Comparison of genetic algorithms with heuristic and classical optimization methods is compared. The classic methods include coordinate descent method, gradient descent method, conjugate gradient method (Fletcher-Reeves method), Newton method and Marquardt method, and among the heuristic methods, bat algorithm and simulated annealing.

Conclusions. *Genetic algorithms have been used for minimization of functions and solve the traveling salesman problem. The efficiency of genetic algorithms is studied depending on the values of its parameters when finding the global minimum of some test functions. Comparison of genetic algorithms with bat algorithm, simulated annealing and classical optimization methods are compared. Continuous genetic algorithm and Binary genetic algorithm have been found to perform better for most test functions.*

Thus, it is advisable to apply the genetic algorithms under consideration to solve optimization problems.

Keywords: *genetic algorithm, optimization problem, traveling salesman problem.*

Одержано редакцією 18.07.2019 р.
Прийнято до публікації 09.10.2019 р.

УДК 519.6

DOI 10.31651/2076-5886-2019-2-43-58

ГОЛОВНЯ Борис Петрович
доктор технических наук, доцент кафедры прикладной математики и информатики Черкасского национального университета имени Богдана Хмельницкого
e-mail: bpgolovnya@gmail.com
ORCID 0000-0002-9242-3937

К ВОПРОСУ О ПРЕПОДАВАНИИ МЕТОДА СОПРЯЖЕННЫХ ГРАДИЕНТОВ

Метод сопряженных градиентов является лучшим из известных итерационных методов решения систем линейных уравнения с симметричной матрицей. На его основе разработано много высокоскоростных методов решения произвольных систем алгебраических уравнений. В то же время, традиционное объяснение принципов работы этого метода очень сложно. Как показывает практика, студенты плохо понимают его. В работе предложено интуитивно понятное объяснение принципов работы метода сопряженных градиентов.

***Ключевые слова*:** *системы линейных алгебраических уравнений, итерационные методы решения СЛАУ, метод сопряженных градиентов.*

Постановка задачи

Метод сопряженных градиентов является одним из самых быстросходящихся итерационных методов для решения систем линейных алгебраических уравнений с симметричной матрицей. Но, при всей простоте программной реализации метода, его математические основы достаточно сложны (см. например [1],[2],[3]). Очень часто при его пояснении используются такие понятия, как проекционные методы, ортогонализация по Арнольди и Ланцошу, скорейший спуск, пространства Крылова, сопряженные направления и многое другое. Все это делает метод непонятным на интуитивном уровне. По этой причине метод сопряженных градиентов очень непросто объяснять студентам. В работе приведено несколько отличающееся от традиционных обоснование метода. Здесь используются только понятие ортогональности, метод Грамма-Шмидта и понятие А-скалярного произведения. Понятие проекции используется только для улучшения уже построенного и понятного метода. Такой подход оказывается интуитивно понятным и гораздо более простым методически чем

традиционный подход. В итоге в данном описании метол усваивается студентами гораздо легче.

Цель статьи

Задача работы – предложить простое и понятное обоснование метода сопряженных градиентов

Методы решения

В основе метода сопряженных градиентов лежит понятие ортогонализации. Рассмотрим его подробней

Мы имеем векторное пространство со скалярным произведением.

Согласно определению в *n*-мерном линейном пространстве задано скалярное умножение векторов, если любой паре векторов *a* и *b* из этого пространства поставлено в соответствие действительное число, обозначаемое (*a*, *b*) и называемое скалярным произведением, причем выполняются следующие условия:

1. $(a,b)=(b,a)$.
2. $(a+b,c)=(a,c)+(b,c)$.
3. $(\alpha a,b)=\alpha(a,b)$.
4. $a\neq 0\rightarrow(a,a)>0$.

Методика нахождения этого числа, т.е. скалярного произведения, в определении не оговаривается. Общеизвестный вариант скалярного произведения - $(a,b)=|a||b|\cos\alpha$. Здесь α - угол между векторами a и b. Нетрудно убедиться, что оно удовлетворяет всем вышеперечисленным требованиям. По определению вектора считаются ортогональными, если их скалярное произведение равно 0.

Имеем задачу $Ax=b$. Разложим матрицу A на множители $A=C\times D$ так, чтобы одна из матриц легко обращалась, а система с другой матрицей легко решалась. Пусть матрица C легко обращается, а система с матрицей D легко решается. Перепишем систему как $(C\times D)x=C(Dx)=b$ и введем переменную $y=Dx$. Тогда исходная система распадается на две легко решаемых системы $Cy=b$ и $Dx=y$. Какая именно из матриц легко обращается роли не играет.

Классическим примером легко обращающихся матриц служат унитарные матрицы, т.е. матрицы вида $A\times A^{*}=A^{*}\times A=E$. Если элементами матрицы являются действительные числа, то выражение упрощается $A\times A^{T}=A^{T}\times A=E$, а матрица называется ортогональной. Рассмотрим, что значит требование ортогональности. Пусть матрица имеет порядок n и состоит из n строк. Мы трактуем каждую строку как n-мерный вектор. Тогда требование ортогональности выглядит как

$$A\times A^{T}=\begin{pmatrix}a_1\\a_2\\\dots\\a_n\end{pmatrix}\begin{pmatrix}a_1\\a_2\\\dots\\a_n\end{pmatrix}^{T}=\begin{pmatrix}a_1\\a_2\\\dots\\a_n\end{pmatrix}\begin{pmatrix}a_1^{T} & a_2^{T} & \dots & a_n^{T}\end{pmatrix}=\left(a_i a_j^{T}\right)=E.$$

Отсюда мы сразу имеем $\left(a_i a_j^{T}\right)=\begin{cases}1 & \text{i}=\text{j}\\0 & \text{i}\neq\text{j}\end{cases}$. Но тогда можно сказать, что строки матрицы представляют собой ортонормированную систему векторов. Транспонировав это выражение, получаем, что столбцы матрицы также представляют собой ортонормированную систему.

Таким образом, одна из стоящих задач – преобразовать систему векторов, образующих столбцы (строки) матрицы в ортонормированную систему. Наиболее известный метод решения этой задачи это процесс Грама-Шмидта.

Процесс Грамма-Шмидта

Имеем векторное пространство, натянутое на векторы $a_1, a_2, ..., a_n$. Построим в нем ортогональный базис $u_1, u_2, ..., u_n$. Процедура выглядит следующим образом. В качестве первого вектора нового базиса выбираем первый вектор исходного базиса.

$$u_1 = a_1.$$

Основная идея построения всех последующих векторов состоит в следующем. Векторы $a_1 = u_1$ и a_2 – элементы базиса и не могут быть параллельны. Поэтому на них можно натянуть плоскость. Любой вектор этой плоскости можно выразить как

$$u_2 = \alpha u_1 + \beta a_2. \quad (1)$$

Будем искать в этой плоскости вектор ортогональный u_1. Очевидно, что у этого вектора множитель β не может равняться нулю, т.к. вы этом случае мы получим параллельный u_1 вектор. В то же время нас не интересует длина ортогонального вектора, а только его направление. Поэтому общее выражение для этого вектора делим на неравное нулю число β и ищем его в виде (2).

$$u_2 = a_2 - \lambda_{2,1} u_1. \quad (2)$$

Минус перед $\lambda_{2,1}$ взят для упрощения дальнейших формул. Потребуем ортогональности этого вектора вектору u_1. Для этого умножим (2) скалярно на u_1 и приравняем результат нулю. Имеем $(u_2 u_1) = (a_2 u_1) - \lambda_{2,1}(u_1 u_1) = 0$. Отсюда $\lambda_{2,1} = (a_2 u_1)/(u_1 u_1)$. Понятно, что плоскость, натянутая на векторы u_1 и u_2, совпадает с плоскостью, натянутой на a_1 и a_2

Построим третий вектор. Векторы a_1, a_2 и a_3 принадлежат к базису, поэтому вектор a_3 не лежит в плоскости, натянутой на векторы u_1 и u_2. В связи с этим вектор u_3 строим в виде $u_3 = a_3 - \lambda_{3,2} u_2 - \lambda_{3,1} u_1$. Из требования ортогональности вектора u_3 векторам u_1 и u_2 получаем $\lambda_{3,1} = (a_3 u_1)/(u_1 u_1)$, $\lambda_{3,2} = (a_3 u_2)/(u_2 u_2)$

Окончательное описание ортогонализации по Граму-Шмидту выглядит следующим образом

$$\begin{aligned} u_1 &= a_1, \\ u_2 &= a_2 - \lambda_{2,1} u_1, \\ &\cdots\cdots\cdots\cdots \\ u_k &= a_k - \sum_{j=1}^{k-1} \lambda_{k,j} u_j. \end{aligned} \quad (3)$$

Проводя ортогонализацию по методике Грама-Шмидта умножим (3) скалярно на u_i ($i < k$). Получаем

$$(u_k u_i) = (a_k u_i) - \sum_{j=1}^{k-1} \lambda_{k,j} (u_j u_i). \quad (4)$$

Из требования ортогональности $(u_k u_i) = 0$ при $i < k$ получаем

$$\lambda_{k,i} = \frac{(a_k u_i)}{(u_i u_i)}. \quad (5)$$

Элементарная модификация алгоритма позволяет получить ортонормированный базис. Для этого мы будем считать, что выражение (3) рассчитывает не окончательный, а промежуточный вариант вектора u_k, т.е. операторы (3) и (4) перепишем как

$$u_{k-temp} = a_k - \sum_{j=1}^{k-1} \lambda_{k,j} u_j.$$

$$(u_{k-temp} u_i) = (a_k u_i) - \sum_{j=1}^{k-1} \lambda_{k,j} (u_j u_i).$$

При расчете $\lambda_{k,i}$ считаем, что все полученные ранее векторы уже нормированы, поэтому в знаменателе стоит единица

$$\lambda_{k,i} = (a_k u_i).$$

В заключение находим новый базисный вектор как

$$u_k = u_{k-temp} / \|u_{k-temp}\|$$

Задача построения А-ортогонального базиса

Очень удобные методы решения систем линейных алгебраических уравнений можно построить используя понятие А-ортогональности.

Пусть мы имеем скалярное произведение и симметричную, положительно определенную матрицу A. Напомним, что согласно одному из эквивалентных определений матрица называется положительно определенной если для любого ненулевого вектора a $(Aa,a) > 0$. Введем операцию над векторами вида $(a,b)_A = (Aa,b)$. Очевидно, что эта операция удовлетворяет всем требованиям, предъявляемым к скалярному произведению и может быть названа как А-скалярное произведение. Теперь можно ввести понятие А-нормы $\|x\|_A = \sqrt{(x,x)_A}$. Как известно ортогональными векторами называются векторы с нулевым скалярным произведением. Отсюда мы можем ввести понятие А-ортогональности. Часто А-ортогональные векторы называют сопряженными.

Рассмотрим, что это нам дает. Решается система уравнений $Ax = b$ с симметричной, положительно определенной матрицей A. В пространстве R^n, т.е. в пространстве, которому принадлежит решение системы, построен А-ортогональный базис $R^n = span\{u_1, u_2, ..., u_n\}$. Разложим решение по этому базису $x = \sum c_i u_i$ и подставим разложение в систему $\sum c_i A u_i = b$. Умножим результат скалярно на u_k - $\sum c_i (Au_i, u_k) = (b, u_k)$. Из требования А-ортогональности имеем $c_k (Au_k, u_k) = (b, u_k)$, где

$$c_k = \frac{(b,u_k)}{(Au_k,u_k)}. \quad (6)$$

Ситуацию можно интерпретировать так. После всех этих преобразований система уравнений приобрела диагональный вид

$$\begin{pmatrix} (Au_1,u_1) & & & \\ & (Au_2,u_2) & & \\ & & \dots & \\ & & & (Au_n,u_n) \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \dots \\ c_n \end{pmatrix} = \begin{pmatrix} (b,u_1) \\ (b,u_2) \\ \dots \\ (b,u_n) \end{pmatrix}.$$

Мы можем находить векторы базиса последовательно, компонента решения, связанная с очередным найденным вектором не влияет на последующие выкладки. Само решение исходной системы мы также находим в процессе расчетов как

$$x_{k+1} = x_k + c_k u_k, \tag{7}$$

где x_{k+1} – очередное приближение к решению. Замечательно, что хотя процесс очень похож на итерационный, он приводит к точному решению ровно за n шагов.

A-ортогонализация на основе процесса Грама-Шмидта

Построим алгоритм перехода от базиса $(r_1, r_2, ..., r_n)$ к A-ортогональному базису $(u_1, u_2, ..., u_n)$. Процедура ортогонализации по Граму-Шмидту выглядит следующим образом

$$\begin{gathered} u_1 = r_1, \\ u_2 = r_2 - \lambda_{1,2} u_1, \\ u_3 = r_3 - \lambda_{2,3} u_2 - \lambda_{1,3} u_1, \\ u_4 = r_4 - \lambda_{3,4} u_3 - \lambda_{2,4} u_2 - \lambda_{1,4} u_1, \\ \ldots\ldots\ldots\ldots\ldots \\ u_j = r_j - \sum_{i=1}^{j-1} \lambda_{i,j} u_i. \end{gathered} \tag{8}$$

Умножим (8) A-скалярно на u_k $k < j$. Получаем $(Au_j, u_k) = (Ar_j, u_k) - \sum_{i=1}^{j-1} \lambda_{k,j} (Au_i, u_k)$. Требование A-ортогональности значит, что $(Au_i, u_j) = (Au_j, u_i) = 0$ при $j \neq i$, откуда

$$(Ar_j, u_k) - \sum_{i=1}^{j-1} \lambda_{i,j} (Au_i, u_k) = 0$$

Но $(Au_i, u_k) = 0$ при $i \neq k$. В итоге в сумме остается единственное слагаемое

$$(Ar_j, u_i) - \lambda_{i,j} (Au_i, u_i) = 0. \tag{9}$$

Коэффициент $\lambda_{i,j}$ находим из выражения (9)

$$\lambda_{i,j} = \frac{(Ar_j, u_i)}{(Au_i, u_i)} = \frac{(Au_i, r_j)}{(Au_i, u_i)}.$$

Последний переход возможен, т.к. матрица A по условию симметрична.

Мы построили процедуру A-ортогоналиции на основе классического процесса Грамма-Шмидта

Оформим эти действия в виде алгоритма на псевдокоде. Представим его в двух вариантах. Первый вариант полностью соответствует вышеописанным действиям

1. $u_1 = r_1$
2. *for i=2:n*
3. $u_i = r_i$
4. *for j=1:i-1*
5. $\lambda_{i,j} = (Au_i, r_j) / (Au_i, u_i)$
6. $u_i = u_i - \lambda_{i,j} u_j.$

7. end
8. end

Второй вариант строит A-ортонормированную систему векторов

1. $u_1 = r_1 / \|r_1\|$
2. *for i=2:n*
3. $u_{temp} = r_i$
4. *for j=1:i-1*
5. $\lambda_{i,j} = (Au_i, r_j)$
6. $u_{temp} = u_{temp} - \lambda_{i,j} u_j.$
7. end
8. $u_i = u_{temp} / \sqrt{(Au_{temp}, u_{temp})}$
9. end

Простой метод решения систем уравнений основанный на A-ортогонализации.

Теперь мы можем построить простой метод решения систем уравнений основанный на A-ортогонализации. Собственно метод был описан немного выше. Нам только нужно дополнить его процедурой построения A-ортогонального базиса. Здесь, как и ранее, $(r_1, r_2, ..., r_n)$ - уже имеющийся базис, $(u_1, u_2, ..., u_n)$ - A-ортогональный базис. Алгоритм используется в виде без нормировки. Его можно оформить следующим образом

1. $x_1 = 0$
2. $u_1 = r_1$
3. *for i=1:n-1*
4. $c_i = (b, u_i) / (Au_i, u_i)$
5. $x_{i+1} = x_i + c_i u_i$
6. $u_i = r_i$
7. *for j=1:i-1*
8. $\lambda_{i,j} = (Au_i, r_j) / (Au_i, u_i)$
9. $u_i = u_i - \lambda_{i,j} u_j$
10. end
11. end

Алгоритм вполне работоспособен. Недостатки его хорошо заметны. Он требует очень большой оперативной памяти. В частности, мы должны помнить исходный базис. Для этого необходима память равная полной матрице исходной задачи. Кроме того, память необходима для запоминания A-ортогонального базиса. А это еще один такой же объем памяти. Эти два факта сводят на нет все достоинства метода.

Метод сопряженных градиентов по идеологии полностью совпадет с описанным методом. Но в нем, путем удачного выбора исходного базиса, удалось резко сократить потребности в оперативной памяти. Чтобы разобраться в способе построения этого базиса нам понадобится понятие проектирования (проекции).

Проектирование

Проектирование очень широко применяется в численных методах. Например. Приближенные соотношения для производных получаются отбрасыванием хвоста ряда Тейлора. Но использование только части базиса в разложении произвольного вектора, функции и т.д. и есть проектирование. Отсюда следует, что все конечно-разностные методы, а заодно и методы решения задачи Коши по смыслу являются проекционными. Конечно-элементные методы сразу называют себя проекционными. Очевидно, что все методы численного интегрирования попадают сюда же. Т.е. прием упрощения задачи путем отбрасывания части ее компонент очень популярен в вычислительной математике. Другое дело, что при решении систем уравнений мы будем использовать его несколько иначе.

Что такое проекция

Рассмотрим проекцию вектора в двумерном случае (рис. 1). В этом случае вектор может проектироваться только на прямую, т.е. на направление. Саму проекцию можно рассматривать как тень от вектора, отбрасываемую на это направление. Естественно, что в этом случае нам надо знать направление на источник света. Например. Мы проектируем вектор $\overline{ab}$ на направление $\overline{acd}$. В случае если источник света находится на линии $\overline{cb}$, то результатом будет вектор $\overline{ac}$, если же на направлении $\overline{db}$ - то вектор $\overline{ad}$. Для нахождения этой проекции достаточно разложить исходный вектор по базису, состоящему из вектора – направления проекции и вектора – направления источника света. Отбросив компоненту, связанную с источником света, получаем решение. В дальнейшем вместо выражения «вектор направления на источник» будем использовать выражение «параллельно вектору». Естественно, что этот вектор и задает направление на источник. Для простоты выкладок мы будем описывать направление, параллельно которому идет проектирование, ортогональной этому направлению прямой. Тогда отбрасывание компоненты сводится к скалярному умножению разложения на вектор, параллельный этой прямой.

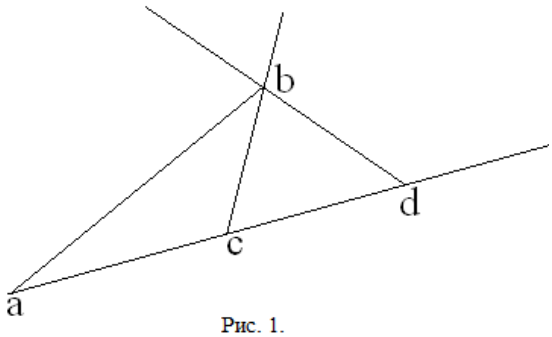

Рис. 1.

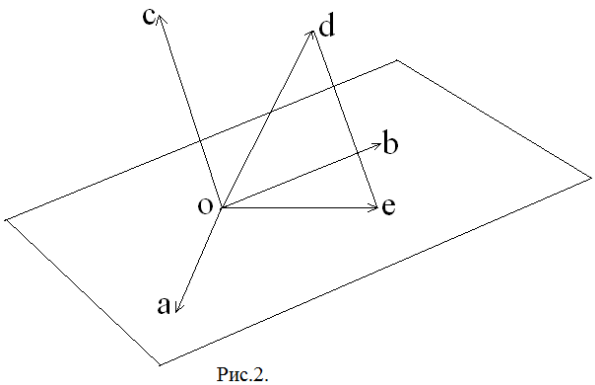

Рис.2.

Расчет выглядит следующим образом. Построим проекцию вектора u на направление заданное вектором v_1, параллельно вектору w, причем вектор v_2 ортогонален вектору w. Разложим вектор u векторам v_1 и w - $u = c_1 v_1 + c_2 w$. Вектор $c_2 w = u - c_1 v_1$ ортогонален v_2. Запишем это условие – $c_2 w = (u v_2) - c_1 (v_1 v_2)$. Отсюда $c_1 = \frac{(u v_2)}{(v_1 v_2)}$. Искомая проекция имеет вид $P(u) = \frac{(u v_2)}{(v_1 v_2)} v_1$. Наибольшее распространение получили ортогональные проекции, т.е. проекции, когда вектор, параллельно которому строится проекция, ортогонален

направлению проекции. В этом случае можно выбрать v_2=v_1 и формула упрощается $P(u)=\frac{(uv_1)}{(v_1v_1)}v_1$. Если вектор v_1 нормирован, то имеем $P(u)=(uv_1)v_1$.

И еще. Отброшенную компоненту разложения, т.е. вектор c_2w, вполне можно рассматривать как погрешность замены исходного вектора его проекцией. Будем измерять эту погрешность длиной отброшенного вектора. Очевидно, что расстояние от точки до прямой будет кратчайшим, если измерять его по перпендикуляру, опущенному из точки на прямую. Отсюда имеем, что в случае ортогональной проекции эта погрешность минимизируется.

Теперь рассмотрим проекцию вектора на плоскость параллельно заданному вектору. Мы будем проектировать вектор $v=\overline{od}$ параллельно направлению $w=\overline{oc}$. Плоскость на которую строится проекция задается двумя векторами - $v_1=\overline{oa}$ и $v_2=\overline{ob}$. Разложим вектор v по базису из векторов v_1, v_2 и w - $v=c_1v_1+c_2v_2+c_3w$. Как и в прошлый раз, будем задавать направление проекции ортогональным этому направлению подмножеством трехмерного пространства. Ортогональность мы будем понимать следующим образом – любой вектор, принадлежащий этому подмножеству ортогонален вектору w. Очевидно, что в нашем случае это будет плоскость. Будем считать, что эта плоскость натянута на векторы w_1 и w_2, причем $(ww_1)=(ww_2)=0$. Тогда, умножая соотношение $c_3w=v-c_1v_1-c_2v_2$ на векторы w_1 и w_2, получим систему уравнений для расчета коэффициентов c_1 и c_2

$$\begin{cases} c_1(v_1,w_1)+c_2(v_2,w_1)=(v,w_1), \\ c_1(v_1,w_2)+c_2(v_2,w_2)=(v,w_2). \end{cases}$$

Итоговая проекция выглядит как $P(v)=c_1v_1+c_2v_2$.

Несложными преобразованиями получаем решение

$$\begin{pmatrix} (v_1,w_1) & (v_2,w_1) \\ (v_1,w_2) & (v_2,w_2) \end{pmatrix}\begin{pmatrix} c_1 \\ c_2 \end{pmatrix}=\begin{pmatrix} (v,w_1) \\ (v,w_2) \end{pmatrix}$$

$$\begin{pmatrix} c_1 \\ c_2 \end{pmatrix}=\begin{pmatrix} (v_1,w_1) & (v_2,w_1) \\ (v_1,w_2) & (v_2,w_2) \end{pmatrix}^{-1}\begin{pmatrix} (v,w_1) \\ (v,w_2) \end{pmatrix}=\begin{pmatrix} (v_1,w_1) & (v_2,w_1) \\ (v_1,w_2) & (v_2,w_2) \end{pmatrix}^{-1}\left(\begin{pmatrix} w_1 \\ w_2 \end{pmatrix},v\right)$$

$$P(v)=c_1v_1+c_2v_2=\begin{pmatrix} v_1 & v_2 \end{pmatrix}\begin{pmatrix} c_1 \\ c_2 \end{pmatrix}=\begin{pmatrix} v_1 & v_2 \end{pmatrix}\begin{pmatrix} (v_1,w_1) & (v_2,w_1) \\ (v_1,w_2) & (v_2,w_2) \end{pmatrix}^{-1}\left(\begin{pmatrix} w_1 \\ w_2 \end{pmatrix},v\right)$$

Решение заметно упрощается, если проектирование производится ортогонально плоскости и векторы v_1 и v_2 ортогональны. В этом случае можно выбрать $w_1=v_1$ и $w_2=v_2$. Тогда $c_1=\frac{(v,v_1)}{(v_1,v_1)}$ и $c_2=\frac{(v,v_2)}{(v_2,v_2)}$. Если же векторы ортонормированны, то соотношения получаются следующими $c_1=(v,v_1)$ и $c_2=(v,v_2)$, или $\begin{pmatrix} c_1 \\ c_2 \end{pmatrix}=\begin{pmatrix} (v,v_1) \\ (v,v_2) \end{pmatrix}=\left(\begin{pmatrix} v_1 \\ v_2 \end{pmatrix},v\right)$, а проекция вычисляется как

$$P(v) = c_1 v_1 + c_2 v_2 = \begin{pmatrix} v_1 & v_2 \end{pmatrix} \begin{pmatrix} (v, v_1) \\ (v, v_2) \end{pmatrix} = \begin{pmatrix} v_1 & v_2 \end{pmatrix} \left(\begin{pmatrix} v_1 \\ v_2 \end{pmatrix}, v \right).$$

Решение может трактоваться следующим образом. Сначала мы находим коэффициенты разложения исходного вектора по базисным векторам v_1 и v_2, а затее умножаем эти коэффициенты на базисные вектора и находим проекцию.

При переходе к произвольному *n*-мерному пространству технология решения задачи не меняется. Нужно только окончательно определить, что значит одно подпространство ортогонально другому. Пусть $(u_1, u_2, ..., u_n)$ ортогональный базис в пространстве R^n. Выделим в базисе $(u_1, u_2, ..., u_n)$ две непересекающихся части $(u_1, u_2, ..., u_k)$ и $(u_{k+1}, u_{k+2}, ..., u_m) = (v_1, v_2, ..., v_l)$, $m = r + l \le n$. Можно сказать, что мы выделили в исходном пространстве R^n два подпространства $R^k = span\{u_1, u_2, ..., u_k\}$ и $R^l = span\{v_1, v_2, ..., v_l\}$. Здесь $span\{u_1, u_2, ..., u_k\}$ - линейная оболочка векторов $u_1, u_2, ..., u_k$. Так как каждый вектор базиса $(u_1, u_2, ..., u_k)$ ортогонален каждому вектору базиса $(v_1, v_2, ..., v_l)$, то любой вектор подпространства $R^k = span\{u_1, u_2, ..., u_k\}$ ортогонален любому вектору подпространства $R^l = span\{v_1, v_2, ..., v_l\}$. В этом случае мы считаем подпространства ортогональными. Определение несколько отличается от бытового понимания ортогональности. В частности, в трехмерном пространстве не может быть двух ортогональных плоскостей. В самом деле, если плоскости пересекаются, то на обеих плоскостях можно найти совпадающие векторы, лежащие на их пересечении.

Проектирование систем уравнений

Рассмотрим проектирование системы уравнений в пространстве R^n на некое подпространство $K = R^k \subset R^n$. Под проекцией будет пониматься решение следующей задачи. Решается система уравнений $A\bar{x} = b$, где A – невырожденная матрица $n \times n$. Пусть x проекция точного решения $\bar{x}$ этой системы на подпространство K. Требуется построить систему, решением которой будет x. Подпространство, на которое производится проекция, и подпространство, ортогонально которому производится проекция, заданы.

Пусть $(v_1 \quad v_2 \quad ... \quad v_k)$ - базис в K. Если x искомая проекция, то она принадлежит K и может быть разложена по базису $(v_1 \quad v_2 \quad ... \quad v_k)$ -

$$x = \sum_{i=1}^{k} x_i v_i = \begin{pmatrix} v_1 & v_2 & ... & v_k \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ ... \\ x_k \end{pmatrix}.$$

Здесь x_i – коэффициенты разложения, $(v_1 \quad v_2 \quad ... \quad v_k)$ - матрица, состоящая из векторов-столбцов базиса. Подставим это выражение в исходную систему. Обозначая произведение матрицы A на $(v_1 \quad v_2 \quad ... \quad v_k)$ как $A_1 = (A(v_1 \quad v_2 \quad ... \quad v_k))$, получаем

$$A\left(\begin{pmatrix} v_1 & v_2 & \dots & v_k \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_k \end{pmatrix}\right) = \left(A\begin{pmatrix} v_1 & v_2 & \dots & v_k \end{pmatrix}\right)\begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_k \end{pmatrix} = A_1\begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_k \end{pmatrix} = b\,.$$

В итоге мы перебрались в подпространство, но получили матрицу размерности $k \times n$. Полученную задачу решать нельзя. Причины очевидны проектирование не только переход к новому базису, а и отбрасывание «лишних» деталей. Способов отбрасывания существует много. Для более детально понимания, что надо делать с системой дальше, рассмотрим простой пример.

Решается система с двумя неизвестными

$$\{\, Ax = b\} = \begin{cases} a_1 x = b_1 \\ a_2 x = b_2 \end{cases}.$$

Очевидно, что задача имеет точное решение если и только если вектор b коллинеарен вектору a, или вектор b находится в подпространстве, натянутом на вектор a. В общем случае задача не имеет решения. Будем искать его по методу наименьших квадратов. Т.е. будем искать такое значение x чтобы выражение

$$(b_1 - a_1 x)^2 + (b_2 - a_2 x)^2 = (b - ax)^T (b - ax) = b^T b - 2a^T bx + a^T ax^2$$

приобретало минимальное значение. Дифференцируя по x и приравнивая нулю производную, получаем

$$a^T b - a^T ax = a^T (b - ax) = 0\,, \tag{10}$$

откуда следует $x = \dfrac{a^T b}{a^T a} = \dfrac{(a,b)}{(a,a)}$. Решение совпадает со случаем проектирования вектора b на вектор a, т.е. мы нашли ближайшую к точке b точку на направлении, заданном вектором a. Можно сказать, что мы нашли ближайшую к точке b точку в подпространстве, заданном вектором a. Отметим некоторые особенности полученного решения. Во-первых, мы минимизировали норму $\|b - ax\|$. Во-вторых, вектор невязки $b - ax$ в этом случае получается ортогональным подпространству, заданному вектором a коэффициентов исходной системы

Вернемся к исходной задаче. Мы проектировали систему на подпространство L с базисом $(v_1 \;\; v_2 \;\; \dots \;\; v_k)$. Согласно только что сказанному, для получения ортогонально проекции нам достаточно умножить слева соотношение (10) на транспонированную матрицу, составленную из базисных векторов.

$$\begin{pmatrix} v_1 \\ v_2 \\ \dots \\ v_k \end{pmatrix}\left(b - A\begin{pmatrix} v_1 & v_2 & \dots & v_k \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_k \end{pmatrix}\right) = 0\,. \tag{11}$$

Чтобы избежать громоздкости записи обозначим матрицу из базисных векторов как $V = \begin{pmatrix} v_1 & v_2 & \dots & v_k \end{pmatrix}$ и вектор коэффициентов разложения проекции x по этому базису как $x_V = \begin{pmatrix} x_1 & x_2 & \dots & x_k \end{pmatrix}^T$ и перепишем выражение (11) как

$$\left\{ V^T \left(b - AVx_V \right) = 0 \right\} = \left\{ V^T AVx_V = V^T b \right\}. \tag{12}$$

Для повышения общности можно потребовать ортогональности невязки не подпространству K, а какому-либо другому подпространству L такой же размерности. Тогда соотношение (12) перепишется как

$$\left\{ W^T \left(b - AVx_V \right) = 0 \right\} = \left\{ W^T AVx_V = W^T b \right\}, \tag{13}$$

где W – матрица, составленная из векторов-столбцов подпространства L. Сокращать W^T нельзя, т.к. сокращение эквивалентно умножению на обратную матрицу. Но матрица W^T не квадратная и обратной не имеет.

Напомним, что вектор проекции x здесь разложен по базису V, т.е. $x = Vx_V$, откуда $b - AVx_V = b - Ax = r$ - невязка исходной системы уравнений при подстановке в нее вектора проекции. Отсюда выражение (13) можно переписать как $W^T r = 0$.

С точки зрения итерационного расчета задачу можно сформулировать следующим образом. Решается система уравнений $A\bar{x} = b$. Имеем некоторое приближение x^k к точному решению $\bar{x}$. Требуется дополнить это приближение до проекции, т.е. найти такую поправку $\delta^k \in K$ к приближению, чтобы вектор невязки $b - A\left(x^k + \delta^k \right)$ стал ортогонален подпространству L. Тогда можно записать

$$W^T \left(b - A\left(x^k + \delta^k \right) \right) = W^T \left(b - AV\left(x_V^k + \delta_V^k \right) \right) = W^T \left(b - AVx_V^k \right) - W^T AV\delta_V^k = 0.$$

и

$$\left\{ W^T AV\delta_V^k = W^T \left(b - AVx_V^k \right) \right\} = \left\{ \delta_V^k = \left(W^T AV \right)^{-1} W^T \left(b - AVx_V^k \right) \right\} = \left\{ \delta_V^k = \left(W^T AV \right)^{-1} W^T r^k \right\}.$$

Окончательно получаем

$$\delta^k = V\delta_V^k = V\left(W^T AV \right)^{-1} W^T r^k$$

и

$$\mathrm{x}^{\mathrm{k+1}} = \mathrm{x}^{\mathrm{k}} + \delta^k = x^k + V\left(W^T AV \right)^{-1} W^T r^k.$$

Здесь r^k – вектор невязки системы при подстановке в нее x^k. Отметим, что так как матрицы W^T и V не квадратные и обратных не имеют, то выражение $\left(W^T AV \right)^{-1} = V^{-1} A^{-1} \left(W^T \right)^{-1}$ неверно. В то же время, если удачно выбрать матрицы W и V, то вычисление $\left(W^T AV \right)^{-1}$ может оказаться очень несложным.

В завершение отметим несколько полезных соотношений. Во первых

$$r^{k+1} = b - A\left(x^k + \delta^k \right) = b - Ax^k - A\delta^k = r^k - A\delta^k.$$

Отсюда получаем

$$r^{k+1} - r^k = -A\delta^k = -A\left(x^{k+1} - x^k\right)$$

и

$$W^T\left(r^k - A\delta^k\right) = 0\,.$$

Метод сопряженных градиентов

Метод сопряженных градиентов вполне соответствует предложенному методу. Но если в этом методе в качестве исходного использовался некий уже имеющийся базис, то в методе сопряженных градиентов используется базис из векторов-невязок системы, рассчитываемых на каждом шаге. Рассмотрим, что это нам дает. Для этого вспомним некоторые свойства ортогонального проектирования систем уравнений.

В пространстве решается система уравнений $A\bar{x} = b$. Задано некоторое подпространство $K^k = R^k \subset R^n$. Имеем некоторое приближение $x_k \in K^k$ к точному решению $\bar{x}$. Дополнить это приближение до проекции, значит найти такую поправку $\delta_k \in K^k$ к приближению, чтобы вектор невязки $r^{k+1} = b - A\left(x^k + \delta^k\right) = b - Ax^{k+1}$ стал ортогонален подпространству K^k. Эта ортогональность эквивалентна минимальности нормы невязки.

В то же время $r_k \in K^k$ и из ортогональности $r_{k+1} \in K^{k+1}$ подпространству K^k следует, что $r_{k+1} \perp r_k$. Рассуждая по индукции имеем $r_1 \perp r_2 \perp r_3 \perp \ldots \perp r_n$, т.е. невязки образуют ортогональный базис в R^n, причем этот базис находится по ходу вычислений. Но тогда этот базис мы можем преобразовывать в А-ортогональный базис, который позволяет нам находить новые поправки к вектору неизвестных. Алгоритм замкнулся.

Учитывая, что

$$r_{j+1} = b - A\left(x_j + \delta_j\right) = b - Ax_j - A\delta_j = r_j - A\delta_j\,.$$

его можно записать следующим образом.

1. x_1 - *выбрать*
2. $r_1 = b - Ax_1$
3. $u_1 = r_1$
4. *For* j=1:n *do*
4. $c_j = \left(b, u_j\right) / \left(Au_j, u_j\right)$
6. $x_{j+1} = x_j + c_j u_j$
7. $r_{j+1} = r_j - c_j Au_j$
8. *For* i=1,j *do*
9. $\lambda_{i,j+1} = \left(Au_i, r_{j+1}\right) / \left(Au_i, u_i\right)$
10. $u_{j+1} = r_{j+1} - \sum_{i=1}^{j} \lambda_{i,j+1} u_i.$
11. *End*

Ортогональность базиса $r_1, r_2, \ldots, r_n$ и методика его построения вносит в процедуру расчета заметные упрощения. В самом деле. Рассмотрим сумму в 9-ой строке алгоритма. Для этого оператор в 7-ой строке запишем в виде $r_{i+1} = r_i - c_i Au_i$ и умножим скалярно на r_{j+1}. Получаем

$$c_i\left(Au_i, r_{j+1}\right) = \left(r_i, r_{j+1}\right) - \left(r_{i+1}, r_{j+1}\right) \tag{13}$$

В силу ортогональности r_k правая часть (13) не равна нулю только если $\left(r_i, r_{j+1}\right) \neq 0$ или $\left(r_{i+1}, r_{j+1}\right) \neq 0$, т.е. $i = j+1$ или $i+1 = j+1$. Но из оператора цикла в 8-ой строке следует, что $i \leq j$. Но тогда получаем, что в сумме в 9-ой строке осталось единственное слагаемое. В результате оператор 10-ой строки приобретает вид

$$u_{j+1} = r_{j+1} - \lambda_{j,j+1} u_j = u_{j+1} = r_{j+1} - \frac{\left(Au_j, r_{j+1}\right)}{\left(Au_j, u_j\right)} u_j .$$

Теперь алгоритм метода можно переписать как

1. x_1 - *выбрать*
2. $r_1 = b - Ax_1$
3. $u_1 = r_1$
4. *For* j=1:n *do*
4. $c_j = \left(b, u_j\right) / \left(Au_j, u_j\right)$
6. $x_{j+1} = x_j + c_j u_j$
7. $r_{j+1} = r_j - c_j Au_j$
8. $\beta_j = \left(Au_j, r_{j+1}\right) / \left(Au_j, u_j\right)$
9. $u_{j+1} = r_{j+1} - \beta_j u_j$
10. *End*

Чтобы получить традиционную форму алгоритма сделаем небольшие преобразования. Во первых. Умножив равенство $r_{j+1} = r_j - c_j Au_j$ на u_j получаем $c_j\left(Au_j, u_j\right) = \left(r_j, u_j\right) - \left(r_{j+1}, u_j\right)$. Но из смысла процесса Грамма-Шмидта следует, что $\forall j \ span\{r_1, r_2, ..., r_k\} = span\{u_1, u_2, ..., u_k\}$, а из ортогональности базиса $r_1, r_2, ..., r_{j+1}$ следует, что $r_{j+1} \perp span\{r_1, r_2, ..., r_j\}$ откуда $r_{j+1} \perp span\{u_1, u_2, ..., u_j\}$. Следовательно $r_{j+1} \perp u_j$ и $c_j = \left(r_j, u_j\right) / \left(Au_j, u_j\right)$. В то же время $u_j = r_j - \beta_{j-1} u_{j-1}$, откуда $\left(r_j, u_j\right) = \left(r_j, r_j\right) - \beta_{j-1}\left(r_j, u_{k-1}\right)$. Так как r_j ортогонально u_{j-1}, то $\left(r_j, u_{j-1}\right) = 0$ и $\left(r_j, u_j\right) = \left(r_j, r_j\right)$. В результате получаем стандартное выражение $c_j = \left(r_j, r_j\right) / \left(Au_j, u_j\right)$.

Во вторых. Из $r_{j+1} = r_j - c_j Au_j$ имеем $Au_j = \left(r_j - r_{j+1}\right) / c_j$ откуда $\left(r_{j+1}, Au_j\right) = \left(r_{j+1}, r_j - r_{j+1}\right) / c_j = -\left(r_{j+1}, r_{j+1}\right) / c_j$. Сопоставляя это выражение с выражением для c_j и формулой $\beta_j = \left(r_{j+1}, Au_j\right) / \left(Au_j, u_j\right)$ получаем $\beta_j = -\left(r_{j+1}, r_{j+1}\right) / \left(r_j, r_j\right)$.

Окончательно

1. x_1 - *выбрать*
2. $r_1 = b - Ax_1$
3. $u_1 = r_1$
4. *For* j=1:n *do*
4. $c_j = \left(r_j, r_j\right) / \left(Au_j, u_j\right)$
6. $x_{j+1} = x_j + c_j u_j$

7. $r_{j+1} = r_j - c_j A u_j$
8. $\beta_j = (r_{j+1}, r_{j+1}) / (r_j, r_j)$
9. $u_{j+1} = r_{j+1} + \beta_j u_j$
10. *End*

Отметим, что знак у β заменен на плюс.

Достоинства и недостатки метода сопряженных градиентов.

Метод очень экономичен. Он требует хранения в памяти только трех векторов. При грамотном написании программы на один шаг цикла используется всего одна операция умножения матрицы на вектор, два скалярных умножения векторов и три операции сложения векторов. При этом программу можно организовать так, что матрица явно в вычислениях участвовать не будет, а будет доступна лишь через программу, вычисляющую произведение матрицы на вектор. Этот факт делает метод сопряженных градиентов и родственные ему методы чрезвычайно привлекательным при хранении матриц в любом разреженном формате.

Основной недостаток метода – возможная потеря ортогональности. Что это значит. Как уже говорилось, теоретически метод дает точное решение за n шагов. Но этот результат будет достигаться при вычислениях с бесконечным количеством цифр. Мы же работаем в машинной арифметике, т.е. с ограниченным числом значащих цифр. В связи с этим точное решение достигается за n шагов разве что случайно. Причины состоят в следующем. Мы считаем, что невязки образуют ортогональный базис. Этот факт гарантирует теория метода. В то же время, поиск очередной невязки требует умножения матрицы на вектор и сложения двух векторов. Ошибки плавающей арифметики здесь практически гарантированы. Поэтому мы получим $|(r_{k+1}, r_k)| = \varepsilon > 0$. Эта погрешность остается в расчете навсегда. Более того. Эта погрешность накапливается, т.е. $|(r_{k+2}, r_k)| = \varepsilon_1 > \varepsilon > 0$ и т.д. В результате базис из невязок получается не ортогональным. При больших системах уравнений отклонения от ортогональности могут быть весьма значительными. Можно сказать, что ортогональность нам не нужна, достаточно иметь базис. Это верно, но при выводе соотношений для c_k и β_k теоретическая ортогональность невязок использовалась очень активно. Это значит что используемые выражения для c_k и β_k только приближенны. В результате система уравнений для c_k

$$\begin{pmatrix} (Au_1, u_1) & & & \\ & (Au_2, u_2) & & \\ & & \dots & \\ & & & (Au_n, u_n) \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \dots \\ c_n \end{pmatrix} = \begin{pmatrix} (b, u_1) \\ (b, u_2) \\ \dots \\ (b, u_n) \end{pmatrix}.$$

получается не диагональной, а сами векторы u_k оказываются не А-ортогональными.

Эти эффекты привели к тому, что метод сопряженных градиентов сначала был объявлен неустойчивым и заброшен. И лишь потом пришло понимание, что это очень хороший итерационный метод.

Для преобразования метода в итерационный достаточно заменить оператор цикла со счетчиком на оператор цикла «пока не будет достигнута требуемая точность» или «пока невязка не станет достаточно мала». Окончательная форма метода

1. x_1 - *выбрать*
2. $r_1 = b - Ax_1$
3. $u_1 = r_1$
4. *For j*=1,2,3… *до достижения сходимости do*
4. $c_j = (r_j, r_j)/(Au_j, u_j)$
6. $x_{j+1} = x_j + c_j u_j$
7. $r_{j+1} = r_j - c_j A u_j$
8. $\beta_j = (r_{j+1}, r_{j+1})/(r_j, r_j)$
9. $u_{j+1} = r_{j+1} + \beta_j u_j$
10. *End*

И наконец, для борьбы с накапливающимися ошибками рекомендуется время от времени перезапускать расчет. Это значит, что мы объявляем x_1 равным уже найденному приближению и начинаем расчет с начала.

Метод сопряженных градиентов и по нынешнее время остается самым эффективным итерационным методом решения систем линейных алгебраических уравнений с симметричной матрицей. В основе метода лежит построение А-ортогонального базиса пространства, в котором решается система – ортогонального базиса, состоящего из невязок. Это можно делать на основе любого подхода, в частности на основе рассмотренного процесса Грама-Шмидта, и алгоритм будет работать. Но на основе базиса составленного из невязок получается меньше выкладок, т.к. невязки получаются автоматически и не требуют усилий для нахождения. Ортогональность этого базиса нужна единственно в качестве гарантии того, что на очередном шаге мы повысим размерность подпространства, в котором решении уже найдено на единицу. Отсюда следует, что за *n* шагов мы исчерпаем пространство, и система будет решена. Естественно, что ортогональность упрощает вычисления, но она не изменяет конечный результат. Если мы сможем на каждом шаге гарантировать увеличение размерности подпространства каким-либо другим способом, то мы также придем к решению.

Результаты исследования. Предложена достаточно простая методика преподавания метода сопряженных градиентов в курсе вычислительной алгебры. Практика показывает, что в данном изложении метод не вызывает у студентов каких-либо затруднений.

Список использованной литературы:

1. Shewchuk, J.R. An Introduction to the Conjugate Gradient Method Without the Agonizing Pain / J.R. Shewchuk // Tech. rep. School of Computer ScienceCarnegie Mellon University. – 1994.
2. Gower, R.M. Conjugate Gradients: The short and painful explanation with oblique projections. [Електронний ресурс] – 2015. – Режим доступу: https://gowerrobert.github.io/pdf/reports/GowerR_Painful_PCG_projections.pdf
3. Saad, Y. Iterative methods for sparse linear systems / Y. Saad. – Philadelphia, Pa.: Society for Industrial and Applied Mathematics. – 2003. – p. 195.

Bibliography:

1. Shewchuk, J.R. (1994). An Introduction to the Conjugate Gradient Method Without the Agonizing Pain. *Tech. rep. School of Computer ScienceCarnegie Mellon University*.
2. Gower, R.M. (2015). Conjugate Gradients: The short and painful explanation with oblique projections. Retrieved from: https://gowerrobert.github.io/pdf/reports/GowerR_Painful_PCG_projections.pdf

3. Saad, Y. (2003). Iterative methods for sparse linear systems. Philadelphia, Pa.: Society for Industrial and Applied Mathematics.

GOLOVNYA Boris,
Doctor of Science, Professor, Department of Applied Mathematics and Informatics, The Bohdan Khmelnytsky National University of Cherkasy

ON THE TEACHING OF THE CONJUGATE GRADIENTS METHOD

Summary. Introduction. The conjugate gradient method is one of the fastest converging iterative methods for solving systems of linear algebraic equations with a symmetric positive definite matrix. But, if the software implementation of the method is very simple, then its mathematical foundations are quite complicated (see, for example, [1], [2], [3]). Very often, when explaining it, such concepts as projection methods, orthogonalization according to Arnoldi and Lanczos, the steepest descent, Krylov spaces, conjugate directions, and much more are used. All this makes the method incomprehensible on an intuitive level. For this reason, the conjugate gradient method is very difficult to explain to students. The paper presents a slightly different substantiation of the method. Here we use only the concept of orthogonality, the Gram-Schmidt method, and the concept of an A-scalar product. The concept of projection is used only to improve an already constructed and understandable method. This approach is intuitive and much simpler methodically than the traditional approach. As a result, in this description, the method is assimilated by students much easier.

The Purpose is to offer a simple and understandable explanation of the conjugate gradient method.

Result and conclusion. The simple and understandable explanation of the conjugate gradient method is offered.

Keywords: systems of linear algebraic equations, iterative methods for solving systems of linear algebraic equations, conjugate gradient method.

Одержано редакцією 16.04.2019 р.
Прийнято до публікації 04.09.2019 р.