Promotor **dr hab. inż. Zbigniew Mitura**	Imię i nazwisko **Witold Kęsek**	Data **28.09.2020**
Sprawozdanie z projektu w ramach praktyk studenckich		

1. **Cel projektu**

   Celem projektu było napisanie programu który służy do obliczania natężeń RHEED dla powierzchni arsenku galu (GaAs).

2. **Metodologia**

   Aby obliczyć natężenie RHEED dla danej powierzchni będziemy posiłkować się metodami numerycznymi. Najpierw musimy obliczyć amplitudę wiązki elektronów przy pomocy równania:

$$r_w = \left\{r_{w-1}\left[1 + h\left(ik + \frac{v_w}{2ik}\right) + \frac{h^2}{2}(-k^2 + v_w)\right] + h\frac{v_w}{2ik}\right\} / \left\{1 - h\left(ik + \frac{v_w}{2ik}\right) + \frac{h^2}{2}(-k^2 + v_w) - r_{w-1}h\frac{v_w}{2ik}\right\}, \quad (6)$$

   Przy założeniach, że:

$$r_0 = 0.$$

   Oraz, że

$$r = r_W.$$

   Oraz:

$$v_w = v\left(\frac{z_{w-1}+z_w}{2}\right).$$

   A także skorzystać z wzoru na potencjał vw, który będzie sumą potenjałów części dwuwymiarowych siatek atomów równoległych do badanych powierzchni. Wzór na j-tą część:

$$v_j(z) = -\Theta_j\left(1 + \frac{|q_e|U}{m_0c^2}\right) \times (\gamma_{Re} + i\gamma_{Im})\frac{8\pi^{3/2}}{\Omega} \times \left\{\sum_l \frac{a_l^{(j)}}{\left(b_l^{(j)}+B^{(j)}\right)^{1/2}} \exp\left[-\frac{4\pi^2}{b_l^{(j)}+B^{(j)}}\left(z - z_j\right)^2\right]\right\},$$

   W naszym doświadczeniu zakładamy, że mamy na przemian warstwy Arsenu oraz Galu, a odległość między kolejnymi warstwami wynosiła 5,65/4 Å.

3. **Analiza kodu**

Główna funkcja programu (tzw. “main”) to funkcja obliczająca nasze “r”.
Pierwsza pętla to wyliczenie “pokrycia (0-0.05…..-0.95-1) oraz wyliczenie wszystkich **v(z)** przy pomocy funkcji napisanej w innym pliku (opisanej niżej):

```
double covlc=covli+f*(covlf-covli)/ncov;
ComplexNumber []vz=v(covlc);
```

Później tworzona jest nazwa i otwierany plik do którego będą zapisywane wyniki dla danego pokrycia:

```
double angc=j*(angf-angi)/(nang-1);
double k = Wyliczenia.KdlaTety( teta: angc/180*Math.PI);
```

Kolejna pętla jest zagnieżdżona w poprzedniej i każde kolejne wykonanie pętli to delikatnie inny kąt oraz inne “**k**” wyliczane przy pomocy funkcji napisanej w innym pliku (opisanej niżej):

```
double angc=j*(angf-angi)/(nang-1);
double k = Wyliczenia.KdlaTety( teta: angc/180*Math.PI);
```

Jeśli **k^2** wychodzi mniejsze od **0.01** to wynik wynosi **-1**:

```
if(k*k<0.01){
    out.println(df.format( number: -1));
```

Jeśli nie to wyliczane są kolejne elementy przy pomocy wzoru:

$$r_w = \left\{r_{w-1}\left[1 + h\left(ik + \frac{v_w}{2ik}\right) + \frac{h^2}{2}(-k^2 + v_w)\right] + h\frac{v_w}{2ik}\right\} / \left\{1 - h\left(ik + \frac{v_w}{2ik}\right) + \frac{h^2}{2}(-k^2 + v_w) - r_{w-1}h\frac{v_w}{2ik}\right\}, \quad (6)$$

W środku pętli obliczana jest górna część ułamka:

```
ComplexNumber top1 = new ComplexNumber( real: 0, k);

ComplexNumber top2 = new ComplexNumber( real: vz[g].imaginary / (2 * k), imaginary: -vz[g].real / (2 * k));

ComplexNumber top3 = new ComplexNumber( real: 1 + h * (top1.real + top2.real), imaginary: h * (top1.imaginary + top2.imaginary));

ComplexNumber top4 = new ComplexNumber( real: (-k * k + vz[g].real) * h * h * 0.5, imaginary: 0.5 * h * h * vz[g].imaginary);

top3.add(top4);

ComplexNumber top_result = new ComplexNumber(r_prev.real,r_prev.imaginary);

top_result.multiply(top3);

top_result.add(new ComplexNumber( real: h * top2.real, imaginary: h * top2.imaginary));
```

(**ComplexNumber** to klasa napisana przeze mnie służąca do przechowywania liczb zespolonych oraz obliczeń z nimi związanych).

Następnie obliczana jest dolna część ułamka i dzielone jest jedno przez drugie:

```
ComplexNumber bot_result = new ComplexNumber( real: 1 - h * (top1.real + top2.real),  imaginary: 0 - h * (top1.imaginary + top2.imaginary));

ComplexNumber fd = new ComplexNumber( real: r_prev.real * top2.real - r_prev.imaginary * top2.imaginary,  imaginary: r_prev.imaginary * top2.real + r_prev.real * top2.imaginary);

bot_result.substract(new ComplexNumber( real: h * fd.real,  imaginary: h * fd.imaginary));

bot_result.add(top4);

top_result.divide(bot_result);

r_prev = top_result;
```

W drugim pliku (Wyliczenia) znajduje się prosta funkcja obliczająca **k** dla podanej **Θ**:

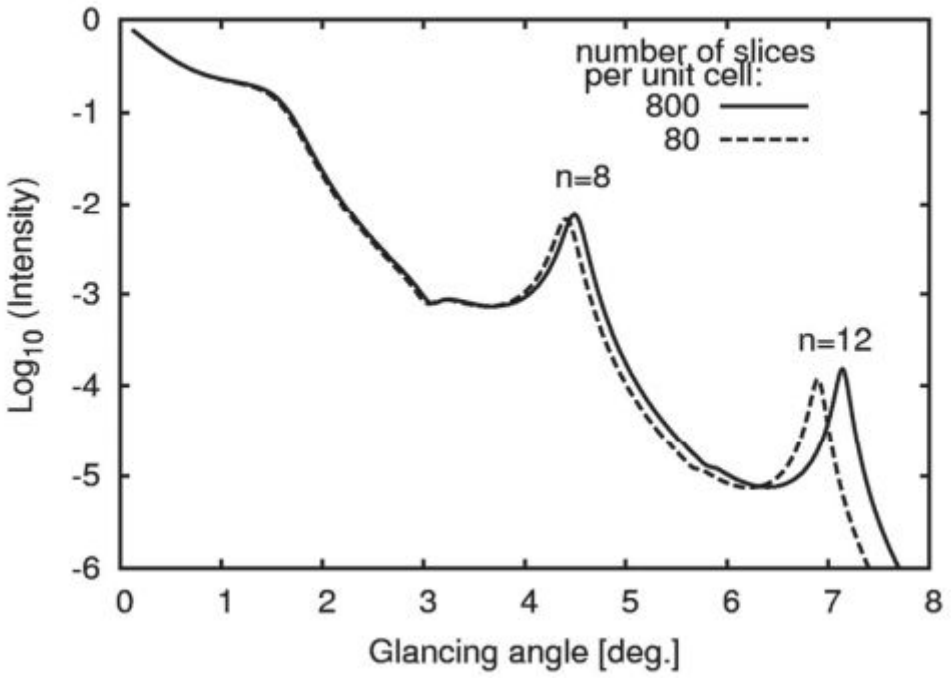
```
public static double KdlaTety(double teta){
    double K2=((2*m0*qe*2*Math.PI*2*Math.PI)/2*(Planck*Planck))*0.01*(1+U/(2*c))*U;
    return Math.sqrt(K2)*Math.sin(teta);
}
```

Oraz funkcja **v(covt)**, która w zależności od podanego pokrycia **covt** oblicza potencjał **vz** na całej wysokości (**22000** punktów pomiarowych między v początkowym i końcowym). Główną częścią tej funkcji są dwie zagnieżdżone pętle, z których wewnętrzna wykonuje działania związane z współczynnikami a i b danego pierwiastka, a zewnętrzna mnoży je przez pozostałe współczynniki.

```
    for(int k=0;k<nvcr;k++){
        double zcurr=zb+(k+1)*zstep;
        double vcurr=0;
        for(int j=0;j<natl;j++){
            double z_diff=(zcurr-zpl[j])*(zcurr-zpl[j]);
            for(int l=0;l<5;l++) {
                if (j % 2 == 0) {
                    a_tmp[l] = a_As[l];
                    b_tmp[l] = b_As[l];
                } else {
                    a_tmp[l] = a_Ga[l];
                    b_tmp[l] = b_Ga[l];
                }
                double suma_a_b_1 = ((-4 * Math.PI * Math.PI) / b_tmp[l]) * z_diff;
                if (suma_a_b_1 < -15)
                    continue;
                double suma_a_b_2 = (a_tmp[l] / Math.sqrt(b_tmp[l])) * Math.exp(suma_a_b_1);
                vcurr += covl[j]*suma_a_b_2;
            }
        }
        double conf=-8*Math.PI*Math.sqrt(Math.PI)/omega;
        vcurr*=conf;
        ComplexNumber tmp=new ComplexNumber( real: YRe*vcurr, imaginary: Ylm*vcurr);
        tmpv[k]=tmp;
    }
    return tmpv;
```

4. Porównanie wyników

Wyniki uzyskane w publikacji:



Wyniki uzyskane w programie napisanym przeze mnie: