Paweł Kocimski
Interpolacja

Zad.1 Wielomiany interpolujące

Definicja funkcji $f(x) = \frac{1}{1 + x^2}$

```
def function(x):
    x =np.float32(x)
    return np.float32(1/(1+x**2))
```

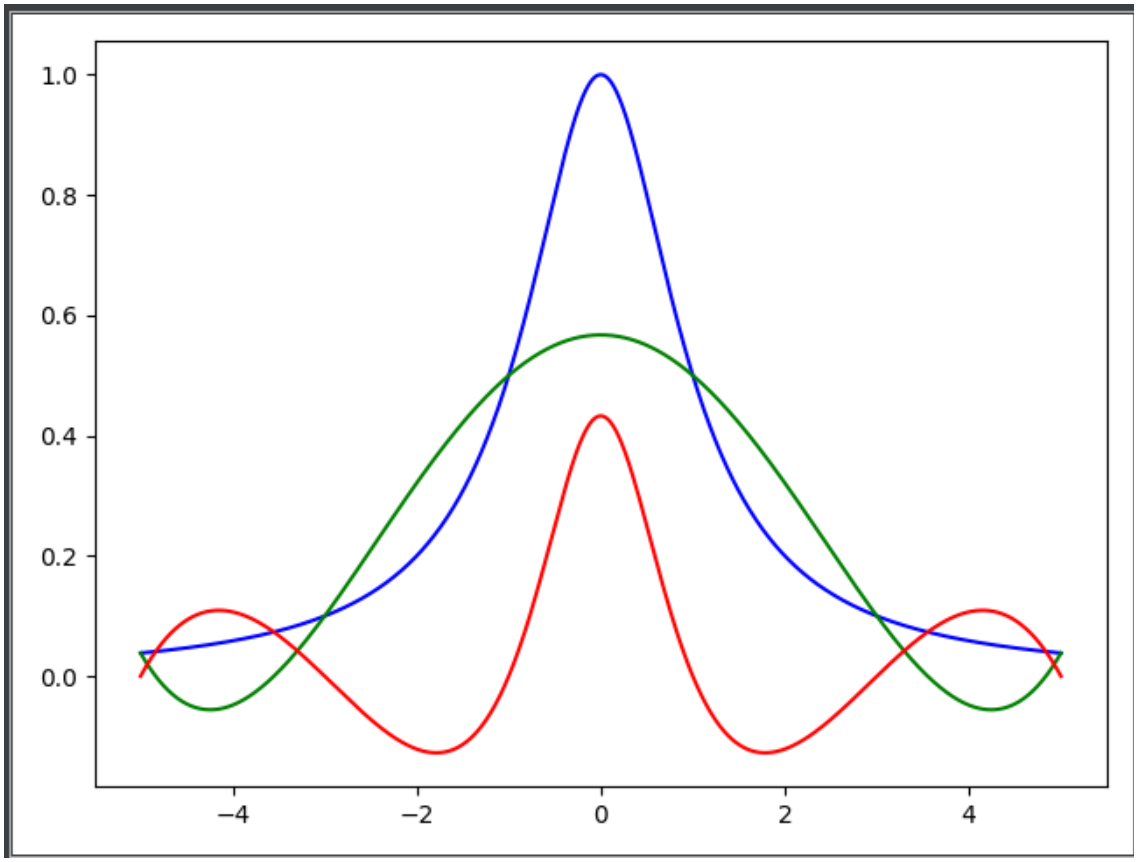
Funkcja wyliczająca wielomian interpolacyjny pn z n + 1
węzłami równoodległymi w przedziale [−5, 5] dla n =5,10,15. W tej metodzie użyto macierz vandermonde'a.

```
def vandermond_matrix(start, end, n):
    d = np.linspace(start, end, n + 1)
    Y = []
    d = np.array(d)
    vander = []
    original_x = np.linspace(-5, 5, 1000)
    original_y = []
    #make vandermonde'a matrix the same
    # vander = np.vander(d, n+1, increasing=True)
    for point in d:
        x = []
        for i in range(n+1):
                x.append(np.float32(point**i))
        vander.append(x)

    #make matrix of value in n+1 points
    for i in d:
        Y.append(np.float32(function(i)))
    A = np.linalg.solve(vander, Y)

    result = []
    delta = []
    for point_x in original_x:
        sum = 0
        for (power, pol_factor) in enumerate(A):
            sum += pol_factor*point_x**power
        func_value = np.float32(function(point_x))
        result.append(sum)
        original_y.append(func_value)
        diff = func_value - sum
        delta.append(diff)

    plt.plot(original_x, original_y, 'b', markersize=1)
    plt.plot(original_x, result, "g", markersize=1)
    plt.plot(original_x, delta, "r", markersize=1)
    plt.show()
```

niebieska linia – originalny wykres
zielona linia – wielomian interpolacyjny
czerwona linia – różnica między wartością funkjci, a wielomianem interpolacyjnym
reszta interpolacji f(x) − pn(x)

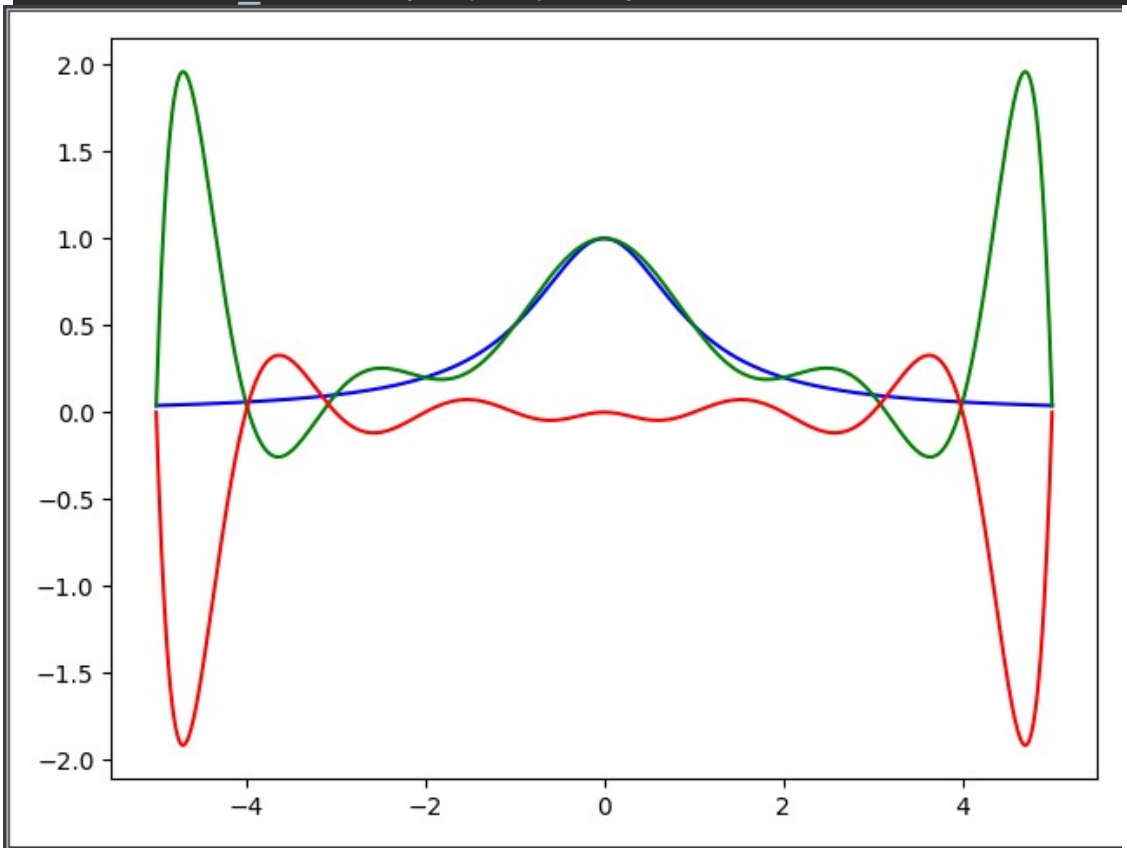
n = 5

```
vandermond_matrix(-5, 5, 5)
```

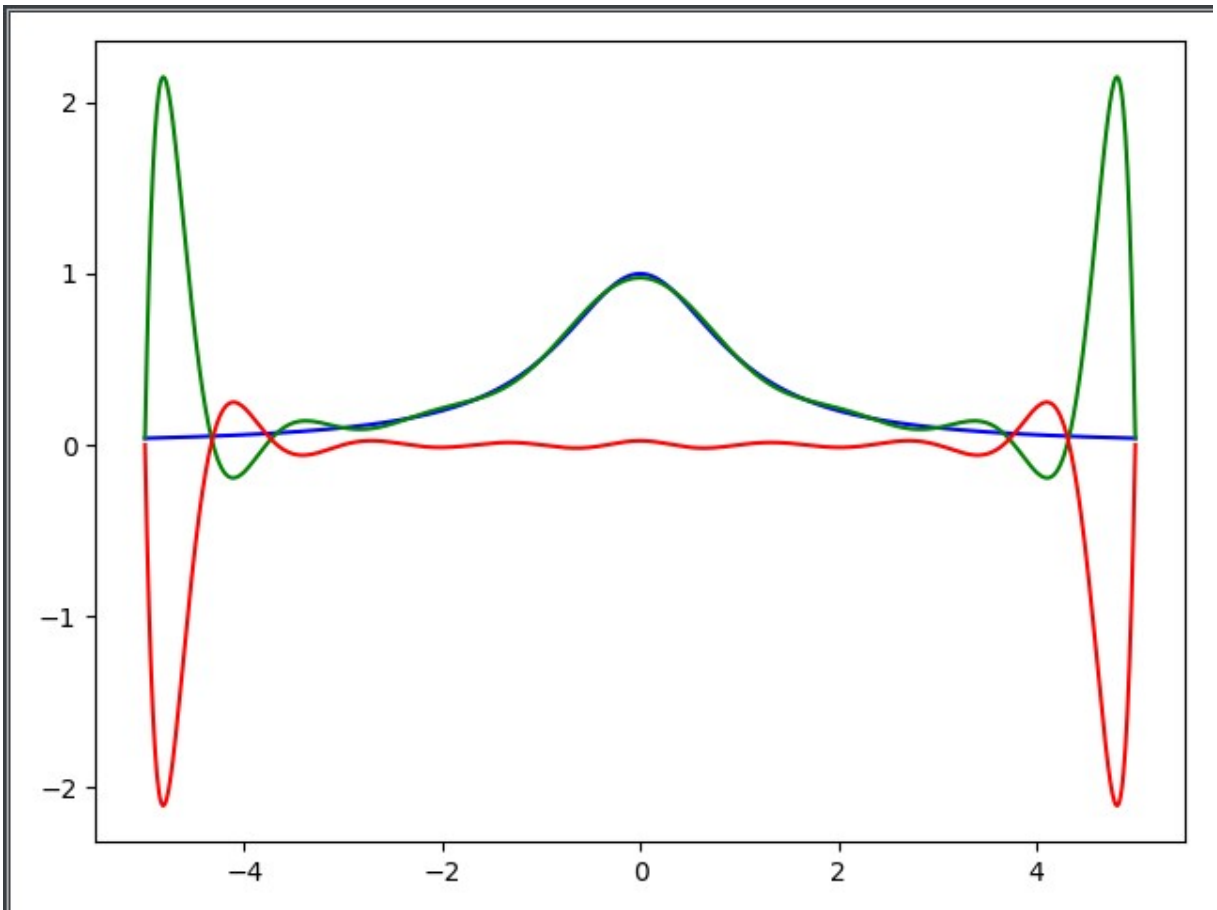


n = 10

```
vandermond_matrix(-5, 5, 10)
```



n = 15



Dla równoodległych punktów można zaobserwować na końcach przedziałów efekt Rungego.

Efekt Rungego – pogorszenie jakości interpolacji wielomianowej, mimo zwiększenia liczby jej węzłów. Początkowo ze wzrostem liczby węzłów *n* przybliżenie poprawia się, jednak po dalszym wzroście *n,* zaczyna się pogarszać, co jest szczególnie widoczne na końcach przedziałów

Takie zachowanie się wielomianu interpolującego jest zjawiskiem typowym dla interpolacji za pomocą wielomianów wysokich stopni przy stałych odległościach węzłów. Występuje ono również, jeśli interpolowana funkcja jest nieciągła albo odbiega znacząco od funkcji gładkiej.

Ponieważ zgodnie z twierdzeniem Weierstrassa istnieje ciąg interpolujących wielomianów coraz wyższych stopni, które przybliżają jednostajnie funkcje ciągłą, można uważać to za paradoks, iż efekt Rungego ma dokładnie odwrotny wynik. Jest to spowodowane nałożeniem warunku na równoodległość węzłów.

Aby uniknąć tego efektu, stosuje się interpolację z węzłami coraz gęściej upakowanymi na krańcach przedziału interpolacji. Np. węzłami interpolacji *n*-punktowej wielomianowej powinny być miejsca zerowe wielomianu Czebyszewa *n*-tego stopnia.

Zad. 2 Wielomiany interpolujące Węzły Czebyszewa

Węzły Czebyszewa - są to pierwiastki wielomianu Czebyszewa pierwszego rodzaju stopnia Dla węzłów w dowolnym przedziale . można zastosować przekształcenie afiniczne:

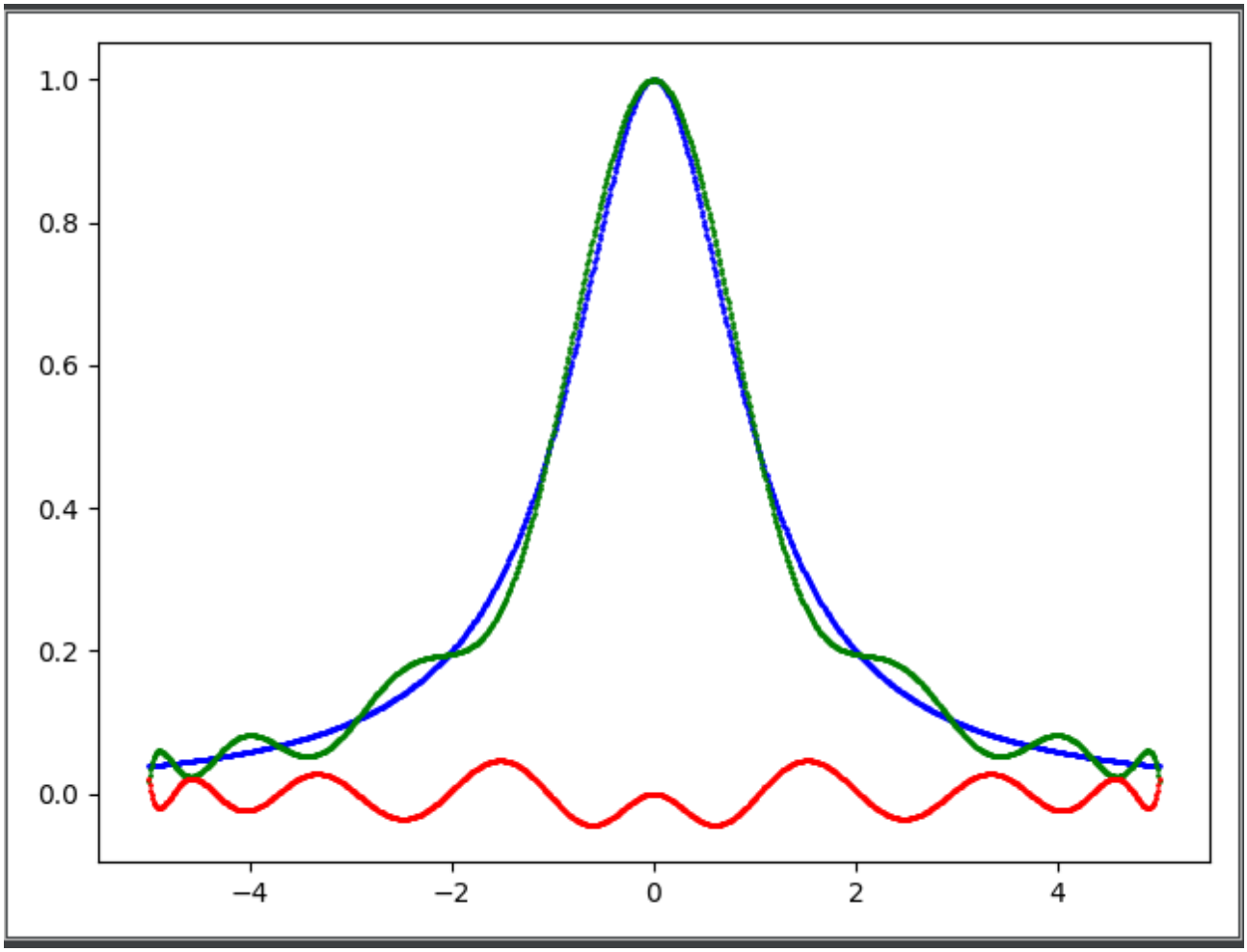
$$x_k = \frac{1}{2}(a+b) + \frac{1}{2}(b-a)\cos\left(\frac{2k-1}{2n}\pi\right), \quad k = 1, \ldots, n.$$

```
def vandermond_matrix_czebyszew(start, end, n):
    czybyszew = []
    for k in range(1, n + 1):
        point_x = 0.5 * (start + end) + 0.5 * (end - start) *
        math.cos(((2 * k - 1) / (2 * n)) * math.pi)
        czybyszew.append(point_x)

    y = []
    x = []

    for (i, c) in enumerate(czybyszew):
        x.append(c)
        y.append(function(c))

    x = np.array(x)
    y = np.array(y)
    van = np.vander(x, n, increasing=True)
    A = np.linalg.solve(van, y)

    result = []
    original_y = []
    delta = []
    original_x = np.linspace(-5, 5, 1000)

    for point_x in original_x:
        sum = 0
        for (power, pol_factor) in enumerate(A):
            sum += pol_factor*point_x**power
        func_value = np.float32(function(point_x))
        result.append(sum)
        original_y.append(func_value)
        diff = func_value - sum
        delta.append(diff)

    plt.plot(original_x, original_y, 'bo', markersize=1)
    plt.plot(original_x, result, 'go', markersize=1)
    plt.plot(original_x, delta, 'ro', markersize=1)
    plt.show()
```

Są często używane jako węzły w interpolacji wielomianowej, ponieważ wynikowy wielomian interpolacyjny minimalizuje efekt Rungego, czyli duże oscylacje wielomianu interpolacyjnego przy krańcach przedziału. Fakt, że miejsca zerowe wielomianów Czebyszewa zagęszczają się ku krańcom przedziału, pozwala lepiej związać wielomian zapobiegając naturalnym dla wielomianów wysokiego rzędu oscylacjom.

niebieska linia – originalny wykres
zielona linia – wielomian interpolacyjny
czerwona linia – różnica między wartością funkjci, a wielomianem interpolacyjnym
reszta interpolacji f(x) − pn(x)

```
vandermond_matrix_czebyszew(-5, 5, 15)
```



Zad. 3 Interpolacja krzywych funkcjami sklejanymi

Interpolacja funkcjami sklejanymi – metoda numeryczna polegająca na przybliżaniu nieznanej funkcji wielomianami niskiego stopnia.

Dla przedziału $[a, b]$, zawierającego wszystkie $n + 1$ węzły interpolacji, tworzy się m przedziałów:

$t_0 \cdots t_1$

$t_1 \cdots t_2$

$\cdots$

$t_{m-1} \cdots t_m$

takich, że $a = t_0 < t_1 < \cdots < t_m = b$

i w każdym z nich interpoluje się funkcję wielomianem interpolacyjnym (najczęściej niskiego stopnia). „Połączenie" tych wielomianów ma utworzyć funkcję sklejaną.

Funkcja sklejana S jest funkcją interpolującą funkcję F, jeżeli:

$F(x_i) = S(x_i)$ dla $x_i, i \in 0, 1, \ldots, n$ są węzłami interpolacyjnymi funkcji F.

Mamy daną elipsę w postaci parametrycznej:

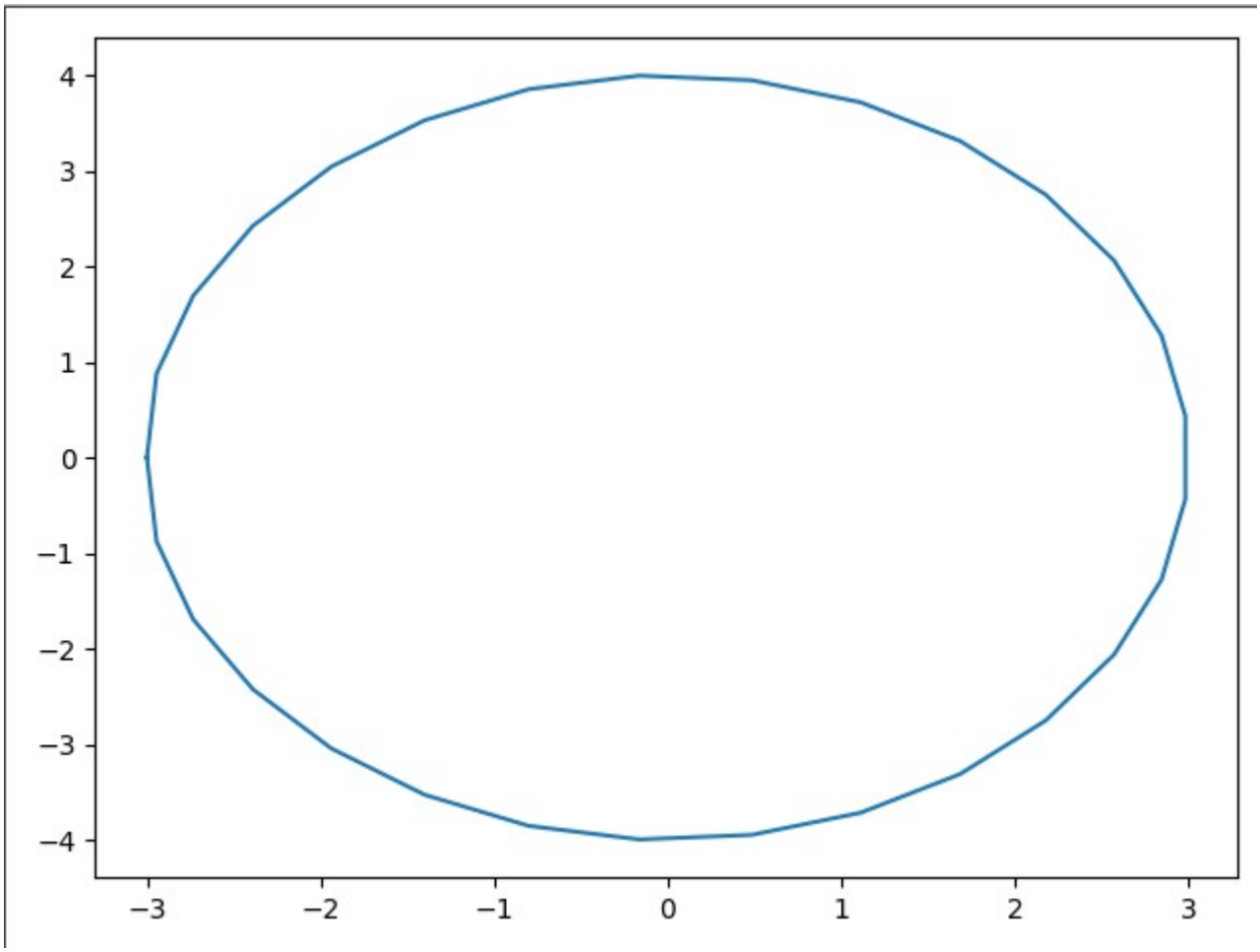
$$\begin{cases} x = a * cos(t), \\ y = b * sin(t) \end{cases}$$

t ∈ [0, 2π]

Funkcja interpolująca sześciennymi funkcjami sklejanymi dla 10 równoodległych puktów

```
def cubic_spline_ellipse():
    t_determine = np.linspace(0, 2*math.pi, 10)
    spline_x = interpolate.interp1d(t_determine, list(map(x_func,
t_determine)), kind = 'cubic')
    spline_y = interpolate.interp1d(t_determine, list(map(y_func,
t_determine)), kind = 'cubic')
    t_plot = np.linspace(0, 2*math.pi, 30)
    x_plot = []
    y_plot = []
    for t in t_plot:
        x_plot.append(spline_x(t))
        y_plot.append(spline_y(t))
    plt.plot(x_plot, y_plot)
    plt.show()
```

```
cubic_spline_ellipse()
```



Wnioski:

1) Dla węzłów równoodległych od siebie wraz ze wzrostem węzłów interpolacja wielomianowa staja się coraz to dokładniejsza.

2)Interpolacja przy pomocy węzłów będących w takiej samej odległości od siebie powoduje wzrost błędu interpolacji na końcach przedziałów(efekt Rungego). Do naprawienia tej wady interpolacji zagęszcza się węzły na końcach przedziału interpolując w węzłach Czebyszewa.

3) Interpolacja przy pomocy funkcji sklejanych kubicznych dobrze interpoluje funkcję