

Wstęp do metod numerycznych

Algebraiczna metoda gradientów sprzężonych

P. F. Góra

<http://th-www.if.uj.edu.pl/zfs/gora/>

2013

Metoda gradientów sprzężonych — motywacja

Rozważmy funkcję $f : \mathbb{R}^N \rightarrow \mathbb{R}$

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \mathbf{A} \mathbf{x} - \mathbf{b}^T \mathbf{x} + c, \quad (1)$$

gdzie $\mathbf{x}, \mathbf{b} \in \mathbb{R}^N$, $c \in \mathbb{R}$, $\mathbf{A} = \mathbf{A}^T \in \mathbb{R}^{N \times N}$ jest *symetryczna i dodatnio określona*. Przy tych założeniach, funkcja (1) ma dokładnie jedno minimum, będące zarazem minimum globalnym. Szukanie minimów dodatnio określonych form kwadratowych jest (względnie) łatwe i z praktycznego punktu widzenia ważne. Minimum to leży w punkcie spełniającym

$$\nabla f = 0. \quad (2)$$

Obliczmy

$$\begin{aligned}
 \frac{\partial f}{\partial x_i} &= \frac{1}{2} \frac{\partial}{\partial x_i} \sum_{j,k} A_{jk} x_j x_k - \frac{\partial}{\partial x_i} \sum_j b_j x_j + \underbrace{\frac{\partial c}{\partial x_i}}_0 \\
 &= \frac{1}{2} \sum_{j,k} A_{jk} \left(\underbrace{\frac{\partial x_j}{\partial x_i}}_{\delta_{ij}} x_k + x_j \underbrace{\frac{\partial x_k}{\partial x_i}}_{\delta_{ik}} \right) - \sum_j b_j \underbrace{\frac{\partial x_j}{\partial x_i}}_{\delta_{ij}} \\
 &= \frac{1}{2} \sum_k A_{ik} x_k + \frac{1}{2} \sum_j A_{ji} x_j - b_i = \frac{1}{2} \sum_k A_{ik} x_k + \frac{1}{2} \sum_j A_{ij} x_j - b_i \\
 &= (\mathbf{Ax} - \mathbf{b})_i .
 \end{aligned} \tag{3}$$

Widzimy zatem, że funkcja (1) osiąga minimum w punkcie, w którym zachodzi

$$\mathbf{Ax} - \mathbf{b} = 0 \Leftrightarrow \mathbf{Ax} = \mathbf{b}. \quad (4)$$

Rozwiązywanie układu równań liniowych (4) z macierzą symetryczną, dodatnio określoną jest równoważne poszukiwaniu minimum dodatnio określonej formy kwadratowej.

Przypuśćmy, że macierz \mathbf{A} jest przy tym *rzadka* i duża (lub co najmniej średnio-duża). Wówczas metoda gradientów sprzężonych jest godną uwagi metodą rozwiązywania (4)

Metoda gradientów sprzężonych, *Conjugate Gradients*, CG

$\mathbf{A} \in \mathbb{R}^{N \times N}$ symetryczna, dodatnio określona, \mathbf{x}_1 — początkowe przybliżenie rozwiązania równania (4), $0 < \varepsilon \ll 1$.

$$\begin{aligned} & \mathbf{r}_1 = \mathbf{b} - \mathbf{A}\mathbf{x}_1, \mathbf{p}_1 = \mathbf{r}_1 \\ & \textbf{while } \|\mathbf{r}_k\| > \varepsilon \\ & \quad \alpha_k = \frac{\mathbf{r}_k^T \mathbf{r}_k}{\mathbf{p}_k^T \mathbf{A} \mathbf{p}_k} \\ & \quad \mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k \mathbf{A} \mathbf{p}_k \\ & \quad \beta_k = \frac{\mathbf{r}_{k+1}^T \mathbf{r}_{k+1}}{\mathbf{r}_k^T \mathbf{r}_k} \\ & \quad \mathbf{p}_{k+1} = \mathbf{r}_{k+1} + \beta_k \mathbf{p}_k \\ & \quad \mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k \\ & \textbf{end} \end{aligned} \tag{5}$$

Wówczas zachodzą twierdzenia:

Twierdzenie 1. *Ciągi wektorów $\{\mathbf{r}_k\}$, $\{\mathbf{p}_k\}$ spełniają następujące zależności:*

$$\mathbf{r}_i^T \mathbf{r}_j = 0, \quad i > j, \quad (6a)$$

$$\mathbf{r}_i^T \mathbf{p}_j = 0, \quad i > j, \quad (6b)$$

$$\mathbf{p}_i^T \mathbf{A} \mathbf{p}_j = 0, \quad i > j. \quad (6c)$$

Twierdzenie 2. *Jeżeli $\mathbf{r}_M = 0$, to \mathbf{x}_M jest ścisłym rozwiązaniem równania (4).*

Dowód. Oba (sic!) dowody przebiegają indukcyjnie. □

Ciąg $\{x_k\}$ jest w gruncie rzeczy “pomocniczy”, nie bierze udziału w iteracjach, służy tylko do konstruowania kolejnych przybliżeń rozwiązania.

Istotą algorytmu jest konstruowanie dwu ciągów wektorów spełniających zależności (6). Wektory $\{r_k\}$ są wzajemnie prostopadłe, a zatem *w arytmetyce dokładnej* $r_{N+1} = 0$, wobec czego x_{N+1} jest poszukiwanym ścisłym rozwiązaniem.

Zauważmy, że ponieważ A jest symetryczna, dodatnio określona, warunek (6c) oznacza, że wektory $\{p_k\}$ są wzajemnie prostopadłe w metryce zadanej przez A . Ten właśnie warunek nazywa się warunkiem *sprzężenia względem A* , co daje nazwę całej metodzie.

Ten wariant metody gradientów sprzężonych nazywamy “algebraicznym”, gdyż przy założeniu, że *znamy* macierz A oraz wektor x_1 , możemy skonstruować ciągi $\{r_k, p_k, x_k\}$ metodami algebraicznymi.

W przyszłości poznamy wariant metody gradientów sprzężonych, w którym wszystkich kroków nie uda się w ten sposób wykonać.

Koszt metody

W arytmetyce dokładnej metoda zbiega się po N krokach, zatem jej koszt wynosi $O(N \cdot \text{koszt_jednego_kroku})$. Koszt jednego kroku zdominowany jest przez obliczanie iloczynu $A p_k$. Jeśli macierz A jest pełna, jest to $O(N^2)$, a zatem całkowity koszt wynosi $O(N^3)$, czyli tyle, ile dla metod dokładnych. Jeżeli jednak A jest rzadka, koszt obliczania iloczynu jest mniejszy, o ile obliczenie to jest odpowiednio zaprogramowane. Jeśli A jest pasmowa o szerokości pasma $M \ll N$, całkowity koszt wynosi $O(M \cdot N^2)$.

Problem!

W arytmetyce o skończonej dokładności kolejne generowane wektory nie są *ściśle* ortogonalne do swoich poprzedników — na skutek akumulującego się błędu zaokrąglenia rzut na poprzednie wektory może stać się z czasem znaczny. Powoduje to istotne spowolnienie metody.

Twierdzenie 3. *Jeżeli \mathbf{x} jest ścisłym rozwiązaniem równania (4), \mathbf{x}_k są generowane w metodzie gradientów sprzężonych, zachodzi*

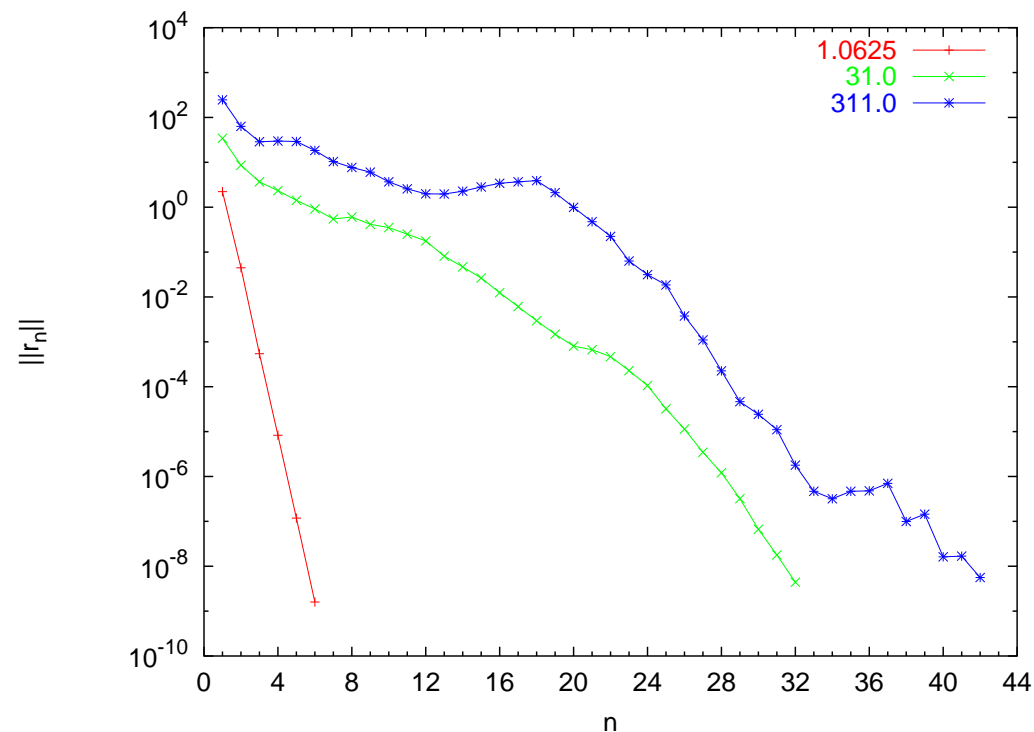
$$\|\mathbf{x} - \mathbf{x}_k\| \leq 2\|\mathbf{x} - \mathbf{x}_1\| \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right)^{k-1}, \quad (7)$$

gdzie κ jest współczynnikiem uwarunkowania macierzy \mathbf{A} .

Jeżeli $\kappa \gg 1$, zbieżność może być bardzo wolna.

Przykład

Rozwiązujemy układy równań z *małymi* (32×32) macierzami symetrycznymi, rzeczywistymi, dodatnio określonymi, o różnych współczynnikach uwarunkowania. Poniższy rysunek pokazuje normy kolejnych wektorów \mathbf{r}_n . Iteracje zatrzymywano, gdy $\|\mathbf{r}_n\| \leq 10^{-8}$. W arytmetyce dokładnej $\|\mathbf{r}_{n>32}\| \equiv 0$.



“Prewarunkowana” (*preconditioned*) metoda gradientów sprzężonych

Spróbujmy przyspieszyć zbieżność odpowiednio modyfikując równanie (4) i algorytm (5), jednak tak, aby

- nie zmienić rozwiązania,
- macierz zmodyfikowanego układu pozostała symetryczna i dodatnio określona, aby można było zastosować metodę gradientów sprzężonych,
- macierz zmodyfikowanego układu pozostała rzadka, aby jeden krok iteracji był numerycznie tani,
- macierz zmodyfikowanego układu miała niski współczynnik uwarunkowania.

Czy to się w ogóle da zrobić? [Okazuje się, że tak!](#)

Postępujemy następująco: Niech $C \in \mathbb{R}^{N \times N}$ będzie odwracalną macierzą symetryczną, rzeczywistą, dodatnio określoną. Wówczas $\widetilde{A} = C^{-1}AC^{-1}$ też jest symetryczna, rzeczywista, dodatnio określona.

$$C^{-1}A \underbrace{C^{-1}C}_{\mathbb{I}} x = C^{-1}b, \quad (8a)$$

$$\widetilde{A}\tilde{x} = \tilde{b}, \quad (8b)$$

gdzie $\tilde{x} = Cx$, $\tilde{b} = C^{-1}b$. Do równania (8b) stosujemy teraz metodę gradientów sprzężonych.

W każdym kroku iteracji musimy obliczyć (tylko, bo odnosi się to do “tyldowanego” układu (8b))

$$\alpha_k = \frac{\tilde{\mathbf{r}}_k^T \tilde{\mathbf{r}}_k}{\tilde{\mathbf{p}}_k^T \tilde{\mathbf{A}} \tilde{\mathbf{p}}_k} = \frac{\tilde{\mathbf{r}}_k^T \tilde{\mathbf{r}}_k}{\tilde{\mathbf{p}}_k^T \mathbf{C}^{-1} \mathbf{A} \mathbf{C}^{-1} \tilde{\mathbf{p}}_k}, \quad (9a)$$

$$\tilde{\mathbf{r}}_{k+1} = \tilde{\mathbf{r}}_k - \alpha_k \tilde{\mathbf{A}} \tilde{\mathbf{p}}_k = \tilde{\mathbf{r}}_k - \alpha_k \mathbf{C}^{-1} \mathbf{A} \mathbf{C}^{-1} \tilde{\mathbf{p}}_k, \quad (9b)$$

$$\beta_k = \frac{\tilde{\mathbf{r}}_{k+1}^T \tilde{\mathbf{r}}_{k+1}}{\tilde{\mathbf{r}}_k^T \tilde{\mathbf{r}}_k}, \quad (9c)$$

$$\tilde{\mathbf{p}}_{k+1} = \tilde{\mathbf{r}}_{k+1} + \beta_k \tilde{\mathbf{p}}_k, \quad (9d)$$

$$\tilde{\mathbf{x}}_{k+1} = \tilde{\mathbf{x}}_k + \alpha_k \tilde{\mathbf{p}}_k. \quad (9e)$$

Równania (9) zawierają jawne odniesienia do macierzy C^{-1} , co nie jest zbyt wygodne. Łatwo się przekonać, iż za pomocą prostych przekształceń macierz tę można „usunąć”, tak, iż pozostaje tylko jedno jej nietrywialne wystąpienie. Zdefiniujmy mianowicie

$$\tilde{\mathbf{r}}_k = C^{-1}\mathbf{r}_k, \quad \tilde{\mathbf{p}}_k = C\mathbf{p}_k, \quad \tilde{\mathbf{x}}_k = C\mathbf{x}_k. \quad (10)$$

W tej sytuacji $\tilde{\mathbf{r}}_k^T \tilde{\mathbf{r}}_k = (C^{-1}\mathbf{r}_k)^T C^{-1}\mathbf{r}_k = \mathbf{r}_k^T (C^{-1})^T C^{-1}\mathbf{r}_k = \mathbf{r}_k^T C^{-1} C^{-1}\mathbf{r}_k = \mathbf{r}_k^T (C^{-1})^2 \mathbf{r}_k$ etc.

Wówczas równania (9) przechodzą w

$$\alpha_k = \frac{\mathbf{r}_k^T (\mathbf{C}^{-1})^2 \mathbf{r}_k}{\mathbf{p}_k^T \mathbf{A} \mathbf{p}_k}, \quad (11a)$$

$$\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k \mathbf{A} \mathbf{p}_k, \quad (11b)$$

$$\beta_k = \frac{\mathbf{r}_{k+1}^T (\mathbf{C}^{-1})^2 \mathbf{r}_{k+1}}{\mathbf{r}_k^T (\mathbf{C}^{-1})^2 \mathbf{r}_k}, \quad (11c)$$

$$\mathbf{p}_{k+1} = (\mathbf{C}^{-1})^2 \mathbf{r}_{k+1} + \beta_k \mathbf{p}_k, \quad (11d)$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k. \quad (11e)$$

W powyższych równaniach rola macierzy C sprowadza się do obliczenia — *jeden raz w każdym kroku iteracji* — wyrażenia $(C^{-1})^2 r_k$, co, jak wiadomo, robi się rozwiązując odpowiedni układ równań. Zdefiniujmy

$$M = C^2. \quad (12)$$

Macierz M należy rzecz jasna dobrać tak, aby równanie $Mz = r$ można było szybko rozwiązać.

Ostatecznie otrzymujemy następujący algorytm:

$$\begin{aligned} & \mathbf{r}_1 = \mathbf{b} - \mathbf{A}\mathbf{x}_1 \\ & \text{rozwiąż } \mathbf{M}\mathbf{z}_1 = \mathbf{r}_1 \\ & \mathbf{p}_1 = \mathbf{z}_1 \\ & \textbf{while } \|\mathbf{r}_k\| > \varepsilon \\ & \quad \alpha_k = \frac{\mathbf{r}_k^T \mathbf{z}_k}{\mathbf{p}_k^T \mathbf{A} \mathbf{p}_k} \\ & \quad \mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k \mathbf{A} \mathbf{p}_k \\ & \quad \text{rozwiąż } \mathbf{M}\mathbf{z}_{k+1} = \mathbf{r}_{k+1} \\ & \quad \beta_k = \frac{\mathbf{r}_{k+1}^T \mathbf{z}_{k+1}}{\mathbf{r}_k^T \mathbf{z}_k} \\ & \quad \mathbf{p}_{k+1} = \mathbf{z}_{k+1} + \beta_k \mathbf{p}_k \\ & \quad \mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k \\ & \textbf{end} \end{aligned} \tag{13}$$

Incomplete Cholesky preconditioner

Niech rozkład QR macierzy C ma postać $C = QH^T$, gdzie Q jest macierzą ortogonalną, H^T jest macierzą trójkątną górną. Zauważmy, że

$$M = C^2 = C^T C = (QH^T)^T QH^T = HQ^T QH^T = HH^T, \quad (14)$$

a więc macierz H jest czynnikiem Cholesky'ego macierzy M . Niech rozkład Cholesky'ego macierzy A ma postać $A = GG^T$. *Przypuśćmy, iż $H \simeq G$.*

Wówczas

$$\begin{aligned}\widetilde{\mathbf{A}} &= \mathbf{C}^{-1} \mathbf{A} \mathbf{C}^{-1} = (\mathbf{C}^T)^{-1} \mathbf{A} \mathbf{C}^{-1} = \left((\mathbf{Q} \mathbf{H}^T)^T \right)^{-1} \mathbf{A} (\mathbf{Q} \mathbf{H}^T)^{-1} = \\ &= (\mathbf{H} \mathbf{Q}^T)^{-1} \mathbf{A} (\mathbf{H}^T)^{-1} \mathbf{Q}^T = \mathbf{Q} \underbrace{\mathbf{H}^{-1} \mathbf{G}}_{\simeq \mathbb{I}} \underbrace{\mathbf{G}^T (\mathbf{H}^T)^{-1}}_{\simeq \mathbb{I}} \mathbf{Q}^T \simeq \mathbf{Q} \mathbf{Q}^T = \mathbb{I}.\end{aligned}\tag{15}$$

Ponieważ $\widetilde{\mathbf{A}} \simeq \mathbb{I}$, współczynnik uwarunkowania tej macierzy powinien być bliski jedności.

Niepełny rozkład Cholesky'ego — algorytm w wersji *GAXPY*

```
for     $k = 1:N$ 
     $H_{kk} = A_{kk}$ 
    for     $j = 1:k-1$ 
         $H_{kk} = H_{kk} - H_{kj}^2$ 
    end
     $H_{kk} = \sqrt{H_{kk}}$ 
    for     $l = k+1:N$ 
         $H_{lk} = A_{lk}$ 
        if     $A_{lk} \neq 0$ 
            for     $j = 1:k-1$ 
                 $H_{lk} = H_{lk} - H_{lj}H_{kj}$ 
            end
             $H_{lk} = H_{lk}/H_{kk}$ 
        endif
    end
end
end
```

Uwagi

- Ponieważ A jest rzadka, powyższy algorytm na obliczanie przybliżonego czynnika Cholesky'ego wykonuje się szybko. Wykonuje się go *tylko raz*.
- Równanie $Mz = r$ rozwiązuje się szybko, gdyż znamy czynnik Cholesky'ego $M = HH^T$.
- Obliczone H jest rzadkie, a zatem równanie $Mz = r$ rozwiązuje się szczególnie szybko.
- Mamy nadzieję, że macierz \tilde{A} ma współczynnik uwarunkowania bliski jedności, a zatem nie potrzeba wielu iteracji (13).

Przykład — macierz pasmowa z pustymi diagonalami

Rozważmy macierz o następującej strukturze:

$$\mathbf{A} = \begin{bmatrix} a_1 & 0 & b_3 & 0 & c_5 & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & a_2 & 0 & b_4 & 0 & c_6 & 0 & 0 & 0 & 0 & \dots \\ b_3 & 0 & a_3 & 0 & b_5 & 0 & c_7 & 0 & 0 & 0 & \dots \\ 0 & b_4 & 0 & a_4 & 0 & b_6 & 0 & c_8 & 0 & 0 & \dots \\ c_5 & 0 & b_5 & 0 & a_5 & 0 & b_7 & 0 & c_9 & 0 & \dots \\ 0 & c_6 & 0 & b_6 & 0 & a_6 & 0 & b_8 & 0 & c_{10} & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \end{bmatrix} \quad (16)$$

Macierz ta jest symetryczna, zakładamy też, że jest dodatnio określona.

Niepełny czynnik Cholesky’ego macierzy (16) ma postać

$$\mathbf{H} = \begin{bmatrix} p_1 & & & & & & \\ 0 & p_2 & & & & & \\ q_3 & 0 & p_3 & & & & \\ 0 & q_4 & 0 & p_4 & & & \\ r_5 & 0 & q_5 & 0 & p_5 & & \\ 0 & r_6 & 0 & q_6 & 0 & p_6 & \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \end{bmatrix} \quad (17)$$

(W *pełnym* czynniku Cholesky’ego macierzy (16) zera leżące w (17) *po-między* diagonalą “*p*” a diagonalną “*r*” znikłyby — w ogólności mogłyby tam znajdować się jakieś niezerowe liczby.)

Zgodnie z podanym algorytmem, elementy ciągów $\{p_k\}$, $\{q_k\}$, $\{r_k\}$ wyliczamy z następujących wzorów:

$$\begin{array}{lll}
 p_1 & = & \sqrt{a_1}, \\
 q_3 & = & b_3/p_1, \\
 r_5 & = & c_5/p_1, \\
 p_3 & = & \sqrt{a_3 - q_3^2}, \\
 q_5 & = & (b_5 - r_5 q_3)/p_3, \\
 r_7 & = & c_7/p_3, \\
 p_5 & = & \sqrt{a_5 - q_5^2 - r_5^2}, \\
 q_7 & = & (b_7 - r_7 q_5)/p_5, \\
 r_9 & = & c_9/p_5, \\
 p_7 & = & \sqrt{a_7 - q_7^2 - r_7^2}, \\
 q_9 & = & (b_9 - r_9 q_7)/p_7, \\
 r_{11} & = & c_{11}/p_7, \\
 \dots & \dots & \dots
 \end{array}
 \qquad
 \begin{array}{lll}
 p_2 & = & \sqrt{a_2}, \\
 q_4 & = & b_4/p_2, \\
 r_6 & = & c_6/p_2, \\
 p_4 & = & \sqrt{a_4 - q_4^2}, \\
 q_6 & = & (b_6 - r_6 q_4)/p_4, \\
 r_8 & = & c_8/p_4, \\
 p_6 & = & \sqrt{a_6 - q_6^2 - r_6^2}, \\
 q_8 & = & (b_8 - r_8 q_6)/p_6, \\
 r_{10} & = & c_{10}/p_6, \\
 p_8 & = & \sqrt{a_8 - q_8^2 - r_8^2}, \\
 q_{10} & = & (b_{10} - r_{10} q_8)/p_8, \\
 r_{12} & = & c_{12}/p_8, \\
 \dots & \dots & \dots
 \end{array}$$

Macierze niesymetryczne

Jeżeli w równaniu

$$\mathbf{Ax} = \mathbf{b} \quad (18)$$

macierz \mathbf{A} nie jest symetryczna i dodatnio określona, sytuacja się komplikuje. Zakładając, że $\det \mathbf{A} \neq 0$, równanie (18) możemy “zsymetryzować” na dwa sposoby.

CGNR:

$$\mathbf{A}^T \mathbf{Ax} = \mathbf{A}^T \mathbf{b}, \quad (19)$$

lub CGNE:

$$\mathbf{AA}^T \mathbf{y} = \mathbf{b}, \quad (20a)$$

$$\mathbf{x} = \mathbf{A}^T \mathbf{y}. \quad (20b)$$

Do dwu powyższych równań formalnie rzecz biorąc *można* używać metody gradientów sprzężonych. Trzeba jednak pamiętać, że nawet jeśli macierz A jest rzadka, macierze $A^T A$, AA^T nie muszą być rzadkie, a co gorsza, ich współczynnik uwarunkowania jest kwadratem współczynnika uwarunkowania macierzy wyjściowej.

Alternatywnie, zamiast “symetryzować” macierz, można zmodyfikować algorytm, tak aby zamiast dwu, generował on *cztery* ciągi wektorów. Należy jednak pamiętać, że dla wielu typów macierzy taki algorytm bywa bardzo wolno zbieżny, a niekiedy nawet dochodzi do kompletnej stagnacji przed uzyskaniem rozwiązania:

Metoda gradientów bi-sprzężonych (*Bi-Conjugate Gradients*, Bi-CG)

$$\begin{aligned} & \mathbf{r}_1 = \mathbf{b} - \mathbf{A}\mathbf{x}_1, \mathbf{p}_1 = \mathbf{r}_1, \bar{\mathbf{r}}_1 \neq 0 \text{ dowolny}, \bar{\mathbf{p}}_1 = \bar{\mathbf{r}}_1 \\ & \textbf{while } \|\mathbf{r}_k\| > \varepsilon \\ & \quad \alpha_k = \frac{\bar{\mathbf{r}}_k^T \mathbf{r}_k}{\bar{\mathbf{p}}_k^T \mathbf{A} \mathbf{p}_k} \\ & \quad \mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k \mathbf{A} \mathbf{p}_k \\ & \quad \bar{\mathbf{r}}_{k+1} = \bar{\mathbf{r}}_k - \alpha_k \mathbf{A}^T \bar{\mathbf{p}}_k \\ & \quad \beta_k = \frac{\bar{\mathbf{r}}_{k+1}^T \mathbf{r}_{k+1}}{\bar{\mathbf{r}}_k^T \mathbf{r}_k} \\ & \quad \mathbf{p}_{k+1} = \mathbf{r}_{k+1} + \beta_k \mathbf{p}_k \\ & \quad \bar{\mathbf{p}}_{k+1} = \bar{\mathbf{r}}_{k+1} + \beta_k \bar{\mathbf{p}}_k \\ & \quad \mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k \\ & \textbf{end} \end{aligned} \tag{21}$$

Wektory wygenerowane w algorytmie (21) spełniają następujące relacje:

$$\bar{\mathbf{r}}_i^T \mathbf{r}_j = \mathbf{r}_i^T \bar{\mathbf{r}}_j = 0, \quad i > j, \quad (22a)$$

$$\bar{\mathbf{r}}_i^T \mathbf{p}_j = \mathbf{r}_i^T \bar{\mathbf{p}}_j = 0, \quad i > j, \quad (22b)$$

$$\bar{\mathbf{p}}_i^T \mathbf{A} \mathbf{p}_j = \mathbf{p}_i^T \mathbf{A}^T \bar{\mathbf{p}}_j = 0, \quad i > j. \quad (22c)$$

Jeżeli w algorytmie (21) weźmiemy $\bar{\mathbf{r}}_1 = \mathbf{A} \mathbf{r}_1$, we wszystkich krokach zachodzić będzie $\bar{\mathbf{r}}_k = \mathbf{A} \mathbf{r}_k$ oraz $\bar{\mathbf{p}}_k = \mathbf{A} \mathbf{p}_k$. Jest to wersja przydatna dla rozwiązywania układów równań z macierzami symetrycznymi, ale nie-określonymi dodatnio. Jest to przy okazji szczególny wariant algorytmu GMRES (*generalised minimum residual*), formalnie odpowiadającego minimalizacji funkcjonału

$$\Phi(\mathbf{x}) = \frac{1}{2} \|\mathbf{A} \mathbf{x} - \mathbf{b}\|^2. \quad (23)$$