке MATLAB ниже).

Финализация
(мастер выбирает наибольшее значение)

```
OptimMatrix=...
[FFoptim aoptim boptim alphaaoptim K0 muuoptim];
if labindex ~=1 % отправляем первому worker (мастеру)
    for otherLab=2:numlabs
        labSend(OptimMatrix,1);
        % 1- номер (ID мастера)
    end
    OptimParams=0;
    % переменная OptimParams будет отличаться
    % от нуля только у мастера
else % мастер принимает данные
    for otherLab=2:numlabs
        OptimMatrixGet=labReceive(otherLab)
        OptimMatrix=[OptimMatrix;OptimMatrixGet];
    end

    %выбираем наибольшее значение по F
    [FFoptim I]=max(OptimMatrix(:,1));
    OptimRow=OptimMatrix(I,:);% вектор содержащий
    %оптимальные параметры
    % Инициализируем структуру, которая будет
    % являться выходным значением
    % OptimParams отлична от нуля только у мастера
    OptimParams.FFoptim=OptimRow(1);
    OptimParams.aoptim=OptimRow(2);
    OptimParams.boptim=OptimRow(3);
    OptimParams.alphaaoptim=OptimRow(4);
    OptimParams.K0=OptimRow(5);
    OptimParams.muuoptim=OptimRow(6);
end
```

Некоторые вспомогательные функции, введенные ранее, используются в цикле перебора вариантов вместо встроенных функций MATLAB, чтобы ускорить расчет. Встроенные функции MATLAB (например, коэффициент корреляции Пирсона $\text{corr}(x', y')$), удобно использовать в последовательных частях программы, если нужно подсчитать дополнительные характеристики сравниваемых временных рядов. В частях программы, которые обрабатываются большое количество раз, подготовка быстро исполняемого кода дает существенный выигрыш во времени.

Из кода для основного вложенного цикла видно, что коэффициенты близости и похожести расчетных и статистических временных рядов используются в свертке критериев и сами по себе только для двух макропоказателей, выраженных в постоянных ценах 2000 г.: валового внутреннего продукта $Y(t)$ и инвестиций в основной капитал $J(t)$. Это снова сделано для ускорения расчета параллельной части программы. Здесь учтен тот факт, что близость расчетных и статистических рядов для остальных показателей обеспечивается автоматически в силу соотношений (3.27)-(3.30), если близки расчетные и статистические временные ряды для макропоказателя $Y(t)$.

В коде на языке MATLAB (m-файлах) для инициализации и основного вложенного цикла приведен пример распараллеливания только самого верхнего цикла. Можно распараллелить и большее число циклов, например объединив эти циклы в один цикл и распараллелив последний по приведенной выше схеме.

Начинающий пользователь MATLAB может проделать такое упражнение на распараллеливание — объединение нескольких циклов в один и разбиение общего цикла на порции для используемых параллельно процессов — самостоятельно, если для расчета задачи идентификации параметров математической модели сложной системы у него есть возможность использовать многопроцессорную технику, например кластерный суперкомпьютер, с числом процессоров, исчисляющихся десятками или сотнями.

3.2.2 Результаты идентификации модели, их графическое представление

Результаты идентификации модели представлены в графическом виде на рис. 3.5 - 3.7, где дано сравнение рассчитанных по модели и статистических временных рядов для макропоказателей экономики России.

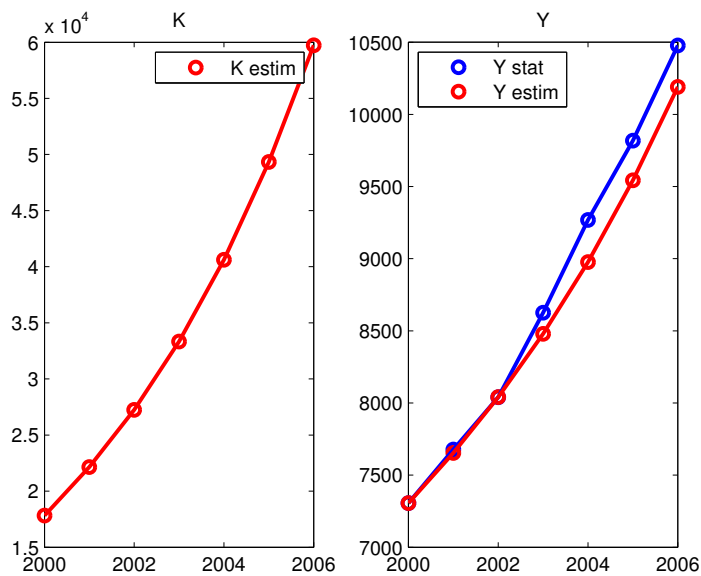


Рис. 3.5. Данные для капитала $K(t)$, рассчитанные на основе модели (K_{estim}), и данные для выпуска $Y(t)$, рассчитанные по модели (Y_{estim}) и статистические (Y_{stat})

В результате идентификации модели экономики России получена оптимальная в смысле свертки (3.25) критериев близости и

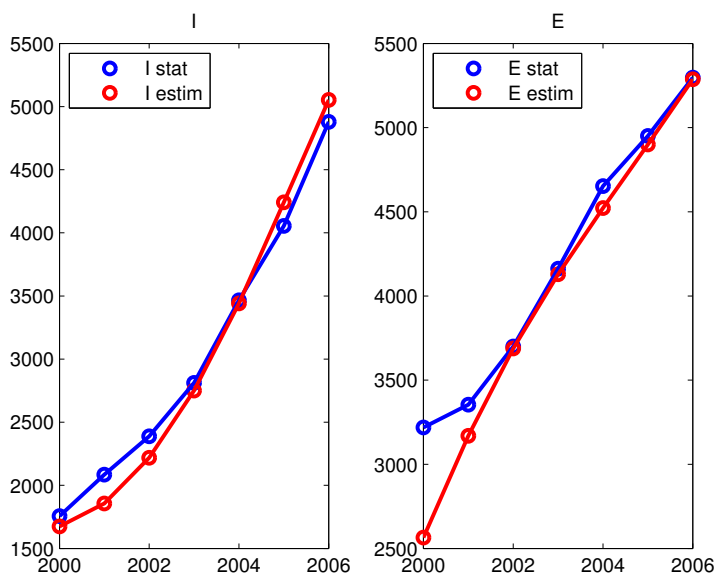


Рис. 3.6. Исторические данные и данные, рассчитанные на основе модели, для импорта (I_{stat} , I_{estim}) и экспорта (E_{stat} , E_{estim})

корреляции сравниваемых макропоказателей $Y(t)$ и $J(t)$ оценка ее параметров $a = 0.84$, $b = -0.78$, $\mu = -0.175$, $\alpha = 0.41$. На основе этих параметров из (3.36) определено начальное значение для капитала $K_0 = Y_0/\alpha = 17819$ млрд. руб 2000 г., по уравнению (3.26) восстановлен временной ряд для капитала $K(t)$, а с помощью уравнений (3.1), (3.27)-(3.30) восстановлены временные ряды рассматриваемых в модели макропоказателей для ВВП $Y(t)$, инвестиций $J(t)$, потребления $Q(t)$, импорта $I(t)$ и экспорта $E(t)$.

Представленный на рис. 3.5 капитал является эффективным

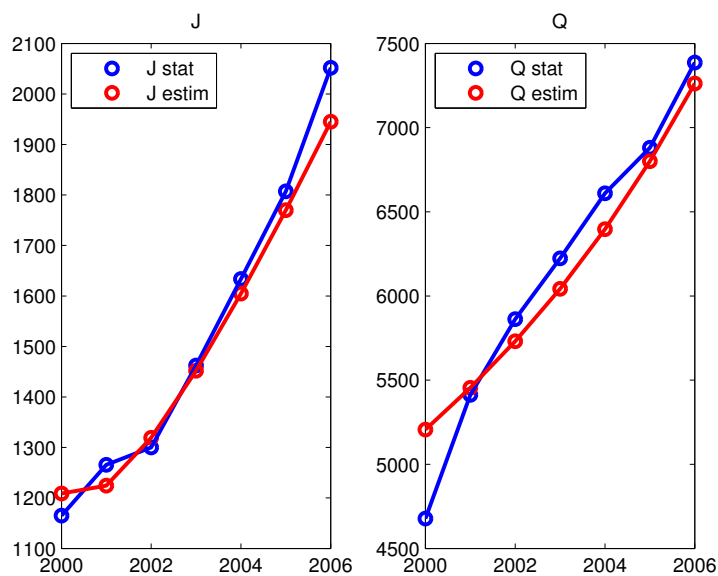


Рис. 3.7. Исторические данные и данные, рассчитанные на основе модели, для инвестиций в основной капитал (J_{stat} , J_{estim}) и потребления (Q_{stat} , Q_{estim})

капиталом, реально используемым в производстве. Он был получен при идентификации рассмотренной здесь модели экономики. Эффективный капитал отличается от значения, формально рассчитываемого статистическими органами на основе многократных переоценок в условиях огромной инфляции стоимости существующих производственных фондов.

Как ранее уже упоминалось, отрицательное значение параметра μ означает, что капитал прирастает с большей скоростью, чем это обусловлено инвестициями, за счет вовлечения в процесс

производства простаивавшего ранее капитала (производственных фондов). Но этот процесс вовлечения не может продолжаться долго, поскольку объем неиспользуемого в процессе производства капитала, доставшегося современной экономике от советских времен, ограничен.

Оценим время T с 2000 г., за которое будет вовлечен весь неиспользуемый до конца этого года капитал. Будем исходить из следующих правдоподобных предположений: (1) объем новых инвестиций совпадает с объемом капитала, выбывающего вследствие физического и морального износа; (2) за весь процесс вовлечения капитал может вырасти в четыре раза в сравнении с его начальным уровнем в 2000 г. Тогда общее время процесса вовлечения старого капитала истечет через $T = (1/|\mu|) \ln(K_T/K_0) = \ln(4)/0.175 \approx 8$ лет после 2000 г., т.е. в 2008 г. исчерпается лимит вовлечения простаивавших производственных фондов.

Другая проблема, связанная с вовлечением старых производственных фондов, состоит в том, что они чрезвычайно изношены. Это означает, что после вовлечения всего старого капитала, темпы физической деградации его долгое время могут быть чрезвычайно велики, пока сильно изношенный капитал не будет заменен на новый капитал, идущий от новых инвестиций.

На рис. 3.6 - 3.7 представлены статистические и рассчитанные по модели временные ряды макропоказателей, характеризующих структуру использования реального валового внутреннего продукта в ценах 2000 г. (экспорта $E(t)$, импорта $I(t)$, инвестиций $J(t)$ и потребления $Q(t)$). В 2000-2006 гг. все рассмотренные макропоказатели растут (см. рис. 3.5 - 3.7). Поведение этих показателей после 2006 г. можно оценить, задав сценарии будущего развития.

3.3 Сценарные расчеты с моделью

В сценарных расчетах с моделью прогноз изменения внешних параметров модели задается сценарием. Будем считать, что ин-

дексы относительных цен во всех рассматриваемых сценариях заданы функциями (3.18)-(3.20), так что мы имеем рис. 3.8 - 3.10.

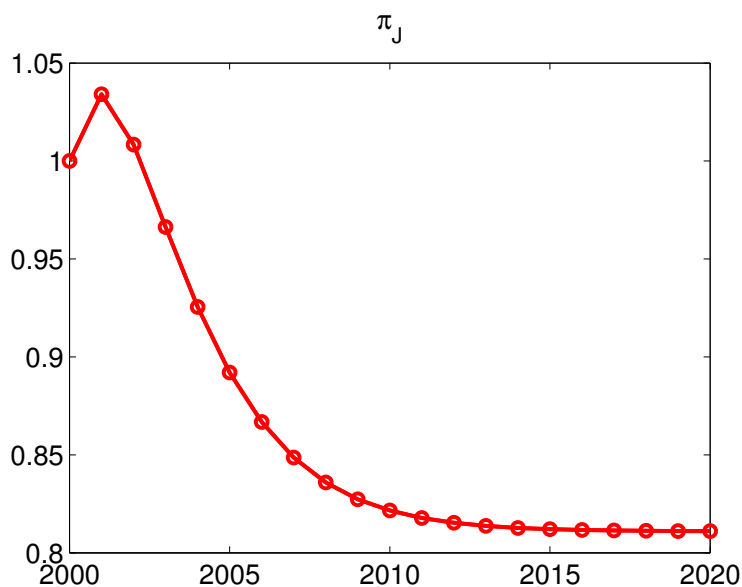


Рис. 3.8. Заданный индекс относительных цен на инвестиции в основной капитал $\pi_J(t)$

3.3.1 Базовый, он же пессимистический сценарий

Обычно в качестве базового сценария развития экономики рассматривают такой сценарий, в котором на прогнозный период времени предполагают продолжение тенденций, выявленных за период времени, на котором оценивалась модель. Однако в нашем случае так сделать нельзя, поскольку на прогнозный период

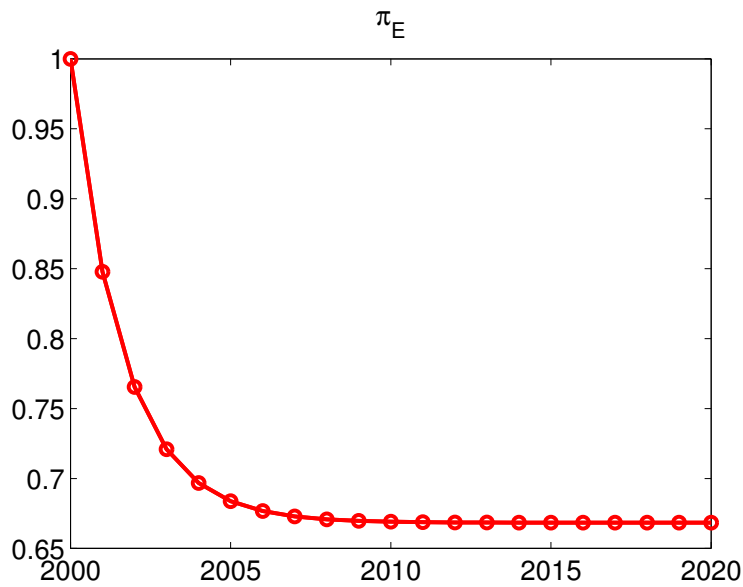


Рис. 3.9. Заданный индекс относительных цен на экспорт $\pi_E(t)$

времени с 2007 г. по 2020 г. по мнению многих экономистов выполняются следующие условия.

1. Источник вовлечения производственных фондов из наследства, доставшегося от советских времен, вот-вот будет исчерпан.
2. Производственные фонды, используемые в производстве, сильно изношены — доля производственных фондов, которые скоро выйдут из строя и которые поэтому надо срочно менять, в большинстве отраслей народного хозяйства превысила 50%.

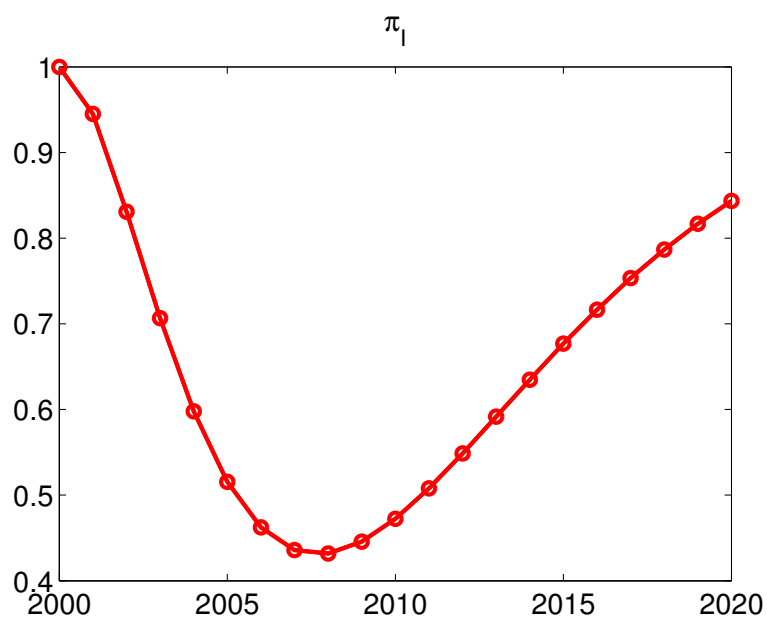


Рис. 3.10. Заданный индекс относительных цен на импорт $\pi_I(t)$

3. Прирост числа занятых в экономике, наблюдавшийся в период оценки, в прогнозный период практически невозможен, так как начнет сказываться демографический кризис, в который Россия попала в начале 90-х гг. XX в.

Поэтому базовый вариант прогноза оказывается пессимистическим.

Зададим базовый сценарий прогноза с 2007 г. до 2020 г. (он же пессимистический) следующими условиями, накладываемыми на внешние параметры модели:

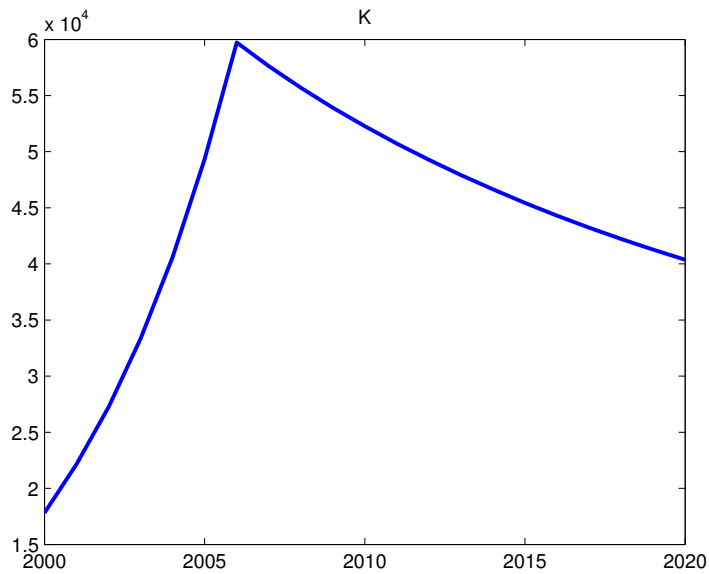


Рис. 3.11. Данные для капитала $K(t)$ (базовый сценарий)

1. Считаем, что параметры $a, b, \alpha, \delta, \sigma, \rho$ остаются постоянными, принимающими определенные ранее значения для 2000-2006 гг.: $a = 0.84, b = -0.78, \alpha = 0.41, \delta = 0.3511, \sigma = 0.1346, \rho = 0.3532$, так что $\beta = 0.1569$.
2. Параметр μ резко меняет свое значение и экономический смысл с 2009 г.: $\mu = -0.175 < 0$ до 2008 г., а начиная с 2009 г. он становится положительным $\mu = \alpha\beta = J_0/K_0 = 0.0678$ и означает темп выбытия мощностей вследствие износа. Предполагаем, что источник, из которого в производство вовлекались бесплатные производственные фонды,

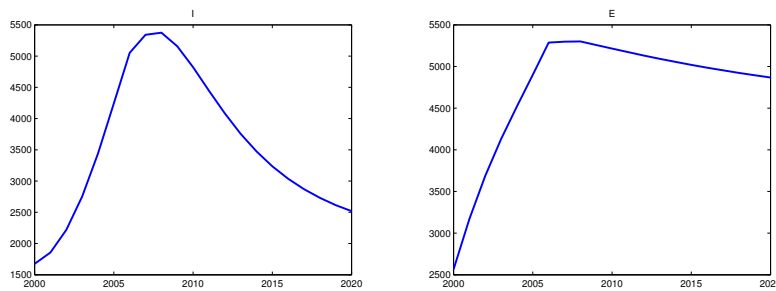


Рис. 3.12. Прогноз по базовому (пессимистическому) сценарию импорта $I(t)$ и экспорта $E(t)$

к началу 2009 г. иссякнет. Капитал меняется в силу уравнения (3.26) или в относительных величинах в силу уравнения (3.37), но μ в них зависит от времени t .

3. Считаем, что объем используемого в производстве труда до 2008 г. возрастает в силу соотношения (3.14), достигает максимального значения $L_8 = 70.94$ млн. человек в 2008 г. и далее не меняется⁵.
4. Изменение относительных цен задается функциями (3.18)-(3.20), идентифицированными по статистическим данным для периода 2000-2006 гг.
5. Выпуск валового внутреннего продукта $Y(t)$ рассчитываем по однородной функции с постоянной эластичностью замещения (3.1). Макропоказатели — инвестиции $J(t)$, экспорт

⁵Можно было бы в pessimistic сценарии задать падение числа занятых, но будем считать, что улучшение качества труда (человеческого капитала [33]) все-таки компенсирует демографический спад.

$E(t)$ и импорт $I(t)$ — определяем благодаря соотношениям (3.11)-(3.13), а потребление $Q(t)$ — из основного макроэкономического баланса (3.5).

На рис. 3.11 показан график эффективного капитала $K(t)$. Видно, что, начиная с 2009 г., капитал не растет, а деградирует.

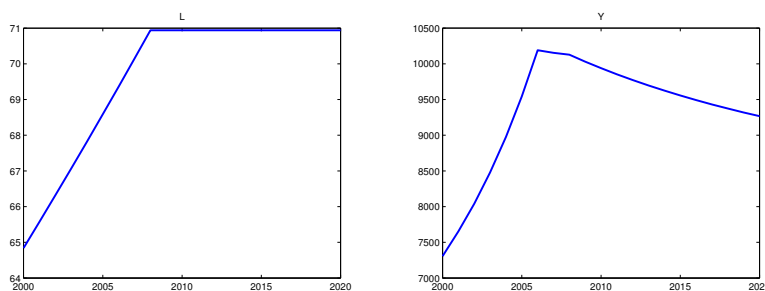


Рис. 3.13. Прогноз по базовому (пессимистическому) сценарию числа занятых $L(t)$ и валового внутреннего продукта $Y(t)$

На рис. 3.12 - 3.14 показана динамика макропоказателей российской экономики для базового сценария. Видно, что труд $L(t)$, начиная с 2009 г., не возрастает, а остальные макропоказатели — инвестиции $J(t)$, экспорт $E(t)$, импорт $I(t)$ и потребление $Q(t)$ — падают, вслед за выпуском $Y(t)$.

Пессимистический сценарий предупреждает, что будет с нами, с нашей экономикой, если не предпринимать никаких усилий к переходу от сложившейся экономической структуры — с доминированием сырьевого сектора, быстрым развитием экспортно-импортных услуг в городах-миллионерах, отставанием перерабатывающих секторов и депрессивным состоянием большинства регионов — к новой, инновационной структуре экономики, к пе-

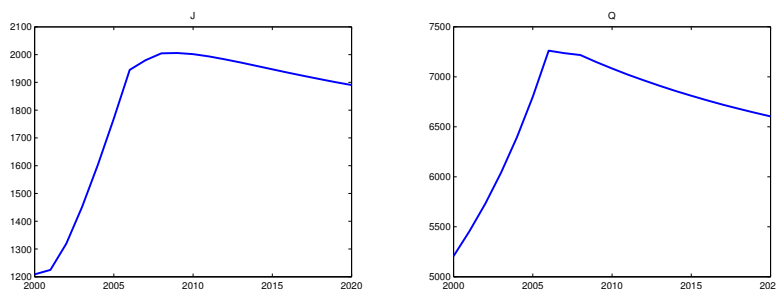


Рис. 3.14. Прогноз по базовому (пессимистическому) сценарию для инвестиций в основной капитал $J(t)$ и потребления $Q(t)$

реходу, который требует постоянных усилий всего российского общества.

3.3.2 Оптимистический сценарий

Обычно в качестве оптимистического сценария рассматривают такой сценарий, в котором модель остается той же самой, а некоторые параметры меняются. Для использовавшейся модели можно было бы предположить, что источник бесплатных производственных фондов останется всегда и рассчитать маниловский прогноз. Реалистичный оптимистический сценарий требует небольшой перестройки модели, вернее, идентификации ее по набору других параметров.

В оптимистическом сценарии прогноза с 2007 г. по 2020 г. предполагаем выполненными следующие условия.

1. Рост российской экономики идет не столько за счет вовлечения старых фондов, сколько за счет инноваций, научно-технического прогресса. В рамках модели это означает, что

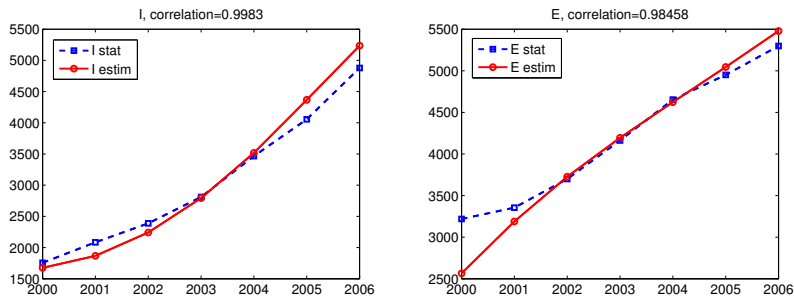


Рис. 3.15. Сравнение статистических данных с расчетом по оптимистическому сценарию на интервале оценки для импорта $I(t)$ и экспорта $E(t)$

мы имеем возрастающую отдачу на используемые производственные факторы. Значит надо вместо соотношения (3.1) для производственной функции использовать однородную степени $c > 1$ производственную функцию с постоянной эластичностью замещения:

$$Y(t) = Y_0 \left[a \left(\frac{L(t)}{L_0} \right)^{-b} + (1 - a) \left(\frac{K(t)}{K_0} \right)^{-b} \right]^{-\frac{c}{b}}. \quad (3.38)$$

Но такое изменение не дает возможности однозначно найти параметры модели, дающие лучшее значение критерия близости (3.25). Нужно делать дополнительные предположения.

2. Пусть параметр выбытия изношенных эффективных производственных фондов определяется соотношением, полученным при предположении отсутствия прироста капитала в базовом 2000 г. — вышел из употребления (был амортизирован) такой же объем эффективных производственных

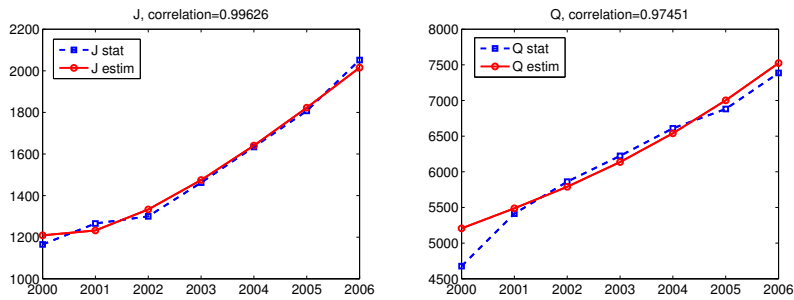


Рис. 3.16. Сравнение статистических данных с расчетом по оптимистическому сценарию на интервале оценки для инвестиций в основной капитал $J(t)$ и потребления $Q(t)$

фондов (эффективного капитала), какой поступил вновь в результате инвестиций: $\mu = J_0/K_0 = \alpha\beta$ (см. (3.35)). Значит, для оптимистического сценария вместо варьирования по параметру μ при идентификации модели появилось варьирование по параметру $c \in (1, 6)$.

3. Считаем, что качество труда в инновационной экономике возрастает, доля квалифицированного труда растет. Объем используемого труда в пересчете на простой труд может увеличиваться даже при снижении числа занятых. Поэтому считаем, что труд все время прогноза экспоненциально возрастает в силу соотношения (3.14).
4. Так же, как и в базовом сценарии, мы здесь предполагаем, что изменение относительных цен за весь период прогноза задается функциями (3.18)-(3.20), полученными для периода 2000-2006 гг. Мы вынуждены сделать это предположение, поскольку рассматриваемая модель не описывает

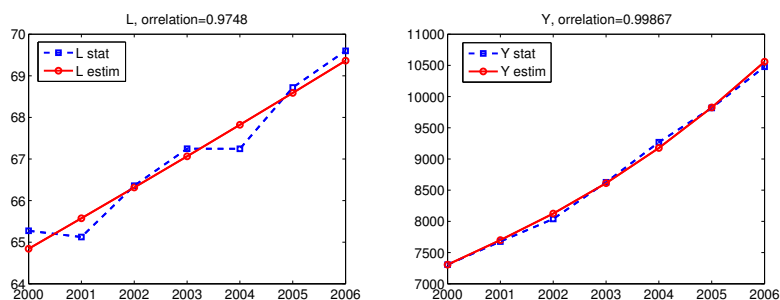


Рис. 3.17. Сравнение статистических данных с расчетом по оптимистическому сценарию на интервале оценки для числа занятых $L(t)$ и выпуска валового внутреннего продукта $Y(t)$

экономические механизмы формирования цен. Считая, что тенденции изменения индексов цен сохраняются, мы тем самым в оптимистическом сценарии предполагаем, что инновационные процессы в экономике начались с 2000 г.

- Макропоказатели модели в постоянных ценах 2000 г. – инвестиции в основные фонды $J(t)$, экспорт $E(t)$ и импорт $I(t)$ – определяем согласно соотношениям (3.11)-(3.13), а потребление $Q(t)$ – из основного макроэкономического баланса (3.5).

На рис. 3.15 - 3.17 показаны результаты идентификации модели для оптимистического сценария – динамика макропоказателей российской экономики на интервале оценки. В результате идентификации параметров данного варианта получены следующие значения параметров, доставляющие максимум критерию (3.25): $a = 0.9316$, $b = 0.82$, $\alpha = 0.9899$, $c = 5.0268$, так что $K_0 = 7380.4$ млрд. руб 2000 г., что составляет чуть менее половины от значения начального капитала в пессимистическом сценарии.

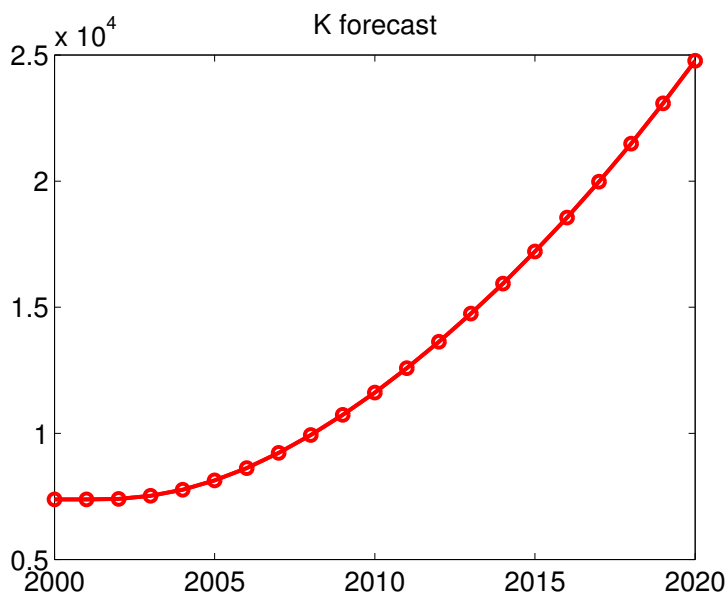


Рис. 3.18. Динамика капитала в оптимистическом сценарии

На рис. 3.18 показана динамика изменения эффективного капитала в оптимистическом сценарии. Капитал экспоненциально возрастает все время.

Прогноз для остальных макропоказателей экономики России по оптимистическому сценарию показан на рис. 3.19 - 3.21. Из рис. 3.18 - 3.21 видно, что инвестиции в основной капитал $J(t)$, экспорт $E(t)$, импорт $I(t)$ и потребление $Q(t)$ растут вслед за выпуском $Y(t)$, трудом $L(t)$ и капиталом $K(t)$. При этом вначале прогнозного периода реальный объем импорта $I(t)$ не растет, а все остальные показатели, включая конечное потребление $Q(t)$, экспоненциально растут все время. Такое поведение

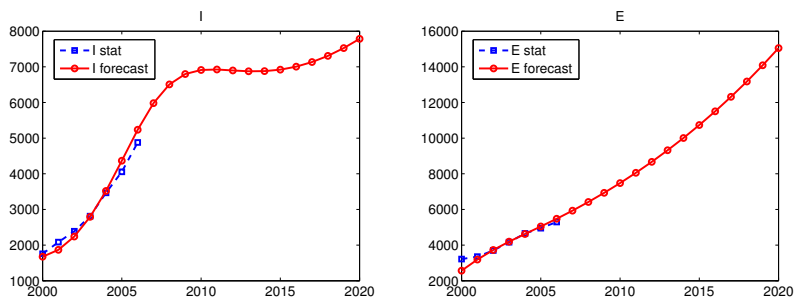


Рис. 3.19. Прогноз по оптимистическому сценарию и исторические данные для импорта $I(t)$ и экспорта $E(t)$

импорта обусловлено поведением индекса относительной цены на импорт $\pi_I(t)$, который на прогнозном периоде увеличивается (см. рис. 3.10) в соответствии с нашим предположением о виде функции для этого индекса (3.16). Нужно иметь в виду, что это предположение довольно произвольно. Можно было бы предположить например, что индекс относительных цен на импорт будет и дальше падать, как он падал ранее, но тогда он снизился бы практически до нуля.

Вполне возможно, что процесс перехода к инновационной экономике уже начался, поскольку с того уровня, на котором находится большая часть экономики России, когда более 60% населения в глубинке живет за счет натурального хозяйства в условиях бездорожья – это сделать нетрудно: чуть-чуть помочь с дорогами, с другими инфраструктурами (предоставить бесплатный доступ в Интернет, например), – и дело сдвинется с мертвой точки. Труднее сделать этот процесс необратимым. А тут еще много предстоит сделать, как, впрочем, и для построения более совершенных моделей экономики.

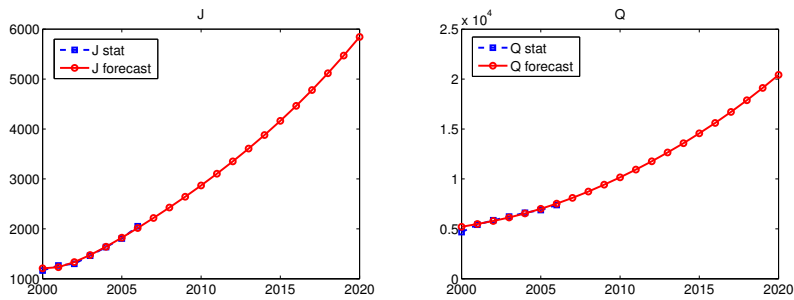


Рис. 3.20. Прогноз по оптимистическому сценарию и исторические данные для инвестиций в основной капитал $J(t)$ и для потребления $Q(t)$

3.3.3 Сравнение, выводы

Следует подчеркнуть, что к полученным результатам следует относиться осторожно. Мы рассмотрели только эконометрический вариант простейшей динамической модели экономики современной России. Назначение такого варианта модели — служить иллюстрацией для тех или иных предположений. Более глубокие результаты можно получить с помощью более продвинутых моделей (см., например, [35]), в которых описываются экономические механизмы, позволяющие сделанные здесь предположения получать внутри модели (эндогенно).

Основная цель в рассмотрении простейшей модели экономики заключалась в том, что она дает возможность на обозримом материале продемонстрировать все те трудности, с которыми сталкиваются исследователи, строящие и эксплуатирующие новые математические модели, и показывает пути решения этих проблем с помощью новых программных технологий MATLAB на современной высокопроизводительной технике.

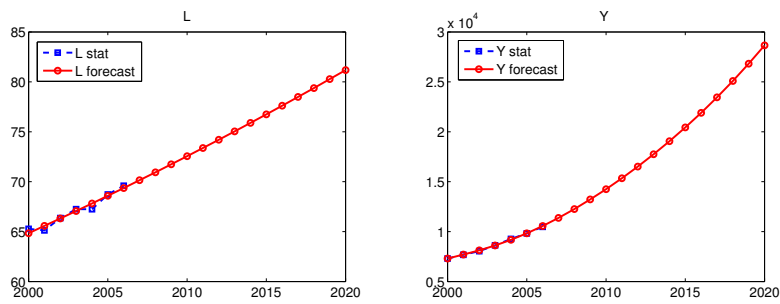


Рис. 3.21. Прогноз по оптимистическому сценарию и исторические данные для числа занятых $L(t)$ и выпуска $Y(t)$

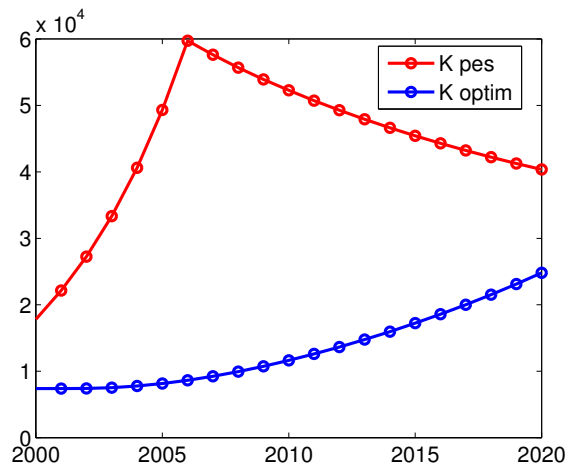


Рис. 3.22. Оценка объема капитала в российской экономике (различные сценарии)

Заключение

Данная работа является первой попыткой описания параллельного программирования в системе MATLAB. Появление новых архитектурных решений: многоядерных платформ, — подвигнуло создателей системы MATLAB и других систем уделять больше внимания параллельным аспектам вычислений, что влечет быстрые их изменения. Соответственно увеличиваются потенциальные возможности использования параллельных вычислений в различных прикладных областях. Исходя из имеющихся тенденций, авторы настоящей монографии планируют ее модификацию как для учета быстрого развития технологических возможностей системы MATLAB в области параллельных вычислений, так и для учета быстрого расширения области использования параллельных вычислений.

Кроме того, в каждом из разделов остались нерассмотренными интересные и важные вопросы. В первом разделе не были рассмотрены такие вопросы, как:

- работа с произвольной реализацией MPI,
- работа MATLAB с существующими системами очередей.

Во втором разделе не был рассмотрен вопрос, связанный с распределенным хранением исходных данных некой вычислительной задачи (в том случае, если размер исходных данных не позволяет хранить ее на рабочей станции).

В третьем разделе не затронуты вычислительные задачи, связанные с пространственно-распределенной эколого-экономической системой, в которой могут использоваться не только пространственно-распределенные данные, но и отличающиеся модели для экологического и экономического блоков. В таких задачах применение параллельных вычислений существенно, эти вычисления требуют проведения однотипных вычислений в огромном числе пространственно-распределенных точек.

Еще одним типом задач для приложения параллельных вычислений являются задачи нелинейной математической экономики, включающие поиск решения дифференциальных уравнений с частными производными. Эти задачи и методы их решения похожи на соответствующие задачи математической физики, и пример такой задачи нужно будет рассмотреть.

Эти и некоторые другие вопросы планируется осветить в следующем издании.

Лучшие применения параллельных вычислений в моделировании экономики расширяют возможности разработчиков моделей. Появилась возможность идентифицировать внешние параметры сложных нормативных балансовых динамических моделей экономических систем. Можно идентифицировать сложные, пространственно-распределенные модели эколого-экономических систем, чтобы исследовать на них последствия изменения климата на протекание экономических и экологических процессов в различных регионах и странах. Для этого разрабатываются специальные критерии близости и похожести для статистических и рассчитанных по модели временных рядов макропоказателей изучаемой экономической системы страны или региона [34], применяются информационные технологии создания математических моделей с помощью системы ЭКОМОД для уменьшения числа независимых внешних параметров, совершенствуются методы глобальной оптимизации, сокращающие время расчета. Построение спе-

циальной эконометрической модели упрощает изложение.

При изучении рассмотренной в разд. III простой эконометрической модели современной экономики России получены интересные результаты. Получена оценка величины эффективного капитала и его динамики для различных сценариев развития (см. рис. 3.22). Интересно то, что параметр μ , средний темп выбытия старых производственных фондов, в базовом сценарии принимает отрицательное значение, $\mu < 0$. В рамках рассмотренной модели это означает, что часть доставшихся от советского времени производственных фондов вовлекается в процесс производства с большим темпом, чем амортизируются производственные фонды, уже вовлеченные в процесс производства. Но такой процесс не может продолжаться бесконечно долго. Объем неиспользуемых фондов сокращается, и наступит момент, когда больше не выгодно будет вовлекать в процесс производства старые, доставшиеся с советских времен фонды. Рост трудовых ресурсов после 2008 г. также не может продолжаться из-за демографических проблем. Это описывает базовый пессимистический сценарий будущего развития экономики России на основе сложившихся в ней экономических структур.

Рассмотрен и оптимистический сценарий, в котором предполагается, что уже начался рост экономики за счет инноваций. В модели это выражается в повышенной отдаче от вложенных производственных факторов. В инновационной экономике труд увеличивает свое качество, поэтому можно считать, что количество простого труда, учитываемого в производственной функции в качестве одного из факторов, возрастает даже при уменьшении числа занятых. Хотелось бы надеяться на лучшее, но такие выводы делать преждевременно. Для более полного учета процессов, происходящих в реальной экономике, и для более обоснованных рекомендаций лицам, принимающим решения, надо строить и исследовать более сложные модели системного анализа развивающейся экономики [35].

Многие, пока не поддающиеся решению задачи математической и прикладной экономики могут быть решены благодаря использованию современных суперкомпьютеров, так же как и сложные математические задачи [36].

Надо заметить, что значительная часть кода MATLAB, отвечающего за запуск процессов с помощью планировщика, написана на языке Java. Наряду с таким преимуществом, как кроссплатформенность, этот язык обладает существенным недостатком – низкой скоростью реализованного программного кода. Но развитие многоядерных архитектур в скором будущем приведет к тому, что этот недостаток станет несущественным.

Использование высокопроизводительных вычислений расширяет область применения и исследования для математических методов. Так, значительное увеличение скорости оценки параметров математической модели сложной системы можно достичь за счет применения направленного перебора, реализованного в численных методах глобальной оптимизации [37].

Web-страничка данной книги находится по адресу: <http://www.ccas.ru/olenev/parmatlab.html>.

Новости и дальнейшие планы авторов книги можно узнать по указанным ниже Web-адресам их домашних страниц.

<http://www.ccas.ru/olenev/>

<http://pechonkin.pochta.ru>

<http://www.ccas.ru/chernetsov/>

Замеченные опечатки, пожелания, комментарии просьба высылать на указанные ниже адреса электронной почты, для нас они будут очень ценными.

Оленёв Николай Николаевич olenev@ccas.ru

Печёнкин Руслан Викторович ruslan.pechenkin@mail.ru

Чернецов Андрей Михайлович chernetsovam@mail.ru

Литература

1. *Distributed Computing Toolbox For Use with MATLAB*. 2004-2006 by The MathWorks, Inc.
http://www.mathworks.com/access/helpdesk/help/pdf_doc/distcomp/distcomp.pdf
2. *MATLAB Distributed Computing Engine For Use with MATLAB*. 2004-2006 by The MathWorks, Inc.
http://www.mathworks.com/access/helpdesk/help/pdf_doc/mdce/mdce.pdf
3. *HPC-Grid for Maple*. <http://www.maplesoft.com/products/toolboxes/HPCgrid/index.aspx>
4. *Mathematica Parallel Toolkit*. <http://documents.wolfram.com/applications/parallel/>
5. *PoochMPI Toolkit for Mathematica*
<http://daugerresearch.com/pooch/mathematica.shtml>
6. *Parallel MATLAB Survey*.
<http://www.interactivesupercomputing.com/reference/parallelMatlabsurvey.php>
7. *MPI Toolbox for MATLAB*.
http://atc.ugr.es/javier-bin/mpitb_eng

8. *LAM MPI Homepage.*
www.lam-mpi.org
9. *MPICH2 Homepage.*
<http://www-unix.mcs.anl.gov/mpi/mpich2/>
10. *MPITB for Windows.*
<http://www.wakun.com/download.htm>
11. *Akhter Sh., Roberts J. Multi-Core Programming. Increasing Performance through Software Multi-threading.* Intel Press. 2006. 344 p.
12. *Оленев Н.Н. Основы параллельного программирования в системе MPI.* М.: ВЦ РАН. 2005. 80 с.
13. *Стандарт MPI-2.* <http://www.mpi-forum.org/docs/mpi-20-html/mpi2-report.html>
14. *OpenPBS.* <http://www.openpbs.org>
15. *Sun Grid Engine.* <http://gridengine.sunsource.net>
16. *Голуб Дж., Ван Лоун Ч. Матричные вычисления.* М.: Мир, 1999. 548 с.
17. *Михайлов Г.М., Копытов М.А., Рогов Ю.П., Самоваров О.И., Чернецов А.М. Параллельные вычислительные системы в локальной сети ВЦ РАН.* М.: ВЦ РАН, 2003. 75 с.
18. *Михайлов Г.М., Оленев Н.Н., Петров А.А., Рогов Ю.П., Чернецов А.М. Опыт использования кластера ВЦ РАН в образовательных целях.//Высокопроизводительные параллельные вычисления на кластерных системах. Материалы V межд. научно-практ. сем. Нижний Новгород, 2005. С.170-175.*
19. *Bug Reports - 306262.* <http://www.mathworks.com/support/bugreports/details.html?rp=306262>

20. *Ramsey F.P.* A mathematical theory of saving. The Economic Journal, Vol. 38, No. 152. (Dec., 1928). PP. 543-559.
21. *Интрилигатор М.* Математические методы оптимизации и экономическая теория. М.: Айрис-пресс, 2002. 576 с.
22. *Моисеев Н.Н.* Простейшие математические модели экономического прогнозирования. М.: Знание, 1975.
23. *Иванчиков Ю.П., Лотов А.В.* Математические модели в экономике. М.: Наука, 1979. 304 с.
24. *Ашманов С.А.* Введение в математическую экономику. М.: Наука, 1984. 296 с.
25. *Weitzman M. L.* Soviet Postwar Economic Growth and Capital-Labor Substitution. The American Economic Review, Vol. 60, No. 4. (Sep., 1970). PP. 676-692.
26. *Оленев Н.Н.* Параллельные вычисления для идентификации параметров в моделях экономики. //Высокопроизводительные параллельные вычисления на кластерных системах. Мат. IV межд. научно-практ. сем. Самара, 2004. С. 204-209.
27. *Фишер С., Дорнбуш Р., Шмалензи Р.* Экономика: Пер. с англ. со 2-го изд. М.: Дело ЛТД, 1993. 864 с.
28. *Столерю Л.* Равновесие и экономический рост (принципы макроэкономического анализа): Пер. с франц. М.: Статистика, 1974. 472 с.
29. *Бурнаев Е.В., Оленев Н.Н.* Меры близости на основе вейвлет-коэффициентов для сравнения статистических и расчетных временных рядов. //Межвуз. сб. научн. и научно-методич. тр. за 2005 год (10 выпуск). Киров: Изд-во ВятГУ, 2006. С. 41-51.

30. *Оленев Н.Н.* Параллельные вычисления для оценки параметров динамической многосекторной балансовой модели региональной экономики.//Научный сервис в сети ИНТЕРНЕТ: технологии параллельного программирования: Тр. Всерос. научн. конф. М.: Изд-во МГУ, 2006. С. 36-37.
31. *Оленев Н.Н.* Параллельные вычисления в математическом моделировании региональной экономики.//Параллельные вычислительные технологии - 2007. Тр. I междун. научн. конф. Челябинск: Изд-во ЮУрГУ, 2007. Т.1. С.140-151.
32. *Тейл Г.* Экономические прогнозы и принятие решений. М., 1971. 488 с.
33. *Olenev N.* Production Function of Skilled and Unskilled Labour in a Model of a Non-Growing Russian Economy.//International Labour Market Conference Proceedings. - Aberdeen: Robert Gordon University, 1999. PP. 560-575.
34. *Burnaev E.V., Olenev N.N., Starikov A.S.* Parameter estimation of a macroeconomic model // Proc. of the Vth Moscow Intern. Conf. on Oper. Res. (ORM2007), dedic. to the outst. Russian scientist N.N. Moiseev 90th birthday. 2007. Moscow. PP.71-73.
35. *Петров А.А., Поспелов И.Г., Шананин А.А.* Опыт математического моделирования экономики. М.: Энергоатомиздат, 1996. 544 с.
36. *Нивергельт Ю., Фаррар Дж., Рейнголд Э.* Машинный подход к решению математических задач. М.: Мир, 1977. 352 с.
37. *Гергель В.П., Стронгин Р.Г.* Параллельные методы вычисления для поиска глобально-оптимальных решений// Высокопроизводительные параллельные вычисления на кластерных системах. Мат. 4-го межд. науч.-практ. сем. Самара, 2004. С. 54-59.