Rozwiązywanie układu równań liniowych o współczynnikach zespolonych

1. **Zastosowanie**

Procedura complexmatrix rozwiązuje system n równań liniowych dla zespolonych współczynników i wolnych wyrazów postaci:

$$\sum_{j=1}^{n} a_{ij} x_j = a_{i,n+1}, \qquad i = 1, 2, \dots , n,$$

dla zespolonych współczynników a_{ij} (i = 1, 2, …, n, j = 1, 2, …, n+1).

2. **Opis Metody**

Układ równań jest rozwiązywany metodą eliminacji Gaussa-Jordana z pełnym wyborem elementu podstawowego

3. **Wywołanie procedury**

complexmatrixp(n, a, x, st)

4. **Dane**

n - liczba równań układu,
a - tablica rekordów zawierających współczynniki układu i wyrazy wolne systemu
(elementy a[i,j].re powinny zawierać części rzeczywiste
współczynników, elementy a[i,j].im - części urojone współczynników, gdzie
i =1,2,...,n oraz j=1,2,...,n +1; elementy a[i,n+1].re i a[i,n+1].im powinny zawierać odpowiednio części rzeczywiste i urojone wyrazów wolnych; i=1,2,...,n).

Uwaga: Po wykonaniu procedury complexmatrixp wartości elementów tablicy a są zmienione.

5. **Wynik**

x - tablica rekordów zawierających części rzeczywiste i urojone niewiadomych.
Uwaga: Rozwiązanie ma postać
x[k].re+i*x[k].im,
gdzie i oznacza jednostkę urojoną oraz k=1,2,...,n.

6. **Inne parametry**

st - zmienna, która wewnątrz procedury complexmatrix otrzymuje wartość:
1, jeśli n<1,
2, jeśli macierz systemu jest osobliwa lub prawie osobliwa,
0, w przeciwnym razie.
Uwaga: Jeśli st ≠ 0, to elementy x nie są obliczane.

7. **Typy parametrów**

Integer: n, st
cplxvectorp: x
cplxmatrixp: a

8. Identyfikatory nielokalne

inter - identyfikator typu o postaci:

interval_arithmetic::Interval<long double> inter

przedział liczb, końce przedziału reprezentowane są przez liczby typu long double.

Complexp - identyfikator typu o postaci:

std::complex<inter> Complexp

liczba zespolona która jako współczynnik przy liczbie rzeczywistej i urojonej przyjmuje wartość inter.

cplxvectorp - identyfikator typu o postaci:

std::vector<Complexp> cplxvectorp

jednowymiarowa tablica dynamiczna o elementach typu Complexp.

cplxmatrixp - identyfikator typu o postaci:

std::vector<cplxvectorp> cplxmatrixp

dwuwymiarowa tablica dynamiczna o elementach typu Complexp.

9. Tekst procedury

```
void complexmatrixp(int n, cplxmatrixp& a, cplxvectorp& x, int& st)
{
   int ih,k,n1;
   inter d;
   Complexp aa,b,c;
   bool alb;

   if (n < 1) {
      st = 1;
   } else
   {
      st = 0;
      k = 0;
      do {
         k++;
         d = inter (0, 0);
         for (int i = k; i <= n; i++)
         {
            b.real(interval_arithmetic::IAbs(a[i - 1][k - 1].real()) + interval_arithmetic::IAbs(a[i -
1][k - 1].imag()));
            if (b.real().a > d.a && b.real().b > d.b) {
               d = b.real();
               ih = i;
            }
         }
         if (d.a == 0  && d.b == 0)
            st = 2;
         else {
            aa = a[ih - 1][k - 1];
            alb = (interval_arithmetic::IAbs(aa.real()).a < interval_arithmetic::IAbs(aa.imag()).a
&& interval_arithmetic::IAbs(aa.real()).b < interval_arithmetic::IAbs(aa.imag()).b);
```

```
            if (alb)
            {
                b.real(aa.real());
                aa.real(aa.imag());
                aa.imag(b.real());
            }
            b.real(aa.imag() / aa.real());
            aa.imag(inter (1, 1) / (b.real() * aa.imag() + aa.real()));
            aa.real(aa.imag() * b.real());
            if (!alb)
            {
                b.real(aa.real());
                aa.real(aa.imag());
                aa.imag(b.real());
            }
            a[ih - 1][k - 1] = a[k - 1][k - 1];
            n1 = n + 1;
            for (int j = k + 1; j <= n1; j++) {
                c = a[ih - 1][j - 1];
                if (d.a < (interval_arithmetic::IAbs(c.real()).a +
interval_arithmetic::IAbs(c.imag()).a) * 1e-16 && d.b < (interval_arithmetic::IAbs(c.real()).b +
interval_arithmetic::IAbs(c.imag()).b) * 1e-16)
                    st = 2;
                else
                {
                    a[ih - 1][j - 1] = a[k - 1][j - 1];
                    b.real(c.imag() * aa.imag() + c.real() * aa.real());
                    b.imag(c.imag() * aa.real() - c.real() * aa.imag());
                    a[k - 1][j - 1] = Complexp(b.real(), b.imag());
                    for (int i = k + 1; i <= n; i++) {
                        c = a[i - 1][k - 1];
                        a[i - 1][j - 1].real(a[i - 1][j - 1].real() - c.real() * b.real() + c.imag() * b.imag());
                        a[i - 1][j - 1].imag(a[i - 1][j - 1].imag() - c.real() * b.imag() - c.imag() *
b.real());
                    }
                }
            }
        }
    } while ((k != n) && (st != 2));



    if (st == 0) {
        x[n - 1] = a[n - 1][n];
        for (int i = n - 1; i >= 1; i--) {
            aa = a[i-1][n];
            for (int j = i + 1; j <= n; j++) {
                b = a[j - 1][n];
```

```
            c = a[i - 1][j - 1];
            aa.real(aa.real() - c.real() * b.real() + c.imag() * b.imag());
            aa.imag(aa.imag() - c.real() * b.imag() - c.imag() * b.real());
          }
          a[i-1][n] = aa;
          x[i-1] = aa;
        }
      }
    }
}
```

10. Przykłady

a) Układ równań:

$$2*x_1 + i*x_2 + (1+i)*x_3 = -2 + 11i,$$
$$(1+i)*x_1 + 2*x_2 + i*x_3 = 1+3i,$$
$$i*x_1 + (1+i)*x_2 + 2*x_3 = -3+5i.$$

Dane:
n = 3,
a[1, 1].re = 2, a[1, 1].im = 0, a[1, 2].re = 0, a[1, 2].im = 1, a[1, 3].re = 1, a[1, 3].im = 1,
a[1, 4].re = -2, a[1, 4].im = 11,
a[2, 1].re = 1, a[2, 1].im = 1, a[2, 2].re = 2, a[2, 2].im = 0, a[2, 3].re = 0, a[2, 3].im = 1,
a[2, 4].re = 1, a[2, 4].im = 3,
a[3, 1].re = 0, a[3, 1].im = 1, a[3, 2].re = 1, a[3, 2].im = 1, a[3, 3].re = 2, a[3, 3].im = 0,
a[3, 4].re = -3, a[3, 4].im = 5,
Wyniki:
x[1] = [-7.3183646642771550E-19, 7.0473141211557789E-19] +
[3.9999999999999999E0, 4.0000000000000001E0]*i
rozmiar przedziału części rzeczywistej: 1.437e-18 oraz urojonej 0.000e+00,
x[2] = [2.9999999999999999E0, 3.0000000000000001E0] +
[-1.6263032587282567E-19 1.6263032587282567E-19]*i
rozmiar przedziału części rzeczywistej: 0.000e+00 oraz urojonej 3.253e-19,
x[3] = [-1.0000000000000001E0 -9.9999999999999999E-1] +
[9.9999999999999999E-1 1.0000000000000001E0]*i
rozmiar przedziału części rzeczywistej: 0.000e+00 oraz urojonej 0.000e+00.

b) Układ równań:

$$([1.99; 2.01] + [-0.01; 0.02]*i)*x_1 + ([-0.02; 0.01]+[0.99; 1.02]*i)*x_2 +$$
$$([0.98; 1.01] + [0.99; 1.02]*i)*x_3 = [-2.01; -1.98] + [10.98; 11.01]*i,$$
$$([0.98; 1.01] + [0.99; 1.02]*i)*x_1 + ([1.99; 2.01] + [-0.02; 0.01]*i)*x_2 +$$
$$([-0.01; 0.02] + [0.99; 1.01]*i)*x_3 = [0.98; 1.01] + [2.98; 3.01]*i,$$
$$([-0.01; 0.01] + [0.99; 1.01]*i)*x_1 + ([0.99; 1.01]+ [0.99; 1.01]*i)*x_2 +$$
$$([1.98; 2.01] + [-0.01; 0.01]*i)*x_3 = [-3.01; -2.98] + [4.99; 5.01]*i.$$

Dane:
n = 3,
a[1, 1].re = [1.99; 2.01], a[1, 1].im = [-0.01; 0.02],

a[1, 2].re = [-0.02; 0.01], a[1, 2].im = [0.99; 1.02],
a[1, 3].re = [0.98; 1.01], a[1, 3].im = [0.99; 1.02],
a[1, 4].re = [-2.01; -1.98], a[1, 4].im = [10.98; 11.01],
a[2, 1].re = [0.98; 1.01], a[2, 1].im = [0.99; 1.02],
a[2, 2].re = [1.99; 2.01], a[2, 2].im = [-0.02; 0.01],
a[2, 3].re = [-0.01; 0.02], a[2, 3].im = [0.99; 1.01],
a[2, 4].re = [0.98; 1.01], a[2, 4].im = [2.98; 3.01],
a[3, 1].re = [-0.01; 0.01], a[3, 1].im = [0.99; 1.01],
a[3, 2].re = [0.99; 1.01], a[3, 2].im = [0.99; 1.01],
a[3, 3].re = [1.98; 2.01], a[3, 3].im = [-0.01; 0.01],
a[3, 4].re = [-3.01; -2.98], a[3, 4].im = [4.99; 5.01]

Wyniki:
x[1] = [-6.2835833506801867E-1, 7.1900506069988910E-1] + [3.3634896232079548E0, 4.6696346475803534E0]*i
rozmiar przedziału części rzeczywistej: 1.347e+00 oraz urojonej 1.306e+00
x[2] = [2.7532156197384646E0, 3.2564502359557177E0] + [-2.5374383714664298E-1, 2.6258883668515133E-1]*i
rozmiar przedziału części rzeczywistej: 5.032e-01 oraz urojonej 5.163e-01
x[3] = [-1.4579393657592668E0, -6.1734746096758373E-1] + [5.6095983281667480E-1, 1.4076231393725849E0]
rozmiar przedziału części rzeczywistej: 8.406e-01 oraz urojonej 8.467e-01

c) Układ równań:

$$(1+i)*x_1 + (1+i)*x_2 = 1 + i,$$
$$(1+i)*x_1 + (1+i)*x_2 = 1 + i.$$

Dane:
n = 3,
a[1, 1].re = 1, a[1, 1].im = 1, a[1, 2].re = 1, a[1, 2].im = 1, a[1, 3].re = 1, a[1, 3].im = 1,
a[2, 1].re = 1, a[2, 1].im = 1, a[2, 2].re = 1, a[2, 2].im = 1, a[2, 3].re = 1, a[2, 3].im = 1,
a[3, 1].re = 1, a[3, 1].im = 1, a[3, 2].re = 1, a[3, 2].im = 1, a[3, 3].re = 1, a[3, 3].im = 1,
Wyniki:
st = 2,