# Rozwiązywanie układu równań liniowych metodą Gaussa-Seidla

# Zastosowanie

Procedura rozwiązuje układ równań liniowych postaci

**,**

gdzie oznacza macierz kwadratową stopnia n, a , metodą Gaussa-Seidla.

# Opis metody

Macierz układu jest przekształcana na sumę trzech macierzy, tj.

gdzie oznacza macierz trójkątną dolną, macierz diagonalną, a oznacza macierz trójkątną górną. Uwzględniając rozkład macierzy **,** układ równań można zapisać w postaci

skąd

Z powyższej zależności wynika następujący proces iteracyjny:

tj.

Jeżeli promień spektralny macierzy jest mniejszy od to proces iteracyjny jest zbieżny. Z zależności wynika, że -sze przybliżenie składowej rozwiązania jest określone wzorem

przy czym Proces iteracyjny kończy się, gdy

gdzie

a oznacza zadaną dokładność, lub gdy lub też, gdy liczba iteracji w procesie jest większa od przyjętej wartości maksymalnej.

# Wywołanie procedury

# Dane

liczba równań (równa liczbie niewiadomych),

tablica zawierająca wartości elementów macierzy (element powinien zawierać wartość , gdzie ),

tablica zawierająca wartości składowych wektora (element powinien zawierać wartość , gdzie );

maksymalna liczba iteracji w procesie ,

względna dokładność rozwiązania,

tablica zawierająca początkowe przybliżenia wartości .

**Uwaga:**

Po wykonaniu procedury wartości elementów tablicy są zmienione.

# Wyniki

tablica zawierająca rozwiązanie (element zawiera wartość ),

liczba iteracji wykonanych w procesie .

# Inne parametry

zmienna, której w procedurze przypisuje się jedną z następujących wartości:

* 1, jeżeli ,
* 2, gdy macierz jest osobliwa,
* 3, jeżeli wymagana dokładność rozwiązania nie jest osiągnięta po iteracjach,
* 0, w przeciwnym wypadku.

**Uwaga:**

Jeżeli , to po wykonaniu procedury elementy tablicy nie są zmienione. Gdy , to zawiera ostatnio obliczone przybliżenie rozwiązania.

# Typy parametrów

# Identyfikatory nielokalne

nazwa typu rekordowego postaci:

*Rekord zawiera przeciążone operatory, procedury oraz funkcje dotyczące obliczeń na arytmetyce przedziałowej. Szczegóły implementacji zawarte w pliku*

nazwa typu tablicowego o elementach typu **,** gdzie oraz

nazwa typu tablicowego o elementach typu **,** przy czym i

# Kod źródłowy

1. **procedure** iZGaussSeidel(n: Integer; **var** a: imatrix; **var** b: ivector;
2. mit: Integer; eps: interval; **var** x: ivector; **var** it, st: Integer);
3. **var**
4. i, ih, k, kh, khh, lz1, lz2: Integer;
5. max, r: interval;
6. cond: Boolean;
7. x1: ivector;
8. \_0, \_1: interval;
9. **begin**
10. \_0 := int\_read('0');
11. \_1 := int\_read('1');
12. SetLength(x1, n);
13. **if** n < 1 **then**
14. st := 1
15. **else**
16. **begin**
17. st := 0;
18. cond := true;
19. **for** k := 1 **to** n **do**
20. x1[k - 1] := \_0;
21. **repeat**
22. lz1 := 0;
23. khh := 0;
24. **for** k := 1 **to** n **do**
25. **begin**
26. lz2 := 0;
27. **if** a[k - 1, k - 1] = \_0 **then**
28. **begin**
29. kh := k;
30. **for** i := 1 **to** n **do**
31. **if** a[i - 1, k - 1] = \_0 **then**
32. lz2 := lz2 + 1;
33. **if** lz2 > lz1 **then**
34. **begin**
35. lz1 := lz2;
36. khh := kh
37. **end**
38. **end**
39. **end**;
40. **if** khh = 0 **then**
41. cond := false
42. **else**
43. **begin**
44. max := \_0;
45. **for** i := 1 **to** n **do**
46. **begin**
47. r := iabs(a[i - 1, khh - 1]);
48. **if** (r > max) **and** (x1[i - 1] = \_0) **then**
49. **begin**
50. max := r;
51. ih := i
52. **end**
53. **end**;
54. **if** max = \_0 **then**
55. st := 2
56. **else**
57. **begin**
58. **for** k := 1 **to** n **do**
59. **begin**
60. r := a[khh - 1, k - 1];
61. a[khh - 1, k - 1] := a[ih - 1, k - 1];
62. a[ih - 1, k - 1] := r
63. **end**;
64. r := b[khh - 1];
65. b[khh - 1] := b[ih - 1];
66. b[ih - 1] := r;
67. x1[khh - 1] := \_1;
68. **end**
69. **end**;
70. **until** **not** cond **or** (st = 2);
71. **if** **not** cond **then**
72. **begin**
73. it := 0;
74. **repeat**
75. it := it + 1;
76. **if** it > mit **then**
77. **begin**
78. st := 3;
79. it := it - 1
80. **end** **else** **begin**
81. **for** i := 1 **to** n **do**
82. **begin**
83. r := b[i - 1];
84. **for** k := 1 **to** i - 1 **do**
85. r := r - a[i - 1, k - 1] \* x[k - 1];
86. **for** k := i + 1 **to** n **do**
87. r := r - a[i - 1, k - 1] \* x1[k - 1];
88. x1[i - 1] := r / a[i - 1, i - 1]
89. **end**;
90. cond := true;
91. i := 0;
92. **repeat**
93. i := i + 1;
94. max := iabs(x[i - 1]);
95. r := iabs(x1[i - 1]);
96. **if** max < r **then**
97. max := r;
98. **if** NotZero(max) **then**
99. **if** iabs(x[i - 1] - x1[i - 1]) / max >= eps **then**
100. cond := false;
101. **until** (i = n) **or** **not** cond;
102. **for** i := 1 **to** n **do**
103. x[i - 1] := x1[i - 1]
104. **end**;
105. **until** (st = 3) **or** cond;
106. **end**
107. **end**
108. **end**;

# Przykłady

## Przykład I

***Dane*:**

*n=2*;

***Wyniki*:**

## Przykład II

***Dane*:**

*n=4*;

***Wyniki*:**

## Przykład III

***Dane*:**

*n=4*;

***Wyniki*:**