Московский государственный университет
им. М.В.Ломоносова
Механико-математический факультет

На правах рукописи
УДК 519.6

Богачев Кирилл Юрьевич

Эффективное решение задачи фильтрации вязкой сжимаемой многофазной многокомпонентной смеси на параллельных ЭВМ

05.13.18 – Математическое моделирование, численные методы и комплексы программ

ДИССЕРТАЦИЯ
на соискание ученой степени
доктора физико-математических наук

Москва – 2012

Содержание

Список иллюстраций

Введение

Актуальность работы Разрешающая способность сеточных данных и сложность математической модели, используемой при описании фильтрационных течений многофазной многокомпонентной смеси в нефтегазовых месторождениях, постоянно растут. Это неизбежно приводит к возрастанию времени расчета при численном решении задачи моделирования процессов разработки углеводородного сырья. При этом все компании-поставщики микропроцессоров повышают производительность своей продукции в основном за счет увеличения количества вычислительных ядер. Поэтому для достижения максимальной эффективности расчетов требуется разрабатывать методы решения, максимально использующие параллельность ЭВМ. Несмотря на большие усилия в этом направлении, из-за специфики задачи фильтрации, большинство программ для ее численного решения не показывают ускорения, более чем в 12–16 раз, по сравнению с временем работы последовательной программы, а начиная с 20–30 использованных логических процессоров время расчета начинает возрастать. Таким образом, решение задачи фильтрации оказалось в стороне от развития высокопроизводительных вычислений, где речь уже идет о сотнях и тысячах процессоров.

Цель диссертационной работы состоит в создании эффективного метода расчета и соответствующего комплекса программ для решения задачи фильтрации вязкой сжимаемой многофазной многокомпонентной смеси на параллельных ЭВМ.

Для достижения поставленных целей были решены следующие задачи:

- разработан метод блочного хранения разреженных несимметричных матриц в памяти и блочные варианты построения предобуславливателей, основанных на неполном LU разложении;

- разработан параллельный алгоритм построения предобуславливателей, основанных на неполном LU разложении;
- разработан метод использования ресурсов ЭВМ с неоднородным доступом к памяти (NUMA);
- разработан метод построения гибридного MPI–многопоточного вычислительного процесса для решения задачи фильтрации на ЭВМ с распределенной памятью (кластере).

Научная новизна работы состоит в разработке метода построения гибридного MPI–многопоточного вычислительного процесса для решения задачи фильтрации на ЭВМ с распределенной памятью (кластере), позволившего достигать ускорения, более чем в 50 раз, по сравнению с временем работы последовательной программы, причем сокращение времени работы идет даже при использовании более 200 логических процессоров.

Практическая значимость. Результаты, изложенные в диссертации, могут быть использованы при численном решении задачи моделирования процессов разработки углеводородного сырья, а также при решении других систем уравнений в частных производных на параллельных ЭВМ.

На защиту выносятся следующие основные результаты и положения:

- математическая модель техногенной трещиноватости вблизи скважин;
- метод блочного хранения разреженных несимметричных матриц в памяти и блочные варианты построения предобуславливателей, основанных на неполном LU разложении;
- алгоритм распараллеливания построения предобуславливателей, основанных на неполном LU разложении;

- эффективная технология, включающая алгоритмы, структуры данных и комплекс программ, для построения гибридного MPI–многопоточного вычислительного процесса для решения задачи фильтрации на ЭВМ с распределенной памятью (кластере).

- комплекс программ tNavigator для решения задачи фильтрации на современных параллельных ЭВМ.

Апробация работы. Основные результаты диссертации докладывались на следующих конференциях:

- Уфимская международная математическая конференция "Теория функций, дифференциальные уравнения, вычислительная математика" (Уфа, 2007)

- 4-я международная конференция "Математические идеи П.Л. Чебышева и их приложения к современным проблемам естествознания" (Обнинск, 2008)

- Научная конференция "ЛОМОНОСОВСКИЕ ЧТЕНИЯ" — 2008 (Москва, 2008)

- Всероссийская конференция по вычислительной математике КВМ-2009 (Академгородок, Новосибирск, 2009)

- Научная конференция "ЛОМОНОСОВСКИЕ ЧТЕНИЯ" — 2009 (Москва, 2009)

- Научная конференция "ЛОМОНОСОВСКИЕ ЧТЕНИЯ" — 2010 (Москва, 2010)

- Конференция "Суперкомпьютеры в нефтегазовой отрасли" (Москва, 2010)

Также основные результаты диссертации докладывались на научно-исследовательских семинарах:

- Кафедры вычислительной математики механико-математического факультета МГУ им. М.В. Ломоносова
- Института математического моделирования РАН (ИММ РАН)
- Кафедры механики композиционных материалов механико-математического факультета МГУ им. М.В. Ломоносова
- Вычислительного центра РАН (ВЦ РАН)
- Института вычислительной математики РАН (ИВМ РАН)
- Института проблем безопасного развития атомной энергетики РАН (ИБРАЭ РАН)

Разработанный в диссертационной работе комплекс программ tNavigator свободно доступен для загрузки с сайта `rfdyn.ru`.

Публикации. Материалы диссертации опубликованы в 14 печатных работах, из них 4 монографии [1–4], 8 статей в рецензируемых журналах [5–12], 2 статьи в специализированных журналах [13, 14].

Отметим также, что без предшествующих работ [15–29], посвященных эффективным методам решения жестких нелинейных уравнений в частных производных, и монографии [30], посвященной архитектурам операционных систем, эта работа тоже была бы невозможной.

Личный вклад автора. Содержание диссертации и основные положения, выносимые на защиту, отражают персональный вклад автора в опубликованные работы. Подготовка к публикации полученных результатов проводилась совместно с соавторами, причем вклад диссертанта был определяющим.

Структура и объем диссертации. Диссертация состоит из введения, обзора литературы, 4 глав, заключения и библиографии. Общий объем диссертации 201 страница, из них 184 страницы текста, включая 62 рисунка. Библиография включает 127 наименований на 17 страницах.

Обзор литературы

Постановке задачи фильтрации вязкой сжимаемой многофазной многокомпонентной смеси в пористой среде посвящено множество классических книг, ставших фактически учебниками: [31], [32], [33], [34], [35], [36], [37], [38], [39], [40], [41].

В настоящем разделе мы дадим самодостаточное изложение задачи для последующего исследования. Все детали, касающиеся постановки задачи, могут быть взяты из указанных выше источников.

1. Постановка задачи

Рассмотрим стандартную трехфазную трехкомпонентную изотермическую модель черной нефти (black-oil). Форма уравнений выбрана максимально приближенной к многокомпонентной модели (т.н. композиционной) для облегчения развития комплекса программ в будущем.

1.1. Основные определения

Выделяют три фазы и три или более (в композиционном случае) компонент:

- водная фаза (**вода**) — не смешивается с углеводородными фазами, состоит из одной компоненты – **воды**;
- жидкая углеводородная фаза (**нефть**) — состоит из смеси **углеводородных компонент**, находящихся при данном давлении и концентрациях других компонент в жидком состоянии;
- газообразная углеводородная фаза (**газ**) — состоит из смеси **углеводородных компонент**, находящихся при данном давлении и концентраци-

ях других компонент в газообразном состоянии.

Каждый элементарной объем V_b представляется в виде

$$V_b = V_R + V_p, \quad V_p = V_W + V_O + V_G$$

где

- V_R – **объем породы** (участвует в описании тепловых процессов),
- V_p – **поровый объем** (доступный для заполнения пластовой смесью),
- V_P, $P = W, O, G$ – **объем фазы** воды, нефти, газа.

Пористость ϕ — доля объема, доступная для заполнения смесью:

$$\phi = \frac{V_p}{V_b}$$

Насыщенность S_P фазы ($P = W, O, G$) — доля порового объема, занимаемая фазой:

$$S_P = \frac{V_P}{V_p} = \frac{V_P}{V_W + V_O + V_G}, \quad P = W, O, G, \quad S_W + S_O + S_g = 1. \tag{1}$$

1.2. Параметры фаз

Состояние каждой фазы P, $P = W, O, G$ (Water, Oil, Gas) задается следующими величинами (**неизвестными**, определяются в процессе расчета):

- $p_P = p_P(t, x, y, z)$ – **давление** в фазе P;
- $S_P = S_P(t, x, y, z)$ ($P = W, O, G$) – **насыщенность** P-ой фазы.

Количество неизвестных величин обычно сокращают с помощью соотношений

$$p_O - p_G = P_{cOG},$$

$$p_O - p_W = P_{cOW},$$

$$S_W + S_O + S_G = 1.$$

где $P_{cOG} = P_{cOG}(S_g)$ – **капиллярное давление** в системе нефть-газ и $P_{cOW} = P_{cOW}(S_w)$ – капиллярное давление в системе вода-нефть (заданные функции). В дальнейшем под давлением будем понимать давление в нефтяной фазе $p = p_O$. Удобно обозначить $P_{cP} = -P_{cOP}$, где $P_{cOO} = 0$, и считать $p_P = p + P_{cP}$.

1.3. Параметры компонент

Обозначим: $N_c = N_c(t,x,y,z)$ – **молярная плотность** компонента c, $c = 1,\ldots,n_c$ в поровом объеме, так, что $N_c \cdot V_p = N_c \cdot \phi \cdot V_b$ – количество компоненты c в объеме V_b.

Распределение компонент по фазам задается $n_c \times n_P$ **матрицей концентраций** $x_{c,P} = x_{c,P}(p_P, \boldsymbol{N})$, $\boldsymbol{N} = (N_1,\ldots,N_{n_c})$:

$$\sum_{P=1}^{n_P} x_{c,P}\xi_P S_P = N_c, \quad c \in \{1,\ldots,n_c\}. \tag{2}$$

Здесь $\xi_P = \xi_P(p_P, \boldsymbol{N})$ – **молярная плотность** фазы P. Выполнены соотношения:

$$\sum_{c=1}^{n_c} x_{c,P} = 1, \quad P \in \{1,\ldots,n_P\}, \qquad x_{c,W} = 0, \quad c \in \{2,\ldots,n_c\}. \tag{3}$$

1.4. Дифференциальные уравнения

Уравнения имеют вид

$$\frac{\partial}{\partial t}(\phi N_c) = \operatorname{div} \sum_{P=O,W,G} x_{c,P}\xi_P \left(\boldsymbol{k}\frac{k_{rP}}{\mu_P}(\nabla p_P - \gamma_P \nabla D) \right) + q_c,\ c = 1,\ldots,n_c \tag{4}$$

$$p_O - p_G = P_{cOG}, \tag{5}$$

$$p_O - p_W = P_{cOW}, \tag{6}$$

$$S_W + S_O + S_G = 1. \tag{7}$$

Здесь используются следующие обозначения для функций

- $N_c = N_c(t,x,y,z)$ (**неизвестна**) – $c = 1,\ldots,n_c$ молярная плотность компонента.

Для модели черной нефти $n_c = 3$, компоненты – вода, нефть и газ:

$$N_W = \xi_{W,SC}\frac{S_W}{B_W},\ N_o = \xi_{O,SC}\left(\frac{S_O}{B_O} + R_{O,G}\frac{S_G}{B_G}\right),\ N_G = \xi_{G,SC}\left(\frac{S_G}{B_G} + R_{G,O}\frac{S_O}{B_O}\right)$$

  - $S_P = S_P(t,x,y,z)$ (**неизвестна**) - насыщенность P-ой фазы, $P = O,G,W$,
  - $R_{G,O} = R_{G,O}(p_O)$ – растворимость газа в нефтяной фазе (**известная** функция) (см. 1.11),
  - $R_{O,G} = R_{O,G}(p_O)$ – содержание нефти в газе (**известная** функция) (см. 1.12),
  - $B_P = B_P(p_P, \boldsymbol{N})$ – коэффициент объемного расширения фазы (**известная** функция) (см. 1.14).

- $\phi = \phi(p,x,y,z)$ – пористость (**известная** функция) (см. 1.8),
- $p_W = p_W(t,x,y,z)$ (**неизвестно**) - давление водяной фазы,
- $p_O = p_O(t,x,y,z)$ (**неизвестно**) - давление нефтяной фазы,
- $p_G = p_G(t,x,y,z)$ (**неизвестно**) - давление газовой фазы,
- $x_{c,P} = x_{c,P}(p, \boldsymbol{N})$ (**известная** функция) – молярная доля компонента c в фазе P (см. 1.17),
- $\xi_P = \xi_P(p, \boldsymbol{N})$ – молярная плотность фазы (**известная** функция), (см. 1.15),
- $\boldsymbol{k} = \boldsymbol{k}(p_W, p_O, p_G, x, y, z)$ – тензор абсолютной проницаемости (**известная** функция) (см. 1.7),
- $k_{rP} = k_{rP}(S_W, S_G)$ – относительная фазовая проницаемость (**известная** функция) (см. 1.9),
- $\mu_P = \mu_P(p_P, \boldsymbol{N})$ – вязкость фазы (**известная** функция) (см. 1.13),
- $\gamma_P = \rho_P g$ – вертикальный градиент давления (**известная** зависимость),

- $D = D(x,y,z)$ – вектор глубины (сверху вниз) (**известная** координатная функция),
- $\rho_P = \rho_P(p_P)$ – массовая плотность фазы (**известная** функция) (см. 1.16),
- $P_{cOG} = P_{cOG}(S_G)$ – капиллярное давление в системе нефть-газ (**известная** функция) (см. 1.10),
- $P_{cOW} = P_{cOW}(S_W)$ – капиллярное давление в системе вода-нефть (**известная** функция) (см. 1.10),
- $q_c = q_c(p, \boldsymbol{N}, t, x, y, z)$ – функция источников компонента c (нагнетательные и добывающие скважины, водонапорные горизонты) (**известная** функция) (см. 1.18, 1.19)

и постоянной

- $g = \text{const}$ – **известная** постоянная величина

Далее определим способы ввода известных данных.

1.5. Граничные условия

Используются стандартные граничные условия непротекания (однородные условия Неймана) на внешней границе пласта:

$$\Big(\sum_{P=O,W,G} x_{c,P} \xi_P \Big(\boldsymbol{k} \frac{k_{rP}}{\mu_P} (\nabla p_P - \gamma_P \nabla D) \Big), \boldsymbol{n} \Big) = 0, \quad c = 1, \ldots, n_c \tag{8}$$

1.6. Начальные условия

Начальными условиями могут быть известные значения p_P, S_P (непосредственно заданные), либо значения p_P, S_P, вычисленные из гидростатических

условий равновесия

$$\operatorname{div} \sum_{P=O,W,G} x_{c,P}\xi_P \left(\boldsymbol{k} \frac{k_{rP}}{\mu_P} (\nabla p_P - \gamma_P \nabla D) \right) = 0 \quad (9)$$

$$p_O - p_G = P_{cOG} \quad (10)$$

$$p_O - p_W = P_{cOW} \quad (11)$$

$$S_W + S_O + S_G = 1 \quad (12)$$

с граничными условиями из 1.5.

1.7. Тензор проницаемости

Тензор абсолютной проницаемости $\boldsymbol{k} = \boldsymbol{k}(p_W, p_O, p_G, x, y, z)$ – **заданная** матричная функция, определенная во всех точках пласта. Чаще всего используется диагональный тензор, а зависимость $\boldsymbol{k}$ от давления опускается.

При аппроксимации на сетке обычно размеры блока (элементарной ячейки) значительно отличаются по направлениям (обычно в направлении Z в $10 \ldots 100$ раз меньше, чем по XY). Это приходиться учитывать при аппроксимации тензора $\boldsymbol{k}$, вводя в него **анизотропию**.

1.8. Пористость

Пористость $\phi = \phi(p, x, y, z)$ – **заданная** функция, определенные во всех точках пласта. Обычно представляется в следующем виде:

$$\phi(p, x, y, z) = \psi(x, y, z)\phi(x, y, z)(1 + c(p - p_{\text{ref}}) + c^2(p - p_{\text{ref}})^2/2)$$

где

- $\psi(x, y, z)$ – коэффициент песчанистости (**заданная** функция, определенная во всех точках пласта),
- $\phi(x, y, z)$ – пористость при давлении p_{ref} (**заданная** функция, определенная во всех точках пласта),

- c – сжимаемость (**заданные** данные),
- p_{ref} – опорное давление (**заданные** данные).

1.9. Относительная фазовая проницаемость

Относительная фазовая проницаемость (ОФП) $k_{rP} = k_{rP}(S_W, S_G)$ определяется по экспериментальным данным. Используются следующие стандартные допущения:

$$k_{rW} = k_{rW}(S_W) \tag{13}$$

$$k_{rG} = k_{rG}(S_G) \tag{14}$$

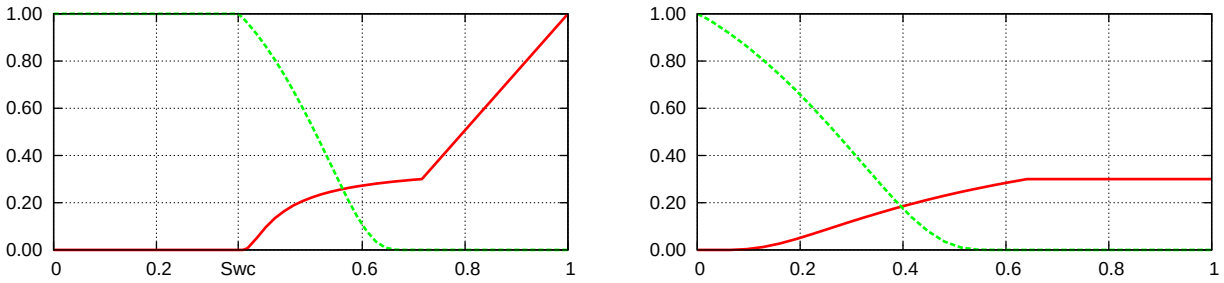
$$k_{rO} = k_{rO}(S_W, S_G) \tag{15}$$

Обычно указываются два набора пар функций относительной проницаемости:

- $k_{rWO} = k_{rWO}(S_W)$, $k_{rOW} = k_{rOW}(S_W)$ — для двухфазной системы вода-нефть,
- $k_{rGO} = k_{rGO}(S_G)$, $k_{rOG} = k_{rOG}(S_G)$ — для двухфазной системы газ-нефть.

Типичный вид этих функций приведен на рис. 1. Здесь обозначено `Swc`=S_{Wc} – остаточная водонасыщенность (значение насыщенности, меньше которого течения воды нет). Эти функции можно получить либо в лабораторных условиях, либо аппроксимировать посредством аналитических функций на основе следующих **задаваемых** данных:

- S_{Wc} – предельная водонасыщенность,
- S_{OrW} – остаточная нефтенасыщенность в системе вода–нефть,
- S_{Gr} – критическая газонасыщенность,
- S_{OrG} – остаточная нефтенасыщенность в системе газ–нефть,



($а$). $k_{rW} = k_{rW}(S_W)$ (красная сплошная линия), ($б$). $k_{rG} = k_{rG}(S_G)$ (красная сплошная линия), $k_{rOW} = k_{rOW}(S_W)$ (зеленая пунктирная линия) $k_{rOG} = k_{rOG}(S_G)$ (зеленая пунктирная линия)

Рис. 1. Пример относительных фазовых проницаемостей в двухфазных системах

- k_{rWrO} – относительная проницаемость по воде при остаточной нефти и $S_G = 0$,
- k_{rOcW} – относительная проницаемость по нефти при S_{Wc} и S_{Gr},
- k_{rGcW} – относительная проницаемость по газу при S_{Wc} и $S_o = 0$,
- N_W, n_{OW}, N_G, n_{OG} – постоянные параметры функции.

Тогда можно определить следующие аналитические аппроксимации для выше перечисленных функций:

$$k_{rWO}(S_W) = \begin{cases} 0 & \text{если } S_W \le S_{Wc} \\ k_{rWrO}\left(\dfrac{S_W - S_{Wc}}{1 - S_{Wc} - S_{OrW}}\right)^{N_W} & \text{иначе} \end{cases}$$

$$k_{rOW}(S_W) = \begin{cases} 0 & \text{если } 1 - S_W - S_{OrW} < 0 \\ k_{rOcW}\left(\dfrac{1 - S_W - S_{OrW}}{1 - S_{Wc} - S_{OrW}}\right)^{n_{OW}} & \text{иначе} \end{cases}$$

$$k_{rGO}(S_G) = \begin{cases} 0 & \text{если } S_G < S_{Gr} \\ k_{rGcW}\left(\dfrac{S_G - S_{Gr}}{1 - S_{Gr} - S_{wr}}\right)^{N_G} & \text{иначе} \end{cases}$$

$$k_{rOG}(S_G) = \begin{cases} 0 & \text{если } 1 - S_G - S_{Wc} - S_{OrG} < 0 \\ k_{rOcW}\left(\dfrac{1 - S_G - S_{Wc} - S_{OrG}}{1 - S_{Wc} - S_{OrG}}\right)^{n_{OG}} & \text{иначе} \end{cases}$$

Определим

$$k_{rW}(S_W) = k_{rWO}(S_W), \tag{16}$$

$$k_{rG}(S_G) = k_{rGO}(S_G). \tag{17}$$

Для расчета относительной проницаемости по нефти k_{rO} — обычно используется первая или вторая модель Стоуна.

Первая модель Стоуна

Определим следующие постоянные величины:

- S_{Wc} – насыщенность связанной воды, то есть минимальное допустимое значение S_W для двухфазной системы вода–нефть,
- $k_{rOcW} = k_{rOW}(S_{Wc})$ – относительная проницаемость по нефти при связанной воде,
- S_{Wr} – остаточная водонасыщенность, являющаяся самым большим значением S_W, где $k_{rW}(S_W) = k_{rWO}(S_W) = 0$,
- S_{OrW} – остаточная насыщенность нефти в системе нефть–вода, являющаяся самым большим значением S_W, где $k_{rOW}(S_W) = 0$,
- S_{OrG} – остаточная насыщенность нефти в системе нефть–газ, являющаяся самым большим значением S_G, где $k_{rOG}(S_G) = 0$.

Тогда положим

$$\begin{aligned}
\alpha(S_G) &= 1 - \frac{S_G}{1 - S_{Wc} - S_{OrG}}, \\
S_{om}(S_G) &= \alpha(S_G) S_{OrW} + (1 - \alpha(S_G)) S_{OrG}, \\
S_O^*(S_O, S_G) &= \begin{cases} \dfrac{S_O - S_{om}(S_G)}{1 - S_{Wc} - S_{om}(S_G)}, & \text{если } S_O \geq S_{om}(S_G) \\ 0, & \text{иначе} \end{cases}, \\
S_W^*(S_W, S_G) &= \begin{cases} \dfrac{S_W - S_{Wc}}{1 - S_{Wc} - S_{om}(S_G)}, & \text{если } S_W \geq S_{Wc} \\ 0, & \text{иначе} \end{cases}, \\
S_G^*(S_G) &= \frac{S_G}{1 - S_{Wc} - S_{om}(S_G)}, \qquad (18)
\end{aligned}$$

и

$$k_{rO} = k_{rO}(S_O, S_W, S_G) = \frac{S_O^*(S_O, S_G)}{k_{rOcW}} \frac{k_{rOW}(S_W)}{1 - S_W^*(S_W, S_G)} \frac{k_{rOG}(S_G)}{1 - S_G^*(S_G)}. \qquad (19)$$

Вторая модель Стоуна

Используя определенные в предыдущем параграфе параметры

- S_{Wc} – насыщенность связанной воды, то есть минимальное допустимое значение S_W для двухфазной системы вода–нефть,
- $k_{rOcW} = k_{rOW}(S_{Wc})$ – относительная проницаемость по нефти при связанной воде,

получаем, что в соответствии со второй моделью Стоуна относительная проницаемость по нефти равна:

$$\begin{aligned}
k_{rO}(S_O, S_W, S_G) = k_{rOcW} \left(\frac{k_{rOW}(S_W)}{k_{rOcW}} + k_{rW}(S_W) \right) \left(\frac{k_{rOG}(S_G)}{k_{rOcW}} + k_{rG}(S_G) \right) \qquad (20) \\
- k_{rOcW}(k_{rW}(S_W) + k_{rG}(S_G))
\end{aligned}$$

Масштабирование фазовых проницаемостей по 2 точкам

Опция масштабирования конечных точек фазовых проницаемостей используется при адаптации модели. Концевые точки могут задаваться в каждой точке пласта или как функции глубины.

При включении этой опции насыщенности и фазовые проницаемости пересчитываются в соответствии со следующими формулами.

Масштабирование насыщенностей

Введем следующие обозначения

- $S_W,\ S_G$ – насыщенности воды и газа в блоке,
- $S_{Wcr},\ S_{Gcr}$ – критические водо- и газонасыщенности, т.е. максимальное значение насыщенности воды (газа) в **заданной** таблице ОФП, для которого $k_{rW} = 0$ ($k_{rG} = 0$),
- $S_{Wmax},\ S_{Gmax}$ – максимальные значения водонасыщенности (и газонасыщенности) в **заданной** таблице ОФП,
- $SWCR$, $SGCR$ – **заданные** значения критической водонасыщенности (газонасыщенности) в каждой точке пласта,
- SWU, SGU – **заданные** значения максимальной водонасыщенности (газонасыщенности) в каждой точке пласта,
- $\tilde{S}_W,\ \tilde{S}_G$ – отнормированные водонасыщенность и газонасыщенность,
- $k_{rWO},\ k_{rGO}$ – относительные проницаемости по воде и газу в **заданной** таблице ОФП,
- $k_{rW\ max}(table),\ k_{rG\ max}(table)$ – максимальные значения проницаемости по воде (газу) в **заданной** таблице ОФП.

$$\tilde{S}_W = S_{Wcr} + \frac{(S_W - SWCR)(S_{Wmax} - S_{Wcr})}{SWU - SWCR} \tag{21}$$
$$\tilde{S}_G = S_{Gcr} + \frac{(S_G - SGCR)(S_{Gmax} - S_{Gcr})}{SGU - SGCR}$$

Соответствующие относительные фазовые проницаемости вычисляются по формулам:

$$k_{rW}(S_W) = \begin{cases} 0 & S_W \le SWCR \\ k_{rWO}(\tilde{S}_W) & SWCR < S_W < SWU \\ k_{rW\ max}(table) & S_W \ge SWU \end{cases} \tag{22}$$

$$k_{rG}(S_G) = \begin{cases} 0 & S_G \le SGCR \\ k_{rGO}(\tilde{S}_G) & SGCR < S_G < SGU \\ k_{rG\ max}(table) & S_G \ge SGU \end{cases} \tag{23}$$

Аналогично вычисляются фазовые проницаемости по нефти.

Масштабирование фазовых проницаемостей

В случае, если задан хотя бы один из массивов данных *KRO*, *KRW*, *KRG*, *KRORW*, *KRORG*, *KRWR*, *KRGR*, проводится также масштабирование полученных значений фазовых проницаемостей по следующей схеме. Здесь SR и SPCR определяются по аналогии с 3-точечным масштабированием насыщенностей.

Вода:

Если задан только массив KRW

$$k_{rW}^{scaled}(S_W) = k_{rW}(S_W) \cdot \frac{KRW}{k_{rW\ max}(table)} \tag{24}$$

Если заданы массивы KRW и KRWR

$$S_W <= SR \quad k_{rW}^{scaled}(S_W) = k_{rW}(S_W) \cdot \frac{KRWR}{k_{rW}(SR)} \tag{25}$$

$$S_W > SR \quad k_{rW}^{scaled}(S_W) = KRWR + \frac{KRW - KRWR}{k_{rW\ max}(table) - k_{rW}(SR)} \times \times (k_{rW}(S_W) - k_{rW}(SR))$$

Газ:

Если задан только массив KRG

$$k_{rG}^{scaled}(S_G) = k_{rG}(S_G) \cdot \frac{KRG}{k_{rG\ max}(table)} \tag{26}$$

Если заданы массивы KRG и KRGR

$$S_G <= SR \quad k_{rG}^{scaled}(S_G) = k_{rG}(S_G) \cdot \frac{KRGR}{k_{rG}(SR)} \tag{27}$$

$$S_G > SR \quad k_{rG}^{scaled}(S_G) = KRGR + \frac{KRG - KRGR}{k_{rG\ max}(table) - k_{rG}(SR)} \times \times (k_{rG}(S_G) - k_{rG}(SR))$$

Нефть: здесь “P” обозначает фазу газа или воды

Если задан только массив KRO

$$k_{rOP}^{scaled}(S_P) = k_{rOP}(S_P) \cdot \frac{KRO}{k_{rOP\ max}(table)} \tag{28}$$

Если заданы массивы KRO и KRORP

$$S_P <= SPCR \quad k_{rOP}^{scaled}(S_P) = k_{rOP}(S_P) \cdot \frac{KRORP}{k_{rOP}(SPCR)} \tag{29}$$

$$S_P > SPCR \quad k_{rOP}^{scaled}(S_P) = KRORP + + \frac{KRO - KRORP}{k_{rOP\ max}(table) - k_{rOP}(SPCR)} \times \times (k_{rOP}(S_P) - k_{rOP}(SPCR))$$

Масштабирование фазовых проницаемостей по 3 точкам

Если выбран метод масштабирования по трем точкам, то насыщенности и фазовые проницаемости перевычисляются следующим образом.

Масштабирование насыщенностей

Как и в случае масштабирования по 2 точкам, сперва введем обозначения.

- S_W, S_G насыщенности воды и газа в блоке,
- $S_{W\ co}$, $S_{G\ co}$ насыщенности связанной фазой, т.е. минимальные значения водонасыщенности (и газонасыщенности) в **заданной** таблице ОФП,
- S_{Wcr}, S_{Gcr} критические водо- и газонасыщенности, т.е. максимальное значение насыщенности воды (газа) в **заданной** таблице ОФП, для которого $k_{rW} = 0$ ($k_{rG} = 0$),
- S_{OWcr}, S_{OGcr} максимальное значение нефтенасыщенности в **заданной** таблице ОФП, для которого относительная фазовая проницаемость по нефти равна нулю: $k_{rOW} = 0$ ($k_{rOG} = 0$),
- S_{Wmax}, S_{Gmax} максимальные значения водонасыщенности (и газонасыщенности) в **заданной** таблице ОФП,
- *SWL*, *SGL* **заданные** значения минимальной водонасыщенности (газонасыщенности) в каждой точке пласта,
- *SWCR*, *SGCR* **заданные** значения критической водонасыщенности (газонасыщенности) в каждой точке пласта,
- *SOWCR*, *SOGCR* **заданные** значения критической нефтенасыщенности по воде (по газу) каждой точке пласта,

- SWU, SGU **заданные** значения максимальной водонасыщенности (газонасыщенности) в каждой точке пласта,
- $\tilde{S}_W$, $\tilde{S}_G$ отнормированные водонасыщенность и газонасыщенность.

1. Масштабирование проницаемости по воде

   Обозначим

   - для 3фазной системы $SR = 1 - SOWCR - SGL$, $S_r = 1 - S_{OWcr} - S_{G\ co}$
   - для системы нефть-вода $SR = 1 - SOWCR$, $S_r = 1 - S_{OWcr}$

   Тогда водонасыщенность масштабируется по формуле

   $$\tilde{S}_W = \begin{cases} S_{Wcr} + \frac{(S_W - SWCR)(S_r - S_{Wcr})}{SR - SWCR} & SWCR < S_W < SR \\ S_r + \frac{(S_W - SR)(S_{Wmax} - S_r)}{SWU - SR} & SR < S_W < SWU \end{cases} \tag{30}$$

   Относительная фазовая проницаемость рассчитывается, как в (22).

2. Масштабирование проницаемости по газу

   Обозначим

   - для 3фазной системы $SR = 1 - SOGCR - SWL$, $S_r = 1 - S_{OGcr} - S_{W\ co}$
   - для системы газ-вода $SR = 1 - SWCR$, $S_r = 1 - S_{Wcr}$

   Тогда газонасыщенность масштабируется по формуле

   $$\tilde{S}_G = \begin{cases} S_{Gcr} + \frac{(S_G - SGCR)(S_r - S_{Gcr})}{SR - SGCR} & SGCR < S_G < SR \\ S_r + \frac{(S_G - SR)(S_{Gmax} - S_r)}{SGU - SR} & SR < S_G < SGU \end{cases} \tag{31}$$

   Относительная фазовая проницаемость рассчитывается, как в (23).

Масштабирование фазовых проницаемостей

В случае, если задан хотя бы один из массивов *KRO*, *KRW*, *KRG*, *KRORW*, *KRORG*, *KRWR*, *KRGR*, проводится также масштабирование полученных значений фазовых проницаемостей по формулам, аналогичным 2-точечному случаю, см. (24) - (29). Единственное различие состоит в том, что SR и SPCR задаются естественным образом, см. масштабирование насыщенностей.

1.10. Капиллярное давление

Капиллярное давление системы нефть-газ

Капиллярное давление системы нефть-газ $P_{cOG} = P_{cOG}(S_G)$ — **заданная** функция. Задается в некотором количестве точек S_G и получается интерполяцией для других точек.

Капиллярное давление системы нефть-вода

Капиллярное давление системы нефть-вода $P_{cOW} = P_{cOW}(S_W)$ — **заданная** функция. Задается в некотором количестве точек S_W и получается интерполяцией для других точек.

Масштабирование капиллярного давления

Если требуется масштабирование конечных точек насыщенностей, то капиллярное давление пересчитывается в соответствии с заданными данными по двухточечному методу.

Сперва введем следующие обозначения

- S_W, S_G насыщенности воды и газа в блоке,
- $S_{W\ co}$, $S_{G\ co}$ насыщенности связанной фазой, т.е. минимальные значения водонасыщенности (и газонасыщенности) в **заданной** таблице ОФП,

- $S_{Wmax},\ S_{Gmax}$ максимальные значения водонасыщенности (и газонасыщенности) в **заданной** таблице ОФП,

- SWL, SGL – **заданные** значения минимальной водонасыщенности (газонасыщенности) в каждой точке пласта,

- SWU, SGU – **заданные** значения максимальной водонасыщенности (газонасыщенности) в каждой точке пласта,

- PCW, PCG – **заданные** массивы максимальных капиллярных давлений,

- $P_{cOW\ max}(table), P_{cOG\ max}(table)$ максимальные значения капиллярных давлений из **заданной** таблицы ОФП (значения при насыщенности связанной водой или газом),

- $\tilde{S}_W,\ \tilde{S}_G$ отнормированные водонасыщенность и газонасыщенность.

Насыщенности масштабируются по формулам:

$$\begin{aligned} \tilde{S}_W &= S_{W\ co} + \frac{(S_W - SWL)(S_{Wmax} - S_{W\ co})}{SWU - SWL} \\ \tilde{S}_G &= S_{G\ co} + \frac{(S_G - SGL)(S_{Gmax} - S_{G\ co})}{SGU - SGL} \end{aligned} \tag{32}$$

Капиллярные давление вычисляются следующим образом:

$$P_{cOW}(S_W) = P_{cOW}(\tilde{S}_W)(table) * \frac{PCW}{P_{cOW\ max}(table)} \tag{33}$$

$$P_{cOG}(S_G) = P_{cOG}(\tilde{S}_G)(table) * \frac{PCG}{P_{cOG\ max}(table)} \tag{34}$$

Расчет капиллярных давлений по J-функции Леверетта

Иногда требуется рассчитывать капиллярные давления с помощью модели J-функции Леверетта для некоторых выбранных фаз. Расчет идет по следу-

ющим формулам:

$$P_{cOW}(S_W) = J_W(S_W)(table) * J_{mult\ W} \tag{35}$$

$$P_{cOG}(S_G) = J_G(S_G)(table) * J_{mult\ G} \tag{36}$$

Здесь J_W, J_G вводятся в таблице ОФП, как функции от насыщенностей, вместо капиллярных давлений фаз. Множители рассчитываются как

$$J_{mult\ W} = ST_W * (\phi)^{\alpha}/(K)^{\beta} * 0.318316 \tag{37}$$

$$J_{mult\ G} = ST_G * (\phi)^{\alpha}/(K)^{\beta} * 0.318316$$

здесь

- ST_W – заданное поверхностное натяжение нефть-вода;
- ST_G – заданное поверхностное натяжение нефть-газ;
- ϕ – пористость в данной точке;
- K – проницаемость, рассчитанная по одному из следующих методов:
  - XY: $K = (k_{xx} + k_{yy})/2$
  - X: $K = k_{xx}$
  - Y: $K = k_{yy}$
  - Z: $K = k_{zz}$
- α и β – заданные параметры.

1.11. Газовый фактор

Газовый фактор $R_{G,O} = R_{G,O}(p_O)$ — **заданная** функция, определяющая долю газа, растворенную в нефти. Задается в некотором количестве точек p_O и получается интерполяцией для других точек.

1.12. Содержание нефти в газе

Содержание нефти в газе $R_{O,G} = R_{O,G}(p_G)$ — **заданная** функция, определяющая долю нефти, испарившейся в газ. Задается в некотором количестве точек p_G и получается интерполяцией для других точек.

1.13. Вязкость фазы

Вязкость фазы $\mu_P = \mu_P(p_P, \boldsymbol{N})$ — заданная функция. Для нефтяной и газовой фаз задается в некотором количестве точек p_P и получается интерполяцией для других точек. Для водной фазы обычно задается в одной точке с указанием значения производной по давлению.

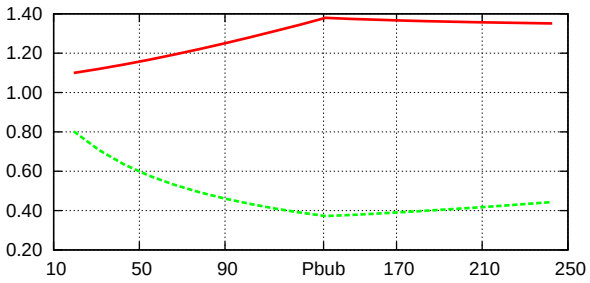
1.14. Коэффициент объемного расширения

Коэффициент объемного расширения $B_P = B_P(p_P, \boldsymbol{N})$ — **заданная** функция. Для нефтяной и газовой фаз задается в некотором количестве точек p_P и получается интерполяцией для других точек.

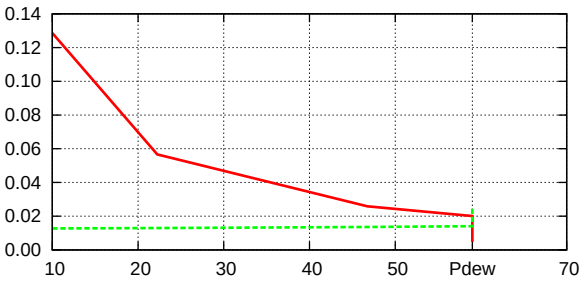
Для воды обычно задается коэффициент объемного расширения в одной точке (при опорном давлении) с указанием значения производной по давлению (сжимаемости). Сжимаемость определяется как:

$$c_W = -\frac{1}{B_W}\frac{\partial B_W}{\partial p} \tag{38}$$

Типичный вид зависимости коэффициента объемного расширения (красная сплошная линия) и вязкости (зеленая пунктирная линия) от давления для нефтяной (слева) и газовой (справа) фаз приведены на рис. 2. Здесь обозначено: `Pbub` – давление точки кипения $P_b = P_b(p, \boldsymbol{N})$, `Pdew` – давление точки росы $P_d = P_d(p, \boldsymbol{N})$. Левее точки P_b (P_d) значения B_O, μ_O (B_G, μ_G) зависят только от давления фазы, правее – зависят также от состава фаз, т.е. от $R_{G,O}$ ($R_{O,G}$).



(*а*). Зависимость для нефти



(*б*). Зависимость для газа

Рис. 2. Пример зависимости объемного коэффициента (красная сплошная линия) и вязкости (зеленая пунктирная линия) от давления

1.15. Молярная плотность фазы

Молярная плотность фазы зависит от коэффициента объемного расширения B_P. Для воды она рассчитывается как

$$\xi_W = \frac{\xi_{W,SC}}{B_W} \tag{39}$$

Здесь $B_W = B_W(p_W)$ – коэффициент объемного расширения для воды, см. 1.14.

Для нефтяной и газовой фаз молярная плотность рассчитывается как

$$\xi_O = \frac{R_{G,O}\xi_{G,SC} + \xi_{O,SC}}{B_O}, \quad \xi_G = \frac{R_{O,G}\xi_{O,SC} + \xi_{G,SC}}{B_G} \tag{40}$$

Здесь $\xi_{P,SC}$ – молярная плотность фазы P в стандартных условиях.

1.16. Массовая плотность фазы

Массовая плотность фазы ρ_P — **заданная** функция. Обычно вводятся следующие величины:

- $\rho_{O,SC}$ – плотность нефти в поверхностных условиях,
- $\rho_{G,SC}$ – плотность газа в поверхностных условиях,
- $\rho_{W,SC}$ – плотность воды в поверхностных условиях,

Тогда в качестве $\rho_P(p_P)$ используются следующие функции:

$$\rho_W = \frac{\rho_{W,SC}}{B_W} \tag{41}$$

$$\rho_O = \frac{R_{G,O}\rho_{G,SC} + \rho_{O,SC}}{B_O} \tag{42}$$

$$\rho_G = \frac{R_{O,G}\rho_{O,SC} + \rho_{G,SC}}{B_G} \tag{43}$$

1.17. Матрица концентраций компонент в фазах

Матрица $x_{c,P}$, $c = 1,\ldots,n_c$, $P = W,O,G$ задает распределение компонент по фазам. Вычисляется по следующим формулам:

$$\begin{array}{lll} x_{w,W} = 1, & x_{o,W} = 0, & x_{g,W} = 0 \\ x_{w,O} = 0, & x_{o,O} = \dfrac{\xi_{O,SC}}{R_s\xi_{G,SC} + \xi_{O,SC}}, & x_{g,O} = \dfrac{R_{G,O}\xi_{G,SC}}{R_{G,O}\xi_{G,SC} + \xi_{O,SC}} \\ x_{w,G} = 0, & x_{o,G} = \dfrac{R_{O,G}\xi_{O,SC}}{R_{O,G}\xi_{O,SC} + \xi_{G,SC}}, & x_{g,G} = \dfrac{\xi_{G,SC}}{R_{O,G}\xi_{O,SC} + \xi_{G,SC}} \end{array}$$

1.18. Скважина

Скважина аппроксимируется разными способами в зависимости от расчетной сетки. Рассмотрим источник фазы $Q_P = Q_P(p_P, \boldsymbol{N}, t)$ в блоке l в случае равномерной сетки и конечно-разностной аппроксимации. Мы определяем Q_P на поверхности цилиндра радиуса r_w с перфорированной областью скважины, взятой за ось цилиндра, как

$$Q_P(p_P, \boldsymbol{N}, t) = \beta_c T(t) M_p(p_P, S_W, S_G)(p_P - p_{BH}(t) - \bar{\rho}_{av}(p, \boldsymbol{N})g(D - D_{BH})) \tag{44}$$

где

- $M_p(p_P, S_W, S_G)$ – подвижность фазы, **известна**, будет определена ниже, см. раздел 2.4,
- $p_{BH}(t)$ – забойное давление, **известное** или **рассчитанное** из заданного дебита скважины $q(t)$,

- $\bar{\rho}_{av}(p, \boldsymbol{N})$ – средняя плотность флюидов в стволе скважины, зависит от аппроксимации, выбранной для уравнений (4)–(7), будет определена ниже, см. 2.4 (**известна**)
- D, $g = \text{const}$ были определены выше,
- D_{BH} – глубина забоя (**известна**),
- $T(t)$ – индекс продуктивности, или проводимость (**известен**), может быть задан или рассчитывается по формуле $T = \dfrac{2\pi K_{mult}(t)\beta_c Kh}{(\log(r_0/r_w)+s)}$, см. раздел 2.4. Здесь
  - $K_{mult}(t)$ – множитель индекса продуктивности (**известен**)
  - $\beta_c = \text{const}$ – множитель перевода систем единиц;
  - Kh – гидропроводность (**известна**), может быть задана или рассчитывается как произведение $h = \text{const}$ – высоты перфорированного интервала (**известна**), и K, проницаемости в плоскости, перпендикулярной оси цилиндра, ее аппроксимация зависит от дискретной аппроксимации, выбранной для уравнений (4)–(7), и будет описана ниже, см. раздел 2.4 (**известна**)
  - r_0 эквивалентный радиус, может либо задан либо вычислен способом, зависящим от дискретной аппроксимации, выбранной для уравнений (4)–(7), и будет описан ниже, см. раздел 2.4
  - $r_w = d_w/2 = \text{const}$ – радиус скважины (**известен**)
  - $s = s(x, y, z, t)$ – скин-фактор (**известен**).

В этом случае источник компонента c вычисляется как

$$q_c = \sum_P x_{c,P}\xi_P Q_P(p, \boldsymbol{N}) \tag{45}$$

где

- $x_{c,P} = x_{c,P}(p, \boldsymbol{N})$ – молярная доля компонента c в фазе P,
- $\xi_P = \xi_P(p, \boldsymbol{N})$ – молярная плотность фазы, см. 1.15,
- $Q_P(p, \boldsymbol{N})$ – дебит фазы в пластовых условиях, определяется уравнением (44).

1.19. Водонапорные горизонты

Водонапорные горизонты обычно моделируются (Fetkovitch) как резервуар с водой, имеющий в начальный момент времени объем V_0^{aq}, давление p_0^{aq}, сжимаемость c_t^{aq} и находящийся на глубине d^{aq}. Этот резервуар соединен с пластом в точках A_l, $l = 1, \ldots, L^{aq}$. Дебит из (в) этот резервуар в момент времени t и за промежуток времени τ определяется, таким образом, как

$$Q^{aq}(t, \tau) = \sum_{l=1}^{L^{aq}} \delta_l Q^l(t, \tau)$$

где δ_l – δ-функция Дирака в точке A_l,

$$Q^l(t, \tau) = J \alpha^l \psi_z^l (p^{aq}(t) - p^l - P_{cOW}^l + \rho_w^{aq}(t) g (d^l - d^{aq})) \left(\frac{1 - \exp(-\tau / T_c)}{\tau / T_c} \right)$$

где

- J – индекс эффективности (**задается**);
- α^l – доля дебита в точке l, $\sum_{l=1}^{L^{aq}} \alpha^l = 1$ (**задается**);
- ψ_z^l – коэффициент песчанистости;
- $p^{aq}(t)$ – давление в водонапорном горизонте в момент времени t;
- $\rho_w^{aq}(t)$ – плотность воды в водонапорном горизонте в момент времени t;
- $T_c = c_t^{aq} V_0^{aq} / J$.

Давление $p^{aq}(t)$ вычисляется из соотношения

$$p^{aq}(t) = p_0^{aq} - \frac{W_{aq\ total}(t)}{c_t^{aq} V_0^{aq}}, \quad W_{aq\ total}(t) = \int_0^t \sum_{l=1}^{L^{aq}} Q^l \, dt. \tag{46}$$

Другие модели водонапорных горизонтов (Carter-Tracy) также приводят к похожим соотношениям.

1.20. Пример постановки задачи

Рассмотрим стандартную двухфазную двухкомпонентную изотермическую модель нелетучей нефти при отсутствии капиллярных сил.

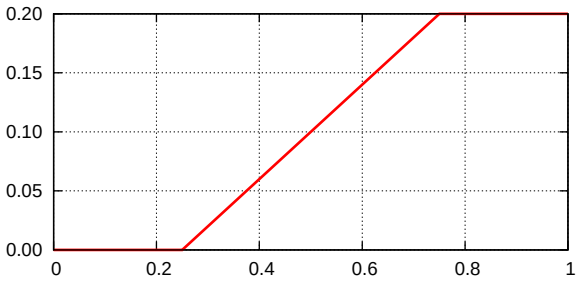
$$\begin{aligned}
\frac{\partial}{\partial t}(\phi N_w) &= \operatorname{div} \frac{\xi_{W,SC}}{B_W}\Big(\boldsymbol{k}\frac{k_{rW}}{\mu_W}\big(\nabla p - \frac{\rho_{W,SC}}{B_W} g \nabla D\big)\Big) + Q_w, \\
\frac{\partial}{\partial t}(\phi N_o) &= \operatorname{div} \frac{\xi_{O,SC}}{B_O}\Big(\boldsymbol{k}\frac{k_{ro}}{\mu_O}\big(\nabla p - \frac{\rho_{O,SC}}{B_O} g \nabla D\big)\Big) + Q_o, \\
S_W + S_O &= 1, \quad S_W = B_W N_w, \quad S_O = B_O N_o \\
\xi_{W,SC} &= 1000, \quad \xi_{O,SC} = 800, \quad \rho_{W,SC} = 1, \quad \rho_{O,SC} = 0.8 \\
\phi(p) &= 0.1(1 + 5 * 10^{-5}(p - 250) + (5 * 10^{-5})^2 (p - 250)^2/2) \\
\boldsymbol{k} &= \operatorname{diag}(1, 1, 0.1) \\
\mu_W &= 0.43, \quad \mu_O = 2.93, \\
B_W(p) &= 1.02 * e^{-5*10^{-5}(p-250)}, \quad B_O(p) = 1.15 * e^{-2*10^{-4}(p-250)}
\end{aligned}$$

Относительные фазовые проницаемости приведены на рис. 3.

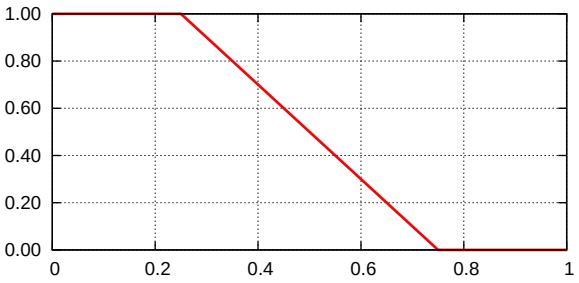
1.21. Особенности системы уравнений

Формально в системе уравнений (4) отсутствуют ненулевые производные по времени от давления. Действительно,

$$\frac{\partial}{\partial t}(\phi N_c) = \frac{\partial \phi}{\partial p}\frac{\partial p}{\partial t} N_c + \phi \frac{\partial N_c}{\partial t}$$



(*а*). $k_{rW} = k_{rW}(S_W)$



(*б*). $k_{rO} = k_{rO}(S_O)$

Рис. 3. Пример относительных фазовых проницаемостей в двухфазном случае

и здесь может быть $N_c = 0$. Однако, если сложить все уравнения и воспользоваться $\sum\limits_{c=1}^{n_c} x_{c,P} = 1$, то получим **уравнение для давления**

$$\frac{\partial}{\partial t}\left(\phi \sum_{c=1}^{n_c} N_c\right) = \operatorname{div} \sum_{P=1}^{n_P} \xi_P \boldsymbol{k} \frac{k_{rP}}{\mu_P}\left(\nabla p + \nabla P_{cP} - \rho_P g \nabla d\right) + \sum_{c=1}^{n_c} q_c \qquad (47)$$

Отметим еще

- вырождение каждого из уравнений (4) в своей части расчетной области из-за того, что k_{rP} могут обращаться в 0 (но не одновременно);
- существенную нелинейность уравнений;
- присутствие источников и стоков типа δ-функции Дирака;
- возникновение внутренних погранслоев.

2. Дискретизация задачи

Рассмотрим один из распространенных способов дискретизации системы уравнений (4).

2.1. Аппроксимация по времени

Для расчетов возникающих на практике задач обычно используют полностью неявную схему:

$$\begin{aligned}\frac{\widehat{\phi}\widehat{N}_c - \phi N_c}{\Delta t} &= \operatorname{div} \sum_{P=1}^{n_P} \widehat{x}_{c,P}\widehat{\xi}_P \boldsymbol{k} \frac{\widehat{k}_{rP}}{\widehat{\mu}_P}\Big(\nabla\widehat{p} + \nabla\widehat{P}_{cP} - \widehat{\gamma}_P \nabla d\Big) \\ &\quad + \widehat{q}_c, \quad c = 1,\ldots,n_c \end{aligned} \tag{48}$$

где знак $\widehat{\ }$ означает, что в качестве давления и молярных плотностей использованы значения на новом шаге по времени. Использование полностью неявной схемы диктуется высокой жесткостью задачи: при малом локальном изменении свойств среды вблизи скважины (слагаемое q_c) может значительно меняться ее режим, что уже приводит большим нелокальным изменениям в результатах расчета.

2.2. Аппроксимация по пространству

Для выполнения законов сохранения все методы аппроксимации должны **точно** приближать оператор div. Для этого уравнения (48) интегрируются по каждому блоку сетки V_i и используется формула Гаусса-Остроградского:

$$\begin{aligned}\int\limits_{V_i} \frac{\widehat{\phi}\widehat{N}_c - \phi N_c}{\Delta t}\, dV &= \sum_{P=1}^{n_P} \int\limits_{\partial V_i} \left(\widehat{x}_{c,P}\widehat{\xi}_P \boldsymbol{k} \frac{\widehat{k}_{rP}}{\widehat{\mu}_P}\Big(\nabla\widehat{p} + \nabla\widehat{P}_{cP} - \widehat{\gamma}_P \nabla d)\Big), \boldsymbol{n} \right) ds \\ &\quad + \int\limits_{V_i} \widehat{q}_c\, dV, \quad c = 1,\ldots,n_c \end{aligned} \tag{49}$$

В **методе конечных объемов** используется константная аппроксимация для функций в блоке и разностные приближения для вычисления оператора ∇:

$$\begin{aligned}\frac{\widehat{\phi}\widehat{N}_c - \phi N_c}{\Delta t}|V_i| &= \sum_{P=1}^{n_P} \sum_{j:V_i\cap V_j\neq\emptyset} |V_i \cap V_j|\, \widehat{x}_{c,P}^{(i,j)} \widehat{\xi}_P^{(i,j)} \boldsymbol{k}|_{V_i\cap V_j} \frac{\widehat{k}_{rP}^{(i,j)}}{\widehat{\mu}_P^{(i,j)}} \widehat{\Phi}_{i,j} \\ &\quad + \widehat{q}_c, \quad c = 1,\ldots,n_c \end{aligned} \tag{50}$$

где поток

$$\widehat{\Phi}_{i,j} = \pm \frac{(\widehat{p}^i_P + \nabla \widehat{P}^i_{cP}) - (\widehat{p}^j_P + \nabla \widehat{P}^j_{cP}) - \frac{g}{2}(\widehat{\rho}^i_P + \widehat{\rho}^j_P)(d^i - d^j)}{\mathrm{dist}(V_i, V_j)} \tag{51}$$

а значения $\cdot^{(i,j)}$ есть значения в точке i или j в зависимости от знака $\Phi_{i,j}$ (аппроксимация по потоку).

2.3. Пример построения аппроксимации по пространству

Рассмотрим вид аппроксимации на квазиортогональной осям координат блочно-центрированной сетке в случае диагонального тензора абсолютной проницаемости (см. 1.7). На этом примере уже видны все особенности рассматриваемой задачи и он не перегружен излишними деталями, возникающими при рассмотрении произвольной нестыкующейся сетки и полного тензора проницаемости.

Обозначим через $(\bar{x}^i, \bar{y}^j, \bar{z}^k)$, $i = 0, 1, \ldots, N^x$, $j = 0, 1, \ldots, N^y$, $k = 0, 1, \ldots, N^z$ вершины квазиортогональной сетки в расчетной области; через (x^i, y^j, z^k), $i = 1, \ldots, N^x$, $j = 1, \ldots, N^y$, $k = 1, \ldots, N^z$ — центры блоков $[\bar{x}^{i-1}, \bar{x}^i] \times [\bar{y}^{j-1}, \bar{y}^j] \times [\bar{z}^{k-1}, \bar{z}^k]$.

Обозначим для $i = 1, \ldots, N^x$, $j = 1, \ldots, N^y$, $k = 1, \ldots, N^z$, числа α и функции F на сетке:

- $V^{i,j,k} = [\bar{x}^{i-1}, \bar{x}^i] \times [\bar{y}^{j-1}, \bar{y}^j] \times [\bar{z}^{k-1}, \bar{z}^k]$ – блок сетки,
- $v^{i,j,k}$ объем блока $V^{i,j,k}$
- $F^{i,j,k} = F(x^i, y^j, z^k)$ значение функции F в точке (x^i, y^j, z^k).

Для аппроксимации сомножителя

$$T = \frac{|V_i \cap V_j| \, \boldsymbol{k}|_{V_i \cap V_j}}{\mathrm{dist}(V_i, V_j)}$$

(называемом обычно **проводимостью** между блоками V_i и V_j) в (50) и (51) вводят величины

- $T_x^{i+,j,k}(\boldsymbol{k}) = \int_{\bar{x}^i \times [\bar{y}^{j-1}, \bar{y}^j] \times [\bar{z}^{k-1}, \bar{z}^k]} k_{xx}\, ds / (x^{i+1} - x^i)$
- $T_x^{i-,j,k}(\boldsymbol{k}) = \int_{\bar{x}^{i-1} \times [\bar{y}^{j-1}, \bar{y}^j] \times [\bar{z}^{k-1}, \bar{z}^k]} k_{xx}\, ds / (x^i - x^{i-1})$
- $T_y^{i,j+,k}(\boldsymbol{k}) = \int_{[\bar{x}^{i-1}, \bar{x}^i] \times \bar{y}^j \times [\bar{z}^{k-1}, \bar{z}^k]} k_{yy}\, ds / (y^{j+1} - y^j)$
- $T_y^{i,j-,k}(\boldsymbol{k}) = \int_{[\bar{x}^{i-1}, \bar{x}^i] \times \bar{y}^{j-1} \times [\bar{z}^{k-1}, \bar{z}^k]} k_{yy}\, ds / (y^j - y^{j-1})$
- $T_z^{i,j,k+}(\boldsymbol{k}) = \int_{[\bar{x}^{i-1}, \bar{x}^i] \times [\bar{y}^{j-1}, \bar{y}^j] \times \bar{z}^k} k_{zz}\, ds / (z^{k+1} - z^k)$
- $T_z^{i,j,k-}(\boldsymbol{k}) = \int_{[\bar{x}^{i-1}, \bar{x}^i] \times [\bar{y}^{j-1}, \bar{y}^j] \times \bar{z}^{k-1}} k_{zz}\, ds / (z^k - z^{k-1})$

Значения элементов тензора проницаемости на гранях блоков сетки вычисляют как среднее гармоническое между значениями в блоках. Именно, обозначим через $\perp$ направление, перпендикулярное к грани, по которой здесь ведется интегрирование, а через 1 и 2 – два оставшихся направления. Будем обозначать блок сетки индексом ‘+’, если его $\perp$-координата больше, а индексом ‘-’, если его $\perp$-координата соответственно меньше. Обозначим также размеры поперечников блоков ‘+’ и ‘-’ через $\Delta_\perp^+$, Δ_1^+, Δ_2^+ и $\Delta_\perp^-$, Δ_1^-, Δ_2^-.

Введем вножитель, учитывающий коэффициент песчанистости (см. 1.8):

$$n_i^\pm = \begin{cases} 1, & \text{если } i \text{ соответствует направлению } x \text{ или } y \\ \psi^\pm, & \text{если } i \text{ соответствует направлению } z \end{cases}$$

и соответствующую взвешенную проницаемость в направлении $\perp$:

$$\varkappa^\pm = k_{\perp\perp}^\pm (n_1^\pm \Delta_1^\pm)(n_2^\pm \Delta_2^\pm)$$

Тогда общий вид для аппроксимации проводимостей $T_x^{i+,j,k}$, $T_x^{i-,j,k}$, $T_y^{i,j+,k}$, $T_y^{i,j-,k}$, $T_z^{i,j,k+}$, $T_z^{i,j,k-}$:

$$T = \beta_c T_{mult}^- \frac{2\varkappa^+ \varkappa^-}{\varkappa^+ \Delta_\perp^- + \varkappa^- \Delta_\perp^+} \tag{52}$$

Для аппроксимации гравитационного слагаемого $\gamma_P = \rho_P g$ на грани используют арифметическое осреднение:

$$\begin{aligned}
\gamma_P^{i+,j,k}(p,\boldsymbol{N}) &= \gamma_P(p,\boldsymbol{N})|_{\bar{x}^i\times[\bar{y}^{j-1},\bar{y}^j]\times[\bar{z}^{k-1},\bar{z}^k]} &= g(\rho_P^{i,j,k}+\rho_P^{i+1,j,k})/2\\
\gamma_P^{i-,j,k}(p,\boldsymbol{N}) &= \gamma_P(p,\boldsymbol{N})|_{\bar{x}^{i-1}\times[\bar{y}^{j-1},\bar{y}^j]\times[\bar{z}^{k-1},\bar{z}^k]} &= g(\rho_P^{i,j,k}+\rho_P^{i-1,j,k})/2\\
\gamma_P^{i,j+,k}(p,\boldsymbol{N}) &= \gamma_P(p,\boldsymbol{N})|_{[\bar{x}^{i-1},\bar{x}^i]\times\bar{y}^j\times[\bar{z}^{k-1},\bar{z}^k]} &= g(\rho_P^{i,j,k}+\rho_P^{i,j+1,k})/2\\
\gamma_P^{i,j-,k}(p,\boldsymbol{N}) &= \gamma_P(p,\boldsymbol{N})|_{[\bar{x}^{i-1},\bar{x}^i]\times\bar{y}^{j-1}\times[\bar{z}^{k-1},\bar{z}^k]} &= g(\rho_P^{i,j,k}+\rho_P^{i,j-1,k})/2\\
\gamma_P^{i,j,k+}(p,\boldsymbol{N}) &= \gamma_P(p,\boldsymbol{N})|_{[\bar{x}^{i-1},\bar{x}^i]\times[\bar{y}^{j-1},\bar{y}^j]\times\bar{z}^k} &= g(\rho_P^{i,j,k}+\rho_P^{i,j,k+1})/2\\
\gamma_P^{i,j,k-}(p,\boldsymbol{N}) &= \gamma_P(p,\boldsymbol{N})|_{[\bar{x}^{i-1},\bar{x}^i]\times[\bar{y}^{j-1},\bar{y}^j]\times\bar{z}^{k-1}} &= g(\rho_P^{i,j,k}+\rho_P^{i,j,k-1})/2
\end{aligned}$$

Аппроксимация числителя в потоке через грань (51):

$$\begin{aligned}
\Phi_P^{i+,j,k}(p,\boldsymbol{N}) =& p^{i+1,j,k}-p^{i,j,k}+P_{cOP}^{i+1,j,k}(p,\boldsymbol{N})-P_{cOP}^{i,j,k}(p,\boldsymbol{N})-\\
&-\gamma_P^{i+,j,k}(p,\boldsymbol{N})(d^{i+1,j,k}-d^{i,j,k})\\
\Phi_P^{i-,j,k}(p,\boldsymbol{N}) =& p^{i,j,k}-p^{i-1,j,k}+P_{cOP}^{i,j,k}(p,\boldsymbol{N})-P_{cOP}^{i-1,j,k}(p,\boldsymbol{N})-\\
&-\gamma_P^{i-,j,k}(p,\boldsymbol{N})(d^{i,j,k}-d^{i-1,j,k})\\
\Phi_P^{i,j+,k}(p,\boldsymbol{N}) =& p^{i,j+1,k}-p^{i,j,k}+P_{cOP}^{i,j+1,k}(p,\boldsymbol{N})-P_{cOP}^{i,j,k}(p,\boldsymbol{N})-\\
&-\gamma_P^{i,j+,k}(p,\boldsymbol{N})(d^{i,j+1,k}-d^{i,j,k})\\
\Phi_P^{i,j-,k}(p,\boldsymbol{N}) =& p^{i,j,k}-p^{i,j-1,k}+P_{cOP}^{i,j,k}(p,\boldsymbol{N})-P_{cOP}^{i,j-1,k}(p,\boldsymbol{N})-\\
&-\gamma_P^{i,j-,k}(p,\boldsymbol{N})(d^{i,j,k}-d^{i,j-1,k})\\
\Phi_P^{i,j,k+}(p,\boldsymbol{N}) =& p^{i,j,k+1}-p^{i,j,k}+P_{cOP}^{i,j,k+1}(p,\boldsymbol{N})-P_{cOP}^{i,j,k}(p,\boldsymbol{N})-\\
&-\gamma_P^{i,j,k+}(p,\boldsymbol{N})(d^{i,j,k+1}-d^{i,j,k})\\
\Phi_P^{i,j,k-}(p,\boldsymbol{N}) =& p^{i,j,k}-p^{i,j,k-1}+P_{cOP}^{i,j,k}(p,\boldsymbol{N})-P_{cOP}^{i,j,k-1}(p,\boldsymbol{N})-\\
&-\gamma_P^{i,j,k-}(p,\boldsymbol{N})(d^{i,j,k}-d^{i,j,k-1})
\end{aligned} \tag{53}$$

Для задания апроксимации по потоку (когда свойства фазы берутся из того блока сетки, откуда она вытекает), обозначим

- $\widehat{U}^{i+,j,k}(F,\alpha) = \begin{cases} F^{i,j,k} & \text{если } i = N^x \text{ или } \alpha \le 0 \\ F^{i+1,j,k} & \text{если } i < N^x \text{ и } \alpha > 0 \end{cases},$
- $\widehat{U}^{i-,j,k}(F,\alpha) = \begin{cases} F^{i-1,j,k} & \text{если } i > 1 \text{ и } \alpha \le 0 \\ F^{i,j,k} & \text{если } i = 1 \text{ или } \alpha > 0 \end{cases},$
- $\widehat{U}^{i,j+,k}(F,\alpha) = \begin{cases} F^{i,j,k} & \text{если } j = N^y \text{ или } \alpha \le 0 \\ F^{i,j+1,k} & \text{если } j < N^y \text{ и } \alpha > 0 \end{cases},$
- $\widehat{U}^{i,j-,k}(F,\alpha) = \begin{cases} F^{i,j-1,k} & \text{если } j > 1 \text{ и } \alpha \le 0 \\ F^{i,j,k} & \text{если } j = 1 \text{ или } \alpha > 0 \end{cases},$
- $\widehat{U}^{i,j,k+}(F,\alpha) = \begin{cases} F^{i,j,k} & \text{если } k = N^z \text{ или } \alpha \le 0 \\ F^{i,j,k+1} & \text{если } k < N^z \text{ и } \alpha > 0 \end{cases},$
- $\widehat{U}^{i,j,k-}(F,\alpha) = \begin{cases} F^{i,j,k-1} & \text{если } k > 1 \text{ и } \alpha \le 0 \\ F^{i,j,k} & \text{если } k = 1 \text{ или } \alpha > 0 \end{cases}$
- $U^{i+,j,k}(F,\alpha) = \alpha\widehat{U}^{i+,j,k}(F,\alpha)$
- $U^{i-,j,k}(F,\alpha) = \alpha\widehat{U}^{i-,j,k}(F,\alpha)$
- $U^{i,j+,k}(F,\alpha) = \alpha\widehat{U}^{i,j+,k}(F,\alpha)$
- $U^{i,j-,k}(F,\alpha) = \alpha\widehat{U}^{i,j-,k}(F,\alpha)$
- $U^{i,j,k+}(F,\alpha) = \alpha\widehat{U}^{i,j,k+}(F,\alpha)$
- $U^{i,j,k-}(F,\alpha) = \alpha\widehat{U}^{i,j,k-}(F,\alpha)$

Тогда уравнение (50) принимает вид:

$$
\begin{aligned}
F_c \equiv v^{i,j,k}\phi^{i,j,k}(p)N_c - \tau q_c^{i,j,k,\beta}(p,\boldsymbol{N},p_{ref}) \\
- \tau\sum_P \Big(T_x^{i+,j,k}(\boldsymbol{k})U^{i+,j,k}\left(\Lambda_{c,P}(p,\boldsymbol{N}),\Phi_P^{i+,j,k}(p,\boldsymbol{N})\right) \\
- T_x^{i-,j,k}(\boldsymbol{k})U^{i-,j,k}\left(\Lambda_{c,P}(p,\boldsymbol{N}),\Phi_P^{i-,j,k}(p,\boldsymbol{N})\right) \\
+ T_y^{i,j+,k}(\boldsymbol{k})U^{i,j+,k}\left(\Lambda_{c,P}(p,\boldsymbol{N}),\Phi_P^{i,j+,k}(p,\boldsymbol{N})\right) \\
- T_y^{i,j-,k}(\boldsymbol{k})U^{i,j-,k}\left(\Lambda_{c,P}(p,\boldsymbol{N}),\Phi_P^{i,j-,k}(p,\boldsymbol{N})\right) \\
+ T_z^{i,j,k+}(\boldsymbol{k})U^{i,j,k+}\left(\Lambda_{c,P}(p,\boldsymbol{N}),\Phi_P^{i,j,k+}(p,\boldsymbol{N})\right) \\
- T_z^{i,j,k-}(\boldsymbol{k})U^{i,j,k-}\left(\Lambda_{c,P}(p,\boldsymbol{N}),\Phi_P^{i,j,k-}(p,\boldsymbol{N})\right)\Big) \\
- v^{i,j,k}f_c^{i,j,k} = 0, \quad c = 1,\ldots,n_c \quad (54)
\end{aligned}
$$

где $f_c = \bar{\phi}\bar{N}_c$ – значение с предыдущего шага по времени,

$$\Lambda_{c,P}(p,\boldsymbol{N}) = x_{c,P}(p,\boldsymbol{N})k_{rP}(p,\boldsymbol{N})\xi_P(p,\boldsymbol{N})/\mu_P(p,\boldsymbol{N}).$$

2.4. Аппроксимация скважины

После дискретизации уравнений необходимо для каждого блока сетки, в котором расположена скважина, установить взаимосвязь между притоком каждой из фаз, давлением в скважине и давлением в блоке. Данная взаимосвязь между забойным давлением в скважине и ее общим дебитом используется для расчета забойного давления в случае, если для скважины известен ее дебит (т.н. “контроль по дебиту”), и для расчета дебитов в случае, если для скважины известно ее забойное давление (т.н. “контроль по забойному давлению”).

Расчет притока в скважину

При расчете притока в скважину используются следующие предположения:

- Предполагается, что скважина проходит через весь блок сетки, через его центр, ось скважины квазиортогональна двум его противоположным граням.

- На каждом расчетном временном шаге предполагается, что плотность смеси в стволе скважины постоянна (т.е. не зависит от глубины).

- Трением в стволе скважины мы пренебрегаем.

- При расчете притока в скважину мы пренебрегаем различием давлений фаз; в качестве расчетного давления используется давление в нефтяной фазе.

После дискретизации уравнений (44) мы получаем следующую формулу, задающую объемный приток в скважину из блока l с координатами (i, j, k), для фаз $P = \{W, O, G\}$, то есть вода, нефть, газ:

$$Q_P^l(p^l, \boldsymbol{N}^l, t) = T^l(t) \cdot M_P(p^l, S_W^l, S_G^l)(p^l - p_{con}^l(t)) \tag{55}$$

где

- $Q_P^l = Q_P^l(p^l, \boldsymbol{N}^l, t)$ – объемный приток фазы P через перфорацию l в пластовых условиях;

- $T^l(t)$ – проводимость перфорации (индекс продуктивности), будет описана ниже, см. раздел 2.4;

- $M_P = M_P(p^l, S_W^l, S_G^l)$ – общая подвижность фазы в блоке l, будет описана ниже, см. раздел 2.4;

- p^l, S_W^l, S_G^l – давление и насыщенности в блоке l, в котором расположена перфорация;

- $\boldsymbol{N}_c$ – молярные плотности компонентов в блоке l;

- $p^l_{con}(t)$ – давление в скважине на уровне перфорации l см. раздел 2.4

Расчет проводимости

Проводимость перфорации $T^l(t)$ может быть задана явно (как входная информация), в противном случае она вычисляется по формуле

$$T^l(t) = \frac{2\pi K^l_{mult}(t)\beta_c(Kh)^l}{(\log(r^l_o/r^l_w) + s^l)}. \tag{56}$$

Здесь

- $K^l_{mult}(t)$ – множитель эффективности для перфорации l (задан);
- β_c – множитель перевода систем единиц;
- $(Kh)^l$ – гидропроводность, может быть задана явно, иначе вычисляется как произведение K^l, средней проницаемости в плоскости, перпендикулярной направлению скважины, см. ниже и h^l, размера блока в направлении, параллельном направлению скважины в данном блоке;
- r^l_o – эквивалентный радиус, см. ниже;
- $r^l_w = d^l_w/2$ – радиус участка скважины в блоке l;
- s^l – т.н. скин-фактор в блоке l.

Расчет средней проницаемости

В случае, если тензор проницаемости имеет диагональный вид, средняя проницаемость K^l рассчитывается как среднее геометрическое проницаемостей в направлениях, перпендикулярных направлению скважины. То есть,

- для скважин, направленных параллельно оси Z, проницаемость K^l рассчитывается как $K^l = (k^l_{xx}k^l_{yy})^{1/2}$

- для скважин, направленных параллельно оси X, проницаемость K^l рассчитывается как $K^l = (k^l_{yy} k^l_{zz})^{1/2}$

- для скважин, направленных параллельно оси Y, проницаемость K^l рассчитывается как $K^l = (k^l_{xx} k^l_{zz})^{1/2}$

Здесь k^l_{xx}, k^l_{yy}, k^l_{zz} – диагональные элементы тензора проницаемости, соответствующие блоку l.

Расчет эквивалентного радиуса

Эквивалентный радиус r^l_o определяется как расстояние от оси скважины, давление на котором, рассчитанное в соответствии с (55), равно давлению в ячейке, содержащей перфорацию. В декартовой сетке для его аппроксимации используется формула Писмана, которая применима для прямоугольной сетки в случае анизотропии проницаемости. Как указано выше, мы предполагаем, что скважина проходит через весь блок, через его центр, перпендикулярно двум граням.

Эквивалентный радиус для интервала перфорации l рассчитывается следующим образом:

$$r^l_o = 0.28 \frac{\left({D^l_1}^2 \cdot \left(\frac{k^l_2}{k^l_1}\right)^{1/2} + {D^l_2}^2 \cdot \left(\frac{k^l_1}{k^l_2}\right)^{1/2} \right)^{1/2}}{\left(\frac{k^l_2}{k^l_1}\right)^{1/4} + \left(\frac{k^l_1}{k^l_2}\right)^{1/4}} \tag{57}$$

где

- D^l_1 и D^l_2 – размеры блока, содержащего интервал перфорации, в направлениях, перпендикулярных направлению скважины;

- k^l_1 и k^l_2 – проницаемости в блоке, содержащем интервал перфорации в плоскости, перпендикулярной направлению скважины.

Расчет подвижностей

Подвижности фаз в случае добывающей и нагнетательной скважины рассчитываются по-разному. Для добывающих скважин подвижность зависит от условий в блоке, содержащем интервал перфорации. Для нагнетательных скважин мы используем общепринятую аппроксимацию “вниз по потоку”, которая приводит к тому, что подвижность нагнетаемой фазы равна сумме подвижностей всех трех фаз.

Для добывающих скважин подвижность фазы рассчитывается как:

$$M_P(p^l, S_W^l, S_G^l) = \frac{k_{rp}(S_W^l, S_G^l)}{\mu_P(p^l)} \tag{58}$$

Для нагнетательных скважин подвижность фазы рассчитывается как:

$$M_P(p^l, S_W^l, S_G^l) = \begin{cases} \dfrac{k_{rO}(S_W^l)}{\mu_O(p^l)} + \dfrac{k_{rW}(S_W^l, S_G^l)}{\mu_W(p^l)} + \dfrac{k_{rG}(S_G^l)}{\mu_G(p^l)}, & \text{для нагнетаемой фазы} \\ 0, & \text{для двух остальных фаз} \end{cases} \tag{59}$$

где

- $k_{rP} = k_{rP}(S_W^l, S_G^l)$ – относительная фазовая проницаемость в условиях блока сетки;
- $\mu_P = \mu_P(p^l)$ – вязкость фазы в условиях блока сетки;

Расчет средней плотности в стволе скважины и давления в скважине

Влияние трения в стволе скважины и в пласте обычно мало, и им можно пренебречь. Мы предполагаем, что флюиды движутся в соответствии с законом Дарси. Также используется предположение, что средняя плотность в

стволе скважины на каждом временном шаге постоянна и равна:

$$\bar{\rho}_{av} = \frac{\sum_l \rho_{O,SC}\tilde{q}^l_O + \rho_{W,SC}\tilde{q}^l_W + \rho_{G,SC}(\tilde{q}^l_G + R_{G,O}(p^l)\tilde{q}^l_O)}{\sum_l B_O(p_{av})\tilde{q}^l_O + B_W(p_{av})\tilde{q}^l_W + B_G(p_{av})(\tilde{q}^l_G + (R_{G,O}(p^l) - R_{G,O}(p_{av}))\tilde{q}^l_O)} \tag{60}$$

где

- $\rho_{P,SC}$ – плотность фазы P в стандартных условиях;
- $B_P = B_P(p_{av})$ – объемный коэффициент фазы, рассчитанный для среднего давления в стволе скважины;
- p_{av} – среднее давление в стволе скважины;
- $\tilde{q}^l_P = \tilde{q}^l_P(p^l, S^l_W, S^l_G, t)$ – объемный приток фазы в скважину через участок перфорации l, приведенный к стандартным условиям;

В этом случае давление в скважине на участке перфорации l $p^l_{con}(t)$ рассчитывается как

$$p^l_{con}(t) = p_{BH}(t) - \bar{\rho}_{av}(t)g(D^l - D_{BH}) \tag{61}$$

здесь

- $p_{BH}(t)$ – забойное давление в скважине, заданное или рассчитанное из уравнения (44);
- D_{BH} – глубина забоя;
- g гравитационная постоянная;
- D^l глубина середины участка перфорации l,
- средняя плотность в стволе скважины $\bar{\rho}_{av}(t)$ рассчитывается по формуле (60).

2.5. Метод Ньютона

Система уравнений (54) (или (50)), (7) является нелинейной системой алгебраических уравнений вида

$$\boldsymbol{F}(\boldsymbol{p},\boldsymbol{N}_1,\ldots\boldsymbol{N}_{n_c})=0$$

где $\boldsymbol{p}=(p^i)$, $\boldsymbol{N}_c=(N_c^i)$ — вектора значений в блоках сетки. Для решения нелинейного уравнения $\boldsymbol{F}(\boldsymbol{U})=0$, $\boldsymbol{U}\equiv(\boldsymbol{p},\boldsymbol{N})$ используется стандартный метод Ньютона:

$$\boldsymbol{U}^{m+1}=\boldsymbol{U}^m-\left(\frac{\partial\boldsymbol{F}(\boldsymbol{U}^m)}{\partial\boldsymbol{U}}\right)^{-1}\boldsymbol{F}(\boldsymbol{U}^m) \tag{62}$$

Здесь $\partial\boldsymbol{F}(\boldsymbol{U}^m)/\partial\boldsymbol{U}$ — отображение (матрица) $\boldsymbol{R}^{(1+n_c)N}\to\boldsymbol{R}^{(1+n_c)N}\times\boldsymbol{R}^{(1+n_c)N}$, N — число блоков сетки. На каждом шаге метода Ньютона надо решать систему с несимметричной матрицей $\partial\boldsymbol{F}(\boldsymbol{U}^m)/\partial\boldsymbol{U}$.

Из-за высокой жесткости задачи (особенно из-за слагаемых вида δ-функций) и разрывов производных при фазовых переходах (см. с. 33), все производные при формировании матрицы Якобиана обычно приходится вычислять аналитически. Именно, требуется вычислить

$$\frac{\partial F_m(\boldsymbol{p},\boldsymbol{N})}{\partial p^{i',j',k'}},\quad \frac{\partial F_m(\boldsymbol{p},\boldsymbol{N})}{\partial N_\alpha^{i',j',k'}},\quad \alpha=1,\ldots,n_c$$

для всех $i'=1,\ldots,N^x$, $j'=1,\ldots,N^y$, $k'=1,\ldots,N^z$ и $m=1,\ldots,n_c$.

Обозначим

$$\delta^{i,j,k}_{i',j',k'}=\begin{cases}1 & \text{если } i=i',j=j',k=k'\\ 0 & \text{иначе}\end{cases}$$

Тогда в (54) имеем для всех $\alpha=1,\ldots,n_c$

$$\frac{\partial\phi^{i,j,k}(p)}{\partial p^{i',j',k'}}=\delta^{i,j,k}_{i',j',k'}\left.\frac{\partial\phi(p)}{\partial p}\right|^{i,j,k}\equiv\delta^{i,j,k}_{i',j',k'}\frac{\partial\phi^{i,j,k}(p)}{\partial p},\quad \frac{\partial\phi^{i,j,k}(p)}{\partial N_\alpha^{i',j',k'}}=0 \tag{63}$$

Для потоковых аппроксимаций:

$$\frac{\partial U^{i+,j,k}(F(\boldsymbol{p}),\alpha(\boldsymbol{p}))}{\partial p^{i',j',k'}} = \frac{\partial \alpha(\boldsymbol{p})}{\partial p^{i',j',k'}} \left\{ \begin{array}{lll} F^{i,j,k}(\boldsymbol{p}) & \text{если } i = N^x \text{ или} & \alpha(\boldsymbol{p}) \le 0 \\ F^{i+1,j,k}(\boldsymbol{p}) & \text{если } i < N^x \text{ и} & \alpha(\boldsymbol{p}) > 0 \end{array} \right\}$$

$$+\alpha(\boldsymbol{p}) \left\{ \begin{array}{lll} \dfrac{\partial F^{i,j,k}(\boldsymbol{p})}{\partial p^{i',j',k'}} & \text{если } i = N^x \text{ или} & \alpha(\boldsymbol{p}) \le 0 \\ \dfrac{\partial F^{i+1,j,k}(\boldsymbol{p})}{\partial p^{i',j',k'}} & \text{если } i < N^x \text{ и} & \alpha(\boldsymbol{p}) > 0 \end{array} \right.$$

$$= \frac{\partial \alpha(\boldsymbol{p})}{\partial p^{i',j',k'}} \left\{ \begin{array}{lll} F^{i,j,k}(\boldsymbol{p}) & \text{если } i = N^x \text{ или} & \alpha(\boldsymbol{p}) \le 0 \\ F^{i+1,j,k}(\boldsymbol{p}) & \text{если } i < N^x \text{ и} & \alpha(\boldsymbol{p}) > 0 \end{array} \right\}$$

$$+\alpha(\boldsymbol{p}) \left\{ \begin{array}{lll} \delta^{i,j,k}_{i',j',k'} \dfrac{\partial F^{i,j,k}(\boldsymbol{p})}{\partial p} & \text{если } i = N^x \text{ или} & \alpha(\boldsymbol{p}) \le 0 \\ \delta^{i+1,j,k}_{i',j',k'} \dfrac{\partial F^{i+1,j,k}(\boldsymbol{p})}{\partial p} & \text{если } i < N^x \text{ и} & \alpha(\boldsymbol{p}) > 0 \end{array} \right.$$

Тогда

$$\frac{\partial U^{i+,j,k}(F(\boldsymbol{p},\boldsymbol{N}),\alpha(\boldsymbol{p},\boldsymbol{N}))}{\partial p^{i',j',k'}} = U^{i+,j,k}\left(\frac{\partial F(\boldsymbol{p},\boldsymbol{N})}{\partial p^{i',j',k'}}, \alpha(\boldsymbol{p},\boldsymbol{N})\right) + \frac{\partial \alpha(\boldsymbol{p},\boldsymbol{N})}{\partial p^{i',j',k'}} \widehat{U}^{i+,j,k}(F(\boldsymbol{p},\boldsymbol{N}),\alpha(\boldsymbol{p},\boldsymbol{N}))$$

Аналогично вычисляются производные относительно $N_c^{i',j',k'}$.

Вычислим производные по давлению от потока (53):

$$\frac{\partial \Phi_P^{i+,j,k}(p,\boldsymbol{N})}{\partial p^{i',j',k'}} = \delta_{i',j',k'}^{i+1,j,k} - \delta_{i',j',k'}^{i,j,k} + \delta_{i',j',k'}^{i+1,j,k}\frac{\partial P_{cOP}^{i+1,j,k}(p,\boldsymbol{N})}{\partial p} - \delta_{i',j',k'}^{i,j,k}\frac{\partial P_{cOP}^{i,j,k}(p,\boldsymbol{N})}{\partial p} - \frac{\partial \gamma_P^{i+,j,k}(p,\boldsymbol{N})}{\partial p^{i',j',k'}}(d^{i+1,j,k} - d^{i,j,k}) \quad (64)$$

$$\frac{\partial \Phi_G^{i-,j,k}(p,\boldsymbol{N})}{\partial p^{i',j',k'}} = \delta_{i',j',k'}^{i,j,k} - \delta_{i',j',k'}^{i-1,j,k} + \delta_{i',j',k'}^{i,j,k}\frac{\partial P_{cOP}^{i,j,k}(p,\boldsymbol{N})}{\partial p} - \delta_{i',j',k'}^{i-1,j,k}\frac{\partial P_{cOP}^{i-1,j,k}(p,\boldsymbol{N})}{\partial p} - \frac{\partial \gamma_P^{i-,j,k}(p,\boldsymbol{N})}{\partial p^{i',j',k'}}(d^{i,j,k} - d^{i-1,j,k})$$

$$\frac{\partial \Phi_P^{i,j+,k}(p,\boldsymbol{N})}{\partial p^{i',j',k'}} = \delta_{i',j',k'}^{i,j+1,k} - \delta_{i',j',k'}^{i,j,k} + \delta_{i',j',k'}^{i,j+1,k}\frac{\partial P_{cOP}^{i,j+1,k}(p,\boldsymbol{N})}{\partial p} - \delta_{i',j',k'}^{i,j,k}\frac{\partial P_{cOP}^{i,j,k}(p,\boldsymbol{N})}{\partial p} - \frac{\partial \gamma_P^{j,j+,k}(p,\boldsymbol{N})}{\partial p^{i',j',k'}}(d^{i,j+1,k} - d^{i,j,k})$$

$$\frac{\partial \Phi_G^{i,j-,k}(p,\boldsymbol{N})}{\partial p^{i',j',k'}} = \delta_{i',j',k'}^{i,j,k} - \delta_{i',j',k'}^{i,j-1,k} + \delta_{i',j',k'}^{i,j,k}\frac{\partial P_{cOP}^{i,j,k}(p,\boldsymbol{N})}{\partial p} - \delta_{i',j',k'}^{i,j-1,k}\frac{\partial P_{cOP}^{i,j-1,k}(p,\boldsymbol{N})}{\partial p} - \frac{\partial \gamma_P^{j,j-,k}(p,\boldsymbol{N})}{\partial p^{i',j',k'}}(d^{i,j,k} - d^{i,j-1,k})$$

$$\frac{\partial \Phi_P^{i,j,k+}(p,\boldsymbol{N})}{\partial p^{i',j',k'}} = \delta_{i',j',k'}^{i,j,k+1} - \delta_{i',j',k'}^{i,j,k} + \delta_{i',j',k'}^{i,j,k+1}\frac{\partial P_{cOP}^{i,j,k+1}(p,\boldsymbol{N})}{\partial p} - \delta_{i',j',k'}^{i,j,k}\frac{\partial P_{cOP}^{i,j,k}(p,\boldsymbol{N})}{\partial p} - \frac{\partial \gamma_P^{j,j,k+}(p,\boldsymbol{N})}{\partial p^{i',j',k'}}(d^{i,j,k+1} - d^{i,j,k})$$

$$\frac{\partial \Phi_G^{i,j,k-}(p,\boldsymbol{N})}{\partial p^{i',j',k'}} = \delta_{i',j',k'}^{i,j,k} - \delta_{i',j',k'}^{i,j,k-1} + \delta_{i',j',k'}^{i,j,k}\frac{\partial P_{cOP}^{i,j,k}(p,\boldsymbol{N})}{\partial p} - \delta_{i',j',k'}^{i,j,k-1}\frac{\partial P_{cOP}^{i,j,k-1}(p,\boldsymbol{N})}{\partial p} - \frac{\partial \gamma_P^{j,j,k-}(p,\boldsymbol{N})}{\partial p^{i',j',k'}}(d^{i,j,k} - d^{i,j,k-1})$$

Вычислим $\partial F_c^{i,j,k}/\partial p^{i',j',k'}$, $c = 1,\ldots,n_c$:

$$
\begin{aligned}
\frac{\partial F_c^{i,j,k}}{\partial p^{i',j',k'}} \;=\; & \delta^{i,j,k}_{i',j',k'} \nu^{i,j,k} \frac{\partial \phi^{i,j,k}(p)}{\partial p} N_c^{i,j,k} - \tau \frac{\partial q_c^{i,j,k,\beta}(p,\boldsymbol{N})}{\partial p^{i',j',k'}} \\
& -\tau \sum_P \Bigg(T_x^{i+,j,k}(\boldsymbol{k}) \Bigg(U^{i+,j,k} \left(\frac{\partial \Lambda_{c,P}(p,\boldsymbol{N})}{\partial p^{i',j',k'}}, \Phi_P^{i+,j,k}(p,\boldsymbol{N}) \right) \\
& \quad + \frac{\partial \Phi_P^{i+,j,k}(p,\boldsymbol{N})}{\partial p^{i',j',k'}} \widehat{U}^{i+,j,k} \left(\Lambda_{c,P}(p,\boldsymbol{N}), \Phi_P^{i+,j,k}(p,\boldsymbol{N}) \right) \Bigg) \\
& -T_x^{i-,j,k}(\boldsymbol{k}) \Bigg(U^{i-,j,k} \left(\frac{\partial \Lambda_{c,P}(p,\boldsymbol{N})}{\partial p^{i',j',k'}}, \Phi_P^{i-,j,k}(p,\boldsymbol{N}) \right) \\
& \quad + \frac{\partial \Phi_P^{i-,j,k}(p,\boldsymbol{N})}{\partial p^{i',j',k'}} \widehat{U}^{i-,j,k} \left(\Lambda_{c,P}(p,\boldsymbol{N}), \Phi_P^{i-,j,k}(p,\boldsymbol{N}) \right) \Bigg) \\
& +T_y^{i,j+,k}(\boldsymbol{k}) \Bigg(U^{i,j+,k} \left(\frac{\partial \Lambda_{c,P}(p,\boldsymbol{N})}{\partial p^{i',j',k'}}, \Phi_P^{i,j+,k}(p,\boldsymbol{N}) \right) \\
& \quad + \frac{\partial \Phi_P^{i,j+,k}(p,\boldsymbol{N})}{\partial p^{i',j',k'}} \widehat{U}^{i,j+,k} \left(\Lambda_{c,P}(p,\boldsymbol{N}), \Phi_P^{i,j+,k}(p,\boldsymbol{N}) \right) \Bigg) \\
& -T_y^{i,j-,k}(\boldsymbol{k}) \Bigg(U^{i,j-,k} \left(\frac{\partial \Lambda_{c,P}(p,\boldsymbol{N})}{\partial p^{i',j',k'}}, \Phi_P^{i,j-,k}(p,\boldsymbol{N}) \right) \\
& \quad + \frac{\partial \Phi_P^{i,j-,k}(p,\boldsymbol{N})}{\partial p^{i',j',k'}} \widehat{U}^{i,j-,k} \left(\Lambda_{c,P}(p,\boldsymbol{N}), \Phi_P^{i,j-,k}(p,\boldsymbol{N}) \right) \Bigg) \\
& +T_z^{i,j,k+}(\boldsymbol{k}) \Bigg(U^{i,j,k+} \left(\frac{\partial \Lambda_{c,P}(p,\boldsymbol{N})}{\partial p^{i',j',k'}}, \Phi_P^{i,j,k+}(p,\boldsymbol{N}) \right) \\
& \quad + \frac{\partial \Phi_P^{i,j,k+}(p,\boldsymbol{N})}{\partial p^{i',j',k'}} \widehat{U}^{i,j,k+} \left(\Lambda_{c,P}(p,\boldsymbol{N}), \Phi_P^{i,j,k+}(p,\boldsymbol{N}) \right) \Bigg) \\
& -T_z^{i,j,k-}(\boldsymbol{k}) \Bigg(U^{i,j,k-} \left(\frac{\partial \Lambda_{c,P}(p,\boldsymbol{N})}{\partial p^{i',j',k'}}, \Phi_P^{i,j,k-}(p,\boldsymbol{N}) \right) \\
& \quad + \frac{\partial \Phi_P^{i,j,k-}(p,\boldsymbol{N})}{\partial p^{i',j',k'}} \widehat{U}^{i,j,k-} \left(\Lambda_{c,P}(p,\boldsymbol{N}), \Phi_P^{i,j,k-}(p,\boldsymbol{N}) \right) \Bigg) \Bigg)
\end{aligned}
\tag{65}
$$

Аналогично вычисляются производные относительно $N_c^{i',j',k'}$.

Глава 1

Математическая модель техногенной трещиноватости

Технология гидроразрыва пласта (ГРП) применяется практически на 100% скважин в условиях низкопроницаемого коллектора (малой величины компонент тензора проницаемости $\boldsymbol{k}$ в (4)). Реалистичный учет влияния искусственных трещин, возникающих при проведении на скважинах мероприятий ГРП, и так называемых техногенных трещин, образующихся на нагнетательных скважинах при превышении давлением закачки величины горного давления, остается одной из острых проблем при математическом моделировании крупных (с большим числом скважин) месторождений нефти и газа. Существующие методы позволяют либо детально описывать трещиноватость вблизи одиночных скважин путем значительного увеличения разрешения сетки и приближения реальной геометрии трещин, либо, считая трещину локализованной в одной ячейке сетки, описывать трещиноватость повышением эффективности перфорации, задавая отрицательное значение скин фактора s^l в (56). Последний подход применим для крупных месторождений, однако, при увеличении степени детализации сеточных аппроксимаций все чаще возникает ситуация, когда размеры трещин начинают превышать размеры ячеек сетки, и применение этого подхода приводит к значительному искажению свойств математической модели и фактически разрушает ее прогнозную силу.

Ниже мы рассмотрим математическую модель для описания гидравлических трещин с помощью создания сети дополнительных “виртуальных” перфораций скважины в блоках сетки, через которые проходит трещина, что позволяет реалистично моделировать динамику поведения крупных гидравлических трещин на сотнях и тысячах скважин. Также мы рассмотрим, как влия-

ет такой способ описания гидравлических трещин на сложность получаемой в результате аппроксимации задачи фильтрации матрицы системы линейных уравнений и сложность решения этой системы на параллельных ЭВМ.

1.1. Проблемы описания техногенной трещиноватости

Рассмотрим ситуацию, в которой полудлина трещины ГРП или техногенной трещины на нагнетательной скважине превышает размер блока ячейки сетки. Как отмечалось выше, задание отрицательного значения скин-фактора уже не является достаточным для адекватного описания трещины ([42]). Существующие подходы описания трещин ГРП в полномасштабных гидродинамических моделях подразумевают формирование каналов вдоль направления трещин с помощью комбинирования ячеек исходной сетки и значительного увеличения в них проницаемости, как показано на рис. 1.1 (где показаны искусственно созданные зоны с повышенной проницаемостью, расположенные вблизи добывающих (красные) и нагнетательных (синие) скважин). Данный подход частично воспроизводит быстрый прорыв жидкости или газа вдоль направления трещины, но имеет два существенных недостатка. Во-первых, значительно замедляется расчет из-за значительного увеличения гетерогенности тензора проницаемости. Во-вторых, поскольку “каналы” формируются из числа обычных ячеек сетки, то прорыв жидкости или газа по трещине неизбежно моделируется через фильтрационный механизм. Физика фильтрации подразумевает, что жидкость или газ могут прорваться по каналу только за счет полного или частичного (с учетом относительных фазовых проницаемостей и капиллярных эффектов) вытеснения запасов жидкости или газа, находящихся в самих ячейках, формирующих этот канал. В реальности, учитывая относительную слабость влияния смачивания и капиллярных эффектов в трещине, прорыв жидкости или газа происходит очень быстро, скорее по

гидравлическому, а не фильтрационному механизму. Запасы ячеек, через которые проходит трещина, дренируются постепенно через поверхность трещины. Такое фундаментальное нарушение в моделировании динамики извлечения запасов приводит к формированию систематически искаженного распределения остаточных запасов жидкостей и газа и ухудшает предсказательную силу прогнозных расчетов.

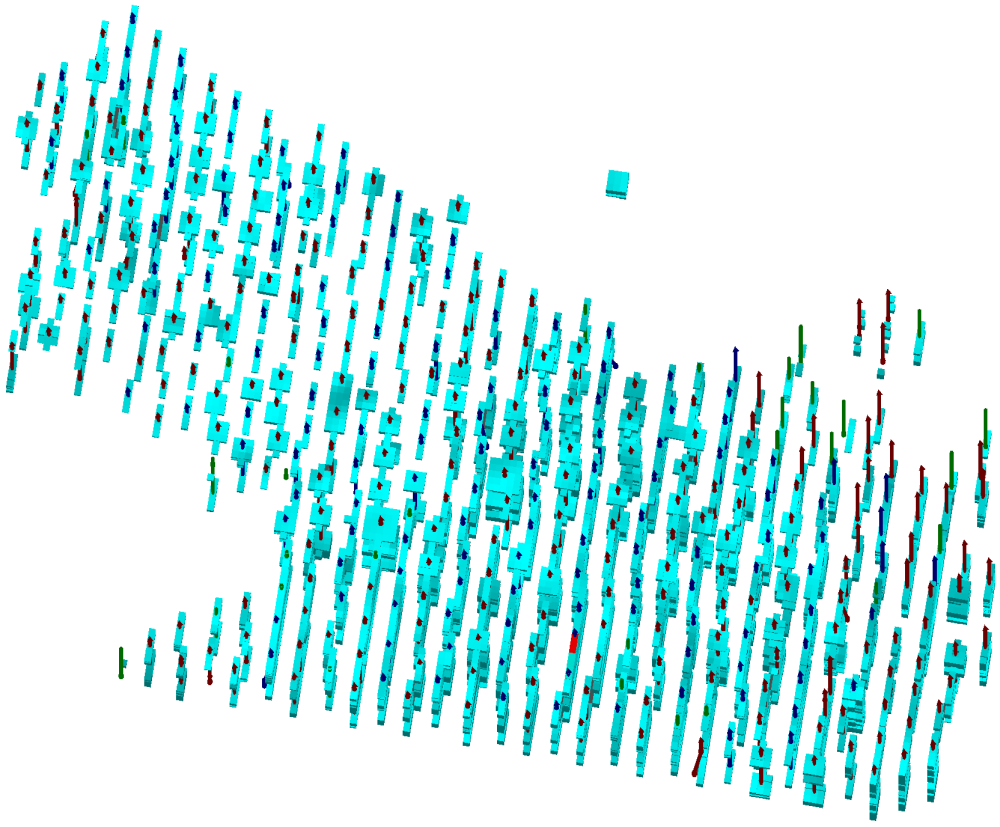


Рис. 1.1. Пример моделирования трещин ГРП путем создания высокопроводящих “каналов”

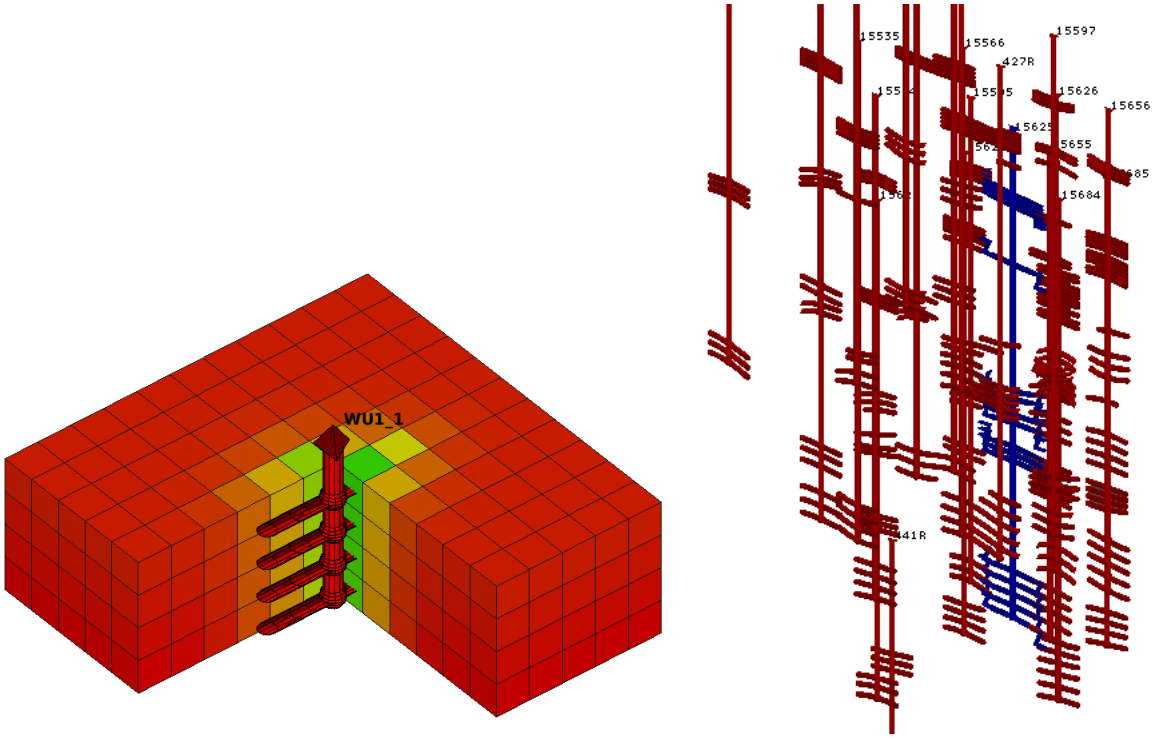
Для моделирования автоматического раскрытия техногенных трещин на инжекторах сейчас применяется другой, еще менее физически обоснованный подход. Задают зависимость тензора проницаемости $\boldsymbol{k}$ в (4) от давления так, что проницаемость начинает возрастать с ростом давления и происходит как бы “развитие” трещины. В этом подходе имеются 2 проблемы:

- С точки зрения физики проницаемость падает с ростом давления, так как происходит уплотнение пласта.

- “Трещина” получается не ориентированной, и в случае однородного пласта возникает концентрическая (с центром в перфорации) зона повышенной проницаемости.

1.2. Реалистичное моделирование трещин ГРП через сеть виртуальных перфораций

Для более реалистичного моделирования динамики поведения трещин ГРП рассмотрим трещину не как часть сетки гидродинамической модели с повышенной проницаемостью, а как новые участки перфораций скважины в блоках, пересекаемых траекторией трещины, см. рис. 1.2.



(*а*). Одна скважина (*б*). Система скважин

Рис. 1.2. Пример системы “виртуальных” перфораций

Для описания гидроразрыва пласта (ГРП) используются следующие данные:

- имя скважины и дата проведения мероприятия;
- свойства закачанного пропанта (проницаемость в зависимости от давления в пласте);
- азимут образовавшихся трещин;
- их полудлина;
- их раскрытие (толщина трещины у скважины);
- их высота (номера первой и последней перфорации);
- зенитный угол трещин.

В математической модели мы должны описать:

- связи участков перфорации с блоками, возникшие вследствие ГРП (приток в трещину ГРП);
- течение вдоль трещины ГРП.

Мы рассмотрим два варианта описания трещины через сеть виртуальных перфораций:

- трещина представляется в виде цепочки соединенных блоков, начинающейся из “реальной” перфорации скважины;
- трещина представляется в виде набора блоков, соединенных со скважиной.

Достоинством первого варианта является возможность разделить приток из трещины ГРП между участками перфорации скважины (т.е. приток вычисляется из трещины в конкретный участок перфорации), а недостатком – высокая вычислительная сложность. У второго варианта все наоборот: низкая вычислительная сложность и невозможность разделить приток из трещины ГРП

между участками перфорации скважины (т.е. приток вычисляется из трещины в скважину).

В обоих вариантах приток в виртуальную перфорацию будем вычислять так же, как в обычную, а сопротивление течению вдоль трещины будем учитывать, вводя в формулу (56) дополнительный множитель γ^l.

1.2.1. Моделирование цепочки перфораций

Рассмотрим трещину как цепочку перфораций, выходящую из ячейки сетки с номером l, где находится "реальная" перфорация.

Вычисление течения вдоль трещины Для участка перфорации l, соединенного (посредством трещины) с блоком i, положим (индекс j ниже проходит только по номерам блоков трещины между блоками l и i):

$$\gamma_i^l = \frac{M_l}{\sum\limits_{j=i}^{l} \frac{1}{K^j}} \tag{1.1}$$

где K^j – проводимость вдоль трещины в блоке j. Эти выражения аналогичны формулам вычисления коэффициентов проводимости блоков сетки и обеспечивают:

- **нулевое значение** $\gamma_i^l = 0$, если проницаемость K^j **хотя бы одного** из участков трещины между участками l и i (включительно) равна нулю;
- **накопительный** характер сопротивления течению между участками l и i в зависимости от **длины и проводимости** пути между ними.

Для моделирования эффектов, связанных с зависимостью эффективности ГРП от проницаемости и вымывания пропанта, забивания трещины и от давления, можно сделать величины K^j зависящими от:

- площади трещины и свойств пропанта;
- давления в блоке, через который проходит трещина;
- прошедшего через связь блоков потока;
- времени.

Вычисление притока из блока в трещину Заменим трещину в пределах блока j на виртуальный участок перфорации скважины, такой, что:

- периметр сечения ствола скважины равен периметру трещины (что дает формулу для вычисления диаметра “скважины”);
- длина перфорированного участка равна длине трещины в блоке;
- приток в участок перфорации скважины определяется проницаемостью блока в направлении, ортогональном грани трещины с наибольшей площадью.

По этим данным мы можем вычислить множитель проводимости этого виртуального участка перфорации (обозначим его $\tilde{\Theta}^{w,j}$), а затем вычислить приток по обычной формуле.

Иллюстрация вычислений приведена на рисунке 1.3, где изображена трещина, идущая вдоль оси Y, высотой h и шириной w, намного меньшей h. Приток в такую трещину будем считать равным притоку в скважину диаметром $d = 2(w+h)/\pi$, имеющей ориентацию Y и находящейся в блоке с проницаемостью по X и Z, равной проницаемости по X блока с трещиной. Получаем

$$\tilde{\Theta}^{w,j} = \frac{2\pi\beta_c K^{w,j} h^{w,j}}{\log(r_0^{w,j}/r_w^{w,j})}, \quad r_w^{w,j} = (w+h)/\pi,$$
$$r_0^{w,j} = 0.14 * \sqrt{\mathbf{DX}^2 + \mathbf{DZ}^2}, \quad K^{w,j} = \boldsymbol{k}_x, \quad h^{w,j} = \mathbf{DY}$$

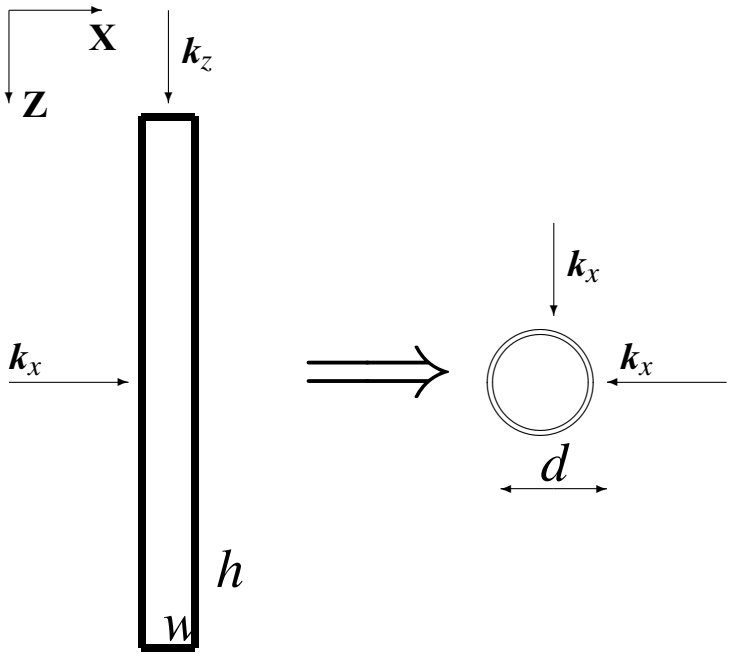


Рис. 1.3. Вычисление эквивалентного притока в трещину

где **DX**, **DY**, **DZ** – геометрические размеры блока, через который проходит трещина, $\boldsymbol{k}_x$, $\boldsymbol{k}_y$, $\boldsymbol{k}_z$ – диагональ тензора проницаемости (который здесь предполагается диагональным).

1.2.2. Моделирование большой сети перфораций

Создадим виртуальную перфорацию в каждой ячейке сетки, через которую проходит трещина. В частности, можно рассматривать трещины ГРП, перпендикулярные стволу скважины, см. пример на рис. 1.4, где показаны множественные ГРП, обычно производимые на месторождениях сланцевого газа.

Обозначим блоки, через которые прошла трещина, через $l_1, \ldots, l_L$, средний периметр трещины в блоке l_i – через Π_{l_i}, среднюю длину трещины в блоке l_i – через L_{l_i}, площадь сечения трещины в блоке l_i – через S_{l_i}, расстояние от блока l_i до ствола скважины – через D_{l_i}. Для упрощения обозначений будем считать плоскость трещины вертикальной и квази-ортогональной направле-

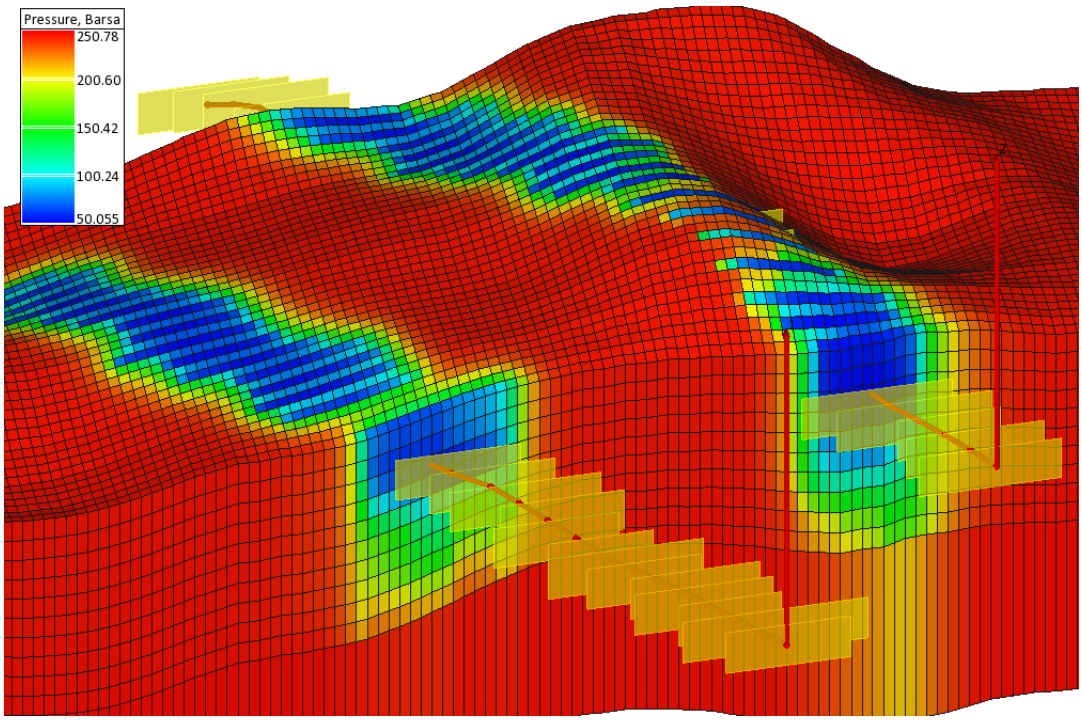


Рис. 1.4. Пример моделирования трещин ГРП на месторождении сланцевого газа

нию OX, см. рис. 1.3. Для других положений трещины вычисления аналогичны. Для вычисление притока из блока l_i в трещину используем формулу из предыдущего раздела

$$\tilde{\Theta}^{w,l_i} = \frac{2\pi\beta_c \cdot \boldsymbol{k}_x \cdot L_{l_i}}{\log\left(0.28 * \pi * \sqrt{\mathbf{DX}^2 + \mathbf{DZ}^2}/S_{l_i}\right)},$$

Для вычисления течения вдоль трещины мы должны приблизить выражение (1.1) для эффективности течения вдоль трещины (фактически множителя перед $\tilde{\Theta}^{w,l_i}$). Для этого должны быть заданы свойства пропанта в трещине:

- **проницаемость пропанта** $K(p)$ как функция давления в блоке; если не задано, то считается $K = \infty$;

- **зависимость проницаемости от потока** указанной фазы или жидкости; задается как функция f безразмерного потока s (отношения накопленного потока из блока к поровому объему блока), такая, что

  1. $f(0) = 1$;

2. $f(s) \geq 0$ для всех s.

Если не задана, то считается $f(s) = 1$.

Результирующая эффективная проводимость пропанта в блоке l_i с площадью сечения трещины S_{l_i} полагается равной

$$K_{l_i}(p^{l_i}, s) = K(p^{l_i}) \cdot S_{l_i} \cdot f(s).$$

Заметим, что функция $K_{l_i}(p^{l_i}, s)$ является безразмерной. Мы хотим определить выражение γ_{l_i} (см. (1.1)) для эффективности течения вдоль трещины из блока l_i в скважину так, чтобы:

1. γ_{l_i} является только функцией D_{l_i} при $K(p) = \infty$, $\gamma_{l_i} = 1$ при $K(p) = \infty$ и $D_{l_i} = 0$, что соответствует дренированию блока, в котором находится сама скважина;

2. $\gamma_{l_i} = 0$ при $K(p) = 0$;

3. $\gamma_{l_i} = 0$ при $s = \infty$.

Рассмотрим следующую функцию:

$$\gamma_{l_i}(p^{l_i}, s) = \frac{K_{l_i}(p, s)}{1 + K_{l_i}(p, s)} \cdot \frac{1}{1 + D_{l_i}/L_{l_i}}.$$

Заметим, что величина $\gamma_{l_i}(p^{l_i}, s)$ является безразмерной.

1.3. Апробация моделирования трещин ГРП через сеть виртуальных перфораций

Предложенная математическая модель ГРП проверялась на воспроизведение качественных и количественных показателей процесса разработки месторождения. Например, изображенная на рис. 1.4 карта давления качественно

воспроизводит падение давления вдоль трещины. На рис. 1.5 показана карта нефтенасыщенности и хорошо видны овальные зоны ее падения вокруг нагнетательных скважин (показаны синим цветом). Овальная форма этой зоны обеспечивает правильное воспроизведение времени прорыва воды и распределения остаточных запасов.

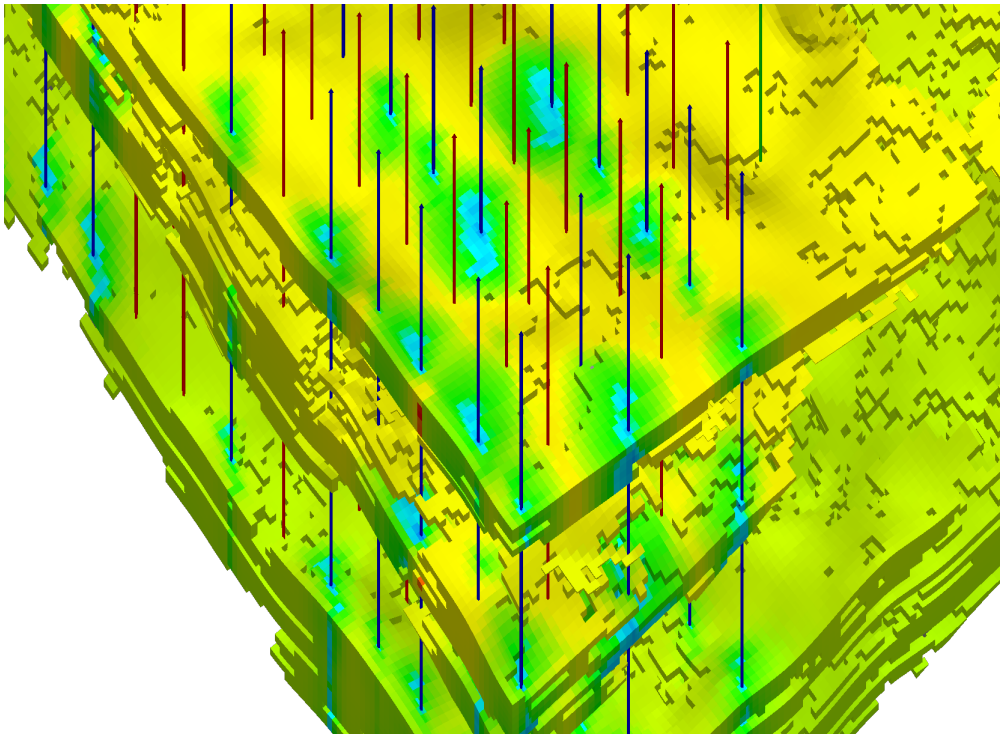


Рис. 1.5. Пример моделирования трещин ГРП на месторождении нефти

Эта математическая модель гидроразрыва пласта реализована в комплексе программ tNavigator, предложенном в работе. Она была опробована при расчете реальных месторождений Западной Сибири и была рекомендована к применению в Российских компаниях “ТНК-ВР” и “Газпромнефть”.

Немаловажным фактором принятия этой модели ГРП к практическому использованию явилась высокая скорость расчета по сравнению с другими способами моделирования ГРП. Заметим, что предложенная математическая модель гидроразрыва пласта значительно повышает сложность решаемой после аппроксимации задачи фильтрации системы уравнений и является очень

серьезным испытанием для любого метода ее решения. Рассмотрим основные факторы, влияющие на сложность решаемой задачи, особенно на параллельных ЭВМ.

1.4. Решение системы линейных уравнений

Рассмотрим способы решения системы линейных уравнений с матрицей (65), возникающей на каждой итерации метода Ньютона (62), с помощью которого обычно решается система нелинейных уравнений (54), аппроксимирующая исходную систему (4).

1.4.1. Особенности системы уравнений

Разреженная матрица (65) системы линейных уравнений, возникающей на каждой итерации метода Ньютона (62), во многом отличается от разреженных матриц, к которым обычно приводят аппроксимации, например, уравнений теплопроводности или Навье-Стокса. Выделим основные из них:

- большой шаблон матрицы (количество ненулевых элементов в строке) из-за геометрических особенностей сетки, аппроксимации скважин и водонапорных горизонтов;
- большое количество дополнительных нелинейных алгебраических уравнений, возникающих при аппроксимации скважин и водонапорных горизонтов;
- высокая жесткость задачи из-за источников и стоков.

Количество ненулевых элементов в строке матрицы (65) определяется количеством связей блока (i, j, k) с другими блоками — т.е. количеством бло-

ков сетки, из(в) которых возможно течение. Сами связи можно разделить на несколько групп:

- Геометрические — это связи в силу геометрии самой сетки. Для равномерной сетки течение из блока (i, j, k) возможно только из(в) его непосредственных соседей: $(i \pm 1, j, k)$, $(i, j \pm 1, k)$, $(i, j, k \pm 1)$. Это дает 7 ненулевых элементов в строке. Но при наличии разломов и выклинивания (см. ниже) это количество может быть значительно выше.

- В силу скважин — это связи из-за наличия скважин (см. ниже).

- В силу водонапорных горизонтов — это связи из-за наличия водонапорных горизонтов (см. ниже).

Матрица и геологические особенности

При наличии разломов и выклинивания течение смеси из одного блока возможно в десятки и даже сотни других блоков (например, когда высота столбика из сотни блоков равна высоте стороны одного блока), см. рис. 1.6 и рис. 1.7.

Матрица и скважины

Ствол скважины представляет (с точки зрения математической модели) трубу, в которой в некоторых местах сделаны отверстия (перфорации). Таких перфораций в скважине обычно много и они находятся в разных блоках сетки. С силу возможности течения в стволе скважины получается, что все такие блоки гидродинамически связаны. Другими словами, если скважина проходит и перфорирована в блоках сетки $l_1, \ldots, l_M$ (см. рис. 1.8), то в матрице (65) в строках $l_1, \ldots, l_M$, будут ненулевые элементы в столбцах $l_1, \ldots, l_M$.

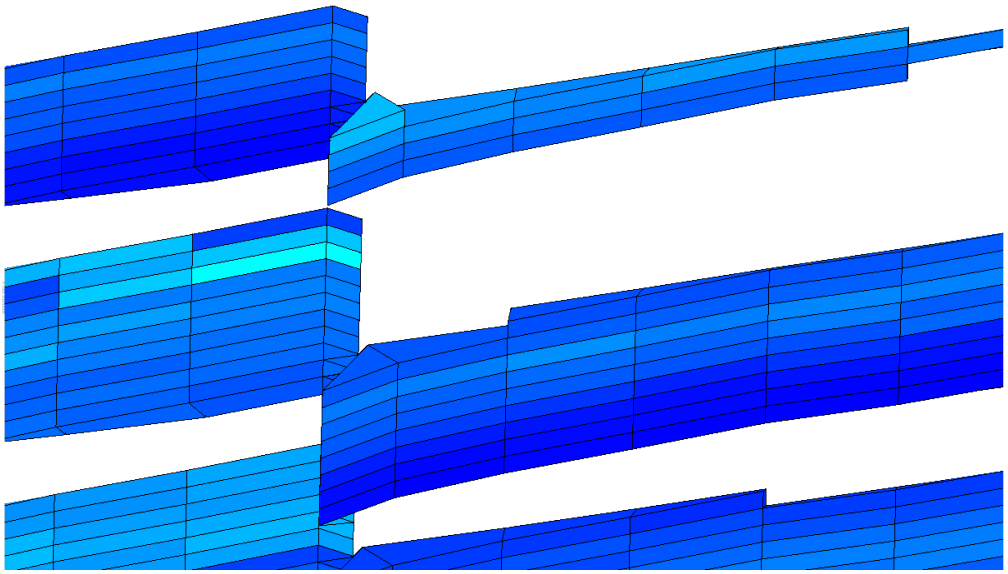


Рис. 1.6. Примеры разломов

При типичном геометрическом размере сетки в плоскости XY, равном 100×100 метров возможно, что через этот блок проходит более одной скважины. Особенно часто это случается при рассмотрении т.н. кустов скважин, когда несколько скважин бурятся с поверхности из одной точки и расходятся на глубине, см. рис. 1.9. Тогда получается, что блоки сетки, через которые проходит одна из скважин, будут гидродинамически связаны. Например, если скважина 1 проходит через блоки $l_1^1, l_2^1 \ldots, l_{M_1}^1$, а скважина 2 – через блоки $l_1^2, l_2^2 \ldots, l_{M_2}^2$, и $l_1^1 = l_1^2$, то в матрице (65) в строках $l_1^1, l_2^1 \ldots, l_{M_1}^1, l_2^2 \ldots, l_{M_2}^2$, будут ненулевые элементы в столбцах $l_1^1, l_2^1 \ldots, l_{M_1}^1, l_2^2 \ldots, l_{M_2}^2$.

Поверхностное оборудование (например, газосборная сеть) могут диктовать дополнительные ограничения на работу скважин, например, суммарная добыча с группы скважин должна быть равна заданному значению, см. рис. 1.10. Это приводит к дополнительным нелинейным алгебраическим уравнениям, связывающим режимы работы разных скважин.

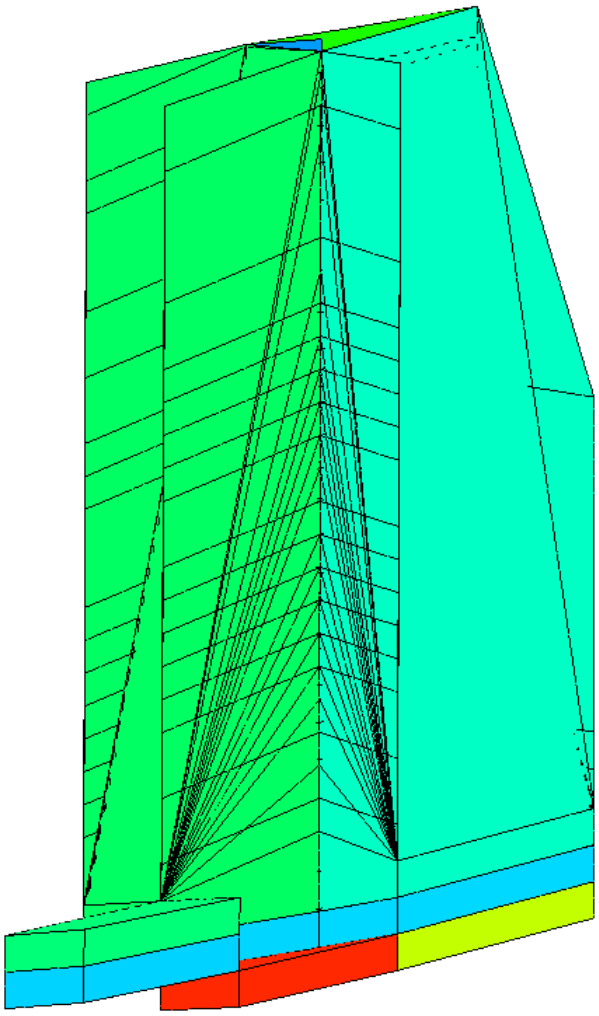


Рис. 1.7. Примеры выклинивания

Матрица и водонапорные горизонты

Водонапорные горизонты (см. рис. 1.11), часто используемые для моделирования незамкнутости резервуара, приводят к возникновению потоков, зависящих от предыдущей истории и текущего суммарного потока, т.е. к еще одному линейному уравнению, связывающему все блоки, подключенные к водонапорному горизонту. Для этого уравнения (пусть его номер l) число ненулевых элементов в соответствующей строке матрицы (65) будет равно количеству блоков $l_1, \ldots, l_M$, к которым подключен водонапорный горизонт. Также в каждой из строк $l_1, \ldots, l_M$ добавляется ненулевой элемент в столбце l.

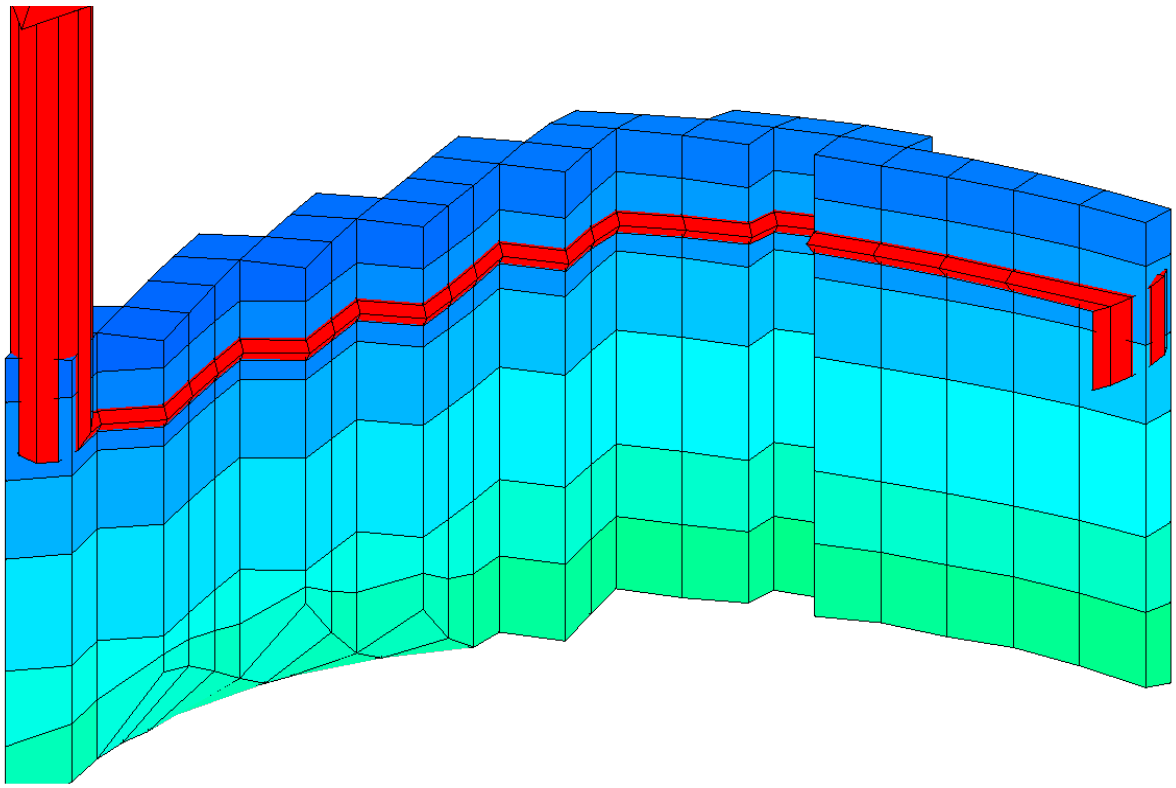


Рис. 1.8. Пример ствола скважины

1.4.2. Решение системы линейных уравнений

Для решения системы линейных уравнений с несимметричной разреженной матрицей (65) в-основном используются 2 алгоритма

- GMRES (Generalized Minimum Residual Method) и его варианты — основаны на минимизации невязки в пространстве Крылова;
- BCGS (BiConjugate Gradient Stabilized) — основан на методе биортогонализации Ланцоша.

Вопрос о том, какой итерационный метод эффективнее, с математической точки зрения открыт.

Самым сложным вопросом при решении системы уравнений (65) является выбор предобуславливателя. Из-за высокой жесткости системы решать систему без предобуславливателя или с простейшими вариантами (Якоби, на-

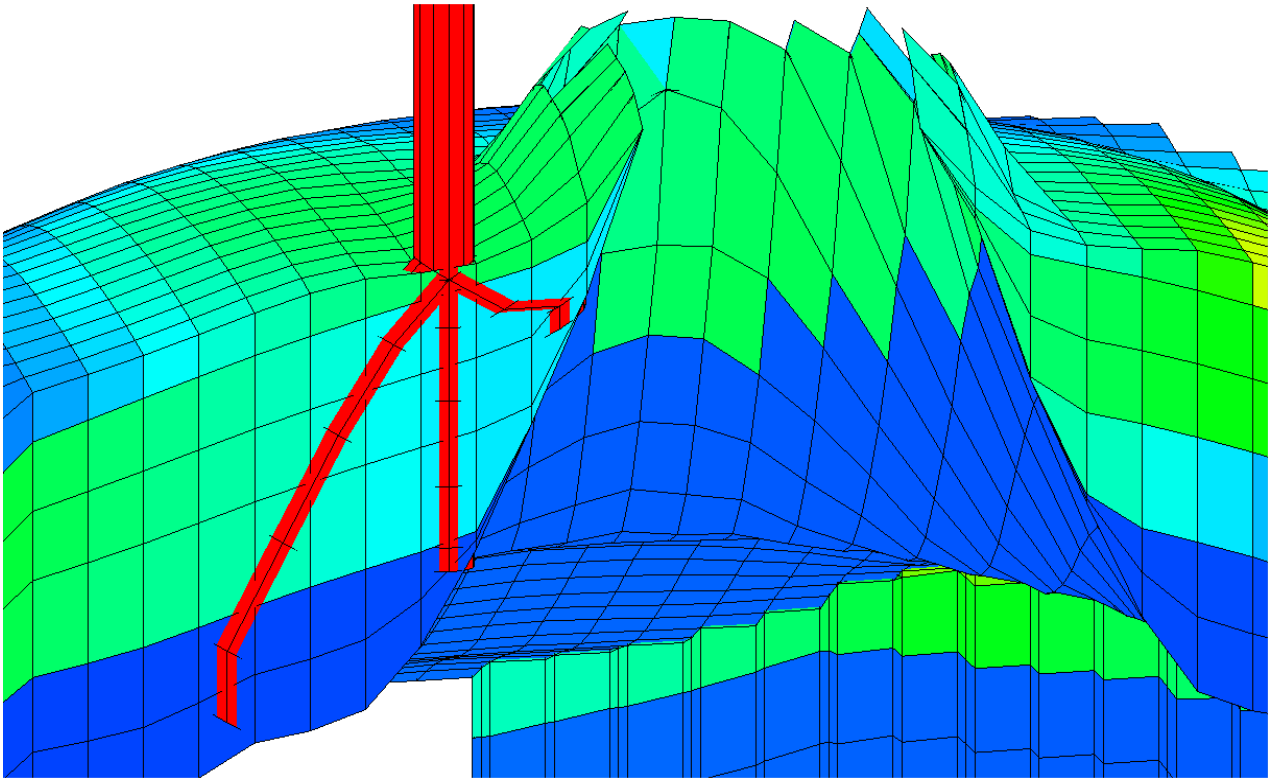


Рис. 1.9. Пример куста скважин

пример) не удается, поскольку при большом числе итераций накопление вычислительной ошибки приводит к значительному искажению приближенного решения. В основном рассматривают несколько способов построения предобуславливателей:

- Nested factorisation — приближенные покоординатные разложения матрицы на матрицу, задающую фильтрацию по вертикали (по оси Z) и по горизонтали (в плоскости XY), последняя затем приближенно факторизуется на фильтрацию по X и Y.

- Constraint Pressure Residual (CPR) — в качестве предобуславливателя фактически берется матрица, полученная при аппроксимации системы (4) неявно только по давлению (т.е. в уравнениях (48) с верхнего слоя берется только p); эту матрицу затем, в свою очередь, предобуславливают с помощью алгебраического многосеточного метода (AMG), пользуясь

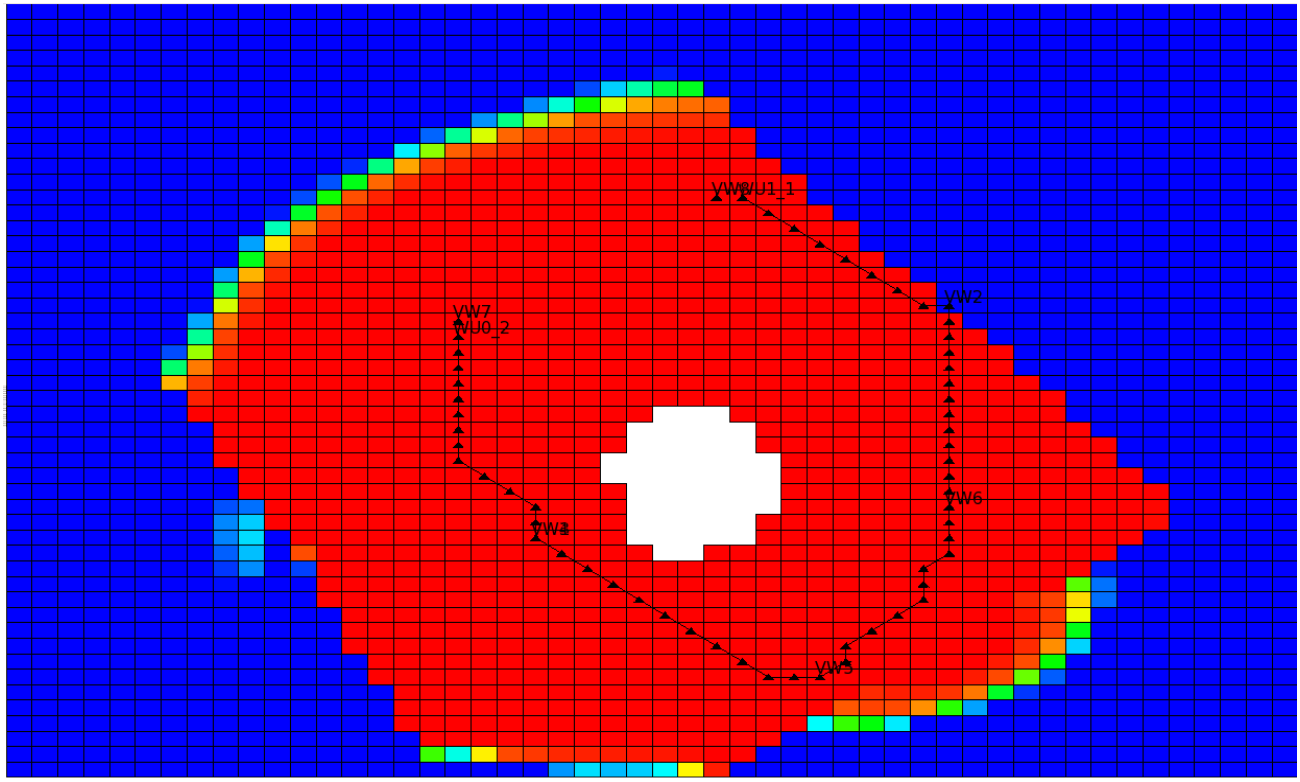


Рис. 1.10. Пример поверхностного трубопровода

ее схожестью с эллиптическими задачами.

- ILU(p) и его вариации — неполное LU разложение.

Вопрос о том, какой метод предобуславливания эффективнее, с математической точки зрения открыт.

1.5. Решение задачи на параллельных ЭВМ

Большой объем вычислительной работы, когда на каждом шаге по времени надо методом Ньютона решать нелинейную систему алгебраических уравнений, что требует решения системы линейных алгебраических уравнений (СЛАУ) на каждой итерации, делают применение высокопроизводительных параллельных ЭВМ особенно актуальным. Однако, на пути использования параллельных ЭВМ при решении задачи фильтрации встают значительные

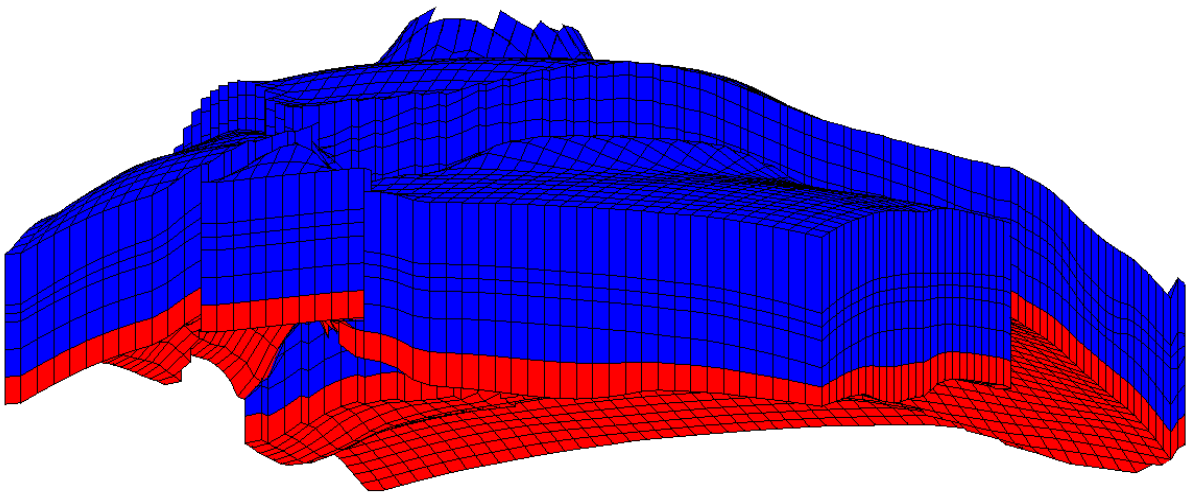


Рис. 1.11. Пример водоплавающей залежи (водонапорный горизонт подключен ко всех подошве пласта)

трудности:

- Большой шаблон матрицы СЛАУ (количество ненулевых элементов в строке) (см. 1.4.1) приводит к большому количеству связей между между частями матрицы (или вычислительной области), распределенными на разные логические процессоры. Это приводит к большому объему данных, которыми должны обмениваться вычислительные процессы на каждой итерации.

- Большие объемы обрабатываемых на каждой итерации данных делают невозможным обработку последовательных частей алгоритма (если они есть) одним логическим процессором из-за недостатка оперативной памяти. Все данные должны быть распределены между работающими процессами, что приводит к большому объему обменов между ними.

- Необходимость применения сложных предобуславливателей (см. 1.4.2), многие из которых существенно последовательные по своей природе.

- Часто невозможно отнести к одному логическому процессору целиком набор блоков, через которые проходит

  – одна скважина,

  – несколько “пересекающихся” скважин,

  – одна группа скважин, связанная поверхностными трубопроводами,

  – один водонапорный горизонт,

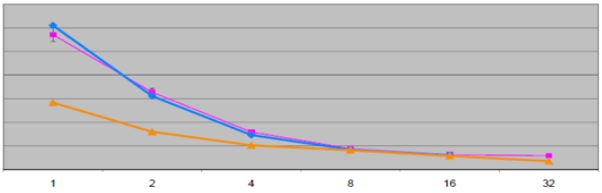
  – одна трещина ГРП при ее моделировании через набор виртуальных перфораций.

  Это приводит к тому, что алгебраические уравнения для определения параметров притока в(из) скважину или водонапорный горизонт придется решать совместно на нескольких вовлеченных логических процессорах, передавая между ними всю необходимую информацию.

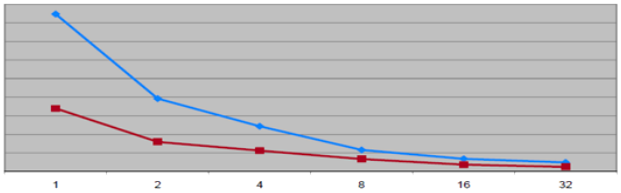
- Объем входных данных не определяет полностью сложность вычислительной работы. Получается, что при равномерном разделении входных данных между логическими процессорами вычислительная работа при этом делится неравномерно. Это происходит из-за отсутствия фильтрационных течений в ряде подобластей резервуара, причем для определения этих подобластей требуется произвести часть расчета.

Указанные причины вызывают быстрое насыщение параллельных алгоритмов, т.е. ситуации, когда добавление новых логических процессоров не приводит к сокращению времени расчета. Например, в лучших западные проектах (по их собственным данным):

- Eclipse (Schlumberger): максимальное ускорение 8–12 раз, насыщение за пределами 16 логических процессоров (см. рис. 1.12, *а*).
- Intersect (Schlumberger): максимальное ускорение 14–16 раз, насыщение за пределами 20 логических процессоров (см. рис. 1.12, *б*).



(*а*). Eclipse (Schlumberger)



(*б*). Intersect (Schlumberger)

Рис. 1.12. Пример параллельного масштабирования лучших западных проектов

1.6. Выводы к первой главе

В первой главе описана новая математическая модель техногенной трещиноватости, возникающей около скважин при проведении гидравлического разрыва пласта (ГРП) или при превышении давления закачки величины горного давления. Модель позволяет описать:

- гидравлический характер движения жидкости или газа по трещине;
- раскрытие или схлопывание трещины при повышении или понижении давления в ней;
- влияние на течение по трещине закачанного при проведении ГРП пропанта;
- эффекты разрушения пропанта от времени и выноса его потоком жидкости или газа;

- эффекты увеличения гидродинамического сопротивления течению в трещине за счет осаждения высоковязких фракций.

Эта математическая модель гидроразрыва пласта реализована в комплексе программ tNavigator, предложенном в работе. Она была опробована при расчете реальных месторождений Западной Сибири и была рекомендована к применению в Российских компаниях “ТНК-ВР” и “Газпромнефть”.

Глава 2

Блочные предобуславливатели класса ILU

При решении систем с разреженной матрицей, возникающих при аппроксимации систем дифференциальных уравнений в частных производных, используются различные форматы хранения матриц, например, Sparse или MSR [43]. Однако при таком подходе совершенно не учитывается, что при аппроксимации системы уравнений структуры ненулевых элементов каждого уравнения системы для одного блока сетки практически полностью совпадают. Именно поэтому возникла идея формализовать и исследовать блочный формат хранения разреженных матриц, в котором соответствующие элементы аппроксимации уравнений системы объединены в блоки. Например, при использовании метода конечных объемов это фактически означает перенумерацию неизвестных в системе таким образом, что сначала идут все компоненты векторов неизвестных, соответствующие первой базисной функции, затем — второй и т.д. Такой подход требует изменения формата хранения матрицы и переформулировки алгоритмов ILU разложения в терминах блоков.

Блочные варианты алгоритмов LU и ILU ранее рассматривались в литературе. Однако, они применялись к общим разреженным матрицам, а уравнения группировались в блоки ([44], [45], [46], [47], [48], [49]) либо для полного задействования кеш-памяти процессора (см., например, [4]), либо ([50], [51]) для использования идей метода разделения области. В изложенном ниже варианте для рассматриваемой системы блочный алгоритм дает помимо количественного выигрыша в скорости работы еще и качественный эффект, расширяя область применимости предобуславливателя.

2.1. Блочная форма записи матричных операций

Рассмотрим основную мотивацию, стимулировавшую исследования в области блочных алгоритмов.

Возьмем для примера задачу умножения двух заполненных $N \times N$ квадратных матриц: $C = AB$. Вычислим произведение матриц двумя способами:

1. Цикл проведем по строкам матрицы C (стандартный способ):

$$\begin{aligned} &\text{для всех} \quad m = 1, 2, \ldots, N \\ &\text{для всех} \quad i = 1, 2, \ldots, N \\ &\text{вычислять} \quad c_{mi} = \sum_{j=1}^{N} a_{mj} b_{ji} \end{aligned}$$

2. Зададимся параметром n (для простоты ниже предполагается, что N делится на n нацело, хотя это не требуется). Всякая матрица M может быть представлена как составленная из блоков:

$$M = \begin{pmatrix} M_{11} & M_{12} & \ldots & M_{1k} \\ M_{21} & M_{22} & \ldots & M_{2k} \\ \ldots & \ldots & \ddots & \ldots \\ M_{k1} & M_{k2} & \ldots & M_{kk} \end{pmatrix}$$

где M_{ij} – $n \times n$ матрица, $k = N/n$. Тогда каждый блок C_{im} произведения $C = AB$ матриц A и B может быть вычислен через блоки матриц A и B:

$$C_{im} = \sum_{j=1}^{n} A_{ij} B_{jm}$$

В вычислении произведения $n \times n$ матриц A_{ij} и B_{jm} участвует только подмножество из n^2 элементов матриц A и B. При небольшом n это подмножество полностью поместится в кэш памяти процессора и каждое слагаемое последней суммы будет вычислено максимально быстро.

Исходный текст соответствующих программ приведен в [4]. При сравнении времени работы этих вариантов на современных процессорах оказалось, что блочный вариант работает в 8–10 раз быстрее! Реальный прирост производительности пропорционален отношению фактических тактовых частот процессора и оперативной памяти. Получается, что

- процессор в стандартном алгоритме работал только 1/8–1/10 своей производительности, все остальное время он ожидал данных из памяти;
- стандартный алгоритм бессмысленно распараллеливать: добавление к одному простаивающему на 90% процессору других вычислительных мощностей вряд ли что-то изменит;
- блочный алгоритм, напротив, идеален для распараллеливания, так как процессор большую часть времени работает со своим кэшем, не мешая другим процессорам обращаться к памяти.

Поэтому естественным было применить блочный алгоритм для решения задачи фильтрации.

2.2. Описание блочного варианта алгоритма построения ILU разложения

Мы не будем здесь подробно останавливаться на описании классического алгоритма построения ILU разложения для разреженных матриц, а также на описании его модификаций — алгоритмов ILU(1) и ILUT(p, τ), так как их описание можно найти, например, в [52], [53], [43], [54], [1] Отметим лишь, что при проведении численных экспериментов использовался IKJ алгоритм.

Рассмотрим блочный вариант построения ILU разложения.

Пусть A — блочная матрица размерности n:

$$A = \begin{pmatrix} A_{11} & A_{12} & \dots & A_{1m} \\ A_{21} & A_{22} & \dots & A_{2m} \\ \dots & \dots & \dots & \dots \\ A_{m1} & A_{m2} & \dots & A_{mm} \end{pmatrix}, \tag{2.1}$$

где A_{ij} — матрицы размерности d и $dm = n$. Матрицы L и U также имеют блочную структуру.

Требуется найти блочное ILU разложение, т.е. матрицы L и U блочного вида. Матрицы L и U хранятся на месте матрицы вида (2.1).

Элементы матриц L и U вычисляются по следующим формулам:

$$\begin{aligned} L_{ii} &= A_{ii} - \sum_{j=1}^{i-1} L_{ij}U_{ji}, \\ L_{ji} &= A_{ji} - \sum_{k=1}^{i-1} L_{jk}U_{ki}, \quad i < j, \\ U_{ij} &= L_{ii}^{-1}(A_{ij} - \sum_{k=1}^{i-1} L_{ik}U_{kj}), \quad i < j. \end{aligned}$$

Рассмотрим алгоритм вычисления элементов матриц L и U. Для этого введем некоторые обозначения. Пусть

$$\begin{aligned} P &= \{(i,j) \mid A_{ij} \neq 0;\ 1 \leqslant i,j \leqslant m\}, \\ P' &= \{(i,j,k,l) \mid (A_{ij})_{kl} \neq 0; \\ &\qquad 1 \leqslant i,j \leqslant m,\ 1 \leqslant k,l \leqslant d\}, \end{aligned}$$

т.е. P — множество ненулевых элементов при блочном представлении разреженой матрицы, а P' — при обычном представлении матрицы. Заметим, что при полном заполнении блоков матрицы мощности множеств соотносятся как $|P'| = d^2|P|$.

Блочный вариант IKJ алгоритма нахождения ILU разложения. Будем

производить вычисления следующим образом:

для всех $i = 2, \ldots, m$:

для всех $k = 1, \ldots, i-1$, таких что $(i,k) \in P$:

$A_{ik} = A_{kk}^{-1} A_{ik}$;

для всех $j = k+1, \ldots, m$, таких что $(i,j) \in P$:

$A_{ij} = A_{ij} - A_{ik} A_{kj}$.

Так как при вычислениях необходимо находить обратную к диагональному блоку A матрицу, то описанный алгоритм применим в том случае, когда определители всех диагональных блоков исходной матрицы не равны нулю.

Блочный вариант IKJ алгоритма нахождения ILU(1) разложения. Этот алгоритм получается на основе ILU алгоритма расширением каждой строки шаблона ненулевых элементов для матриц L и U не более чем на 1 элемент [43]. Стратегия добавления элементов в блочной версии абсолютно идентична скалярной версии алгоритма. Стоит лишь отметить, что так как при блочном подходе все элементы матрицы — это блоки, то при добавлении нового элемента в шаблон ненулевых элементов в матрицу добавляется целый блок.

Блочный вариант IKJ алгоритма нахождения ILUT(p, τ) разложения. Этот алгоритм также получается на основе ILU алгоритма расширением шаблона ненулевых элементов для матриц L и U [52], [43]. Стратегия добавления элементов заключается в том, что в качестве кандидатов необходимо рассматривать лишь те элементы, относительная норма которых превышает τ, и при этом расширять каждую строку шаблона ненулевых элементов не более чем на p элементов. В скалярной версии алгоритма в качестве нормы используется модуль числа, стоящего на соответствующей позиции. В блочной же версии есть больше возможности для маневра: в качестве нормы блока можно выбирать любую матричную норму.

2.3. Блочная форма хранения разреженных матриц

В связи с сильной разреженностью структуры матрицы хранятся лишь ненулевые элементы. Оптимальным по объему требуемой памяти является формат Modified Sparse Row (MSR) [43].

Скалярный вариант. При использовании такого формата хранения матриц хранятся сначала все диагональные элементы (n штук), потом один пустой элемент (необходимость его наличия станет понятна позже), а затем все ненулевые внедиагональные элементы в том порядке, в котором они следуют в исходной матрице. Соответственно в таком случае каждая строка ненулевых элементов имеет свою длину, поэтому необходимо знать начальную и конечную позицию. Так как конец i-той строки является началом $(i+1)$-й, то необходимо хранить $n+1$ число. Они записываются в начало массива ненулевых элементов, а затем следуют номера столбцов элементов, стоящих на соответствующей позиции в массиве ненулевых элементов. Таким образом, пустой элемент в массиве ненулевых элементов нужен для того, чтобы, начиная с $(n+2)$-го элемента, индексация обоих массивов совпадала.

Блочные варианты. Блочный вариант полностью аналогичен скалярному хранению матрицы в формате MSR, за исключением того факта, что в матрице ненулевых элементов вместо скалярных величин хранятся блоки размерности d. Каждый блок хранится как вектор, т.е. сначала лежит первая строка блока, за последним элементом первой строки — вторая и т.д. Таким образом обеспечивается непрерывность обращения к памяти внутри одного блока.

Следствием использования блочного варианта является значительное уменьшение объема памяти, необходимого для хранения матрицы, так как при таком подходе на каждый блок матрицы (т.е. на d^2 величин) дополнительно хранится лишь одно число, соответствующее номеру столбца, в котором стоит блок (напомним, что в скалярной версии каждому элементу матрицы сопоставляется

номер его столбца).

2.4. Свойства блочного представления разреженных матриц

Блочная форма представления матрицы для задачи, полученной при аппроксимации **системы** дифференциальных уравнений является естественной.

Сравним с искусственной группировкой строк матрицы, используемой для лучшего использования кэш памяти процессора. Возьмем единичную матрицу 4×4 и переставим в ней местами 2-ю и 3-ю строки. Затем будем рассматривать ее как блочную с размером блока 2:

$$\left(\begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{array}\right) \Longrightarrow \left(\begin{array}{cc|cc} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \hline 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{array}\right)$$

В полученной матрице все блоки вырождены и блочный вариант LU разложения не применим. Даже если не брать во внимание такие экстремальные случаи, то блочные алгоритмы могут оказаться менее стабильными, чем их неблочные варианты, см. [45]. В случае матрицы для задачи, полученной при аппроксимации **системы** дифференциальных уравнений, имеет место противоположная ситуация: при традиционной нумерации на диагонали в процессе построения неполного LU разложения могут оказаться нулевые элементы, это соответствует ситуации отсутствия фазы в данной ячейки сетки. При блочной нумерации невырожденность блока на диагонали обеспечивается законами сохранения: хотя бы одна из фаз подвижна всегда.

Перечислим основные свойства построенного блочного представления:

- **Расширяется класс матриц**, для которых применимы ILU предобуслав-

ливатели. Действительно, при обычном (скалярном) ILU разложении в ходе разложения на диагонали матрицы не должно появляться нулей, а при блочном ILU разложении требования к матрице снижаются: диагональные блоки должны иметь ненулевой определитель.

- **Происходит избирательное увеличение ширины шаблона матрицы**. Так как блоки матрицы хранятся целиком (даже если там есть нули), то фактически обычный метод ILU разложения немного видоизменяется и в чем-то походит на ILU(p) метод. Эксперименты показывают (см. ниже), что уже ILU(1) не дает ускорения, в то время, как переход к блочному варианту дает выигрыш даже в числе итераций. Расширение ленты приводит к тому, что получаемое ILU разложение точнее приближает исходную матрицу, и, следовательно, улучшаются характеристики сходимости алгоритмов с использованием ILU разложения.

- **Появляется ускорение за счет использования кэш-памяти**. Вследствие того, что везде в блочном алгоритме мы оперируем блочными матрицами размерности d, происходит меньшее число обращений к данным на медленных носителях (оперативная память). В начале работы с новой строкой матрицы она вся попадает в кэш, и дальше идет работа уже только с кэш-памятью, а вследствие использования MSR формата хранения матрицы обеспечивается непрерывность обращения к памяти от первых до последних блоков матрицы (без «прыжков» через сегменты памяти).

- **Уменьшается требуемый объем памяти** для хранения матрицы. Это также сказывается и на времени работы, так как повышается эффективность кеширования используемой памяти.

- **Увеличиваются возможности для масштабирования на параллель-**

ных ЭВМ за счет увеличения использования кэша (процессоры реже обращаются к общей шине памяти).

2.5. Численные эксперименты

Для численных экспериментов взяты матрицы системы линейных уравнений с матрицей (65), полученные при дискретизации уравнений (4) с данными для реальных месторождений на разных шагах по времени. Для решения системы линейных уравнений используется итерационный алгоритм BCGS, в котором в качестве предобуславливателя берется одно из рассматриваемых разложений класса ILU. В качестве критерия окончания итерационного процесса выбрано условие падения относительной невязки на 6 порядков.

В таблице 2.1 приведены некоторые характеристики матриц:

- d — размер блоков,
- $rows$ — количество блочных строк матрицы (соответственно, количество скалярных строк равно $rows \cdot d$),
- $blocks$ — количество хранящихся блоков (количество скалярных величин равно $blocks \cdot d^2$),
- $nonzero$ — количество ненулевых скалярных элементов;
- max — максимальное количество ненулевых блоков в строке матрицы;
- $width$ — максимальное удаление ненулевого блока матрицы относительно диагонали (ширина ленты матрицы);
- k_{sk} — коэффициент кососимметричности, вычисленный по формуле:

$$k_{sk} = \max_{i=1,\ldots,rows} \sum_{j=1}^{rows} \|A_{ij} - A_{ji}\|_\infty,$$

	rows	*d*	*blocks*	*nonzero*	*max*	*width*	k_{sk}
1	9000	3	60380	430352	8	360	$2{,}2 \cdot 10^{3}$
2	31923	2	158611	634444	5	182	$2{,}4 \cdot 10^{0}$
3	31923	2	158611	634444	5	182	$2{,}5 \cdot 10^{0}$
4	38804	3	230054	1526530	8	536	$1{,}5 \cdot 10^{4}$
5	102825	2	689595	2579928	48	1643	$4{,}6 \cdot 10^{6}$
6	159174	3	819146	5239880	24	6332	$1{,}2 \cdot 10^{12}$
7	126648	3	827884	7450750	7	2461	$1{,}4 \cdot 10^{4}$
8	141900	2	991880	3967422	47	1290	$2{,}7 \cdot 10^{3}$
9	181602	3	1002582	6547768	19	1912	$1{,}8 \cdot 10^{4}$
10	207373	2	1186036	4111448	7	755	$1{,}7 \cdot 10^{1}$
11	199497	2	1222461	4507842	12	1947	$1{,}2 \cdot 10^{6}$
12	196472	3	1292806	11634998	7	2927	$1{,}4 \cdot 10^{4}$
13	198164	3	1312246	11675938	44	2927	$6{,}1 \cdot 10^{4}$
14	239901	2	1454483	5053844	18	1871	$8{,}1 \cdot 10^{5}$
15	234135	3	1609689	13615219	7	2544	$1{,}6 \cdot 10^{4}$
16	282543	2	1783625	7134500	7	879	$5{,}9 \cdot 10^{3}$
17	282543	2	1791975	7165156	7	879	$9{,}7 \cdot 10^{3}$
18	379873	2	1905447	5077020	7	3139	$3{,}1 \cdot 10^{5}$
19	359017	3	2320305	14260347	7	3085	$7{,}5 \cdot 10^{2}$
20	359017	3	2324285	14296969	30	3085	$7{,}4 \cdot 10^{2}$
21	399634	3	2740012	19315408	69	7521	$8{,}7 \cdot 10^{5}$
22	464624	3	2979682	20828280	7	2199	$1{,}1 \cdot 10^{5}$
23	464624	3	2980626	21307098	16	2199	$8{,}2 \cdot 10^{5}$
24	1317828	2	8485552	28213300	65	6082	$8{,}1 \cdot 10^{6}$
25	2307383	3	12218699	99472731	7	8172	$7{,}6 \cdot 10^{4}$
26	2058210	3	14309350	71697212	26	12474	$1{,}2 \cdot 10^{3}$
27	2431789	2	15818315	56348856	48	8958	$1{,}3 \cdot 10^{6}$

Таблица 2.1. Характеристики матриц для тестирования блочного ILU

где $\|\cdot\|_\infty$ — строчная норма матрицы.

Для проведения численных экспериментов матрицы, свойства которых приведены в табл. 2.1, были сохранены в трех форматах:

1. блочный MSR формат;

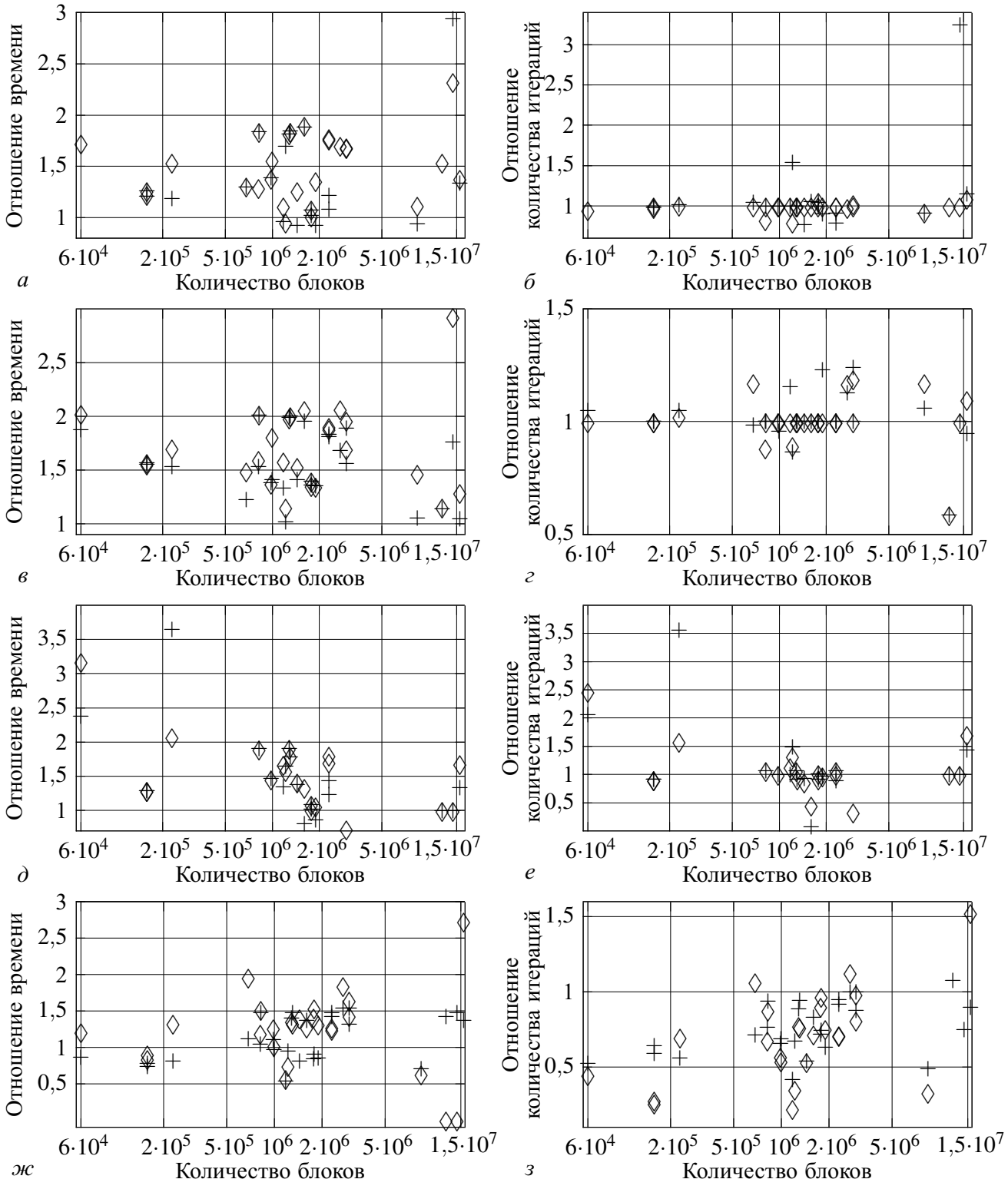
2. скалярный MSR формат: он ничем не отличается от блочного MSR формата, кроме организации хранения матрицы;

3. скалярный MSR формат с выбрасыванием всех нулевых элементов: он получен из скалярного MSR формата в результате выбрасывания из шаблона матрицы всех нулевых элементов.

На рис. 2.1 приведено соотношение времени работы и количества итераций, необходимых для решения систем линейных уравнений с использованием ILU, ILU(1), ILUT(5,10^{-3}) разложений для всех трех форматов матриц.

Сравним сначала результаты, полученные на матрицах блочного и скалярного формата (на рис. 2.1,*а–е* значком «◊» отмечено время и количество итераций, необходимых для получения решения с использованием скалярного формата, относительно времени и количества итераций, необходимых для получения решения с использованием блочного формата).

На рис. 2.1,*б* видно, что качественные характеристики итерационных алгоритмов с использованием блочной и скалярной реализации ILU разложения практически не различаются: количество итераций на некоторых матрицах отличаются лишь вследствие различий в накоплении вычислительной погрешности. Использование ILU(1) и ILUT разложений уже дает качественные отличия — во многих случаях (матрицы 5, 21, 23, 24, 27 для ILU(1) и 1, 4, 7, 11, 27 для ILUT; на рис. 1,*г,е* точки, соответствующие этим матрицам, лежат ниже прямой $y = 1$) количество итераций уменьшилось вследствие того, что в блочной версии к матрице присоединяются не скалярные величины, а целые блоки. По времени работы алгоритмов (рис. 2.1,*а,в,д*) однозначно видны преимущества блочной реализации, причем независимо от используемого для разложения матрицы алгоритма: при размере блока 2 время, затрачиваемое на решение системы линейных уравнений, уменьшается на 10–20 % (например, матрицы 2, 3) а при размере блока 3 на некоторых матрицах время работы



а,б — отношение времени и количества итераций скалярного варианта (◊) и скалярного варианта с выбрасыванием нулевых элементов (+) к времени и количеству итераций блочного варианта ILU разложения; *в,г* — аналогичный график для ILU(1) разложения; *д,е* — аналогичный график для ILUT(5,10^{-3}) разложения; *ж,з* — отношение времени и количества итераций блочного варианта ILU(1) разложения (+) и блочного варианта ILU(1) разложения (◊) к времени и количеству итераций блочного варианта ILU разложения.

Рис. 2.1. Результаты численных экспериментов для ILU предобуславливателей

уменьшается более чем в 2 раза (например, матрица 26)! Также на примере ILUT алгоритма видно, что при блочной организации хранения матрицы требуется меньший объем памяти (на матрицах 5, 9, 21, 23–26 результаты не были получены вследствие того, что алгоритму не хватало имеющейся памяти).

Теперь рассмотрим результаты, полученные на матрицах блочного и скалярного формата с выбрасыванием нулевых элементов (на рис. 2.1,*а–е* значком «+» отмечено время и количество итераций, необходимых для получения решения с использованием скалярного формата с выбрасыванием нулевых элементов, относительно времени и количества итераций, необходимых для получения решения с использованием блочного формата). Эти эксперименты показывают, что увеличение шаблона матрицы, производящееся в блочной версии, существенно влияет на качество ILU разложения и, как следствие, на сходимость алгоритма. Для многих матриц в такой скалярной реализации итерационный алгоритм вообще не сошелся (например, для ILU разложения это матрицы 6, 9, 21–23, 25).

Также на рис. 2.1,*ж,з* сравниваются блочные версии ILU, ILU(1) и ILUT разложений. Практически на всех матрицах вследствие увеличения шаблона ненулевых элементов алгоритмы с использованием ILU(1) и ILUT разложений сходятся за меньшее число итераций, нежели алгоритмы, использующие ILU разложение (рис. 2.1,*з*). Но время, необходимое для построения разложения и проведения каждой итерации при использовании ILU алгоритма, оказывается существенно меньше, и поэтому по общему времени работы в среднем ILU алгоритм оказывается наилучшим (рис. 2.1,*ж*).

2.6. Проверка предобуславливателя на матрицах, полученных при многоточечной аппроксимации

Рассмотренные в разделе 2.5 матрицы получены при двухточечной аппроксимации (см., например [32]) потока каждой из фаз через грань между соприкасающимися блоками сетки. Известно, что такая аппроксимация приводит к большой погрешности, если поток не ортогонален общей грани блоков ([55]) (например, при диагональном тензоре абсолютной проницаемости если грани блоков сетки не ортогональны координатным осям), а также к так называемому «ориентационному эффекту» ([40]), когда мгновенное значение потока в пространстве зависит от выбора сетки.

В работах [56], [57], [58] были предложены многоточечные методы (базисов связей, O-метод, U-метод), которые для вычисления потока через грань между двумя блоками сетки привлекают значения в соседних к ним блоках. Эти методы хорошо приближают поток, не ортогональный грани блока, но не решают проблему ориентационного эффекта и добавляют еще одну проблему — возникновение нефизичных потоков ([59], [60]) из блока с меньшим давлением в блок с большим давлением при сильной неортогональности граней координатным осям и сильной анизотропии тензора абсолютной проницаемости.

Ниже рассмотрен новый метод многоточечной аппроксимации — метод подсетки — лишенный указанных недостатков. На большом количестве реальных наборов данных проведено сравнение предложенного метода с другими методами многоточечной аппроксимации и двухточечным методом. В этих численных экспериментах нас прежде всего будет интересовать поведение предложенного блочного предобуславливателя на значительно более сложных матрицах, получающихся в результате многоточечной аппроксимации.

Общий вид аппроксимируемых уравнений. После использования неявной аппроксимации по времени уравнения полудискретной системы (48) могут быть представлены в виде

$$\Phi(p,N) = \nabla(p + C(p,N)) + D(p,N)\nabla g, \quad \operatorname{div}(AB(p,N)\Phi(p,N)) + Q(p,N) = f. \tag{2.2}$$

Здесь

- $A = A(x,y,z)$ — известная матричная функция размера 3×3, содержащая тензор абсолютной проницаемости $\boldsymbol{k}$,
- $B = B(x,y,z,p,N)$ — известная функция, определяемая физическими свойствами фаз,
- $C = C(x,y,z,p,N)$ — известная функция, включающая капиллярные давления фаз,
- $D = D(x,y,z,p,N)$, $g = g(x,y,z)$ — известные функции, определяющие гравитационное влияние,
- $Q = Q(x,y,z,p,N)$ — известная функция (вида δ-функции Дирака), содержащая аппроксимацию скважин,
- $f = f(x,y,z)$ — известная функция, определяемая данными с предыдущего слоя по времени,
- $p = p(x,y,z)$, $N = \{N_c(x,y,z) \,|\, c = 1,\ldots,n_c\}$ — искомые функции.

Пространственная аппроксимация. Для пространственной аппроксимации уравнений такого типа используется метод конечных объемов (см. раздел 2.2). Уравнения (2.2) интегрируются по каждому блоку сетки V_i и используется

формула Гаусса-Остроградского (49):

$$\int\limits_{\partial V_i} (AB(p,N)\Phi(p,N),\boldsymbol{n}')\,ds = \int\limits_{V_i} (f - Q(p,N))\,dv,$$

где $\boldsymbol{n}'$ — вектор внешней нормали к блоку V_i.

Рассмотрим произвольную грань E сетки. Если она является внешней гранью для расчетной области, т.е. существует только один блок V_i сетки, содержащий E, то положим $\boldsymbol{n}$ равным вектору внешней нормали к V_i и $\delta_{iE} = 1$. Если E является внутренней гранью, т.е. существуют такие блоки V_{i_1} и V_{i_2} сетки, что $E = V_{i_1} \cap V_{i_2}$ и $i_1 < i_2$, то положим $\boldsymbol{n}$ равным вектору внешней нормали к V_{i_1}, $\delta_{i_1 E} = 1$ и $\delta_{i_2 E} = -1$. Будем обозначать индекс i_1 через $E-$, а индекс i_2 через $E+$. Пусть

$$I_E(p,N) = \int\limits_E (AB(p,N)\Phi(p,N),\boldsymbol{n})\,ds.$$

Тогда при переходе к дискретной задаче над пространством P^h кусочно-постоянных на блоке сетки функций получим нелинейную систему уравнений относительно неизвестных $p^h, N^h \in P^h$:

$$\sum_{E \in \partial V_i} \delta_{iE} I_E^h(p^h,N^h) + Q(p_i^h,N_i^h) = v_i f_i, \tag{2.3}$$

где I_E^h — дискретный аналог оператора I_E, v_i — объем блока V_i, f_i — значение f в блоке V_i. Эта система может быть решена методом Ньютона (см. раздел 2.5). Для построения системы нам необходимо определить оператор I_E^h, а точнее, указать алгоритм вычисления $I_E^h(p^h,N^h)$, $\dfrac{\partial I_E^h}{\partial p_j}(p^h,N^h)$ и $\dfrac{\partial I_E^h}{\partial N_{c,j}^h}(p^h,N^h)$ для всех j,c.

Функция $B(p,N)$, исходя из физического смысла, аппроксимируется «по потоку»:

$$I_E^h(p^h,N^h) = \begin{cases} B(p_{E+},N_{E+})\tilde{I}_E^h(p^h,N^h), & \text{если } \tilde{I}_E^h(p^h,N^h) \geqslant 0, \\ B(p_{E-},N_{E-})\tilde{I}_E^h(p^h,N^h), & \text{если } \tilde{I}_E^h(p^h,N^h) < 0, \end{cases} \tag{2.4}$$

где $\tilde{I}_E^h(p^h,N^h)$ — дискретный аналог оператора $\int\limits_E (A\Phi(p,N),\boldsymbol{n})\,ds$. Будем обозначать через $\tilde{E}$ блок для $B(p,N)$, выбранный в (2.4).

Полагают, что

$$\tilde{I}_E^h(p^h,N^h) = \sum_{k=1}^{m} t_k \left(p_{i_k} + C(p_{i_k},N_{i_k}) + D^h(p^h,N^h) g_{i_k} \right). \tag{2.5}$$

Индексы i_k и коэффициенты t_k, $k = 1,\ldots,m$ определяют действие дискретного аналога оператора $\int\limits_E (A\nabla\cdot,\boldsymbol{n})\,ds$ на сеточную функцию. В случае, когда $m = 2$ для всех граней E, метод аппроксимации называется двухточечным (TPFA, Two Point Flux Approximation), а когда существуют грани с $m > 2$ — многоточечным методом (MPFA, Multipoint Flux Approximation). Значения t_k, как правило, носят название коэффициентов проводимости (transmissibility coefficients) для грани E. Функция $D(p,N)$ является с точностью до константы плотностью фазы и аппроксимируется как

$$D^h(p^h,N^h) = \frac{D(p_{E-},N_{E-}) + D(p_{E+},N_{E+})}{2}. \tag{2.6}$$

Объединяя формулы (2.4)–(2.6), получим выражения для оператора I_E^h и его производных:

$$I_E^h(p^h,N^h) = B(p_{\tilde{E}},N_{\tilde{E}}) \sum_{k=1}^{m} t_k \left(p_{i_k} + C(p_{i_k},N_{i_k}) + \frac{D(p_{E-},N_{E-}) + D(p_{E+},N_{E+})}{2} g_{i_k} \right),$$

$$\begin{aligned}
\frac{\partial I_E^h}{\partial p_j}(p^h,N^h) = {} & \delta_{\tilde{E}}^j \frac{\partial B}{\partial p}(p_{\tilde{E}},N_{\tilde{E}}) \sum_{k=1}^{m} t_k \left(p_{i_k} + C(p_{i_k},N_{i_k}) \right. \\
& \left. + \frac{D(p_{E-},N_{E-}) + D(p_{E+},N_{E+})}{2} g_{i_k} \right) \\
& + B(p_{\tilde{E}},N_{\tilde{E}}) \sum_{k=1}^{m} t_k \left(\delta_{i_k}^j \left(1 + \frac{\partial C}{\partial p}(p_{i_k},N_{i_k}) \right) \right. \\
& \left. + \frac{\delta_{E-}^j \frac{\partial D}{\partial p}(p_{E-},N_{E-}) + \delta_{E+}^j \frac{\partial D}{\partial p}(p_{E+},N_{E+})}{2} g_{i_k} \right),
\end{aligned}$$

$$\frac{\partial I_E^h}{\partial N_{c,j}}(p^h,N^h) = \delta_{\tilde{E}}^j \frac{\partial B}{\partial N_c}(p_{\tilde{E}},N_{\tilde{E}}) \sum_{k=1}^{m} t_k \left(p_{i_k} + C(p_{i_k},N_{i_k}) + \frac{D(p_{E-},N_{E-}) + D(p_{E+},N_{E+})}{2} g_{i_k}\right) + B(p_{\tilde{E}},N_{\tilde{E}}) \sum_{k=1}^{m} t_k \left(\delta_{i_k}^j \frac{\partial C}{\partial N_c}(p_{i_k},N_{i_k}) + \frac{\delta_{E-}^j \frac{\partial D}{\partial N_c}(p_{E-},N_{E-}) + \delta_{E+}^j \frac{\partial D}{\partial N_c}(p_{E+},N_{E+})}{2} g_{i_k}\right).$$

Итак, для окончательного определения пространственной аппроксимации для уравнений (2.2) требуется для каждой грани E сетки предъявить алгоритм вычисления индексов i_k и коэффициентов проводимости t_k в соотношении (2.5).

Двухточечная и многоточечная аппроксимации. При аппроксимации по двум точкам $m = 2$, $i_1 = E-$, $i_2 = E+$, $t_1 = -T$, $t_2 = T$. Значение T называется базовым коэффициентом проводимости грани, является положительным числом и равно

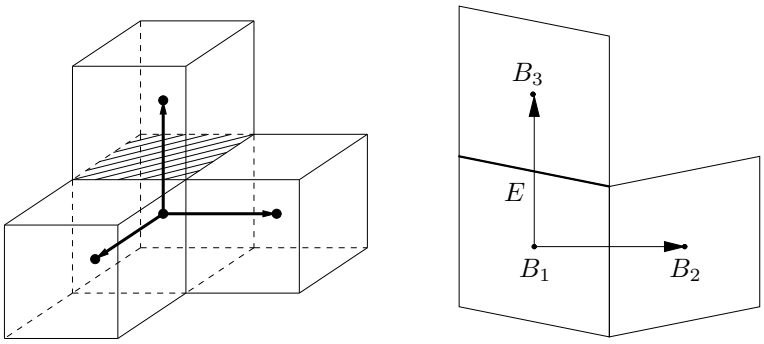
$$T = \mathrm{mes}(E) \frac{A_{E-}^{r,r} A_{E+}^{r,r}}{d_{E-} A_{E+}^{r,r} + d_{E+} A_{E-}^{r,r}},$$

где d_j — расстояние от центра масс блока с индексом j, $j = E-, E+$ и гранью E, $A^{r,r}$ — диагональный элемент тензора A с индексами (r,r), а индекс r определяется осью, которой квазиортогональна грань E (равно $1,2,3$ для осей Ox, Oy, Oz соответственно). Гармоническое осреднение тензора A выбрано для того, чтобы исключить переток в том случае, когда $A_{E-}^{r,r}$ или $A_{E+}^{r,r}$ равно нулю.

Основной недостаток этого метода заключается в том, что порядок аппроксимации оператора $(A\nabla\cdot, \boldsymbol{n})$ равен $O(h)$ только в случае, когда сетка структурированная и ортогональная, а тензор A диагональный. В остальных случаях порядок равен $O(1)$, так как по двум точкам можно приблизить лишь одну

компоненту потока. Таким образом, качество аппроксимации падает тем больше, чем более неортогональны прямая, соединяющая центры блоков, и общая между этими блоками грань. Известно, что при отличии указанного угла от 90° более, чем на 10°, погрешность аппроксимации потока становится значительной ([40]).

Для исправления ситуации были разработаны методы многоточечной аппроксимации. Самый простой из них — метод базисов связей ([56], доработан для использования в трехмерной области на произвольных сетках) — основан на более точном приближении градиента. Сначала он вычисляется в центре каждого из блоков $E-$ и $E+$, затем осредняется с весами d_{E-} и d_{E+} на грань. Для приближения градиента в центре блока рассматриваются все базисы из векторов, соединяющих центр этого блока с центрами соседних, такие, чтобы один из векторов соединял $E-$ и $E+$ (см. рис. 2.2, *а*); затем градиент восстанавливается по проекциям на каждый базисный вектор. Тензор A и интеграл по грани применяются так же, как в TPFA.



(*а*). Экземпляр базиса для метода базисов связей

(*б*). Случай нефизичного течения

Рис. 2.2. Метод базисов связей

Порядок аппроксимации оператора $(A\nabla\cdot, \boldsymbol{n})$ для этого метода равен $O(h)$. Тем не менее, такой способ приближения потока содержит ряд недостатков. Самый главный из них — появление нефизичного течения в некоторых случа-

ях. Рассмотрим для примера систему из трех блоков в двумерном случае (см. рис. 2.2, *б*).

Пусть тензор проницаемости единичный и давление в блоке B_i равно p_i. Приближение градиента в центре B_1 будет равно $\left(\frac{p_2-p_1}{d_{12}}, \frac{p_3-p_1}{d_{13}}\right)$. Нормаль к грани E, внешняя по отношению к B_1, равна $\frac{(\alpha,1)}{\sqrt{\alpha^2+1}}$, $\alpha > 0$. Это означает, что приближение потока, построенное по этому базису равно

$$\frac{1}{\sqrt{\alpha^2+1}}\left(\alpha\frac{p_2-p_1}{d_{12}} + \frac{p_3-p_1}{d_{13}}\right).$$

Если окажется, что $p_3 > p_1$ и $p_3 - p_1 < \frac{\alpha d_{13}}{d_{12}}(p_1 - p_2)$, то поток станет отрицательным, т.е. будет переток из B_1 в B_3 (хотя $p_3 > p_1 > p_2$), увеличивающийся тем сильнее, чем меньше давление p_2. В анизотропной среде для блоков, размеры которых отличаются в десятки раз по различным направлениям, этот эффект проявляется еще сильнее. Основная причина такого поведения заключается в том, что положение центров блоков слабо связано с направлением нормалей к граням.

Во многих пакетах решения задачи фильтрации используются такие многоточечные аппроксимации, как O-метод, U-метод и другие ([57], [58]). Они обычно направлены на приближение оператора $A\nabla$, поскольку такой подход лучше описывает физическое явление. К сожалению, они не лишены проблемы возникновения нефизичного течения. Это во многом обусловлено недостаточным качеством приближения геометрии сетки.

Метод подсетки. Разработанный метод подсетки позволяет решить описанную выше проблему. Прежде всего, для каждой вершины сетки строится так называемая область взаимодействия. В двумерном случае для каждого блока, содержащего вершину, нужно соединить центр масс с серединой каждой стороны, содержащей вершину, получив тем самым область вокруг вершины (см. рис. 2.3). В трехмерном случае аналогичная область является объединением

всевозможных тетраэдров $PMEV$, где V — рассматриваемая вершина, E — середина содержащего ее ребра, M — центр масс содержащей его грани, P — центр масс содержащего ее блока. В равномерном случае количество таких тетраэдров для внутренней вершины сетки равно 48.

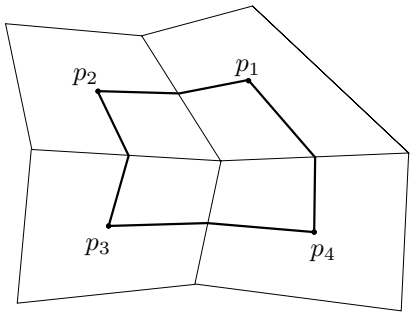


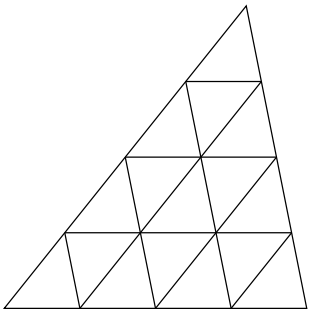
Рис. 2.3. Область взаимодействия в двумерном случае

После определения области взаимодействия, она должна быть точно аппроксимирована сеткой из тетраэдров. Для этого предлагается задать единый параметр разбиения n и разбить каждый тетраэдр $PMEV$ на n^3 равных по объему тетраэдров так, чтобы каждая грань исходного тетраэдра была равномерна разбита на треугольники (см. рис. 2.4, *а*). Если обозначить через

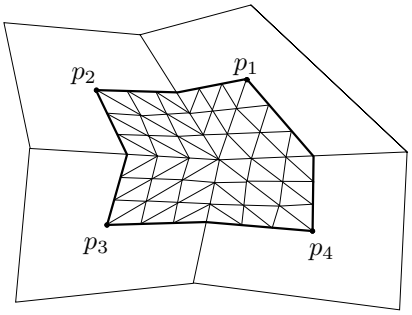
$$f_{ijk}(a,b,c) = \left(\frac{i+a}{n}, \frac{j+b}{n}, \frac{k+c}{n}, \frac{n-i-j-k-a-b-c}{n}\right)$$

барицентрические координаты узла разбиения относительно исходного тетраэдра, то тетраэдры, на которые он разбит, определяются следующими четверками точек:

$$\begin{aligned}
&f_{ijk}(0,0,0), f_{ijk}(1,0,0), f_{ijk}(0,1,0), f_{ijk}(0,0,1), && \forall i,j,k : i+j+k < n,\\
&f_{ijk}(1,0,0), f_{ijk}(0,1,1), f_{ijk}(0,0,1), f_{ijk}(0,1,0), && \forall i,j,k : i+j+k < n-1,\\
&f_{ijk}(1,0,0), f_{ijk}(0,1,1), f_{ijk}(0,1,0), f_{ijk}(1,1,0), && \forall i,j,k : i+j+k < n-1,\\
&f_{ijk}(1,0,0), f_{ijk}(0,1,1), f_{ijk}(1,1,0), f_{ijk}(1,0,1), && \forall i,j,k : i+j+k < n-1,\\
&f_{ijk}(1,0,0), f_{ijk}(0,1,1), f_{ijk}(1,0,1), f_{ijk}(0,0,1), && \forall i,j,k : i+j+k < n-1,\\
&f_{ijk}(0,1,1), f_{ijk}(1,0,1), f_{ijk}(1,1,0), f_{ijk}(1,1,1), && \forall i,j,k : i+j+k < n-2.
\end{aligned}$$



(*а*). Грань тетраэдра при разбиении с $n = 4$



(*б*). Область взаимодействия с подсеткой

Рис. 2.4. Построение метода подсетки

В силу того, что грани тетраэдров $PMEV$ разбиты одинаково, вся область взаимодействия оказывается разбита структурированной сеткой из тетраэдров, называемой подсеткой. В равномерном случае количество узлов подсетки для внутренней вершины сетки равно $(2n+1)^3$, а количество тетраэдров подсетки — $48n^3$. На рис. 2.4, *б* изображена подсетка в двумерном случае.

На подсетке вводится конечномерное пространство U^h непрерывных функций, линейных в каждом тетраэдре, которые, тем самым, могут быть заданы в узлах подсетки. Поставим задачу: зная значения $p_1, \ldots, p_s$ в узлах, совпадающих с точками P, найти функцию $u^h \in U^h$, которая задаст приближение давления u на подсетке, а следовательно, позволит вычислить поток через части граней, попавшие в область взаимодействия. Это можно сделать множеством способов. Основные требования к функции заключаются в том, чтобы она удовлетворяла локальному принципу максимума во внутренних точках, а также чтобы ее минимальное и максимальное значение совпадали с $min(p_1, \ldots, p_s)$ и $max(p_1, \ldots, p_s)$ в случае граничных условий непротекания.

Одним из вариантов такого построения является следующий. Рассмотрим тетраэдр $PMEV$. Его грани могут относиться к трем типам:

1. внутренние для области взаимодействия,

2. внешние к ней и внешние к резервуару,

3. внешние к области взаимодействия, но внутренние для резервуара.

Наша цель — задать граничные условия на границе области взаимодействия, поэтому грани первого типа нас не интересуют. На гранях второго типа условия уже заданы как условия на границе резервуара. Определим условия первого рода для граней третьего типа. Заметим, что такой гранью может быть только грань PME, т.к. любая грань, содержащая точку V, может относиться только к первым двум типам — либо найдется парный тетраэдр, содержащий эту грань, либо не найдется, и тогда V — граничный узел сетки, а грань тетраэдра составляет часть границы резервуара. Значение в точке P нам известно. Построим приближенные значения в точках M и E. Точка M является центром масс грани, которая принадлежит двум или одному блоку сетки. В случае двух блоков с центрами P_1 и P_2, значение в M вычисляется как

$$u(M) = \frac{c_2 u(P_1) + c_1 u(P_2)}{c_1 + c_2}, \quad c_i = \|A_i^{-1}\overrightarrow{P_i M}\|. \tag{2.7}$$

Когда же существует только один блок с центром P, то M принадлежит границе резервуара. В случае, когда на грани Ξ, содержащей M, заданы условия первого рода, значение $u(M)$ определяется ими, а когда второго рода с заданным потоком Φ — вычисляется по формуле:

$$u(M) = u(P) + \frac{\Phi}{\mathrm{mes}(\Xi)}\left(\boldsymbol{n}(\Xi), \overrightarrow{PM}\right). \tag{2.8}$$

Для задания значения в точке E можно воспользоваться взвешенным осреднением значений в точках M, которые лежат в тех же гранях, что и E:

$$u(E) = \frac{\sum_M \|\overrightarrow{ME}\|^{-1} u(M)}{\sum_M \|\overrightarrow{ME}\|^{-1}}. \tag{2.9}$$

После того, как мы определили значения в точках P, M и E, функцию можно линейно доопределить на весь треугольник PME.

Теперь, когда заданы граничные условия на области взаимодействия, нам достаточно численно найти приближение решения задачи $\operatorname{div} A\nabla u = 0$ в пространстве U^h, например, методом конечных элементов ([54]). Таким образом, полученная функция будет удовлетворять локальному принципу максимума, а значит, будет задавать качественно правильное распределение давления.

Обратимся к вычислению потоков. Части граней сетки, попавшие в область взаимодействия, могут быть представлены как объединение треугольников MEV, являющихся внутренними по отношению к области. Каждый такой треугольник в свою очередь разбит на n^2 «маленьких» треугольников, являющихся внутренними гранями подсетки (см. рис. 2.4, *а*). Это означает, что для каждого «маленького» треугольника найдется два тетраэдра подсетки, которые его содержат. Для каждого из этих тетраэдров известна линейная функция u^h, а следовательно, можно вычислить поток $\int_T (A\nabla u^h, \boldsymbol{n})\, ds = \operatorname{mes}(T)(A\nabla u^h, \boldsymbol{n})$ через рассматриваемый треугольник. Таким образом, поток через «маленький» треугольник можно определить как полусумму потоков, вычисленных относительно каждого из тетраэдров. Поток через часть грани складывается из потоков составляющих ее «маленьких» треугольников. Поток через полную грань равен сумме потоков через ее части, содержащиеся в областях взаимодействия четырех вершин.

Шаблоном метода пространственной аппроксимации по отношению к грани называется набор блоков $V_{i_1}, \ldots, V_{i_m}$ (или их центров), из которых используются значения параметров, входящих в формулу потока. В регулярном случае шаблон метода базисов связей содержит 10 точек, а шаблоны O-метода и метода подсетки состоят из 18 точек (см. рис. 2.5). Необходимо отметить, что на ортогональной сетке шаблоны метода базисов связей и O-метода вырождаются в двухточечные, совпадающие с шаблоном TPFA, в то время как для метода подсетки он остается неизменным. Это обстоятельство влияет на скорость расчета, которую можно повысить, выбросив нулевые слагаемые, од-

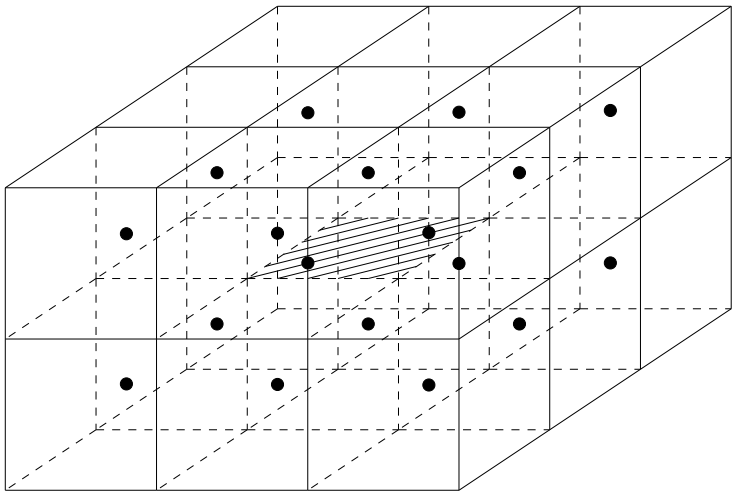
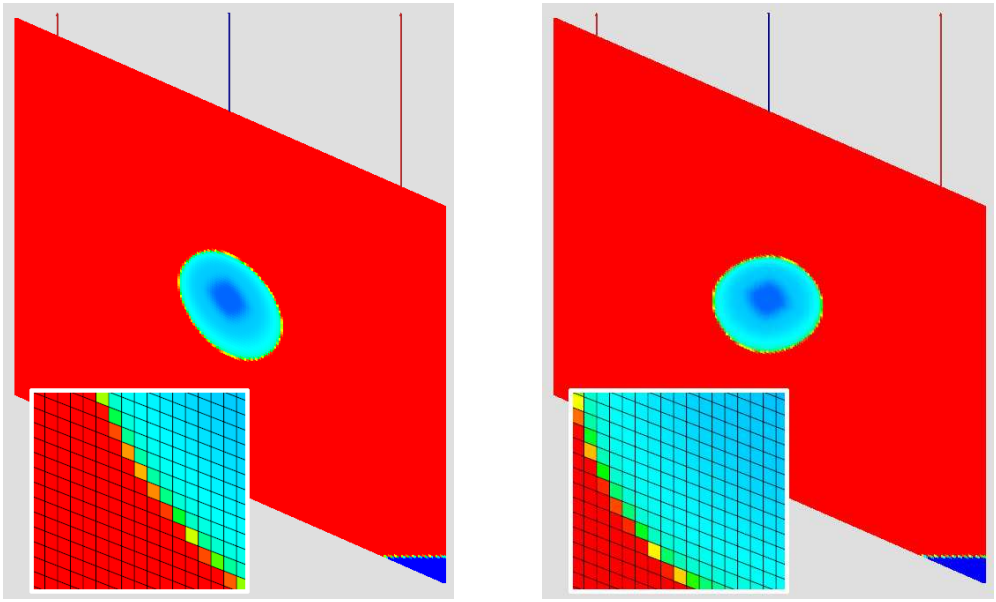


Рис. 2.5. Шаблон метода подсетки

нако говорит в пользу метода подсетки, поскольку такое поведение лучше соответствует физической картине явления — поток через грань должен зависеть от всех значений в ее окрестности.

Численные эксперименты. Вначале проиллюстрируем эффект от применения многоточечной аппроксимации. Рассмотрим синтетическую модель с квазиравномерной сеткой, блоки которой в сечении плоскостью, параллельной плоскости Oxz, являются параллелограммами. В модели участвуют одна нагнетательная скважина, которая закачивает воду с постоянной скоростью в центр резервуара, и две добывающих, которые выкачивают жидкость, стабилизируя тем самым среднее давление. На срезе функции насыщености нефти видно, что фронт распространения изменений при расчете многоточечным методом близок к окружности, тогда как двухточечный дает эллипс, ось которого наклонена вдоль характерного искривления сетки, что противоречит физической картине течения (см. рис. 2.6). Результаты расчетов различными методами MPFA на этой модели оказались близки друг к другу.

Проверим теперь поведение предобуславливателя из раздела 2.2 на наборах данных, для которых были построены матрицы, перечисленные в табл. 2.1.



(*а*). Расчет методом TPFA (*б*). Расчет методом MPFA

Рис. 2.6. Нефтенасыщенность в модели с параллелограммами

Матрицы при многоточечной аппроксимации стали существенно сложнее: среднее количество ненулевых элементов увеличилось на 20, что для некоторых матриц из таблицы может означать почти учетверение количества ненулевых элементов. Соответственно, усложнился граф матрицы и увеличилась ширина ленты. Численные эксперименты подтвердили эффективность блочного ILU разложения, который сохранил свои преимущества и на таком наборе матриц. Замедление относительно расчета с TPFA составляет от 2-х до 4-х раз в зависимости от сложности модели, причем большее замедление относится как раз к более «простым» матрицам, где максимальное количество ненулевых элементов было 5–8.

2.7. Выводы ко второй главе

Во второй главе описан предложенный метод блочного хранения разреженных несимметричных матриц в памяти и блочные варианты построения предобуславливателей, основанных на неполном LU разложении, для систе-

мы уравнений, получающейся при аппроксимации задачи фильтрации вязкой сжимаемой смеси в пористой среде. Предложенные методы обладают следующими свойствами:

- Расширяется класс матриц, для которых применимы ILU предобуславливатели.
- Происходит избирательное увеличение ширины шаблона матрицы, улучшающее характеристики сходимости алгоритмов с использованием ILU разложения.
- Появляется ускорение за счет использования кэш-памяти.
- Уменьшается требуемый объем памяти для хранения матрицы.
- Увеличиваются возможности для масштабирования на параллельных ЭВМ.

Глава 3

Параллельные блочные предобуславливатели класса ILU

Для решения систем линейных алгебраических уравнений (СЛАУ) большой размерности на параллельных ЭВМ применяются различные методы разбиения матрицы между процессами и различные предобуславливатели. Классическим методом разбиения является блочно-диагональный [43], с помощью которого естественным образом распараллеливаются стандартные алгоритмы решения СЛАУ. Если для предобуславливателя Якоби такой метод дает ожидаемый результат, то для более сложных алгоритмов вроде ILU, параллельная масштабируемость практически отсутствует: на p процессорах каждая итерация идет в p раз быстрее, но итераций требуется практически в p раз больше. Это легко объяснимо: при таком разделении (см. рис. 3.1) каждый процесс теряет значительную часть информации о матрице (зеленым цветом отмечены внедиагональные элементы, которые учтены при построении разложения, красным – отброшенные элементы).

Задаче построения вариантов LU и ILU разложений, эффективно работающих на параллельных ЭВМ, посвящено много работ. В основном, для распараллеливания предлагались те или иные способы обхода неизвестных в системе для того, чтобы уменьшить количество обменов между параллельно работающими процессами ([61], [62], [63], [64], [65], [66], [67], [68], [51], [69]), либо введение на графе матрицы иерархической структуры аналогично с многосеточными методами и методами декомпозиции ([70], [50], [71], [72], [73], [74]).

И. Е. Капорин и И. Н. Коньшин [75], [76] при решении симметричных положительно-определенных СЛАУ методом сопряженных градиентов предло-

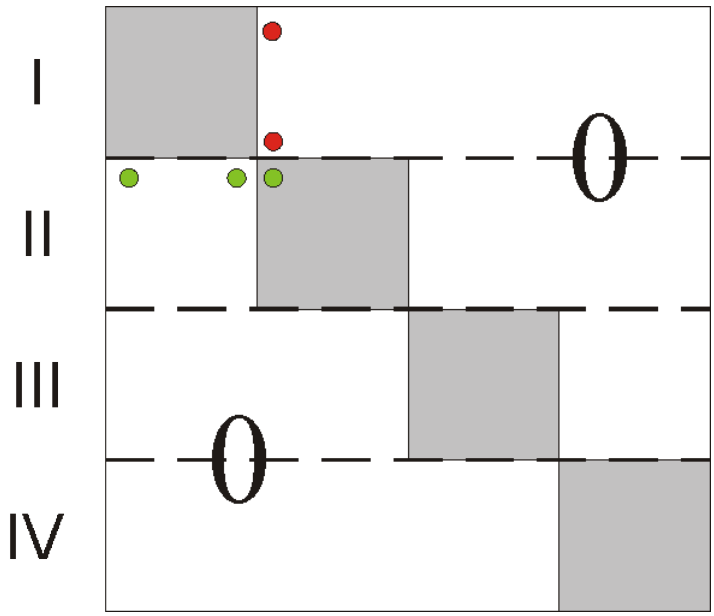


Рис. 3.1. Примеры блочно-диагонального разбиения матрицы

жили использовать для построения предобуславливателя метод перекрывающихся разбиений матрицы СЛАУ на блоки. К матрице применяется блочная версия двустороннего неполного обратного разложения Холецкого, а затем каждый из блоков заменяется на аппроксимацию — результат неполного разложения Холецкого. Полученный таким образом метод обладает лучшими характеристиками сходимости и параллелизма по сравнению с большинством других методов разбиения.

Ниже мы рассмотрим решение несимметричных СЛАУ методом бисопряженных градиентов на параллельных ЭВМ, где предобуславливателем служит ILU-разложение, получающееся с помощью метода разбиения матрицы, предложенного И. Е. Капориным и И. Н. Коньшиным. Будут исследованы различные стратегии выбора перекрытий: выбор с фиксированным размером перекрытия и выбор элементов с помощью графа связности ненулевых элементов матрицы коэффициентов СЛАУ; а также проведено их сравнение.

3.1. Представление ILU предобуславливателя в аддитивной форме

Рассмотрим матрицу A для решаемой системы вида

$$A = \begin{pmatrix} A_{11} & A_{12} & 0 \\ A_{21} & A_{22} & A_{23} \\ 0 & A_{32} & A_{33} \end{pmatrix}.$$

Пусть L и U — матрицы, соответствующие точному LU-разложению матрицы A. Обозначим: $L' = L^{-1}$, $U' = U^{-1}$. Следуя [75], рассмотрим произведение

$$H = \begin{pmatrix} U'_{11} & U'_{12} & 0 \\ 0 & U'_{22} & U'_{23} \\ 0 & 0 & U'_{33} \end{pmatrix} \cdot \begin{pmatrix} L'_{11} & 0 & 0 \\ L'_{21} & L'_{22} & 0 \\ 0 & L'_{32} & L'_{33} \end{pmatrix} =$$

$$= \begin{pmatrix} \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}^{-1} & 0 \\ 0 \quad 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 0 & \begin{pmatrix} A_{22} & A_{23} \\ A_{32} & A_{33} \end{pmatrix}^{-1} \end{pmatrix} - \begin{pmatrix} 0 & 0 & 0 \\ 0 & A_{22}^{-1} & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

Введем обозначения:

$$A'' = \begin{pmatrix} A_{22} & A_{23} \\ A_{32} & A_{33} \end{pmatrix},$$

а L'' и U'' — матрицы, соответствующие точному LU-разложению матрицы A''. Тогда

$$\begin{pmatrix} A_{22} & A_{23} \\ A_{32} & A_{33} \end{pmatrix}^{-1} - \begin{pmatrix} A_{22}^{-1} & 0 \\ 0 & 0 \end{pmatrix}$$

$$= \begin{pmatrix} U''_{22} & U''_{23} \\ 0 & U''_{33} \end{pmatrix}^{-1} \begin{pmatrix} I & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} L''_{22} & 0 \\ L''_{32} & L''_{33} \end{pmatrix}^{-1} - \begin{pmatrix} U''_{22} & U''_{23} \\ 0 & U''_{33} \end{pmatrix}^{-1} \begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} L''_{22} & 0 \\ L''_{32} & L''_{33} \end{pmatrix}^{-1}$$

$$= \begin{pmatrix} U''_{22} & U''_{23} \\ 0 & U''_{33} \end{pmatrix}^{-1} \begin{pmatrix} 0 & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} L''_{22} & 0 \\ L''_{32} & L''_{33} \end{pmatrix}^{-1}.$$

Таким образом, исходная матрица H представлена в виде суммы двух матриц меньшей размерности, для каждой из которых можно независимо находить LU-разложение.

Приведенные выкладки показывают, что матрица H близка к матрице A^{-1} и является хорошим предобуславливателем для итерационных алгоритмов решения СЛАУ.

Заменив в выкладках точное LU-разложение на ILU-разложение, мы получим аналогичный результат и для ILU-разложения.

Отметим также, что аналогично можно представить любую блочную трехдиагональную матрицу размерности ($k = 4,5,\ldots$) в виде суммы $k-1$ матриц меньшей размерности, для каждой из которых можно независимо находить LU (или ILU)-разложение.

3.2. Параллельный вариант блочного алгоритма построения ILU разложения

Рассмотрим произвольную матрицу A размерности n. Пусть p — число процессов, $n_1, n_2, \ldots, n_{p+1}$ — числа такие, что $n_1 + n_2 + \ldots + n_{p+1} = n$. Очевидно, что, обнулив некоторые элементы матрицы A, можно получить блочную трехдиагональную матрицу A', такую, что ее i-й диагональный блок будет иметь размерность n_i. Как было показано в разд. 1, предобуславливатель для упомянутой блочной трехдиагональной матрицы A' можно представить в виде суммы p матриц меньшей размерности, для каждой из которых можно независимо найти ILU-разложение.

Поскольку мы решаем систему линейных уравнений с матрицей (65), возникающей на каждой итерации метода Ньютона (62), с помощью которого решается система нелинейных уравнений (54), аппроксимирующая исходную систему (4), то в соответствии с разделом 2.2 мы будем использовать блочное

хранение матрицы и блочный вариант ILU разложения в каждом из параллельно работающих процессов. Для получения законченного алгоритма осталось указать правило, по которому будут распределяться строки матрицы на имеющиеся логические процессоры и как будут вычисляться номера строк, попадающих в более, чем 1 процесс.

3.3. Методы разбиения матрицы на части

Для дальнейшего изложения важно следующее замечание: чтобы вычислить элементы ILU-разложения, находящиеся в i-й строке, нам необходимо использовать уже рассчитанные элементы всех строк с меньшими номерами. При аддитивном представлении матрицы A в каждом слагаемом A^k ($k = 1, \ldots, p$) есть строки, начинающиеся лишь с некоторого номера. Соответственно получаемые при ILU-разложении матриц A^k матрицы L^k и U^k при всех $k > 1$ значительно отличаются от соответствующих частей ILU-разложения исходной матрицы A.

Отметим, однако, что в силу разреженной структуры матриц при вычислении i-й строки ILU-разложения нужны только строки с номерами ненулевых элементов i-й строки.

Отсюда следует, что, с одной стороны, чем больше строк будет включено в каждую из подматриц, тем точнее получится ILU-разложение и соответственно будет меньшее число итераций, а с другой — чем больше строк мы добавим, тем существеннее увеличится объем вычислений и каждая итерация будет занимать большее время. Это означает, что существует оптимальное разбиение исходной матрицы на части.

3.3.1. Геометрический подход

Пусть задача решается в трехмерной области, аппроксимированной трехмерной сеткой, размерность которой по x, y и z составляет n_1, n_2 и n_3 соответственно (см. раздел 2.3). Положим

$$t = \alpha \min_{i \in \{1,2,3\}} \prod_{\substack{j \in \{1,2,3\} \\ j \neq i}} n_i,$$

т.е. t — минимальное из попарных произведений n_i, увеличенное в α раз. Разобьем матрицу A на p равных частей и добавим t предыдущих строк к каждой части, начиная со второй.

У такого подхода есть значительные преимущества: нахождение разбиения не требует больших численных затрат; шаблоны ненулевых элементов матрицы и предобуславливателя совпадают.

3.3.2. Алгебраический подход

У геометрического подхода есть существенный недостаток: в случае, когда физическая модель обладает сложной структурой, т.е. имеются связи между элементами, которые лежат не в соседних слоях сетки (например, сильно неравномерная сетка), данный подход может привести к получению плохого ILU-разложения, так как игнорируются существенные элементы исходной матрицы.

Алгебраический подход устраняет этот недостаток геометрической реализации.

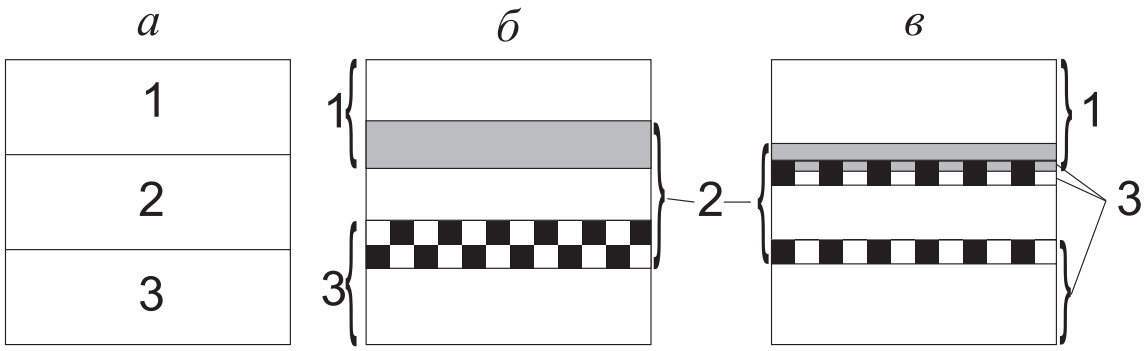
Итак, пусть A_k^0 — k-я часть матрицы A при равномерном разбиении, J_k^0 — номера всех строк, попавших в A_k^0. Проведем первый шаг алгоритма. Обозначим $J_k^1 = J_k^0 \cup \{j | \exists i \in J_k^0 : a_{ij} \neq 0\}$, т.е. мы присоединили к уже имеющимся в множестве J_k^0 индексам все номера столбцов ненулевых элементов, содержащихся в строках матрицы A_k^0. Матрицу, состоящую из строк с номерами из

множества J_k^1 обозначим A_k^1. На втором шаге алгоритма описанную для множества J_k^0 и матрицы A_k^0 процедуру проведем для множества J_k^1 и матрицы A_k^1, в результате получим множество J_k^2 и матрицу A_k^2, и т.д. Таким образом можно построить последовательность множеств $\{J_k^i\}_{i=1}^{\infty}$ и матриц $\{A_k^i\}_{i=1}^{\infty}$. Очевидно, что в силу конечности строк исходной матрицы A, начиная с некоторого l все члены указанных последовательностей совпадут.

В результате таких операций нумерация строк матрицы A_k^i, вообще говоря, может нарушиться. Чтобы устранить этот недостаток, пересортируем строки исходной матрицы так, чтобы номера строк каждой из матриц A_k^i шли подряд, без пропусков.

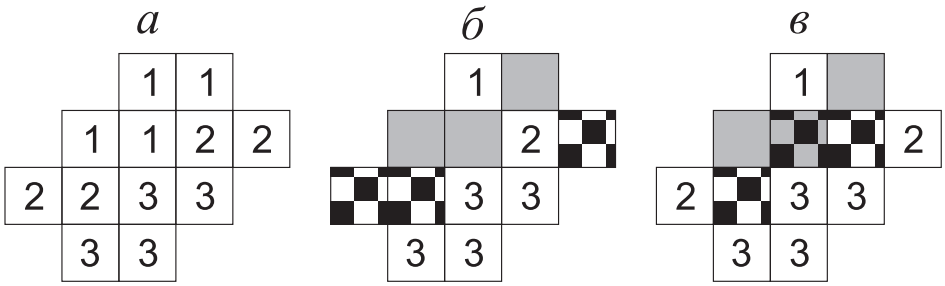
Такая реализация более затратна с точки зрения вычислений, а из-за стратегии выборки строк предобуславливатель имеет отличный от исходной матрицы шаблон ненулевых элементов, который придется хранить в памяти.

На рис. 3.2 показаны различия в выборке перекрытий при равномерном разбиении матрицы на 3 части, а также при использовании геометрического и алгебраического подходов разбиения матрицы. На рис. 3.3 представлено соответствующее рис. 3.2 разбиение элементов сетки между процессами. Видно, что в геометрическом подходе никак не учитываются свойства сетки, и поэтому могут как присоединиться к k-му процессу элементы, с которыми нет связи, так и не учитываться те элементы, с которыми связи есть. С другой стороны, в геометрическом подходе могут быть присоединены элементы, не попадающие в граф связей, но стоящие в соседних строках матрицы, и потому оказывающие большое влияние на рассматриваемую часть матрицы из-за структуры построения ILU разложения, см. рис. 3.4, где такой элемент (имеющийся в шаблоне матрицы при геометрическом подходе, но не в алгебраическом) отмечен белым кружком, а отсутствующий в геометрическом подходе, но имеющийся в алгебраическом – красным.



а — равномерное разбиение; *б* — геометрический подход; *в* — алгебраический подход. (Серым цветом отмечены строки, присоединенные ко второму процессу, а клеткой — к третьему.)

Рис. 3.2. Пример разбиения матрицы между тремя процессами

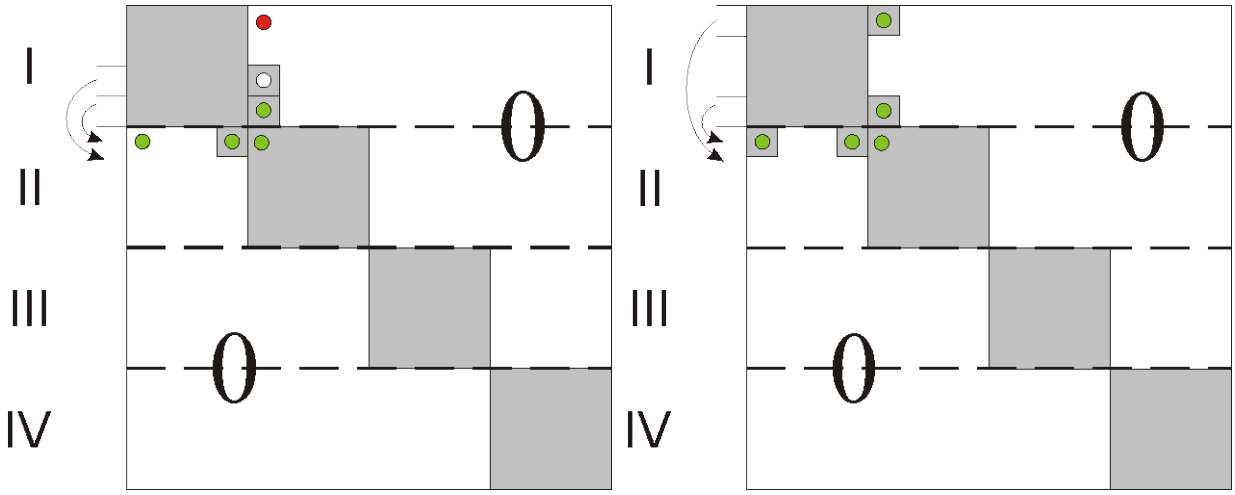


а — равномерное разбиение; *б* — геометрический подход при $\alpha = 1$; *в* — алгебраический подход 1 шаг. (Серым цветом отмечены строки, присоединенные ко второму процессу, а клеткой — к третьему.)

Рис. 3.3. Пример распределения между процессами элементов сетки для прямоугольной области с вырезом

3.4. Численные эксперименты

Для численных экспериментов взяты матрицы системы линейных уравнений с матрицей (65), полученные при дискретизации уравнений (4) с данными для реальных месторождений на разных шагах по времени. Для решения системы линейных уравнений используется итерационный алгоритм BCGS, в котором в качестве предобуславливателей берутся блочные варианты рассматриваемых ILU-разложений: ILU-разложение с использованием геометрического подхода при $\alpha = 2$ (ILUG) и алгебраического подхода с двумя шагами (ILUA); ILU(1)-разложение; ILUT-разложение (с параметрами $p = 5$ и $\tau = 10^{-3}$). В качестве критерия окончания итерационного процесса выбрано условие падения



(*а*). Геометрический подход (*б*). Алгебраический подход

Рис. 3.4. Сравнение способов разбиения матрицы

относительной невязки на 6 порядков.

В таблице 3.1 приведены некоторые характеристики матриц:

- n_x — размер сетки по x;
- n_y — размер сетки по y;
- n_z — размер сетки по z;
- d — размер блоков;
- r — количество скалярных строк матрицы (соответственно количество блочных строк равно r/d);
- b — количество хранящихся блоков;
- n — количество хранимых скалярных элементов (равно $b \cdot d^2$);
- m — максимальное количество ненулевых блоков в строке матрицы;
- k_{sk} — коэффициент кососимметричности, вычисленный по формуле

$$k_{sk} = \max_{i=1,\ldots,r/d} \sum_{j=1}^{r/d} \|A_{ij} - A_{ji}\|_\infty,$$

	n_x	n_y	n_z	d	r	b	n	m	k_{sk}
1	120	160	6	2	199490	649075	2596300	11	$3{,}564 \cdot 10^5$
2	83	107	85	2	205650	699501	2798004	49	$2{,}234 \cdot 10^7$
3	126	109	54	2	283996	801368	3205472	12	$1{,}146 \cdot 10^8$
4	130	123	17	2	382580	1200506	4802024	18	$1{,}226 \cdot 10^6$
5	114	238	12	2	364002	1200577	4802308	15	$7{,}955 \cdot 10^7$
6	88	71	28	3	281457	552287	4970583	24	$1{,}620 \cdot 10^7$
7	133	120	50	2	499322	1491985	5967940	47	$1{,}813 \cdot 10^7$
8	133	120	50	2	568006	1669725	6678900	36	$4{,}108 \cdot 10^5$
9	392	495	3	2	565086	1786979	7147916	7	$1{,}340 \cdot 10^5$
10	71	71	575	2	759746	1905443	7621772	7	$1{,}632 \cdot 10^6$
11	194	121	55	2	589580	1905520	7622080	40	$4{,}786 \cdot 10^6$
12	122	147	35	2	679262	2190961	8763844	22	$4{,}276 \cdot 10^6$
13	138	81	50	2	748776	2437818	9751272	40	$1{,}395 \cdot 10^8$
14	46	99	173	3	510750	1156672	10410048	14	$1{,}860 \cdot 10^{11}$
15	97	135	30	3	575025	1205459	10849131	7	$2{,}139 \cdot 10^3$
16	133	103	117	2	815932	2743130	10972520	53	$3{,}042 \cdot 10^5$
17	80	71	125	2	858382	2943033	11772132	63	$2{,}904 \cdot 10^6$
18	398	204	3	3	730728	1539038	13851342	7	$2{,}550 \cdot 10^3$
19	90	174	42	3	702405	1609689	14487201	7	$9{,}891 \cdot 10^5$
20	60	55	276	3	766098	1721412	15492708	17	$1{,}821 \cdot 10^6$
21	276	381	12	2	1391786	4242249	16968996	14	$2{,}362 \cdot 10^6$
22	420	280	30	2	1353710	4536033	18144132	30	$1{,}396 \cdot 10^6$
23	99	165	126	3	895212	2043612	18392508	10	$4{,}724 \cdot 10^5$
24	181	116	83	3	1028625	2240069	20160621	39	$1{,}024 \cdot 10^7$
25	78	263	31	3	1121196	2456774	22110966	23	$7{,}889 \cdot 10^9$
26	389	244	30	2	1749732	5726484	22905936	28	$1{,}400 \cdot 10^8$
27	249	198	12	3	1393869	2980557	26825013	16	$3{,}651 \cdot 10^6$
28	190	134	575	2	2702850	6746339	26985356	7	$2{,}847 \cdot 10^3$
29	407	412	37	2	2400402	7367991	29471964	28	$1{,}268 \cdot 10^6$
30	60	220	85	2	2188842	7484949	29939796	88	$2{,}486 \cdot 10^6$
31	219	454	150	2	2715768	8579656	34318624	65	$3{,}658 \cdot 10^7$
32	117	214	64	3	1965948	4055996	36503964	45	$3{,}096 \cdot 10^{10}$
33	354	750	36	3	2348076	4921354	44292186	10	$6{,}782 \cdot 10^{10}$
34	198	366	104	2	4863578	15818315	63273260	48	$5{,}120 \cdot 10^6$
35	280	290	80	3	4970010	8818554	79366986	18	$5{,}144 \cdot 10^6$
36	88	215	177	3	7256967	16524695	148722255	9	$5{,}475 \cdot 10^6$

Таблица 3.1. Характеристики матриц для тестирования параллельного блочного ILU

где $\|\cdot\|_\infty$ — строчная норма матрицы.

Пусть α_i^{jp} — время работы (или количество итераций) j-го алгоритма на p процессах на i-й матрице. Тогда рассмотрим следующее усреднение:

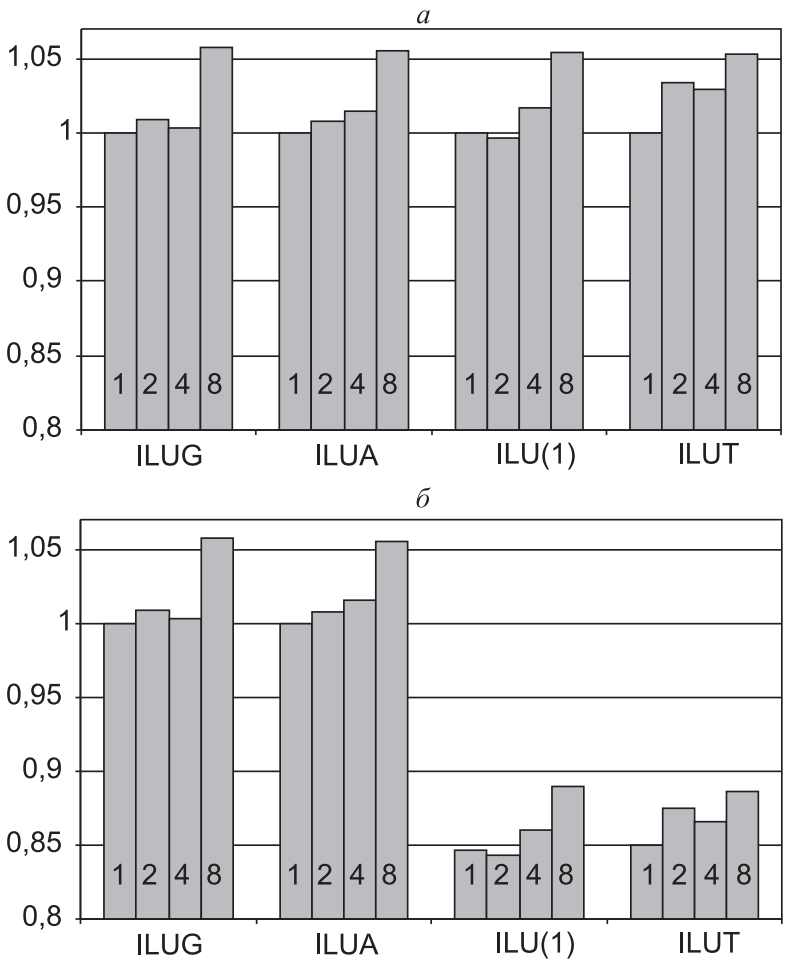
$$c_{j_2 p_2}^{j_1 p_1} = \frac{1}{N} \sum_{i=1}^{N} \frac{\alpha_i^{j_1 p_1}}{\alpha_i^{j_2 p_2}}, \tag{3.1}$$

здесь p_1, p_2 — число процессов, на которых запускался расчет задачи, а j_1, j_2 обозначают алгоритм, использующийся при решении, и принимают значения ILUG, ILUA, ILU(1), ILUT. Данный коэффициент $c_{j_2 p_2}^{j_1 p_1}$ показывает среднее ускорение (или среднее увеличение числа итераций).

Решение задач выполнялось на системе с 2-мя процессорами Intel Nehalem (2xXeon 5570 с общей памятью 20 Гб, частота каждого процессора 3,0 ГГц) и на системе «Регатта» Московского Государственного Университета (IBM eServer pSeries 690 с общей памятью 64 Гб, состоящего из 16 процессоров Power4, частота каждого процессора 1,3 ГГц). На системе Intel Nehalem задачи запускались на 1, 2, 4 и 8 процессорах; а на системе «Регатта» — на 1, 2, 4, 8 и 15 процессорах (один процессор занимает операционная система, поэтому запуск на 16 ядрах не дает ускорения и лишь искажает результаты численных экспериментов).

На рис. 3.5, 3.6 приведены результаты сравнения между собой алгоритмов с использованием ILUG-, ILUA-, ILU(1)- и ILUT$(5, 10^{-3})$-разложений при расчете на системе Intel Nehalem, а на рис. 3.7, 3.8 — сравнение тех же алгоритмов при расчете на системе «Регатта».

Как показывают проведенные численные эксперименты, средний рост числа итераций не превышает 6% для всех рассматриваемых алгоритмов, что говорит о том, что благодаря использованию метода Капорина–Коньшина параллельной реализации блочных предобуславливатей получаемое ILU-разложения обладает хорошими характеристиками масштабируемости. Наибольший рост числа итераций не превышет 40% при расчете на 8 ядрах.

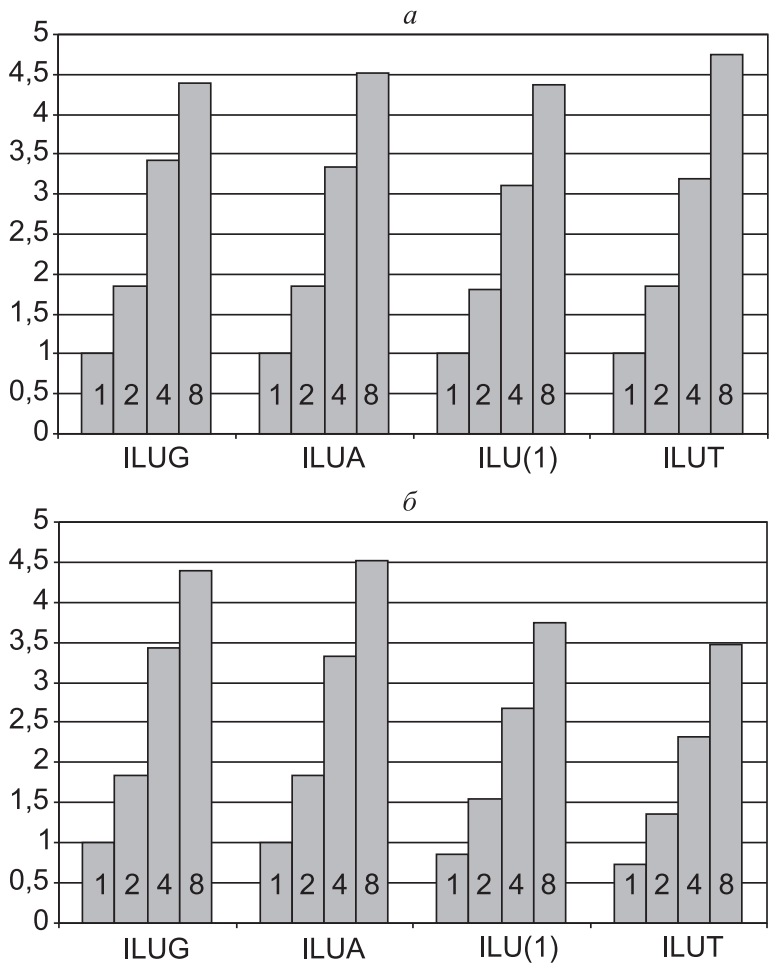


Сравнение среднего (усреднение бралось по формуле (3.1)) количества итераций при расчете на системе Intel Nehalem со следующими значениями параметров j_i, p_i: $j_2 = j_1 \in \{\text{ILUG, ILUA, ILU(1), ILUT}\}$, $p_1 \in \{1;2;4;8\}$, $p_2 = 1$ (*а*); $j_1 = \text{ILUG}$, $j_2 \in \{\text{ILUG, ILUA, ILU(1), ILUT}\}$, $p_1 \in \{1;2;4;8\}$, $p_2 = 1$ (*б*).

Рис. 3.5. Сравнение среднего количества итераций на системе Intel Nehalem

Также видно, что с точки зрения асимптотики роста количества итераций наилучшими оказываются ILUA-разложение. Это объясняется тем, что в данном алгоритме учитываются свойства матрицы, вследствие чего от нее зависит ширина перекрытия строк матриц между процессами и размер шаблона матриц, в то время как все остальные алгоритмы используют фиксированные параметры.

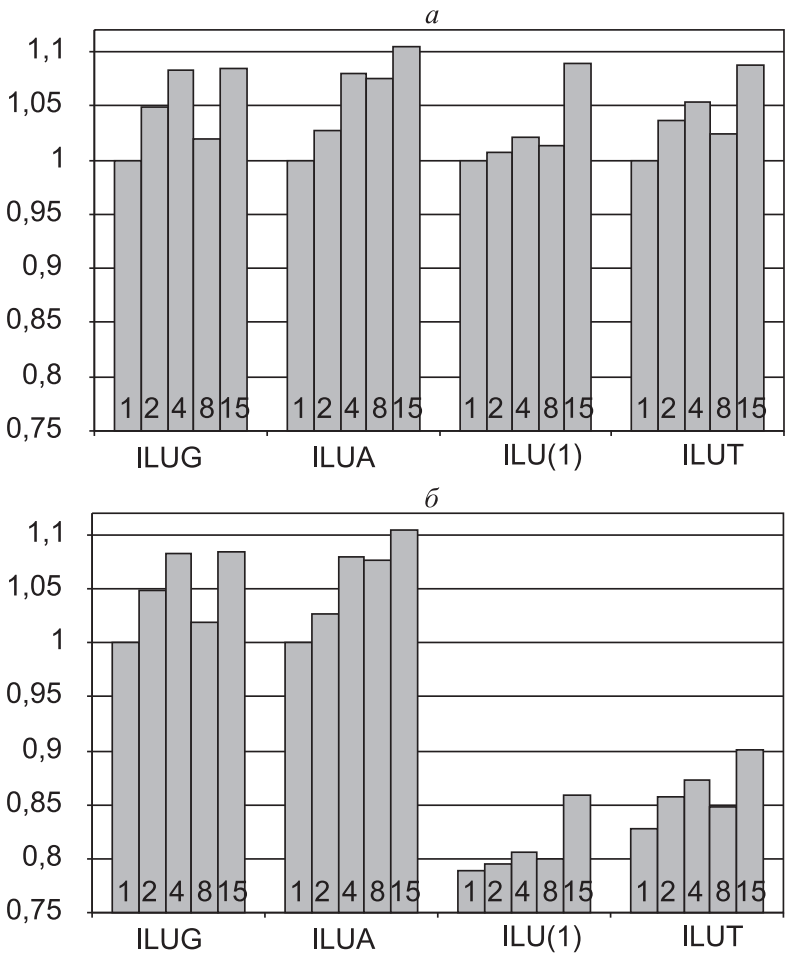
Отметим также, что за счет расширения шаблона ненулевых элементов число итераций для ILU(1)- и ILUT-разложений существенно меньше, чем для ILUG- и ILUA-разложений.



Сравнение среднего (усреднение бралось по формуле (3.1)) ускорения при расчете на системе Intel Nehalem со следующими значениями параметров j_i, p_i: $j_1 = j_2 \in \{\text{ILUG, ILUA, ILU(1), ILUT}\}$, $p_1 = 1$, $p_2 \in \{1;2;4;8\}$ (*а*); $j_1 \in \{\text{ILUG, ILUA, ILU(1), ILUT}\}$, $j_2 = \text{ILUG}$, $p_1 = 1$, $p_2 \in \{1;2;4;8\}$ (*б*).

Рис. 3.6. Сравнение среднего ускорения на системе Intel Nehalem

Как показывают графики сравнения времени работы алгоритмов в зависимости от количества процессов, все рассматриваемые алгоритмы обладают хорошими характеристиками распараллеливания. Во многом это объясняется малым увеличением количества итераций с увеличением расчетных узлов. Отметим здесь, что при использовании ILU(1)-разложения характиристики параллелизации хуже, чем при использовании остальных алгоритмов: это объясняется тем, что стратегия расширения шаблона ненулевых элементов предобуславливателя никак не учитывает величины добавляемых элементов, что приводит к увеличению арифметических операций, но не к повышению точности

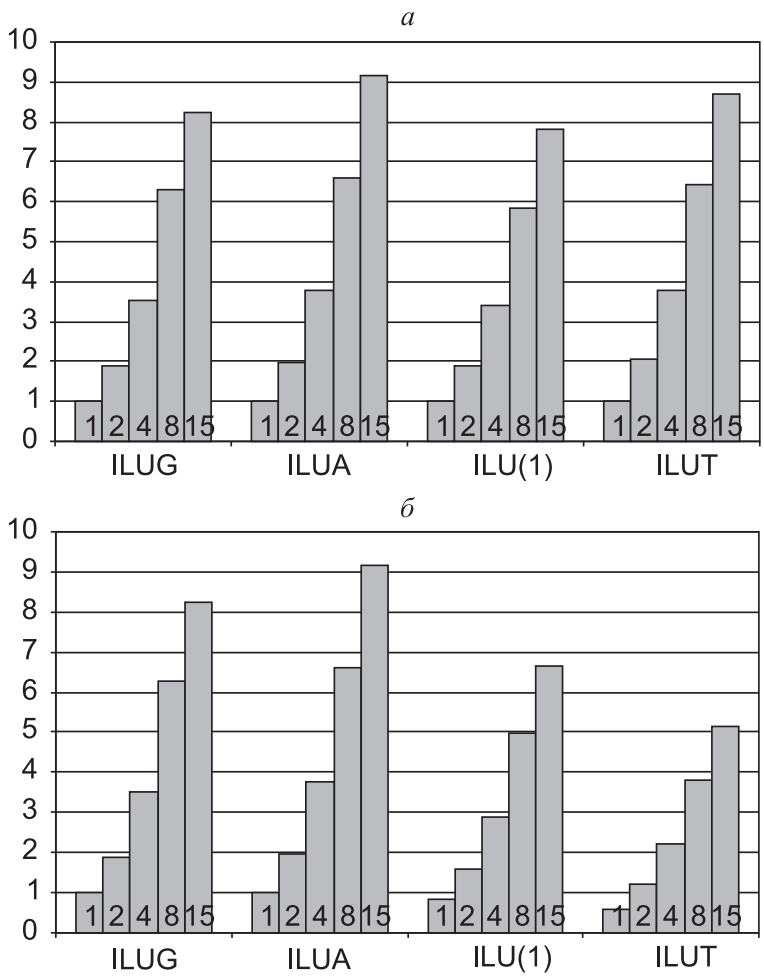


Сравнение среднего (усреднение бралось по формуле (3.1)) количества итераций при расчете на системе «Регатта» со следующими значениями параметров j_i, p_i: $j_2 = j_1 \in \{\text{ILUG, ILUA, ILU(1), ILUT}\}$, $p_1 \in \{1;2;4;8;15\}$, $p_2 = 1$ (*а*); $j_1 = \text{ILUG}$, $j_2 \in \{\text{ILUG, ILUA, ILU(1), ILUT}\}$, $p_1 \in \{1;2;4;8;15\}$, $p_2 = 1$ (*б*).

Рис. 3.7. Сравнение среднего количества итераций на системе «Регатта»

разложения. ILUT-разложение устраняет данный недостаток ILU-разложения и как следствие обладает лучшими характеристиками параллелизации.

Отметим, что на одном процессе в среднем ILUG- и ILUA-разложения гораздо предпочтительнее, чем ILU(1)- и ILUT-разложения. В первую очередь это объясняется вычислительными затратами на построение предобуславливателя и большим числом ненулевых элементов в предобуславливателе для ILU(1)- и ILUT-разложений, в результате чего каждая итерация занимает большее время. Однако с увеличением количества процессов картина заметно меняется: качество предобуславливателя играет все более значимую роль и



Сравнение среднего (усреднение бралось по формуле (3.1)) ускорения при расчете на системе «Регатта» со следующими значениями параметров j_i, p_i: $j_1 = j_2 \in \{\text{ILUG, ILUA, ILU(1), ILUT}\}$, $p_1 = 1$, $p_2 \in \{1;2;4;8;15\}$ (*а*); $j_1 \in \{\text{ILUG, ILUA, ILU(1), ILUT}\}$, $j_2 = ILUG$, $p_1 = 1$, $p_2 \in \{1;2;4;8;15\}$ (*б*).

Рис. 3.8. Сравнение среднего ускорения на системе «Регатта»

время, затрачиваемое на итерационный процесс, компенсирует время, затраченное на построение предобуславливателя. При увеличении числа расчетных ядер в среднем наиболее предпочтительным оказывается ILUA-разложение.

3.5. Сравнение с предобуславливателем CPR

Метод CPR (Constrained Pressure Residual), основанный на идее выбора в качестве предобуславливателя матрицы, фактически отвечающей за неявную только по давлению аппроксимацию (IMPES [32]) и потому имеющую в

несколько раз меньшую размерность (в 3 раза для типичной 3-х компонентной смеси), был впервые предложен в работе [77]. Его эффективность для полностью неявной схемы на модельных задачах со структурированной сеткой детально исследована в работе [78], в сравнении с другими предобуславливателями – в работе [79]. В последнее время метод стал достаточно популярен в качестве параллельного предобуславливателя для задачи фильтрации, см. [80], [81], [82], [83], [84], [85], [86]. Ниже мы рассмотрим реализацию этого алгоритма и проведем сравнение с описанным выше методом.

Идея метода «Constrained Pressure Residual». Алгоритм предобуславливания методом CPR базируется на том факте, что уравнения системы (62) в ячейке с номером i слабо зависят от значений переменных N_c^i в ячейках, отличных от i-й. В результате появляется возможность приближенно найти переменные p_i, приравняв к нулю все производные по переменным N_c^i в блоках, отличных от диагональных. Данный алгоритм очень похож на метод IMPES – если взять значения насыщенностей не с верхнего слоя, а с предыдущего, то производные по ним в матрице A будут как раз равны нулю. При этом в диагональном блоке не обязательно приравнивать значения к нулю. Их можно сделать нулями с помощью гауссовского исключения. Полученное приближенное значение давления используется для предобуславливания всей матрицы A. Таким образом, мы избавляемся от основного недостатка IMPES – слишком малого шага по времени, на котором данный метод продолжает сходиться. Для приближенного вычисления p используем часть матрицы, состоящую из первых элементов блоков, соответствующих переменной давления.

В пользу данного метода говорит также и тот факт, что значения производных в матрице A, соответствующие давлению, в случае реальной модели часто на порядок превосходят по модулю значения производных, соответствующих молярным плотностям. В результате основной вклад в вектор невязки

дает именно та часть матрицы, которая соответствует переменной давления.

Данный алгоритм можно также назвать сильно упрощенной версией алгебраического многосеточного метода, учитывающей специфику конкретной задачи. Более того, получающаяся в результате подматрица A_p часто обладает следующими свойствами:

- Все диагональные элементы (за исключением, быть может, нескольких) больше нуля;
- Абсолютное большинство внедиагональных элементов <= 0;
- Сумма элементов в строке >= 0 (слабое диагональное преобладание).

Эти свойства проистекают из того факта, что наши исходные уравнения являются «почти эллиптическими» относительно переменной давления. Реальные вычисления подтверждают эту гипотезу. В случае, когда некоторые строки матрицы не имеют описанных свойств, можно немного их подкорректировать как с помощью исключения части блоков методом Гаусса, так и просто заменой некоторых элементов (ввиду того, что мы ищем лишь приближенное значение давления). Таким образом, для решения подсистемы с матрицей A_p можно применить быстрый алгебраический многосеточный метод AMG, см. [87] [88] [89] [86].

Алгоритм метода CPR. Обозначим процесс обнуления соответствующих элементов матрицы через $Null(A)$, процесс коррекции методом Гаусса через $Gauss(A)$ и процесс выделения из матрицы A подматрицы давления через $Reduce(A)$. Тогда матрица для приближенного вычисления давления A_p находится по формуле:

$$A_p = Reduce(Gauss(Null(A))).$$

На примере одной блочной строки с размером блока 3×3 это выглядит следующим образом.

Для диагонального блока:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \overset{Null}{\Longrightarrow} \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \overset{Gauss}{\Longrightarrow} \begin{pmatrix} a''_{11} & 0 & 0 \\ a'_{21} & a'_{22} & 0 \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \overset{Reduce}{\Longrightarrow} a''_{11}.$$

Для внедиагонального блока:

$$\begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{pmatrix} \overset{Null}{\Longrightarrow} \begin{pmatrix} b_{11} & 0 & 0 \\ b_{21} & 0 & 0 \\ b_{31} & 0 & 0 \end{pmatrix} \overset{Gauss}{\Longrightarrow} \begin{pmatrix} b''_{11} & 0 & 0 \\ b'_{21} & 0 & 0 \\ b_{31} & 0 & 0 \end{pmatrix} \overset{Reduce}{\Longrightarrow} b''_{11}.$$

Соответствующая правая часть:

$$\begin{pmatrix} r_1 \\ r_2 \\ r_3 \end{pmatrix} \overset{Null}{\Longrightarrow} \begin{pmatrix} r_1 \\ r_2 \\ r_3 \end{pmatrix} \overset{Gauss}{\Longrightarrow} \begin{pmatrix} r''_1 \\ r'_2 \\ r_3 \end{pmatrix} \overset{Reduce}{\Longrightarrow} r''_1.$$

Метод Гаусса в данном случае выглядит как:

$$\begin{aligned} (2)' &= (2) - ((3)/a_{33}) * a_{23}, \\ (1)' &= (1) - ((3)/a_{33}) * a_{13}, \\ (1)'' &= (1)' - ((2)'/a'_{22}) * a'_{12}. \end{aligned}$$

Здесь (1), (2) и (3) - соответственно первая, вторая и третья сроки в блоке (или векторе). Для блоков большей (или меньшей) размерности преобразования полностью аналогичны.

Обозначим теперь через M^{cpr} искомый предобуславливатель для матрицы A. Необходимо найти действие M^{cpr} на вектор r, т.е найти $\bar{x}^{cpr} = M^{cpr} r$. Мы стремимся к тому, чтобы действие матрицы M^{cpr} на r давало такой $\bar{x}^{cpr}$, что

$$A\bar{x}^{cpr} \approx r.$$

Действие предобуславливателя M^{cpr} выглядит так:

$$M^{cpr}Ax = M^{cpr}r = Mr + \begin{pmatrix} 0 \\ \bar{x}_p^{cpr} \end{pmatrix} - MA \begin{pmatrix} 0 \\ \bar{x}_p^{cpr}, \end{pmatrix} = \bar{x}^{cpr}, \tag{3.2}$$

где M - любой «обычный» (в дальнейшем будем называть его глобальным) предобуславливатель (например ILU(0)), а $\bar{x}_p^{cpr}$ есть подвектор давления, являющийся результатом решения системы:

$$A_p \bar{x}_p^{cpr} = r_p. \tag{3.3}$$

При этом r_p получено из r соответственно по формуле:

$$r = \begin{pmatrix} P^1 \\ N_1^1 \\ N_2^1 \\ \dots \\ P^n \\ N_1^n \\ N_2^n \end{pmatrix} \Longrightarrow r_p = \begin{pmatrix} P^1 \\ P^2 \\ \dots \\ P^n \end{pmatrix}.$$

Отметим, что формула (3.2) имеет смысл только в том случае, если вектор r есть результат действия матрицы A на искомый вектор x.

Предобуславливание методом CPR состоит из двух этапов. Сначала нужно применить в качестве предобуславливателя матрицу A_p^{-1} для переменных, соответствующих давлению p. Для этого необходимо выделить из вектора r соответствующий подвектор r_p и решить систему (3.3), найдя, таким образом, приближенное значение подвектора $\bar{x}_p^{cpr}$. Полученный результат следует подставить в формулу (3.2).

Обозначим через $\bar{x}$ величину MAx. Рассмотрим следующие векторы разностей в обычном и CPR случаях:

$$x - \bar{x} = x - MAx = (I - MA)x, \tag{3.4}$$

$$x - \bar{x}^{cpr} = x - Mr - \begin{pmatrix} 0 \\ \bar{x}_p^{cpr} \end{pmatrix} + MA \begin{pmatrix} 0 \\ \bar{x}_p^{cpr} \end{pmatrix} = \tag{3.5}$$

$$(I - MA)x - (I - MA) \begin{pmatrix} 0 \\ \bar{x}_p^{cpr} \end{pmatrix} = (I - MA) \left(x - \begin{pmatrix} 0 \\ \bar{x}_p^{cpr} \end{pmatrix} \right). \tag{3.6}$$

Таким образом, формула (3.2) в случае, если значения $\bar{x}_p^{cpr}$ близки к значениям x_p, значительно более эффективна.

Заметим, что метод CPR не зависит ни от метода решения локальной системы с матрицей A_p и, соответственно, от типа используемого локального предобуславливателя, ни от метода решения всей системы в целом, а также от типа глобального предобуславливателя M. Следовательно, он может быть использован в любой задаче, обладающей вышеописанными свойствами, как способ улучшить сходимость, и для его внедрения практически не потребуется менять уже имеющиеся алгоритмы решения. Применимость данного метода зависит от того, насколько найденный нами вектор $\bar{x}_p^{cpr}$ будет близок к соответствующей части x_p вектора x.

Особенности практической реализации на параллельных ЭВМ. Метод CPR достаточно легко интегрируется в параллельные алгоритмы решения систем линейных уравнений, основанных на перекрытии расчетных областей в разных процессах (см. раздел 3.3).

Обозначим

- $x_i = (p_i, N_i^1, \cdots, N_i^{n_c})$ - i-я компонента решения,
- $X^k = (x_{i_1^k}, \cdots, x_{i_{m_k}^k})$ - компоненты, отнесенные к k-му процессу (потоку исполнения),

- A^k, b^k - соответствующие части матрицы A (размера $m_k \times n_c \times N$) и правой части b в k-м процессе (потоке).

Для алгоритмов с перекрытием $\sum_{k=1}^{s} m_k > (n_c \times N)$, s - число процессов. Тогда, например, метод простой итерации в k-м процессе есть

$$M^k \frac{X^k_{l+1} - X^k_l}{\tau_l} + A^k X^k_l = b^k$$

с обычными методами синхронизации компонент решения в общих подобластях. Метод CPR применяется к матрице A^k в каждом процессе:

$$M^{cpr,k} A^k x^k = M^{cpr,k} r^k = M^k r^k + \begin{pmatrix} 0 \\ \bar{x}_p^{cpr,k} \end{pmatrix} - M^k A^k \begin{pmatrix} 0 \\ \bar{x}_p^{cpr,k} \end{pmatrix} = \bar{x}^{cpr,k}, \quad (3.7)$$

где индекс k означает соответствующую часть матрицы или вектора в k-м процессе. Заметим, что решение системы

$$A^k_p \bar{x}^{cpr,k}_p = r^k_p \quad (3.8)$$

происходит независимо в каждом из процессов и не требует пересылки данных между ними.

В качестве глобального предобуславливателя M^k брался параллельный вариант ILU(0) (см. раздел 3.2), в качестве предобуславливателя для матрицы A^k_p использовался либо обычный ILU(0) либо алгебраический многосеточный метод AMG.

Заметим, что такой подход обеспечивает эффективное распараллеливание AMG для этой задачи.

3.5.1. Использование ILU(0) совместно с CPR

Тесты проводились на наборах данных для реальных месторождений. Критерием остановки алгоритма решения системы (3.3) является выполнение

условия:

$$\frac{||A_p \bar{x}_p^{cpr}||}{||r_p||} < \varepsilon,$$

где ε - точность локального алгоритма (норма евклидова). Этот дополнительный параметр алгоритма CPR существенно влияет на общую скорость работы. В таблице 3.2 приведены результаты расчета на наборе данных с числом неизвестных 255366×3, в первой строке – результат алгоритма из раздела 3.4, далее идут результаты алгоритма CPR при различных значениях ε.

Метод	Кол. ит. метода Ньют.	Кол. ит. глоб. алг.	Время (сек.)
ILU(0)	79	643	177
CPR+ILU(0) ($\varepsilon = 5 * 10^{-1}$)	78	262	163
CPR+ILU(0) ($\varepsilon = 10^{-1}$)	78	188	150
CPR+ILU(0) ($\varepsilon = 5 * 10^{-2}$)	78	164	150
CPR+ILU(0) ($\varepsilon = 10^{-2}$)	79	165	156
CPR+ILU(0) ($\varepsilon = 5 * 10^{-3}$)	79	172	165
CPR+ILU(0) ($\varepsilon = 10^{-3}$)	79	172	176

Таблица 3.2. Результаты работы алгоритма CPR при разных параметрах

Видим, в качестве критерия остановки локального алгоритма логично брать значение $\varepsilon = 5 * 10^{-2}$ или $\varepsilon = 10^{-2}$, хотя, к сожалению, это не является универсальным правилом, справедливым для всех наборов данных. Заметим, что число глобальных итераций линейного алгоритма при этом сокращается почти в 4 раза, хотя, к сожалению, общее время работы уменьшается всего на несколько процентов.

3.5.2. Использование алгебраического многосеточного метода совместно с CPR

Метод AMG и его модификации были детально рассмотрены в статьях [90], [91]. Авторами этих статей был реализован алгоритм на языке FORTRAN (код AMG1R1). Для применения в алгоритме CPR при решении задачи фильтрации алгоритм был реализован на языке C++ с изменениями способа выбора

укрупненной сетки, описанными в [92], [93], и призванными расширить область применения AMG на сильно несимметричные матрицы.

Тем не менее, применение алгоритма AMG для реальных наборов данных столкнулось с проблемами нестабильной сходимости метода при сильном отличии решаемой системы от той, что получается при аппроксимации эллиптических задач. Эта проблема отмечалась и другими авторами, см., например, [94]. Для определения области применимости алгоритма AMG был проведен набор тестовых расчетов для простейшей задачи $-div(k\nabla u) + \lambda u = f$ с нулевыми краевыми условиями на области $[0,1]$ x $[0,1]$, где для аппроксимации берется равномерная сетка, число узлов по x и по y равно 4096.

В таблицах ниже представлены результаты работы AMG в сравнении с методом BCGS с предобуславливателем ILU(0) (см. 3.4) для нескольких вариантов коэффициента k.

Пример 1. k - матрица следующего вида:

$$k = \begin{pmatrix} 1 & 0 \\ 0 & w \end{pmatrix}$$

При различных значениях параметра w сравним скорости работы алгоритмов. В AMG используется V - цикл. В таблице 3.3 представлено сравнение времени работы и количества итераций методов. Число итераций указано в скобках после указания времени работы.

w	3	100	500	2000	10000	40000
AMG	2(10)	2(8)	1(7)	1(7)	1(7)	1(7)
BCGS+ILU	8(53)	13(87)	23(148)	36(228)	61(373)	80(471)

Таблица 3.3. Сравнение алгоритмов AMG и ILU на примере 1

Таким образом, с возрастанием параметра w алгоритм AMG все больше и больше превосходит по скорости работы метод BCGS + ILU.

Пример 2. Теперь рассмотрим случай W - цикл. k - матрица следующего вида:

$$k = \begin{pmatrix} 1 & \varepsilon \\ -\varepsilon & 1 \end{pmatrix}$$

В таблице 3.4 представлено сравнение времени работы и количества итераций методов. Число итераций указано в скобках после указания времени работы.

ε	0.001	0.01	0.03	0.035	0.038	0.042
AMG	4(10)	5(10)	7(16)	11(24)	20(40)	расходится
BCGS+ILU	10(69)	10(68)	12(71)	12(81)	13(85)	21(138)

Таблица 3.4. Сравнение алгоритмов AMG и ILU на примере 2

При малых ε AMG превосходит по скорости BCGS + ILU, однако при увеличении параметра ε AMG постепенно теряет свои преимущества и начинает проигрывать. Далее алгоритм AMG начинает расходиться, тогда как BCGS + ILU продолжает сходиться.

Пример задачи фильтрации. Рассмотрим SPE 9 — синтетический трехфазный тест небольшого размера, предлагаемый Сообществом Инженеров Нефтяников (Society of Petroleum Engineers, SPE) как эталонный для тестирования работоспособности программ расчета задачи фильтрации, см. [95].

Расчет производится для 899 дней. «Отчетные» временные шаги достаточно большие; при расчете разделяются на более мелкие «расчетные». Длины «отчетных» шагов представлены в таблице 3.5.

Номер шага	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Длина, дней	1	1	1	47	50	100	100	60	60	60	60	60	60	60	179

Таблица 3.5. Сравнение алгоритмов AMG и ILU на примере 1

Расчет стартует с состояния равновесия.

Параметры будем варьировать следующим образом:

1. длину шага по времени ограничим значением 10 дней (ввиду того, что величина расчетного шага выбирается программой автоматически, в частности не длиннее текущего «отчетного» шага, данное значение может и не достигаться),

2. ускорение свободного падения положим равным 0,

3. скважины удалим,

4. капиллярное давление положим равным 0.

Данные упрощения применим в различных сочетаниях к модели и решаем ее методом CPR. В качестве алгоритма решения локальной системы используем AMG или BCGS+ILU(0).

Результаты тестирования приведены в таблице:

Номер варианта	0	1	2	3	4	5	8	9	10	11	12	13
Ум. макс. дл. ш.	−	−	−	−	−	−	+	+	+	+	+	+
Откл. гравитации	−	−	−	−	+	+	−	−	−	−	+	+
Удаление скважин	−	−	+	+	−	−	−	−	+	+	−	−
Выкл. кап. давл.	−	+	−	+	−	+	−	+	−	+	−	+
CPR+AMG, шагов	4	4	4	15	4	4	10	10	4	15	6	4
CPR+ILU, шагов	15	15	15	15	15	15	15	15	15	15	15	15

Таблица 3.6. Результаты работы алгоритма CPR в комбинации с AMG и ILU(0)

Знак «+» означает наличие упрощения, а «-» - отсутствие. В двух последних строках показано число пройденных отчетных шагов до первого случая расходимости. Если указано 15 отчетных шагов – расчет дошел до конца модели. Варианты (6, 7, 14, 15, соответствующие удалению скважин и отключению гравитации), являются тривиальными, поскольку для них решением задачи является нулевое приближение в методе Ньютона.

Расходимость во всех вариантах, где она имела место, была вызвана отсутствием сходимости локального алгоритма, поскольку замена AMG на

BCGS+ILU(0) приводит к тому, что расчет доходит до конца модели (последняя строка таблицы).

Тем не менее AMG работоспособен на версиях теста с некоторыми комбинациями упрощений. Во всех вариантах модели он сумел сделать первые 4 коротких шага, на которых производится ввод всех скважин. Это говорит о влиянии длины шага на сходимость.

Кроме того, когда упрощением было только ограничение по длине шага (вариант 8), алгоритм посчитал больше шагов (10), чем когда к нему добавлялось удаление скважин (вариант 10) – 4 шага, или «отключение гравитации» (вариант 12) – 6 шагов. Это обусловлено тем, что усложнение матрицы приводит к тому, что метод автоматического вычисления шага не позволяет расчету сразу «разогнаться» до длины в 10 дней.

Отметим, что на некоторых достаточно сложных моделях алгоритм CPR + AMG все же сходится, причем без применения вышеописанных упрощений. При этом число итераций практически совпадает с таковым для комбинации CPR+ILU.

Выводы. Поскольку матрица в задаче фильтрации более похожа на матрицу из примера 2, алгоритм AMG требует дополнительной настройки для успешной работы с такими матрицами, см., например, [96]. Сам алгоритм CPR может обгонять «чистый» алгоритм ILU (см. раздел 3.4), но требует подбора параметра точности локального алгоритма.

3.6. Выводы к третьей главе

В третьей главе описана предложенная параллельная реализация блочного варианта построения предобуславливателей класса ILU (ILU(0), ILU(1), ILUT) для итерационных методов решения систем с разреженными матрица-

ми, возникающими при аппроксимации систем дифференциальных уравнений в частных производных, описывающих фильтрацию многокомпонентной смеси в пористой среде. Предложенная реализация предобуславливателей хорошо сохраняет их качество и обеспечивает значительное ускорение по сравнению с последовательной версией при увеличении числа используемых логических процессоров. Проведены численные эксперименты с использованием различных матриц, полученных при дискретизации задач с реальными данными нефтяных месторождений в комплексе программ tNavigator, предложенном в работе, которые подтверждают хорошую масштабируемость предложенного алгоритма.

Глава 4

Гибридный алгоритм распараллеливания решения задач фильтрации

Решению задачи фильтрации многокомпонентной смеси в пористой среде на параллельных ЭВМ посвящено большое количество работ. В ранних работах ([97], [98]) еще рассматриваются векторные и смешанные векторно-параллельные компьютеры и ускорение в 3–4 раза считается очень хорошим. В работах [99], [100] используются метод Nested Factorisation — приближенные покоординатные разложения матрицы на матрицу, задающую фильтрацию по вертикали (по оси Z) и по горизонтали (в плоскости XY), последняя затем приближенно факторизуется на фильтрацию по X и Y. Ускорение работы этого подхода, реализованного в пакете Eclipse (компания Schlumberger), показано на рис. 1.12, *а* и не является оптимальным. В этом же пакете также реализован описанный в [101] алгоритм распараллеливания, основанный на комбинации двух предобуславливателей: из [99] и из [102]. В работе [103] также использовано два предобуславливателя: ILU(0) и блочный вариант SOR из-за недостаточной эффективности использованного ILU(0) (применяется в пакете ATHOS компании IFP Group (Beicip-Franlab)). В работах [104], [105] использован предобуславливатель, основанный на неполной аппроксимации многочленами Неймана, и метод обобщенных сопряженных невязок. В работе [106] решается система, полученная после полунеявной (IMPES) аппроксимации, и использованы предобуславливатели SOR и ILU с красно-черным упорядочиванием строк (применяется в пакете Falcon компании Amoco). В работе [107] используется последовательный алгоритм построения ILU разложения, который в силу наличия нераспараллеленного участка очень неэффективен (применяется в пакете IMEX компании CMG). В работе [108] исполь-

зуется алгоритм CPR совместно с алгебраическим многосеточным методом, ускорение работы этого подхода, реализованного в пакете Intersect (компания Schlumberger), показано на рис. 1.12, *б* и не является оптимальным. Аналогичный подход используется в [89], где время работы на параллельных ЭВМ даже не приводится (используется в пакете Tempest компании Roxar). В работе [109] рассматривается задача размерности 2.5D – предполагается возможность декомпозиции задачи на плоскую и вертикальную составляющие. Это ограничивает применимость этого подхода, реализованного в пакете POWERS (компания Saudi Arabian Oil Company).

В качестве программного интерфейса для параллельных ЭВМ обычно использовались OpenMP ([110]) для систем с общей памятью (в работах [103], [107]) и MPI ([111], [4]) для систем с распределенной памятью.

Ниже мы рассмотрим эффективную технологию, включающая алгоритмы, структуры данных и комплекс программ, для построения гибридного MPI–многопоточного вычислительного процесса для решения задачи фильтрации на ЭВМ с распределенной памятью (кластере).

4.1. Иерархическое строение современных вычислительных кластеров

За последние несколько лет строение вычислительных кластеров претерпело значительные изменения, которые по масштабам являются самыми большими за 20-летнюю историю применения параллельных вычислений на таких системах. Каждый узел современного кластера сам является большим многопроцессорным компьютером, причем с неоднородным доступом к памяти (NUMA, см. [112]). Не учет этих изменений при разработке программного обеспечения ведет к значительным потерям в производительности и слабому использованию ресурсов оборудования.

4.1.1. Мощные многопроцессорные узлы

Кластер представляет из себя набор компьютеров (называемых узлами кластера), соединенных высокопроизводительными каналами связи. Каждый из узлов обычно имеет два или (реже) четыре физических процессора и свою оперативную память, в которую загружен свой экземпляр операционной системы. Доступ к памяти других узлов идет посредством линии связи, а система выглядит как единое целое с точки зрения пользователя, взаимодействующего с т.н. управляющим узлом, который поддерживает очередь заданий на исполнение и распределяет задачи по узлам.

Такое определение кластера действует последние 20 лет и не поменялось и в настоящий момент. Изменились два (или четыре) физических процессора, установленные в узле: они стали многоядерными. Фактически внутри одного физического корпуса процессора расположены несколько процессоров, называемых ядрами, которые доступны операционной системе как «обычные» процессоры. В настоящий момент производятся варианты с 4, 6, 8, 10 и 12 ядрами и это количество постоянно растет, поскольку оно стало главным источником повышения производительности физического процессора из-за проблем с повышением его тактовой частоты. Более того, часто каждое ядро имеет по два комплекта всех вычислительных блоков и для оптимального их заполнения вводят многопоточность (hyperthreading), позволяя одному физическому ядру исполнять два потока данных. Это удваивает количество «процессоров», доступных операционной системе и работающим приложениям. Для единообразия вводят понятие «логического процессора» — это «процессор» предоставляемый операционной системе и работающим приложениям, без относительно его физической природы. Например,

- Система с 1-м физическим процессором Intel Core i7-980X: 6 ядер, 12 логических процессоров;

- Система с 2-мя физическими процессорами Intel Xeon 5650: 6 ядер в каждом, 12 логических процессоров в каждом, 24 логических процессоров в системе;

- Система с 4-мя физическими процессорами Intel Xeon 7560: 8 ядер в каждом, 16 логических процессоров в каждом, 64 логических процессоров в системе.

Последние две системы часто используются для построения узлов вычислительных кластеров.

Для программирования кластеров де-факто стандартом стал интерфейс MPI ([111], [4]). Пока логических процессоров на узле было 2, накладные расходы на обмен сообщениями между процессами, работающими на одном узле, были минимальны. Учет того, что некоторые процессы могут взаимодействовать между собой намного эффективнее, слишком усложнял и без того тяжелую в разработке MPI программу, не принося заметного ускорения в ее работе. Все изменилось, когда узлы стали состоять из более, чем десятка логических процессоров. Потери в производительности стали составлять уже разы, и это уже нельзя было не замечать. Поэтому возникла идея разработки гибридных приложений: взаимодействие между узлами осуществляется с помощью интерфейса MPI, а внутри узла программа работает, как на системе с общей памятью, используя интерфейс OpenMP ([110]) или потоки исполнения, предоставляемые непосредственно операционной системой ([4]).

Выгоды от такого подхода очевидны. Рассмотрим, например, кластер из 20-ти узлов с 12-ю логическими процессорами в каждом, всего 240 процессоров. При традиционном подходе на этой системе надо было использовать 240 MPI процессов, что обычно уже сталкивается с проблемами масштабируемости задачи. При гибридном подходе надо использовать 20 MPI процессов (по числу узлов) и по 12 потоков исполнения на каждом узле.

4.1.2. Неоднородный доступ к памяти

В течении длительного времени узлы кластера представляли из себя симметричную многопроцессорную систему (SMP) с общим однородным доступом к памяти (UMA - Uniform Memory Access). Однако, появление многоядерных процессоров сделало главное достоинство таких систем – общую шину к памяти, ее главным недостатком. Эта шина является тем «бутылочным горлышком», через который не могут пройти данные, требуемые для загрузки работой все большего числа логических процессоров, см. рис. 4.1, где тройными красными стрелками показан канал, связывающий подсистему оперативной памяти и процессоры.

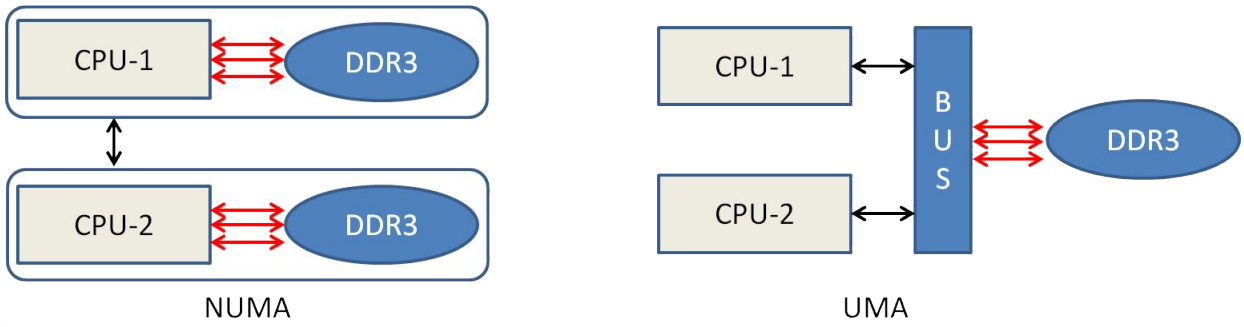


Рис. 4.1. Сравнение архитектур NUMA и UMA

Для решения этой проблемы подсистему оперативной памяти подключили непосредственно к физическим процессорам, см. рис. 4.1, а для эмуляции общей памяти процессоры связаны высокоскоростным каналом (черная одиночная стрелка на рис. 4.1). Так возникает неоднородность в устройстве памяти: данные из «своей» памяти (подключенной непосредственно к нему) процессор берет с одной скоростью, а из «чужой» (подключенной к другому процессору) – с почти вдвое меньшей, так как данные проходят еще и по линии связи между процессорами.

Заметим, что замедление доступа к «чужой» памяти вдвое — это очень хороший показатель. В истории параллельных ЭВМ есть много систем, где этот

показатель мог составлять десятки ([113]), что сделало название NUMA по отношению к параллельной ЭВМ негативным. Для таких систем программирование с помощью MPI было самым эффективным, поскольку планирование процессов в операционных системах (см., например, [30]) берет на себя их распределение по NUMA узлам.

Возвращаясь к узлу современного кластера заметим, что многопоточное (или OpenMP) приложение, не учитывающее специфику NUMA, замедляется как раз на время доступа к «чужой» памяти, т.е. в 2 раза, поскольку не оптимизировало обращения к «чужой» и «своей» памяти.

4.1.3. Оптимальное строение приложения для современных вычислительных кластеров

Сформулируем, как же должно выглядеть приложение для современных кластеров.

- Это должно быть 3-х уровневое гибридное приложение:
  - **MPI интерфейс** для взаимодействия процессов между узлами.
  - **Потоки исполнения** для распределения вычислительной нагрузки между логическими процессорами узла.
  - Учет архитектуры NUMA — минимизация обращения к памяти другого **физического процессора** (к памяти другого логического процессора, находящегося в одном корпусе с рассматриваемым, обращение идет быстро).
- Должно быть обеспечено равномерное распределение вычислительной работы и данных на всех трех уровнях: одинаковая загруженность узлов, потоков исполнения на узле, каждого **физического процессора** (иногда называемого NUMA-узлом).

- Должен быть единый простой интерфейс для обмена данными, вне зависимости от того, поток это или процесс; иначе программу такой высокой сложности почти невозможно отладить.

Идея использования гибридного подхода к написанию вычислительных приложений для иерархических систем не нова, см. [114], [115], [116], [117], [118], [119], [120], [121]. Но сложность задачи фильтрации многокомпонентной смеси в пористой среде делают ее реализацию крайне сложной и, как отмечалось выше, имеющиеся пакеты для решения этой задачи используют:

- либо MPI интерфейс для взаимодействия между **всеми** запущенными процессами (в том числе на одном узле); отметим, что в современных операционных системах (ОС) (Linux или Windows 7) в этом случае учет NUMA устройства узла произойдет автоматически (так как работающему **процессу** ОС строит отображение виртуальной памяти на физическую с учетом NUMA);

- либо OpenMP интерфейс и потому ограничены одним узлом (нет эффективного взаимодействия между узлами); отметим, что некоторые реализации OpenMP учитывают NUMA устройство узла ([122]).

Ниже мы рассмотрим эффективную технологию, включающая алгоритмы, структуры данных и комплекс программ, для построения 3-х уровневого гибридного MPI–многопоточного вычислительного процесса для решения задачи фильтрации на ЭВМ на кластере с NUMA узлами. Предложенная технология реализована в комплексе программ tNavigator, предложенном в работе.

4.2. Гибридная 3-х уровневая технология построения вычислительного процесса решения задачи фильтрации

Опишем основные этапы работы комплекса программ tNavigator, предложенного в работе.

- **Запуск MPI процессов по числу узлов кластера**. Поскольку в дальнейшем из каждого MPI процесса будут запущены потоки исполнения по числу логических процессоров, то требуется использовать реализацию MPI-2, допускающую такую возможность.

- **Загрузка входных данных на каждом из узлов**. Поскольку канал ввода существенно менее производительный, чем единичное ядро узла, загрузка данных для всех логических процессоров на узле будет производиться одним из них. Это уменьшает как количество синхронизаций, так и трафик по каналу ввода. Это уже отличает гибридное приложение от традиционных MPI решений, где каждый из процессов должен считывать данные для себя (или получать от других процессов по коммуникационному каналу).

  При загрузке данных важно обеспечить, чтобы

  – в памяти узла даже кратковременно не находились бы все данные задачи, а только предназначенная ему часть (иначе из-за большого объема данных они могут не поместиться в оперативной памяти узла),

  – вычислительная работа между узлами была бы распределена равномерно.

  Указанные требования для задачи фильтрации удовлетворить не просто из-за того, что по входным данным задачи нельзя без дополнительных

вычислений определить объем работы для узла, а для проведения вычислений надо данные уже загрузить, т.е. разделить между узлами, см. раздел 4.3. Эта проблема не решена в имеющихся пакетах решения задачи фильтрации. Ниже, в разделе 4.3 описана технология, позволяющая достигнуть оптимальной балансировки загруженности узлов без «пика» по требуемой алгоритму памяти.

- **Выделение памяти под данные на узле с учетом неоднородного доступа (NUMA)**. Загружаемые данные требуется разместить в оперативной памяти так, чтобы при последующем решении задачи минимизировать обращение каждого логического процессора к памяти «чужого» NUMA узла, см. раздел 4.4 ниже.

- **Построение графа связей MPI процессов** и определение порядка MPI пересылок, при котором все процессы могут обменяться данными со связанными с ними (в силу стандартного перекрытия и конструкции предобуславливателя, см. 3.3.2) процессами. При этом все потоки, «составляющие» MPI процесс на одном узле, участвуют как одно целое, формируя объединенный узел графа. Это радикально

  – уменьшает количество MPI обменов путем группировки транзакций от всех потоков в одну,

  – уменьшает объем MPI обменов из-за уменьшения размера зоны перекрытия (так как потоки исполнения на одном узле «просто» обращаются к общей памяти друг друга, не формируя интерфейсную зону).

Эти факторы являются одними из главных составляющих ускорения комплекса программ tNavigator, предложенного в работе.

Для задачи фильтрации этот этап приходится делать на каждом шаге по времени из-за постоянно растущего количества скважин. Дело здесь в том (см. 1.4.1), что из-за наличия длинных горизонтальных стволов скважин (или больших водонапорных горизонтов) часто невозможно разделить задачу между узлами так, чтобы одна скважина (водонапорный горизонт) не попала в несколько MPI процессов. Это, приводит к

  - изменению графа связей между процессами и необходимости его перестройки,
  - изменению предобуславливателя, см. 3.3.2,
  - добавлению дополнительных MPI обменов в ходе решения уравнений для скважины.

- **Вычисление матрицы системы** выполняется параллельно и независимо всеми логическими процессорами системы из-за обычного перекрытия (хранения данных в блоках сетки хотя и из другого процесса, но связанными с блоками сетки на рассматриваемом узле). Отметим, что в гибридной реализации здесь тоже есть экономия в оперативной памяти из-за отсутствия дублирования информации между потоками на одном узле.

- **Итерационный алгоритм решения и предобуславливатель**. Построение предобуславливателя было подробно рассмотрено в главе 3. Выбор оптимального итерационного алгоритма решения для гибридного процесса решения рассмотрен ниже в разделе 4.5.

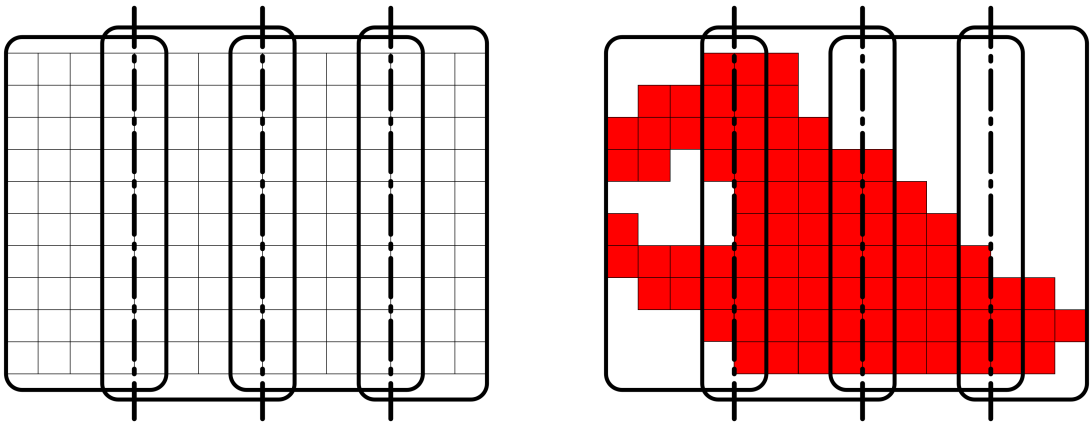
4.3. Балансировка загруженности узлов кластера при расчете задачи фильтрации

Для построения аппроксимации системы дифференциальных уравнений фильтрации (4) вводится сетка, учитывающая геологическое строение месторождения (разломы, выклинивания), см. пример построения в разделе 2.3. Далее будем называть упомянутую сетку исходной. Для решения задачи используются только данные для ячеек сетки, имеющих поровый объем (см. 1.1), больше определенного порогового значения. Такие ячейки сетки называются активными. Назовем расчетной сеткой совокупность активных ячеек исходной сетки. Данные в неактивных ячейках во время расчета не хранятся, но используются для построения расчетной сетки (например, пороговое значение для определения активности ячейки может зависеть от среднего порового объема по всем ячейкам).

Стандартный подход к разделению задачи между узлами кластера состоит в геометрическом разрезании области на части по числу вычислительных узлов. Этому разрезанию области соответствуют разделение исходной и расчетной сеток. В каждом MPI-процессе загружаются данные и выполняется вычислительная работа для ячеек сетки, соответствующих его части области. Из-за особенностей геологического строения месторождения часто невозможно разделить область на части так, чтобы в каждой части одинаково было количество ячеек исходной и расчетной сеток. Все имеющиеся пакеты расчета задачи фильтрации разделяют **исходную** сетку на одинаковые по количеству ячеек исходной сетки части, что идеально балансирует загруженность узлов кластера на этапе чтения входных данных. Однако, **расчетная** сетка при таком геометрическом разделении области оказывается распределена между узлами очень неоднородно. Может даже получится так, что в некоторых частях исходной сетки отсутствуют блоки расчетной сетки, и соответствующему узлу

кластера вообще не достается вычислительной работы. В этом случае большинство программных пакетов решения задачи фильтрации останавливают свою работу.

Целью настоящего раздела является построение алгоритма решения задачи фильтрации, который обладает достоинствами стандартного алгоритма при загрузке данных и эффективно разделяет вычислительную работу между узлами кластера при расчете.



(*а*). Разделение исходной сетки (*б*). Разделение расчетной сетки

Рис. 4.2. Стандартное разделение исходной и расчетной сеток

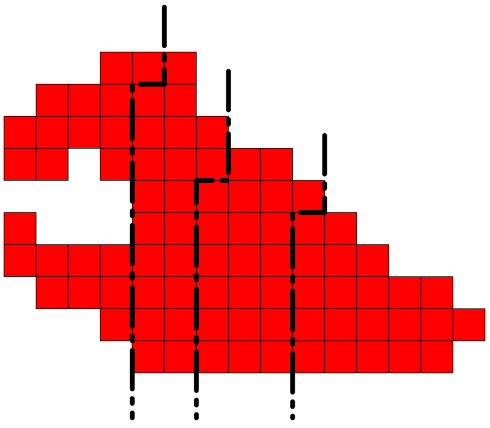


Рис. 4.3. Рассмотренное разделение расчетной сетки

Загрузка данных и построение расчетной сетки. Предлагается разделить загрузку данных исходной сетки и построение распределенной по узлам рас-

четной сетки на три этапа:

1. Исходная сетка делится равномерно между узлами с перекрытиями (как в традиционном подходе). На каждом узле осуществляется построение части расчетной сетки с вычислением графа связей с ячейками в других MPI-процессах. Схематически разделение показано на рис. 4.2, *а*, где маленьким квадратам соответствуют ячейки исходной сетки, пунктирные линии показывают разделение между процессами, а жирные линии объединяют ячейки, распределенные на каждый узел.

2. Осуществляется сборка карты соответствия ячеек расчетной и исходной сеток на каждом узле и на основании этой информации осуществляется вычисление номеров ячеек, которые попадут на каждый из узлов на следующем этапе.

3. Загрузка данных исходной сетки, с сохранением на каждом узле кластера данных только для ячеек расчетной сетки, распределенных на данный узел.

Из-за слоистой структуры месторождения обычно имеется сильное различие свойств по вертикали и в горизонтальной плоскости. Для эффективного построения графа связей желательно распределить ячейки с одинаковыми номерами по горизонтальным координатам x и y («столбики») на один вычислительный процесс, поэтому равномерно разделим область в ее проекции на плоскость xy. Для построения графа связей разделение осуществляется с перекрытием, чтобы ячейки, между которыми теоретически возможна связь, оказались на одном узле. Результат работы изображен на рис. 4.3, для сравнения на рис. 4.2, *б* показано разделение, полученное при стандартном разрезании. На рисунках маленьким квадратам соответствуют ячейки расчетной сетки, пунктирные линии показывают разделение между процессами, а жир-

ные линии объединяют ячейки, распределенные на каждый узел, на рис. 4.3 они не отображены, чтобы не уменьшить читаемость рисунка.

После выполнения первого этапа на основе загруженных данных можно вычислить, какие ячейки из распределенной на узел части исходной сетки попадают в расчетную сетку и построить граф связей. Этой информацией обмениваются все процессы, и в каждом из них строится список ячеек расчетной сетки, который сохраняется в текстовом файле, чтобы исключить первый этап вообще, если рассчитывается несколько задач с одной сеткой.

Третий этап состоит в построении необходимых для параллельного расчета структур. Перед чтением данных исходной сетки на основе полученной информации строится разделение ячеек между узлами кластера с учетом перекрытия между расчетными областями, необходимого для вычисления свойств фильтрационных течений между ячейками в разных MPI-процессах, и необходимые отображения для индексации ячеек расчетной сетки. Нумерация ячеек производится так, чтобы соединенные ячейки имели близкие номера для уменьшения ширины ленты решаемых в последующем систем. После нумерации ячеек разделение ячеек производится равномерно по номерам: каждому узлу соответствует некоторый отрезок номеров ячеек.

Чтение данных происходит по общей схеме на всех этапах. На каждом узле читаются все входные данные, а сохраняется в памяти только та часть, которая соответствует ячейкам этого узла на данном этапе алгоритма. Это позволяет использовать один и тот же код на всех узлах кластера

В предложенном алгоритме требуется производить чтение входных данных дважды, что позволяет иметь общий исходный код для первого и третьего этапов и избежать больших требований к оперативной памяти на каждом узле.

Отметим, что на всех этапах на каждом узле используется объем памяти, достаточный для сохранения геометрических параметров сетки и свойств среды только в ячейках сетки, распределенных на данный узел с учетом пере-

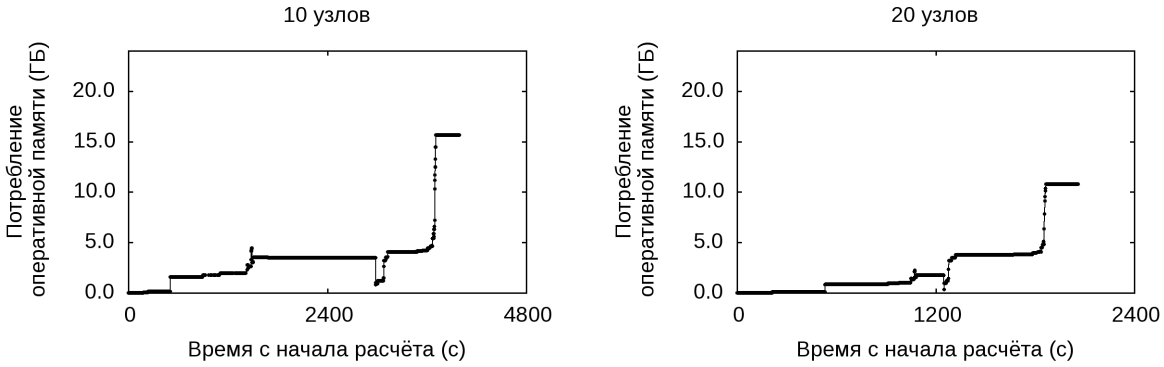
крытия.

Результаты тестирования. Описанный метод был протестирован в комплексе программ tNavigator на кластере из 20 параллельных ЭВМ с двумя процессорами Intel Xeon. Для тестирования была выбрана гидродинамическая модель Самотлорского месторождения на геологической сетке, содержащая 232 миллиона ячеек исходной сетки. Характеристики кластерной системы, на которой производилось тестирование:

- 20 узлов, на каждом из которых:
  - два шестиядерных процессора Intel Xeon X5650 с частотой 2.67GHz,
  - 24GB оперативной памяти DDR-III 1333MHz,
- соединение между узлами осуществляется Infiniband QDR с пропускной способностью 40 Gb/s.

Приложение было запущено в гибридном режиме, оптимизировано для многопроцессорных систем с неоднородной памятью, на каждом узле работал один MPI-процесс, в котором было 12 потоков исполнения.

Размер задачи позволяет ее рассчитывать только на 10 узлах и более из-за недостаточного количества оперативной памяти. Объем оперативной памяти, использованной программой на каждом из узлов, оказался практически одинаковым в каждый момент времени как на этапе загрузки данных, так и на этапе расчета. Диаграмма ее использования в зависимости от времени, прошедшего с начала расчета, показана на рисунке 4.4, *а* при расчете на 10 узлах, на рисунке 4.4, *б* — при расчете на 20 узлах. График зависимости пикового потребления оперативной памяти на узле от их числа показан на рисунке 4.5. Отметим, что максимальный объем выделенной оперативной памяти на узле



(*а*). Расчет на 10 узлах (*б*). Расчет на 20 узлах

По оси абсцисс — время в секундах от начала расчета, по оси ординат — используемая оперативная память в гигабайтах.

Рис. 4.4. Графики использования памяти на каждом узле

при увеличении числа MPI-процессов снижается: на 10 узлах он составляет 16,4 Гб, а на 20 узлах — 11,3 Гб.

Загруженность процессоров, измеряемая стандартно для UNIX-систем, на этапе чтения на всех узлах около единицы, потому что, как описывалось выше, в каждом MPI-процессе независимо происходит чтение входных данных. Диаграмма загруженности процессоров в начале расчета представлена на рисунках 4.6, *а* и 4.6, *б* для запуска на 16-ти и 20-ти узлах соответственно. Максимально достижимое значение равно 12, поскольку всего имеется 12 логических процессоров. Из диаграммы видно, что в среднем 8 процессоров на каждом из узлов постоянно занято вычислительной работой.

Результаты тестирования комплекса программ tNavigator показывают, что предложенный алгоритм балансировки загруженности позволяет эффективно рассчитывать очень большие задачи фильтрации на кластерных системах, обеспечивая равномерность как выделенной оперативной памяти, так и вычислительной нагрузки даже при сложном взаимном расположении исходной и расчетной сеток. Алгоритм использует из входных данных только геомет-

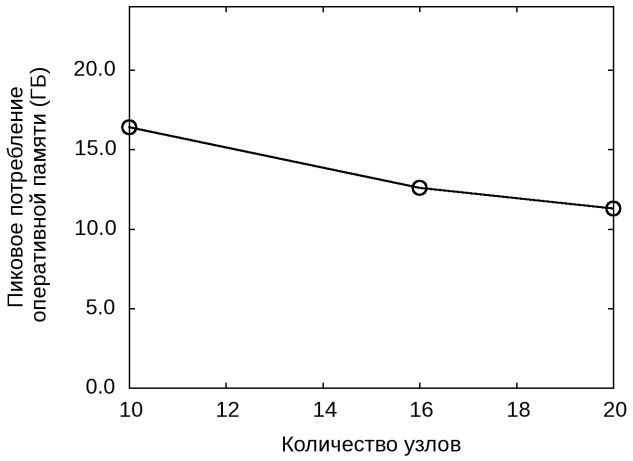
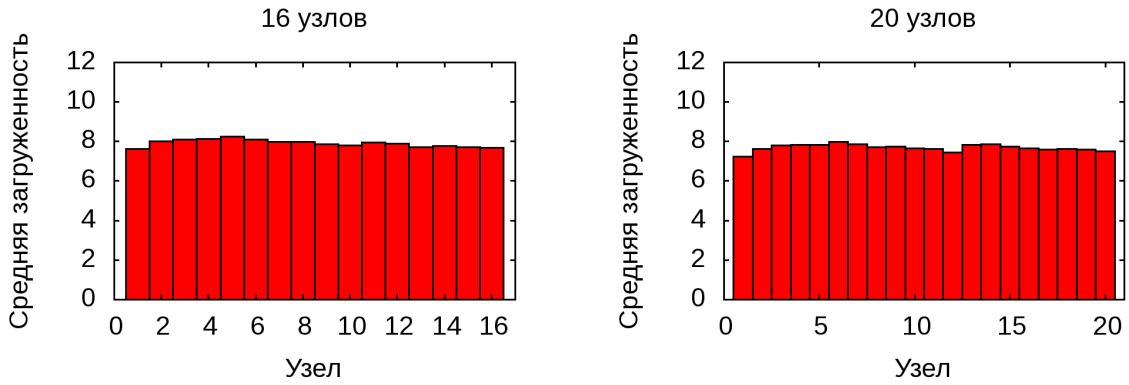


Рис. 4.5. График зависимости пикового потребления оперативной памяти на узле от их числа

рическую информацию, что позволяет распределить данные на ранних этапах их загрузки.

4.4. Оптимизация решения задачи фильтрации для многопроцессорных систем с общей неоднородной памятью

При использовании на каждом узле потоков с разделением данных между ними (гибридная схема организации вычислительного процесса) для учета разного времени обращения логических процессоров к разным участкам памяти (NUMA архитектура) требуется значительно менять код традиционного многопоточного приложения. Способы облегчения подстройки параллельного приложения под NUMA архитектуру уже обсуждались в литературе, см. [123], [124]. Ниже рассмотрена технология, использующая особенности управления памятью в операционных системах Linux и Windows 7, которая не требует значительного изменения программного кода многопоточного приложения и при этом максимально использует возможности оборудования.



(*а*). Расчет на 16 узлах (*б*). Расчет на 20 узлах

По оси абсцисс — номер узла, по оси ординат — средняя загруженность.

Рис. 4.6. Графики загруженности узлов

Подход к оптимизации вычислительных приложений для NUMA систем. Для начала рассмотрим параллельную многопоточную реализацию алгоритма перемножения матрицы на вектор ([4]). Скорость работы такой программы во многом зависит от скорости доступа к оперативной памяти. Это делает ее хорошим примером для демонстрации ключевых идей, лежащих в основе метода оптимизации программного кода для NUMA систем.

В традиционной программной реализации память под матрицы и вектор выделяется в главном потоке. На NUMA системе это приводит к тому, что физически матрица и вектор будут располагаться в локальной памяти одного NUMA узла, что увеличивает как минимум в два раза время доступа к ним потоков, работающих на другом узле.

Покажем, как можно оптимизировать программный код для NUMA систем, не изменив сам численный алгоритм, и получить ускорение $1.5-2$ раза.

Допустим, что размер матрицы равен $K = M \cdot T$, где T – число потоков, матрица хранится в памяти по строкам. Поток с номером i вычисляет результат произведения строк матрицы с номерами $M \cdot (i-1)+1, \ldots, M \cdot i$ на вектор. Будем говорить, что эти строки матрицы принадлежат i-тому потоку. Таким

образом, матрица делится на T непрерывных частей, каждая из которых обрабатывается собственным потоком.

Ключевая идея оптимизации заключается в реализации двух условий, которые позволят минимизировать число обращений к нелокальным участкам памяти:

- каждый из вычислительных потоков работает на фиксированном узле NUMA системы,
- те части матрицы и вектора, с которыми вычислительный поток оперирует большую часть времени, физически находятся в локальной памяти этого узла.

Для выполнения первого условия в каждом из вычислительных потоков сразу после его создания сделаем системный вызов, который привяжет этот i-тый поток к i-тому логическому процессору.

Для удовлетворения второго условия с минимальным переписыванием программы воспользуемся особенностью механизма выделения памяти в операционных системах, поддерживающих NUMA. В системах Linux и Windows 7 при выделении памяти посредством функции malloc отображение затребованной виртуальной памяти на физическую строится не сразу. Когда поток обращается к блоку виртуальной памяти, для которого еще не построено отображение, происходит страничный промах (page fault), который инициирует создание отображения запрошенного блока виртуальной памяти на локальную память того узла NUMA системы, с которого был произведен запрос.

В традиционном подходе выделения памяти в многопоточных приложениях главный поток запрашивает память у операционной системы и инициализирует ее, скажем, нулями. На NUMA системе это приводит к тому, что вся запрошенная память выделится в локальной памяти узла, на котором ра-

ботал главный поток. Остальные потоки в это время простаивают, дожидаясь завершения этой операции.

Модифицируем этот подход, чтобы его можно было эффективно использовать на NUMA системе. Будем выполнять операцию выделения памяти в два этапа. На первом этапе главный поток запрашивает у операционной системы память требуемого объема. Затем следует точка синхронизации, после которой указатель на затребованную память можно использовать во всех потоках. На втором этапе выделенная память параллельно инициализируется всеми потоками, причем каждый из потоков инициализирует только свою часть запрошенной памяти.

Предложенный метод инициализации памяти в два этапа эффективен по двум причинам. Во-первых, та часть памяти, к которой поток будет обращаться чаще всего, выделяется в локальной памяти узла NUMA системы, на котором он работает. Во-вторых, ускоряется операция ее инициализации за счет параллелизма в работе памяти. Минусом этого метода является наличие дополнительной точки синхронизации потоков.

Эти модификации затрагивают ту часть программного кода, которую можно назвать подготовительной стадией алгоритма — создание вычислительных потоков, заполнение матрицы и вектора. Тем самым, алгоритмически существенная часть программного кода, ответственная за непосредственное перемножение матрицы на вектор, остается нетронутой.

Используя эти идеи, представленный метод оптимизации кода для NUMA систем можно применить к любой многопоточной программе, в которой вычислительные потоки большую часть времени проводят, обращаясь к своей части всей выделенной процессу памяти. Получаемый выигрыш зависит от доли операций с памятью в общем времени работы программы.

Практическое применение подхода к оптимизации вычислительных приложений для NUMA систем. Перед тем, как приступать к оптимизации построения системы линейных уравнений в задаче фильтрации и ее решения (как частей, занимающих большую часть времени работы программы), необходимо ответить на несколько вопросов. Какие объекты, имеющие размер порядка K, назовем их большими, используются в программе? Можно ли каждый из этих объектов разбить на части, которые в основном обрабатываются отдельными потоками?

Первое множество больших объектов, обозначим его через T_1, включает в себя различные свойства фаз, компонент, породы в каждом блоке сетки:

1. Главные переменные решаемой задачи — молярные плотности компонент, давление в фазе нефть.

2. Все, что так или иначе входит в коэффициенты уравнения (4) — относительные проницаемости, вязкости, насыщенности фаз, пористость, проводимость, коэффициенты масштабирования фазовых проницаемостей и т.д. Этот список на практике получается гораздо длиннее приведенного и зависит от множества физический явлений, учитываемых в математической модели.

3. Производные вышеперечисленных свойств по главным переменным.

Эти данные, в основном, используются для вычисления коэффициентов системы линейных алгебраических уравнений (СЛАУ) (62). Это процедуру можно представить следующим циклом:

для каждого блока $i = 1, \ldots, K$,

используя данные i-того блока и геометрически соседних с ним,

посчитать коэффициенты СЛАУ, описывающих физику процесса в i-том блоке.

В многопоточном приложении множество всех блоков разбивается между M потоками — i-тый поток обрабатывает блоки с номерами $K_{i-1}, ..., K_i - 1$, где $1 = K_0 < K_1 < ... < K_M = K$. Как правило, разбиение получается таким, что геометрически соседние блоки обрабатываются одним потоком. Таким образом, i-тый поток при вычислении своей части коэффициентов линейной системы, в основном, будет обращаться к элементам в позициях $K_{i-1}, ...K_i - 1$ массивов размера K из множества T_1.

Система линейных уравнений (62) решается методом BCGS (Biconjugate Gradient Stabilized) ([43]) с использованием параллельных предобуславливателя класса ILU, описанных в главе 3. На этом этапе численные операции ведутся с другим множеством больших объектов, обозначим его через T_2. В него входят:

1. Матрица $\boldsymbol{A} = \dfrac{\partial \boldsymbol{F}(\boldsymbol{U}^m)}{\partial \boldsymbol{U}}$ (см. 2.5), хранящаяся в блочном MSR формате, описанном в разделе 2.3.

2. Матрица предобуславливателя, своя для каждого потока, располагающаяся в выделенной ему памяти.

3. Некоторое количество векторов длины K, используемых в методе BCGS.

При решении системы i-тый поток, в основном, оперирует со строками $K_{i-1}, ..., K_i - 1$ матрицы $\boldsymbol{A}$, с элементами в тех же позициях векторов длины K и со своей матрицей предобуславливателя.

Отмеченное свойство вычислительных потоков обращаться, в основном, только к своим частям массивов дает ответ на второй поставленный вопрос о том, «как разбить большие объекты на части». После нахождения ответов на поставленные два вопроса, оптимизация программного кода для NUMA систем становится такой же, как и в случае перемножения матрицы на вектор:

- использование системного вызова, привязывающего i-тый поток к i-тому логическому процессору.

- использование двухэтапной инициализации памяти под массивы из множеств T_1 и T_2, в которой каждый из потоков инициализирует только свою часть этих массивов.

Результаты тестирования. Описанный метод был протестирован в комплексе программ tNavigator на параллельной ЭВМ с двумя процессорами Intel Nehalem и на параллельной ЭВМ с четырьмя процессорами Intel Nehalem EX. Для тестирования был выбран набор данных для реального месторождения, содержащий *несколько миллионов* блоков расчетной сетки.

Тестовая система 1:

- процессор Intel Xeon E5520 2.27GHz, число процессоров = 2, число ядер = 4, число логических процессоров = 16,

- оперативная память DDR-III 1067MHz 12Gb.

а. Умножение матрицы $K \times K$ на вектор длины K.

- Размерность задачи $K = 20000$.

- Объем использованной памяти - 3052 Mb.

б. Решение системы линейных уравнений методом BCGS.

- Размер матрицы $K = 1094421$.

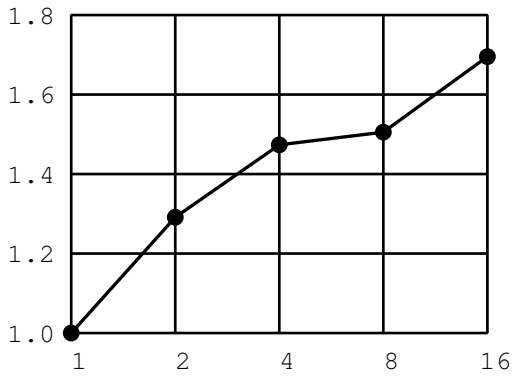
- Число ненулевых элементов в матрице = 7484949.

- Объем использованной памяти - 702 Mb.

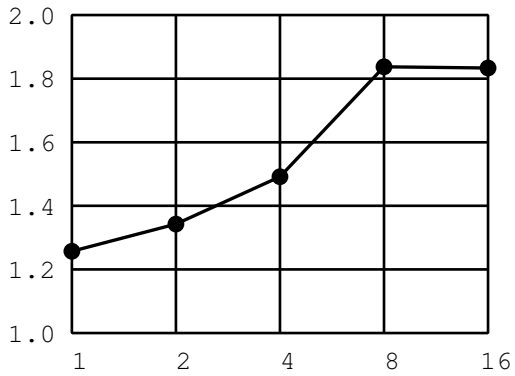
в. Расчет реального месторождения.

- Размерность задачи (число блоков в модели) $K = 1094421$.
- Объем использованной памяти - 1921 Mb.
- Время расчета с использованием одного потока - 1 час 2 минуты.

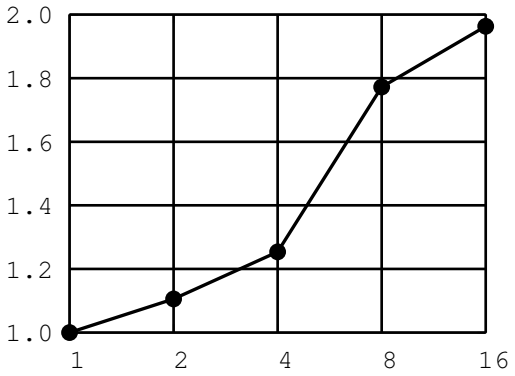
Графики ускорения работы программ после применения оптимизаций для NUMA систем показаны на рисунках 4.7, *а*, 4.7, *б*, 4.7, *в* соответственно.



(*а*). Умножение матрицы на вектор



(*б*). Решение системы линейных уравнений



(*в*). Расчет реального месторождения

Рис. 4.7. Графики ускорения работы программ после применения оптимизаций для NUMA систем в зависимости от числа потоков

Тестовая система 2:

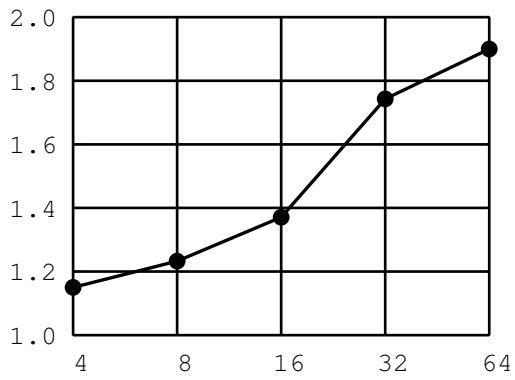
- процессор Intel Xeon X7560 2.27GHz, число процессоров = 4, число ядер = 8, число логических процессоров = 64,

- оперативная память DDR-III 1067MHz 128Gb.

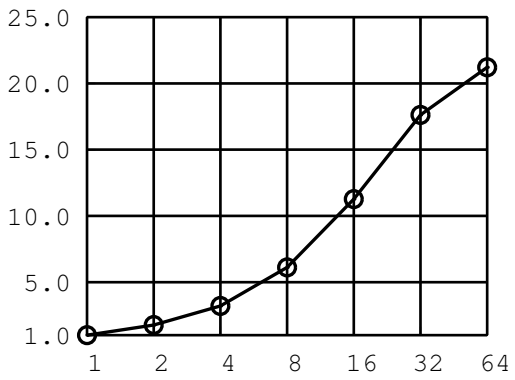
Расчет реального месторождения.

- Размерность задачи (число блоков в модели) $K = 2418989$.
- Объем использованной памяти - 6133 Mb.
- Время расчета с использованием одного потока - 26 часов 47 минут.

График ускорения работы программ после применения оптимизаций для NUMA систем показан на рисунке 4.8, *а*, масштабируемости параллельного расчета — на рисунке 4.8.



(*а*). Ускорение работы программы после применения оптимизаций для NUMA систем



(*б*). Масштабируемость параллельного расчета

Рис. 4.8. Расчет реального месторождения, на горизонтальной оси отложено число потоков

Результаты последнего теста показывают, что оптимизация программы для NUMA системы позволила достичь при решении задачи фильтрации ускорения 21 на системе с 32-мя физическими ядрами, в то время, как ускорение неоптимизированной версии равно 11.

Заключение. Предложен подход, позволяющий значительно сократить время на разработку эффективных вычислительных приложений для NUMA систем. Результаты численных экспериментов с комплексом программ tNavigator показывают, что при использовании всех физических ядер параллельной ЭВМ оптимизированное для NUMA систем приложение работает в $1.5 - 2$ раза быстрее, чем его исходная версия.

4.5. Выбор оптимального итерационного метода решения систем линейных уравнений в задачах фильтрации

Для гибридной MPI-многопоточной программы нахождения решения задачи фильтрации выбор оптимального итерационного метода для решения систем линейных уравнений с разреженной матрицей является непростой задачей. Слишком много параметров влияют на суммарную производительность:

- количество точек синхронизации (когда все работающие процессы и(или) потоки должны обменяться информацией);
- объем пересылаемых данных между процессами;
- объем оперативной памяти, требуемый алгоритму (поскольку оперативная память – намного более медленный ресурс, чем процессор);
- адаптивность алгоритма – его способность подстраиваться под входную матрицу, поскольку решается много разных систем (на каждом шаге по времени и на каждой итерации метода Ньютона) и выбирать набор параметров, оптимальный для каждой из них, невозможно.

При этом производительность надо измерять при разном количестве использованных узлов и логических процессоров на них, чтобы оценить масштабируемость алгоритмов. При таком количестве влияющих факторов теоретическое

сравнение алгоритмов вряд ли возможно. Поэтому было проведено практическое сравнение трех основных алгоритмов решения систем линейных уравнений с несимметричной разреженной матрицей: BCGS, ORTHOMIN, GMRES (последний в двух вариантах — QGMRES, DQGMRES) (см. [43]) по скорости работы и ускорению на параллельных ЭВМ: на SMP-системах, на системах с распределенной памятью и на гибридных системах (системах с распределенной памятью, состоящих из SMP-узлов). Проведено также исследование масштабируемости каждого из алгоритмов (время работы алгоритма сравнивается с временем работы его же на другой конфигурации узлов/потоков) и сравнение алгоритмов между собой (время работы алгоритма сравнивается с временем работы другого алгоритма на той же конфигурации узлов/потоков). Все алгоритмы были реализованы в комплексе программ tNavigator с максимальными оптимизациями для параллельных ЭВМ, включая учет неоднородности доступа к памяти (NUMA), см. раздел 4.4. В качестве предобуславливателя использовался параллельный вариант предобуславливателя ILU(0) с «алгебраическим» разбиением матрицы, показавший наилучший результат на матрицах, полученных при аппроксимации фильтрационных течений для ряда месторождений, см. главу 3.

В качестве тестовых были выбраны матрицы, полученные при аппроксимации задачи фильтрации с данными реальных нефтяных месторождений в комплексе программ tNavigator. Характеристики матриц, участвующих в тестировании, приведены в таблице 4.1.

Системы решались на кластере из 16 двухпроцессорных узлов, с шестиядерными процессорами Intel Xeon X5650 с частотой 2.67GHz и соединенных InfiniBand QDR с пропускной способностью 40Gbit/s. Это достаточно типичная конфигурация для современных систем.

№	число уравнений	ненулевых элементов	макс. ненулевых в строке	ширина ленты матрицы	коэффициент кососимметричности
1	2188842	30044852	175	10094	$1,9 \cdot 10^7$
2	2400402	29509724	93	36096	$1,3 \cdot 10^6$
3	2715768	34268312	13	12716	$2,8 \cdot 10^7$
4	815932	10941448	89	14228	$4,8 \cdot 10^6$
5	1023378	19898604	118	10500	$1,4 \cdot 10^7$
6	858382	11691772	135	15508	$5,2 \cdot 10^4$
7	733713	14319189	58	10167	$1,6 \cdot 10^5$
8	1324920	25038036	64	8868	$3,3 \cdot 10^5$
9	679262	8767636	43	38600	$1,2 \cdot 10^6$
10	2018124	31892472	64	14466	$5,3 \cdot 10^6$
11	14024025	275653395	58	35334	$2,1 \cdot 10^8$

Таблица 4.1. Характеристики тестовых матриц

Особенности реализации алгоритмов. Для гибридного MPI-многопоточного распараллеливания векторы и матрица делились поровну между всеми потоками исполнения. При вычислении скалярного произведения, линейной комбинации векторов или произведения матрицы и вектора каждый поток считал свою часть. Далее, в случае вычисления скалярного произведения, результат получался посредством общей MPI-синхронизации. Перед каждой операцией умножения вектора на матрицу и применения предобуславливателя происходил MPI-обмен, в ходе которого каждый узел получал элементы вектора, необходимые для подсчета его части в соответствующей операции.

Поскольку каждый узел является NUMA-системой, для повышения быстродействия при распараллеливании на уровне узла были применен подход из раздела 4.4: каждый поток был привязан к конкретному физическому ядру и процессору, и именно в банке этого процессора выделялась память, с которой этот поток преимущественно работал.

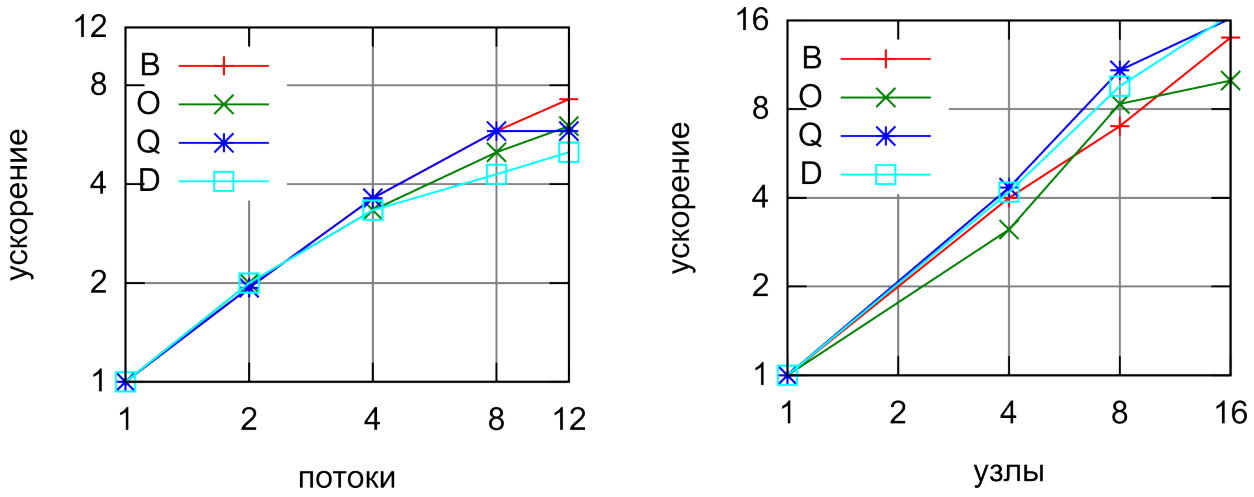
Подбор параметров алгоритмов. Были проведены тестовые расчеты для определения оптимального параметра k в алгоритмах QGMRES(k), DQGMRES(k) и ORTHOMIN(k). Наилучшим значением для всех трех алгоритмов оказалось значение $k = 5$ и параметр рестарта 50.

Из двух отмеченных в [43] возможностей реализации частичной ортогонализации в этих алгоритмах — подсчет скалярных произведений со всеми векторами базиса с последующим вычислением полной линейной комбинации или последовательная ортогонализация относительно каждого вектора базиса — был выбран первый вариант ввиду меньшего количества синхронизаций ($O(i)$ против $O(ik)$, где i — количество итераций), что существенно для системы с распределенной памятью. Правильность выбора была подтверждена тестовыми расчетами на выборке матриц из табл. 4.1.

Результаты тестирования. Усредненные по матрицам графики ускорения алгоритмов при запуске на одном узле с разным количеством потоков и с одним потоком на разном количестве узлов показаны соответственно на рисунках 4.9, *а* и 4.9, *б*. Здесь и далее на графиках буквой B обозначается алгоритм BCGS, O — ORTHOMIN, Q — QGMRES, D — DQGMRES. Из рисунка 4.9, *б* видно, что ускорение как функция числа узлов практически линейно (ускорение близко к идеальному). При использовании одной SMP системы (рис. 4.9, *а*) ускорение линейно только при небольшом числе потоков из-за недостаточной полосы пропускания оперативной памяти.

На рисунке 4.10 показан результат расчетов в разных комбинациях числа узлов и числа потоков исполнения на узле. На рисунке 4.10, *а* показана зависимость ускорения от числа потоков, использующихся при расчете на 16-ти узлах, а на рисунке 4.10, *б* — от числа узлов при использовании всех 12-ти потоков на каждом узле.

В этом случае ускорение невелико: видимо, вычислительные возможности узлов превосходят возможности интерфейса. Оптимальное количество узлов для решения систем с указанными матрицами находится в пределах от 4-х до 8-ми. Замедление при включении всех потоков на всех процессорах связано с деградацией предобуславливателя при большом количестве потоков



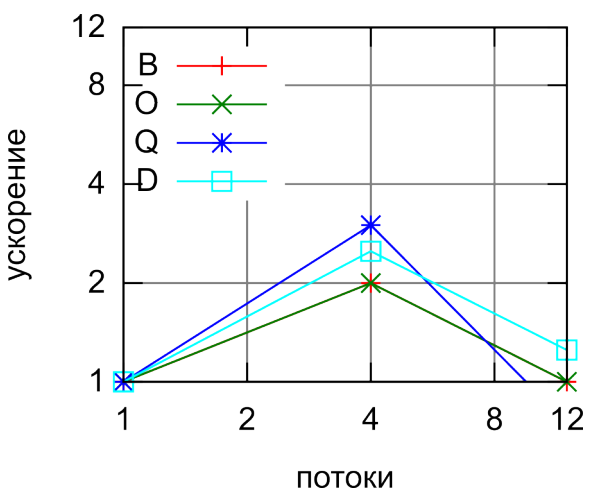
(*а*). Один узел, разное количество потоков (*б*). Один поток, разное количество узлов

Рис. 4.9. Ускорение при негибридном распараллеливании

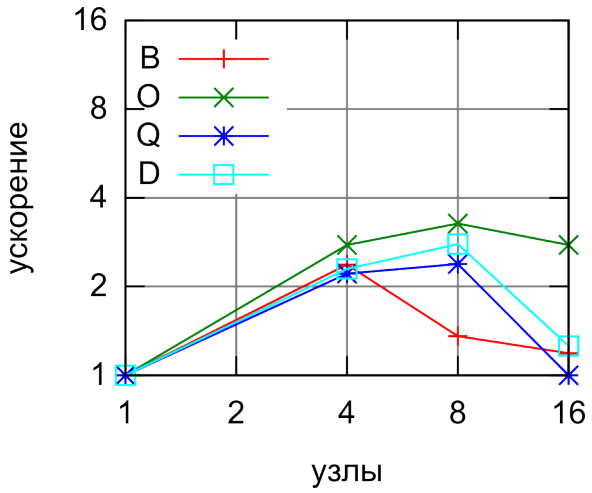
исполнения. Тестируемые матрицы слишком малы для кластера: действительно, для самой большой матрицы (14 млн. уравнений), каждый поток работает всего с несколькими мегабайтами памяти.

Соотношение скорости работы алгоритмов: на графиках (рис. 4.11) представлено отношение скорости алгоритмов ORTHOMIN, QGMRES и DQGMRES к BCGS в зависимости от количества узлов на одном потоке и на 12-ти потоках. Алгоритмы показывают сопоставимое время, но, как правило, быстрейшим оказывается BCGS.

Заключение. Для ряда больших несимметричных разреженных матриц, полученных при аппроксимации задачи фильтрации в предложенном в работе комплексе программ tNavigator, проведено тестирование алгоритмов BCGS, ORTHOMIN, QGMRES, DQGMRES на параллельных ЭВМ при использовании от 1-го до 192-х потоков исполнения. В среднем лучше всего как с точки зрения масштабируемости, так и абсолютного времени решения, проявил себя алгоритм BCGS.



(*а*). 16 узлов, разное количество потоков на каждом узле



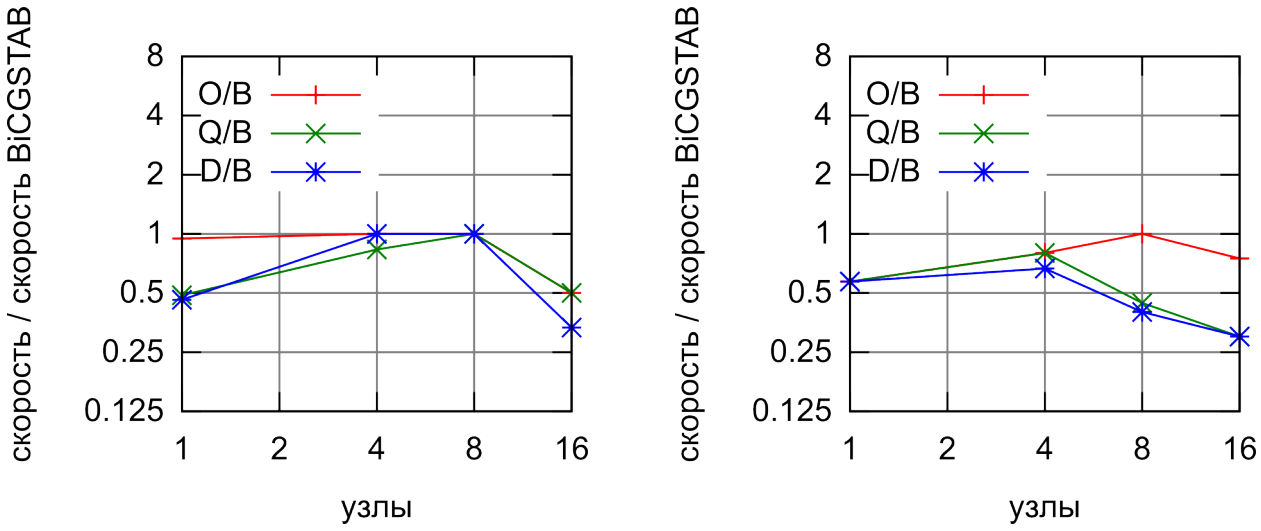
(*б*). 12 потоков на каждом узле, разное количество узлов

Рис. 4.10. Ускорение при гибридном распараллеливании с использованием всех имеющихся (а) узлов и (б) ядер

4.6. Численные эксперименты

Напомним кратко, какие шаги были описаны ранее для того, чтобы максимально эффективно решать систему линейных уравнений с матрицей, получающейся при аппроксимации задачи фильтрации (4) на каждом шаге по времени и на каждой итерации метода Ньютона (62):

- выбран оптимальный метод хранения матрицы в разделе 2.3;
- выбран оптимальный предобуславливатель в главе 2;
- выбран оптимальный способ распараллеливания предобуславливателя в главе 3;
- выбран оптимальный итерационный алгоритм для гибридного MPI-многопоточного метода решения в разделе 4.5;
- выбран оптимальный способ балансировки загруженности узлов в разделе 4.3;



(*а*). 1 поток

(*б*). 12 потоков

Рис. 4.11. Графики сравнения скоростей алгоритмов в зависимости от количества узлов

- выбрана оптимальная технология учета неоднородности доступа к памяти на узле в разделе 4.4.

Оптимальность выбора для каждого из этапов подтверждена численными экспериментами с предложенным комплексом программ tNavigator. Теперь необходимо проверить, какой же эффект дает описанная оптимизация для программы решения задачи фильтрации в целом. Ниже будут рассмотрены расчеты в комплексе программ tNavigator на двух наборах данных реальных месторождений:

- «среднего» размера – набор данных типичен для задач, с которыми ежедневно работают инженеры-разработчики, может быть рассчитан на «обычном» настольном компьютере;
- «большого» размера – набор данных очень большого месторождения, требует применения специализированных рабочих станций.

4.6.1. Пример набора данных «среднего» размера

Рассмотрим **3-х фазную** модель со свободным и растворенным газом, характеристики которой приведены в табл. 4.2, на рис. 4.12 для примера показана начальная карта молярной плотности газа.

Параметр	**Значение**
Общий размер процесса в памяти (при 1-м потоке)	5.5Гб
Число блоков входной сетки	3348840
Число блоков расчетной сетки	2418989
Число скважин	39
Максимальное число участков перфорации на скважину	89
Число участков перфорации	853
Длина исторических данных, лет	10

Таблица 4.2. Характеристики набора данных «среднего» размера

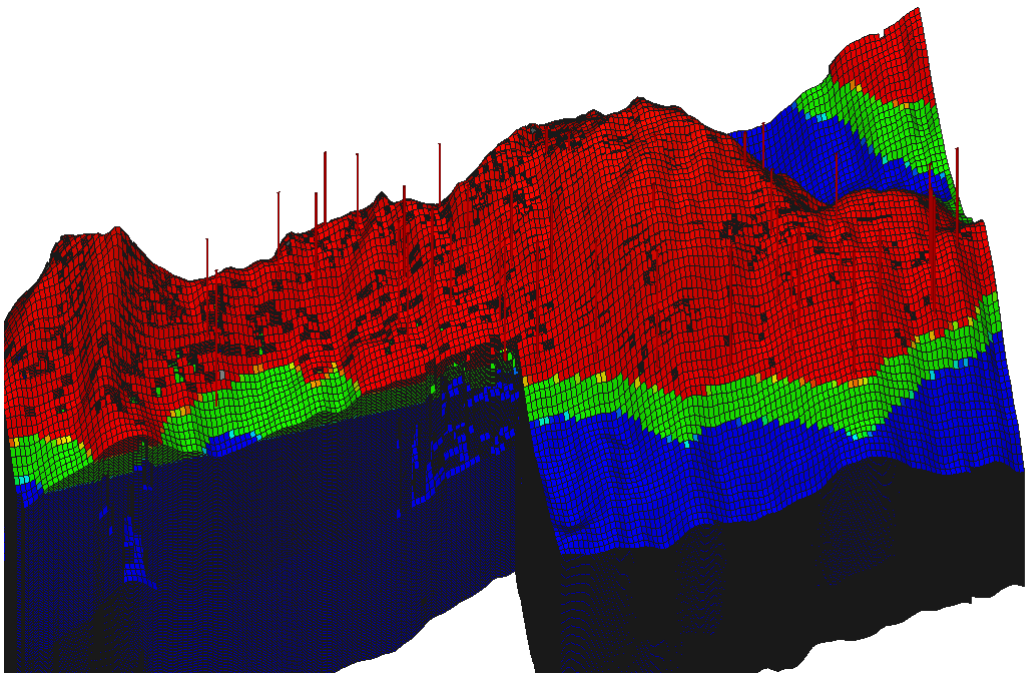


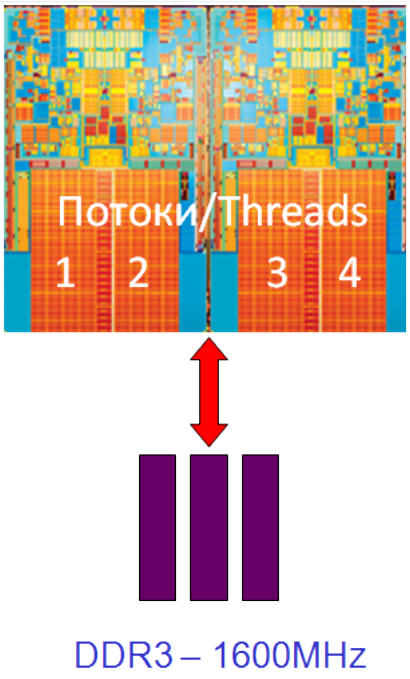
Рис. 4.12. Начальное распределение молярной плотности газа

Тестирование на однопроцессорной системе. Рассмотрим однопроцессорную систему с процессором Intel Core i7-975:

- частота: 3.33ГГц,

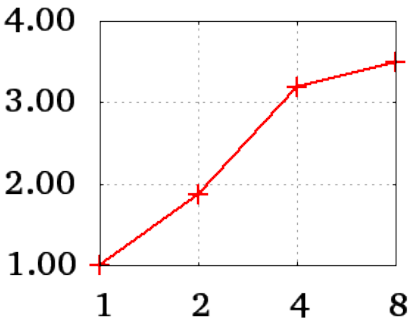
- ядер: 4, логических процессоров: 8,
- частота памяти: 1600МГц, 3 канала памяти.

Это «обычная» настольная система, архитектура которой и полученное на ней ускорение приведены на рис. 4.13. Видим, что время расчета с 12 с половиной часов падает до трех с половиной часов, т.е. в 3.5 раза на 4 физических ядрах. Отметим также, что включение опции многопоточности в процессоре (т.е. увеличение числа логических процессоров до 8-ми) дает сокращение времени расчета почти на 20 минут или 9%, хотя ранее считалось, что на вычислительных задачах опция многопоточности увеличивает время расчета и может использоваться только при обработке потоков информации, см. [125], где указывается достигнутое ускорение в 16% на задаче управления базой данных.



(*а*). Архитектура системы

Число потоков	Время расчета
1	12.31.30
2	06.43.49
4	03.54.03
8	03.34.54



(*б*). Ускорение

Рис. 4.13. Результаты на однопроцессорной системе

Тестирование на двухпроцессорной системе. Рассмотрим двухпроцессорную систему на процессорах Intel Xeon 5580 (см. рис. 4.14):

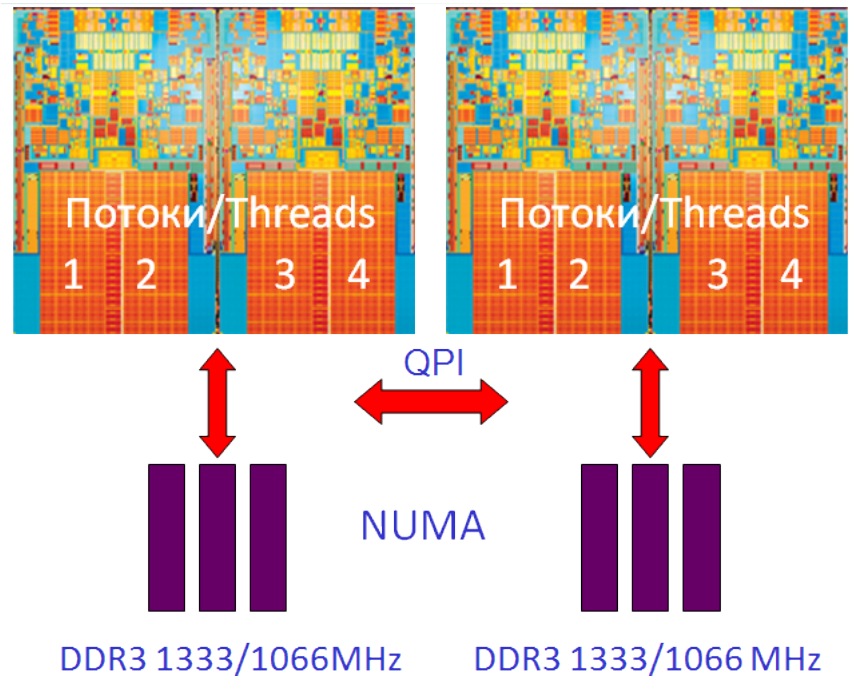


Рис. 4.14. Архитектура двухпроцессорной системы

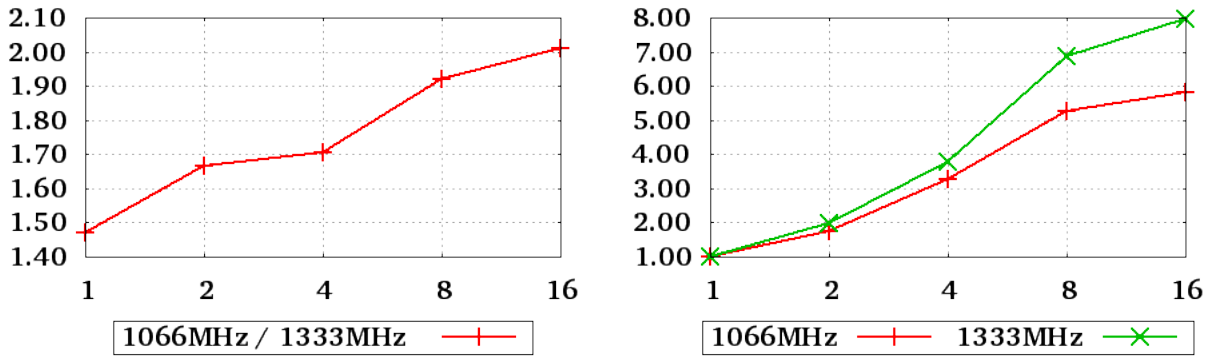
- частота: 3.2ГГц
- процессоров: 2, ядер: 8, логических процессоров: 16
- у каждого процессора свой банк памяти, доступ к памяти другого процессора осуществляется по шине QPI и в 1.5-2 раза медленнее, чем к своей (NUMA),
- 3 канала памяти по 2 модуля на каждый процессор,
- частота памяти: 1333МГц при установке только одного модуля памяти на канал, 1066МГц при установке двух модулей.

Специфическая особенность этой системы, состоящая в том, что «вытаскиванием» каждого второго модуля памяти можно увеличить ее производительность на четверть (с 1066МГц до 1333МГц), позволяет произвести интересный эксперимент по измерению зависимости скорости расчета задачи фильтрации от производительности оперативной памяти. Время расчета на том же

наборе данных для разной частоты работы памяти приведено в табл. 4.3 и на рис. 4.15.

Число потоков	Xeon 5580 / 1066MHz DDR3	Xeon 5580 / 1333MHz DDR3
1	26.01.09	17.41.47
2	15.01.35	09.01.42
4	07.56.41	04.39.46
8	04.56.29	02.34.34
16	04.28.18	02.13.30

Таблица 4.3. Время расчета (часы.минуты.секунды)



(*а*). Относительное время работы (*б*). Ускорение

Рис. 4.15. Сравнение времени расчета при разных частотах памяти

Отметим следующее:

- При использовании одного логического процессора повышение частоты памяти на 25% приводит к ускорению работы на 32%, что свидетельствует о сильной зависимости задачи от производительности оперативной памяти.

- При использовании 16-ти логических процессоров повышение частоты памяти на 25% приводит к ускорению работы в два (!) раза, что говорит о недостаточности полосы пропускания памяти в первой системе (1066МГц) для обеспечения данными всех логических процессоров.

- Вторая система (1333МГц), имеющая 8 физических ядер, достигает теоретически **максимально возможного ускорения** в 8 раз по сравнению с временем работы на 1-м ядре.

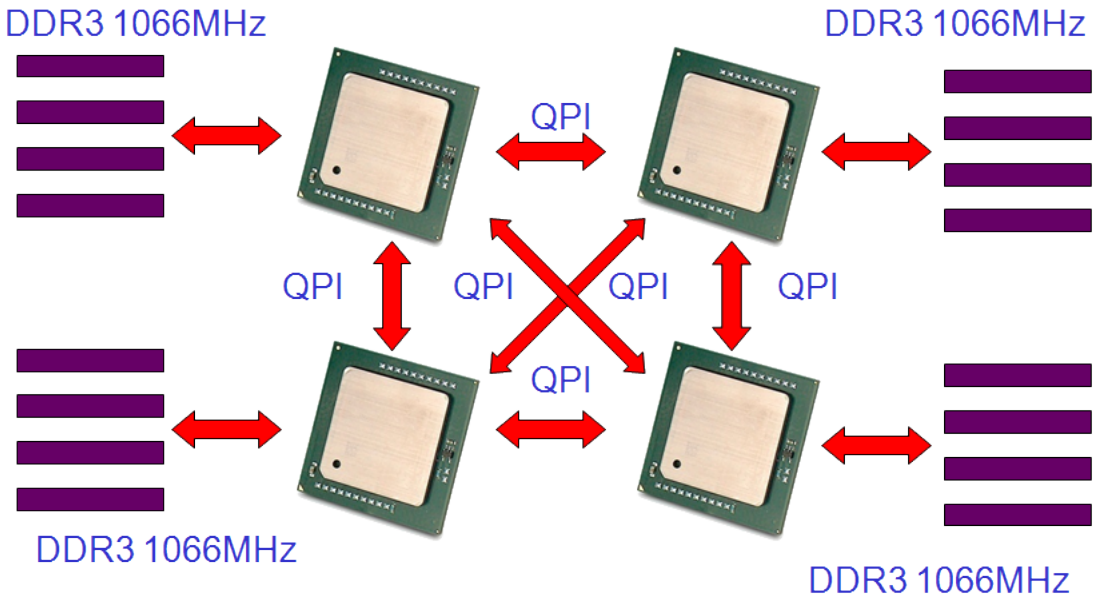


Рис. 4.16. Архитектура четырехпроцессорной системы

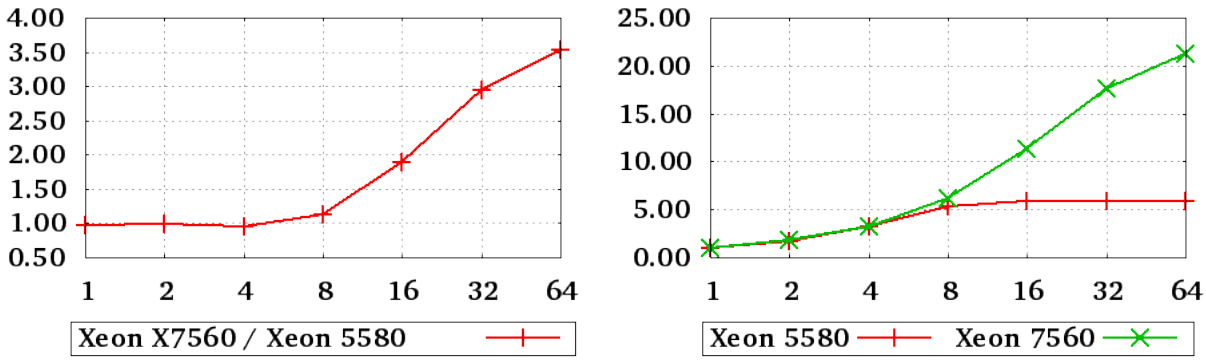
Тестирование на четырехпроцессорной системе. Рассмотрим четырехпроцессорную систему на процессорах Intel Xeon 7560 (см. рис. 4.16):

- частота: 2.27ГГц
- процессоров: 4, ядер: 32, логических процессоров: 64
- у каждого процессора свой банк памяти, доступ к памяти другого процессора осуществляется по шине QPI и в 1.5-2.5 раза медленнее, чем к своей (NUMA),
- 4 канала памяти по 2 модуля на каждый процессор,
- частота памяти: 1066МГц.

Сравнение производительности этой системы с двухпроцессорной с одинаковой частотой памяти приведено в табл. 4.4 и на рис. 4.17.

Число потоков	Xeon 5580 / 1066MHz DDR3	Xeon 7560 / 1066MHz DDR3
1	26.01.09	26.47.32
2	15.01.35	15.12.14
4	07.56.41	08.22.13
8	04.56.29	04.22.47
16	04.28.18	02.22.36
32		01.31.14
64		01.15.48

Таблица 4.4. Время расчета (часы.минуты.секунды)



(*а*). Относительное время работы (*б*). Ускорение

Рис. 4.17. Сравнение времени расчета

Отметим следующее:

- При использовании одного логического процессора повышение частоты процессора на 40% (с 2.27ГГц до 3.2ГГц) приводит к ускорению работы менее 3%, что еще раз подтверждает сильную зависимости задачи от производительности прежде всего оперативной памяти.

- При использовании 16-ти логических процессоров за счет большего количества каналов памяти (16 против 6) четырехпроцессорная система на 77% быстрее двухпроцессорной.

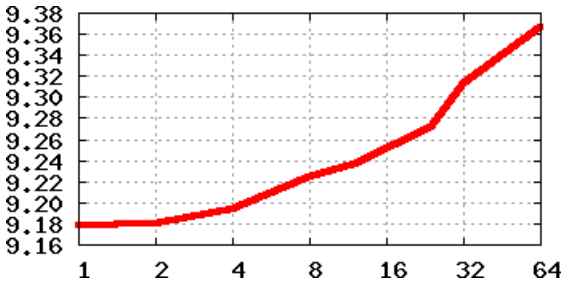
- Четырехпроцессорная система, имеющая 32 физических ядра, достигает ускорения 21 по сравнению с временем работы на 1-м ядре.

Отметим также, что:

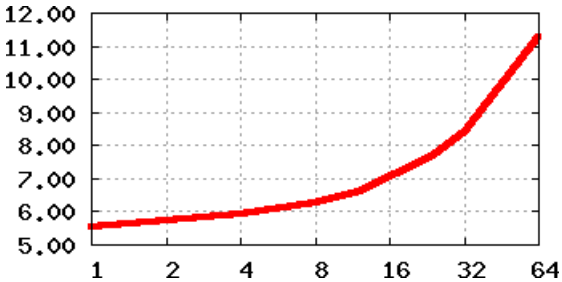
- Полученные результаты невозможны без оптимизации для NUMA архитектуры. При выключении поддержки NUMA расчет замедляется почти в 2 раза, см. рис. 4.8, *а* на стр. 154.

- Также хорошо себя показывает предобуславливатель, см. график числа итераций как функции количества логических процессоров на рис. 4.18, *а*, из которого следует, что при увеличении числа потоков с 1 до 64 качество предобуславливателя ухудшилось на 2%.

- Выбранный способ хранения матрицы и предобуславливателя весьма экономен с точки зрения потребления оперативной памяти, см. график использованной оперативной памяти как функции количества логических процессоров на рис. 4.18, *б*, из которого следует, что при увеличении числа потоков с 1 до 64 объем требуемой оперативной памяти возрос в 2 раза.



(*а*). Количество линейных итераций (за всю задачу), тысяч



(*б*). Использованный объем оперативной памяти, Гб

Рис. 4.18. Характеристики расчета как функция числа логических процессоров

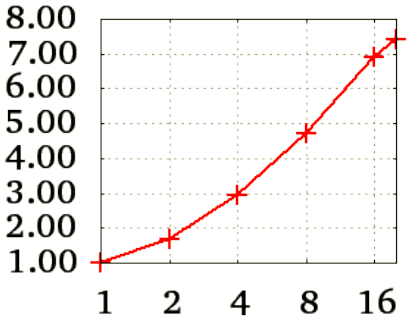
Тестирование на кластерной системе. Рассмотрим кластерную систему на процессорах Intel Xeon 5650:

- частота процессоров: 2.66ГГц,
- 20 узлов,
- процессоров: 40, ядер: 240, логических процессоров: 240,
- на каждом узле 2 физических процессора, у каждого процессора свой банк памяти, доступ к памяти другого процессора осуществляется по шине QPI и в 1.5-2 раза медленнее, чем к своей (NUMA),
- 3 канала памяти по 2 модуля на каждый процессор,
- частота памяти: 1333МГц, суммарный объем 480Гб,
- взаимодействие узлов: QDR 4x Infiniband (40Гб/с соединения).

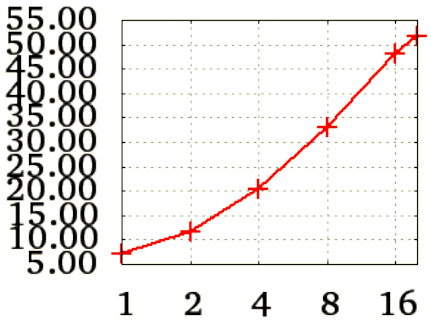
Использован гибридный метод распараллеливания: MPI между узлами, потоки исполнения (с поддержкой NUMA) на узлах.

Узлов	Время
1	02.55.04
2	01.34.11
4	00.53.12
8	00.33.29
16	00.22.41
20	00.21.03

(*а*). Время расчета



(*б*). Ускорение по отношению к времени работы 1 узла



(*в*). Ускорение по отношению к времени работы 1 ядра

Рис. 4.19. Характеристики расчета на кластере

Характеристики расчета приведены на рис. 4.19. Отметим полученное ускорение в 51 раз по сравнению с временем работы последовательной про-

граммы, что разительно отличается от результатов аналогичных программ расчета, см. рис. 1.12 на стр. 74.

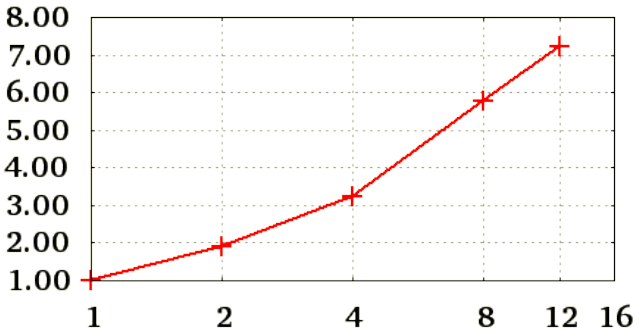
4.6.2. Пример набора данных «большого» размера

Рассмотрим расчет сложного набора данных «большого» размера: **Самотлорское месторождение**, пласты AV1-2, AV1-3, AV2-3, AV4-5. В модели **4.7млн. активных блоков**, почти **13 тысяч скважин**, **40 лет истории** разработки. Модель 3-х фазная, включает подгазовую область месторождения. Краевые условия задаются водоносными горизонтами. Для загрузке на одном компьютере модель требует около **15Гб оперативной памяти**.

Для тестирования будем использовать кластерную систему, описанную в предыдущем разделе. Время расчета на одном узле кластера приведено на рис. 4.20.

Число потоков	Время расчета
1	57.05.46
2	30.03.02
4	17.46.21
8	09.54.11
12	07.54.45

(*а*). Время расчета (часы.минуты.секунды)



(*б*). Ускорение по отношению к времени работы 1 потока

Рис. 4.20. Характеристики расчета на кластере «большого» набора данных

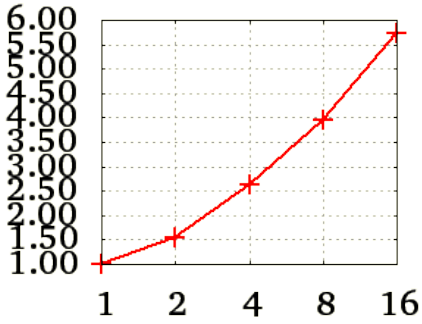
Ускорение на одном узле составляет **7.2 раза** на 12-ти ядрах. Заметим, что и на «среднем» наборе данных мы тоже получали значения порядка 7.5. Это связано с тем, что по сравнению с процессорами Xeon 5580, рассмотренными при тестировании двухпроцессорных систем, процессоры Xeon 5650 добавляют 2 дополнительных ядра без увеличения числа каналов памяти или

ее пропускной способности. В силу отмеченной выше чувствительности задачи фильтрации к производительности оперативной памяти, это объясняет, почему ускорение на системе из 2-х 6-ти ядерных процессоров Xeon 5650 не превышает ускорения на системе из 2-х 4-х ядерных процессоров Xeon 5580.

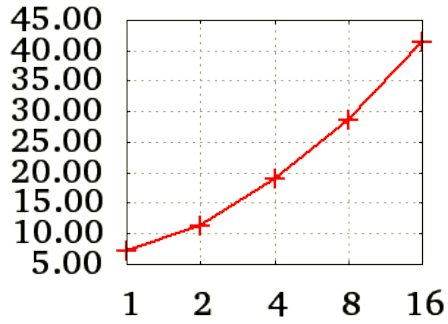
Поведение задачи при расчете на многих узлах гибридным алгоритмом показано на рис. 4.21.

Узлов	Время
1	07.54.45
2	05.07.03
4	03.00.23
8	01.59.55
16	01.22.39

(*а*). Время расчета



(*б*). Ускорение по отношению к времени работы 1 узла



(*в*). Ускорение по отношению к времени работы 1 ядра

Рис. 4.21. Характеристики расчета на кластере «большого» набора данных

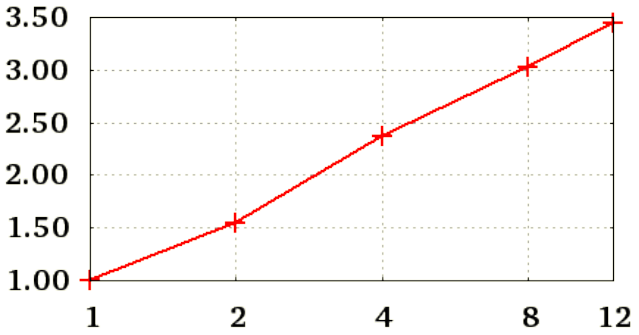
Отметим полученное ускорение в 41 раз по сравнению с временем работы последовательной программы, что разительно отличается от результатов аналогичных программ расчета, см. рис. 1.12 на стр. 74.

В силу гибридного подхода к распараллеливанию задачи каждый логический процессор на узле дает вклад в общее ускорение работы. Для демонстрации этого рассчитаем задачу на всех 16-ти узлах, но будем использовать на каждом из них по одному, 2 и т.д. логических процессоров. Результаты представлены на рис. 4.22.

Видим, что использование многоядерных процессоров на узлах вычислительного кластера оправдано и повышает эффективность расчета.

Число потоков на узле	Время расчета на 16-ти узлах
1	04.45.07
2	03.03.48
4	02.00.06
8	01.34.16
12	01.22.39

(*а*). Время расчета на 16-ти узлах (часы.минуты.секунды)



(*б*). Ускорение по отношению к времени работы 1 потока на 16-ти узлах

Рис. 4.22. Зависимость времени расчета «большого» набора данных от числа потоков на узле

4.7. Сравнение результатов с другими пакетами решения задачи фильтрации

После рассмотрения эффективности решения задачи фильтрации (4) в разделе 4.6 естественно возникает вопрос, не были ли они достигнуты за счет качества получаемого решения.

Так как решается сложная задача на сложно устроенных параллельных ЭВМ, то объем программы tNavigator превышает 60 мегабайт исходного текста на языке C++, и это тоже остро ставит вопрос о проверке корректности ее работы.

Ниже мы рассмотрим несколько примеров сравнения результатов расчета с другими пакетами решения задачи фильтрации.

4.7.1. Пример набора данных SPE 10

Набор данных данных SPE (Society of Petroleum Engineers) 10 был предложен изначально в [126], как тест для так называемого ремасштабирования (upscaling) – процедуры укрупнения сетки для того, чтобы сократить время расчета (и часто сделать его возможным из-за ограничений по объему оперативной памяти). Развитие вычислительной техники сделало возможным вести

расчет на исходной сетке. Это вместе со свободной доступностью данных, которые можно взять на сайте SPE, сделало данный тест достаточно популярным для сравнения результатов и скорости работы различных программ.

Основные характеристики тестового примера SPE 10:

- «кусок» дна Норвежского моря на геологической сетке (реальная модель);
- число блоков расчетной сетки: 1094421;
- очень высокая неоднородность коэффициентов тензора абсолютной проницаемости (значения в соседних ячейках сетки могут отличаться на 4 порядка).

В качестве тестовой использована двухпроцессорная система:

- 2 физических процессора Intel Xeon 5580 частотой 3.2ГГц;
- 8 физических ядер, 8 логических процессоров;
- 12Гб DDR3 памяти частотой 1333МГц;

похожая на описанную в разделе 4.6. В качестве программы для сравнения использовался пакет Parallel Eclipse 2008.2 компании Schlumberger. Хочется поблагодарить специалистов компании «ТНК-ВР» за проведенные численные эксперименты с указанной программой. Обе программы запускались на указанной тестовой системе, используя 8 логических процессоров.

Параметр	**Рассматриваемый пакет 8CPU**	**Eclipse 2008.2 8CPU**
Пиковое потребление памяти	2600Мб	3290Мб
Рабочее потребление памяти	2051Мб	2654Мб
Время работы	24 часа 51 минута	24 суток 13 часов

Таблица 4.5. Сравнение пакетов на тесте SPE 10

Результаты тестирования приведены в табл. 4.5 и на рис. 4.23, 4.24, 4.25, где знаком + отмечены данные для Eclipse, знаком - отмечены данные для тестируемой программы tNavigator. Видим, что

- Время работы программ отличается катастрофически (1 сутки и 3 с половиной недели); проблемы с масштабируемостью и плохой работой при сильной неоднородности входных данных у предобуславливателя nested factorisation, использованном в пакете Eclipse, уже отмечались в литературе ([127]).
- Различия в основных показателях не превышают 5%.

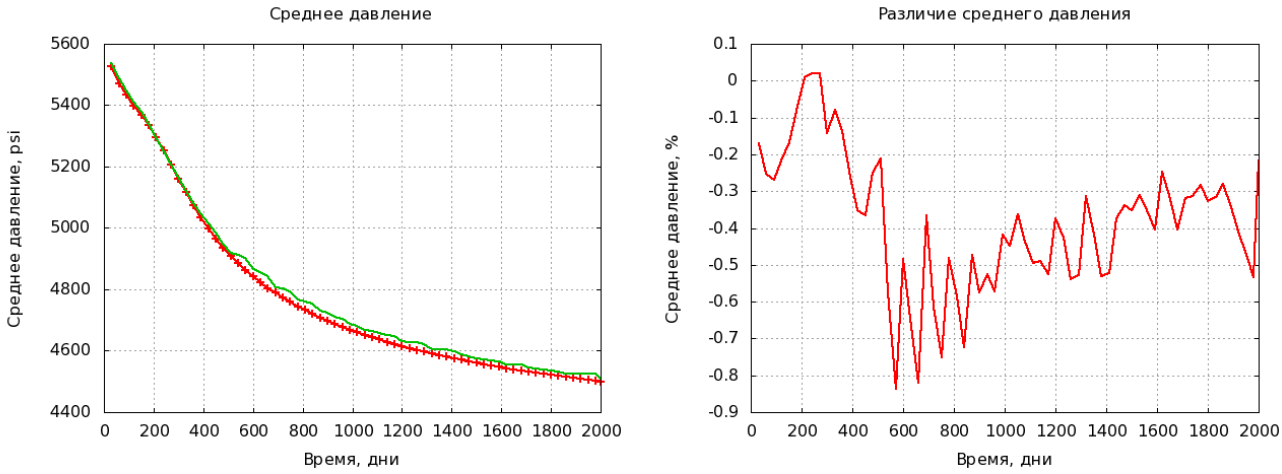


Рис. 4.23. Сравнение среднего давления по пласту на тесте SPE 10

4.7.2. Пример набора данных реального месторождения 1

Ниже представлен результат большой работы по сравнению пакета Eclipse и комплекса программt Navigator, проведенной специалистами компании «Газпромнефть-НТЦ». Хочется выразить благодарность всем участникам тестирования и руководству компании «Газпромнефть-НТЦ», разрешившему публикацию результатов (без указания имен месторождений и их запасов).

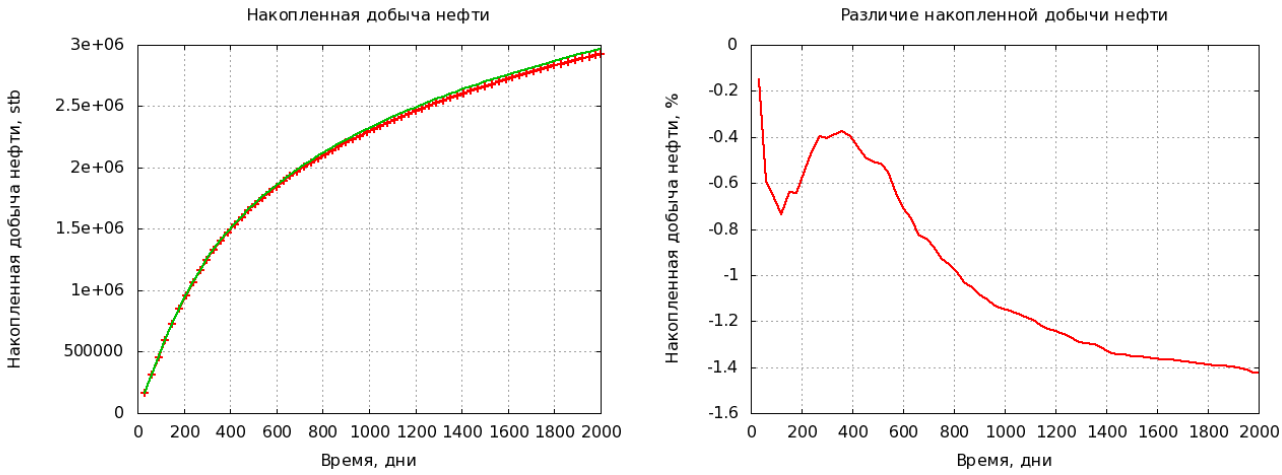


Рис. 4.24. Сравнение добычи нефти на тесте SPE 10

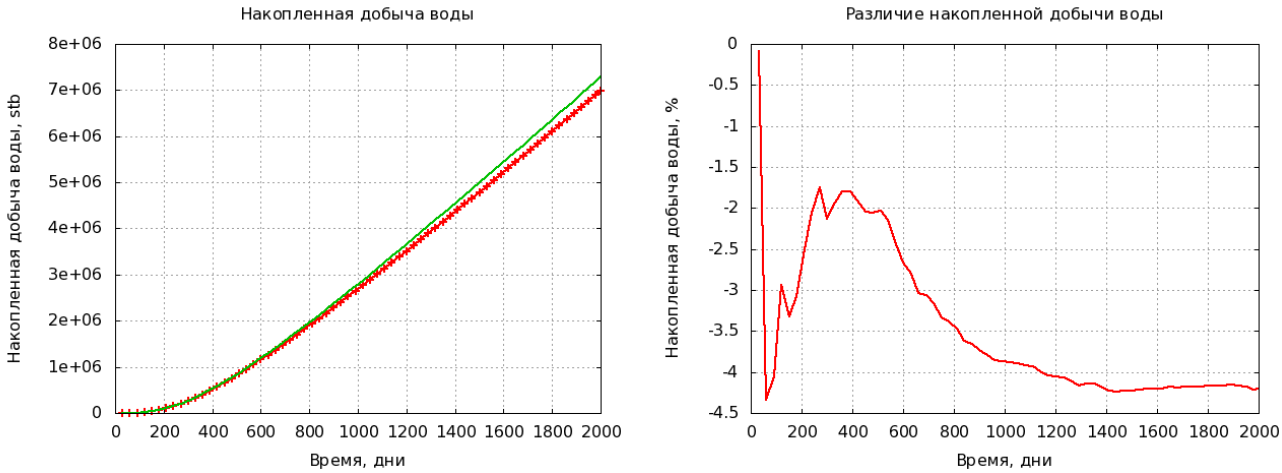


Рис. 4.25. Сравнение добычи воды на тесте SPE 10

Основные характеристики тестового примера 1 и время расчета на компьютере HP Z800 (2 процессора Xeon 5580) приведены в табл. 4.6, характеристики гистограммы распределения карты разности нефтенасыщенности и давления приведены в табл. 4.7.

На рис. 4.26 слева приведены графики среднего давления по месторождению. На том же рисунке справа приведен так называемый кросс-плот по всем скважинам месторождения на последнюю дату для давления на забое скважин (каждая точка на рисунке соответствует скважине, по горизонтальной оси в качестве координаты точки отложено вычисленное значение в пакете tNavigator, по вертикальной оси – в пакете Eclipse; таким образом, отсутствие

Параметр	Значение
Число блоков входной сетки	1755000
Число блоков расчетной сетки	1039871
Число скважин	1558
Длина исторических данных, лет	28
Время расчета в пакете Eclipse (часы:минуты)	11:00
Время расчета в пакете tNavigator (часы:минуты)	3:00

Таблица 4.6. Характеристики набора данных 1

Диапазон на гистограмме	Нефтенасыщенность	Давление
[<-20%]	0.0%	0.0%
[-20%;-5%]	0.0%	0.0%
[-5%;5%]	100.0%	100.0%
[5%;20%]	0.0%	0.0%
[>20%]	0.0%	0.0%

Таблица 4.7. Гистограммы основных карт на последнюю дату расчета

отклонения означает расположение точки на биссектрисе; также изображен угол с раскрытием 5% от биссектрисы в большую и меньшую стороны). Аналогичные кросс-плоты приведены

- для дебита нефти каждой скважины на последнюю дату и накопленную добычу нефти каждой скважины — на рис. 4.27;
- для дебита жидкости каждой скважины на последнюю дату и накопленную добычу жидкости скважины — на рис. 4.28;
- для приемистости воды каждой скважины на последнюю дату и накопленную закачку каждой скважины — на рис. 4.29.

Этот пример демонстрирует практически полное совпадение результатов, измеренное в $1558 * 7 = 10908$ точках (количество скважин на 7-ми кросс-плотах), при более, чем троекратном, сокращении времени расчета.

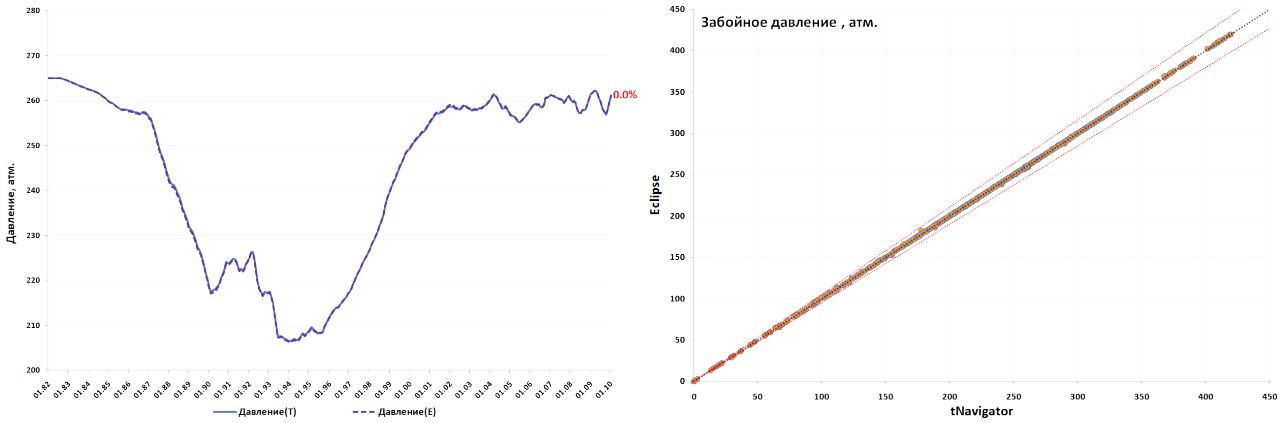


Рис. 4.26. Сравнение среднего давления по пласту и давления на забое скважин на тесте 1

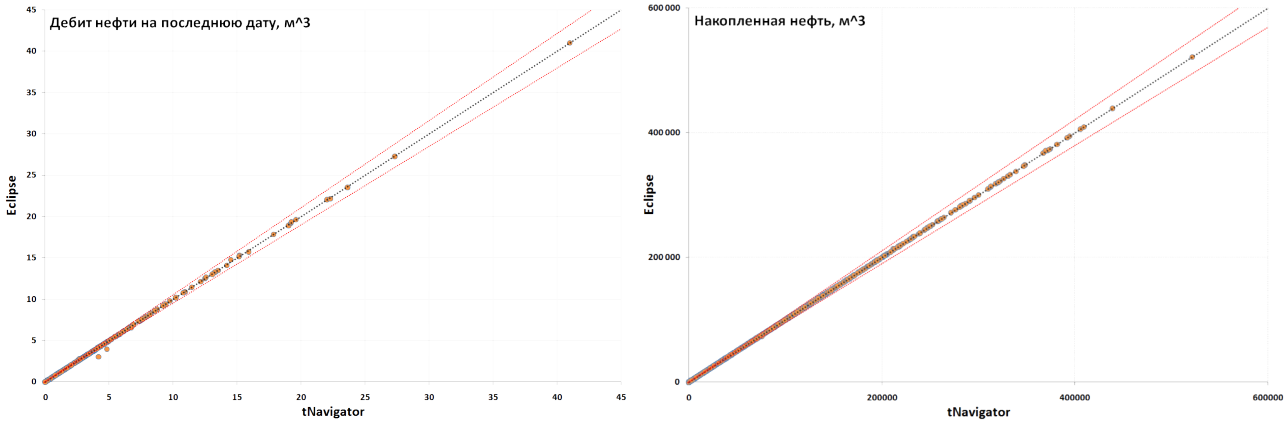


Рис. 4.27. Сравнение дебита нефти и суммарной нефтедобычи на тесте 1

4.7.3. Пример набора данных реального месторождения 2

Ниже представлен другой пример из упоминавшегося выше исследования, проведенного специалистами компании «Газпромнефть-НТЦ».

Основные характеристики тестового примера 2 и время расчета на компьютере HP Z800 (2 процессора Xeon 5580) приведены в табл. 4.8, характеристики гистограммы распределения карты разности нефтенасыщенности и давления приведены в табл. 4.9. Набор данных характеризуется большой сложностью геологического строения, что приводит к сложной неструктурированной сетке с разломами и высокой неоднородностью свойств, поэтому, несмотря на меньшие, по сравнению с предыдущим набором, размеры сетки и меньшее число скважин, расчет традиционными пакетами идет очень медленно.

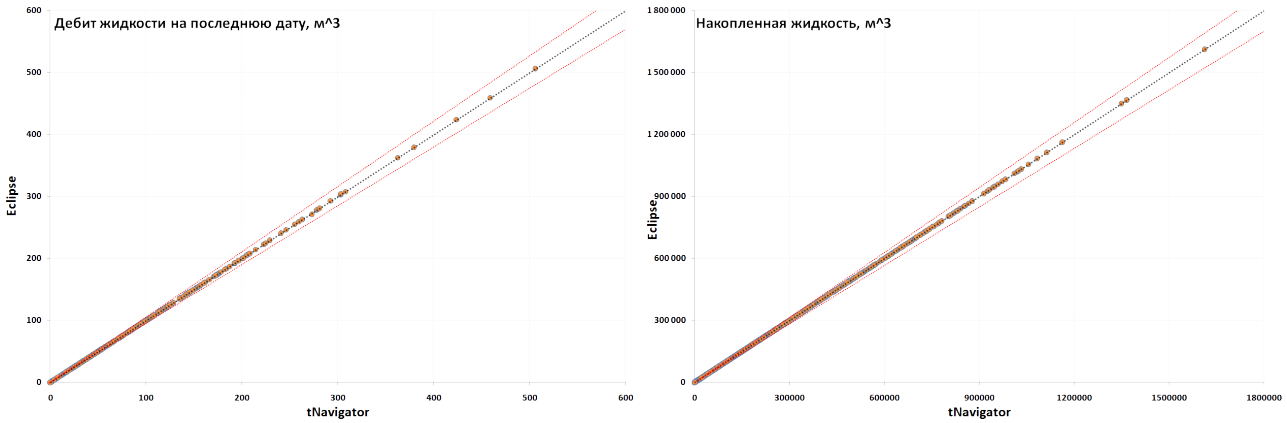


Рис. 4.28. Сравнение дебита жидкости и суммарной добычи жидкости на тесте 1

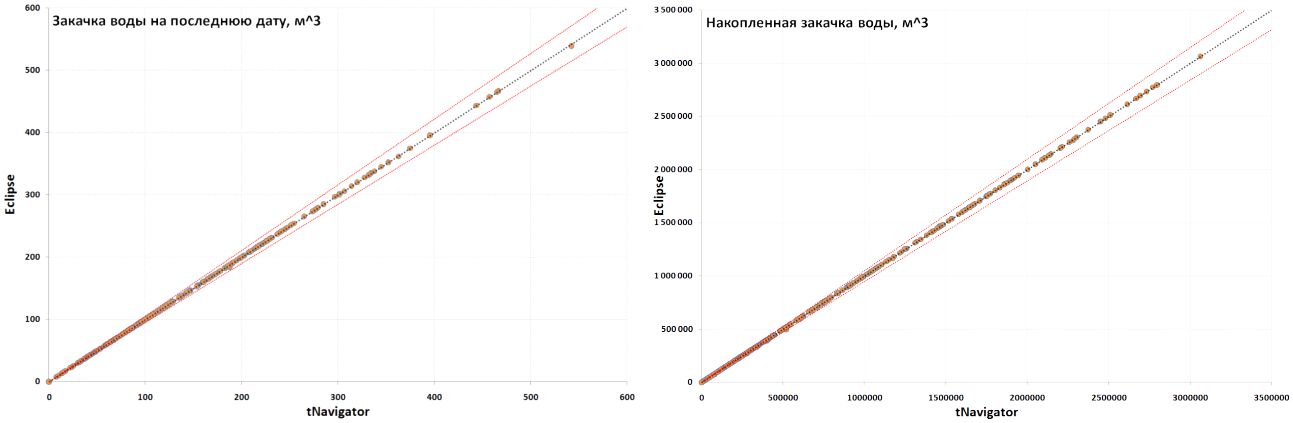


Рис. 4.29. Сравнение приемистости воды и суммарной закачки воды на тесте 1

На рис. 4.26 слева приведены графики среднего давления по месторождению (сплошная линия соответствует Eclipse, пунктирная – tNavigator). На том же рисунке справа приведен кросс-плот по всем скважинам месторождения на последнюю дату для давления на забое скважин. Аналогичные кросс-плоты приведены

- для дебита нефти каждой скважины на последнюю дату и накопленную добычу нефти каждой скважины — на рис. 4.31;
- для дебита жидкости каждой скважины на последнюю дату и накопленную добычу жидкости скважины — на рис. 4.32;
- для приемистости воды каждой скважины на последнюю дату и накопленную закачку каждой скважины — на рис. 4.33.

Параметр	Значение
Число блоков входной сетки	11570000
Число блоков расчетной сетки	924985
Число скважин	726
Длина исторических данных, лет	25
Время расчета в пакете Eclipse (часы:минуты)	26:30
Время расчета в пакете tNavigator (часы:минуты)	2:37

Таблица 4.8. Характеристики набора данных 2

Диапазон на гистограмме	Нефтенасыщенность	Давление
[<-20%]	0.1%	0.0%
[-20%;-5%]	1.7%	0.0%
[-5%;5%]	97.4%	98.7%
[5%;20%]	0.8%	1.3%
[>20%]	0.0%	0.0%

Таблица 4.9. Гистограммы основных карт на последнюю дату расчета

Этот пример демонстрирует хорошее совпадение результатов, измеренное в $726 * 7 = 5082$ точках (количество скважин на 7-ми кросс-плотах), при сокращении времени расчета в 10 раз.

4.8. Выводы к четвертой главе

В четвертой главе описана предложенная технология, включающая алгоритмы, структуры данных и комплекс программ, для построения гибридного MPI–многопоточного вычислительного процесса для решения задачи фильтрации на ЭВМ с распределенной памятью (кластере). Технология включает в себя:

- технологию построения гибридного трехуровневого вычислительного процесса решения задачи фильтрации;
- технологию балансировки загруженности узлов вычислительного кластера при расчете задачи фильтрации;

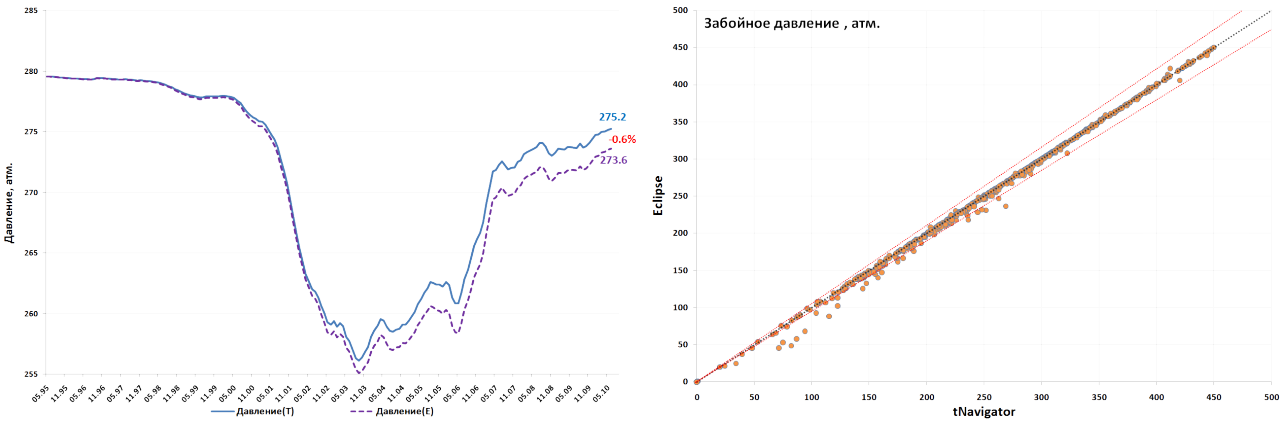


Рис. 4.30. Сравнение среднего давления по пласту и давления на забое скважин на тесте 2

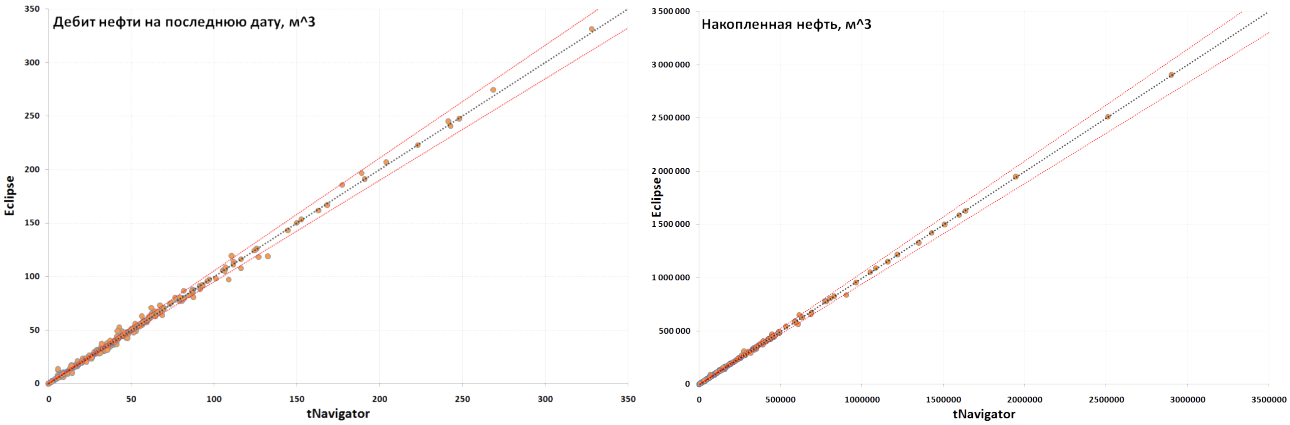


Рис. 4.31. Сравнение дебита нефти и суммарной нефтедобычи на тесте 2

- технологию оптимизации решения задачи фильтрации для многопроцессорных систем с общей неоднородной памятью.

Технология реализована в программном комплексе tNavigator расчета задачи фильтрации.

Для достижения наилучшей эффективности в программном комплексе были реализованы 4 алгоритма решения систем линейных уравнений с разреженной несимметричной матрицей и на большом наборе матриц и разнообразной конфигурации кластерной системы был определен лучший алгоритм с точки зрения времени работы и масштабируемости.

Для тестирования программного комплекса tNavigator проведена серия экспериментов:

- на разнообразных параллельных ЭВМ, от 8-ми до 240 логических про-

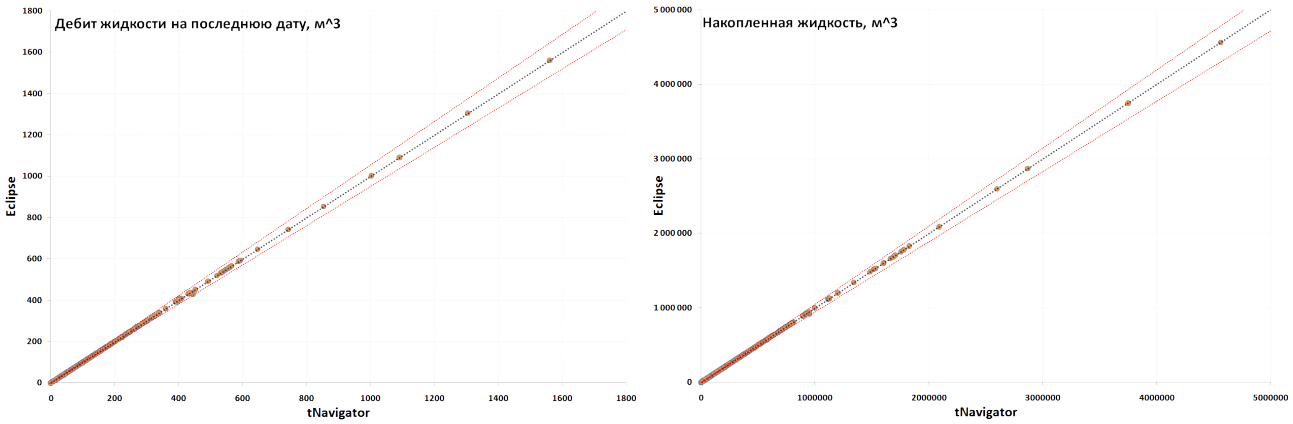


Рис. 4.32. Сравнение дебита жидкости и суммарной добычи жидкости на тесте 2

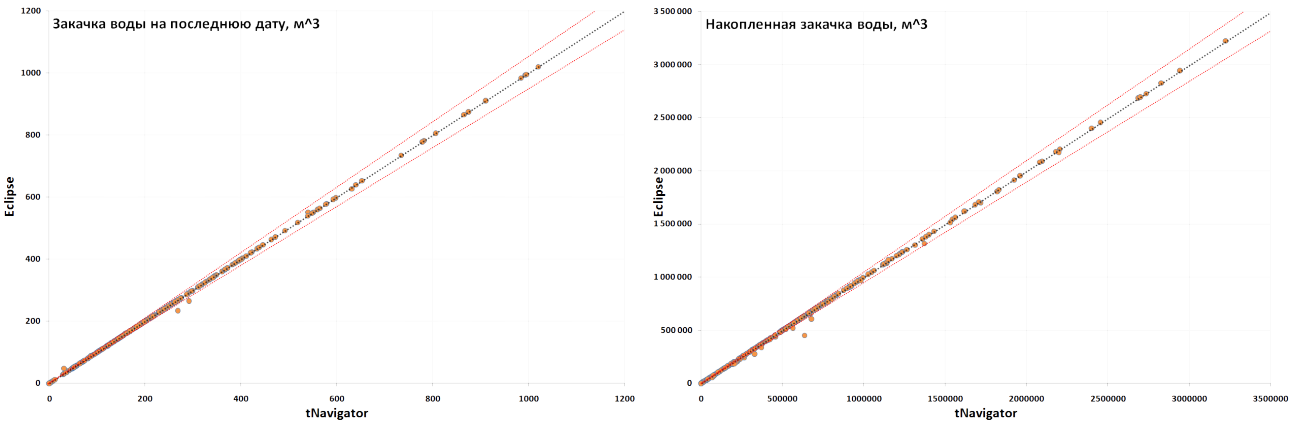


Рис. 4.33. Сравнение приемистости воды и суммарной закачки воды на тесте 2

цессоров, с разными частотами оперативной памяти, — для определения характеристик масштабируемости;

- на разнообразных наборах данных для реальных месторождений — для определения согласованности расчета с другими программными комплексами решения задачи фильтрации.

Численные эксперименты показывают хорошую эффективность программного комплекса tNavigator, реализующего описанную технологию, при хорошем согласовании с результатами работы других программ.

Заключение

В настоящей работе предложен метод решения задачи фильтрации вязкой сжимаемой многофазной многокомпонентной смеси на современных параллельных ЭВМ, учитывающий их основные особенности: многоядерность вычислительных узлов, неоднородность доступа процессорных ядер к оперативной памяти, большой объем кеш-памяти у логических процессоров.

Отметим основные результаты работы:

- математическая модель техногенной трещиноватости вблизи скважин;
- метод блочного хранения разреженных несимметричных матриц в памяти и блочные варианты построения предобуславливателей, основанных на неполном LU разложении;
- алгоритм распараллеливания построения предобуславливателей, основанных на неполном LU разложении;
- эффективная технология для построения гибридного MPI–многопоточного вычислительного процесса для решения задачи фильтрации на ЭВМ с распределенной памятью, включающая в себя:
  - технологию построения гибридного трехуровневого вычислительного процесса решения задачи фильтрации;
  - технологию балансировки загруженности узлов вычислительного кластера при расчете задачи фильтрации;
  - технологию оптимизации решения задачи фильтрации для многопроцессорных систем с общей неоднородной памятью.
- Комплекс программ tNavigator для решения задачи фильтрации на современных параллельных ЭВМ, реализующий описанные в работе методы

и алгоритмы. Программа tNavigator свободно доступна для загрузки с сайта rfdyn.ru.

Для достижения наилучшей эффективности в программном комплексе tNavigator были реализованы и протестированы на большом количестве наборов входных данных и разнообразных конфигураций кластерных систем

- 3 алгоритма построения предобуславливателей класса ILU (ILU(0), ILU(1), ILUT);
- 2 способа разделения матрицы при распараллеливании предобуславливателей класса ILU (ILUG и ILUA);
- 4 алгоритма решения систем линейных уравнений с разреженной несимметричной матрицей (ORTHOMIN, QGMRES, DQGMRES, BCGS).

По результатам тестирования был выбран наилучший для задачи фильтрации алгоритм.

Для тестирования программного комплекса tNavigator в целом проведена серия экспериментов:

- на разнообразных параллельных ЭВМ, от 8-ми до 240 логических процессоров, с разными частотами оперативной памяти, — для определения характеристик масштабируемости;
- на разнообразных наборах данных для реальных месторождений — для определения согласованности расчета с другими программными комплексами решения задачи фильтрации.

Численные эксперименты показали хорошую эффективность программного комплекса tNavigator, реализующего описанный в работе метод решения задачи фильтрации вязкой сжимаемой многофазной многокомпонентной смеси,

при хорошем согласовании с результатами работы других программ решения той же задачи.

Литература

1. К. Ю. Богачев. Практикум на ЭВМ. Методы решения линейных систем и нахождения собственных значений. 2 изд. Москва: Механико-математический ф-т МГУ им. М.В.Ломоносова, 1999. 200 с.

2. К. Ю. Богачев. Практикум на ЭВМ. Методы приближения функций. 3 изд. Москва: Механико-математический ф-т МГУ им. М.В.Ломоносова, 2002. 200 с.

3. К. Ю. Богачев. Основы параллельных вычислений. Москва: ЦПИ при механико-математическом ф-те МГУ им. М.В.Ломоносова, 2002. 352 с.

4. К. Ю. Богачев. Основы параллельного программирования. Москва: Бином, 2003. 342 с. ISBN: 5-94774-037-0.

5. К. Ю. Богачев, Н. С. Мельниченко. О пространственной аппроксимации методом подсеток для задачи фильтрации вязкой сжимаемой жидкости в пористой среде // Вычислительные методы и программирование. 2008. Т. 9. С. 191–199.

6. К. Ю. Богачев, И. Г. Горелов. Применение параллельного предобуславливателя CPR к задаче фильтации вязкой сжимаемой жидкости в пористой среде // Вычислительные методы и программирование. 2008. Т. 9. С. 184–190.

7. К. Ю. Богачев, Я. В. Жабицкий. Блочные предобуславливатели класса ILU для задач фильтрации многокомпонентной смеси в пористой среде // Вестник Моск. ун-та, Сер. 1, Математика, Механика. 2009. Т. 5. С. 19–25.

8. К. Ю. Богачев, Я. В. Жабицкий. Метод Капорина-Коньшина параллельной реализации блочных предобусловливатей для несимметричных мат-

риц в задачах фильтрации многокомпонентной смеси в пористой среде // Вестник Моск. ун-та, Сер. 1, Математика, Механика. 2010. Т. 1. С. 46–52.

9. К. Ю. Богачев, А. Р. Миргасимов. Об оптимизации вычислительных приложений для многопроцессорных систем с общей неоднородной памятью // Вычислительные методы и программирование. 2010. Т. 11. С. 198–209.

10. К. Ю. Богачев, М. Ю. Михалева, И. Г. Горелов. Алгебраический многоуровневый метод AMG: сравнение с методом BICGSTAB+ILU и использование в составе метода CPR // Вестник Моск. ун-та, Сер. 1, Математика, Механика. 2010. Т. 4. С. 24–28.

11. К. Ю. Богачев, А. А. Климовский, А. Р. Миргасимов, А. Е. Семенко. Балансировка загруженности узлов кластера при расчете задачи фильтрации // Вычислительные методы и программирование. 2011. Т. 12. С. 70–73.

12. К. Ю. Богачев, Я. В. Жабицкий, А. А. Климовский и др. Сравнение итерационных методов решения разреженных систем линейных уравнений в задачах фильтрации на вычислительных системах с распределенной памятью // Вычислительные методы и программирование. 2011. Т. 12. С. 74–76.

13. Bogachev K. Y., Shelkov V. G. New Realistic Hydraulic and Technogenic Fracture Modeling Approach in Full-Scale Dynamic Models // SPE paper 138071. Presented at the Russian Oil and Gas Conference and Exhibition, Moscow, Russia. 2010. 6 pp.

14. Bogachev K. Y., Eydunov D. A., Robinson T. et al. A New Approach to Numerical Simulation of Fluid Flow in Fractured Shale Gas Reservoirs //

SPE paper 147021. Presented at the Canadian Unconventional Resources Conference, Alberta, Canada. 2011. 6 pp.

15. К. Ю. Богачев. Итерационные методы решения квазилинейных эллиптических задач в областях сложной формы // ДАН. 1992. Т. 322, № 4. С. 641–645.

16. Bogachev K. Y. Iterative methods of solving main boundary value problems for second-order quasilinear elliptic equations in complexly shaped domains // Rus. J. Numer. Analysis Math. Model. 1992. Vol. 7, no. 4. Pp. 281–298.

17. К. Ю. Богачев. Итерационный метод решения смешанных задач для квазилинейных эллиптических уравнений в областях сложной формы // ДАН. 1995. Т. 340, № 6. С. 727–730.

18. Bogachev K. Y. Iterative method of solving quasilinear elliptic equations with rapidly varying coefficients singulary depending on two parameters // Rus. J. Numer. Analysis Math. Model. 1995. Vol. 10, no. 1. Pp. 9–32.

19. К. Ю. Богачев. Обоснование метода фиктивных областей решения смешанных краевых задач для квазилинейных эллиптических уравнений // Вестник Московского Университета, Серия 1, Математика, Механика. 1996. № 3. С. 16–23.

20. Н. С. Бахвалов, К. Ю. Богачев, М. Э. Эглит. Вычисление эффективных упругих модулей для несжимаемого пористого материала // Механика композиционных материалов. 1996. Т. 32, № 5. С. 579–587.

21. Н. С. Бахвалов, К. Ю. Богачев, Maitre J. F. Эффективный алгоритм решения жестких эллиптических задач с приложениями к методу фиктивных областей // Ж. вычисл. матем. и матем. физ. 1999. Т. 39, № 6. С. 919–931.

22. Bogachev K. Y. On Algorithm for Efficient Solving Stiff Elliptic Problems with Large Parameters // Rus. J. Numer. Analysis Math. Model. 1999. Vol. 14, no. 6. Pp. 479–493.

23. Bogachev K. Y. On Algorithm for Efficient Solving Multiparametrical Stiff Elliptic Problems // Proceedings of the ENUMATH-99, Jyvaskula, Finland. 1999. Pp. 52–53.

24. К. Ю. Богачев. Эффективные алгоритмы решения эллиптических задач с большими параметрами // Труды Математического центра имени Н.И. Лобачевского. 1996. Т. 2. С. 3–44.

25. К. Ю. Богачев. Эффективный алгоритм решения эллиптических задач с большими параметрами // Ж. вычисл. матем. и матем. физ. 2000. Т. 40, № 3. С. 402–415.

26. Bogachev K. Y. On new method of parabolic approximation of the nonstationary Navier-Stokes equations // Proceedings of the ENUMATH-99, Luglio, Italy. 2001. Pp. 201–202.

27. Bogachev K. Y. Efficient Algorithms for Stiff Elliptic Problems with Large Parameters // Rus. J. Numer. Analysis Math. Model. 2002. Vol. 17, no. 4. Pp. 347–366.

28. Н. С. Бахвалов, К. Ю. Богачев, М. Э. Эглит. Исследование эффективных уравнений с дисперсией, описывающих распространение волн в неоднородных тонких стержнях // ДАН. 2002. Т. 387, № 6. С. 749–753.

29. Bogachev K. Y., Kobelkov G. On new method of parabolic approximation of the nonstationary Navier-Stokes equations // Proceedings of the CFD 2003 Conference, Moscow, Russia. Elsevier, 2004. Pp. 163–172.

30. К. Ю. Богачев. Операционные системы реального времени. Москва: ЦПИ при механико-математическом ф-те МГУ им. М.В.Ломоносова, 2001. 200 с.

31. Peaceman D. W. Fundamentals of Numerical Reservoir Simulation. Elsevier Science Ltd, 1977. 176 pp. ISBN: 0-44441-578-5.

32. Aziz K., Settari A. Petroleum Reservoir Simulation. London: Applied Science Publishers, 1979. 497 pp.

33. McCain W. D. The Properties of Petroleum Fluids. PennWell Books, 1990. 596 pp. ISBN: 0-87814-335-1.

34. Craft B., Hawkins M. F. Applied Petroleum Reservoir Engineering. 2 edition. Prentice Hall PTR Englewood Cliffs, 1991. 464 pp. ISBN: 0-13039-884-5.

35. Economides M. J., Hill A. D., Ehlig-Economides C. Petroleum Production Systems. Prentice Hall Ptr, 1993. 624 pp. ISBN: 0-13-658683-X.

36. Firoozabadi A. Thermodynamics of Hydrocarbon Reservoirs. McGraw-Hill Professional, 1999. 353 pp. ISBN: 0-07022-071-9.

37. Ertekin T., Abou-Kassem J. H., King G. R. Basic Applied Reservoir Simulation. Society of Petroleum Engineers, 2001. 421 pp. ISBN: 1-55563-089-8.

38. Walsh M. P., Lake L. W. A Generalized Approach to Primary Hydrocarbon Recovery (Handbook of Petroleum Exploration and Production). 4 edition. Elsevier Science Ltd, 2003. 640 pp. ISBN: 0-44450-683-7.

39. Fanchi J. R. Principles of Applied Reservoir Simulation. 3 edition. Elsevier Science Ltd, 2005. 532 pp. ISBN: 9780750679336.

40. Chen Z., Huan G., Ma Y. Computational Methods for Multiphase Flows in Porous Media. SIAM, 2006. 561 pp. ISBN: 0-89871-606-3.

41. Chen Z. Reservoir Simulation: Mathematical Techniques in Oil Recovery. SIAM, 2007. 227 pp. ISBN: 0-89871-640-3.

42. Р. Д. Каневская. Математическое моделирование разработки месторождений нефти и газа с применением гидравлического разрыва пласта. Недра, 1999. 213 с. ISBN: 5-8365-0009-6.

43. Saad Y. Iterative methods for sparse linear systems. Second edition. Philadelphia, PA, USA: SIAM Press, 2003. Pp. xviii + 528. ISBN: 0-89871-534-2.

44. Demmel J., Higham N., Schreiber R. Block LU Factorization: LAPACK Working Note 40. Knoxville, TN 37996, USA: Department of Computer Science, University of Tennessee, Knoxville, 1992. — February. UT-CS-92-149, February 1992.

45. Demmel J. W., Higham N. J., Schreiber R. S. Stability of block *LU* factorization // Numerical linear algebra with applications. 1995. Vol. 2, no. 2. Pp. 173–190.

46. Saad Y., Zhang J. BILUM: Block Versions of Multielimination and Multilevel ILU Preconditioner for General Sparse Linear Systems // SIAM Journal on Scientific Computing. 1999. — November. Vol. 20, no. 6. Pp. 2103–2121.

47. Zhang J. Sparse approximate inverse and multilevel block ILU preconditioning techniques for general sparse matrices // Applied Numerical Mathematics: Transactions of IMACS. 2000. — September. Vol. 35, no. 1. Pp. 67–86.

48. Yun J. H. Block *ILU* preconditioners for a nonsymmetric block-tridiagonal *M*-matrix // BIT Numerical Mathematics. 2000. — September. Vol. 40, no. 3. Pp. 583–605.

49. Kolotilina L. Y., Nikishin A. A., Yeremin A. Y. An incomplete *LU*-factorization algorithm based on block bordering // Numerical linear algebra with applications. 2000. — October/December. Vol. 7, no. 7–8. Pp. 543–567.

50. de Sturler E. Incomplete block LU preconditioners on slightly overlapping subdomains for a massively parallel computer // Applied Numerical Mathematics: Transactions of IMACS. 1995. — December. Vol. 19, no. 1–2. Pp. 129–146. Massively parallel computing and applications (Amsterdam, 1993–1994).

51. Shen C., Zhang J. Parallel two level block ILU preconditioning techniques for solving large sparse linear systems // Parallel Computing. 2002. — October. Vol. 28, no. 10. Pp. 1451–1475.

52. Saad Y. ILUT: a dual threshold incomplete *LU* factorization // Numerical linear algebra with applications. 1994. Vol. 1, no. 4. Pp. 387–402.

53. Chapman A., Saad Y., Wigton L. High-order ILU preconditioners for CFD problems // International Journal for Numerical Methods in Fluids. 2000. Vol. 33, no. 6. Pp. 767–788.

54. Н. С. Бахвалов, Н. П. Жидков, Г. М. Кобельков. Численные методы. Москва: Бином, 2003. 632 с.

55. Aavatsmark I. An Introduction to Multipoint Flux Approximations for Quadrilateral Grids // Computational Geosciences. 2002. Vol. 6, no. 3-4. Pp. 405–432.

56. А. В. Колдоба, Ю. А. Повещенко, Ю. П. Попов. Об одном алгоритме решения уравнения теплопроводности на неортогональных сетках // Дифференциальные уравнения. 1985. Т. XXI, № 7. С. 1273–1276.

57. Aavatsmark I., Barkve T., Bøe O., Mannseth T. Discretization on Unstructured Grids for Inhomogeneous, Anisotropic Media. Part I: Derivation of the Methods // SIAM Journal on Scientific Computing. 1998. — September. Vol. 19, no. 5. Pp. 1700–1716.

58. Aavatsmark I., Barkve T., Bøe O., Mannseth T. Discretization on Unstructured Grids For Inhomogeneous, Anisotropic Media. Part II: Discussion And Numerical Results // SIAM Journal on Scientific Computing. 1998. — September. Vol. 19, no. 5. Pp. 1717–1736.

59. Nordbotten J. M., Aavatsmark I. An Introduction to Multipoint Flux Approximations for Quadrilateral Grids // Computational Geosciences. 2005. Vol. 9, no. 1. Pp. 61–72.

60. Nordbotten J. M., Aavatsmark I., Eigestad G. T. Monotonicity of control volume methods. 2007. — April. Vol. 106, no. 2. Pp. 255–288.

61. Doi S. On parallelism and convergence of incomplete *LU* factorizations // Applied Numerical Mathematics: Transactions of IMACS. 1991. — June. Vol. 7, no. 5. Pp. 417–436.

62. von Laszewski G., Parashar M., Mohamed A. G., Fox G. C. On the parallelization of blocked LU factorization algorithms on distributed memory architectures. 1992. Pp. 170–179.

63. Buoni J. J., Farrell P. A., Ruttan A. Algorithms for *LU* decomposition on a shared memory multiprocessor // Parallel Computing. 1993. — August. Vol. 19, no. 8. Pp. 925–937.

64. Parallel numerical algorithms, Ed. by D. E. Keyes, A. Sameh, V. Venkatakrishnan. Norwell, MA, USA, and Dordrecht, The Netherlands: Kluwer Aca-

demic Publishers Group, 1997. Vol. 4 of ICASE/LaRC interdisciplinary series in science and engineering. Pp. xi + 395. ISBN: 0-7923-4282-8.

65. Pakzad M., Lloyd J. L., Phillips C. Independent columns: A new parallel ILU preconditioner for the PCG method // Parallel Computing. 1997. — June. Vol. 23, no. 6. Pp. 637–647.

66. Fu C., Jiao X., Yang T. Efficient Sparse LU Factorization with Partial Pivoting on Distributed Memory Architectures // IEEE Transactions on Parallel and Distributed Systems. 1998. — February. Vol. 9, no. 2. Pp. 109–115.

67. Schenk O., Gärtner K., Fichtner W. Efficient sparse *LU* factorization with left–right looking strategy on shared memory multiprocessors // BIT Numerical Mathematics. 2000. — March. Vol. 40, no. 1. Pp. 158–176.

68. Basermann A. Parallel block ILUT/ILDLT preconditioning for sparse eigenproblems and sparse linear systems // Numerical linear algebra with applications. 2000. — October/December. Vol. 7, no. 7–8. Pp. 635–648.

69. Li N., Saad Y., Chow E. Crout Versions of ILU for General Sparse Matrices // SIAM Journal on Scientific Computing. 2003. — March. Vol. 25, no. 2. Pp. 716–728.

70. Stark S., Beris A. N. LU decomposition optimized for a parallel computer with a hierarchical distributed memory // Parallel Computing. 1992. — September. Vol. 18, no. 9. Pp. 959–971.

71. Magolu monga Made M., van der Vorst H. A. A generalized domain decomposition paradigm for parallel incomplete LU factorization preconditionings // Future Generation Computer Systems. 2001. — June. Vol. 17, no. 8. Pp. 925–932.

72. Hysom D., Poten A. A Scalable Parallel Algorithm for Incomplete Factor Preconditioning // SIAM Journal on Scientific Computing. 2001. Vol. 22. Pp. 2194–2215.

73. Kaya D., Wright K. Parallel algorithms for LU decomposition on a shared memory multiprocessor // Applied Mathematics and Computation. 2005. — April. Vol. 163, no. 1. Pp. 179–191.

74. Hénon P., Saad Y. A Parallel Multistage ILU Factorization Based on a Hierarchical Graph Decomposition // SIAM Journal on Scientific Computing. 2006. — January. Vol. 28, no. 6. Pp. 2266–2293.

75. И. Е. Капорин, И. Н. Коньшин. Параллельное решение симметричных положительно-определенных систем на основе перекрывающегося разбиения на блоки // Журнал вычислительной математики и математической физики. 2001. Т. 41, № 4. С. 515–528.

76. Kaporin I. E., Konshin I. N. A parallel block overlap preconditioning with inexact submatrix inversion for linear elasticity problems // Numerical linear algebra with applications. 2002. Vol. 9, no. 2. Pp. 141–162.

77. Wallis J. Incomplete Gaussian Elimination as a Preconditioning for Generalized Conjugate Gradient Acceleration // SPE paper 12265. Presented at the SPE Reservoir Simulation Symposium, San Francisco, CA, USA. 1983. 13 pp.

78. Wallis J., Kendall R., Little T. Constraint Residual Acceleration of Conjugate Residual Methods // SPE paper 13536. Presented at the SPE Reservoir Simulation Symposium, Dallas, Texas, USA. 1985. 14 pp.

79. Behie A. Comparison of Nested Factorization, Constrained Pressure Residual, and Incomplete Factorization Preconditionings // SPE paper 13531. Presented

at the SPE Reservoir Simulation Symposium, Dallas, Texas, USA. 1985. 19 pp.

80. Lacroix S., Vassilevski Y. V., Wheeler M. F. Decoupling preconditioners in the implicit parallel accurate reservoir simulator (IPARS) // Numerical linear algebra with applications. 2001. Vol. 8. Pp. 537–549.

81. Lacroix S., Vassilevski Y. V., Wheeler J., Wheeler M. F. Iterative solution methods for modeling multiphase flow in porous media fully implicitly // SIAM Journal on Scientific Computing. 2002. Vol. 25, no. 3. Pp. 905–926.

82. Fung L. S., Dogru A. H. Parallel Unstructured-Solver Methods for Simulation of Complex Giant Reservoirs // SPE Reservoir Evaluation and Engineering. 2008. — December. Vol. 13, no. 4. Pp. 440–446.

83. Khait M. Efficient Parallel Reservoir Simulation Using Multicore Architectures // SPE paper 119107. Presented at the SPE Reservoir Simulation Symposium, Woodlands, Texas, USA. 2009. 5 pp.

84. Al-Shaalan T. M., Klie H., Dogru A. H., Wheeler M. F. Studies of Robust Two Stage Preconditioners for the Solution of Fully Implicit Multiphase Flow Problems // SPE paper 118722. Presented at the SPE Reservoir Simulation Symposium, Woodlands, Texas, USA. 2009. 13 pp.

85. Jiang Y., Tchelepi H. Scalable Multi-Stage Linear Solver for Coupled Systems of Multi-Segment Wells and Complex Reservoir Models // SPE paper 119175. Presented at the SPE Reservoir Simulation Symposium, Woodlands, Texas, USA. 2009. 10 pp.

86. Clees T., Ganzer L. An Efficient Algebraic Multigrid Solver Strategy for Adaptive Implicit Methods in Oil-Reservoir Simulation // SPE Reservoir Evaluation and Engineering. 2010. — September. Vol. 15, no. 3. Pp. 670–681.

87. Stuben K., Clees T., Klie H. et al. Algebraetic Multigrid Methods (AMG) for the Efficient Solution of Fully Implicit Formulations in Reservoir Simulation // SPE paper 105832. Presented at the SPE Reservoir Simulation Symposium, Houston, Texas, USA. 2007. 11 pp.

88. Clees T. An Efficient Algebraic Multigrid Solver Strategy for Adaptive Implicit Methods in Oil Reservoir Simulation // SPE paper 105789. Presented at the SPE Reservoir Simulation Symposium, Houston, Texas, USA. 2007. 11 pp.

89. Hammersley R., Ponting D. Solving Linear Equations In Reservoir Simulation Using Multigrid Methods // SPE paper 115017. Presented at the SPE Russian Oil and Gas Technical Conference and Exhibition, Moscow, Russia. 2008. 9 pp.

90. Stuben K. A Review of Algebraic Multigrid // GMD Report 69. 1999. — November.

91. Stuben K. Algebraic Multigrid (AMG): an introduction with applications // GMD Report 70. 1999. — November.

92. Brandt A. General highly accurate algebraic coarsening // Electronic Transactions on Numerical Analysis (ETNA). 2000. Vol. 10. Pp. 1–20. Multilevel methods (Copper Mountain, CO, 1999).

93. Livne O. E. Coarsening by compatible relaxation // Numerical linear algebra with applications. 2004. — March/April. Vol. 11, no. 2–3. Pp. 205–227.

94. Klie H., Wheeler M. F. Deflation AMG Solvers for Highly Ill-Conditioned Reservoir Simulation Problems // SPE paper 105820. Presented at the SPE Reservoir Simulation Symposium, Houston, Texas, USA. 2007. 9 pp.

95. Killough J. Ninth SPE Comparative Solution Project: A Reexamination of Black-Oil Simulation // SPE paper 29110. Presented at the SPE Reservoir Simulation Symposium, San Antonio, Texas, USA. 1995. 13 pp.

96. Mishev I., Shaw J., Lu P. Numerical Experiments with AMG Solver in Reservoir Simulation // SPE paper 141765. Presented at the SPE Reservoir Simulation Symposium, Woodlands, Texas, USA. 2011. 8 pp.

97. Chien M., Wasserman M., Yardumian H., Chung E. The Use of Vectorisation and Parallel Processing for Reservoir Simulation // SPE paper 16025. Presented at the SPE Reservoir Simulation Symposium, San Antonio, Texas, USA. 1987. 13 pp.

98. Thompson A. M., Bowen G. R. Parallelisation of an Oil Reservoir Simulation // Proceedings of the International Conference and Exhibition on High--Performance Computing and Networking. HPCN Europe 1996. London, UK: Springer-Verlag, 1996. Pp. 20–28.

99. Wallis J., Foster J., Kendall R. A New Parallel Iterative Linear Solution Method for Large-Scale Reservoir Simulation // SPE paper 21209. Presented at the SPE Reservoir Simulation Symposium, Anaheim, California, USA. 1991. 10 pp.

100. Burrows R., Ponting D., Wood L. Paralell Reservoir Simulation with Nested Factorisation // Proceedings of the 5th European Conference on the Mathematics of Oil Reecovery. Leoben, Austria. 1996.

101. Killough J., Foster J., Nolen J. et al. Paralellization of General-Purpose Reservoir // Proceedings of the 5th European Conference on the Mathematics of Oil Reecovery. Leoben, Austria. 1996.

102. Wallis J., Nolen J. Efficient Iterative Linear Solution of Locally Refined Grids using Algebraic Multilevel Approximate Factorizations // SPE paper 25239. Presented at the SPE Reservoir Simulation Symposium, New Orleans, Louisiana, USA. 1993. 11 pp.

103. Magras J.-F., Quandalle P. High-Performance Reservoir Simulation With Parallel ATHOS // SPE paper 66342. Presented at the SPE Reservoir Simulation Symposium, Houston, Texas, USA. 2001. 8 pp.

104. Li K.-M. A Linear Equation Solver for Massively Parallel Computers // SPE paper 25241. Presented at the SPE Reservoir Simulation Symposium, New Orleans, Louisiana, USA. 1993. 6 pp.

105. Dogru A., Li K., Sunaidi H. et al. A Massively Parallel Reservoir Simulator for Large Scale Reservoir Simulation // SPE paper 51886. Presented at the SPE Reservoir Simulation Symposium, Houston, Texas, USA. 1999. 28 pp.

106. Shiralkar G., Stephenson R., Joubert W. et al. Falcon: A Production Quality Distributed Memory Reservoir Simulator // SPE paper 37975. Presented at the SPE Reservoir Simulation Symposium, Dallas, Texas, USA. 1997. 9 pp.

107. Collins D. A., Grabenstetter J. E., Sammon P. H. A Shared-Memory Paralell Black-Oil Simulator with a Parallel ILU Linear Solver // SPE paper 79713. Presented at the SPE Reservoir Simulation Symposium, Houston, Texas, USA. 2003. 8 pp.

108. DeBaun D., Byer T., Childs P. et al. An Extensible Architecture for Next Generation Scalable Parallel Reservoir Simulation // SPE paper 93274. Presented at the SPE Reservoir Simulation Symposium, Houston, Texas, USA. 2005. 13 pp.

109. Dogru A. H., Fung L. S. K., Middya U. et al. An Next-Generation Parallel Reservoir Simulator for Giant Reservoirs // SPE paper 119272. Presented at the SPE Reservoir Simulation Symposium, Woodlands, Texas, USA. 2009. 29 pp.

110. Chapman B., Jost G., van der Pas R. Using OpenMP, Portable Shared Memory Parallel Programming. Cambridge, MA, USA: MIT Press, 2007. 353 pp. ISBN: 0-262-53302-2.

111. Gropp W., Lusk E., Skjellum A. Using MPI, Portable Parallel Programming with the Message Passing Interface. 2 edition. Cambridge, MA, USA: MIT Press, 1999. 395 pp. ISBN: 0-262-57132-3.

112. Culler D. E., Singh J. P., Gupta A. Parallel computer architecture: a hardware/software approach. San Francisco, CA, USA: Gulf Professional Publishing, 1999. 1025 pp. ISBN: 1-55860-343-3.

113. Laudon J., Lenoski D. The SGI Origin: a ccNUMA highly scalable server // ACM SIGARCH Computer Architecture News. 1997. — May. Vol. 25, no. 2. Pp. 241–251.

114. Chapman B. Parallel Application Development with the Hybrid MPI + OpenMP Programming Model // Lecture Notes in Computer Science. 2002. Vol. 2474. Pp. 13–27.

115. Mahinthakumar G., Saied F. A Hybrid MPI-OpenMP Implementation of an Implicit Finite-Element Code on Parallel Architectures // The International Journal of High Performance Computing Applications. 2002. — Winter. Vol. 16, no. 4. Pp. 371–393.

116. Aversa R., Martino B. D., Rak M. et al. Performance prediction through sim-

ulation of a hybrid MPI/OpenMP application // Parallel Computing. 2005. — October/December. Vol. 31, no. 10–12. Pp. 1013–1033.

117. Nakajima K. Parallel iterative solvers for finite-element methods using an OpenMP/MPI hybrid programming model on the Earth Simulator // Parallel Computing. 2005. — October/December. Vol. 31, no. 10–12. Pp. 1048–1065.

118. Rabenseifner R., Hager G., Jost G. Hybrid MPI/OpenMP Parallel Programming on Clusters of Multi-Core SMP Nodes // Proceedings of the 17th Euromicro International Conference on Parallel, Distributed and Network-Based Processing, PDP 2009, Weimar, Germany, 18-20 Febuary 2009. 2009. Pp. 427–436.

119. Jost G., Robins B. Experiences using hybrid MPI/OpenMP in the real world: Parallelization of a 3D CFD solver for multi-core node clusters // Scientific Programming. 2010. Vol. 18, no. 3-4. Pp. 127–138.

120. Marjanovic V., Labarta J., Ayguadé E., Valero M. Effective communication and computation overlap with hybrid MPI/SMPSs // ACM SIGPLAN Notices. 2010. — May. Vol. 45, no. 5. Pp. 337–338.

121. Jin H., Jespersen D., Mehrotra P., Biswas R. High Performance Computing Using MPI and OpenMP on Multi-core Parallel Systems // Parallel Computing. 2011. — February.

122. Bircsak J., Craig P., Crowell R. et al. Extending OpenMP for NUMA machines // Scientific Programming. 2000. Vol. 8, no. 3. Pp. 163–181.

123. Tao J., Karl W., Schulz M. Memory access behavior analysis of NUMA-based shared memory programs // Scientific Programming. 2002. Vol. 10, no. 1. Pp. 45–53.

124. Williams S., Oliker L., Vuduc R. et al. Optimization of sparse matrix–vector multiplication on emerging multicore platforms // Parallel Computing. 2009. Vol. 35, no. 3. Pp. 178–194.

125. Hassanein W. M., Hammad M. A., Rashid L. Characterizing the Performance of Data Management Systems on Hyper-Threaded Architectures // Proceedings of the 18th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD). 2006.

126. Christie M., Blunt M. Tenth SPE Comparative Solution Project: A Comparison of Upscaling Techniques // SPE Reservoir Evaluation and Engineering. 2001. — August. Vol. 4, no. 4. Pp. 308–317.

127. Gratien J., Guignon T., Magras J. et al. Scalability and Load Balancing Problems in Parallel Reservoir Simulation // SPE paper 106023. Presented at the SPE Reservoir Simulation Symposium, Houston, Texas, USA. 2007. 6 pp.