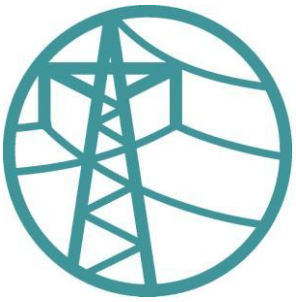


WYDZIAŁ
ELEKTROTECHNIKI
I INFORMATYKI
POLITECHNIKI RZESZOWSKIEJ

Autor pracy Krzysztof Łukasik

Sprawozdanie z zajęć laboratoryjnych

Metody FEM w robotyce

Rzeszów, 2021

Spis treści

1. Realizacja metody FEM

Metoda FEM, czyli metoda elementów skończonych (ang. finite element method) jest zaawansowaną metodą numeryczną służącą do rozwiązywania problemów brzegowych. Problem polega na wyznaczeniu funkcji, która spełnia dane równanie różniczkowe w obszarze Ω i dodatkowe warunki na jego brzegu. Wyznaczanie funkcji polega na zastosowaniu interpolacji jedno, dwu, lub trójwymiarowej na zbiorze jej węzłów, które powstają w wyniku dyskretyzacji dziedziny na tak zwane elementy skończone.

Dokładna data i autor wynalezienia metody FEM nie jest znany, jednak najwcześniejsze wzmianki o jej użyciu można znaleźć w pracach z wczesnych lat 40. XX wieku, których autorami są matematyk Alexander Hrennikoff i inżynier Richard Courant. Służyła im ona do rozwiązywania problemów związanych z elastycznością i analizą strukturalną w inżynierii lądowej.

FEM znajduje szerokie zastosowanie w fizyce, mechanice i meteorologii. Przy jej pomocy bada się wytrzymałość konstrukcji, symuluje ich odkształcania, naprężenia, a także przepływ ciepła. Metoda stosowana jest również do interpolowania wyników pomiarów wykonywanych na dyskretnym zbiorze punktów.

Jak każda metoda numerycznej aproksymacji, FEM nie jest metodą idealną, a na wielkość błędu wpływa kilka elementów:

1) błąd modelowania
2) błąd wartości współczynników
3) błąd odwzorowania obszaru
4) błąd numeryczny
5) błąd zaokrągleń

1.1. Proces rozwiązywania problemu przy pomocy FEM

1) Podzielenie danego obszaru na skończoną liczbę elementów.
2) Wyznaczenie węzłów - połączeń pomiędzy elementami, przyjmuje się, że poszczególne elementy połączone są ze sobą w skończonej liczbie punktów.
3) Określenie funkcji kształtu.
4) Przekształcenie równań różniczkowych w równania algebraiczne.
5) Obliczenie wartości współczynników, oraz wartości prawych stron przy pomocy równań FEM.
6) Wprowadzenie warunków brzegowych do macierzy współczynników i wektorów prawych stron.
7) Rozwiązanie powstałego układu równań - skutkuje otrzymaniem poszukiwanych wielkości fizycznych w węzłach.

1.2. Zastosowane oprogramowanie

Do realizacji metody FEM nada się każde narzędzie programistyczne. Najłatwiej jednak korzystać z oprogramowania które posiada wbudowane, lub łatwo implementowalne biblioteki zawierające zaawansowane narzędzia matematyczne. Szczególnie ważne w przypadku metody FEM jest rozbudowana obsługa macierzy, tablic, algebry liniowej i interpolacji. Do takiego rodzaju oprogramowania zalicza się między innymi język Python, oprogramowanie MATLAB, Scilab, czy Oktawa GNU.

Do realizacji programu został wybrany język Python w wersji 3.8. Dodatkowo zostały użyte następujące biblioteki:

1) NumPy - dodająca obsługę dużych, wielowymiarowych tablic i macierzy, a także duży zbiór funkcji matematycznych wysokiego poziomu do obsługi tych tablic

2) SciPy - używana do obliczeń naukowych i technicznych, zawiera moduły optymalizacji, algebry liniowej, integracji, interpolacji, funkcji specjalnych, FFT, przetwarzania sygnałów i obrazów, rozwiązywania ODE i innych zadań powszechnych w nauce i inżynierii

3) Matplotlib - służy do tworzenia wykresów dla języka programowania Python i jego rozszerzenia numerycznego NumPy, zawiera API "pylab" zaprojektowane tak aby było jak najbardziej podobne do MATLABa

2. Program

Program ma dość prostą architekturę, składa się z ośmiu plików, a każdy z nich jest oddzielną funkcją. Jest to zabieg zrobiony wyłącznie dla czytelności kodu, gdyż główna funkcja (main) podczas wywołania zwyczajnie importuje wszystkie inne funkcje. W kodzie nie zostały zaimplementowane żadne mechanizmy mające na celu weryfikację spójności danych. Program jest na tyle mały, że nie są one potrzebne. Również nie zostały zaimplementowane żadne mechanizmy kontroli danych wprowadzanych przez użytkownika, ponieważ zakładamy, że program będzie wykorzystywany przez osoby świadome.

2.1. Funkcja - Main

```
import numpy as np

import matplotlib.pyplot as plt

import scipy.integrate as spint

from GeometriaDefinicja import *

from AutomatycznyGeneratorGeometrii import *

from RysujGeometrie import *

from Alokacja import *

from FunkcjeBazowe import *

from Aij import *

from RysujRozwiazanie import *

if __name__ == '__main__':

    c = 0

    x_a =  0

    x_b =  1

    n = 5

    WEZLY, ELEMENTY = AutomatycznyGeneratorGeometrii(x_a,x_b,n)

    WB    = [{"ind": 1, "typ":'D', "wartosc":1},

             {"ind": n, "typ":'D', "wartosc":2}]

    RysujGeometrie(WEZLY, ELEMENTY, WB)

    A,b = Alokacja(n)

    stopien_fun_bazowych = 1

    phi, dphi = FunkcjeBazowe(stopien_fun_bazowych)

    liczbaElementow = np.shape(ELEMENTY)[0]

    for ee in np.arange(0, liczbaElementow ):
```

```
    elemRowInd = ee
    elemGlobalInd = ELEMENTY[ee,0]
    elemWezel1 = ELEMENTY[ee,1]
    elemWezel2 = ELEMENTY[ee,2]
    indGlobalneWezlow = np.array([elemWezel1, elemWezel2 ])
    x_a = WEZLY[ elemWezel1-1 ,1]
    x_b = WEZLY[ elemWezel2-1 ,1]
    Ml = np.zeros( [stopien_fun_bazowych+1, stopien_fun_bazowych+1] )
    J = (x_b-x_a)/2
    m = 0; n = 0 ;
    Ml[m,n] = J * spint.quad( Aij(dphi[m], dphi[n], c, phi[m],phi[n]), -1, 1)[0]
    m = 0; n = 1 ;
    Ml[m,n] = J * spint.quad( Aij(dphi[m], dphi[n], c, phi[m],phi[n]), -1, 1)[0]
    m = 1; n = 0 ;
    Ml[m,n] = J * spint.quad( Aij(dphi[m], dphi[n], c, phi[m],phi[n]), -1, 1)[0]
    m = 1; n = 1 ;
    Ml[m,n] = J * spint.quad( Aij(dphi[m], dphi[n], c, phi[m],phi[n]), -1, 1)[0]
    A[np.ix_(indGlobalneWezlow-1, indGlobalneWezlow-1  ) ] =  \
       A[np.ix_(indGlobalneWezlow-1, indGlobalneWezlow-1  ) ] + Ml
print(WB)
if WB[0]['typ'] == 'D':
    ind_wezla = WB[0]['ind']
    wart_war_brzeg = WB[0]['wartosc']
    iwp = ind_wezla - 1
    WZMACNIACZ = 10**14
```

```
        b[iwp] = A[iwp,iwp]*WZMACNIACZ*wart_war_brzeg
        A[iwp, iwp] = A[iwp,iwp]*WZMACNIACZ
    if WB[1]['typ'] == 'D':
        ind_wezla = WB[1]['ind']
        wart_war_brzeg = WB[1]['wartosc']
        iwp = ind_wezla - 1
        WZMACNIACZ = 10**14
        b[iwp] = A[iwp,iwp]*WZMACNIACZ*wart_war_brzeg
        A[iwp, iwp] = A[iwp,iwp]*WZMACNIACZ
    if WB[0]['typ'] == 'N':
        print('Nie zaimplementowano')
    if WB[1]['typ'] == 'N':
        print('Nie zaimplementowano')
    u = np.linalg.solve(A,b)
    RysujRozwiazanie(WEZLY, ELEMENTY, WB, u)
```

Główna funkcja programu, której zadaniem jest wywoływanie w odpowiedniej kolejności innych funkcji w taki sposób, aby uzyskać oczekiwane rozwiązanie. Analizując jej działanie można zauważyć, że najpierw importowane są bibliotek, takie jak NumPy, SciPy i Matplotlib. Następnie wykonywany jest import wszystkich funkcji z których program będzie później korzystał. Oprócz samego wywoływania innych funkcji i korzystania z danych które zwracają, w ciele funkcji main można zauważyć również deklaracje niektórych zmiennych, szczególnie takich, których wartość może być wielokrotnie zmieniana przez użytkownika. Taki zabieg ma na celu ułatwienie konfiguracji programu. Użytkownik podaje dane wejściowe w jednym miejscu, nie musi przechodzić przez każdą funkcję. Wyjątkiem jest tutaj funkcja GeometriaDefinicja, jednak w zamyśle nie będzie ona raczej używana przez użytkownika. Jej zastosowanie wymaga również manipulacji kodu w funkcji main.

2.2. Funkcja - GeometriaDefinicja

```
import numpy as np

def GeometriaDefinicja():

   NODES = np.array([[1, 0],

                     [2, 1],

                     [3, 0.5],

                     [4, 0.75]] )

   ELEMS = np.array( [[1, 1, 3],

                      [2, 4, 2],

                      [3, 3, 4]] )

   WB    = [{"ind": 1, "typ":'D', "wartosc":1},

          {"ind": 2, "typ":'D', "wartosc":2}]

   return NODES, ELEMS, WB
```

Funkcja służąca do zdefiniowania tablicy węzłów, elementów oraz warunków brzegowych w sposób ręczny. Użytkownik musi ręcznie wpisać dane w tablice *NODES*, *ELEMS* oraz *WB*. Te zmienne zostaną następnie zwrócone do funkcji main. Użycie ten funkcji wymaga odkomentowania niektórych linijek kodu i zakomentowania innych w funkcji main. Jest to narzędzie służące do testów we wczesnej fazie projektowania programu. Może okazać się również użyteczna w przypadku próby zdefiniowania nietypowych obiektów.

2.3. Funkcja - AutomatycznyGeneratorGeometrii

```
import numpy as np

def AutomatycznyGeneratorGeometrii(a,b,n):

   lp = np.arange(1,n+1)

   x = np.linspace(a,b,n) ;

   WEZLY = (np.vstack( (lp.T, x.T) )).T

   lp = np.arange(1,n)

   C1 = np.arange(1,n)

   C2 = np.arange(2,n+1)

   ELEMENTY = (np.block( [[lp], [C1], [C2] ] ) ).T

   return WEZLY, ELEMENTY
```

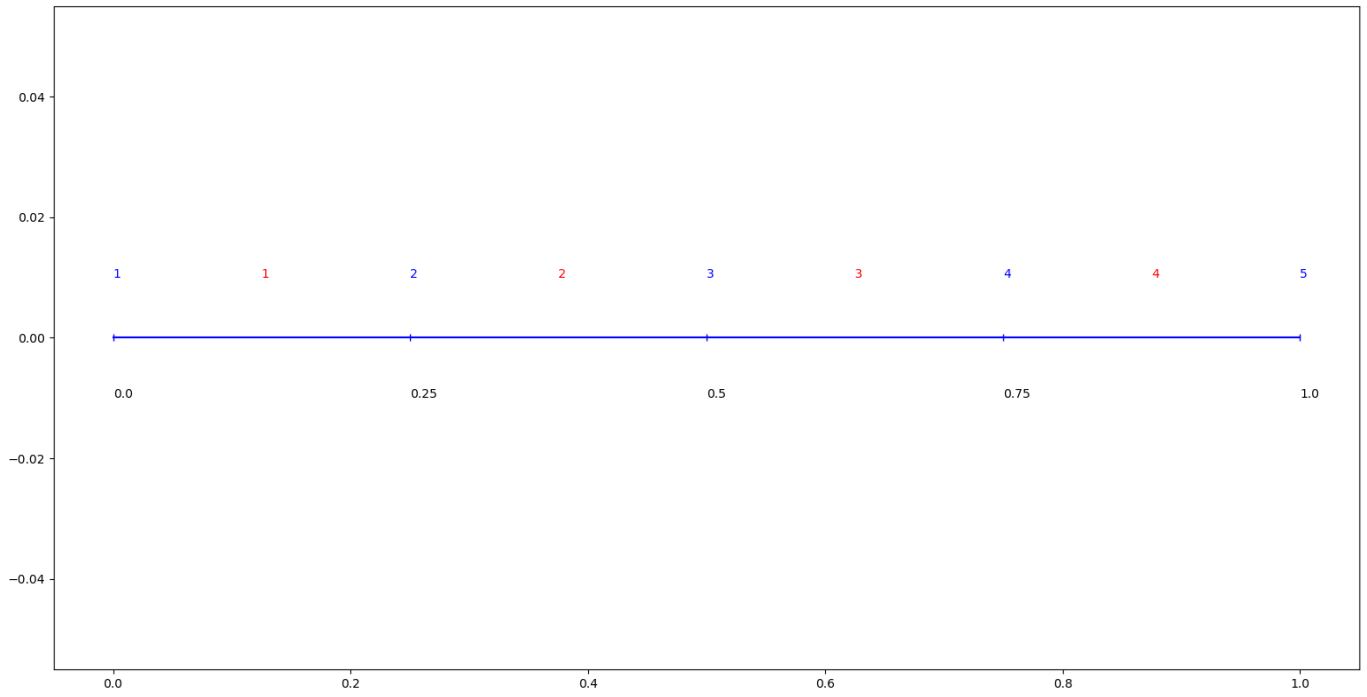
Funkcja która potrafi automatycznie wygenerować geometrię na podstawie przesłanych do niej danych, którymi są krańce przedziału, oraz liczba równomiernie rozmieszczonych węzłów. Funkcja zwraca tablicę węzłów i elementów.

2.4. Funkcja - RysujGeometrie

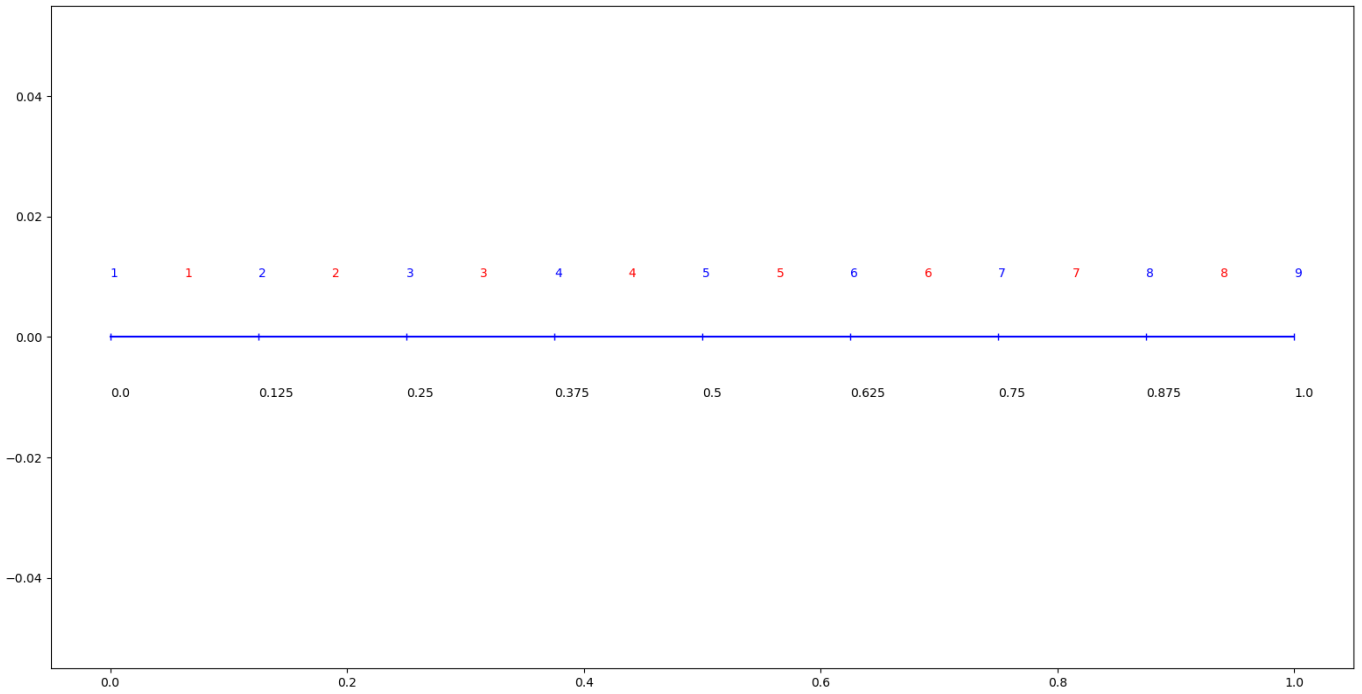
```
import numpy as np
import matplotlib.pyplot as plt
    fh = plt.figure()
    plt.plot(NODES[:,1], np.zeros( (np.shape(NODES)[0], 1) ), '-b|' )
    nodeNo = np.shape(NODES)[0]
    for ii in np.arange(0,nodeNo):
        ind = NODES[ii,0]
        x = NODES[ii,1]
        plt.text(x, 0.01, str( int(ind) ), c="b")
        plt.text(x, -0.01, str(x))
    elemNo = np.shape(ELEMS)[0]
    for ii in np.arange(0,elemNo):
        wp = ELEMS[ii,1]
        wk = ELEMS[ii,2]
        x = (NODES[wp-1,1] + NODES[wk-1,1] ) / 2
        plt.text(x, 0.01, str(ii+1), c="r")
    plt.show()
    return fh
```

Kod przedstawiony wyżej realizuje zadanie rysowania geometrii, pozwala to na graficzną prezentację problemu. Funkcja ta jako argumenty przyjmuje węzły, elementy, oraz warunki brzegowe. Dzięki użyciu biblioteki Matplotlib otrzymamy wykres podobny do wykresów z oprogramowania MATLAB.

Przykładowe wykresy:



Rys. 2.1. Rysunek geometrii dla n = 5



Rys. 2.2. Rysunek geometrii dla n = 9

2.5. Funkcja - Alokacja

```
import numpy as np

def Alokacja(n):

    A = np.zeros([n,n])

    b = np.zeros([n,1])

    return A,b
```

Funkcja pozwalająca na alokację odpowiedniej ilości pamięci dla macierzy *A* i wektora prawej strony *b.* Zwracane zmienne to tablice wypełnione zerami. Ich wielkość jest różna, tablica *A: n* x *n*, natomiast tablica *b: n* x *1*.

2.6. Funkcja - FunkcjeBazowe

```
import numpy as np

def FunkcjeBazowe(n):

   if n==0:

      f  = (lambda x: 1 + 0*x )

      df = (lambda x: 0*x )

   elif n == 1:

      f = (lambda x: -1/2*x + 1/2, lambda x: 0.5*x+0.5 )

      df= (lambda x: -1/2 + 0*x ,   lambda x: 0.5  + 0*x )

   else:

      raise Exception("Blad w funkcji FunkcjeBazowe().")

   return f,df
```

Definiujemy funkcje bazowe, dzięki tej operacji będziemy mogli później policzyć układ równań liniowych zbudowany z macierzy *A,* oraz wektora *B*. Ta funkcja zwraca dwie zmienne, *f* czyli *n+1* elementową listę funkcji bazowych stopnia *n*, oraz *df* czyli *n+1* elementową listę pochodnych funkcji bazowych stopnia *n*.

2.7. Funkcja - Aij

```
import numpy as np

def Aij(df_i, df_j, c, f_i, f_j):

    return lambda x: -df_i(x)*df_j(x) + c*f_i(x)*f_j(x)
```

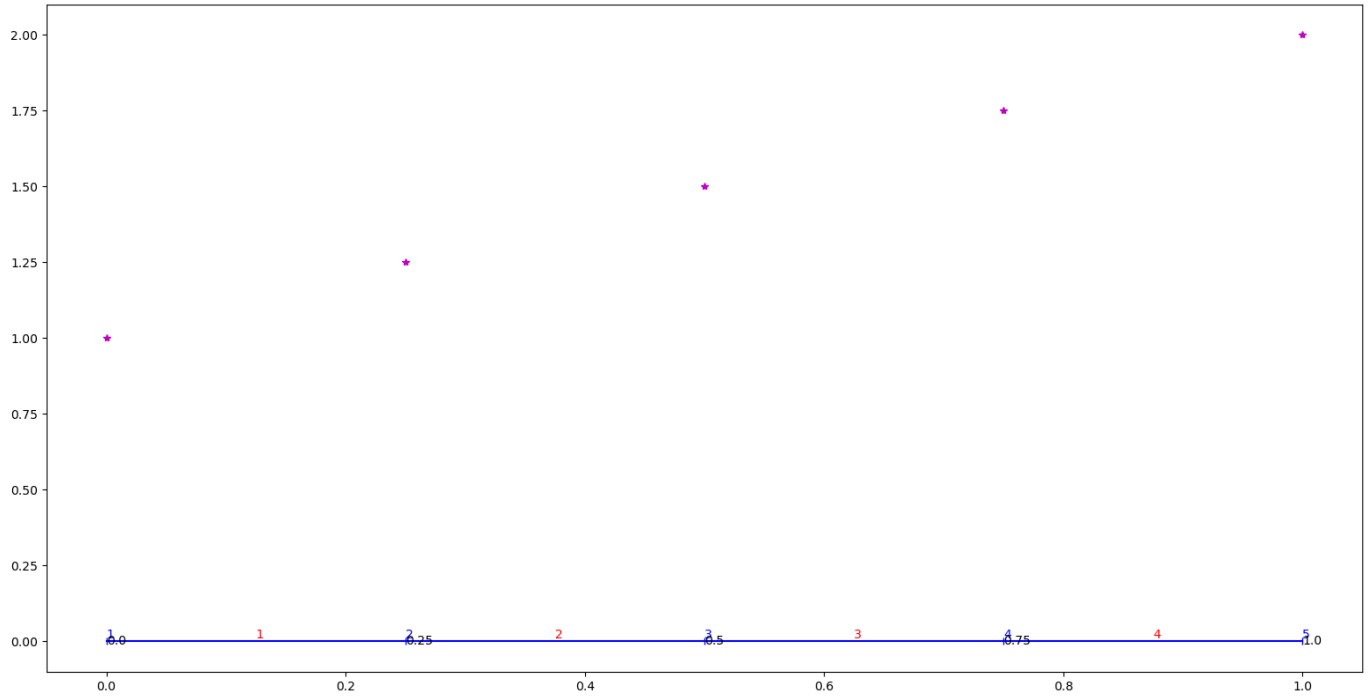
Funkcja licząca całkę i zwracająca wyrażenie lambda obliczające wartości funkcji podcałkowej. Funkcja podcałkowa to iloczyn pochodnych funkcji kształtu.

2.8. Funkcja - RysujRozwiazanie

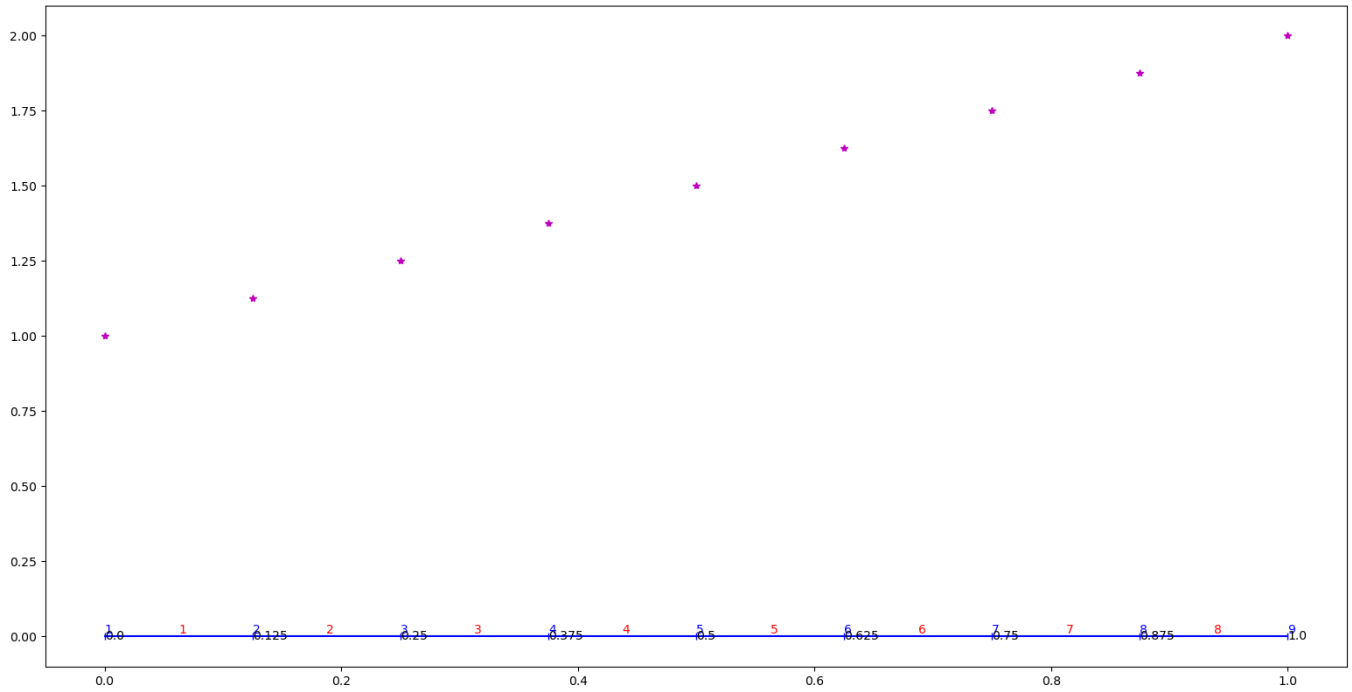
```
import numpy as np

import matplotlib.pyplot as plt

from RysujGeometrie import *

def RysujRozwiazanie(NODES, ELEMS, WB, u):

    RysujGeometrie(NODES,ELEMS,WB)

    x = NODES[:,1]

    plt.plot(x, u, 'm*')
```

Rysowanie rozwiązania, funkcja ta korzysta z funkcji *RysujGeometrie*, oraz współczynnika *u* obliczonego wcześniej. Wykres to działanie funkcji *RysujGeometrie* z naniesionym rozwiązaniem w postaci punktów “ * ”.

Przykładowe wykresy:



Rys. 2.3. Rysunek z rozwiązaniem dla n = 5



Rys. 2.4. Rysunek z rozwiązaniem dla n = 9

3. Podsumowanie

Zrozumienie działania metody elementów skończonych nie jest łatwym zadaniem, jednak jej implementacja w podstawowej wersji dla zagadnień jednowymiarowych nie jest aż tak skomplikowana. Oczywiście pod warunkiem, że ma się odpowiednie narzędzia i umie się z nich korzystać. Całość sprowadza się do realizacji poszczególnych kroków, których nie ma aż tak wiele, a wszystkie obliczenia wykona za nas komputer.