Anes Team

ANES20XE: Код для численного моделирования процессов гидродинамики и тепломассообмена

Версия 2.24

Настройка и работа с кодом

Москва 2019 г.

Оглавление.

1. Установка и настройка кода

1.1 Общие замечания

Код Anes20xe предназначен для работы в операционных системах Windows, Linux и кластерах под управлением Windows или Linux. Главные особенности архитектуры текшей кода:

1) в Решателе кода реализованы параллельные вычисления на основе MPI-интерфейса;
2) пакет адаптирован для работы в ОС Linux и на кластерах;
3) наряду со структурными сетками реализованы неструктурные декартовые сетки с локальным дроблением;
4) для облегчения описания задачи пользователя реализованы подсистемы myForm/myTable, позволяющие в большинстве случаев отказаться от использования Фортран-интерфейса Решателя (написания своих подпрограмм на Фортране).
5) для анализа результатов расчетов, наряду с собственным постпроцессором, можно использовать универсальный постпроцессор ParaView.

После инсталляции код можно использовать без установки дополнительных программных продуктов. Вместе с самим кодом инсталлятор устанавливает:

1) 32/64-разрядную GNU-версию Фортрана из пакета GNU/GCC, необходимую для генерации локальной версии Решателя (в текущей версии кода - это gFortran версии 4.7);
2) библиотеки поддержки MPI-подсистемы MS-MPich2, необходимые для проведения последовательных расчетов;
3) текстовый редактор SciTE для редактирования текстовых файлов.

Если пользователь желает работать с параллельной версией Решателя, то необходимо установить пакет «HPC Pack 2012 MS-MPI Redistributable Package» (или более поздний) фирмы Микрософт для поддержки MPI-интерфейса.

В составе кода имеется встроенный постпроцессор для обработки результатов расчета. Для дополнительной обработки результатов можно использовать универсальной постпроцессор ParaView (www.paraview.org).

Для работы с удаленным Linux-кластером необходимы коммутационные программы. Рекомендуется использовать GNU-версии двух пакетов:

1) Putty – программа удаленной консоли.
2) FileZilla – программа для обмена файлами с удаленным компьютером.

Ниже основное внимание уделяется работе с кодом в операционной системе Windows. Работа с системой Linux описана в главе XX.

1.2 Структура каталогов кода

Код Anes устанавливается в отдельный каталог, который далее будет обозначаться как <Anes> и называться корневым каталогом кода. По умолчанию инсталлятор использует каталог

<Anes> = c:\anes20xe

Инсталлятор в корневом каталоге создает следующую структуру подкаталогов:

bin	- исполняемые модули и утилиты пакета,
_alua	- LUA-версия оболочки Anes,
bin\gfor или bin\gfor64	- стандартные Решатель и А-компилятор,
bin\lua	- интерпретатор LUA,
script	- файлы с исполняемыми скриптами пакета,
doc	- документация,
lib\gfor или lib\gfor64	- библиотеки для сборки локального Решателя,
lib\com_32 ; lib\com_64	- общие библиотеки для сборки локального Решателя ,
etc	- файлы настройки пакета,
project_lib	- библиотека примеров пакета,
work_win	- каталог для проведения расчетов из библиотеки примеров,
SciTE2010	- текстовый редактор SciTE,
gfor-rt	- 32-разрядный компилятор gFortran,
gfor-rt64	- 64-разрядный компилятор gFortran,
geoobj	- файлы с описанием ГеоОбъектов,
property	- база данных свойств материалов,
update	- каталог для скачивания обновлений кода.

1.3 Инсталяция и обновление пакета

Пакет Anes устанавливается с помощью дистрибутива, оформленного в виде exe-файла. Необходимые системные set-переменные (ANES20XE, ANESOS, ANESUSER) инсталлятор устанавливает только для текущего пользователя. Для работы других пользователей необходимо в их сеансе установить вручную.

Для обновления кода используется следующий алгоритм:

1) Обновления разработчиками выкладываются на на ftp-сервер Anes. В ближайшее время для этого будет использоваться сайт кода Anes.
2) Эти обновления оформлены в виде zip-файлов. Для использования обновлений их нужно загрузить и скопировать в каталог update кода Anes.
3) Для установки обновления нужно запустить скрипт обновления aUpdate.bat из каталога script. В меню «Пуск» для этого сделан специальный пункт меню «Обновление Anes».
4) Скрипт aUpdate.bat можно запускать много раз. Он «помнит», какие обновления установлены, и в каком порядке их нужно устанавливать.
5) Какие обновления скачаны и установлены можно просмотреть в оболочке кода – пункт меню «Подсказка/Обновление кода».

1.4 Настройка пакета

Для настройки пакета используются следующие файлы:

etc\anes.acfg	- основной файл настройки пакета,
etc\locale.a	- файл с выбором языка сообщений программ Anes,
script\ComSol_gfor.bat , script\ComSol_gfor64.bat	- скрипты для сборки локального Решателя для разных компиляторов Фортрана,
script\gfor32vars.bat , script\gfor64vars.bat	- скрипты для настройки путей компилятора gFortran.

1.4.1 Файл anes.acfg

Для настройки компонентов Anes используется две группы так называемых SET-переменных. К первой группе относятся переменные окружения операционной системы (системные SET-переменные Windows или переменные окружения Linux), ко второй – собственные SET-переменные пакета. Для корректной работы Anes необходимо установить три *системные* переменные:

```
set ANES20XE=<Anes>
set ANESOS=windows
set ANESUSER=<Anes>
```

Инсталлятор автоматически устанавливает эти переменные для текущего пользователя Windows. «Повторяющиеся» переменные ANES20XE и ANESUSER при однопользовательском режиме работы совпадают. Они используются в многопользовательском Linux для разделения HOME-каталогов с кодом Anes и рабочими файлами пользователя.

☞Замечание. В текущей версии кода диалоговые компоненты переписаны с помощью системы программирования Lazarus. Очень редко возникают проблемы сохранения файлов настройки для пользователей Windows, логин которых содержит русские буквы. В этом случае нужно добавить новую SET-переменную:

```
ANES_LOCINI = T
```

Для настройки *собственных* SET-переменных используется файл etc\anes.acfg. Структура этого файла аналогична структуре файлов настройки ОС Linux:

1) это обычный текстовый файл;
2) все символы, расположенные за символами ‘!’ или '#' считаются комментариями.
3) операторы файла предназначены для присвоения символьного значения SET-переменной пакета; эти операторы имеют стандартную структуру:
   <ИмяПеременной> := <Значение>
   при обработке операторов начальные и конечные пробелы в имени и значении игнорируются;
4) в подстроке <Значение> можно использовать имена, как собственных SET-переменных пакета, так и системных в виде:
   $(<ИмяПеременной>)
   в процессе присвоения вместо этого имени подставляется значение переменной.

Смысл основных собственных SET-переменных:

rBDir	- имя подкаталога (gfor/gfor64) для выбора Решателя и A-компилятора по умолчанию;
rCDir	- имя подкаталога (gfor/gfor64) для выбора библиотек, используемых для компиляции и сборки (*генерации*) локального Решателя;
rAComp	- путь к Компилятору Anes по умолчанию;
rSolver	- путь к Решателю по умолчанию;
rMpich	- путь к модулю mpiexec пакета Mpich;
rFComp	- путь к скрипту для генерации локального Решателя;
rFLib	- тип библиотек, используемых для генерации локального Решателя: release – оптимизированные библиотеки, debug – библиотеки с отладочной информацией;
sEdit	- путь к текстовому редактору, используемому в оболочке Anes;
sParaView	- путь к постпроцессору ParaView;
sPreAnes	- путь к диалоговому препроцессору Anes;
sPostAnes	- путь к постпроцессору Anes.

Некоторые замечания:

1. При решении большинства задач не нужно изменять значение переменной rFComp. В этом случае для генерации локального Решателя используется скрипт script\ComSol_gfor.bat или script\ComSol_gfor64.bat. Если пользователь подключает к задаче свои библиотеки, то нужно создать на основе скрипта ComSol_gfor.bat свой собственный скрипт и путь к нему указать в переменной rFComp.
2. Anes использует следующий алгоритм поиска файла anes.acfg:
   - сначала файл ищется в текущем каталоге WorkDir (что это такое – смотрите ниже),
   - если такого *локального* файла нет, то используется *глобальный* файл – etc\anes.acfg.

Приведем «стандартный» вид файла anes.acfg для ОС Windows:

```
#=========================================
# Anes 20XE: Файл конфигурации системы
#   Оформляется по правилам  bash/make
#   Это файл настройки для WINDOWS
#=========================================
#=========================================
# Секция: Параметры настройки для aSolverRun
#=========================================
# --- подкаталоги в BIN для вызова решателя по умолчанию  ---
 rBdir  := gfor
 rCDir  := gfor
# --- путь к решателю по умолчанию ----
 rSolver := $(ANES20XE)\bin\$(rBdir)\arSolver20.exe
# --- путь к компилятору A-файла  ----
 rAcomp := $(ANES20XE)\bin\$(rBdir)\arCompiler20.exe
# --- путь к mpiexec  ----
 rMpich := C:\Program Files\Microsoft MPI\bin\mpiexec
# --- пути к Скрипту компиляции  ----
 rFComp := $(ANES20XE)\script\CompSol_$(rCdir).bat
# --- тип библиотеки (release/debug) ----
 rFlib := release
#-----------------------------------------
# Секция: Для работы оболочки aShell.exe
#-----------------------------------------
# --- путь к редактору  ----
 sEdit := $(ANES20XE)\sciTE2010\SciTE.exe
# --- путь к ParaView  ----
 sParaView := C:\Program Files (x86)\ParaView 4.0.1\bin\paraview.exe
# --- путь к aVerify/aShaper2D  ----
 sVerify   := $(ANES20XE)\bin\aVerify.exe
sShaper2D   := $(ANES20XE)\bin\aShaper2D.exe
# --- Пре/Пост процессора  ----
 sPreAnes   := $(ANES20XE)\bin\aPre20.exe
 sPostAnes  := $(ANES20XE)\bin\aPost20.exe
```

1.4.2 Файл gfor32vars.bat/ gfor64vars.bat

Эти файлы используются для настройки параметров окружения компиляторов Фортрана. Если используется версия Фортрана по умолчанию – gFortran, то изменять этот файл не нужно. Если пользователь будет использовать свои версии Фортранов, то нужно исправить эти файлы. Эти файлы используются в скриптах script\ComSol_gfor.bat и ComSol_gfor64.bat.

1.4.3 Локализация языкового интерфейса Anes

Код Anes позволяет использовать многоязычный интерфейс как для сообщений A-компилятора, Решателя, пакетного Постпроцессора, так и для всех диалоговых программ.

Выбор языка определяется содержимым файла etc\locale.a

Файл содержит две строчки вида:

```
language = xx
LuaShell = xx
```

где xx – двухсимвольный код языка (en, ru). Если файл отсутствует, то по умолчанию используется русский интерфейс. Оператор LuaShell используется для работы LUA-оболочки Anes (файл bin\luashell.bat).

Тексты всех сообщений А-компилятора, Решателя и Постпроцессора расположены в файлах languages\message_xx.a. В состав пакета входят две заготовки этого файла:

languages\message_ru.a - заготовка файла сообщений на русском языке.
languages\message_en.a - заготовка файла сообщений на английском языке.

Тексты сообщений и элементов диалоговых программ хранятся в файлах локализации Lazarus
languages\lazarus\<программа>_xx.po

Все эти файлы являются обычными текстовыми файлами. Поэтому не составляет труда «перевести» Anes на другие языки.

2. Этапы решения прикладной задачи

Процесс решения прикладной задачи пользователя состоит из трех этапов:

1) подготовка файла описания проекта - А-файла,
2) обработка А-файла и выполнение расчета,
3) просмотр и обработка результатов расчета.

Вся работа с пакетом Anes проводится с помощью программы «Оболочка Anes» bin\laShell.exe. Подробности работы с оболочкой описаны ниже.

2.1 Файл описания проекта

Файл проекта или А-файл представляет собой текстовый файл с произвольным именем и расширением ".a":

<PrjName>.a

В дальнейшем имя этого файла будем называть *именем проекта*. А-файл представляет собой секционный текстовый файл, состоящий из независимых блоков строк - секций. Секция начинается с заголовка секции, которая записывается, начиная с первой позиции строки

[<Имя секции>]

Конец секции - это либо конец файла, либо начало другой секции. В А-файле имена и число секций фиксированы и не могут быть изменены пользователем. Сам текст секции представляет собой набор операторов AIL *(*Anes Input Language*)*. Этот язык подробно описан в документе [1].

В текущей реализации версии Anes для подготовки и редактирования проекта можно использовать два способа:

1) текстовый редактор, заданный в файле конфигурации anes.acfg,
2) диалоговый препроцессор – Дизайнер проекта - bin\laPre20.exe.

Следует отметить, что оба этих способа совместимы. Единственное ограничение: после редактирования файла проекта в диалоговом препроцессоре может исчезнуть часть комментариев.

В данном документе описана работа с файлом проекта с помощью текстового редактора, работа с диалоговым препроцессором описана в документе [2].

2.2 Работа с файлом проекта

Оболочка Anes предлагает два пути работы с проектами:

1) работа с одиночным проектом,
2) использование файла *сессии* пользователя.

При использовании первого пути файл проекта помещается в *любой* каталог. Этот каталог будет называться *рабочим каталогом Anes* – WorkDir. При работе с проектом из оболочки Anes этот каталог всегда устанавливается в качестве *текущего* каталога Windows.

При использовании второго пути пользователь может работать сразу с группой файлов проектов, расположенных в произвольных местах. Для работы с ними используется один рабочий каталог WorkDir.

Для использования второго пути необходимо создать файл сессии с расширением *.ases. Этот файл представляет собой секционный текстовый файл следующей структуры:

```
!-----------------------------------------
! AnesNE: Файл описания библиотеки примеров проектов. Версия: WINDOWS.
!  Формат файла:
!  1) Путь к файлам проекта указывается либо относительно каталога с ДАННЫМ файлом,
!     либо указывается полный путь.
!  2) В путях можно использовать любые SET переменные (например, $(ANES20XE),
```

```
!      $(ANESUSER)...)
!  3) Формат записи для отдельного файла проекта:
!
!     <путь к A-файлу>#<Статус>#<Комментарий>
!
!     здесь стока <Статус> указывается цвет шрифта для строки файла в оболочке Anes:
!       RED, BLUE, GREEN, GRAY -  красный, синий, зеленый или серый
!     любой другой символ - черный цвет.
!  4) Допускается хранение проектов в двух форматах: Список и Дерево
!     Признак дерева - указание уровня секции в виде
!      [X#<Имя>]
!     где X - целое число 0 ... 9.
!----------------------------------------
[Main]
WorkDIR = $(ANESUSER)\work_win\

[ВЕРИФИКАЦИЯ]
VERIFY\e_solver.a # OK # Проверка внешнего солвера !
C:\anes20xe\VERIFY\u_vel_01.a # BLUE # Ламинарное обтекание цилиндра
```

В файле обязательно должна быть секция [Main] в которой указывается путь к *общему рабочему каталогу* WorkDir.

Все остальные секции рассматриваются как папки, в которых группируются проекты. Имена этих секций могут быть произвольными. Эти имена отображаются в левом окне оболочки Anes.

Файл проекта описывается с помощью трех «слов», разделенные символами ‘#’:

1) первое «слово» - путь к файлу проекта; допускается использовать как полные пути, так и относительные пути (относительно каталога с файлом .ases);
2) второе «слово» - цвет для отображения строки с проектом - RED, BLUE, GREEN, GRAY - красный, синий, зеленый или серый; все остальные символы – черный цвет шрифта;
3) третье «слово» - произвольный текст-комментарий.

Для работы с проектом достаточно выбрать его в левом окне Оболочки.

☞Замечание. При создании файла сессии с большим количеством проектов можно использовать представление файлов проекта не в виде списка, а в виде дерева (типичный пример – файл примеров project_lib\example-win.ases).

2.2.1 Обработка A-файла

После подготовки A-файла его необходимо скомпилировать с помощью программы-компилятора кода – A-компилятора (или Компилятора Anes). Вызов A-компилятора осуществляется *автоматически* из оболочки Anes. Компилятор на основе AIL операторов файла проекта создает две группы файлов:

1) файлы *исходных данных* для Решателя,
2) *фортрановские файлы* для создания локальной версии Решателя.

Файлы исходных данных создаются всегда. Фортрановские файлы создаются только, если пользователь в проекте использует:

- виртуальные или внешние функции (Virtual или External) для описания V-переменных,
- свои собственные функции-события.

Если создаются фортрановские файлы, то на втором этапе обработки A-файла производится генерация локальной версии Решателя, которая используется для проведения расчетов. В противном случае используется стандартная версия Решателя из каталогов bin.

Следует отметить, что выбор нужной версии Решателя (по умолчанию или локальной) осуществляется автоматически программой-оболочкой (для этого оболочка анализирует состав A-файла).

Имена и расположение файлов *исходных данных* определяются AIL-оператором:

PrefixResFiles = <PrefRes >

секции [Main] файла проекта. Здесь < PrefRes > - полный путь и префикс имен файлов Решателя (имя файлов варианта расчета без расширения).

В коде принято все данные расчета, включая исходные данные Решателя, размещать в подкаталоге data рабочего каталога проекта WorkDir. В этом случае префикс задается в виде

PrefixResFiles = data\conv

где conv – произвольное имя варианта расчета.

☞Замечание. Самая частая ошибка при создании своего первого проекта пользователя заключается в следующем:

1) пользователь скопировал пример из библиотеки примеров в свой каталог;
2) в этом каталоге нет подкаталога с именем data, который требует пример.

В этом случае при вызове А-компилятора будет выдана ошибка «Сбой при создании AID-файла».
Решение этой ошибки – создать подкаталог data.

В случае удачной компиляции в общем случае в указанном каталоге создаются следующие файлы, включающие в себя файлы *исходных данных* Решателя и файлы для программ - постпроцессоров:

<PreRes>.aclst	- листинг работы компилятора,
<PreRes>.aid	- файл инициализации переменных Решателя,
<PreRes>.agr	- файл с сеточной информацией,
<PreRes>.aphi	- параметры Ф-переменных для программы-постпроцессора Anes,
<PreRes>.ageo	- описание ГеоОбъектов проекта для программы-постпроцессора Anes,
<PreRes>.log	файл с отладочной информацией компилятора, создается, если включены флаги отладки генератора неструктурной сетки в секции [Unstructed Cartesian Grid],
<PreRes>_<имя патча>.stl	- stl-файл для патча <имя патча>, используется для построения патча в постпроцессоре ParaView; создается, если оператор секции [Main] IsSaveStlObj = .TRUE.
<PreRes>_cells.vtk	- сетка КО для ее анализа в постпроцессоре ParaView; создается, если оператор секции [Main] IsSaveVTKgrid = .TRUE.
<PreRes>_obj_faces.vtk	- сеточное представление патчей (грани КО, связанные с патчами) для анализа патчей в постпроцессоре ParaView; создается, если оператор секции [Main] IsSaveVTKgrid = .TRUE.
<PreRes>.apar	файл с информацией о декомпозиции расчетной области для неструктурной сетки,
<PreRes>.cswp ; <PreRes>.cinf	- файлы с информацией о процессе сходимости.

Количество создаваемых файлов зависит от значений специальных переменных-флагов проекта. Решатель для работы использует только три файла: *.agr, *.aid и *.apar.

Если необходимо создать локальную версию Решателя, А-компилятор в рабочем каталоге создает подкаталог exezz и помещает в него два фортрановских файла atempzz0.f90, atempzz1.f90,
которые содержат тексты виртуальных функций и другие операторы и подпрограммы пользователя, описанные в файле проекта.

Генерация локального Решателя производится с помощью скрипта, путь к которому задан в SET-переменной rFComp файла anes.acfg. По умолчанию используется скрипт script\ComSol_gfor.bat.

При создании «сложного» проекта, для работы которого необходимо подключение внешних подпрограмм или библиотек пользователя, необходимо использовать следующий алгоритм:

1) поместите A-файл в произвольный каталог, который будет далее использоваться как рабочий каталогWorkDIR;
2) скопируйте в этот каталог файлы etc\anes.acfg (это будет локальный файл конфигурации) и script\ComSol_gfor.bat (последний файл можно сохранить под любым именем);
3) исправьте в локальном файле строку с оператором rFComp; для этого укажите имя вашего скрипта без пути;
4) добавьте в файл скрипта ваши фортрановские файлы и (или) библиотеки.

Процесс обработки A-файла призводится с помощью скрипта script\aSolverRun.bat, который выполняется в отдельном окне-консоли.

Если в процессе компиляции или сборки Решателя были ошибки, то сообщения A-компилятора или компилятора Фортрана помещаются в файл report.log. Его можно просмотреть в оболочке Anes.

2.2.2 Проведение расчета

Если в процессе обработки A-файла ошибок не было, то скрипт aSolverRun.bat производит запуск Решателя для проведения расчета.

Решатель (по умолчанию или локальный) представляет собой обычное консольное приложение, в окно которого выводится «минимальная» информация о процессе сходимости:

1) номер итерации,
2) одна относительная ошибка, максимальная из всех решаемых Ф-переменных.

Шаг выдачи этих сообщений определяется AIL-оператором секции [Control]:

```
NoDialogSweep = <шаг выдачи в сек>
```

Для получения более подробной информации о процессе сходимости можно использовать либо программу ДиалогСолвера, либо проанализировать процесс сходимости в оболочке Anes после окончания расчета.

В общем случае Решатель создает следующие файлы:

<PreRes>.lst - листинг расчета,

<PreRes>XXX.ars - файлы результатов *для постпроцессора Anes* для шага по времени XXX; при решении стационарной задачи - файл <PreRes>0.ars; эти файлы создаются, если оператор секции [Main]
IsSaveAnesRes = .TRUE.

<PreRes>XXX.vtk - файлы результатов *для постпроцессора ParaView* для шага по времени XXX; при решении стационарной задачи - файл <PreRes>0.vtk; эти файлы создаются, если оператор секции [Main]
IsSaveVTKRes = .TRUE.

<PreRes>.acp - файл рестарта для возобновления расчета с контрольной точки,

<PreRes>.deb - файл с отладочной информацией для последовательного расчета; создается, если оператор секции [Main]

IsDbgMain = .TRUE.

<PreRes>_P_YYY.deb — файлы с отладочной информацией для параллельного расчета, YYY = 0 .. <число процессоров>-1,

INFOPAR.YYY — информация о декомпозиции расчетной области, файлы создаются в текущем каталоге, если оператор секции [Main] IsParInfo = .TRUE.

2.3 Скрипт выполнения расчета aSolverRun.bat

Работа с А-компилятором и Решателем из облочки Anes осуществляется с помощью скрипта aSolverRun.bat, расположенного в подкаталоге script. Формат вызова скрипта aSolverRun.bat:

```
aSolverRun.bat <путь к А-файлу > <Диалог> <число процессоров> [CompOnly]
```

где

<путь к А-файлу > — путь к А-файлу (относительный или абсолютный) без расширения ".a",

<Диалог> — Y – используется программа ДиалогСолвера,
N – не используется,

<число процессоров> — 0,1 – обработка А-файла и выполнение *последовательного* расчета,
N - обработка А-файла и выполнение *параллельного* расчета с использованием N-процессоров,

CompOnly — если указан этот ключ, то проводится только компиляция файла проекта А-компилятором.

Скрипт aSolverRun.bat в конце работы на консоль выдает запрос на закрытие окна консоли. Если необходимо отключить этот запрос (а это нужно при неоднократном вызове этого скрипта в пакетных файлах), то нужно использовать аналог этого скрипта - aSolverRunZZ.bat.

Пример использования скрипта для проведения расчета без оболочки – скрипт aBatRun.bat в каталоге work_win:

```
@echo off
rem ============================
rem Запуск примера из файла ases
rem  Запускается пример, помеченный в ases звездочкой
rem ============================
set bin_dir=%ANES20XE%\bin
set script_dir=%ANES20XE%\script
set ases_name=%ANES20XE%\project_lib\example-win.ases
rem ==== Получаем имя а-файла ====
set atestzz=%bin_dir%\agetafile.exe %ases_name%
for /f "usebackq tokens=1,2 delims==" %%i in (`%atestzz%`) do (
    set %%i=%%j
)
if %aerror%I == 8I (
  echo ***aBatRun error: not correct ases-file!
   pause
   exit 4
)
rem ============================
call %script_dir%\aSolverRun.bat %a_file% N 1
pause
```

3. Программа оболочка Anes

В составе кода имееются две оболочки:

1. bin\laShell.exe - оболочка, написанная на Lazarus/Pascal. Эта программа работает только в операционной системе Windows.
2. bin\luAShell.bat - оболочка, написанная на языке lua с использованием пакета wxLua. Эта программа работает как в ОС Windows, так и в ОС Linux.

Возможности и интерфейс обоих приложений близки. Ниже описывается работа с laShell.exe.

Оболочка Anes является программой-интегратором и предназначена для работы с проектами Anes. Она позволяет:

1) загрузить для работы отдельный проект или сессию Anes;
2) использовать несколько приложений для редактирования файла проекта;
3) редактировать файлы настройки Anes;
4) запускать на выполнение обработку файла проекта и проведение расчетов, как в последовательном, так и в параллельном режиме;
5) просмотреть результаты расчетов.

Основное окно программы показано на рисунке 3.1. Окно имеет два основных элемента-окна:

- в левом окне отображается список (или дерево) папок файла сессии,
- в правом список проектов текущей папки.

Работа с оболочкой осуществляется через пункты меню, которые продублированы кнопками на инструментальной панели. Ниже подробно описываются эти пункты меню.

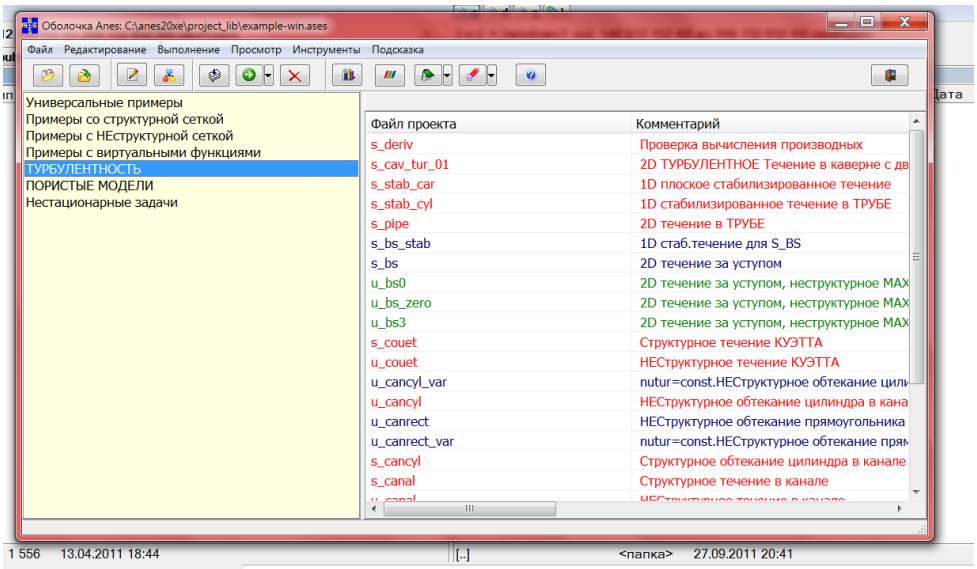


Рисунок 3.1 – Главное окно программы-оболочки

Меню\Подсказка\Описание операторов файла проекта (кнопка Help)

Загрузка описания файла подсказок по всем секциям и операторам языка описания проекта.

Меню\Открыть файл сессии

Выбор и загрузка файла сессии *.ases.

Меню\Файл\Открыть файл проекта

Выбор и загрузка отдельного файла проекта *.a. Для однотипности работы с файлом проекта и файлом сессии, файл проекта загружается в виде псевдо-файла сессии:

- с папкой «Текущий проект», содержащей данный файл проекта,
- рабочий каталог совпадает с каталогом A-файла.

Меню\Настройки

Запуск диалога настройки параметров оболочки (см. рисунок 3.2).

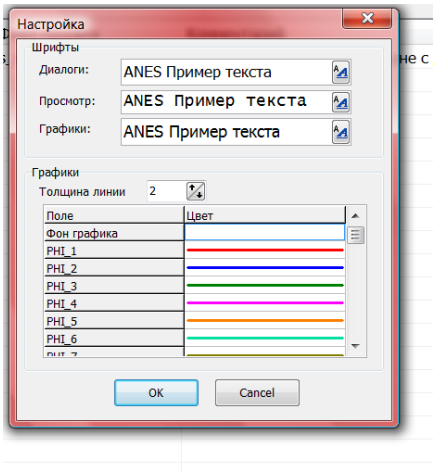


Рисунок 3.2 – Диалог настройки параметров оболочки

Диалог позволяет настроить:

1) шрифты, используемые для диалогов оболочки, окна просмотра файлов результатов и графиков сходимости,
2) Параметры графиков сходимости.

Меню\Редактирование\Текущий A-файл\Текстовый редактор

Редактирование текущего A-файла с помощью текстового редактора SciTE или другого редактора пользователя. Путь к этому редактору задается оператором sEdit глобального файла anes.acfg.

Меню\Редактирование\Текущий A-файл\Дизайнер Anes

Редактирование текущего A-файла с помощью программы-препроцессора Anes laPre20.exe.

Меню\Редактирование\Локальный файл anes.acfg
Меню\Редактирование\Глобальный файл anes.acfg

Редактирование локального anes.acfg файла, расположенного в рабочем каталоге WorkDir с помощью программы-препроцессора Anes (если он там есть) или редактирование глобального файла из каталога <anes>\etc\anes.acfg.

Это редактирование оказывает влияние на операторы файла настройки, используемые в процессе обработки A-файла и выполнения расчета:

rBDir, rCDir, rAComp, rSolver, rMpich, rFLib

Операторы файла настройки

sEdit, sParaView, sPreAnes, sPostAnes

всегда загружаются из глобального файла anes.acfg. Если были изменены эти операторы, то необходимо перезапустить оболочку.

Меню\Редактирование\Текущий файл сессии

Если загружен файл сессии, то его можно отредактировать. Например, сменить «цвет» файла проекта или добавить новый файл проекта. Перед редактированием сессия закрывается. Для продолжения работы с ней нужно заново ее загрузить.

Меню\Выполнение\Только компиляция А-файла

Этот пункт меню (и соответствующая кнопка) используется для проверки правильности А-файла. Эта команда запускает скрипт

```
aSolverRin.bat <имя текущего А-файла> N XX CompOnly
```

который производит компиляцию А-файла для последовательного (XX = 1) или параллельного расчета (XX = число процессоров) и, если необходимо, генерацию локального Решателя. Если при работе скрипта были ошибкы (на этапе компиляции или генерации), то на консоли появляются сообщения:

```
***aCompiler error: see report.log file!
***Fortran Compiler error: see report.log file!
```

Для анализа ошибок необходимо использовать команду «Просмотр\Файл report.log».

При нажатии кнопки возможны два режима компиляции: последовательный и параллельный. Во втором случае А-компилятор создает файлы, используемые для параллельного расчета. Эти файлы позволяют оценить эффективность декомпозиции без проведения расчета.

Меню\Выполнение\Последовательный расчет
Меню\Выполнение\Параллельный расчет

Этот пункт меню (и соответствующая кнопка) используется для обработки А-файла и выполнения расчета. Эта команда запускает скрипт

```
aSolverRin.bat <имя текущего А-файла> Y/N <число процессоров>,
```

который производит обработку А-файла и, если не было ошибок, выполнение расчета.

Перед выполнением запускается диалог, показанный на рисунке 3.3. Пользователь может выбрать число процессоров (для параллельного расчета) и включить флаг использования программы ДиалогСолвера. В последнем случае оболочка перед выполнением автоматически запускает программу ДиалогСолвера.

Эта программа позволяет в режиме реального времени просмотреть процесс сходимости итерационного процесса. В трех окнах программы показываются графики трех параметров:

1) точность решения дискретных уравнений для всех Ф-переменных,
2) значения Ф-переменных в точки мониторинга,
3) значения интегральных балансов.

Подробнее об этих параметров смотрите ниже (секция [Dialog]).

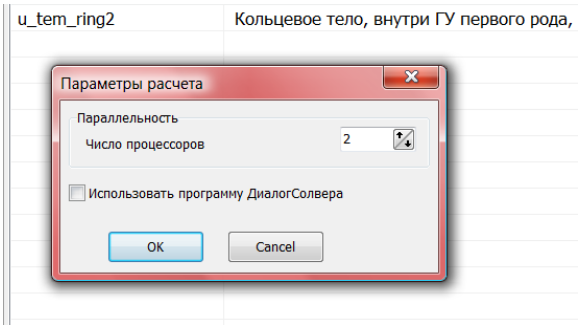


Рисунок 3.3 – Диалог выполнения расчета

☞Замечание. Процесс сходимости можно проанализировать и без программы Диалог-Солвера (правда, после проведения расчета). Для этого используется меню «Просмотр\Процесс сходимости».

Меню\Просмотр…

Эти пункты меню позволяют просмотреть все файлы для текущего проекта:

- Листинг компиляции - *.alst,
- LOG файл компилятора - *.log,
- Листинг расчета - *.lst,
- DEB-файл расчета - *.deb,
- Процесс сходимости - аналог программы ДиалогСолвера,
- Файл report.log - просмотр файла с сообщениями об ошибках компиляции или генерации локального Решателя.

Если в качестве редактора используется SciTE, то для поиска ошибок можно использовать следующий прием:

1) Загрузите в редактор файл report.log.
2) Выполните пункт меню «Tools\log\Загрузить REPORT.LOG».
3) В окне вывода редактора SciTE появится текст report.log, в котором ошибки будут подсвечены другим цветом.
4) Щелкните по строке с ошибкой – будет загружен A-файл и подствечена строка с ошибкой.
5) Если ошибка возникла на этапе генерации локального Решателя, то ссылка на ошибку будет расположена в Фортрановских файлах, сгенерированных Компилятором A-файла. Следует помнить, что эту ошибку нужно исправлять не в этом файле, а в Фортрановских секциях A-файла!

Меню\Инструменты\Постпроцессор ParaView

Этот пункт меню запускает программу-постпроцессор ParaView. Путь к этой программе задается в операторе sParaView файла конфигурации.

Меню\Инструменты\Постпроцессор Anes

Этот пункт меню запускает программу-постпроцессор Anes и загружает результаты текущего файла проекта. Путь к этой программе задается в операторе sPostAnes файла конфигурации.

Меню\Инструменты\Копия постпроцессора Anes

Этот пункт меню запускает копию постпроцессор Anes без загрузки файлов результатов. Это позволяет запустить несколько копий постпроцессора для визуального сравнения различных результатов.

Меню\Инструменты\Утилита верификации

Этот пункт меню запускает утилиту верификации Anes. Путь к этой программе задается в операторе sVerify файла конфигурации.

Меню\Инструменты\Утилита aShaper2D

Этот пункт меню запускает утилиту построения 2GR объектов. Путь к этой программе задается в операторе sShaper2D файла конфигурации.

Меню\Инструменты\Утилита просмотра DAT-файлов

Этот пункт меню запускает утилиту laViewDat.exe для просмотра графиков из текстового файла-таблицы. Файл таблица имеет следующий формат:

```
X-имя  F1-имя  F2-имя .....
x1     f1     f2  ....
```

.....

Таблица содержит значения нескольких функций F1(x), F2(x), ... Первая строка содержит названия осей графиков. Далее идут значения координаты x и функций, разделенные пробелами, символами табуляции или запятыми.

3.1 Программа ДиалогСолвера laDialog20

Программа ДиалогСолвера предназначена для просмотра сходимости итерационнго процесса во время решения. Диалог с программой Решателя осуществляется с использованием механизма сокетов операционной системы Windows. Окна программы изображены на рисунке 3.4.

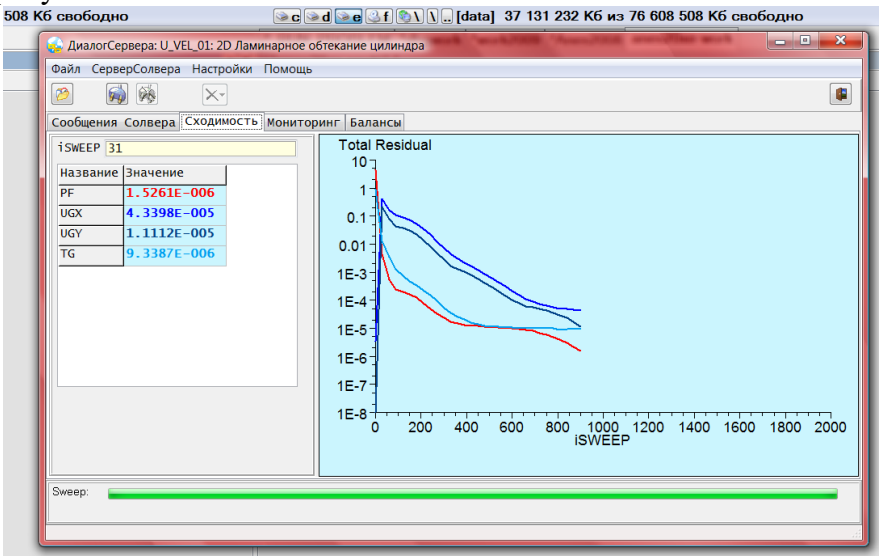


Рисунок 3.4 – Программа ДиалогСолвера

При работе в оболочке Anes запуск программы ДиалогСолвера производится автоматически. При работе вне Оболочки для запуска программы нужно использовать модуль

<anes>\bin\aDialog20.exe

Для активизации соединения с Решателем после старта необходимо нажать кнопку «Запуск сервера». Можно сразу активировать соединение

<anes>\bin\aDialog20.exe /start

3.2 Утилита верификации расчетов laVerify

В состав пакета входит специальная диалоговая программа-утилита laVerify.exe.Эта программа используется для проведения сравнения результатов расчетов с данными других расчетов или экспериментальных данных.

Утилита позволяет сравнить результаты, полученные в расчете с «эталонными» данными. Допускается два вида данных для сравнения:

1) одномерные графики,
2) числовые значения, описанные в виде <имя> = <значение>.

Примеры таких сравнений показаны на рисунках 3.5 и 3.6.

«Эталонные данные» и параметры сравнения хранятся в файле утилиты с произвольным именем и расширением “.asc” – скрипте утилиты верификации.

Файл скрипта представляет собой *набор блоков – кадров*. Каждый кадр служит для построения *окна сравнения* одного из двух типов:

1) Graph - кадр предназначен для построения одного или более графиков от одной общей x- переменной,
2) Comp - кадр предназначен для сравнения табличных переменных.

Файл скрипта содержит набор операторов построения и настройки. Правила записи операторов аналогичны синтаксису операторов AIL:

1) каждый оператор записывается на отдельной строке.
2) все что расположено правее знака “!” считается комментарием.
3) при записи операторов регистр букв не учитывается.

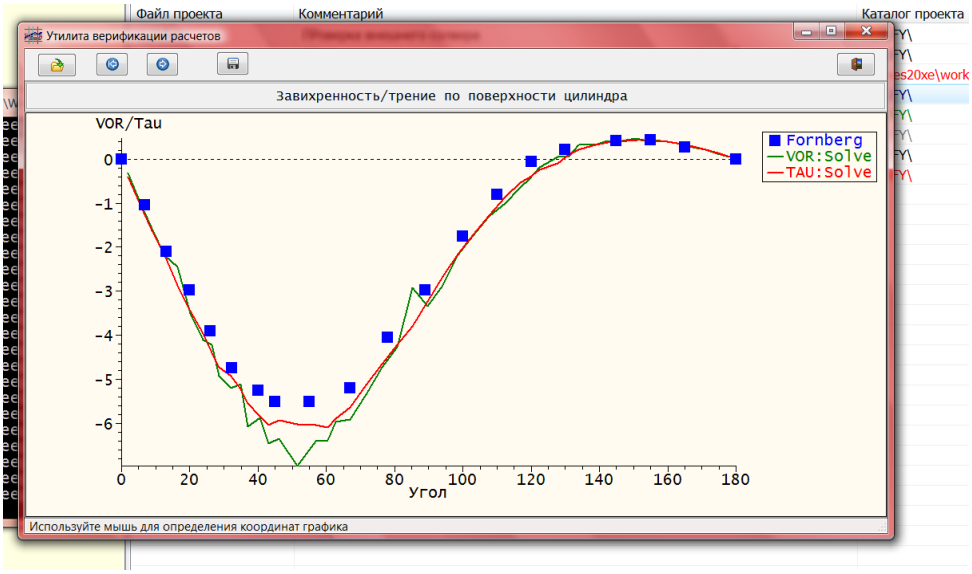
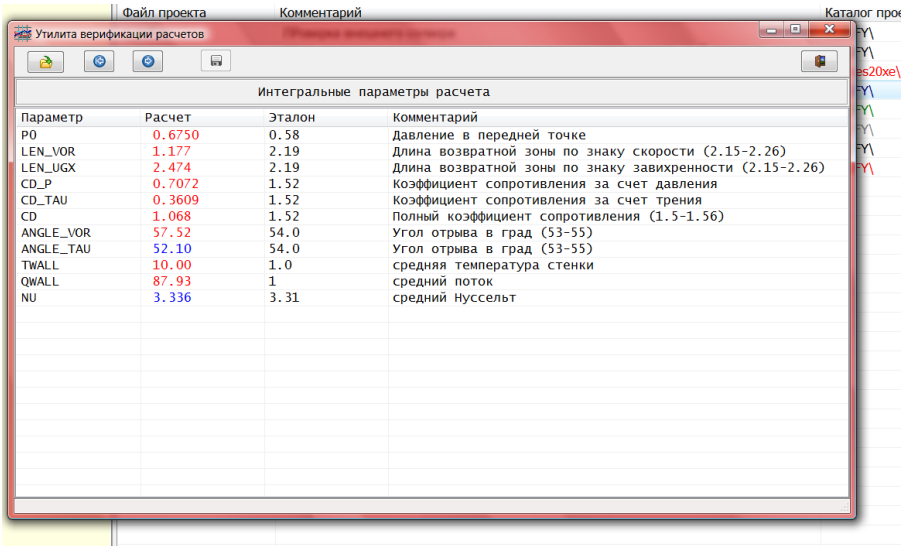


Рисунок 3.5 – Отображение кадра типа Graph



Параметр	Расчет	Эталон	Комментарий
P0	0.6750	0.58	Давление в передней точке
LEN_VOR	1.177	2.19	Длина возвратной зоны по знаку скорости (2.15-2.26)
LEN_UGX	2.474	2.19	Длина возвратной зоны по знаку завихренности (2.15-2.26)
CD_P	0.7072	1.52	Коэффициент сопротивления за счет давления
CD_TAU	0.3609	1.52	Коэффициент сопротивления за счет трения
CD	1.068	1.52	Полный коэффициент сопротивления (1.5-1.56)
ANGLE_VOR	57.52	54.0	Угол отрыва в град (53-55)
ANGLE_TAU	52.10	54.0	Угол отрыва в град (53-55)
TWALL	10.00	1.0	средняя температура стенки
QWALL	87.93	1	средний поток
NU	3.336	3.31	средний Нуссельт

Рисунок 3.6 – Отображение кадра типа Comp

Началом кадра является оператор

CADR = Graph/Comp

концом – либо новый оператор CADR, либо конец файла.

Все кадры обрабатываются последовательно, после выполнения операторов кадра, утилита переходит в состояние ожидания, пока не будет нажата кнопка «Следующий кадр».

3.2.1 Операторы настройки

Для вывода текстовой информации в заголовок кадра используется оператор

```
Title = "произвольный текст"
```

Для настройки цветов объектов графиков используется оператор

```
SetColor(NameObj, WNameColor)
```

здесь

NameObj = имя объекта, для которого устанавливается цвет:

Graph_Back - фон графика,

Graph_Axis - цвет осей графика и его оцифровки.

WNameColor – символьное название Windows-цвета (в операторе кавычки не нужны):

```
'Black', 'Maroon', 'Green',  'Olive', Navy', 'Purple', 'Teal', 'Gray', 'Silver',
'Red', 'Lime', 'Yellow', 'Blue', 'Fuchsia' , 'Aqua', 'White',
'ScrollBar', 'Background', 'ActiveCaption', 'InactiveCaption', 'Menu',
'Window', 'WindowFrame', 'MenuText', 'WindowText' 'CaptionText',
'ActiveBorder', 'InactiveBorder', 'AppWorkSpace', 'Highlight',
'HighlightText', 'BtnFace', 'BtnShadow'.
```

Для выбора фонтов графиков и окна сравнения используется оператор

```
SetFont(Obj, Name, Size, WNameColor)
```

здесь

Obj - имя объекта, для которого устанавливается фонт: GRAPH, MSG,

Name - название фонта, например, “ LUCIDA CONSOLE",

Size - размер фонта, например, для “ LUCIDA CONSOLE" = 10,

WNameColor - Windows-цвет фонта.

Настройки функций SetColor и SetFont распространяются на последующие кадры, пока не будут сменены новыми вызовами функций.

3.2.2 Кадр GRAPH: Построение графиков

Каждый кадр типа Graph используется для построения группы графиков в рамках одних осей абсцисс и ординат. Количество графиков для одного кадра не должно превышать 16.

Для построения графика может использоваться линия заданного цвета и (или) маркер. В качестве маркеров используются специальные символы, сам маркер задается индексом IndMarker = 1 – 8.

Название осей абсцисс и ординат задаются оператором

```
GTitle(Xtitle,YTitle)
```

где Xtitle,Ytitle - названия осей в двойных кавычках.

Для формирования графиков можно использовать три способа:

1) график из непосредственно заданной таблицы («эталонные» данные),
2) график из столбцов таблицы текстового файла («результаты расчета»),
3) график из файла с описанием двумерной поверхности («результаты расчета»).

Во всех случаях добавление графика к кадру начинается с оператора

```
AddGraph(Title,Color, IsLine, IndMarker)
```

где

Title - название графика в двойных кавычках (высвечивается в окне легенды),

Color - Windows-цвет линии или маркера,

IsLine - .True./.False. – вывод линии или маркера,

IndMarker - (1 – 8) – номер маркера.

График из непосредственно заданной таблицы

В этом случае график задается своими координатами с помощью операторов

```
GraphTable(Begin)
   x,y
  ....
GraphTable(End)
```

График из столбцов таблицы текстового файла

В этом случае используется оператор

```
GraphFile(FileName,IsWorkDir,[Blank,ICOL_X,ICOL_Y])
```

где

FileName - путь к файлу,

IsWorkDir - если .TRUE., то путь задается относительно текущего каталога,

Blank - символ-разделитель колонок в файле (один символ в двойных кавычках),

ICOL_X, ICOL_Y - колонки для x,y в файле (по умолчанию icol_x =1, icol_y=2).

Сам файл должен иметь следующую структуру:

```
<первая строка > - любой текст
X1 <BLANK> X2 <BLANK>X3…..
```

……..

Если символ-разделитель – пробел, то несколько рядом расположенных пробелов рассматриваются как один символ-разделитель.

Пример кадра – графика показан на рисунке 1.5а.

3.2.3 Кадр COMP: Таблица сравнение значений

Пример такого кадра показан на рисунке 1.6. Кадр представляет собой таблицу, состоящую из 4 столбцов:

1) NameVar - название переменной,
2) CompValue - значения из файла расчета,
3) ExactValue - эталонное значение,
4) Comm - произвольный текст – комментарий.

Для добавления переменных сравнения используется оператор

```
AddValue("NameVar",ExactValue,"Comm")
```

Для загрузки переменных расчета используется оператор

```
CompFile(FileName,IsWorkDir)
```

где

FileName - путь к файлу,

IsWorkDir - если .TRUE., то путь задается относительно текущего каталога.

Сам файл результатов должен иметь следующую структуру

```
NameVar =  CompValue
……
```

Количество переменных сравнения не ограничено. Расчетные значения выводятся в таблице двумя цветами:

красным – если расчетная и эталонная величина отличаются более чем на 5 %,

синим – если меньше, чем 5 %.

Пример файла *.asc для рисунков 1.5 – 1.6

```
!================
! Пример скрипта для aVERIFY
!================
CADR=GRAPH
!================
 SetColor("GRAPH_BACK",Cream)
```

```
 SetColor("GRAPH_AXIS",BLACK)
 SetFont("MSG","LUCIDA CONSOLE",10,BLACK)
   Title = "Завихренность/трение по поверхности цилиндра"
   GraphTitle("Угол","VOR/Tau")
! --- Данные Киртпатрика/Fornberg ----
AddGraph("Fornberg",BLUE,.false.,2)
GraphTable(Begin)
0.0,    0.0
6.8,   -1.04587
13.0,  -2.09174
20.0,  -2.97248
26.0,  -3.90826
32.4,  -4.73394
40.0,  -5.25688
45.0,  -5.50459
55.0,  -5.50459
67.0,  -5.20183
78.0,  -4.04587
89.0,  -2.97248
100.0, -1.76147
110.0, -7.98165E-01
120.0, -5.50459E-02
130.0,  2.20183E-01
145.0,  4.12844E-01
155.0,  4.40367E-01
165.0,  2.75229E-01
180.0,  0.0
GraphTable(End)
! --- расчет ---
AddGraph("VOR:Solve",GREEN,.true.)
GraphFile("data\vorf.dat",.true.," ",1,2)
AddGraph("TAU:Solve",RED,.true.)
GraphFile("data\vorf.dat",.true.," ",1,3)

!================
CADR=COMP
!================
Title = "Интегральные параметры расчета"
AddValue("P0", 0.58,"Давление в передней точке")
AddValue("LEN_VOR",2.19,"Длина возвратной зоны по знаку скорости (2.15-2.26)")
AddValue("LEN_UGX",2.19,"Длина возвратной зоны по знаку завихренности (2.15-2.26)")
AddValue("CD_P",1.52,"Коэффициент сопротивления за счет давления")
AddValue("CD_TAU",1.52,"Коэффициент сопротивления за счет трения")
AddValue("CD",1.52,"Полный коэффициент сопротивления (1.5-1.56)")
AddValue("ANGLE_VOR",54.0,"Угол отрыва в град (53-55)")
AddValue("ANGLE_TAU",54.0,"Угол отрыва в град (53-55)")
AddValue("TWALL",1.0,"средняя температура стенки")
AddValue("QWALL",1,"средний поток")
AddValue("NU",3.31,"средний Нуссельт")
!
CompFile("data\u2d.cdat",.true.)
```

Файл u2d.cdat

```
P0=  0.6750
LEN_VOR=   1.177
LEN_UGX=   2.474
CD_P=  0.7072
CD_TAU=  0.3609
CD=   1.068
ANGLE_VOR=   57.52
ANGLE_TAU=   52.10
TWALL=   10.00
QWALL=   87.93
NU=   3.336
```

Файл vorf.dat

```
"Angle"    "Vor" "Tau"
   1.791     -0.3234     -0.4047
   5.379     -0.9627      -1.019
```

```
.....
   174.6      0.1263      0.1322
   178.2      4.3495E-02  5.2111E-02
```

3.3 Утилита aShaper2D

Эта утилита предназначена для создания 2GR-файлов. Эти файлы используются для описания трехмерных объектов, которые получаются «перемещением» плоской фигуры в виде двумерной замкнутой полигональной линии (см. документ [1]).

В утилите для описания контура объекта используется специальные последовательности команд построения, которые сохраняются в файле 2D-объекта. Этот файл имеет расширение *.2gs и представляет обычный текстовый файл с командами построения полилинии.

Синтаксис таких команд аналогичен синтаксису языка AIL. При сохранении файла-скрипта автоматически создается и файл 2GR файл с тем же именем и с расширением ".2gr". Общий вид утилиты показан на рисунке 3.7.

Утилита имеет два основных окна. Левое окно – это окно редактора для редактирования 2GR-файла, правое предназначено для отображения полигона объекта (перерисовка объекта проводится при нажатии кнопки «Построить фигуру»).

Команды утилиты:

1. Меню «Файл\Новая фигура» - создание в редакторе заготовки нового 2GS файла.
2. Меню «Файл\Открыть» и соответствующая кнопка – загрузка существующего 2GS файла в окно редактора.
3. Меню «Файл\Сохранить» и соответствующая кнопка – сохранение текущих 2GS и 2GR файлов.
4. Меню «Файл\Сохранить как» – сохранение текущих 2GS и 2GR файлов под новым именем.
5. Кнопка «Построить фигуру» - перерисовка фигуры в правом окне.
6. Кнопка «Координаты фигуры-мышь» - при фиксации кнопки в нажатом состоянии можно «снимать» в графическом окне щелчком мыши. Команды с координатами мыши автоматически добавляются в окно редактора.
7. ЧекБокс «изотропность» - при включенном состоянии в графическом окне соблюдаются пропорции по осям X-Y.
8. Кнопка «Настройки» - запуск диалога настройки цветов графика и фонтов.

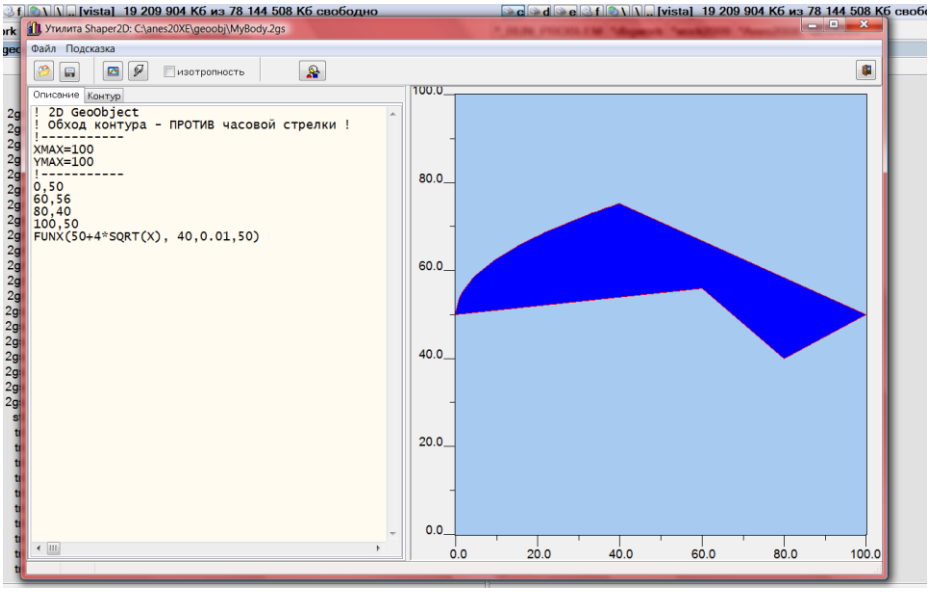


Рисунок 3.7 – Утилита построения 2GR объектов

Первыми операторами файла скрипта должны быть операторы
XMAX = <Размер по оси X>
YMAX = <Размер по оси Y>
задающие масштаб изменения координат 2D – объекта.
Далее идут последовательные команды построения участков объекта.
Допускаются следующие команды:

1) Два числа через запятую: - задается координата точки полилинии. Эти координаты можно вводить вручную, а можно с помощью курсора мыши. Для второго режима необходимо «вдавить» кнопку «Координаты фигуры-мышь».
2) Построение участка окружности: Для этого используется команда

   Circle(XC,YC,R,TetaBeg,TetaEnd,NoPOint)

   где

XC,YC - координаты центра окружности,
TetaBeg, TetaEnd - углы в градусах начала и конца дуги окружности.
Отсчет углов ведется от положительного направления оси x против часовой стрелки.
NoPoint - количество точек полилинии, на которые разбивается дуга.

3) *Линия F(X)*: В этом случае используется оператор
   FUNX(Formula, Xbeg,Xend,NoPoint)
   где

Xbeg, Xend - начало и конец линии по оси X,
NoPoint - число точек разбиения линии.
Formula – строка с формулой, содержащей описание линии. Сама формула записывается по правилам макропеременных AIL и может содержать:

1. числовые константы,
2. арифметические операции, включая возведение в степень, для которой используется символ "^",
3. любое число круглых парных скобок,
4. символ X или x, означающий x-координату функции,
5. арифметические функции:
   'ATAN', 'COS', 'SIN', 'TAN', 'ABS', 'EXP', 'LN', 'LOG', 'ASIN', 'ACOS',
   'SQRT' (квадратный корень) , 'SQR' (квадрат числа), 'SIGN' (знак числа),
   'TANH' (гиперболический тангенс), 'SINH' (гиперболический синус),
   'COSH' (гиперболический косинус).
6. Линия F(Y): В этом случае используется оператор

   FUNY(Formula, Ybeg,Yend,NoPOint)

Этот оператор аналогичен предыдущему, только вместо X используется координата Y.

☞Замечание. В формулах (в силу «неточности» разборщика формул) операция возведения в степень ‘^” и другие операции имеют один приоритет, поэтому лучше операции возведения в степень окружать скобками, например:

```
1 + 2.2 *( (x-1.5)^2)
```

Приведем два типичных примера скрипт-файла:
Построение поворотного канала:

```
!-----------
! 2D GeoObject
! Обход контура - ПРОТИВ часовой стрелки !
!-----------
```

```
XMAX=300
YMAX=100
!-----------
(300,0)
300,20
circle(40,40,20,270,180,16)
circle(40,60,20,180,90,16)
300,80
300,100
circle(40,60,40,90,180,16)
circle(40,40,40,180,270,16)
```

Построение профилированного сопла:

```
!-----------
! 2D GeoObject
! Обход контура - ПРОТИВ часовой стрелки !
!-----------
XMAX=10
YMAX=10
!-----------
funX(-5 * (TANH(2.0-X)- TANH(2.0)), 0,10,26)
10,10
0,10
```

3.4 Библиотека примеров файлов проекта

В состав пакета включена библиотека примеров, оформленная в виде файла – сессии, расположенная в каталоге

<anes>\project_lib\example-win.ases

Для работы нужно просто загрузить эту сессию в оболочку Anes и выбрать нужный проект для изучения.

3.5 Создание нового проекта

Для создания нового проекта используйте подходящий файл проекта из библиотеки примеров. Для его копирования можно использовать кнопку «Копирование файла проекта», расположенную на панели инструментов (третья кнопка слева).

После копирования не забудьте о подкаталоге data!

4. Расширенные средства работы с Anes

4.1 Консольные утилиты кода

Консольные утилиты представляю из себя консольные программы, служащие для «расширения» возможностей скриптов Windows. Утилиты расположены в каталоге <anes>\bin кода.

4.1.1 Утилита aConfig

Утилита предназначена для чтения операторов файла настроек *.acfg и выдачи их значений в стандартный вывод Stdout в виде переменных окружения ОС.

Для Windows (bat-скрипт) переменные выводятся в виде:

```
Name1=Value1
....
```

для Linux (sh – скрипт):

```
Value1
Value2
....
```

Команда запуска утилиты:

```
%ANES20XE%\bin\aconfig.exe  [-f:<файл *.acfg>]  Name1 Name2 ....
```

Если ключ –f: не указан, то в качестве acfg-файла используется файл по умолчанию etc\anes.acfg

Пример использования этой утилиты в Windows:

```
if exist anes.acfg (
set atestzz=%bin_dir%\aconfig.exe -f:anes.acfg rCdir rSolver rAcomp rFcomp rFlib rMpich rBPost
) else (
set atestzz=%bin_dir%\aconfig.exe rCdir rSolver rAcomp rFcomp rFlib rMpich rBPost
)
for /f "usebackq tokens=1,2 delims==" %%i in (`%atestzz%`) do (
   set %%i=%%j
)
if %aerror%l == 8l (
  echo ***aSolverRun error: not found anes.acfg file!
   pause
   exit 4
)
```

После выполнения этого участка будут сформированы SET-переменные с указанными именами. «Смешное» замечание: в операторе for используются обратные кавычки для выполнения строки (привет MS от Linux !!!) и это работает!

Пример использования этой утилиты в Linux:

```
if [ -f anes.acfg ]
then
  set -- $($bin_dir/aconfig.e -f:anes.acfg rCdir rSolver rAcomp rFcomp rFlib rMpich rHost rPost)
else
  set -- $($bin_dir/aconfig.e rCdir rSolver rAcomp rFcomp rFlib rMpich rHost rPost)
fi
rCdir=$1
rSolver=$2
rAcomp=$3
rFcomp=$4
rFlib=$5
rMpich=$6
```

```
rHost=$7
if [ "$rCdir" == "_aerror" ]
then
  echo ***aSolverRun error: not found anes.acfg file!
  exit 1
fi
```

4.1.2 Утилита aCopy

Утилита предназначена для работы с Решателем в пакетном режиме. Она заменяет в a-файле проекта подстроку вида %<name>% на <value>

Строка запуска:

```
%ANES20XE%\bin\acopy  A-in   A-out   [-c:"Name=value" ...]
```

Здесь A-in и A-out – имена входного и выходного файлов проектов, число операторов "-c:" может быть любое.

4.1.3 Утилита aTest

Утилита предназначена для тестирования A-файла с целью получения путей к файлам результатов и самому A-файлу. Кроме того утилита устанавливает флаг генерации локального Решателя. Утилита используется в скриптах генерации/выполнения проекта Anes.

Вызов утилиты:

```
%ANES20XE%\bin\atest.exe <путь к A-файлу>
```

Для windows утилита выдает в stdout:

```
AID=<путь к файлу .aid>
ISFOR=yes/no
AFILEDIR=<путь к A-файлу или .>
```

для Linux:

```
<путь к файлу .aid>
yes/no
<путь к a-файлу или .>
```

В качестве критерия генерации локального Решателя используются следующие условия:

1) в секциях есть операторы Vitrual(или External(;
2) есть хотя бы одна непустая секция [vf ...], [USER Fortran Variables], [USER Fortran Fields], [USER Fortran Subroutines];
3) если используется БД свойств (IsUseDB = .TRUE.) и в БД свойств есть операторы External(.

4.1.4 Утилита aGetAfile

Утилита предназначена для чтения файла сессии и выбора файла проекта, который в строке выделен символом "*". Если звездочки нет, то берется последний файл. Для файла проекта выдается полный путь = <Путь ases> + <путь из строки> . Имя файла выводится в stdout в виде SET переменной (без расширения .a):

Для Windows:

```
a_file=Value1
```

Для Linux :

```
Value1
```

Строка запуска утилиты:

```
%ANES20XE%\bin\agetafile.exe <файл ases>
```

4.2 MultiRun-режим выполнения расчетов

В Решателе кода Anes реализованы параллельные вычисления на основе MPI-интерфейса. Для распараллеливания используется модель декомпозиции расчетной области на субдомены. Каждый субдомен обрабатывается своей копией Решателя, а взаимодействие между копиями Решателя (в частности, синхронизация их работы) осуществляется

средствами MPI. Важно отметить, что в этом случае запускаются одинаковые копии одного исполняемого модуля Решателя.

В коде Anes реализована и другая подсистема – MultiRun-режим расчета, который позволяет параллельно запустить и синхронизировать Решатели Anes, решающие *разные* прикладные задачи. Более того, эта подсистема не использует для синхронизации MPI, поэтому отдельные Решатели в MultiRun-режиме могут использовать параллельные вычисления.

Рассмотрим модельную задачу о теплообменнике с тремя теплоносителями, которая позволяет продемонстрировать возможности MultiRun-режима. Рассмотрим плоский канал, в которым расположены два суб-канала (см. рисунок 4.1). В каждом канале течет свой теплоноситель, кроме того во втором суб-канале теплоноситель движется в другом направлении, чем в основном и в первом каналах.

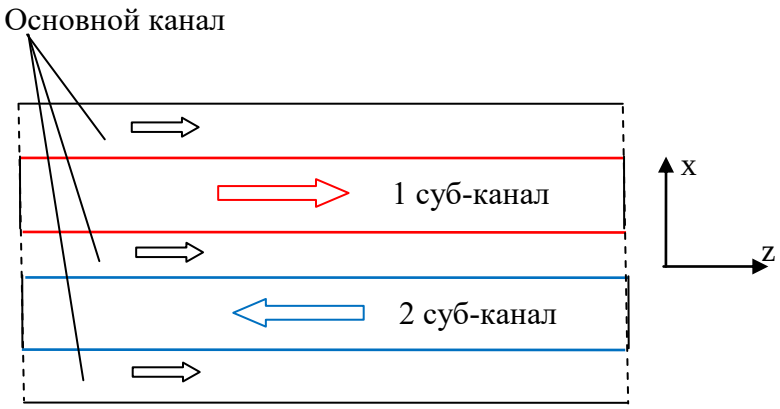


Рисунок 4.1 – Геометрия теплообменника

В принципе такую задачу можно решить с помощью одного файла проекта. Но можно эту задачу разбить на три задачи (см. рисунок 4.2):

1. Задача #0: Рассчитывается течение в основном канале. Субканалы моделируются с помощью BlockWall - патчей, а теплообмен с ними – через ГУ га этих патчах.
2. Задача #1: Течение в первом суб-канале. Теплообмен с основным каналом моделируется через ГУ на стенках канала.
3. Задача #3: Течение во втором субканале. Теплообмен с основным каналом моделируется через ГУ на стенках канала.

Для связи между этими независимыми задачами необходимо выполнить условия сопряжения тепловых потоков на границах каналов. Если использовать структурную сетку и ламинарный режим течения (именно этот режим рассматривается в тестовом примере project_lib/multirun) , то условия сопряжения легко свести к граничным условиям третьего рода (см. рисунок 4.3):

$$q_w = \lambda_N \frac{T_N - T_{wall}}{0.5\Delta x_N} = \lambda_P \frac{T_{wall} - T_P}{0.5\Delta x_P},$$

$$q_w(\text{Pipe1}) = \alpha_w \left(T_P - T_{wall}\right), \alpha_w = \frac{2\lambda_P}{\Delta x_P}, \qquad (4.1)$$

$$q_w(\text{Wall\_R}) = \alpha_w \left(T_N - T_{wall}\right), \alpha_w = \frac{2\lambda_N}{\Delta x_N}$$

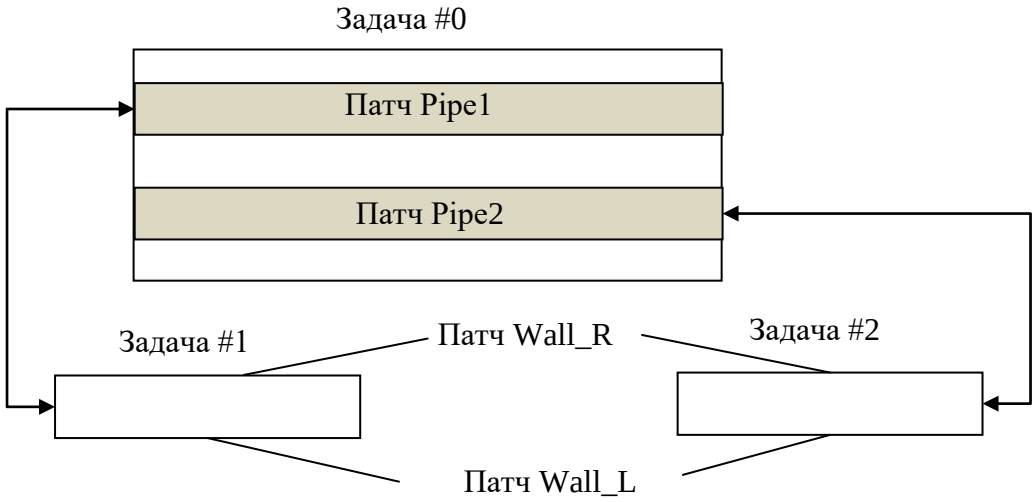


Рисунок 4.2 – Связь задач

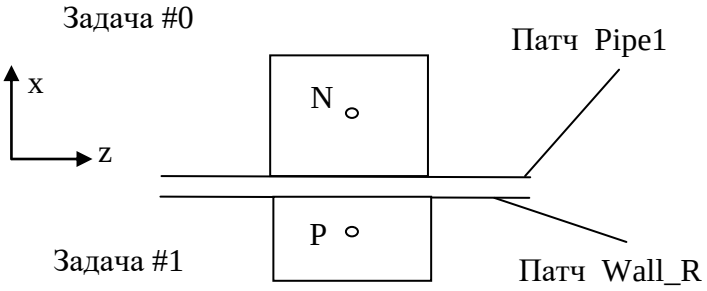


Рисунок 4.3 – Сопряжение задач #0 и #1

Как следует из (4.1) для совместного решения задач необходимо передать из текущей задачи в другие значения «коэффициентов теплоотдачи» и «внешней температуры». При совместном решении такой обмен необходимо делать на каждой итерации (например, в событии пользователя AfterSweep).

4.2.1 Архитектура MultiRun режима

При использовании MultiRun режима Решатели для каждой задачи запускаются не из оболочки Anes а из специальной программы-стартера:

<Anes>\bin\armStarter32.exe – для запуска 32-битного Решателя,
<Anes>\bin\armStarter64.exe – для запуска 64-битного Решателя.

Для синхронизации Решателей используются специальные подпрограммы Решателя, которые пользователь размещает в своих виртуальных функциях. Для доступа к этим подпрограммам необходимо включить оператор

```
use amrFUN
```

Для работы используются всего три подпрограммы: amrInit, amrBarrier и amrRemoteBarrier.

Для начала работы с MultiRun необходимо вызвать подпрограмму инициализации

```
subroutine  amrInit(MyID)
```

где MyID – уникальный индекс задачи 0 ≤ MyID ≤ NoProc, где NoProc – число задач. Проще всего вызов этой функции делать в событии пользователя onInit.

Для синхронизации задач используется подпрограмма
```
subroutine amrBarrier(subWRITE, subREAD)
external subWRITE, subREAD
```

Эта подпрограмма организует операцию барьера для независимых процессов: задачи должны одновременно подойти к точке ее вызова, после чего во всех задачах вызывается подпрограмма
```
call subREAD
```

и запускается режим ожидания выполнения этой подпрограммы во всех задачах. После выполнения подпрограммы во всех задачах аналогичная работа осуществляется с подпрограммой
```
call subWRITE
```

Такой алгоритм позволяет надежно осуществить передачу информации другим задачам (subWRITE) и получить информацию от других задач (subREAD). Проще всего это сделать с помощью файлов обмена (смотрите пример project_lib\multirun). Для хранения файлов рекомендуется использовать подкаталог exchange в рабочем каталоге. Этот подкаталог автоматически создается стартером.

Если текущая задача оканчивает расчет раньше, чем другие задачи, то при выходе из решателя необходимо вызвать третью подпрограмму
```
call amrRemoteBarrier
```
Эта подпрограмма позволит другим процессам продолжить вычисления. При этом оставшиеся в каталоге exchange файлы обмена данной задачи продолжат использоваться другими задачами.

Эти подпрограммы активируются только, если Решатели вызываются из программы-стартера. Если Решатели вызываются из оболочки, то эти подпрограммы являются «пустышками». В частности вызов подпрограмм пользователя subREAD и subWRITE не производится. Это позволяет производить независимую отладку задач из оболочки Anes. Главное, что нужно не забыть – провести инициализацию обменных массивов от других задач.

4.2.2 Программа стартер MultiRun режима

Программа стартер вызывается без агрументов
```
<Anes>\bin\amrStarter32(64).exe
```
при этом в текущем каталоге должен быть создан текстовый файл с именем amrStarter.in следующей структуры:

```
NoProc      : число запускаемых Решателей
Com0        : командная строка для запуска 0-ой задачи
….
Com(NoProc-1)  : командная строка для запуска последней задачи
```

В каталоге примеров расположен скрипт-файл для запуска решателя project_lib\multirun\RunStarter32.bat (или RunStarter64.bat):

```
echo off
rem ================================
rem Anes20XE: Скрипт для выполнения компиляции и расчетов с
rem  с помощью подсистемы MultiRun
rem ЗАМЕЧАНИЕ: Здесь приводится пример для использования трех
rem  процессов. Если процессов будет больше, посто добавьте их по аналогии.
rem ================================
set bin_dir=%ANES20XE%\bin
set script_dir=%ANES20XE%\script
rem --------------------------------
rem Эти переменные описывают имена A-файлов и префиксов файлов результатов
```

```
rem --------------------------------
set NoAFile=3
set afile1=m_proc_0
set infile1=mtest_0_

set afile2=m_proc_1
set infile2=mtest_1_

set afile3=m_proc_2
set infile3=mtest_2_

rem --------------------------------
rem Компиляция файлов проектов
rem --------------------------------
 call :CompAFILE %afile1%
  if errorlevel 1 goto Errors
 call :CompAFILE %afile2%
  if errorlevel 1 goto Errors
 call :CompAFILE %afile3%
  if errorlevel 1 goto Errors
rem --------------------------------
rem Создаем файл данных для amrStarter
rem --------------------------------
echo %NoAFile% > amrStarter.in
echo "exezz\%afile1%.exe data\%infile1%.aid" >> amrStarter.in
echo "exezz\%afile2%.exe data\%infile2%.aid" >> amrStarter.in
echo "exezz\%afile3%.exe data\%infile3%.aid" >> amrStarter.in
rem --------------------------------
rem Запускаем
rem --------------------------------
%bin_dir%\amrStarter32.exe
pause
exit 0

rem ================================
rem     ОШИБКА
rem ================================
:Errors
echo ">>>ERROR: see above."
pause
exit 8

rem ================================
rem ПП для компиляции проекта. Подпрограмма
rem  нужна для переименования исполняемого
rem  файла Решателя из exezz\atempzz.exe в имя
rem  exezz\afile.exe
rem ================================
:CompAFILE
echo ..... Compile A-file %1 ...
call %script_dir%\aSolverRunZZ.bat %1 N 0 CompOnly
if errorlevel 1 exit /B 8
if exist exezz\%1.exe erase exezz\%1.exe
copy exezz\atempzz.exe exezz\%1.exe
exit /B 0
```

Дадим некоторые комментарии к этому скрипту:

1. В этом файле производится как компиляция файлов проекта, так и запуск Решателей на выполнение. При решении сложных задач эти действия можно разбить на два скрипта. Для копиляции файла и генерации Решателя используется «стандартный» скрипт Anes – aSolverRunZZ.bat, который «дополняется» командой переименования исполняемого модуля.
2. Для создания файла amrStarter.in используются стандартные средства bat-файлов – команда echo.

Для запуска Решателя используется стандартная команда выполнения локального Решателя

<путь к Решателю>\Solver.exe <путь к aid-файлу>

```
exezz\%afile1%.exe data\%infile1%.aid
```

Если необходимо запустить параллельную версию Решателя, то для компиляции файла проекта нужно использовать команду

```
call %script_dir%\aSolverRunZZ.bat %1 N NoProc CompOnly
```

где NoProc – число процессоров. Для запуска Решателя нужно использовать следующую командную строку

```
<rMpich> -channel auto -localroot -n NoProc exezz\%afile1%.exe data\%infile1%.aid -p NoProc
```

Здесь <rMpich> - значение соответствующего ключа в файле anes.acfg.

4.2.3 Демонстрационный пример MultiRun-режима

В каталоге project_lib\multirun расположены файлы для описанного выше примера:

m_proc_0.a, m_proc_1.a, m_proc_2.a – файлы проекта задач,

RunStarter32.bat, RunStarter64.bat – скрипты на запуск.

Для отладки проектов в независимом режиме можно использовать оболочку Anes. Для реального расчета необходимо скопировать файлы в произвольный каталог, создать в этом каталоге подкаталог data и запустить нужный скрипт.

Рассмотрим некоторые особенности файлов проекта на примере задачи #0 (основной канал).

Для работы с граничными условиями для температуры на патчах Pipe1 и Pipe2 используются переменные пользователя

```
[USER Fortran Variables]
 ! --- обменные параметры для 1-го канала
 real, pointer:: TVal1_L(:), TAlfa1_L(:) ! низ первого канала
 real, pointer:: TVal1_R(:), TAlfa1_R(:) ! верх первого канала
! --- обменные параметры для 2-го канала
 real, pointer:: TVal2_L(:), TAlfa2_L(:) ! низ первого канала
 real, pointer:: TVal2_R(:), TAlfa2_R(:) ! верх первого канала

 integer IX_1L, IX_1R, IX_2L, IX_2R
```

Динамические массивы используются для хранения коэффициентов теплоотдачи и внешней температуры «чужих» процессов. Индексы IX_1L, … используются для определения ix-индексов верхних и нижних границ патчей.

Для инициализации массивов и MultiRun-режима используется событие пользователя onInit:

```
subroutine iINIT(RetVal)
 use amrFUN
 logical lRC
!==============================
!  Инициализация и открытие MultiRun
!==============================
!  1) Выделяем память
!
  allocate(TVal1_L(NZ), TAlfa1_L(NZ))
  allocate(TVal1_R(NZ), TAlfa1_R(NZ))
  allocate(TVal2_L(NZ), TAlfa2_L(NZ))
  allocate(TVal2_R(NZ), TAlfa2_R(NZ))
```

```
!------------------------------
!  2) Запоминаем координаты приграничных КО
!
  IP = GetIndexPatch('PIPE1')
  IX_1L = Patches(IP)%IXF
  IX_1R = Patches(IP)%IXL
  IP = GetIndexPatch('PIPE2')
  IX_2L = Patches(IP)%IXF
  IX_2R = Patches(IP)%IXL
!------------------------------
!  3) Начальная инициализация "чужих" данных.
!     Важно, что при работе в одиночной режиме эти
!     данные и будут использоваться.
!
  TVal1_L = RUdat(1); TAlfa1_L = 1.e5
  TVal1_R = RUdat(1); TAlfa1_R = 1.e5
  TVal2_L = RUdat(2); TAlfa2_L = 1.e5
  TVal2_R = RUdat(2); TAlfa2_R = 1.e5
!------------------------------
!  4) Активируем подсистему MultiRun.
!     MyID для данного процесса = 0
!
  IRC = amrInit(0)
end subroutine
```

Синхронизация задач осуществляется в событии пользователя AfterSweep:

```
subroutine iAfterSweep(RetVal)
 use amrFUN
 external iDWrite, iDRead
!==============================
!  Синхронизация. Вызываются пп
!  для записи своих данных и чтения чужих
!==============================
 call amrBarrier(iDWrite,iDRead)
end subroutine
```

Подпрограммы iDWrite и iDRead предназначены для записи своих коэффициентов теплоотдачи и температур и для чтения чужих:

```
!==============================
!  Подпрограмма записи наших данных
!==============================
subroutine iDWrite
 use UserData
!-----
  call userDiffCoef(P_TG)
! ---- Данные для процесса 1
  CALL userOpenOutFile(77,'exchange\solver_0_1.dat')
  do IzC = 2,NZ-1
    AlfaWallL = GamG(IX_1L,1,IzC) /(0.5*DX(IX_1L))
    AlfaWallR = GamG(IX_1R,1,IzC) /(0.5*DX(IX_1R))
    WRITE(77,'(4G13.4)')  F8(IX_1L,1,IZC,P_TG), AlfaWallL, &
                          F8(IX_1R,1,IZC,P_TG), AlfaWallR
  enddo
  CLOSE(77)
! ---- Данные для процесса 2
  CALL userOpenOutFile(77,'exchange\solver_0_2.dat')
  do IzC = 2,NZ-1
    AlfaWallL = GamG(IX_2L,1,IzC) /(0.5*DX(IX_2L))
    AlfaWallR = GamG(IX_2R,1,IzC) /(0.5*DX(IX_2R))
    WRITE(77,'(4G13.4)')  F8(IX_2L,1,IZC,P_TG), AlfaWallL, &
                          F8(IX_2R,1,IZC,P_TG), AlfaWallR
  enddo
  CLOSE(77)
end Subroutine

!==============================
```

```
! Подпрограмма чтения чужих данных
!============================
subroutine iDRead
 use UserData
!-----
! Данные от процесса 1
!
  OPEN(77, FILE = 'exchange\solver_1_0.dat' ,STATUS='old')
    DO IZC=2,NZ-1
      READ(77,'(4G13.4)')  TVal1_L(IZC), TAlfa1_L(IZC), TVal1_R(IZC), TAlfa1_R(IZC)
    ENDDO
   CLOSE(77)
!-----
! Данные от процесса 2
!
  OPEN(77, FILE = 'exchange\solver_2_0.dat' ,STATUS='old')
    DO IZC=2,NZ-1
      READ(77,'(4G13.4)')  TVal2_L(IZC), TAlfa2_L(IZC), TVal2_R(IZC), TAlfa2_R(IZC)
    ENDDO
   CLOSE(77)
end Subroutine
```

Прочитанные коэффициенты используются в виртуальных функциях задания COEF и VAL граничных условий, например, VAL для патча Pipe1:

```
!==============================
! VAL для первой трубы.
!==============================
 if(IxC.EQ.IX_1L) then
   RetVal = TVal1_L(IzC)
 else
   RetVal = TVal1_R(IzC)
 endif
end subroutine
```

И, наконец, в конце работы Решателя нужно освободить память для динамически выделенных массивов и закрыть MultiRun подсистему. Это можно сделать в событии пользователя onStop:

```
subroutine iStop(RetVal)
 use amrFUN
 logical lRC
!==============================
! Закрытие MultiRun
!==============================
! 1) Освобождаем память
!
  deallocate(TVal1_L, TAlfa1_L)
  deallocate(TVal1_R, TAlfa1_R)
  deallocate(TVal2_L, TAlfa2_L)
  deallocate(TVal2_R, TAlfa2_R)
!==============================
! 2) Сигнализируем об окончании
!
  call amrRemoteBarrier
end subroutine
```

4.3 Пакетный режим выполнения расчетов

Очень часто необходимо провести серию расчетов, меняя параметры файла проекта. В этом случаем можно воспользоваться утилитой aCopy.exe Anes. Поясним работу с этой утилитой на примере расчета стабилизированного течения в круглой трубе для различных моделей турбулентного числа Прандтля. Пример такого моделирования расположены в каталоге project_lib\batrun кода Anes.

Для работы с примером необходимо скопировать его в любой каталог и запустить скрипт RunBatAnes.bat.

В файле проекта s_reyhard.a расположены три «макропеременные» утилиты acopy: %RE% , %Pran%, %PRT% (заметим, что здесь регистр букв учитывается!).

При вызове утилиты acopy в bat-файле производится замена этих переменных и копирование файла проекта во временный файл проекта с именем test.a.

Файл-скрипта RunBatAnes.bat:

```
echo off
rem ================================================
rem Anes20XE: Подсистема верификации кода
rem  Это скрипт для пакетного выполнения
rem ================================================
set A_FILE=s_reyhard
set PRANLAM=1
rem ================================================
set OUT_FILE=%A_FILE%_10

rem ****************************************
rem    Если нет каталога bdata, создаем его.
rem ****************************************
if not exist exezz\nul mkdir bdata

rem ****************************************
rem    Модель Kays для Prt
rem ****************************************
if exist turbulence.dat erase turbulence.dat
rem =================
call :runSEQ %A_FILE%  "RE=8000" "PRT=-1"
call :runSEQ %A_FILE%  "RE=10000" "PRT=-1"
call :runSEQ %A_FILE%  "RE=20000" "PRT=-1"
call :runSEQ %A_FILE%  "RE=40000" "PRT=-1"
call :runSEQ %A_FILE%  "RE=60000" "PRT=-1"
call :runSEQ %A_FILE%  "RE=80000" "PRT=-1"
call :runSEQ %A_FILE%  "RE=100000" "PRT=-1"
rem =================
if exist %OUT_FILE%_kays.dat erase %OUT_FILE%_kays.dat
copy turbulence.dat %OUT_FILE%_kays.dat


rem ****************************************
rem    Постоянный Prt = 0.89
rem ****************************************
if exist turbulence.dat erase turbulence.dat
rem =================
call :runSEQ %A_FILE%  "RE=8000" "PRT=0.89"
call :runSEQ %A_FILE%  "RE=10000" "PRT=0.89"
call :runSEQ %A_FILE%  "RE=20000" "PRT=0.89"
call :runSEQ %A_FILE%  "RE=40000" "PRT=0.89"
call :runSEQ %A_FILE%  "RE=60000" "PRT=0.89"
call :runSEQ %A_FILE%  "RE=80000" "PRT=0.89"
call :runSEQ %A_FILE%  "RE=100000" "PRT=0.89"
rem =================
if exist %OUT_FILE%_89.dat erase %OUT_FILE%_89.dat
copy turbulence.dat %OUT_FILE%_89.dat

rem ****************************************
rem    Постоянный Prt = 1
rem ****************************************
```

```
if exist turbulence.dat erase turbulence.dat
rem =================
call :runSEQ %A_FILE%  "RE=8000" "PRT=1.0"
call :runSEQ %A_FILE%  "RE=10000" "PRT=1.0"
call :runSEQ %A_FILE%  "RE=20000" "PRT=1.0"
call :runSEQ %A_FILE%  "RE=40000" "PRT=1.0"
call :runSEQ %A_FILE%  "RE=60000" "PRT=1.0"
call :runSEQ %A_FILE%  "RE=80000" "PRT=1.0"
call :runSEQ %A_FILE%  "RE=100000" "PRT=1.0"
rem =================
if exist %OUT_FILE%_10.dat erase %OUT_FILE%_10.dat
copy turbulence.dat %OUT_FILE%_10.dat


rem ===== очистка ======
erase turbulence.dat
erase test.a
erase report.log
rem ===== очистка DATA ======
erase /Q bdata\*.*
erase /Q exezz\*.*
rmdir bdata
rmdir exezz
pause
exit 0

rem ********************************************************
rem Подпрограмма выполнения
rem ********************************************************
:runSEQ
rem без переустановки PATH он в aSolverRun Будет удлинятся
set PATH=c:\windows;c:\windows\system32
%ANES20XE%\bin\acopy.exe %1.a  test.a -c:%2 -c:%3 -c:"Pran=%PRANLAM%"
call %ANES20XE%\script\aSolverRunZZ.bat test N 1
exit /b
```

Дадим некоторые комментарии к этому скрипту:

1. Выполнение расчета производится подпрограммой скрипта с именем runSEQ. В этой подпрограмме сначала вызывается утилита acopy, которая создает временный файл проекта с именем test.a. На втором этапе вызывается стандартный скрипт Anes для выполнения расчета - aSolverRunZZ.bat (см. раздел 2.3).
2. В прилагаемом файле проекта производится расчет интегральных характеристик и вывод их в накопительный файл результатов. Этот пример может быть полезен для написания своих собственных проектов. Приведем главный участок этого файла проекта - наполнение накопительного файла:

```
[vf iReport]
subroutine iReport(RetVal)
 USE usaFVEDATA
 USE usaGRAD
 USE acTurbDATA
 Character(128) nwFile
!--------------------
! Обработка результатов - Смотрим сечение при IZ = 1
!--------------------
  if(IsPARALL) return
  LU = Lunit(U_LST)
!-----
  RhoZZ  = RhoG(2,1,1)
  aNuZZ  = NuG(2,1,1)
  D_HID  = 2.0*X(NX)
  PranZZ = RUDat(2)
  CondZZ = RUDat(4)
  ReZZ   = RUdat(3)
  U0     = REZZ*aNuZZ/D_HID
!-----
```

```
  CALL userDiffCoef(P_UGZ)
  aTAU = GamG(NX-1,1,1) * F8(NX-1,1,1,P_UGZ)/(0.5*DX(NX-1))
  uTau = SQRT(aTau/RhoZZ)
  aKSI_TAU  = 8.0 * aTau/RhoZZ/U0**2
  aKSI_DPDZ = DEV_DPDZ*D_HID/(0.5*RhoZZ*U0**2)
  YPLUSZZ   = uTAU * DX(NX-1)/aNuZZ
  aKSI_FIL = 1.0/((1.82*ALOG10(ReZZ)-1.64)**2)
  aKSI_BLA = 0.184/ReZZ**0.2
!-----
  AlfaZZ  =  DEV_Qwall/(DEV_Twall - DEV_Tbulk)
  aNU     =  AlfaZZ * D_HID/CondZZ
  aNU_PET = (aKSI_FIL/8.0) * REZZ*PranZZ/(1.07 + 12.7*SQRT(aKSI_FIL/8.0)*(PranZZ**0.666 - 1.0))
  aNU_DB  = 0.023*REZZ**0.8 * PranZZ**0.4
!---------------------------
!  Выводим в накопительный файл turbulence.dat
!
  nwFile = 'turbulence.DAT'
  if(.NOT.userFileExists(nwFile)) then
    OPEN(LU_OUT, FILE = nwFile,STATUS='UNKNOWN',POSITION ='APPEND')
    write(LU_OUT,'(a)') '"RE" "PRAN" "KSI_TAU" "KSI_DPDZ" "KSI_FIL" "KSI_BLA" "NU" "NU_PET" '// &
      '"NU_DB" "NU/NU_PET" "NU/NU_DB" "YPLUS"'
  else
    OPEN(LU_OUT, FILE = nwFile,STATUS='UNKNOWN',POSITION ='APPEND')
  endif
  write(LU_OUT,'(12g13.4)') REZZ, PranZZ, aKSI_TAU, aKSI_DPDZ, aKSI_FIL, aKSI_BLA, &
                    aNU, aNU_PET, aNU_DB, aNU/aNU_PET, aNU/aNU_DB , YPLUSZZ
  close(LU_OUT)

end subroutine
```

4.4 Многовариантный режим выполнения расчетов

Если необходимо проводить расчет многих вариантов, то при наличии многопроцессорного компьютера можно реализовать параллельные вычисления со 100% «ускорением». Нужно просто запустить на выполнение столько вариантов расчета, сколько имеется процессоров (или ядер).

Если в файлах проектов вариантов нет фортрановских текстов, то для выполнения используется Решатель по умолчанию. В этом случае нет проблем запустить несколько расчетов (из оболочки или из bat-файла). При наличии фортрановских текстов используется локальный Решатель, который имеет фиксированное имя atestzz.exe и расположен он в подкаталоге exezz рабочего каталога. Поэтому запустить несколько расчетов невозможно.

Для решения этой проблемы в состав скриптов включена модификация основного скрипта aSolverRun.bat - скрипт - aSolverRunMULTY.bat. Формат вызова этого скрипта:

```
%ANES20XE%\script\aSolverRunMULTY.bat   <Имя A-файла> <Число процессоров>
                                                    <Имя локального Решателя>
```

Здесь появился последний параметр, в котором и указывается имя локального Решателя. Пример bat-файла для вызова этого скрипта расположен в каталоге примеров project_lib\varrun:

```
@echo off
rem ============================
rem НЕЗАВИСИМЫЙ Запуск проекта
rem ============================
set script_dir=%ANES20XE%\script
rem ======Менять здесь ==================
set a_file=afiles\s_closure_new_2d-book
set exe_solver=book
rem ====================================
call %script_dir%\aSolverRunMULTY.bat %a_file% 1 %exe_solver%
pause
```

5. Работа в ОС Linux

В данной главе описываются особенности работы с кодом Anes в системе Linux. В текущей версии Anes основная работа выполняется в пакетном режиме без диалоговых компонент. Если пользователь имеет доступ к графическим средствам Linux, то для работы с кодом можно использовать графическую оболочку Anes, написанную на языке LUA с использованием пакета wxLua.

Скрипт оболочки - файл luashell.sh - расположен в каталоге bin пакета. Для удобства вызова оболочки в каталоге $(HOME)/bin расположен файл AnesShell.desktop, который представляет собой иконку для запуска оболочки. Вместе с кодом поставляется run-time версия интерпретатора lua и версия пакета wxLua для 64-разрядной среды GNOME.

Ниже рассматривается работа с пакетным режимом Anes. В качестве примера будет использоваться 48-ядерный сервер кафедры Инженерной теплофизики МЭИ.

5.1 Установка и настройка кода Anes

При работе с Anes в многопользовательском режиме код устанавливается у одного, «главного» пользователя, который в дальнейшем будет называться администратором Anes (anes-root). Домашний каталог anes-root может быть произвольным. Важно, чтобы anes-root принадлежал Linux-группе anes. Anes-root осуществляет генерацию кода (и перегенерацию при поступлении обновлений) и создание (обновление) bin-версии. Bin-версия Anes создается в каталоге

/usr/local/anes,

к которому должны иметь доступ пользователи группы anes (anes-root полный доступ, остальные пользователи для чтения и запуска).

Обычные пользователи работают с Anes из своих произвольных «домашних» каталогов. Единственное требование к ним – принадлежать к группе anes.

Для работы с Anes со стороны Linux необходимы следующие пакеты:

mpich2, gfortran или intel fortran ,

для работы со стороны клиента пользователь должен иметь коммуникационные утилиты Putty и FileZilla.

5.1.1 Установка пакета Anes пользователем-администратором

Дистрибутив Anes состоит из одного файла - anes2XX-cluster.tar.gz. Для установки необходимо скопировать этот файл в любой каталог, распаковать и запустить в терминале команду:

```
sh aInstall.sh
```

Этот скрипт создаст необходимые каталоги в HOME-каталоге anes-root и запустит генерацию библиотек, А-компилятора и Решателя кода. В текущей версии можно использовать два компилятора Фортрана: gFortran или Intel fortran for linux. Выбор компилятора производится при начале работы скрипта aInstall.

После работы скрипта необходимо перелогинится, чтобы нужные переменные окружения (которые скрипт записал в файл .bash_profile или .profile) были активизированы.

Для синхронизации каталога Anes anes-root и каталога anes-bin (и первоначального создания) используется скрипт

```
sh aAnes2usr.sh,
```

расположенный в каталоге HOME/bin.

5.1.2 Настройка пакета Anes для обычного пользователя

Для работы обычного пользователя необходимо установить основные файлы Anes в свой каталог. Для этого необходимо выполнить скрипт из любого каталога пользователя:

sh /usr/local/anes/iUser/aInstallUser.sh.

и перелогинится. Скрипт создаст следующие каталоги:
/home/<Пользователь>/
anes20xe
project_lib - копия каталогов примеров проектов,
work_lin - каталог для выполнения примеров,
data - подкаталог для результатов расчетов примеров.

В каталог work_lin инсталлятор скопирует следующие скрипты:
arun_linux.sh - выполнение расчета на удаленном компьютере;
arun_cluster.sh,
arun_cluster-asub.sh,
arun_cluster-slurm.sh
- выполнение расчета на удаленном компьютере через различные подсистемы пакетной обработки;
arun_ases.sh - выполнение тестовых примеров;
stopsolverok - скрипт для остановки расчета с сохранением результатов;
aget_nix.sh - скрипт для упаковки файлов результатов.

5.2 Работа на удаленном компьютере

Все взаимодействие пользователя с компьютером осуществляется через удаленную консоль Linux. Для работы необходимо установить в Windows две программы:
Putty - для работы с удаленной консолью,
FileZilla - для обмена файлами между компьютерами.

В этом режиме удаленный компьютер используется только для проведения расчетов. Для запуска расчета используется скрипт arun_linux.sh, расположенный в каталоге <Anes>/work_lin (ниже красным курсивом показаны комментарии):

```
#!/bin/bash
# ============================
#  ЭТО рабочий скрипт для выполнения расчета в системе Linux
#  в модели УДАЛЕННОГО доступа
#  с помощью вызова скрипта aSolverRun
# ============================
NOPROC=1              ! Число используемых процессоров
# AFILE=../project_lib/examples/e_tem_01
AFILE=../project_lib/examples/u_vel_cyl       ! Путь к файлу проекта
# ============================
bin_dir=$ANES20XE/bin
script_dir=$ANES20XE/script
# ============================
#  keys :  <A-file without ext> Dialog=Y/N [No.PROC]
# ============================
$script_dir/aSolverRun.sh $AFILE N $NOPROC
```

Для выполнения расчета нужно отредактировать файл arun_linux.sh (выбрать нужный файл проекта и число процессоров) и запустить этот файл на выполнение.

Для выполнения примеров из библиотеки примеров используется скрипт arun_ases.sh. Для выполнения примера нужно выбрать этот пример в файле сессии
project_lib/example-lin.ases
и запустить скрипт arun_ases.sh. Для выбора нужного примера нужно просто поставить символ «*» в первой позиции строки с нужным вариантом.

Некоторые полезные советы:

1) Для просмотра и редактирования файлов и запуска программ лучше всего в консоли использовать коммандер mc. Для запуска введите команду
mc -ca
Если в редакторе возникают проблемы с русскими буквами, войдите в настройку сеанса Putty и в меню Windows/Translation выберите UTF-8 кодировку.

2) При работе с Putty на стороне сервера создается консольный процесс пользователя, в котором и производится работа. У этого способа есть один большой недостаток – если связь порвется, то все данные будут потеряны! В Linux существует полезная утилита screen, которая позволяет создать из консоли Putty любое число новых консолей, которые не будут уничтожены после отключения от консоли Putty. Подробности работы с screen можно найти в справочной системе Linux. Приведем список основных команд screen:

screen - создание первой дополнительной консоли;
screen -ls - печать имен дополнительных консолей пользователя;
screen –r <имя консоли> - подключение к дополнительной консоли;
Ctrl + a+c - создание новой консоли screen;
Ctrl + a+d - отключиться от текущей консоли без ее закрытия;
Ctrl + a+n - переход к следующей консоли;
Ctrl + a+K - уничтожение текущей консоли, если консоль последняя, то будет закрыта сама screen.

3) При переносе файлов проектов из Windows в Linux (через FileZilla) необходимо перекодировать текстовый файл Windows (отличие – другая кодировка русских букв и разные символы перевода строк). Для этого можно использовать команду
toLINUX.sh <имя файла>
Для обратной перекодировки можно использовать команду
toWIN.sh <имя файла>

4) Для остановки расчета (с сохранением результатов) необходимо из mc запустить скрипт stopsolverok.sh.

5) Для переноса результатов расчетов на локальный компьютер можно использовать скрипт aget_nix.sh. Этот скрипт перекодирует листинг расчета и упаковывает результаты в zip-файл. Пользователь может задать какие файлы расчетов нужно упаковать. Для этого используются переменные:

```
LANESRES=Y    ! Упаковываются файлы для постпроцессора Anes
LVTKRES=Y     ! Упаковываются файлы для постпроцессора ParaView
LCP=Y         ! Упаковываются файлы для рестарта
```

5.3 Работа на кластере

Работа на кластере отличается от работы в удаленном режиме только способом расчета. На кластере расчет выполняется в пакетном режиме. Отличие в файле arun_cluster.sh - в способе описания числа процессоров:

```
#!/bin/bash
# ============================
#  ЭТО рабочий скрипт для выполнения расчета в системе CLUSTER
#  с помощью вызова скрипта aSolverRun
# ============================
NONODES=1    ! Число используемых узлов
NOCORE=2     ! Число используемых ядер узла
# ============================
# AFILE=$ANES20XE/project_lib/examples/e_tem_01
#AFILE=$ANES20XE/project_lib/examples/u_tubebank
AFILE=$ANES20XE/project_lib/examples/u_vel_cyl
# ============================
bin_dir=$ANES20XE/bin
script_dir=$ANES20XE/script
# ============================
# keys :  <A-file without ext> NoNODES NoPROC
```

```
# ============================
$script_dir/aSolverRun.sh $AFILE $NONODES $NOCORE
```

Для выполнения расчета нужно отредактировать файл arun_cluster.sh (выбрать нужный файл проекта и число процессоров) и запустить этот файл на выполнение.

Если скрипт выполнен без ошибок, то подсистема пакетной обработки вернет код задания в очереди, например

```
699.cluster
```

Для проверки состояния задания используется команда

```
qstat 699.cluster
```

Для остановки расчета (с сохранением результатов) нужно выполнить скрипт stopsolverok.sh. Для удаления задания из очереди используется команда

```
qdel 699.cluster
```

Литература

1. Код Anes20xe. «Работа с проектом пользователя», версия 2.24, 2019.
2. Код Anes20xe. «Визуализация результатов расчетов. Программы - постпроцессоры», версия 2.24, 2019.