УДК 519.853.32

Минимизация нелинейных функций при линейных ограничениях

Г.И. Забиняко, Е.А. Котельников

Институт вычислительной математики и математической геофизики Сибирского отделения Российской академии наук, просп. Акад. М.А. Лаврентьева, 6, Новосибирск, 630090

E-mail: zabin@rav.sscc.ru (Забиняко Г.И.)

Забиняко Г.И., Котельников Е.А. Минимизация нелинейных функций при линейных ограничениях // Сиб. журн. вычисл. математики / РАН. Сиб. отд-ние. — Новосибирск, 2013. — Т. 16, № 3. — С. 229–242.

В статье приводятся некоторые вопросы численной реализации алгоритмов из пакета программ для решения задач минимизации нелинейных функций (в том числе негладких) с учетом линейных ограничений, заданных разреженными матрицами. Имеются примеры решения тестовых задач.

Ключевые слова: *нелинейное программирование, приведенный градиент, метод сопряженных градиентов, квазиньютоновский метод, субградиентный метод, базис, супербазис.*

Zabinyako G.I., Kotel'nikov E.A. Minimization of nonlinear functions with linear constraints // Siberian J. Num. Math. / Sib. Branch of Russ. Acad. of Sci. — Novosibirsk, 2013. — Vol. 16, № 3. — P. 229–242.

In this paper, some aspects of numerical realization of algorithms from the software package for solving problems of minimization of nonlinear functions including non-smooth functions with allowance for the linear constraints set by sparse matrices are considered. Examples of the solution of test problems are presented.

Key words: *nonlinear programming, reduced gradient, method of conjugate gradients, quasi-Newton method, subgradient method, basis, superbasis.*

Введение

Рассматривается реализация метода приведенного градиента (ПГ) для решения задачи минимизации нелинейной функции при линейных ограничениях: найти

$$\min f(x) \tag{1}$$

при ограничениях:

$$Ax = b, \tag{2}$$

$$\alpha \le x \le \beta. \tag{3}$$

Здесь A — матрица размера $m \times n$; векторы x, α, $\beta \in R^n$, $b \in R^m$. В системе ограничений (2) допустимы ограничения-неравенства, которые сводятся к ограничениям-равенствам введением дополнительных переменных. Область определения дополнительной переменной x_i есть полуось $x_i \ge 0$, а соответствующий столбец матрицы A — единичный орт со знаком "+" или "−" в зависимости от типа неравенства " $\le$ " или " $\ge$ ". Задаваемая начальная точка x^0 может не удовлетворять условиям (2), (3). В этом случае находится допустимая точка, ближайшая в некотором смысле к точке x^0. Кроме того, точка x^0 может быть вообще не задана, тогда к нахождению x^0 привлекается процедура поиска начальной допустимой точки.

Алгоритм решения задачи (1)–(3) построен на базе алгоритма (MINOS), разработанного Муртафом и Сондерсом [1] для решения больших нелинейных задач с линейными ограничениями с использованием технологии ПГ.

В схему алгоритма MINOS авторами статьи встроены три алгоритма безусловной оптимизации: метода сопряженных градиентов, квазиньютоновского метода, субградиентного метода (r-алгоритма). При выборе метода необходимо учитывать размерность задачи и дифференциальные свойства целевой функции. Использование метода сопряженных градиентов позволяет решать задачи большой размерности с непрерывно дифференцируемыми целевыми функциями; квазиньютоновский метод обеспечивает решение с высокой точностью гладких задач небольшой и средней размерности; субградиентный метод применим в задачах с негладкими целевыми функциями.

Кроме того, в схему алгоритма MINOS авторами добавлены: эвристические процедуры, позволяющие, по нашему мнению, повысить численную устойчивость вычислительного процесса; алгоритмы поиска допустимого базиса; алгоритм поиска допустимой точки, ближайшей к заданной; алгоритм устранения недопустимости переменных; алгоритмы перепостроения и корректировки матриц, обратных базисным.

Однократное применение шагов алгоритма ПГ эквивалентно одной итерации модифицированного симплекс-метода в применении к задаче линейного программирования размера $m \times n$ плюс нескольких итераций алгоритма безусловной минимизации к задаче размерности, меньшей n.

В связи с тем, что целевая функция нелинейна, текущая точка оптимизирующей последовательности x^k не обязательно расположена в вершине многогранника, заданного ограничениями. Тогда в дополнение к базисным и небазисным переменным, которые используются в симплекс-методе, появляются переменные еще одного типа, которые называют супербазисными. Список супербазисных переменных формируется на итерациях по следующим правилам: i-я переменная является супербазисной, если, во-первых, она не является базисной и, во-вторых, удовлетворяет одному из условий: $\alpha_i < x_i < \beta_i$; $x_i = \alpha_i$ и $p_i^{(k)} > 0$; $x_i = \beta_i$ и $p_i^{(k)} < 0$, где $p_i^{(k)}$ — i-я компонента вектора смещения из точки x^k. Пусть x_B, x_S, x_N — векторы базисных, супербазисных и небазисных переменных, $x^\top = (x_B^\top, x_S^\top, x_N^\top)$ и B, S, N — матрицы, состоящие из столбцов матрицы ограничений $A = \{\alpha_{ij}\}$, $i = 1, 2, \ldots, m$, $j = 1, 2, \ldots, n$, которые соответствуют базисным, супербазисным и небазисным переменным; B — невырожденная матрица размера $m \times m$, S — матрица размера $m \times s$, N — матрица размера $m \times (n - m - s)$. Тогда линейные ограничения (2) можно записать в следующем виде:

$$Bx_B + Sx_S + Nx_N = b \tag{4}$$

или

$$x_B = B^{-1}(b - Sx_S - Nx_N). \tag{5}$$

Если на каком-либо отрезке процесса решения задачи величина шага вдоль направления сдвига $p^{(k)}$ до точки минимума функции f меньше величины шага до границы параллелепипеда, заданного ограничениями (3), тогда состав базисных, супербазисных и небазисных переменных не меняется, и в дальнейшем такой отрезок оптимизирующей последовательности будем называть подзадачей. При решении подзадачи небазисные переменные сохраняют свои значения, а базисные и супербазисные переменные принимают значения в пределах своих границ, при этом согласно (5) базисные переменные зависимы от супербазисных.

Пусть

$$F(x_S) = f(x_B, x_S, x_N) = f(B^{-1}(b - Sx_S - Nx_N), x_S, x_N),$$

тогда процесс решения подзадачи сводится к минимизации $F(x_S)$ при простых ограничениях $\alpha_B \le x_B \le \beta_B$, $\alpha_S \le x_S \le \beta_S$, где α_B, β_B, α_S, β_S — векторы, компоненты которых есть величины α_j и β_j, соответствующие базисным и супербазисным переменным. Для гладкой функции f градиент h функции F называют приведенным градиентом функции f, а матрицу H вторых производных функции F — приведенной матрицей вторых производных функции f. Обозначим $W = B^{-1}S$ и $V = [-W^\top I \;\; 0]$. Здесь V — матрица размера $s \times n$, состоящая из трех подматриц: $W^\top$ размера $s \times m$, соответствующая базисным переменным; единичная матрица I размера $s \times s$, соответствующая супербазисным переменным; нулевая матрица 0 размера $s \times (n-m-s)$, соответствующая небазисным переменным. Тогда, согласно [1], $h^{(k)} = Vg^{(k)}$ — приведенный градиент в текущей точке x^k, где $g^k = \nabla f(x^k)$, и $H^k = VG^{(k)}V^\top$ — приведенная матрица вторых производных функции f, где $G^{(k)}$ — матрица вторых производных функции f в точке x^k. Если функция f негладкая, то в качестве g^k выбирается субградиент.

Величину шага вдоль направления $p^{(k)}$ до точки минимума функции f можно находить для гладкой функции с помощью квадратичной или кубической интерполяции, для негладкой функции предусмотрен так называемый адаптивный алгоритм.

Подзадача считается решенной, если величина $\| h \|_\infty$ мала. В этом случае происходит, если это возможно, пополнение списка супербазисных переменных за счет небазисных; если пополнение невозможно, то задача решена полностью. Первым направлением спуска p_S любой подзадачи является приведенный антиградиент, т. е. $p_S = -h$.

1. Схема алгоритма ПГ

При описании метода приведенного градиента в применении к решению задачи (1)–(3) будем придерживаться схемы, приведенной в [1].

Пусть мы имеем:

а) допустимую текущую точку $x^\top = (x_B^\top, x_S^\top, x_N^\top)$, удовлетворяющую соотношению (4);

б) соответствующее значение целевой функции $f(x)$ и градиента функции $f : g(x)^\top = [g_B^\top, g_S^\top, g_N^\top]$;

в) LU — разложение обратной базисной матрицы;

г) правило Φ, задающее последовательность операций поиска направления p_S в пространстве супербазисных переменных на основе значения приведенного градиента h $(p_S = \Phi(h))$; правило Φ определяется методом безусловной минимизации;

д) допуски на сходимость и ошибки вычислений.

Вводится допуск Tg на сходимость задачи (входной параметр). Кроме Tg используется допуск Ek на сходимость текущей подзадачи ($Ek \ge Tg$). Значение Ek уменьшается при переходе от одной подзадачи к другой, что позволяет сократить время решения задачи за счет более грубого решения подзадач на начальных стадиях процесса. Начальное значение Ek определяется как некоторая доля $\| h \|_\infty$ в начале процесса.

Шаг 1. Вычислить вектор y, удовлетворяющий равенству $B^\top y = g_B$.

Шаг 2. Если число супербазисных переменных не равно нулю, то вычислить приведенный градиент $h = g_s - S^\top y$. Если $\| h \|_\infty > Ek$, то перейти на шаг 4.

Шаг 3. Начало подзадачи (увеличение числа супербазисных переменных):

а) вычислить вектор $\mu = g_N - N^\top y$;

б) выбрать μ_0 — минимальную отрицательную (максимальную положительную) компоненту вектора μ, соответствующую переменной, равной своей нижней (верхней) границе. Если $|\,\mu_0\,| \geq Ek$, то определить $\bar{\mu} = \max\{Ek, \gamma \cdot |\,\mu_0\,|\}$, где γ — некоторое заранее заданное число, $0 < \gamma \leq 1$. Тогда список супербазисных переменных пополнится за счет тех небазисных переменных, у которых компоненты μ_j вектора μ удовлетворяют условию $\mu_j < -\bar{\mu}$ и $x_j = \alpha_j$; $\mu_j > \bar{\mu}$ и $x_j = \beta_j$. Если $|\,\mu_0\,| < Ek$ и $Ek = Tg$, то процесс останавливается, так как удовлетворены необходимые условия оптимальности решения; если $Ek > Tg$ и $|\mu_0| > Tg$, то в качестве нового значения Ek берется $\max\{Tg, \eta \cdot |\,\mu_0\,|\}$, где $0 < \eta \leq 1$.

Шаг 4. Вычисление направления p:

а) вычислить $p_S = \Phi(h)$;

б) вычислить $p_B = -B^{-1}(Sp_S)$;

в) сформировать вектор $p^\top = (p_B^\top, p_S^\top, 0^\top)$.

Шаг 5. Определение максимально возможного шага вдоль направления p:

а) найти $\lambda_1 \geq 0$ — максимальное положительное λ, при котором точка $x + \lambda p$ допустима;

б) если $\lambda_1 = 0$, то перейти на шаг 8.

Шаг 6. Определение величины сдвига вдоль направления p:

а) найти величину λ_2, такую, что $f(x + \lambda_2 p) = \min_{0<\lambda\leq\lambda_1} f(x + \lambda p)$;

б) определить новое значение текущей точки x, т. е. $x = x+\lambda_2 p$, и вычислить значения функции f и градиента g в этой точке;

в) если $\lambda_1 = \lambda_2$, то перейти на шаг 8.

Шаг 7. Проверка качества вычислительного процесса. Контроль устойчивости решения подзадачи для метода сопряженных градиентов, квазиньютоновского метода и r-алгоритма различен. Результаты проверки повлияют на формирование правила Φ на следующей итерации. После выполнения операций этого шага перейти на шаг 1.

Шаг 8. Базисная или супербазисная переменная с номером i достигла своей границы, $0 < i \leq m + s$.

1. Базисная переменная $i \leq m$ приняла свое граничное значение:

   а) i-й столбец матрицы B перевести в матрицу N, а на его место поставить j-й столбец матрицы S, где j выбирается так, чтобы матрица B осталась невырожденной. Для этого решается уравнение $B^\top y = e_i$, где e_i — единичный орт с единицей на i-м месте, и в качестве j выбирается номер, на котором достигается $\max_k |y^\top Se_k|$;

   б) сделать соответствующую перестановку компонент векторов x_B и x_S;

   в) изменить матрицы L, U с учетом изменений в списке базисных переменных и перейти на шаг 1.

2. Супербазисная переменная i $(m < i \leq m + s)$ приняла граничное значение:

   а) перевести i-ю переменную в небазисную и i-й столбец из матрицы S в матрицу N;

   б) перейти на шаг 1.

2. Алгоритмы безусловной минимизации

Прежде чем переходить к описанию методов безусловной минимизации, целесообразно указать особенности учета двухсторонних ограничений $\alpha \le x \le \beta$ в каждом из них.

В методе сопряженных градиентов при достижении какой-либо базисной или супербазисной переменной своей границы в качестве направления сдвига на следующей итерации выбирается приведенный антиградиент.

В квазиньютоновском методе и r-алгоритме при достижении базисной переменной граничного значения в качестве направления сдвига также выбирается приведенный антиградиент в квазиньютоновском методе и приведенный субградиент с обратным знаком в r-алгоритме. Если супербазисная переменная приняла граничное значение, то в квазиньютоновском методе и r-алгоритме корректируются матрицы, с помощью которых в этих алгоритмах определяются направления смещения.

Рассмотрим теперь процедуру контроля вычислительного процесса решения подзадачи, т. е. когда в течении некоторого числа итераций точка минимума вдоль направления расположена внутри параллелепипеда, определенного ограничениями $\alpha \le x \le \beta$. Задается малое положительное число ε (входной параметр). На каждой итерации подзадачи вычисляется величина del, которая для метода сопряженных градиентов и квазиньютоновского метода равна относительному убыванию функции f:

$$del = \frac{f(x^{k-1}) - f(x^k)}{\max\{1, |f(x^k)|\}},$$

а для r-алгоритма $del = \lambda \cdot \| p_s \|_\infty$, где λ — величина сдвига вдоль направления до точки минимума. Тогда, если в течении пяти итераций подряд $del < \varepsilon$, то список супербазисных переменных пополняется (если это возможно) за счет подходящих небазисных, несмотря на то, что не выполнено необходимое условие окончание подзадачи: $\| h \|_\infty \le Ek$. Если пополнить список супербазисных переменных невозможно, то процесс решения задачи прекращается. В этом случае полученное приближенное решение задачи не удовлетворяет необходимому условию оптимальности $\| h \|_\infty \le Tg$.

2.1. Алгоритм сопряженных градиентов

Пусть функция $f(x)$, которая подлежит минимизации, является дважды непрерывно дифференцируемой функцией векторного аргумента $x \in R^n$.

Приближения к минимуму в стандартной схеме сопряженных градиентов строятся на основе итерационного процесса: $p^{(0)} = -\nabla f(x^0)$; $p^{(i)} = -\nabla f(x^i) + \beta_i p^{(i-1)}$, $i > 0$; $x^{i+1} = \arg\min_{\lambda > 0} f(x^i + \lambda p^{(i)})$. Здесь $\nabla f(x^i)$ — градиент f в текущей точке x^i, коэффициенты β_i вычисляются по формулам, которые обеспечивают сопряженность направлений $p^{(i)}$.

Известно, что результативность цикла из n итераций может существенно зависеть от того, произведено ли обновление. Использование антиградиента в качестве начального направления в цикле часто снижает эффективность итерации, на которой производится обновление. В связи с этим для обновления эффективней использовать не направление антиградиента, а некоторый другой вектор q [2]. В нашем случае используется уточненный алгоритм [3] с нестандартным обновлением.

Пусть $g^k = \nabla f(x^k)$; g_B^k, g_S^k, g_N^k — векторы, составленные из компонент вектора g^k и соответствующие базисным, супербазисным и небазисным переменным; h^k — приведенный градиент в точке x^k.

Действия операций схемы метода сопряженных направлений с нестандартным обновлением происходят на разных шагах схемы ПГ, поэтому при описании шагов схемы сопряженного градиента будет указано место в схеме, в котором данный шаг содержится.

В шаге 3 схемы ПГ положить $is = 0$, $ip = 0$ и выбрать положительные значения параметров ε_1, ε_2, d_1, d_2 таких, что $\varepsilon_1 > \varepsilon_2$ и $d_1 > d_2$.

В шаге 4 в пунктах а) и б) положить $p_S^{(k)} = -h^k$. Если $is = 0$, то перейти на шаг 4, пункт б) схемы ПГ, иначе вычислить

$$v_1 = \frac{|\,(h^k, h^{k-1})\,|}{\|\, h^k \,\|_2 \cdot \|\, h^{k-1} \,\|_2}.$$

Проверить, если $v_1 \geq \varepsilon_1$, то положить $is = 0$, $ip = 0$ и перейти на шаг 4, пункт б). При $v_1 < \varepsilon_1$ переопределить $p_S^{(k)} = p_S^{(k)} + \beta_k p_S^{(k-1)}$. Проверить, если $ip < 3$, то перейти на П1 данной схемы. В противном случае вычислить

$$v_2 = \frac{|\,(h^k, q)\,|}{\|\, h^k \,\|_2 \cdot \|\, q \,\|_2}.$$

Проверить, если $v_2 < \varepsilon_2$, то вычислить $\gamma_k = (h^k, z)/(q, z)$, положить $p_S^{(k)} = p_S^{(k)} + \gamma_k q$ и перейти на П1. При $v_2 \geq \varepsilon_2$ и $ip = 3$ положить $is = 0$, $ip = 0$ и перейти на шаг 4, пункт б). Если же $v_2 \geq \varepsilon_2$ и $ip > 3$, то присвоить $is = s$, где s — число супербазисных переменных текущей подзадачи.

П1. Проверить, если выполнены условия:

$$-d_1 \,\|\, h^k \,\|_2^2 \leq (p_S^{(k)}, h^k) \leq -d_2 \,\|\, h^k \,\|_2^2,$$

то перейти на шаг 4, пункт б), иначе положить $is = 0$, $ip = 0$ и перейти на шаг 3.

В шаге 7 присвоить $k = k+1$, $is = is+1$. При $ip \neq 0$ положить $ip = ip+1$. Проверить, если $is < s$ и $ip \neq 2$, то перейти на шаг 1. При $ip \neq 2$ запомнить векторы $q = p_S^{(k-1)}$ и $z = h^{k-1}$, присвоить $is = 1$, $ip = 1$ и перейти на шаг 1. Если же $ip = 2$, то переопределить $z = h^{k-1} - z$ и перейти на шаг 1.

Коэффициент β_k в методе сопряженных градиентов может определяться по-разному, в данном случае при проведении численных экспериментов использовались формулы Поллака–Рибьера, согласно которым

$$\beta_k = \frac{(h^k - h^{k-1}, h^k)}{\|\, h^{k-1} \,\|_2^2},$$

и Флетчера–Ривса:

$$\beta_k = \frac{\|\, h^k \,\|_2^2}{\|\, h^{k-1} \,\|_2^2}.$$

Корректное применение нестандартного обновления позволяет несколько продвинуться вперед при решении плохо обусловленных задач методом сопряженных градиентов.

Для приближенного определения минимума вдоль направления в алгоритме сопряженных градиентов предусмотрены процедуры квадратичной и кубической интерполяции.

2.2. Квазиньютоновский алгоритм

Приведенные ниже операции соответствуют пункту а) шага 4 схемы ПГ. В квазиньютоновском алгоритме очередное k-е направление спуска $p_S^{(k)}$ в подпространстве супербазисных перемена определяется из системы уравнений $B^k p_S^{(k)} = -h^k$, где B^k — текущая оценка приведенной матрицы вторых производных функции $f(x)$. Наиболее эффективные квазиньютоновские алгоритмы получаются при использовании для пересчета оценок B^k формулы Бройдена–Флетчера–Гольдфарба–Шано:

$$B^{k+1} = B^k + \frac{1}{\big(h^k, p_S^{(k)}\big)} h^k \big(h^k\big)^\top (x^k) + \frac{1}{\lambda_k \big(Y^k, p_S^{(k)}\big)} Y^k \big(Y^k\big)^\top,$$

где $Y^k = h^{k+1} - h^k$, а λ_k — шаг вдоль направления $p_S^{(k)}$.

В [4] предложена экономичная процедура поддержания на итерациях представления матриц B^k в факторизованной форме $B^k = L^k D^k (L^k)^\top$, где L^k — левая треугольная матрица с единицами на главной диагонали, D^k — диагональная матрица. Применение факторизованной формы позволяет обеспечить на итерациях строго положительную определенность матриц B^k с учетом ошибок округления.

Для определения шага λ_k вдоль направления также используются процедуры квадратичной или кубической интерполяции.

2.3. *r*-алгоритм

Для минимизации недифференцируемых функций хорошо себя зарекомендовали алгоритмы субградиентного типа с растяжением пространства в направлении разности двух последовательных субградиентов (r-алгоритмы) [5].

Пусть f — функция, которую необходимо минимизировать, а $\partial f(x)$ — ее субградиент в точке $x \in R^n$, $h(x)$ — приведенный субградиент в точке x. В начале процесса задаются матрица B^0, равная единичной матрице I размера $s \times s$, и коэффициент растяжения $\alpha > 0$ (обычно $2 \le \alpha \le 3$). Смещение из начальной точки x^0 производится в направлении, противоположном $h(x^0)$. Далее на любой k-й итерации r-алгоритма последовательно определяются следующие величины: $Y^k = h^k - h^{k-1}$, $r^k = (B^k)^\top Y^k$, $\xi^k = r^k / \| r^k \|_2$, $B^{k+1} = B^k\big(I + \big(\frac{1}{\alpha} - 1\big)\xi^k\big(\xi^k\big)^\top\big)$, $p_S^{(k)} = \big(B^{k+1}\big)^\top h^k$. Приведенные вычисления производятся в пункте а) шага 4 схемы ПГ.

В итерационном процессе вместо матриц B^k можно использовать симметричные матрицы $H^k = B^k\big(B^k\big)^\top$. При этом

$$H^{k+1} = H^k + \left(\frac{1}{\alpha^2} - 1\right) \frac{H^k Y^k \big(Y^k\big)^\top H^k}{\big(H^k Y^k, Y^k\big)}.$$

Использование симметричных матриц H^k вместо B^k позволило бы существенно сократить требования алгоритма к объему памяти. Трудность непосредственного использования матриц H^k вместо B^k состоит в том, что B^k стремится к нулевой матрице с ростом k. Накопление ошибок округления при рекуррентном пересчете H^k может приводить к тому, что с увеличением k возмущенная матрица $\tilde{H}^k$ перестает быть положительно определенной.

Авторами на основе алгоритма из [4] предложен надежный способ пересчета на итерациях матриц H^k в факторизованной форме. Предложенная процедура, как и в квазиньютоновском алгоритме, обеспечивает переход на итерациях от представления H^k в виде $L^k D^k \big(L^k\big)^\top$ к $L^{k+1} D^{k+1} \big(L^{k+1}\big)^\top$ для H^{k+1}.

Для поиска смещения λ_k можно воспользоваться алгоритмами, основанными на квадратичной и кубической интерполяции, или специальным адаптивным алгоритмом, аналогичным алгоритму из [5]. Первые два рекомендуются для гладких f, а адаптивный алгоритм применим в случае недифференцируемых f.

3. Вспомогательные процедуры

Для поддержки нормального и устойчивого функционирования алгоритма решения задачи минимизации функции при линейных ограничениях используются ряд вспомогательных процедур, которые приводятся ниже.

3.1. Поиск допустимого базиса

Реализованы два способа поиска начального допустимого базиса.

В первом варианте решается вспомогательная задача: минимизировать $\sum_{j=1}^{m}\xi_j$ при условиях $Ax + I_m\cdot\xi = b$, $\alpha \le x \le \beta$, $\xi \ge 0$, где $\xi \in R^m$, I_m — единичная матрица размера $m \times m$.

Очевидно, что если исходная матрица совместна, то в оптимальном решении вспомогательной задачи $\sum_{i=1}^{m}\xi_i = 0$, и ее оптимальный базис является исходным для последующих вычислений.

Во втором варианте начальный базис формируется из переменных, соответствующих "нормальным" столбцам [1] (если удается выбрать m нормальных столбцов, то искомая матрица B — нижнетреугольная), подходящих дополнительных переменных, а в случае, когда лимит их исчерпан, то и искусственных переменных. Область определения дополнительной переменной: $\alpha = 0$, $\beta = +\infty$; искусственной: $\alpha = 0$, $\beta = 0$. Далее небазисным переменным назначается одно из своих граничных значений таким образом, чтобы базисные переменные x_i имели по возможности меньше удаление от отрезка $[\alpha_i, \beta_i]$.

После нахождения значений небазисных переменных $\bar{x}_{Ni}$ переопределяются значения базисных переменных. Пусть $x_B = B^{-1}(b - N\bar{x}_N)$, тогда новые значения базисных переменных $\bar{x}_{Bi}$:

$$\bar{x}_{Bi} = \begin{cases} \alpha_{Bi}, & \text{если} \quad x_{Bi} \le \alpha_{Bi}, \\ \beta_{Bi}, & \text{если} \quad x_{Bi} \ge \beta_{Bi}, \\ x_{Bi}, & \text{если} \quad \alpha_{Bi} < x_{Bi} < \beta_{Bi}. \end{cases}$$

На следующем этапе вычисляется вектор невязок $b' = b - B\bar{x}_B - N\bar{x}_N$. Если $\| b' \|_\infty = 0$, то точка $x^\top = \left(\bar{x}_B^\top, \bar{x}_N^\top\right)$ допустима. Если $\| b' \|_\infty > 0$, то формируется вспомогательная задача: минимизировать x_0 при условиях $Ax + b'x_0 = b$, $\alpha \le x \le \beta$, $x_0 \ge 0$.

Точка $x^\top = \left(\bar{x}_B^\top, \bar{x}_N^\top, x_0\right)$, $x_0 = 1$ удовлетворяет ограничениям этой задачи, поэтому можно взять ее в качестве начальной. Если исходная задача имеет допустимое решение, то в результате решения вспомогательной задачи будем иметь $x_0 = 0$, а оптимальный базис можно использовать в качестве начального для дальнейших вычислений.

Алгоритм решения вспомогательной задачи можно модифицировать, позволив базисным переменным принимать значения вне интервала $[\alpha_{Bi}, \beta_{Bi}]$. В этом случае минимизируется суммарное отклонение недопустимых переменных от своих ближайших границ. Для решения такой задачи используется алгоритм Д. Рарика [1].

3.2. Поиск допустимой точки, ближайшей к начальной

Если заданная начальная точка x^0 не удовлетворяет условиям (2), (3), то находится допустимая точка, ближайшая к точке x^0. Для этого решается задача: минимизировать $\| x - x^0 \|_2^2$ при ограничениях (2), (3). В качестве начального базиса берется базис, найденный в предыдущем пункте. Для решения этой задачи применяется линейный метод сопряженных градиентов с использованием технологии ПГ, а результат решения является стартовым для дальнейших вычислений.

3.3. Корректировка обратной базисной матрицы на итерациях

Рассматривается ситуация, когда на некоторой итерации базисная переменная с номером p достигает своего граничного значения, тогда столбец a_p матрицы B должен быть заменен одним из супербазисных столбцов a_q. Пусть $\bar{B}$ — новая базисная матрица, тогда $\bar{B} = B + (a_q - Be_p)e_p^\top$, где e_p — единичный орт. На итерациях метода при замещении базисного столбца может осуществляться рекуррентный пересчет LU представления текущей базисной матрицы методом Бартелса–Голуба [6] либо используется метод Форреста–Томлина [7] для корректировки на итерациях матриц, обратных базисным.

Мы предлагаем для корректировки использовать метод Форреста–Томлина. Он позволяет в большей степени по сравнению с методом Бартелса–Голуба сохранять разреженность матрицы U^{-1}, хотя численная устойчивость его ниже. Поэтому в программах метод пересчета матрицы U^{-1} дополнен операциями контроля устойчивости.

Подробно вопрос о корректировке матрицы B^{-1} на итерациях рассмотрен в [1].

3.4. Перепостроение обратных матриц

В результате корректировок обратных базисных матриц на итерациях возможно накопление ошибок вычислений, которые вызывают искажение обратной матрицы так, что матрица $B^{-1}B$ может сильно отличаться от единичной. Искажение матрицы B^{-1}, в свою очередь, влечет серьезное последствие для вычислительного процесса, так как текущая точка $x^\top = (x_B^\top, x_S^\top, x_N^\top)$ в этом случае становится недопустимой. Поэтому через некоторое число корректировок необходимо полностью восстановить (перепостроить) обратную базисную матрицу. Авторы располагают двумя версиями процедуры перепостроения матрицы B^{-1}.

В основу первого алгоритма перепостроения обратной матрицы положен алгоритм P^4 [8], в котором на основе эвристик делается попытка минимизировать общее число ненулевых элементов в мультипликативном представлении LU-разложения B^{-1}. Минимизация достигается за счет приведения путем перестановок строк и столбцов матрицы B к виду, по возможности близкому к нижнетреугольной. Однако в алгоритме P^4 не учитываются численные значения кандидатов на ведущие элементы при назначении их позиций, что ведет к численной неустойчивости, поэтому реализован модифицированный алгоритм P^4, в котором при назначении ведущего элемента контролируется его допустимый уровень. При нарушении кандидатом допустимого уровня соответствующий столбец переводится в резерв в надежде использовать их при более выгодных условиях. Такая стратегия позволяет повысить численную устойчивость вычислений с матрицей B^{-1}, однако платой за повышение устойчивости является увеличение заполненности мультипликаторов $U^{-1}L^{-1}$-разложения.

Второй алгоритм основан на идеях, изложенных в [9]. Эта теоретического плана работа указала на новые технологические возможности построения программ решения СЛАУ с разреженными матрицами (см., например, [10–12]). Предварительное определение номеров ведущих элементов (построение трансверсали) является результатом решения задач назначения по критерию максимизации произведения модулей элементов из трансверсали или критерию трансверсали узкого места [13]. С целью уменьшения заполненности мультипликативного представления L^{-1} и U^{-1} очередность выбора ведущих элементов осуществляется с помощью пакета METIS [14, 15]. Построение мультипликаторов сопровождается контролем величины модулей ведущих элементов.

Особенности применения процедур перепостроения нагляднее проиллюстрировать на примере решения двух задач линейного программирования (ЛП) из [16]. В ЛП обычно чаще производятся обращения к перепостроению матриц, нежели в методе приведенного градиента. В табл. 1 приведены существенные различия в применении процедур перепостроения.

Таблица 1.

Имя задачи	$m \times n$	nz	перепостроение B^{-1}	V	α	J_k	J_{er}
MAROS-R7	3136×9408	144848	A_1	1435026	$0.4E-3$	81	28
			A_2	256204	$0.2E-3$	68	40
STOCFOR3	16675×15695	64875	A_1	61711	0.50	90	–
			A_2	189794	0.10	90	–

Обозначения, используемые в таблице:

nz — число ненулевых элементов в исходной матрице A;

A_1 — перепостроение B^{-1} на основе алгоритма P^4;

A_2 — перепостроение B^{-1} на основе поиска трансверсали из условия максимизации произведения модулей ее элементов;

V — максимальный размер памяти, занимаемый B^{-1} после перепостроения;

α — минимальный ведущий элемент на этапе перепостроения B^{-1};

J_k — общее число перепостроений;

J_{er} — в том числе количество перепостроений B^{-1} из-за обнаружения повышенного уровня ошибок вычислений.

При решении задачи MAROS-R7 встречаются плохо обусловленные базисные матрицы. В этом случае проявляются преимущества процедуры перепостроения, основанной на поиске трансверсали из условия максимизации произведения модулей элементов. В задаче STOCFOR3 все элементы базисных матриц хорошо масштабированы, и эвристики, заложенные в алгоритме P^4, позволяют минимизировать заполненность матриц, обратных базисным.

3.5. Устранение недопустимости переменных

После каждого перепостроения обратной базисной матрицы необходимо проверить на допустимость значений базисных переменных $x_B = B^{-1}(b - Sx_S - Nx_N)$ с новой матрицей B^{-1}. Если $\alpha_B \le x_B \le \beta_B$, то процесс решения задачи продолжается; если хотя бы для одной базисной переменной i $x_{Bi} < \alpha_{Bi}$ или $x_{Bi} > \beta_{Bi}$, то необходимо устранить недопустимость переменных. Напомним, что для дополнительных и искусственных переменных $\alpha_i = 0$, $\beta_i = +\infty$. Для устранения недопустимости формируется задача с ограничениями (2), (3) и целевой функцией $c^\top x$, подлежащей максимизации; c — вектор, компоненты которого соответствуют базисным переменным, задается на каждой итерации

$$c_i = \begin{cases} -1, & \text{если} \quad x_{Bi} > \beta_{Bi}, \\ 1, & \text{если} \quad x_{Bi} < \alpha_{Bi}, \\ 0, & \text{если} \quad \alpha_{Bi} \le x_{Bi} \le \beta_{Bi}. \end{cases}$$

Коэффициенты c_i используются для вычисления вектора двойственных переменных и направлений сдвига p_B, p_S для базисных и супербазисных переменных.

Величину сдвига вдоль направления можно вычислять двумя способами:

а) в качестве длины шага можно выбрать величину, при которой какая-либо базисная или супербазисная переменная достигает одного из своих граничных значений;

б) длину шага выбрать из условия минимума суммы отклонений вдоль данного направления, что соответствует алгоритму Рарика [1].

В пакете устранение недопустимости реализовано по второму варианту. Задачу минимизации невязок на итерации можно свести к задаче минимизации кусочно-линейной выпуклой функции $\varphi(x)$ одной переменной, определенной при $x \ge 0$. Выпуклая функция φ является суммой кусочно-линейных функций φ_i, определенных для базисных переменных следующим образом:

а) если $x_{Bi} < \alpha_{Bi}$ и $p_{Bi} < 0$, то $\varphi_i(x) = 0$;

б) если $x_{Bi} < \alpha_{Bi}$ и $p_{Bi} > 0$, то

$$\varphi_i(x) = \begin{cases} p_{Bi}(\gamma_i' - x), & \text{если} \quad x < \gamma_i', \\ p_{Bi}(x - \gamma_i''), & \text{если} \quad x > \gamma_i'', \\ 0, & \text{если} \quad \gamma_i' \le x \le \gamma_i'', \end{cases}$$

где

$$\gamma_i' = \frac{\alpha_{Bi} - x_{Bi}}{p_{Bi}}, \quad \gamma_i'' = \frac{\beta_{Bi} - x_{Bi}}{p_{Bi}};$$

в) если $x_{Bi} > \beta_{Bi}$ и $p_{Bi} > 0$, то $\varphi_i(x) = 0$;

г) если $x_{Bi} > \beta_{Bi}$ и $p_{Bi} < 0$, то

$$\varphi_i(x) = \begin{cases} -p_{Bi}(\gamma_i' - x), & \text{если} \quad x < \gamma_i', \\ -p_{Bi}(x - \gamma_i''), & \text{если} \quad x > \gamma_i'', \\ 0, & \text{если} \quad \gamma_i' \le x \le \gamma_i'', \end{cases}$$

где

$$\gamma_i' = \frac{x_{Bi} - \beta_{Bi}}{-p_{Bi}}, \quad \gamma_i'' = \frac{x_{Bi} - \alpha_{Bi}}{-p_{Bi}};$$

д) если $\alpha_{Bi} \le x_{Bi} \le \beta_{Bi}$, то

$$\varphi_i(x) = \begin{cases} \mid p_{Bi} \mid (x - \gamma_i''), & \text{если} \quad x > \gamma_i'', \\ 0, & \text{если} \quad x \le \gamma_i'', \end{cases}$$

где

$$\gamma_i'' = \begin{cases} \frac{x_{Bi} - \alpha_{Bi}}{p_{Bi}}, & \text{если} \quad p_{Bi} < 0, \\ \frac{\beta_{Bi} - x_{Bi}}{p_{Bi}}, & \text{если} \quad p_{Bi} > 0; \end{cases}$$

е) если $p_{Bi} = 0$, то $\varphi_i(x) = 0$.

Упорядочим величины γ_i', γ_i'' по возрастанию их значений, запомним в соответствующем порядке номера переменных и обозначим через $\bar{\gamma}_i$ элементы отсортированного массива. Тогда на каждом отрезке $[\bar{\gamma}_i, \bar{\gamma}_{i+1}]$ функция φ линейна, а в точках $\bar{\gamma}_i$ имеет изломы. $\varphi(0) = \sum_{i \in I_1} p_{Bi} - \sum_{i \in I_2} p_{Bi}$, где $I_1 = \{i \mid x_{Bi} < \alpha_{Bi}\}$, $I_2 = \{i \mid x_{Bi} > \beta_{Bi}\}$. В симплекс-методе $\varphi(0)$ равнялась бы относительной оценки небазисного столбца, вводимого в базис.

Целью минимизации функции φ является нахождение номера переменной, которая покидает список базисных или супербазисных переменных. Алгоритм поиска минимума функции φ имеет вид:

1. Находится величина шага λ_s вдоль направления p_s в пространстве супербазисных переменных до их граничных значений. Пусть i_s — номер переменной, которая первой достигла границы.
2. Последовательно по возрастанию индекса просматриваются элементы $\bar{\gamma}_i$ и вычисляются $\tau_i = \varphi(0) - \sum_{k=1}^{i} p_{Bi_k}$, где i_k — номер переменной, стоящей на k-м месте после сортировки. Просмотр производится до тех пор, пока или τ_i перестанет быть положительной, или величина $\bar{\gamma}_i$ станет больше λ_s. В первом случае номер переменной i_k, при котором $\tau_k \le 0$, берется в качестве номера столбца, который выводится из базисной матрицы. Если τ_i остается положительной после перебора всех $\bar{\gamma}_i \le \lambda_s$, то из списка супербазисных переменных выводится переменная с номером i_s.

4. Численные эксперименты

Проиллюстрируем работу алгоритма на примере задач, сформированных на основе линейных ограничений задач из набора NETLIB [13] по правилам:

а) выбрать функцию f с известным безусловным минимумом x^* при начальной точке x^0;

б) в задаче линейного программирования из NETLIB определить новый вектор правых частей ограничений (2): $b = Ax^*$;

в) для произвольного целого числа k $(0 < k \le m)$ переопределить значения $b_i : b_i = b_i + \delta_i,\ i \le k$; значения δ_i задаются произвольным образом (можно случайно);

г) назначить тип каждого ограничения: для $i \le k$ и $\delta_i \ge 0$ ограничение “ $\le$ ”; для $i \le k$ и $\delta_i < 0$ ограничение “ $\ge$ ”; для $i > k$ ограничение “=”;

д) значения α, β из (3) выбирать так, чтобы $\alpha \le x^* \le \beta$.

Для построения тестов выбраны две функции:

1. Многомерное обобщение функции Розенброка для $n > 2$ [17]:

$$f(x) = \sum_{i=2}^{n} \Big(100(x_i - x_{i-1}^2)^2 + (1 - x_i)^2 \Big). \tag{6}$$

При начальной точке $\big(x^0\big)^\top = (-1.2, 1, \ldots, 1)$ имеем $\big(x^*\big)^\top = (1, 1, \ldots, 1)$ и $f^* = 0$.

2. Аппроксимация полиномами в l_1, n — любое [17]:

$$f(x) = \sum_{j=1}^{101} \Big| \sum_{i=1}^{n} x_i t_j^{i-1} \sum_{i=1}^{n} x_i^* t_j^{i-1} \Big|, \tag{7}$$

$$t_j = 0.01(j-1), \quad j = 1, 2, \ldots, 101.$$

При $\big(x^0\big)^\top = (0, 0, \ldots, 0)$, $\big(x^*\big)^\top = \big(1/n, 1/n, \ldots, 1/n\big)$, $f^* = 0$.

Во всех задачах $k = m/4,\ \delta_i = 0.1,\ \alpha_j = 0,\ \beta_j = 5$.

Для тестирования использовались следующие задачи из NETLIB: SC105(105×103), KB2(43×41), RECIPE(91×180), SC50A(50×48), SC50B(50×48), SHARE2B(96×79), FINNIS(497×614), SCORPION(388×358), GROW15(300×645), GROW22(440×946), GESA2(1392×1224), BLN2(2324×3489), MAROS-R7(3136×9408). В скобках указана размерность задачи ($m \times n$).

Во всех расчетах перепостроение обратных базисных матриц производилось с помощью модифицированного алгоритма P^4.

В табл. 2, 3 приводятся результаты минимизации функции (6) с учетом ограничений, построенных на основе приведенных выше задач.

В табл. 2–4: it — общее число итераций, включая итерации на поиск допустимой точки, ближайшей к x^0; NF — количество вычислений функции; NG — количество вычислений градиента; $\delta x = \max_i |x_i^* - 1.0|$; $\delta f = |f^*|$; $h_r = ||Ax^* - b||_\infty$; $h_c = \max_{j \in I_B} \left| \sum_{i=1}^m a_{ij} y_i^* - c_j^* \right|$, где I_B — список переменных оптимального базиса, y_i^* — оптимальные двойственные оценки, $c_j = \big(\nabla f(x^*)\big)_j$; t — время решения на PC CELERON 850 (в секундах).

Таблица 2. Метод сопряженных градиентов (вариант Полака–Рибьера). Кубическая интерполяция

Имя задачи	it	NF	NG	δx	δf	h_r	h_c	t
SC 105	654	1931	2524	$5.E-8$	$1.E-12$	$3.E-11$	$3.E-19$	0.4
RECIPE	2298	5955	8180	$3.E-7$	$9.E-11$	$5.E-9$	$1.E-18$	1.5
SHARE 2 B	1090	2975	3969	$3.E-7$	$2.E-10$	$5.E-11$	$8.E-17$	0.5
GROW 15	1180	2867	3723	$2.E-8$	$2.E-12$	$2.E-11$	$1.E-16$	3.5
FINNIS	5685	16588	21633	$3.E-8$	$4.E-12$	$6.E-9$	$2.E-12$	15.1
SCORPION	1316	3146	4156	$2.E-8$	$3.E-13$	$9.E-15$	$3.E-12$	1.5
GROW22	259	2555	525	$3.E-8$	$2.E-12$	$3.E-11$	$2.E-16$	3.0
GESA2	28159	78631	103938	$1.E-8$	$8.E-13$	$1.E-10$	$2.E-13$	102.1
BLN2	49367	132450	177049	$5.E-8$	$9.E-12$	$8.E-10$	$4.E-12$	645.2
MAROS-R 7	3459	1228	1909	$9.E-7$	$5.E-12$	$9.E-9$	$7.E-11$	708.0

Таблица 3. Квазиньютоновский метод. Кубическая интерполяция

Имя задачи	it	NF	NG	δx	δf	h_r	h_c	t
SC 105	186	447	572	$6.E-11$	$4.E-13$	$3.E-10$	$3.E-20$	0.1
KB2	43	46	59	$3.E-7$	$1.E-11$	$6.E-9$	$3.E-16$	0.0
RECIPE	295	556	778	$2.E-8$	$6.E-12$	$5.E-9$	$1.E-19$	0.8
SC50A	30	51	63	$1.E-9$	$8.E-11$	$4.E-11$	$8.E-21$	0.0
SC50B	28	47	59	$1.E-11$	$9.E-13$	$1.E-11$	$7.E-23$	0.0
SHARE 2 B	152	219	276	$3.E-10$	$8.E-9$	$5.E-11$	$9.E-18$	0.0
GROW15	709	946	1331	$5.E-9$	$3.E-13$	$2.E-11$	$2.E-17$	19.5
FINNIS	1976	2980	4316	$3.E-8$	$1.E-12$	$4.E-10$	$3.E-12$	22.6
SCORPION	525	602	821	$6.E-9$	$2.E-12$	$7.E-15$	$3.E-11$	0.9
GROW22	1006	997	210	$3.E-8$	$2.E-11$	$7.E-12$	$6.E-17$	8.0

В табл. 4 показаны результаты минимизации функции (7) r-алгоритмом с адаптивным алгоритмом поиска смещения. Здесь $\delta x = \max_j |x_j^* - 1/n|$.

Таблица 4.

Имя задачи	it	NF	NG	δx	δf	t
SC 105	204	593	146	$2.7E-2$	$4.7E-6$	1.0
KB2	357	3117	322	$2.3E-4$	$2.6E-6$	1.1
RECIPE	440	2697	399	$2.2E-2$	$2.2E-4$	4.1
SC50A	227	1489	190	$4.8E-9$	$3.6E-8$	0.6
SC50B	156	1112	126	$6.7E-9$	$2.9E-8$	0.5
SHARE2B	469	2323	382	$2.4E-3$	$2.3E-5$	1.6

5. Заключение

В статье приводится описание алгоритмов из пакета программ минимизации нелинейных функций с учетом линейных ограничений. При выборе метода минимизации могут быть учтены размерность задачи и дифференциальные свойства целевой функции. Использование метода сопряженных градиентов позволяет решать задачи большой размерности с непрерывно дифференцируемыми функциями. Квазиньютоновский метод обеспечивает решение задачи минимизации гладких функций небольшой и средней размерности с высокой точностью. Субградиентный алгоритм применим в задачах с негладкими целевыми функциями. В алгоритмах учитывается разреженность матриц ограничений. В зависимости от свойств матриц можно выбрать один из двух алгоритмов перепостроения матриц, обратных базисным, в которых предпочтение отдается критерию разреженности или численной устойчивости.

Литература

1. **Муртаф Б.** Современное линейное программирование. Теория и практика. — М.: Мир, 1984.
2. **Powell M.J.D.** Restart procedures for the conjugate gradient method // Math. Programming— 1977. — Vol. 12. — P. 241–254.
3. **Забиняко Г.И.** Процедуры обновления в методе сопряженных градиентов // Оптимизация. — 1989. — Вып. 46(63). — С. 5–13.
4. **Gill P.E., Murray W.** Quasi-Newton methods for unconstrained optimization // J. Inst. Maths. Appl. — 1972. — Vol. 9, № 1. — P. 91–108.
5. **Шор Н.З., Стеценко С.И.** Квадратичные экстремальные задачи и недифференцируемая оптимизация. — Киев: Наукова думка, 1989.
6. **Bartels R.H., Golub G.H.** The simplex method of linear programming using LU decomposition // Communication of ACM. — 1969. — № 12. — P. 266–268.
7. **Forrest J.J.H., Tomlin J.A.** Updating triangular factors of the basis to maintain sparsity in the product-form simplex method // Math. Programming. — 1972. — Vol. 2, iss. 1. — P. 263–278.
8. **Hellerman E., Rarick D.C.** The partitioned preassigned pivot procedure (p^4) // Sparse Matrices and their Applications / D.J. Rose and R.A. Willoughby. — N.Y.: Plenum Press. — 1972. — P. 68–76.
9. **Olschowka M., Neumaier A.** A new pivoting strategy for Gaussian elimination // Linear Algebra Appl. — 1996. — Vol. 240. — P. 131–151.
10. **Duff I.S., Koster J.** The design and use of algorithms for permuting large entries to the diagonal of sparse matrices // SIAM J. Matrix Anal. Appl. — 1999. — Vol. 20, № 4. — P. 889–901.
11. **Li X.S., Demmel J.W.** SuperLU DIST: A scalable distributed-memory sparse direct solver for unsymmetric linear systems // ACM Trans. Math. Software. — 2003. — Vol. 29, № 2. — P. 110–140.
12. **Schenk O., Gartner K.** Solving unsymmetric sparse systems of linear equation with PARDISO // Future Generation Computer Systems. — 2004. — Vol. 20. — P. 475–487.
13. **Забиняко Г.И.** Перепостроение обратных матриц // Сиб. журн. индустр. матем. — 2009. — Т. 12, № 3. — С. 41–51.
14. **Karypis G., Kumar V.** A fast and high quality multilevel scheme for partitioning irregular graphs // SIAM J. on Sci. Computing. — 1998. — Vol. 20, № 1. — P. 359–392.
15. **Karypis G., Kumar V.** METIS. A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices (Version 4.0). — http://www.cs.umn.edu/ ˜karypis.
16. http://www.netlib.org/lp/data.
17. **Поляк Б.Т.** Введение в оптимизацию. — М.: Наука, 1983.

Поступила в редакцию 23 января 2012 г.,
в окончательном варианте 31 мая 2012 г.