

Инструкции: “Практическая часть”

Кирилл Терехов

<https://dodo.inm.ras.ru/terekhov/lect1/prak/presentation.pdf>

4 марта 2024 г.

1 Обзор

- Репозиторий кода
- Содержимое в ветках
- Примеры систем

2 Компиляция программ

- Необходимые инструменты
- Организация примеров

3 Разбор заданий

- Простое задание
 - Формулировка заданий
 - Сбор статистики
 - Симулятор
- Сложное задание
 - Формулировка заданий
 - Данные SPE10
 - Симулятор

Обзор

Пример (Ссылка)

`https://github.com/kirill-terekhov/mipt-solvers`

Репозиторий кода

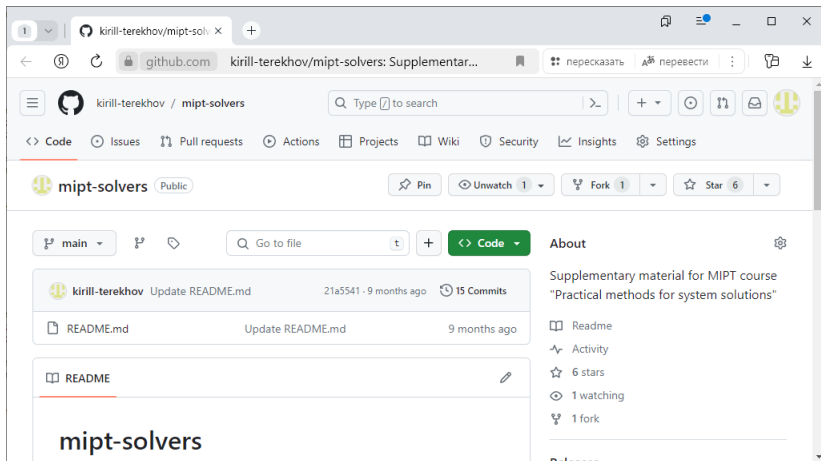


Рис. 1: Сайт репозитория

Содержимое в ветках репозитория

- lect0 Примеры систем линейных уравнений и генераторов систем.
- lect1 Метод сопряженных градиентов и предобуславливатель Чебышева.
- lect2 Метод неполной факторизации Краута с оценкой числа обусловленности обратных факторов.
- lect3 Пример геометрического многосеточного метода в одном измерении.
- lect4 Алгебраический многосеточный метод Рунге-Штюбена со сглаживателем Чебышева.
- lect5 Алгебраический многосеточный метод Рунге-Штюбена со сглаживателем Якоби и Гаусса-Зейделя.
- lect6 S-версия многоуровневого метода неполной факторизации.

Содержимое в ветках репозитория

- lect7 M-версия многоуровневого метода неполной факторизации.
- lect8 Перемасштабирование и переупорядочивание в методах неполной факторизации.
- lect9 Перемасштабирование и переупорядочивание в S и M версиях многоуровневой факторизации.
- lect10 Метод сопряженных градиентов Брамбла-Пасьяка для седловых систем.
- lect11 Многошаговые методы для задач фильтрации и пороупругости.
- lect12 Пример использования метода Ньютона и метода CPR-AMG для решения задачи двухфазного смешиваемого вытеснения.
- lect13 Метод дефляций для поиска многих решений в нелинейной задаче Карриера.

Примеры систем

Содержимое ветки `lect0`, готовые системы в формате `.mtx`:

`poisson_10` Задача Пуассона на регулярной сетке $10 \times 10 \times 10$.

`poisson_40` Задача Пуассона на регулярной сетке $40 \times 40 \times 40$.

`two_wells_tpfa` Задача анизотропной диффузии с двумя скважинами, дискретизация МКО с двухточечной аппроксимацией потока.

`two_wells_mfd` Задача анизотропной диффузии с двумя скважинами, дискретизация методом опорных операторов.

`two_wells_saddle_darcy` Задача анизотропной диффузии с двумя скважинами, дискретизация системы в седловой форме.

`norne_tpfa` Задача анизотропной диффузии с использованием данных месторождения Норна.

Иллюстрации примеров

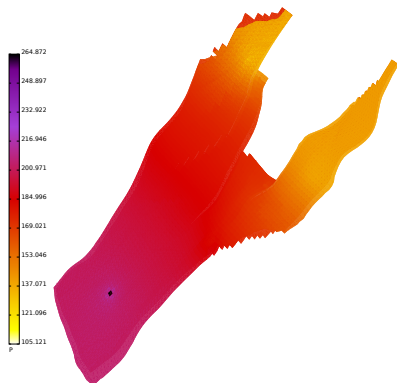
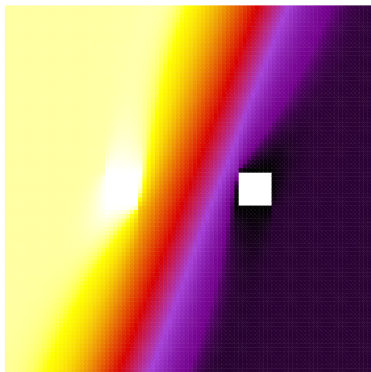


Рис. 2: Задача с двумя скважинами (слева). Задача на данных месторождения Норна (справа).

Примеры генераторов систем

Содержимое ветки [lect0](#), генераторы систем:

[poisson.cpp](#) Задача Пуассона на регулярной сетке $N \times N$.

[stokes.cpp](#) Задача Стокса на регулярной сетке $N \times M$.

[biot.cpp](#) Задача Био на регулярной сетке $N \times M$.

[deadoil.cpp](#) Задача двухфазной фильтрации с двумя скважинами на сетке $N \times M$.

[biharmonic.cpp](#) Дискретизация бигармонического уравнения на сетке $N \times N$.

Информацию о параметрах можно получить при запуске исполняемого файла без параметров.

$$-\Delta p = b \quad (\text{poisson.cpp})$$

$$\begin{cases} -\mu\Delta\vec{u} + \nabla p = \mathbf{b} \\ \operatorname{div}(\vec{u}) = 0 \end{cases} \quad (\text{stokes.cpp})$$

$$\begin{cases} -\mu\Delta\vec{u} - (\mu + \lambda)\nabla\operatorname{div}(\vec{u}) + \alpha\nabla p = \mathbf{b} \\ \partial_t(\zeta p + \alpha\operatorname{div}(\vec{u})) - \kappa\Delta p = q \end{cases} \quad (\text{biot.cpp})$$

$$\begin{cases} \phi\partial_t S - \operatorname{div}(S\kappa\nabla p) = q_o \\ \phi\partial_t(1 - S) - \operatorname{div}((1 - S)\kappa\nabla p) = q_w \end{cases} \quad (\text{deadoil.cpp})$$

$$-\Delta^2 p = b \quad (\text{biharmonic.cpp})$$

+ граничные и начальные условия.

Примеры для визуализации решения

Содержимое ветки [lect0](#), создание файла `.vtk`:

[scalar_grid.cpp](#) Визуализация решения примеров [poisson.cpp](#) и [biharmonic.cpp](#).

[stokes_grid.cpp](#) Визуализация решения примеров [stokes.cpp](#) и [biot.cpp](#).

[deadoil_grid.cpp](#) Визуализация решения примера [deadoil.cpp](#).

Информацию о параметрах можно получить при запуске исполняемого файла без параметров. Для визуализации файлов `.vtk` можно использовать www.paraview.org.

Визуализация решения

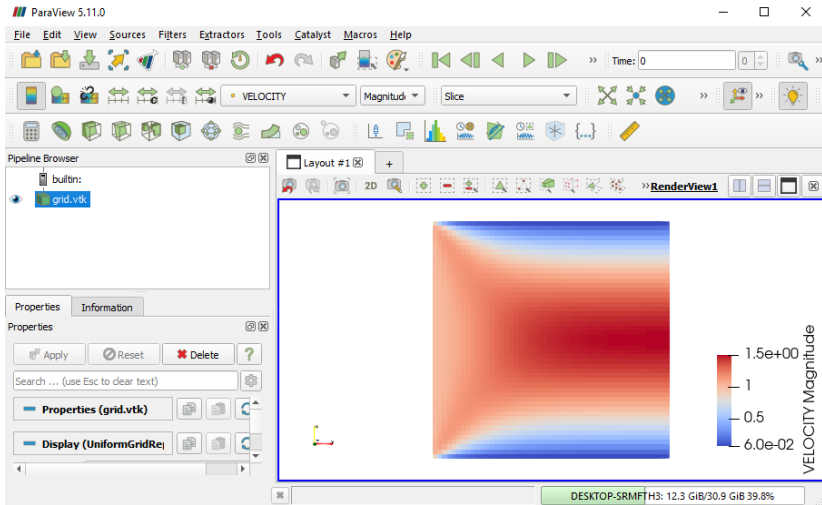


Рис. 3: Пример визуализации решения задачи `stokes.cpp` с $N = 50$ при помощи программы **paraview**.

Компиляция программ

- Программа **cmake**: www.cmake.org
- Компилятор C++:
 - Linux: g++, clang, intel compiler
 - Windows: microsoft visual studio, clang, windows subsystem for linux, cygwin, intel compiler

Пример компиляции

- `git clone https://github.com/kirill-terekhov/mipt-solvers.git`
- `mkdir mipt-solvers-build`
- `cd mipt-solvers`
- `git checkout lect0`
- `cd ../mipt-solvers-build`
- **linux** или linux-подобные компиляторы в windows:
 - `cmake ../mipt-solvers -DCMAKE_BUILD_TYPE=Release`
 - `cmake --build .`
 - Исполняемые файлы появятся в текущей папке.
- **windows** с компилятором из Visual Studio:
 - `cmake ../mipt-solvers`
 - `cmake --build . --config Release`
 - Исполняемые файлы появятся в папке Release.

Если создать папку с результатом компиляции внутри папки `mipt-solvers`, то при переключении на материал другой лекции могут возникнуть конфликты в `git`.

Пример компиляции

```
Windows PowerShell
(C) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

Попробуйте новую кроссплатформенную оболочку PowerShell (https://aka.ms/pscore6)

PS C:\Users\kirill\Documents\Code\prak> git clone https://github.com/kirill-terekhov/mipt-solvers.git
Cloning into 'mipt-solvers'...
remote: Enumerating objects: 496, done.
remote: Counting objects: 100% (10/10), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 496 (delta 2), reused 4 (delta 0), pack-reused 486
Receiving objects: 100% (496/496), 6.87 MiB | 6.97 MiB/s, done.
Resolving deltas: 100% (293/293), done.
PS C:\Users\kirill\Documents\Code\prak> mkdir mipt-solvers-build

Каталог: C:\Users\kirill\Documents\Code\prak

Mode                LastWriteTime         Length Name
----                -
d-----          19.02.2024    15:37         mipt-solvers-build

PS C:\Users\kirill\Documents\Code\prak> cd mipt-solvers
PS C:\Users\kirill\Documents\Code\prak\mipt-solvers> git checkout lect0
Switched to a new branch 'lect0'
branch 'lect0' set up to track 'origin/lect0'.
PS C:\Users\kirill\Documents\Code\prak\mipt-solvers> cd ../mipt-solvers-build
PS C:\Users\kirill\Documents\Code\prak\mipt-solvers-build> cmake ../mipt-solvers
-- Building for: Visual Studio 17 2022
CMake Warning (dev) at CMakeLists.txt:1 (project):
  cmake_minimum_required() should be called prior to this top-level project()
  call.  Please see the cmake-commands(7) manual for usage documentation of
  both commands.
This warning is for project developers.  Use -Wno-dev to suppress it.

-- Selecting Windows SDK version 10.0.22000.0 to target windows 10.0.19045.
-- The C compiler identification is MSVC 19.38.33135.0
-- The CXX compiler identification is MSVC 19.38.33135.0
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: C:/Program Files/Microsoft Visual Studio/2022/Community/VC/Tools/MSVC/14.38.33130/bin/Hostx64/x64/cl.exe - skipped
-- Detecting C compile features
-- Detecting C compile features - done
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: C:/Program Files/Microsoft Visual Studio/2022/Community/VC/Tools/MSVC/14.38.33130/bin/Hostx64/x64/cl.exe - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done (5.0s)
-- Generating done (0.1s)
-- Build files have been written to: C:\Users\kirill\Documents\Code\prak\mipt-solvers-build
```

Рис. 4: Пример компиляции на windows.

Пример компиляции

```
Windows PowerShell
PS C:\Users\kiril\Documents\Code\prak\mipt-solvers-build> cmake --build . --config Release
Версия MSBuild 17.8.5+b2562ef37 для .NET Framework

Checking Build System
Building Custom Rule C:/Users/kiril/Documents/Code/prak/mipt-solvers/CMakelists.txt
biharmonic.cpp
biharmonic.vcxproj -> C:\Users\kiril\Documents\Code\prak\mipt-solvers-build\Release\biharmonic.exe
Building Custom Rule C:/Users/kiril/Documents/Code/prak/mipt-solvers/CMakelists.txt
biot.cpp
biot.vcxproj -> C:\Users\kiril\Documents\Code\prak\mipt-solvers-build\Release\biot.exe
Building Custom Rule C:/Users/kiril/Documents/Code/prak/mipt-solvers/CMakelists.txt
deadoil.cpp
deadoil.vcxproj -> C:\Users\kiril\Documents\Code\prak\mipt-solvers-build\Release\deadoil.exe
Building Custom Rule C:/Users/kiril/Documents/Code/prak/mipt-solvers/CMakelists.txt
deadoil_grid.cpp
deadoil_grid.vcxproj -> C:\Users\kiril\Documents\Code\prak\mipt-solvers-build\Release\deadoil_grid.exe
Building Custom Rule C:/Users/kiril/Documents/Code/prak/mipt-solvers/CMakelists.txt
poisson.cpp
poisson.vcxproj -> C:\Users\kiril\Documents\Code\prak\mipt-solvers-build\Release\poisson.exe
Building Custom Rule C:/Users/kiril/Documents/Code/prak/mipt-solvers/CMakelists.txt
scalar_grid.cpp
scalar_grid.vcxproj -> C:\Users\kiril\Documents\Code\prak\mipt-solvers-build\Release\scalar_grid.exe
Building Custom Rule C:/Users/kiril/Documents/Code/prak/mipt-solvers/CMakelists.txt
stokes.cpp
stokes.vcxproj -> C:\Users\kiril\Documents\Code\prak\mipt-solvers-build\Release\stokes.exe
Building Custom Rule C:/Users/kiril/Documents/Code/prak/mipt-solvers/CMakelists.txt
stokes_grid.cpp
stokes_grid.vcxproj -> C:\Users\kiril\Documents\Code\prak\mipt-solvers-build\Release\stokes_grid.exe
Building Custom Rule C:/Users/kiril/Documents/Code/prak/mipt-solvers/CMakelists.txt
PS C:\Users\kiril\Documents\Code\prak\mipt-solvers-build> ls .\Release\

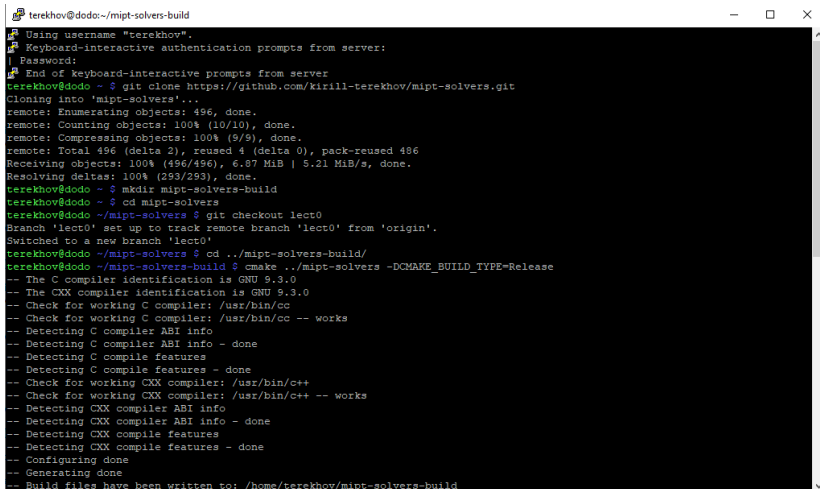
Каталог: C:\Users\kiril\Documents\Code\prak\mipt-solvers-build\Release

Mode                LastWriteTime         Length Name
----                -
-a----             19.02.2024   15:38          33280 biharmonic.exe
-a----             19.02.2024   15:38          38400 biot.exe
-a----             19.02.2024   15:38          33328 deadoil.exe
-a----             19.02.2024   15:38          34304 deadoil_grid.exe
-a----             19.02.2024   15:38          33280 poisson.exe
-a----             19.02.2024   15:38          32256 scalar_grid.exe
-a----             19.02.2024   15:38          35840 stokes.exe
-a----             19.02.2024   15:38          33280 stokes_grid.exe

PS C:\Users\kiril\Documents\Code\prak\mipt-solvers-build>
```

Рис. 5: Пример компиляции на windows.

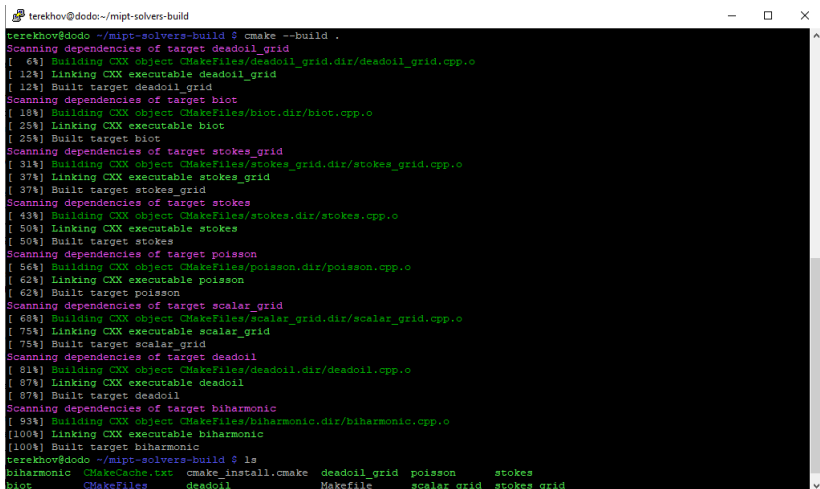
Пример компиляции



```
terekhov@dodo:~/mipt-solvers-build
Using username "terekhov".
Keyboard-interactive authentication prompts from server:
| Password:
| End of keyboard-interactive prompts from server
terekhov@dodo ~ $ git clone https://github.com/kirill-terekhov/mipt-solvers.git
Cloning into 'mipt-solvers'...
remote: Enumerating objects: 496, done.
remote: Counting objects: 100% (10/10), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 496 (delta 2), reused 4 (delta 0), pack-reused 486
Receiving objects: 100% (496/496), 6.07 MiB | 5.21 MiB/s, done.
Resolving deltas: 100% (293/293), done.
terekhov@dodo ~ $ mkdir mipt-solvers-build
terekhov@dodo ~ $ cd mipt-solvers
terekhov@dodo ~/mipt-solvers $ git checkout lect0
Branch 'lect0' set up to track remote branch 'lect0' from 'origin'.
Switched to a new branch 'lect0'
terekhov@dodo ~/mipt-solvers $ cd ../mipt-solvers-build/
terekhov@dodo ~/mipt-solvers-build $ cmake ../mipt-solvers -DCMAKE_BUILD_TYPE=Release
-- The C compiler identification is GNU 9.3.0
-- The CXX compiler identification is GNU 9.3.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /home/terekhov/mipt-solvers-build
```

Рис. 6: Пример компиляции на Linux.

Пример компиляции



```
terekhov@dodo:~/mipt-solvers-build
terekhov@dodo:~/mipt-solvers-build $ cmake --build .
Scanning dependencies of target deadoil_grid
[ 6%] Building CXX object CMakeFiles/deadoil_grid.dir/deadoil_grid.cpp.o
[ 12%] Linking CXX executable deadoil_grid
[ 12%] Built target deadoil_grid
Scanning dependencies of target biot
[ 18%] Building CXX object CMakeFiles/biot.dir/biot.cpp.o
[ 25%] Linking CXX executable biot
[ 25%] Built target biot
Scanning dependencies of target stokes_grid
[ 31%] Building CXX object CMakeFiles/stokes_grid.dir/stokes_grid.cpp.o
[ 37%] Linking CXX executable stokes_grid
[ 37%] Built target stokes_grid
Scanning dependencies of target stokes
[ 43%] Building CXX object CMakeFiles/stokes.dir/stokes.cpp.o
[ 50%] Linking CXX executable stokes
[ 50%] Built target stokes
Scanning dependencies of target poisson
[ 56%] Building CXX object CMakeFiles/poisson.dir/poisson.cpp.o
[ 62%] Linking CXX executable poisson
[ 62%] Built target poisson
Scanning dependencies of target scalar_grid
[ 68%] Building CXX object CMakeFiles/scalar_grid.dir/scalar_grid.cpp.o
[ 75%] Linking CXX executable scalar_grid
[ 75%] Built target scalar_grid
Scanning dependencies of target deadoil
[ 81%] Building CXX object CMakeFiles/deadoil.dir/deadoil.cpp.o
[ 87%] Linking CXX executable deadoil
[ 87%] Built target deadoil
Scanning dependencies of target biharmonic
[ 93%] Building CXX object CMakeFiles/biharmonic.dir/biharmonic.cpp.o
[100%] Linking CXX executable biharmonic
[100%] Built target biharmonic
terekhov@dodo:~/mipt-solvers-build $ ls
biharmonic  CMakeCache.txt  cmake_install.cmake  deadoil_grid  poisson      stokes
biot        CMakeFiles      deadoil             Makefile     scalar_grid  stokes_grid
```

Рис. 7: Пример компиляции на Linux.

Переключение материалов лекции

- `cd ../mipt-solvers`
- `git checkout lect1`
- `cd ../mipt-solvers-build`
- `cmake -build . -config Release`

Переключение материалов лекции

```
Windows PowerShell
PS C:\Users\kiril\Documents\Code\prak\mipt-solvers> cd ..\mipt-solvers\
PS C:\Users\kiril\Documents\Code\prak\mipt-solvers> git checkout lect1
Switched to branch 'lect1'
Your branch is up to date with 'origin/lect1'.
PS C:\Users\kiril\Documents\Code\prak\mipt-solvers> cd ..\mipt-solvers-build\
PS C:\Users\kiril\Documents\Code\prak\mipt-solvers-build> cmake --build . --config Release
CMake is re-running because C:\Users\kiril\Documents\Code\prak\mipt-solvers-build\CMakeFiles\generate.stamp is out-of-date.
The file 'C:\Users\kiril\Documents\Code\prak\mipt-solvers\CMakeLists.txt'
is newer than 'C:\Users\kiril\Documents\Code\prak\mipt-solvers-build\CMakeFiles\generate.stamp.depend'
result='-1'
CMake Deprecation Warning at CMakeLists.txt:1 (cmake_minimum_required):
Compatibility with CMake < 2.8.12 will be removed from a future version of
CMake.

Update the VERSION argument <min> value or use a ...<max> suffix to tell
CMake that the project does not need compatibility with older versions.

-- Selecting Windows SDK version 10.0.22000.0 to target Windows 10.0.19045.
-- Configuring done (0.0s)
-- Generating done (0.2s)
-- Build files have been written to: C:\Users\kiril\Documents\Code\prak\mipt-solvers-build
Версия MSBuild 17.8.5+b526ef37 для .NET Framework

cg.cpp
CG.vcxproj -> C:\Users\kiril\Documents\Code\prak\mipt-solvers-build\Release\CG.exe
cmpvec.cpp
CMPVEC.vcxproj -> C:\Users\kiril\Documents\Code\prak\mipt-solvers-build\Release\CMPVEC.exe
conv.cpp
Convert.vcxproj -> C:\Users\kiril\Documents\Code\prak\mipt-solvers-build\Release\Convert.exe
pcg.cpp
PCG.vcxproj -> C:\Users\kiril\Documents\Code\prak\mipt-solvers-build\Release\PCG.exe
symm.cpp
SYMM.vcxproj -> C:\Users\kiril\Documents\Code\prak\mipt-solvers-build\Release\SYMM.exe
PS C:\Users\kiril\Documents\Code\prak\mipt-solvers-build> ls .\Release\

Каталог: C:\Users\kiril\Documents\Code\prak\mipt-solvers-build\Release

Mode                LastWriteTime         Length Name
----                -
-a----             19.02.2024   15:38           33280 biharmonic.exe
-a----             19.02.2024   15:38           38400 biot.exe
-a----             19.02.2024   18:00           95232 CG.exe
-a----             19.02.2024   18:00           33280 CMPVEC.exe
-a----             19.02.2024   18:00           42496 Convert.exe
-a----             19.02.2024   15:38           35328 deadoil.exe
-a----             19.02.2024   15:38           34304 deadoil_grid.exe
-a----             19.02.2024   18:00           97792 PCG.exe
-a----             19.02.2024   15:38           33280 poisson.exe
-a----             19.02.2024   15:38           32256 scalar_grid.exe
-a----             19.02.2024   15:38           35840 stokes.exe
-a----             19.02.2024   15:38           33280 stokes_grid.exe
-a----             19.02.2024   18:00           36864 SYMM.exe
PS C:\Users\kiril\Documents\Code\prak\mipt-solvers-build>
```

Рис. 8: Пример компиляции на windows.

Переключение материалов лекции

```
terekhov@dodo:~/mipt-solvers-build
terekhov@dodo ~/mipt-solvers-build $ cd ../mipt-solvers
terekhov@dodo ~/mipt-solvers $ git checkout lect1
Branch 'lect1' set up to track remote branch 'lect1' from 'origin'.
Switched to a new branch 'lect1'
terekhov@dodo ~/mipt-solvers $ cd ../mipt-solvers-build/
terekhov@dodo ~/mipt-solvers-build $ cmake --build .
-- Configuring done
-- Generating done
-- Build files have been written to: /home/terekhov/mipt-solvers-build
Scanning dependencies of target PCG
[ 10%] Building CXX object CMakeFiles/PCG.dir/Utils/pcg.cpp.o
[ 20%] Linking CXX executable PCG
[ 20%] Built target PCG
Scanning dependencies of target CMPVEC
[ 30%] Building CXX object CMakeFiles/CMPVEC.dir/Utils/cmpvec.cpp.o
[ 40%] Linking CXX executable CMPVEC
[ 40%] Built target CMPVEC
Scanning dependencies of target CG
[ 50%] Building CXX object CMakeFiles/CG.dir/Utils/cg.cpp.o
[ 60%] Linking CXX executable CG
[ 60%] Built target CG
Scanning dependencies of target SYMM
[ 70%] Building CXX object CMakeFiles/SYMM.dir/Utils/symm.cpp.o
[ 80%] Linking CXX executable SYMM
[ 80%] Built target SYMM
Scanning dependencies of target Convert
[ 90%] Building CXX object CMakeFiles/Convert.dir/Utils/conv.cpp.o
[100%] Linking CXX executable Convert
[100%] Built target Convert
terekhov@dodo ~/mipt-solvers-build $
```

Рис. 9: Пример компиляции на Linux.

Разбор заданий

Вариант простого задания:

- Статистики по методам в зависимости от роста размера задачи N .
- Из примера [biot.cpp](#) сделать симулятор.

Используя генераторы систем с параметром N , составить статистику по методам решения:

- 1 I - количество линейных итераций.
- 2 T - полное время решение системы.
- 3 M - занимаемая в процессе решения память.

- windows:
 - ./Release/poisson.exe 50
 - ./Release/PCG.exe A.mtx b.txt
- linux:
 - ./poisson 50
 - ./PCG A.mtx b.txt

Запуск теста

```
Windows PowerShell
PS C:\Users\skifri\Documents\Code\mip\solvers-build> .\release\win64.exe 50
PS C:\Users\skifri\Documents\Code\mip\solvers-build> .\release\PCG.exe .\A.M.
\A.txt
Iter: 2500 2500 iterations 12300
Cannot open params_cg.txt
Loaded parameters:
itol = 1e-10
maxiters = 5000
name = PCG
ncon = 45
tol = 1e-07
tol = 1e-10
true_residual = 1
verbosity = 1
Preconditioner: maxiters = 2
Preconditioner: name = Cholesky
Preconditioner: tol = 0
Preconditioner: verbosity = 1
Estimated bounds of eigenvalues: 0:8.
Consumed by D: 0.91809440b.
Eigenvalues: 0.24669781 center: 4.13333 foci: 3.86667
PCG 0 0.0184606 1.84606e-09 true 0.02
PCG 1 0.0847995 1.84606e-09 true 0.0063989
PCG 2 0.0407936 1.84606e-09 true 0.0000627
PCG 3 0.0485608 1.84606e-09 true 0.0550856
PCG 4 0.0418048 1.84606e-09 true 0.051012
PCG 5 0.0381353 1.84606e-09 true 0.0464574
PCG 6 0.0345784 1.84606e-09 true 0.0421372
PCG 7 0.0311755 1.84606e-09 true 0.0379786
PCG 8 0.02791 1.84606e-09 true 0.0339053
PCG 9 0.0247546 1.84606e-09 true 0.0301547
PCG 10 0.0217341 1.84606e-09 true 0.0264740
PCG 11 0.0188347 1.84606e-09 true 0.0229478
PCG 12 0.0160518 1.84606e-09 true 0.0195616
PCG 13 0.0133795 1.84606e-09 true 0.0163122
PCG 14 0.0108102 1.84606e-09 true 0.0131875
PCG 15 0.00832377 1.84606e-09 true 0.0101779
PCG 16 0.00593362 1.84606e-09 true 0.0072464
PCG 17 0.00358444 1.84606e-09 true 0.00444227
PCG 18 0.00170481 1.84606e-09 true 0.00209316
PCG 19 0.00093554 1.84606e-09 true 0.00124679
PCG 20 0.00155884 1.84606e-09 true 0.00173616
PCG 21 0.00078115 1.84606e-09 true 0.00088572
PCG 22 0.000595256 1.84606e-09 true 0.000762077
PCG 23 0.000364731 1.84606e-09 true 0.000450889
PCG 24 0.000255977 1.84606e-09 true 0.000308556
PCG 25 0.000208519 1.84606e-09 true 0.000257726
PCG 26 0.00014705 1.84606e-09 true 0.00017561
PCG 27 0.000234e-09 1.84606e-09 true 0.000187431
PCG 28 5.80988e-05 1.84606e-09 true 7.01796e-05
PCG 29 3.71846e-05 1.84606e-09 true 4.51359e-05
PCG 30 -2.2968e-05 1.84606e-09 true 2.83292e-05
PCG 31 1.27251e-05 1.84606e-09 true 1.56731e-05
PCG 32 6.56429e-06 1.84606e-09 true 7.98877e-06
PCG 33 3.19078e-06 1.84606e-09 true 3.83361e-06
PCG 34 1.48572e-06 1.84606e-09 true 1.81445e-06
PCG 35 7.42658e-07 1.84606e-09 true 9.38925e-07
PCG 36 4.12728e-07 1.84606e-09 true 5.14793e-07
PCG 37 1.85084e-07 1.84606e-09 true 2.32379e-07
PCG 38 8.81356e-08 1.84606e-09 true 1.07754e-07
PCG 39 4.83153e-08 1.84606e-09 true 6.00045e-08
PCG 40 2.03118e-08 1.84606e-09 true 2.45495e-08
PCG 41 1.12314e-08 1.84606e-09 true 1.39118e-08
PCG 42 5.25787e-09 1.84606e-09 true 6.34714e-09
PCG 43 2.18037e-09 1.84606e-09 true 2.69006e-09
PCG 44 4.29567e-09 1.84606e-09 true 2.50718e-09
PCG 45 1.27857e-09 1.84606e-09 true 1.56224e-09
NS 46 true 1.56224e-09
Time setup 0.0025754 sec iterations 0.0763784 sec solve 0.0794439 sec
Solver consumed: 97 KB
Matrix consumed: 133 KB
Vector consumed: 19 KB
Final residual 1.56224e-09
PS C:\Users\skifri\Documents\Code\mip\solvers-build>

Linux
terehek@bade:/mip/solvers-build$ ./release/win64.exe 50
terehek@bade:/mip/solvers-build$ ./release/PCG.exe .\A.M.txt
Iter: 2500 2500 iterations 12300
Cannot open params_cg.txt
Loaded parameters:
itol = 1e-10
maxiters = 5000
name = PCG
ncon = 45
tol = 1e-07
tol = 1e-10
true_residual = 1
verbosity = 1
Preconditioner: maxiters = 2
Preconditioner: name = Cholesky
Preconditioner: tol = 0
Preconditioner: verbosity = 1
Estimated bounds of eigenvalues: 0:8.
Consumed by D: 0.91809440b.
Eigenvalues: 0.24669781 center: 4.13333 foci: 3.86667
PCG 0 0.0184606 1.84606e-09 true 0.02
PCG 1 0.0847995 1.84606e-09 true 0.0063989
PCG 2 0.0407936 1.84606e-09 true 0.0000627
PCG 3 0.0485608 1.84606e-09 true 0.0550856
PCG 4 0.0418048 1.84606e-09 true 0.051012
PCG 5 0.0381353 1.84606e-09 true 0.0464574
PCG 6 0.0345784 1.84606e-09 true 0.0421372
PCG 7 0.0311755 1.84606e-09 true 0.0379786
PCG 8 0.02791 1.84606e-09 true 0.0339053
PCG 9 0.0247546 1.84606e-09 true 0.0301547
PCG 10 0.0217341 1.84606e-09 true 0.0264740
PCG 11 0.0188347 1.84606e-09 true 0.0229478
PCG 12 0.0160518 1.84606e-09 true 0.0195616
PCG 13 0.0133795 1.84606e-09 true 0.0163122
PCG 14 0.0108102 1.84606e-09 true 0.0131875
PCG 15 0.00832377 1.84606e-09 true 0.0101779
PCG 16 0.00593362 1.84606e-09 true 0.0072464
PCG 17 0.00358444 1.84606e-09 true 0.00444227
PCG 18 0.00170481 1.84606e-09 true 0.00209316
PCG 19 0.00093554 1.84606e-09 true 0.00124679
PCG 20 0.00155884 1.84606e-09 true 0.00173616
PCG 21 0.00078115 1.84606e-09 true 0.00088572
PCG 22 0.000595256 1.84606e-09 true 0.000762077
PCG 23 0.000364731 1.84606e-09 true 0.000450889
PCG 24 0.000255977 1.84606e-09 true 0.000308556
PCG 25 0.000208519 1.84606e-09 true 0.000257726
PCG 26 0.00014705 1.84606e-09 true 0.00017561
PCG 27 0.000234e-09 1.84606e-09 true 0.000187431
PCG 28 5.80988e-05 1.84606e-09 true 7.01796e-05
PCG 29 3.71846e-05 1.84606e-09 true 4.51359e-05
PCG 30 -2.2968e-05 1.84606e-09 true 2.83292e-05
PCG 31 1.27251e-05 1.84606e-09 true 1.56731e-05
PCG 32 6.56429e-06 1.84606e-09 true 7.98877e-06
PCG 33 3.19078e-06 1.84606e-09 true 3.83361e-06
PCG 34 1.48572e-06 1.84606e-09 true 1.81445e-06
PCG 35 7.42658e-07 1.84606e-09 true 9.38925e-07
PCG 36 4.12728e-07 1.84606e-09 true 5.14793e-07
PCG 37 1.85084e-07 1.84606e-09 true 2.32379e-07
PCG 38 8.81356e-08 1.84606e-09 true 1.07754e-07
PCG 39 4.83153e-08 1.84606e-09 true 6.00045e-08
PCG 40 2.03118e-08 1.84606e-09 true 2.45495e-08
PCG 41 1.12314e-08 1.84606e-09 true 1.39118e-08
PCG 42 5.25787e-09 1.84606e-09 true 6.34714e-09
PCG 43 2.18037e-09 1.84606e-09 true 2.69006e-09
PCG 44 4.29567e-09 1.84606e-09 true 2.50718e-09
PCG 45 1.27857e-09 1.84606e-09 true 1.56224e-09
NS 46 true 1.56224e-09
Time setup 0.0025754 sec iterations 0.0763784 sec solve 0.0794439 sec
Solver consumed: 97 KB
Matrix consumed: 133 KB
Vector consumed: 19 KB
Final residual 1.56224e-09
terehek@bade:/mip/solvers-build$
```

Рис. 10: Запуск примера на windows и linux.

```
PCG 42 5.25787e-09 | 1.84666e-09 true 6.34714e-09
PCG 43 3.19639e-09 | 1.84666e-09 true 3.90904e-09
PCG 44 2.05678e-09 | 1.84666e-09 true 2.50718e-09
PCG 45 1.27857e-09 | 1.84666e-09 true 1.56224e-09
PCG 45 true 1.56224e-09
Time: 0.0035334 sec iterations 0.0768704 sec solve 0.0794439 sec
Solver consumed 97 KB
Matrix consumed: 199 KB
Vector consumed: 19 KB
Final residual 1.56224e-09
PS C:\Users\Kiril\Documents\Code\mpt-solvers-build>
```

Рис. 11: Данные в примере.

- 1 I - количество линейных итераций.
- 2 T - полное время решение системы.
- 3 M - занимаемая в процессе решения память.

N	25	50	100	200	400	800
I	23	45	87	173	347	698
T	$8.25 \cdot 10^{-4}$	$3.72 \cdot 10^{-3}$	$2.54 \cdot 10^{-2}$	0.21	2.12	17.9
M	24	97	390	1562	6250	25000

Таблица 1: Пример таблицы для метода **PCG** и системы **poisson**

Изменяя способ и параметр отбрасывания в методах на основе неполной факторизации на фиксированной системе:

- 1 I - количество линейных итераций.
- 2 T - полное время решение системы.
- 3 P - время построения предобуславливателя.
- 4 M - занимаемая в процессе решения память.

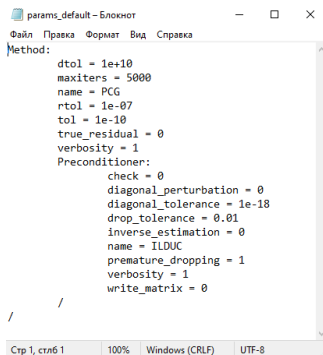
```
PCG 42 5.25787e-09 | 1.84606e-09 true 6.34714e-09
PCG 43 3.19639e-09 | 1.84606e-09 true 3.90904e-09
PCG 44 2.05678e-09 | 1.84606e-09 true 2.50718e-09
PCG 45 1.27857e-09 | 1.84606e-09 true 1.56224e-09
PCG 45 true 1.56224e-09
Time [setup 0.0025734 sec iterations 0.0768704 sec solve 0.0794439 sec
Solver consumed: 97 KB
Matrix consumed: 153 KB
Vector consumed: 19 KB
Final residual 1.56224e-09
PS C:\Users\kiril\Documents\Code\nipt-solvers-build>
```

Рис. 12: Данные в примере.

Изменение параметров

Как поменять параметры:

- При запуске исполняемого файла без параметров создается файл **params_default.txt**.
- Исполняемый файл считывает параметры из файла **params.txt**.



```
params_default - Блокнот
Файл  Правка  Формат  Вид  Справка
Method:
dtol = 1e+10
maxiters = 5000
name = PCG
rtol = 1e-07
tol = 1e-10
true_residual = 0
verbosity = 1
Preconditioner:
  check = 0
  diagonal_perturbation = 0
  diagonal_tolerance = 1e-18
  drop_tolerance = 0.01
  inverse_estimation = 0
  name = ILDUC
  premature_dropping = 1
  verbosity = 1
  write_matrix = 0
/
/

Стр 1, столб 1    100%  Windows (CRLF)  UTF-8
```

Рис. 13: Параметры PCG_ILDUC.exe

Изменение параметров

Как поменять параметры:

- **drop_tolerance** - параметр отбрасывания τ .
- **inverse_estimation** - оценка обратных факторов.

```
params_default - Блокнот
Файл  Правка  Формат  Вид  Справка
Method:
dtol = 1e+10
maxiters = 5000
name = PCG
rtol = 1e-07
tol = 1e-10
true_residual = 0
verbosity = 1
Preconditioner:
  check = 0
  diagonal_perturbation = 0
  diagonal_tolerance = 1e-18
  drop_tolerance = 0.01
  inverse_estimation = 0
  name = ILDUC
  premature_dropping = 1
  verbosity = 1
  write_matrix = 0
/
/
Стр 1, столб 1    100%    Windows (CRLF)    UTF-8
```

Рис. 14: Параметры **PCG_ILDUC.exe**

τ	0.1	0.01	0.001	0.0001	0
I	65	30	15	7	1
T	0.11	$7.5 \cdot 10^{-2}$	0.11	0.23	1.15
P	$3.18 \cdot 10^{-2}$	$4.17 \cdot 10^{-2}$	$8.02 \cdot 10^{-2}$	0.21	1.16
M	1445	2645	5346	11421	25092

Таблица 2: Пример таблицы для метода **PCG_ILDUC** и системы **poisson** с $N = 100$ и **inverse_estimation** = 0.

Для седловых [lect10](#), [lect11](#) и блочных систем [lect11](#)

- Для системы Стокса:
 - `./Release/stokes.exe N`
 - `./Release/BPCG_AMG_GS.exe N * (N + 1) * 2 A.mtx b.txt`
- Для системы Био:
 - `./Release/biot.exe N`
 - `./Release/FIXED_STRESS_AMG.exe N * (N + 1) * 2 A.mtx b.txt`
- Для системы двухфазной фильтрации:
 - `./Release/deadoil.exe N`
 - `./Release/CPR_*.exe N * N A.mtx b.txt` (для всех методов CPR)

Выводы по статистике:

- 1 Какие из методов лучше выбрать для каждой из рассматриваемых систем?
- 2 Какие из методов применимы к блочным и седловым системам?
- 3 Какие из методов имеют линейную сложность решения?
- 4 Как зависит число итераций и время решения от параметра отбрасывания τ в методе неполной факторизации?
- 5 Результаты в таблицах должны подтверждать выводы.

Выборка тестов

лекция	метод	poisson	biot	stokes	deadoil	biharmonic
lect1	CG	+	+	-	-	+
lect1	PCG	+	+	-	-	+
lect2	PCG_ILDUC	+	±	±	-	±
lect2	BICGSTAB_ILDUC	+	±	±	±	±
lect4	PCG_AMG_CHEB	+	-	-	-	-
lect5	PCG_AMG_JAC	+	-	-	-	-
lect5	PCG_AMG_GS	+	-	-	-	-
lect6	MLILDUCS	+	±	±	±	±
lect7	MLILDUCM	+	±	±	±	±
lect8	MPT_ILDUC	+	±	±	±	±
lect8	SYM_ILDUC	+	±	±	±	±
lect8	MPT_WRCM_ILDUC	+	±	±	±	±
lect8	SYM_WRCM_ILDUC	+	±	±	±	±
lect9	MPT_WRCM_MLILDUCS	+	±	±	±	±
lect9	MPT_WRCM_MLILDUCM	+	±	±	±	±
lect10	BPCG_AMG_GS	-	-	+	-	-
lect10	VANKA	-	-	+	-	-
lect11	CPR_TS_ILU	-	-	-	+	-
lect11	CPR_TS_GS	-	-	-	+	-
lect11	CPR_TSGS_ILU	-	-	-	+	-
lect11	CPR_TSGS_GS	-	-	-	+	-
lect11	FIXED_STRESS_AMG	-	+	-	-	-

Таблица 3: Тесты методов на рост размеры системы.

Варианты с \pm могут не решаться или заметно зависеть от параметров и размера системы.

Из примера [biot.cpp](#) сделать симулятор.

- 1 Добавить цикл шагов по времени.
- 2 Добавить вызов метода для решения системы.
- 3 Модифицировать правую часть системы в зависимости от решения на прошлом шаге.
- 4 Вывод файлов сетки в каждый момент времени (как в [stokes_grid.cpp](#)).
- 5 Поэкспериментировать с расстановкой скважин.
- 6 Пример симулятора двухфазной фильтрации на основе [deadoil.cpp](#) находится в ветке [lect12](#). Задача двухфазной фильтрации нелинейная и требует дополнительный цикл для метода Ньютона, в то время как задача Био - линейная.

Пример `biot.cpp`:

- Система уравнений:

$$\begin{aligned} -\mu\Delta\vec{u} - (\mu + \lambda)\nabla\operatorname{div}(\vec{u}) + \alpha\nabla p &= \mathbf{b}, \\ \partial_t(\zeta p + \alpha\operatorname{div}(\vec{u})) - \kappa\Delta p &= q, \end{aligned}$$

- μ, λ - параметры Ламэ, κ - проницаемость среды, α - коэффициент Био, ζ - ёмкость породы, $\theta = \zeta p + \alpha\operatorname{div}(\vec{u})$ - пористость;
- начальные условия: нулевые, $u^0 = 0, p^0 = 0$;
- граничные условия на жидкость: непротекание;
- граничные условия на породу: перемещение фиксировано.

Пример [biot.cpp](#):

- Зависимость от прошлого шага по времени в $\partial_t (\zeta p + \alpha \operatorname{div}(\vec{u}))$.
- Дискретизация по времени:

$$\partial_t (\zeta p + \alpha \operatorname{div}(\vec{u})) \approx \zeta \frac{p^{n+1} - p^n}{\Delta t} + \alpha \operatorname{div} \left(\frac{u^{n+1} - u^n}{\Delta t} \right).$$

- Дискретизация по пространству:

$$\operatorname{div}(\vec{u}) = \partial_x u + \partial_y v \approx \frac{u_{i+1,j} - u_{i,j}}{h} + \frac{v_{i,j+1} - v_{i,j}}{h}$$

- Коэффициенты при неизвестных с p^{n+1} и u^{n+1} попадают в матрицу.
- Прибавка к правой части q :

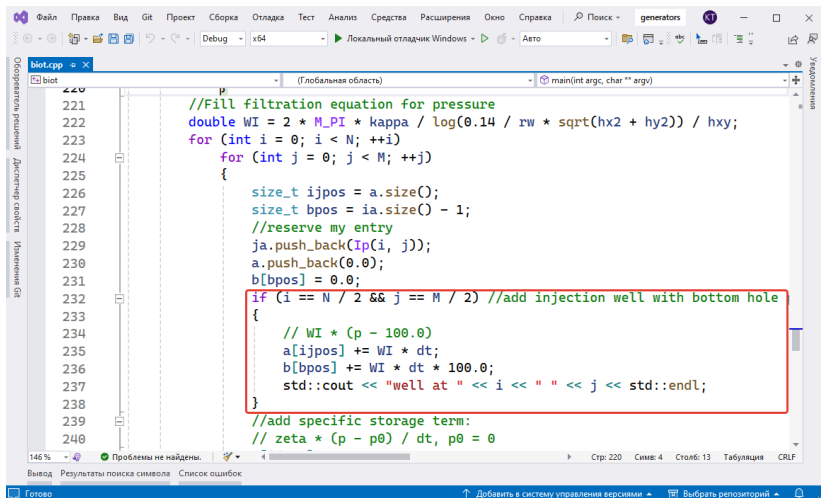
$$\zeta \frac{p^n}{\Delta t} - \alpha \frac{u_{i+1,j}^n - u_{i,j}^n}{h \Delta t} - \alpha \frac{v_{i,j+1}^n - v_{i,j}^n}{h \Delta t}$$

Пример [biot.cpp](#):

- Формула Писмана для скважины радиуса r_w с забойным давлением p_{bhp} :

$$q = 2\pi\kappa(p - p_{bhp}) / \log\left(\frac{14\sqrt{2}h}{100r_w}\right).$$

- r_w, κ - вводные параметры, по умолчанию $r_w = 10^{-4}$, $\kappa = 1$.
- Задано положение скважины в середине области $i = N/2, j = M/2$ и давление $p_{bhp} = 100$.
- Можно поменять или добавить еще несколько скважин в других местах и с другим забойным давлением, например, зависящим от времени.



```
220
221 //Fill filtration equation for pressure
222 double WI = 2 * M_PI * kappa / log(0.14 / rw * sqrt(hx2 + hy2)) / hxy;
223 for (int i = 0; i < N; ++i)
224     for (int j = 0; j < M; ++j)
225     {
226         size_t ijpos = a.size();
227         size_t bpos = ia.size() - 1;
228         //reserve my entry
229         ja.push_back(Ip(i, j));
230         a.push_back(0.0);
231         b[bpos] = 0.0;
232         if (i == N / 2 && j == M / 2) //add injection well with bottom hole
233         {
234             // WI * (p - 100.0)
235             a[ijpos] += WI * dt;
236             b[bpos] += WI * dt * 100.0;
237             std::cout << "well at " << i << " " << j << std::endl;
238         }
239         //add specific storage term:
240         // zeta * (p - p0) / dt, p0 = 0
```

146% Проблемы не найдены. Стр: 220 Симв: 4 Столб: 13 Табуляция CRLF

Готово ↑ Добавить в систему управления версиями ▾ Выбрать репозиторий ▾

Рис. 15: Участок кода для скважины в biot.cpp.

- В результате работы примера `biot.cpp` собирается массивы ia, ja, a матрицы в CSR-формате и вектор правой части b .
- Примеры с методами решения принимают на вход объект класса `CSRMatrix`, который имеет конструктор вида `CSRMatrix(ia,ja,a)`.
- В качестве примера передачи матрицы в решатель можно посмотреть на код `utils/deadoil_nln.cpp` в материалах [lect12](#).
- Для визуализации вектора решения можно использовать `stokes_grid.cpp` из материалов [lect0](#).

Пример (Команды в python)

- `scipy.sparse.csr_matrix` – создать матрицу в формате CSR;
- `scipy.sparse.coo_matrix` – создать матрицу в формате COO;
- `scipy.io.mmread` – считать файл формата *it MatrixMarker*;
- `matplotlib.pyplot.spy` – отрисовать “след” матрицы;
- `scipy.linalg.solve` – решить систему (по умолчанию используется пакет *UMFPACK*).

Можно перевести формирование системы в python.

Вариант сложного задания:

- либо из примера [deadoil.cpp](#) сделать симулятор с использованием данных SPE10,
- либо из комбинации примеров [biot.cpp](#) и [deadoil.cpp](#) сделать новый пример для задачи двухфазной фильтрации в пороупругой среде с использованием данных SPE10.
- Пример симулятора двухфазной фильтрации на основе [deadoil.cpp](#) находится в ветке [lect12](#).

Можно перевести формирование системы в python.

Данные SPE10:

- Ссылка на данные
<https://www.spe.org/web/csp/datasets/set02.htm#download>.
- В архиве лежат файлы **spe_phi.dat** и **spe_perm.dat** в ASCII формате.
- Файл **spe_phi.dat** содержит информацию о пористости θ в каждой ячейке.
- Файл **spe_perm.dat** содержит информацию о тензоре κ в каждой ячейке.

Данные SPE10:

- Данные проницаемости заданы в виде диагональной матрицы:

$$\kappa = \begin{bmatrix} \kappa_x & & \\ & \kappa_y & \\ & & \kappa_z \end{bmatrix}, \quad \kappa_x = \kappa_y \neq \kappa_z.$$

- Будем считать, что проницаемость скалярная, возьмем $\kappa = \kappa_x$.
- В данных есть непроницаемые ячейки с нулевыми θ и κ : можно заменить их минимальными значениями, например, 10^{-4} , чтобы система была не сингулярна,
- Размеры массивов: $60 \times 220 \times 85 = 1\,122\,000$ ячеек, можно взять только один слой, например 45-ый.

Данные SPE10:

- Размеры массивов: $60 \times 220 \times 85 = 1\,122\,000$ ячеек, можно взять только один слой данных по вертикали, например, 45-ый.
- Для уменьшения задачи из слоя можно выделить отдельный участок данных.
- Физические размеры ячеек: $20\text{ ft} \times 10\text{ ft} \times 2\text{ ft}$.

Пример (Считывание пористости в C++)

- `std::ifstream fporo("spe_phi.dat");`
- `double poro[60][220][85];`
- `for (int k = 0; k < 85; ++k)`
- `for (int j = 0; j < 220; ++j)`
- `for (int i = 0; i < 60; ++i) fporo » poro[i][j][k];`
- `fporo.close();`

Пример (Считывание проницаемости в C++)

- `std::ifstream fperm("spe_phi.dat");`
- `double perm[60][220][85][3];`
- `for (int l = 0; l < 3; ++l)`
- `for (int k = 0; k < 85; ++k)`
- `for (int j = 0; j < 220; ++j)`
- `for (int i = 0; i < 60; ++i) fperm » perm[i][j][k][l];`
- `fperm.close();`

Пример (Считывание в python)

- `import numpy as np`
- `phi = np.loadtxt("spe_phi.dat").reshape((60 * 220 * 85)).reshape((60,220,85),order='F')`
- `perm = np.loadtxt("spe_perm.dat").reshape((60 * 220 * 85 * 3)).reshape((60,220,85,3),order='F')`

Считывание данных SPE10

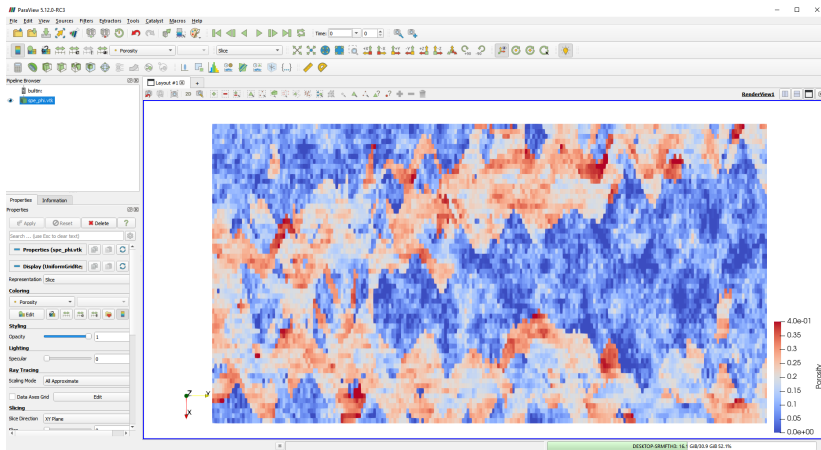


Рис. 16: 45-ый слой данных пористости из данных SPE10 в paraview.

Из примера [deadoil.cpp](#) сделать симулятор.

- 1 Учесть неоднородность данных пористости и проницаемости.
- 2 Добавить цикл шагов по времени.
- 3 Добавить цикл метода Ньютона решения нелинейной задачи.
- 4 Добавить вызов метода для решения системы.
- 5 Модифицировать правую часть системы в зависимости от решения на прошлом шаге.
- 6 Расположить скважины.
- 7 Пример двухфазной фильтрации на основе [deadoil.cpp](#) находится в ветке [lect12](#): можно сделать на основе этого примера.

Уравнение двухфазной фильтрации

Примеры `deadoil.cpp` и `lect12`:

- Описывают вытеснение нефти водой.
- Неизвестные: $S \in [0, 1]$ - насыщенность нефти, p - давление воды.
- Система уравнений:

$$\begin{cases} \partial_t \theta S - \operatorname{div}(S \kappa \nabla p) = q_o \\ \partial_t \theta (1 - S) - \operatorname{div}((1 - S) \kappa \nabla p) = q_w \end{cases}$$

- Граничные условия: непротекание.
- Скважина радиуса r_w с забойным давлением p_{bhp} : $q = 2\pi\kappa(p - p_{bhp}) / \log\left(\frac{14\sqrt{2}h}{100r_w}\right)$, $q_o = Sq$, $q_w = (1 - S)q$.
- θ, κ - данные из SPE10.

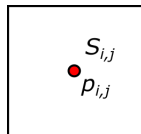


Рис. 17: Степени свободы

Примеры [deadoil.cpp](#) и [lect12](#):

- Определим поток: $q = -\kappa n \cdot \nabla p$.
- На регулярной прямоугольной сетке с шагом h и неоднородным коэффициентом κ :

$$q = -\kappa n \cdot \nabla p = -\kappa \frac{\partial p}{\partial x}.$$

- Введем давление на грани p_f и шаг сетки h , тогда:

$$q \approx \kappa_1 \frac{p_1 - p_f}{h/2} = \kappa_2 \frac{p_f - p_2}{h/2}.$$

- Решим относительно p_f и подставим в q :

$$p_f = \frac{\kappa_1 p_1 + \kappa_2 p_2}{\kappa_1 + \kappa_2}, \quad q = \frac{2\kappa_1 \kappa_2}{\kappa_1 + \kappa_2} \frac{p_1 - p_2}{h}.$$

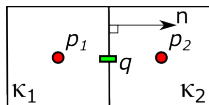


Рис. 18: Поток

Связка `deadoil.cpp` и `biot.cpp`:

- Неизвестные: смещение породы $\vec{u} = [u, v]^T$ и давление жидкости p .
- Система уравнений:

$$\begin{cases} -\mu\Delta\vec{u} - (\mu + \lambda)\nabla\text{div}(\vec{u}) + \alpha\nabla p = \mathbf{b} \\ \partial_t\theta S - \text{div}(S\kappa\nabla p) = q_o \\ \partial_t\theta(1 - S) - \text{div}((1 - S)\kappa\nabla p) = q_w \end{cases}$$

- $\theta = \theta_0 + \zeta p + \alpha\text{div}(\vec{u})$ - пористость.
- θ_0, κ - данные из SPE10.
- μ, λ - заданные константы.

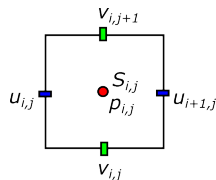


Рис. 19: Степени свободы