

УДК 533.6:519.6

**С.В. ЕРШОВ**, д-р техн. наук; проф. ИПМаш НАН України, Харків;  
**В.А. ЯКОВЛЕВ**, канд. техн. наук; н.с. ИПМаш НАН України, Харків;  
**Д.А. КОЗЫРЕЦ**, аспірант НТУ «ХПИ»

## РАСПАРАЛЛЕЛИВАНИЕ ВЫЧИСЛЕНИЙ ПРИ РАСЧЕТЕ ТЕЧЕНИЙ ВЯЗКОГО ГАЗА В ТУРБОМАШИНАХ

Статья посвящена вопросам распараллеливания вычислений при расчетах трехмерных турбулентных течений вязкого газа в многоступенчатых турбинах и компрессорах. Рассматривается *CFD* решатель *F*, предназначенный для решения таких задач. Распараллеливание выполнено применительно к многопроцессорным вычислительным системам с распределенной и общей памятью. Программная реализация использует межпроцессорный обмен информацией, построенный на основе сокет-интерфейса. Для этой цели привлекаются функции *Windows Socket API*. Тестирование разработанного алгоритма показало его высокую эффективность. Рассмотрены направления дальнейшей работы по совершенствованию алгоритмов распараллеливания.

**Ключевые слова:** распараллеливание, сокет-интерфейс, *CFD* решатель, многоблочная область, турбомашины.

### Введение

Общей тенденцией развития современных *CFD*-решателей является использования параллельных вычислений для ускорения расчетов. Распараллеливание выполняется на вычислительных системах с общей и распределенной памятью, на центральных и графических процессорах. Каждый из этих видов распараллеливания имеет свои особенности и свою специфику реализации. Для распараллеливания обычно используются специальные библиотеки и языки, такие как *MPI*, *PVM*, *Cuda*, *OpenCL* и др. Можно найти большое количество публикаций на эту тему, например [1–6].

В настоящей статье рассмотрено распараллеливание вычислений в *CFD*-решателе *F* [7], разработанном на основе кода *FlowER* [8]. Распараллеливание выполнено для вычислительных систем с общей и распределенной памятью с помощью межпроцессорного обмена информацией, построенного на основе сокет-интерфейса. Библиотеки параллельных вычислений типа *MPI* при этом не применялись.

### Краткое описание *CFD*-решателя

Трехмерное турбулентное течение вязкого теплопроводного сжимаемого газа в проточной части многоступенчатой турбомашины: турбины или компрессора – описывается системой нелинейных уравнений Навье-Стокса, осредненных по Рейнольдсу. Используется семейство дифференциальных моделей турбулентности семейства *k- $\omega$*  [9]. Исходные уравнения интегрируются численно с помощью итерационной явно-неявной разностной схемы, явный оператор которой основан на *ENO* схеме Годунова [10], а неявная аппроксимация реализована с помощью итерационного метода Ньютона [11] и схемы Бима-Уорминга-Стегера [12].

Расчетная область имеет многоблочную структуру. Каждый блок включает в себя область течения в одном межлопаточном канале одного венца – направляющего (спрямляющего) аппарата или рабочего колеса. Обмен информацией между блоками в исходном нераспараллеленном коде осуществляется на общих границах этих блоков одномерными массивами осредненных в окружном направлении газодинамических параметров (давление, плотность, компоненты скорости, параметры турбулентности).

---

© С.В. Ершов, В.А. Яковлев, Д.А. Козырец, 2013

При этом на каждой итерации расчета информация с выхода некоторого лопаточного венца передается на вход следующего венца и, наоборот, с входа текущего венца передается на выход предыдущего. Алгоритм реализован таким образом, что начало следующей итерации вычислительного процесса возможно только после завершения этого обмена данными. В противном случае на одном этапе вычислений использовались бы данные с различных итераций, что может негативно сказаться как на скорости сходимости решения, так и на его устойчивости.

Программные модули *CFD*-решателя *F* написаны на алгоритмическом языке *Fortran-95* и включают в себя, кроме собственно решателя, препроцессор для подготовки данных, модули расчета координат сеточных узлов, характеристик сетки, построения начального приближения и постпроцессор для обработки результатов расчетов. Графические возможности программной оболочки реализованы с помощью средств *GUI Windows API*, в то время как сам решатель использует стандарт алгоритмического языка и может работать в других операционных системах.

#### **Реализация распараллеливания вычислений для многоблочных областей**

Расчетная область многоступенчатой турбомашин имеет многоблочную структуру и для каждого из блоков необходимо проводить однотипные вычисления. Как правило, количество таких блоков (лопаточных венцов) может варьироваться от одного до двух-трех десятков. В многопроцессорной системе вычисления для каждого блока логично поручить одному процессору (ядру или потоку). Причем, это может быть как система с общей памятью, так и с распределенной. Если доступных процессоров меньше, чем лопаточных венцов, то имеет смысл в каждый блок включать по несколько лопаточных венцов. Так как время расчета для каждого блока будет определяться, в основном, двумя факторами – размерностью разностной сетки в блоке и производительностью процессора, то при распределении лопаточных венцов по блокам и выборе количества узлов в каждом из них, необходимо это учитывать таким образом, чтобы исключить неравномерную загруженность процессоров. В программной оболочке пакета *F* реализовано автоматическое распределение вычислительной работы между выбранными процессорами многопроцессорной системы.

Каждый процесс идентифицируется номером, который определяет, для какого блока расчетной области выполняются вычисления. Во избежание конфликтов при одновременном открытии файлов на чтение (перед выполнением вычислений) и запись (после окончания вычислений) эти операции для всех процессов выполняются последовательно путем передачи-приема разрешений. В итерационном цикле перед началом вычислений каждый процесс отправляет данные с границ блока соответствующей расчетной области, являющихся общими с блоками других процессов, а затем принимает информацию для этих же границ, отправленную другими процессами. Так как операция получения информации по сети может быть реализована как блокирующая (т.е. выполнение программы приостанавливается до окончания этой операции), то при правильной организации такого обмена не требуется никакого другого согласования процессов, и возникновение ситуаций взаимного блокирования процессов или рассинхронизации вычислений невозможно.

Реализация параллельных вычислений в решателе *F* выполнена с помощью дополнительных процедур, написанных на алгоритмическом языке *Fortran-95* с привлечением функций *WinSock* интерфейса прикладного программирования *API* [13, 14]. По мнению авторов настоящей статьи, непосредственное использование сокет-

интерфейса вместо стандартных библиотек типа *MPI* позволит достичь более высокой эффективности распараллеливания.

#### **Организация удаленного запуска параллельно выполняемых задач**

Программный комплекс *F* позволяет расчетчику автоматически выполнить локальный запуск нераспараллеленного задания на процессоре того компьютера, на котором запущена управляющая оболочка. Для распараллеленных заданий необходима возможность удаленного запуска. В частности, при расчетах на распределенной вычислительной системе отдельные ее компьютеры могут одновременно использоваться для выполнения других вычислительных задач. Если же параллельные вычисления проводятся на удаленной многопроцессорной системе с общей памятью, то работа через терминал не всегда удобна и даже возможна, если, например, удаленная система использует операционную систему *Linux*. В связи с этим в комплексе программ *F* реализован удаленный запуск параллельно выполняющихся расчетных заданий. На каждом удаленном компьютере, подготовленном для такого запуска, работает служба *ParallelService*, которая предназначена для:

- связи с программами-клиентами, с которых удаленно запускаются задания;
- передаче программам-клиентам информации об удаленном компьютере, а именно: количество процессоров (ядер или потоков), их производительность и доступность;
- передаче программам-клиентам информации о запущенных на удаленном компьютере процессах (кто запустил и когда);
- копирования необходимых для расчетов данных (файлов базы данных) с компьютера программы-клиента на удаленный компьютер и обратно;
- выполнение команд запуска и останова заданий на удаленном компьютере.

Служба *ParallelService* и программы-клиенты написаны на языке *Fortran-95* с использованием функций *Windows Socket API*.

#### **Тестирование алгоритма**

Тестирование разработанного алгоритма выполнялось на ПК с процессором Intel i7 3770K, который имеет 4 ядра и, благодаря технологии *hyper-threading*, может реализовать 8 потоков. Рассчитывалось течение в проточной части компрессора, содержащей восемь венцов. В расчетной области была построена трехмерная разностная сетка с 10 616 832 (8×96×96×144) ячеек – более 1 300 000 ячеек в венце.

Результаты расчета течения в компрессоре приведены на рис. 1 и 2. Распределение осредненного в окружном направлении давления по проточной части компрессора показано на рис. 1, а изолинии числа Маха в средних тангенциальных сечениях лопаточных венцов компрессора даны на рис. 2.

Выполнено две серии тестов для оценки эффективности распараллеливания. В первой из них нераспараллеленный расчет в одном потоке и распараллеливание на двух и четырех потоках проводились с выключенной технологией *hyper-threading*. Во второй нераспараллеленный расчет и распараллеливание на два, четыре и восемь потоков были реализованы с использованием технологии *hyper-threading*. В последнем случае для рассматриваемого процессора скорость выполнения одного процесса на рассматриваемых задачах снижается приблизительно в полтора раза, поэтому результаты первой и второй серии не могут быть сопоставлены непосредственно. В тестах обеспечивалась полная загруженность всех потоков независимо от количества запускаемых параллельных процессов (при необходимости – запуском дополнительных «фантомных» процессов).

Полученные результаты для обеих серий тестов приведены в таблицах 1 и 2. Ускорение распараллеливания рассчитывалась как

$$S_p = T_1/T_p,$$

где  $T_1$  – время исполнения нераспараллеленной задачи,  $T_p$  – время исполнения задачи при распараллеливании на  $p$  процессорах. Идеальное время распараллеливания определялось как

$$T_{ид} = T_1/p.$$

Эффективность использования процессоров (загруженность распараллеливания) находилась как отношение идеального времени распараллеливания к реальному:

$$\eta_p = T_{ид}/T_p = T_1/pT_p.$$

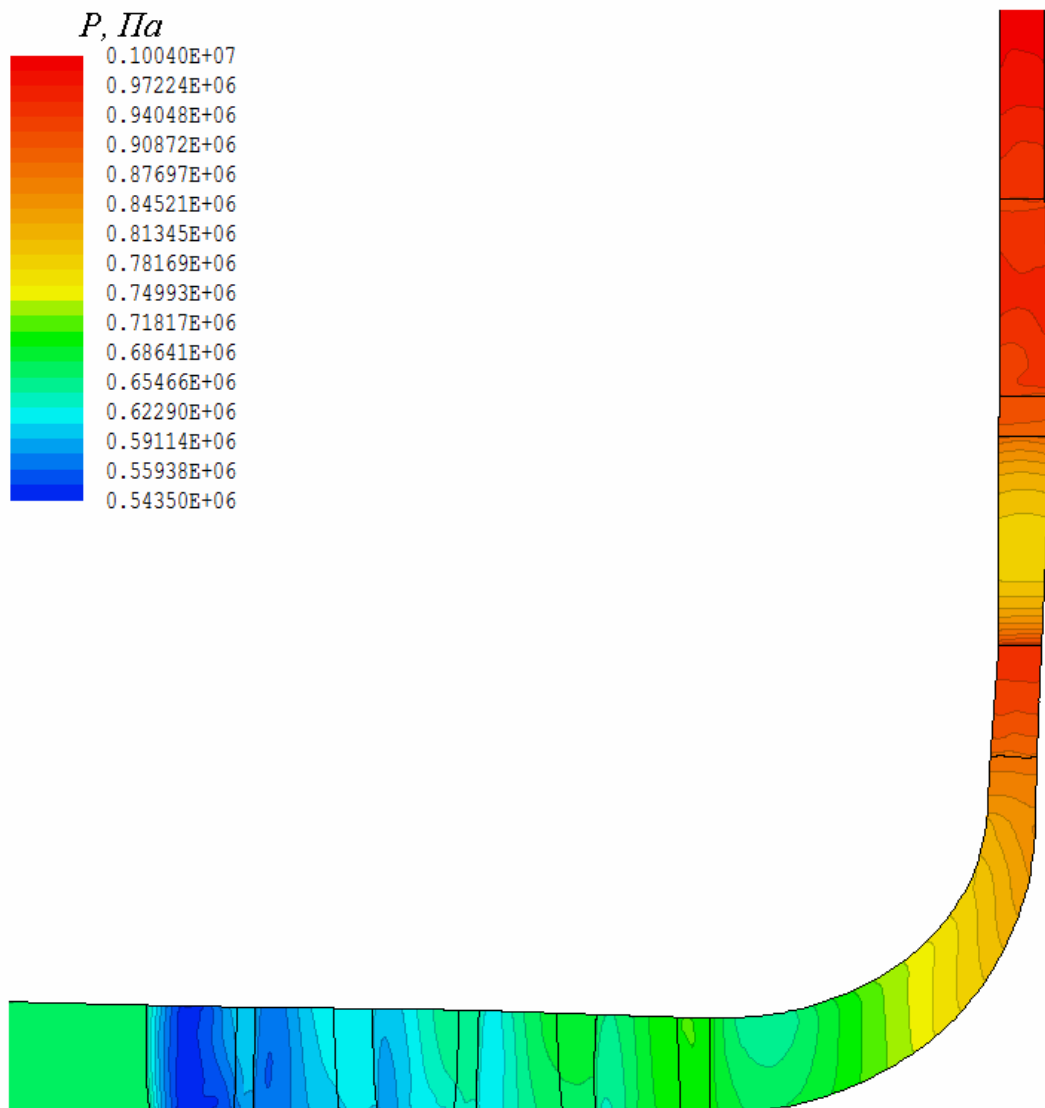


Рис. 1 – Изолинии осредненного в окружном направлении давления в проточной части компрессора

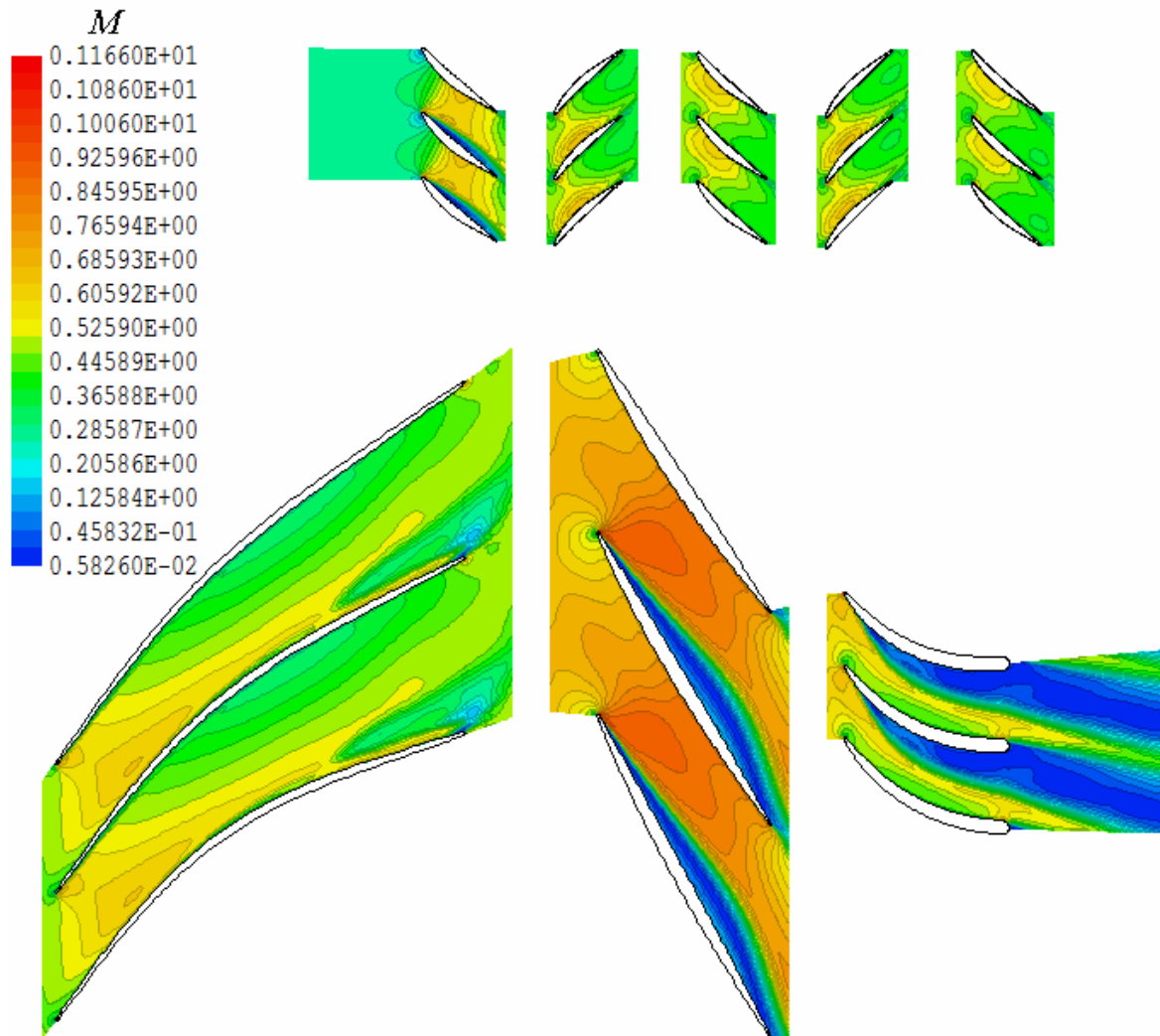


Рис. 2 – Изолинии числа Маха в средних тангенциальных сечениях лопаточных венцов компрессора

Так как по процессорному времени эффективность распараллеливания оказывается близкой к 100 %, то основное внимание уделялось сопоставлению астрономического времени выполнения задач.

График эффективности распараллеливания для обеих серий численных экспериментов приведен на рис. 3. Следует отметить, что в настоящей работе для решателя  $F$  получено более высокое ускорение расчетов, чем было достигнуто ранее [15] для комплекса программ *FlowER*. Это связано с тем, что, с одной стороны, в новой технологии распараллеливания исполняемые процессы непосредственно обмениваются информацией друг с другом, не используя никаких дополнительно запускаемых программ-посредников, а, с другой стороны, реализован более компактный способ кодирования пересылаемых данных.

Таблица 1  
Эффективность распараллеливания (без *hyper-threading*)

Количество процессов	Время одной итерации, с	Ускорение	Эффективность
1	60,3	1,00	1,00
2	30,9	1,95	0,98
4	15,9	3,79	0,95

Эффективность распараллеливания (с *hyper-threading*)

Количество процессов	Время одной итерации, с	Ускорение	Эффективность
1	94,4	1,00	1,00
2	48,0	1,97	0,98
4	24,2	3,91	0,98
8	12,2	7,73	0,97

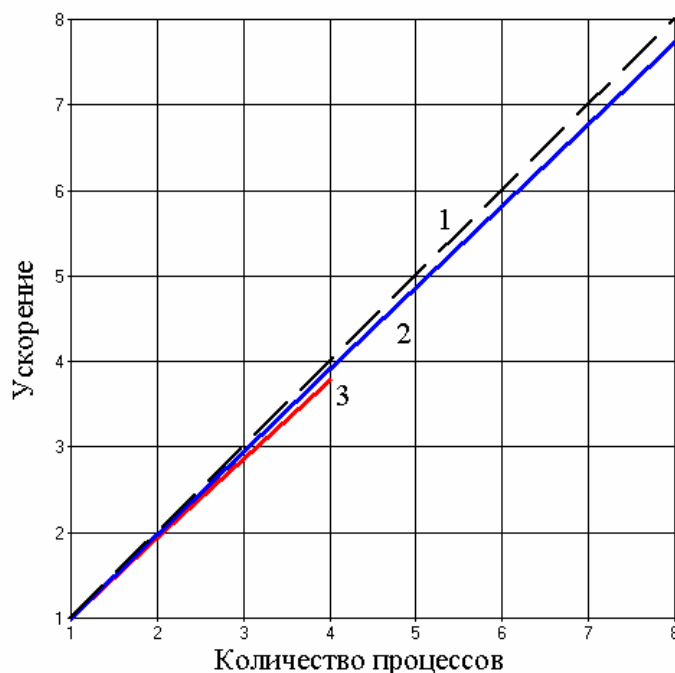


Рис. 3 – Эффективность распараллеливания:

1 – идеальное распараллеливание; 2 – с *hyper-threading*; 3 – без *hyper-threading*

### Перспективные направления дальнейшей работы

Ближайшим заданием дальнейшей работы является адаптация разработанного программного обеспечения для функционирования в ОС типа *Linux*. Учитывая тот факт, что программирование сокетов в разных операционных системах выполняется подобным образом, то очевидно, что доработка разработанных программ для функционирования в ОС *Linux* не вызовет принципиальных трудностей.

Весьма перспективным представляется применение внутреннего многопоточного распараллеливания для использования возможностей современных графических процессоров. Решение такой задачи, требующее серьезной реорганизации программного кода, должно позволить достичь ускорения получения решений почти на два порядка, что является необходимым для включения *CFD* решателя в программные системы оптимизации проточных частей турбомашин.

### Выводы

Для *CFD*-решателя *F* реализовано распараллеливание вычислений на многопроцессорных системах с распределенной и общей памятью. Разработанный алгоритм основан на использовании функций *Windows Socket* интерфейса прикладного программирования *API*. Тестирование распараллеливания показало его высокую эффективность.

**Список літератури:** **1.** Gao, X. Parallel Adaptive Mesh Refinement Scheme for Three-Dimensional Turbulent Non-Premixed Combustion [Text] / X. Gao, C.P.T. Groth // AIAA Paper – 2008. – № 2008-1017. – 18 p. **2.** Bush R.H. WIND: The Production Flow Solver of the NPARC Alliance [Text] / R.H. Bush, G.D. Power, C.E. Towne // AIAA Paper. – 1998. – № 98-0935. – 14 p. **3.** Antoniou, A.S. Acceleration of a Finite-Difference WENO Scheme for Large-Scale Simulations on Many-Core Architectures [Text] / A.S. Antoniou, K.L. Karantasis, E.D. Polychronopoulos, J.A. Ekaterinaris // AIAA Paper. – 2010. – № 2010-0525. – 12 p. **4.** Niemeyer, K.E. Accelerating reactive-flow simulations using graphics processing units [Text] / K.E. Niemeyer, C.-J. Sung // 51st AIAA Aerospace Sciences Meeting, Grapevine, Texas, USA. 6–10 January 2013. – P. 1-13. **5.** Brandvik, T. Acceleration of a two-dimensional Euler flow solver using commodity graphics hardware [Text] / T. Brandvik, G. Pullan // Journal of Mechanical Engineering Science. – 2007. – V.221, № 12. – P.1745-1748. **6.** Schalkwijk, J. High-Performance Simulations of Turbulent Clouds on a Desktop PC. Exploiting the GPU [Text] / J. Schalkwijk, E.J. Griffith, F.H. Post, H.J.J. Jonker // Bulletin of the American Meteorological Society. – 2012. – V. 93, № 3. – P. 307-314. **7.** Ершов, С.В. Развитие комплекса программ расчета трехмерных течений вязкого сжимаемого газа в лопаточных аппаратах турбомашин [Текст] / С.В. Ершов, В.А. Яковлев, А.И. Деревянко, М.Н. Гризун, Д.А. Козырец // Энергетичні та теплотехнічні процеси й устаткування. Вестник НТУ «ХПІ». Зб. наук. праць. – 2011. – № 5. – С. 25-32. – ISSN 2078-774X **8.** Ершов, С.В. Комплекс програм розрахунку тривимірних течій газу в багатовісцевих турбомашинах «FlowER» / С.В. Ершов, А.В. Русанов: Свідоцтво про державну реєстрацію прав автора на твір, ПА № 77. Державне агентство України з авторських та суміжних прав, 19.02.1996. **9.** Wilcox, D.C. Turbulence Modeling for CFD [Text] / D.C. Wilcox. – Second Edition. Palm Drive: DCW Industries. – 2004. – 540 p. **10.** Ершов, С.В. Квазімонотонна схема підвищеної точності для інтегрування рівнянь Ейлера і Нав'є-Стокса [Текст] / С.В. Ершов // Мат. моделювання. – 1994. – Т. 6, № 11. – С. 63-75. **11.** Ершов, С.В. Метод Ньютона для неявної схеми численного інтегрування рівнянь газової динаміки [Текст] / С.В. Ершов, А.И. Деревянко, М.Н. Гризун // Проблеми машиностроєння. – 2010. – Т. 13, № 4. – С. 27-36. – ISSN 0131-2928. **12.** Beam, R.M. An implicit factored scheme for the compressible Navier-Stokes equations [Text] / R.M. Beam, R.F. Warming // AIAA Journal. – 1978. – V. 16, № 4. – P. 393-402. **13.** Lawrence, N. Compaq Visual Fortran: A guide to creating windows applications [Text] / N. Lawrence. – Elsevier Digital Press. – 2002. – 534 p. **14.** Bonner, P. Network Programming with Windows Sockets [Text] / P. Bonner. – Prentice Hall. Englewood Cliffs. New Jersey. – 1995. – 328 p. **15.** Ершов, С.В. Распараллеливание вычислений при расчете трехмерных вязких течений в многоступенчатых турбомашинах [Текст] / С.В. Ершов, А.В. Русанов, Д.С. Ершов // Пробл. машиностроєння. – 2002. – Т. 5, N 3. – С. 3-8.

*Поступила в редколлегию 18.01.13*

УДК 533.6:519.6

**Распараллеливание вычислений при расчете течений вязкого газа в турбомашинах [Текст]** / С.В. Ершов, В.А. Яковлев, Д.А. Козырец // Вісник НТУ «ХПІ». Серія: Енергетичні та теплотехнічні процеси й устаткування. – Х.: НТУ «ХПІ», 2013. – № 12(986). – С. 10-16. – Бібліогр.: 15 назв. – ISSN 2078-774X.

Стаття присвячена питанням розпаралелювання обчислень при розрахунках тривимірних турбулентних течій в'язкого газу в багатоступеневих турбінах та компресорах. Розглядається CFD-розв'язувач *F*, що є призначеним для розв'язування таких задач. Розпаралелювання виконано стосовно до багатопроцесорних обчислювальних систем з розподіленою та спільною пам'яттю. Програмна реалізація використовує міжпроцесорний обмін інформацією, що побудовано на основі сокет-інтерфейсу. Для цього залучені функції *Windows Socket API*. Тестування розробленого алгоритму показало його високу ефективність. Розглянуто напрямки подальшої роботи щодо удосконалення алгоритмів розпаралелювання.

**Ключові слова:** розпаралелювання, сокет-інтерфейс, CFD розв'язувач, багатоблочна область, турбомашини.

The paper deals with the parallelisation of 3D turbulent viscous flow computations for multi-stage turbines and compressors. The CFD solver *F* is considered for this goals. Parallelisation is performed for computer systems with distributed and shared memory. Software implementation uses interprocessor communication, based on socket interface. The *Windows Socket API* functions are exploited for that. Testing of the developed parallel algorithm shows its high efficiency. Possible lines of further improvement of parallel algorithm are considered.

**Keywords:** parallelisation, socket-interface, CFD solver, multi-block domain, turbomachinery.