МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Кафедра вычислительной математики

Стабилизированный метод бисопряженных градиентов(BICGStab)

Курсовой проект

Малиев Эмиль Енгибарович
студента 3 курса,
специальность «прикладная математика»

Научный руководитель:
старший преподователь кафедры вычислительной математики,
магистр физико-математических наук
Левчук Е.А.

Минск, 2023

АННОТАЦИЯ

Малиев Э. Е. Стабилизированный метод бисопряженных градиентов: Курсовая проект / Минск: БГУ, 2023. — 25 стр.

В данной работе происходит исследование стабилизированного метода биспореженных градиентов, его реализация и сравнение с методом Якоби и метотодом минимальных невязок.

АНАТАЦЫЯ

Маліеў Э. Е. Стабілізаваны метад бисапряженых градыентаў : Курсавы праект / Мінск: БДУ, 2023. - 25 ст.

У дадзенай працы адбываецца даследаванне стабілізаванага метаду бисапряженых градыентаў, яго рэалізацыя і параўнанне з метадам Якобі і метатадам мінімальных неввяз.

ANNOTATION

Maliev E. E. BiConjugate Gradient Stabilized method: Course project / Minsk: BSU, 2023. - 25 p.

In this work, we study the BiConjugate Gradient Stabilized method, its implementation and comparison with the Jacobi method and minimal residual method.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Линейные системы уравнений играют важную роль в различных областях науки и техники, предоставляя мощный инструмент для моделирования и анализа разнообразных явлений. От физики и инженерии до информатики и экономики, линейные системы становятся фундаментальным инструментом для описания взаимосвязей между переменными. Эти системы часто возникают в результате дискретизации дифференциальных уравнений и других математических моделей. Дифференциальные уравнения являются мощным инструментом для описания различных явлений в природе и технике. Однако, для решения практических задач, связанных с такими уравнениями, необходимо перейти от бесконечного континуума к конечному числу узлов или точек. Этот процесс дискретизации приводит к появлению системы линейных уравнений, которая может быть решена для приближенного нахождения решения исходного дифференциального уравнения. Примерами могут служить уравнения теплопроводности, уравнения Навье-Стокса в механике жидкостей, а также уравнения в частных производных, используемые в задачах физики, биологии, экономики и других областях. Переход от непрерывных моделей к дискретным системам становится неотъемлемой частью математического моделирования и численного анализа.

Однако, с ростом сложности моделей и увеличением объема данных, связанных с реальными проблемами, возникает необходимость в эффективных методах решения линейных систем. Задача поиска точного решения становится крайне трудоемкой, особенно при работе с большими и разреженными матрицами. В связи с этим, актуальность разработки итерационных методов решения линейных систем, которые обеспечивают высокую степень вычислительной эффективности, становится неоспоримой.

В свете современных вычислительных возможностей и требований к обработке больших объемов данных, эффективные методы решения линейных систем приобретают особенное значение. Итерационные методы становятся важным инструментом для нахождения приближенных решений, обеспечивая компромисс между точностью и вычислительной сложностью, что делает их незаменимыми в современных прикладных науках и инженерии.

Цель настоящего исследования заключается в изучении, реализации и

сравнении эффективности одного из таких итерационных методов - BiCGStab (Stabilized BiConjugate Gradient) с двумя другими иттерационными методами методами, один из которых проекционный. Анализ и сопоставление данных методов позволит лучше понять их применимость в различных сценариях и определить оптимальный выбор при решении конкретных задач линейной алгебры.

1 Свойства и структура алгоритма

1.1 Общее описание алгоритма

Стабилизированный метод бисопряженных градиентов, так же известный как BiCGStab (BiConjugate Gradient Stabilized method), был предложен голландским математиком Хенком ван дер Ворстом в 1992 году. Статья, в которой описывался метод, стала самым цитируемым научным трудом в области математики в 1990х годах.

Метод появился, как ответ на задачу о построении способа решения систем линейных алгебраических уравнений произвольного вида. На момент появления BiCGStab уже существовали методы решения СЛАУ произвольного типа, в том числе и в комплексном пространстве, например метод бисопряженных градиентов. Однако существующие на тот момент методы либо обладали недостаточной скоростью сходимости, либо использовали в вычислениях тяжелые операции, такие как транспонирования матрицы, которые усложняли распараллеливание на многопроцессорных машинах.

BiCGStab является итерационным методом решения систем линейных алгебраических уравнений. Мощь метода заключается в том, что его можно применять к матрицам общего вида, в том числе с комплексными коэффициентами. В процессе выполнения BiCGStab не использует для вычисления плохо распараллеливаемых операций, не допускает накопления погрешностей округления и нестабильного поведения невязки. Эти факторы выгодно отличают его от предшественников: метода бисопряженных градиентов и квадратичного метода сопряженных градиентов, из которого BiCGStab фактически и вытекает. К преимуществам так же можно добавить более высокую в общем случае скорость сходимости относительно ближайших конкурентов. Тем не менее, для некоторых видов СЛАУ BiCGStab может уступать более простым методам.

Благодаря свойствам алгоритма область применения метода весьма широка. Фактически, он используется во всех задачах, где требуется решать СЛАУ, в том числе СЛАУ большой размерности, к которым сводится большое количество численных методов, применяемых при решении задач математического моделирования.

BiCGStab относится к классу так называемых проекционных методов. Основная идея методов данного класса: искать итерационным способом наилучшее приближение решения в некотором подпространстве пространства $\mathbb{R}^n$, и чем шире подпространство, в котором ищется приближение решения, тем ближе будет приближенное решение к точному на каждой итерации. В качестве такого подпространства используются пространства Крылова, порожденные матрицей системы и невязкой первого приближения:

$$K_m(A, r_0) = \text{span}(r_0, Ar_0, A^2r_0, \ldots, A^{m-1}r_0).$$

Изложим основную идею любого проекционного метода, включая BiCGStab. Для простоты выкладок и рассуждений будем считать, что матрица системы и ее правая часть вещественны. Без ограничения общности все дальнейшие рассуждения могут быть перенесены и на комплексный случай.

Пусть решается система $Ax = b$ с невырожденной матрицей $A \in \mathbb{R}^{n\times n}$ и ненулевым вектором $b \in \mathbb{R}^n$. Имеются два подпространства K_n и L_n. Требуется найти вектор $x \in K_n$, который обеспечивает решение системы, оптимальное относительно подпространства L, то есть требуется выполнение условия Петрова-Галёркина: $(Ax, z) = (b, z), \forall z \in L_n$. В данном случае вектор x называется обобщенным решением системы.

Условие Петрова-Галеркина можно записать и через определение невязки. Пусть $r = b - Ax$ — невязка, тогда условие Петрова-Галеркина перепишется в виде: $(b - Ax, z) = (r, z), \forall z \in L_n$. Это равносильно тому, что вектор невязки ортогонален пространству L_n.

Пусть для СЛАУ известно некоторое приближение x_0 точного решения x^*. Данное приближение требуется уточнить поправкой $\delta : b - Ax_0 + \delta \perp L$. Невязка уточненного решения имеет вид: $r_{\text{new}} = b - Ax_0 + \delta = r_0 - A\delta$. Условие Петрова-Галеркина тогда можно переписать так: $(r_{\text{new}}, z) = r_0 - (A\delta, z) = 0, \forall z \in L$.

Пусть $\dim K = \dim L = m$, $V = (v_1, \ldots, v_n) \in \mathbb{R}^{n\times m}$, $W = (w_1, \ldots, w_n) \in \mathbb{R}^{n\times m}$, где V и W – матрицы, столбцы которых образуют базисы пространств K и L соответственно. Пусть также $\delta = Vy$, где y – коэффициенты разложения вектора поправки по базису пространства K. Тогда новое приближение решения можно записать в виде: $x_1 = x_0 + \delta = x_0 + Vy$. Из условия ортогональности Петрова-Галеркина получаем, что: $(r - AVy, Wq) =$

$(W^T r_0 - AVy, q) = 0$. Поскольку $q \neq 0$, то получаем СЛАУ для вектора y: $W^T AVy = W^T r_0$. Если предположить, что матрица $W^T AV$ невырождена, то вектор y однозначно определяется и равен: $y = (W^T AV)^{-1} W^T r_0$. Подставляя найденные коэффициенты разложения в выражение для первого приближения, получаем: $x_1 = x_0 + \delta = x_0 + (W^T AV)^{-1} W^T r_0 y$.

Подведем итог. Общий порядок шагов любого проекционного метода следующий:

1. Выбрать пару подпространств K и L
2. Найти базис V пространства K и базис W пространства L;
3. Вычислить невязку
4. Найти $y = W^T AV^{-1} W^T r_0$
5. Положить $x = x + Vy$
6. Если не достигнут критерий остановки итерационного процесса, то повторить алгоритм, начиная с первого шага.

Как будет показано далее, в BiCGStab не требуется вычислять матрицу $W^T AV^{-1}$ в явном виде.

1.2 Математическое описание алгоритма

Рассмотрим СЛАУ с вещественными коэффициентами:

$$Ax = b, \quad A \in \mathbb{R}^{N \times N}, \quad b \in \mathbb{R}^N.$$

В основу алгоритма ложится идея проекционного метода и использование свойства A-бисопряженности системы векторов, а именно: векторы $p_1, p_2, \ldots$ и $\overline{p}_1, \overline{p}_2, \ldots$ бисопряжены, если $(Ap_i, p_j) = 0$, при $i \neq j$. Фактически, данное условие эквивалентно биортогональности относительно энергетического скалярного произведения.

Общая концепция проекционных методов предписывает нам выбрать два подпространства. В нашем случае подпространства мы выбираем два

подпространства, задаваемые матрицей системы A, вектором невязки на нулевой итерации r_0 и вектором $\overline{r}_0$, который выбирается из условия $(r_0, \overline{r}_0) \neq 0$:

$$K = K_k(A, r_0), \quad L = K_k(A, \overline{r}_0).$$

Метод основан на построении биортогонального базиса $p_0, p_1, \ldots, p_k, \ldots$ пространства $K_k(A, r_0)$ и вычислении поправки такой, что новое приближение на очередной итерации было бы ортогонально второму подпространству Крылова. Базисные вектора строятся до тех пор, пока не будет достигнут критерий остановки итерационного процесса, а каждое последующее приближение формируется как сумма приближения на предыдущей итерации и найденной поправки. Критерием останова итерационного процесса является достижение невязкой значения, которое меньше некоторого наперед заданного ε.

Основной итерационный процесс задается формулами:

$$x_{k+1} = x_k + \alpha_k p_k + \omega_k s_k,$$
$$p_{k+1} = r_{k+1} + \beta_k p_k - \omega_k A p_k,$$

где:

x_{k+1} — приближение на следующей итерации,

x_k — приближение на предыдущей итерации,

p_k — базисный вектор подпространства Крылова,

α_k, ω_k — вспомогательные скаляры, определяемые для каждого шага итерации,

s_k — вспомогательный вектор,

p_{k+1} — $(k+1)$-ый базисный вектор,

r_{k+1} — невязка $(k+1)$-ой итерации,

β_k — вспомогательный коэффициент.

Вспомогательный вектор выражается следующим образом:

$$s_k = r_k - \alpha_k A p_k.$$

Вспомогательные коэффициенты вычисляются по формулам:

$$\alpha_k = \frac{(r_k, \overline{r}_0)}{(Ap_k, \overline{r}_0)}, \quad \omega_k = \frac{(As_k, s_k)}{(As_k, As_k)}, \quad \beta_k = \frac{(r_{k+1}, \overline{r}_0)}{(r_k, \overline{r}_0)}.$$

Подведем итог и запишем итерационный процесс:

Подготовительный этап:

1. Выбирается некоторое начальное приближение x_0.
2. По выбранному приближению находится невязка $r_0 = b - Ax_0$.
3. Из условия $(r_0, \overline{r}_0) \neq 0$ выбирается вектор $\overline{r}_0$.
4. Положим $p_0 = r_0$.
5. Устанавливаем счетчик итераций: $n = 0$.

Итерационный процесс:

1. Находим α_k.
2. По найденному α_k вычисляем s_k.
3. По найденному s_k вычисляется вспомогательный коэффициент ω_k.
4. Находим новое приближение x_{k+1}.
5. Вычисляется невязка r_{k+1}. Если выполнено условие $\|r_{k+1}\| < \varepsilon\|b\|$, то алгоритм завершается.
6. Вычисляется вспомогательный коэффициент β_k.
7. Вычисляется новый базисный вектор: p_{k+1}.
8. Увеличивается номер итерации: $n = n + 1$.
9. Переход к первому шагу итерационного процесса.

1.3 Последовательная сложность алгоритма

Всего в вычислительном блоке алгоритма присутствует 20 трудоемких операций: 2 операции умножения матрицы на вектор, 6 скалярных произведений и 6 сложений/вычитаний векторов и 6 умножений вектора на число.

Каждое скалярное произведение требует N произведений и $N-1$ сложение, то есть $2N-1$ операций. Скалярных произведений всего 6 штук (включая поиск нормы), что дает нам $12N-6$ операций.

Умножение матрицы размерности $N \times N$ на вектор эквивалентно взятию еще N скаляров, и так как таких операций в вычислительном ядре 2, то дополнительно имеется еще $2N(2N-1)$ операций.

Сложение или вычитание векторов требует $6N$ соответствующих операций. Столько же требует 6 произведений вектора на число.

Итого: $2N(2N-1)+12N-6+6N+6N = 4N^2+22N-6$ операций за одну итерацию. Домножая это на количество итераций I, получаем итоговую сложность алгоритма: $I \cdot (4N^2+22N-6)$.

Имеются также операции как извлечение квадратного корня и деление вещественных чисел друг на друга, но они не принимаются в расчет, так как они не влияют на итоговый результат. Окончательно можно заключить, что последовательная сложность алгоритма эквивалентна $O(I \cdot N^2)$.

1.4 Информационный граф

На графе (Рисунок 1.1) представлены связи между операциями, выполняющимися в первых трех итерациях метода. При выполнении большего, чем 3, количества итераций, структура каждой новой будет повторять структуру третьей. Связи между итерациями не представлены, т.к. они не важны для понимания параллельной структуры алгоритма. Серым обозначено перемножение чисел, красным - умножение числа на вектор

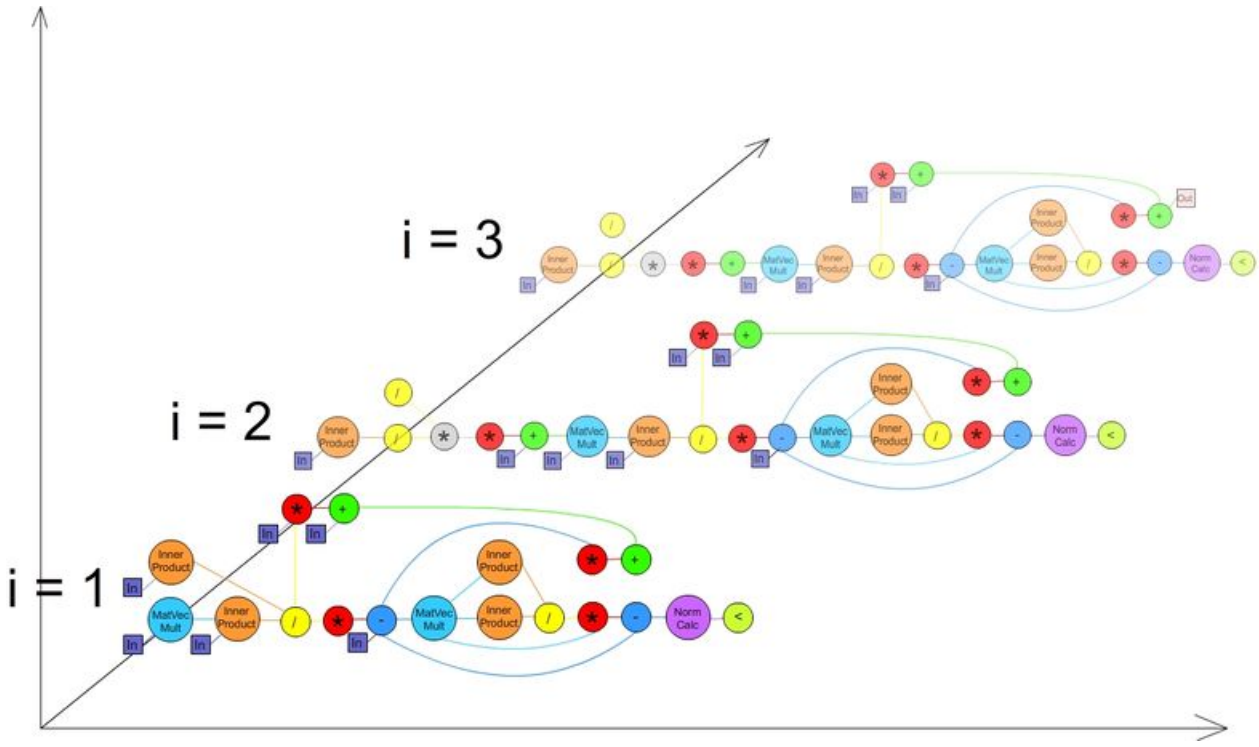

Рис. 1.1 – Информационный граф

1.5 Ресурс параллелизма алгоритма

Основной прирост производительности параллельной версии алгоритма дает использование параллельных операций вычисления скалярного произведения, суммы векторов, а также произведения матрицы на вектор. Каждая из операций вычисления скалярного произведения и суммы векторов, выполняемая на p процессорах, за счет метода сдваивания может ускориться до $\frac{p}{\log_2 p}$ раз. Ускорение же умножения матрицы на вектор может достигать p раз.

Ширина и высота ярусно-параллельной формы графа(ЯПФ) графа данного алгоритма напрямую зависит от конкретных реализаций макроопераций и входных данных.

Для последовательно-параллельной реализации операции взятия скалярного произведения в наиболее оптимальном случае ($p = \sqrt{N}$) высота будет равна $2\sqrt{N} - 1$, а ширина $\sqrt{N}$, где N - это длина вектора решения.

Высота ЯПФ простейшего алгоритма умножения матрицы на вектор равна $2N$, а ширина m, где N - это уже упомянутая длина вектора (она же ширина матрицы), а m - высота матрицы.

Высота ЯПФ операции умножения вектора на число равна Np, а ширина

p.

Высота ЯПФ операции вычисления разности векторов равна $\log_2 p$, а ширина p.

В конечном счете общую высоту ЯПФ одной итерации алгоритма можно оценить, как $6\sqrt{N}-3+4N+3+1+3\lceil Np\rceil+2\lceil Np\rceil = 4N+6\sqrt{N}+5\lceil Np\rceil+1$, а ширину, как $2\sqrt{N}$, если не брать в расчет проверку критерия сходимости и упрощенность первой итерации по сравнению с остальными.

1.6 Свойства алгоритма

BiCGStab хорошо зарекомендовал себя для решения систем линейных алгебраических уравнений, поскольку обладает некоторыми важными свойствами:

- Благодаря такому применению подхода Петрова-Галеркина и технике ортогонализации, он численно устойчив.
- Не меняется вид матрицы в процессе решения.
- Эффективен для решения систем большой размерности с разреженной матрицей, поскольку в методе самая трудоемкая операция - умножение матрицы на вектор.
- Лучше подходит под распределенные системы, поскольку не содержит операций транспонирования матрицы.
- Применим для решения систем с несимметричными матрицами.
- Вычислительная мощность: $N/4$.

2 Пргораммная реализация алгоритма

2.1 Особенности реализации последовательного алгоритма

Для начало нужно задать некоторое начальное приближение x_0. Начальное приближение в методе BiCGStab может зависеть от конкретной задачи и доступной информации о системе уравнений. Ниже приведены несколько способов задания начального приближения:

1. *Нулевое начальное приближение:*

   Просто установите начальное приближение $\mathbf{x}_0$ равным нулевому вектору. Этот метод может быть эффективен в том случае, если у вас нет никакой дополнительной информации о системе уравнений.

2. *Использование предыдущего решения:*

   Если у вас уже есть решение близкой задачи или вы провели предыдущие вычисления с похожими параметрами, можете использовать решение из предыдущего расчета в качестве начального приближения.

3. *Приближение методом Якоби или Гаусса-Зейделя:*

   Вы можете использовать результаты нескольких итераций метода Якоби или метода Гаусса-Зейделя в качестве начального приближения. Это особенно полезно, если эти методы быстро сходятся.

4. *Использование других методов предварительной обработки:*

   Некоторые задачи требуют использования специализированных методов предварительной обработки данных для создания хорошего начального приближения.

Выбор оптимального метода зависит от конкретных особенностей задачи. В некоторых случаях проведение нескольких начальных итераций метода Якоби, например, может помочь улучшить начальное приближение перед запуском BiCGStab. Экспериментирование и адаптация к конкретной задаче часто требуются для достижения наилучших результатов. Так как сравнение метода BICGSTAB будет производится с таким иттерационным

проекционным методам, как метод Якоби, то особенностью задачи будет то, что матрицы, на которых будут производится сравнение методов, имеют свойства диагонального преобладания. Соответственно прибегнем к первому пункту и зададим нулевое начально приближение.

После установления начального приближения вычисляем начальный остаток $r_0 = b - Ax_0$. Далее устанавливаем начальные значения для вспомогательных векторов:

$$\hat{r}_0 = r_0,$$
$$\hat{p}_0 = 0,$$
$$\hat{v}_0 = 0.$$

Задаем начальные значения для параметров: $\rho_0 = \alpha_0 = \omega_0 = 1.0$.

Остается лишь задать точность ε (10^{-5}) , при котрой будет остановлен иттерационный процесс($\|r_{k+1}\| < \varepsilon\|b\|$).

Итерационный процесс: Для каждой итерации $k = 1, 2, \ldots$:

1. Вычислите $\rho_k = \langle \hat{r}, r \rangle$.
2. Вычислите $\beta_k = \frac{\rho_{k-1}}{\rho_k} \frac{\alpha_{k-1}}{\omega_{k-1}}$.
3. Обновите направляющий вектор $\hat{p}_k = r + \beta_k(\hat{p}_{k-1} - \omega_{k-1}\hat{v}_{k-1})$.
4. Вычислите $\hat{v}_k = Ap_k$.
5. Вычислите $\alpha_k = \frac{\rho_k}{\langle \hat{r}, \hat{v}_k \rangle}$.
6. Обновите приближенное решение $x_k = x_{k-1} + \alpha_k \hat{p}_k$.
7. Обновите остаток $r_k = r_{k-1} - \alpha_k \hat{v}_k$.
8. Вычислите $\hat{r}_k^T = A^T r_k - \beta_k A^T \hat{v}_k$.
9. Вычислите $\omega_k = \frac{\langle \hat{r}_k^T, r_k \rangle}{\langle \hat{r}_k^T, \hat{r}_k^T \rangle}$.
10. Обновите $x_k = x_k + \omega_k r_k$.
11. Если выполнено условие сходимости, завершите итерационный процесс.

Псевдокод реализованного метода BICGStab:

Algorithm 1 BICGStab Algorithm

procedure BICGSTAB($A, b, x_0, e, \max\_iter$)

$n \leftarrow \text{len}(b)$

$x \leftarrow \text{zeros}(n)$ **if** x_0 **is None else** x_0

$r \leftarrow b - A \cdot x$

$r\hat{} \leftarrow r.\text{copy}()$

$p \leftarrow \text{zeros}(n)$

$v \leftarrow \text{zeros}(n)$

$s \leftarrow \text{zeros}(n)$

$\rho \leftarrow \alpha \leftarrow \omega \leftarrow 1.0$

for $k \leftarrow 0$ **to** max_iter **do**

$\rho_{\text{new}} \leftarrow r\hat{} \cdot r$

$\beta \leftarrow (\rho_{\text{new}}/\rho) \cdot (\alpha/\omega)$

$p \leftarrow r + \beta \cdot (p - \omega \cdot v)$

$v \leftarrow A \cdot p$

$\alpha \leftarrow \rho_{\text{new}}/(r\hat{} \cdot v)$

$s \leftarrow r - \alpha \cdot v$

$t \leftarrow A \cdot s$

$\omega \leftarrow t \cdot s/(t \cdot t)$

$x \leftarrow x + \alpha \cdot p + \omega \cdot s$

$r \leftarrow s - \omega \cdot t$

if $\|r\| < e \cdot \|b\|$ **then**

return $x, k + 1, \|r\|$

end if

$\rho \leftarrow \rho_{\text{new}}$

end for

raise ValueError("Метод BICGStab не сошелся за максимальное количество итераций")

end procedure

2.2 Результаты

Входными данными для реализации и сравнения метода BICGStab с двумя иттерационными методами, один из которых проекционный, будут матрицы разных типов(плотная 3000x3000, трехдиагональная 10000x10000 и положительно определенная симмтеричная 3000x3000), с соотвутсвующими вектороми b. Матрицы задавались случайно, но со свойством диагонального преобладания,так как сравнивать будем с методом Якоби. Это необходимо для сходимости метода Якоби. Для сравнения с положительно опрделенной симмтеричной матрицы, также реализован метод минимальных невязок. Положительно определенную симмтеричную матрицу получаем путем математических преобразований плотной матрицы $A^T Ax = A^T b$.

Матрица	Плотная	Трехдиагональная	Положительно определенная симметричная
Количество иттераций	168	17	4049
Время выполнения	1.0193 сек	1.133 сек	21.9224 сек

Таблица 1 – Результаты BICGStab

Матрица	Плотная	Трехдиагональная
Количество иттераций	49	5
Время выполнения	1.2152814865112305 сек	0.759965181350708 сек

Таблица 2 – Результаты метода Якоби

Матрица	Положительно опредленная симмтеричная
Количество иттераций	4101
Время выполнения	10.943423748016357 сек

Таблица 3 – Результаты метода минимальных невязок

2.3 Анализ и сравнение

Сравним и проанализируем полученные результаты на матрицах разного типа.

BICGStab и метод Якоби:

- Плотная матрица. Для плотной матрицы пристутсвует большая разница в количестве иттераций(168 и 49). Следовательно скорость сходимости метода Якоби намного быстрее. Это можно объяснить тем, что матриа обладает диагональным свойством и BICGSTab, как метод, более подходит для разреженных матриц, и его эффективность может снижаться при работе с плотными матрицами. При это скорость выполнения программы приблизительно равен, BICGStab даже немного быстрее. Отсюда можно сделать вывод, что одна иттерация метода Якоби требует большего времени, чем у метода BICGStab, а значит вычислительная сложность Якоби выше, чем последовательного алгоритма BICGStab.

- Трехдиагональная матрица. В данном случае, метод Якоби оказывается более эффективным, как в отношении количества итераций, так и времени выполнения. Метод Якоби может эффективно использовать структуру трехдиагональной матрицы, так как ему необходимо обрабатывать только три диагонали на каждой итерации, что уменьшает ее вычислительную мощность. И не забываем о том, что матрица так же обладает свойством диагонального преобладания.

BICGStab и метод минимальных невязок:

- Положительно опрделенная симметричная матрица. В данном случае, метод минимальных невязок оказывается более эффективным в смысле времени выполнения, несмотря на близкое количество итераций. Метод минимальных невязок хорошо работает с симметричными матрицами, так как он сохраняет эту симметрию на каждой итерации. Свойство положительной определённости также может влиять на эффективность метода минимальных невязок. При близком количестве иттераций методов скорость выполнения различаются в 2 раза. Отсюда можно

сделать вывод, что вычислительная мощность метода минимальных невязок лучше, чем у последовательного алгоритма BICGStaB.

Вывод: Последовательный алгоритм BICGStab менее эффективен в скорости сходимости по сравнению с методом Якоби, и его вычислительная сложность выше, чем у метода минимальных невязок. Но стоит подметить два фактора:

1. Для сравнения методов пришлось матрицам задвать определенные свойства, такие как, диагональное преобладание, симметричность и положительно опрделенность. Это требовалось для сходимости двух других методов, т.е. для метода Якоби и для метода минимальных невязок. BICGStab даже можно применять к матрицам общего вида, в том числе с комплексными коэффициентами.

2. Ресурс параллелизма BICGStab позволит в несколько раз увеличить его вычислительную мощность.В процессе выполнения BiCGStab не использует для вычисления плохо распараллеливаемых операций, не допускает накопления погрешностей округления и нестабильного поведения невязки.

Эти факторы и делают этот метод мощным и частоиспользуемым.

ЗАКЛЮЧЕНИЕ

В ходе курсовой работы был изучен стабилизированный метод бисопряженных градиентов(BICGStab). В процессе изучения метода были выявлены свойства его алгоритма, причины популярности метода,ресурс параллелизма алгоритма. Так же данный метод был реализован, его последвательный алгоритм, проведено сравнение с иттерциоными методами Якоби и минимальных невязок на матрицах различных типов. При исследовании результатов был провден анализ, выявлены причины отличий в быстроедействии методов.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Саад Ю. Итерационные методы для разреженных линейных систем. Том 1. – М.: Издательство Московского университета, 2013. – 324 с

2. Электрон. ресурс – https://edufpmi.bsu.by/ Численные методы (5 семестр, 1 поток)/ Лекции], лекция «Методы крыловского типа».

3. Электрон. ресурс – https://algowiki-project.org/ru/Стабилизированный-метод-бисопряженных-градиентов(BiCGStab).

4. Barrett R., Berry M., Chan T.F. - Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition.

5. Vorst van der H. Iterative Krylov methods for large linear systems. – Cambridge: Cambridge University Press, 2003. – 221 p.

```
import numpy as np
import time

def BICGSTAB(A, b, x0=None, e=1e-5, max_iter=100000):
    n = len(b)
    x = np.zeros(n) if x0 is None else x0
    r = b - np.dot(A, x)
    r_hat = r.copy()
    p = np.zeros(n)
    v = np.zeros(n)
    s = np.zeros(n)
    rho = alpha = omega = 1.0

    for k in range(max_iter):
        rho_new = np.dot(r_hat, r)
        beta = (rho_new / rho) * (alpha / omega)
        p = r + beta * (p - omega * v)
        v = np.dot(A, p)
        alpha = rho_new / np.dot(r_hat, v)
        s = r - alpha * v
        t = np.dot(A, s)
        omega = np.dot(t, s) / np.dot(t, t)
        x = x + alpha * p + omega * s
        r = s - omega * t

        if np.linalg.norm(r) < e:
            return x, k + 1, np.linalg.norm(r)

        rho = rho_new

    raise ValueError("Метод BICGSTAB не сошелся за максимальное
    количество итераций")
```

```
# Плотная матрица
b = np.random.rand(3000)
n = 3000
diagonal_values = np.random.rand(n) * 10
off_diagonal_values = np.random.rand(n, n)

for i in range(n):
    off_diagonal_values[i, i] = 0
    row_sum = np.sum(np.abs(off_diagonal_values[i]))
    off_diagonal_values[i] /= row_sum

A = np.diag(diagonal_values) + off_diagonal_values

# Трехдиагональная матрица
main_diag = [np.random.uniform(10,100) for _ in range(10000)]
sub_diag = np.random.rand(9999)
sup_diag = np.random.rand(9999)
tridiagonal_A = np.diag(main_diag) + np.diag(sub_diag, k=-1)
+ np.diag(sup_diag, k=1)
tridiagonal_b = np.random.rand(10000)

# Положительно определенная симметричная матрица
sym_b = A.T.dot(b)
sym_A = np.matmul(A.T, A)
```

```
import numpy as np

def jacobi_convergence(A, b, x0=None, tol=1e-5, max_iter=100000):
    n = len(b)
    x = np.zeros(n) if x0 is None else x0
    residuals = []

    for k in range(max_iter):
        x_new = np.zeros(n)
        for i in range(n):
            sigma = np.dot(A[i, :i], x[:i]) + np.dot(A[i, i+1:],
            x[i+1:])
            x_new[i] = (b[i] - sigma) / A[i, i]

        residual = np.linalg.norm(A @ x_new - b, ord=np.inf)
        residuals.append(residual)

        if residual < tol:
            return x_new, k + 1, np.linalg.norm(A @ x_new - b,
            ord=np.inf)

        x = x_new

    raise ValueError("Метод Якоби не сошелся за максимальное
    количество итераций")
```

```
def least_squares(A, b, max_iter=10000, tol=1e-5):
m, n = A.shape
x = np.zeros(n)
r = b - np.dot(A, x)
p = r.copy()

for k in range(max_iter):
    Ap = np.dot(A, p)
    alpha = np.dot(r, r) / np.dot(Ap, p)
    x = x + alpha * p
    r_new = r - alpha * Ap

    if np.linalg.norm(r_new) < tol:
        break

    beta = np.dot(r_new, r_new) / np.dot(r, r)
    p = r_new + beta * p
    r = r_new

return x, k + 1, np.linalg.norm(r_new)
```