POLITECHNIKA WARSZAWSKA
Wydział Samochodów i Maszyn Roboczych

PRACA DYPLOMOWA
Magisterska

Studia niestacjonarne – zaoczne

Wykorzystanie algorytmu ławicowego do optymalizacji konstrukcji

Application of Particle Swarm Optimization algorithm to structural optimization

Opiekun naukowy:	Wykonał:
Prof. dr hab. inż. Mariusz Pyrz	**Łukasz Tyl**
Prowadzący:	Numer albumu:
Prof. dr hab. inż. Mariusz Pyrz	**229179**

Warszawa, 2015

Łukasz Tyl

OŚWIADCZENIE

Jako autor pracy dyplomowej pt.: **Wykorzystanie algorytmu ławicowego do optymalizacji konstrukcji,** którą wykonałem przestrzegając praw autorskich, zezwalam na publiczne udostępnienie pracy i wyrażam zgodę na jej udostępnienie w Bibliotece Wydziału Samochodów i Maszyn Roboczych Politechniki Warszawskiej w ramach realizacji zadań statutowych biblioteki.

Warszawa, dnia …… …….

SŁOWA KLUCZOWE

1. optymalizacja konstrukcji
2. algorytm ławicowy
3. metoda elementów skończonych
4. konstrukcje prętowe

Streszczenie

Praca dotyczy wykorzystania algorytmu ławicowego do optymalizacji konstrukcji prętowych. Wykorzystany algorytm zaliczyć można do nowych niedeterministycznych metod optymalizacji globalnej, inspirowanych zjawiskami obserwowanymi w naturze. W pierwszej części pracy zostały krótko scharakteryzowane zagadnienia optymalizacji, oraz podstawowe metody numeryczne poszukiwania najlepszych rozwiązań. Następnie opisano algorytm ławicowy podając historię powstania, sposób działania oraz możliwe zastosowanie. W dalszej części omówiono podstawy metody elementów skończonych skupiając się na elementach prętowych, które zostały użyte w optymalizowanych konstrukcjach. W następnej kolejności zostały zaprezentowane i omówione programy napisane w środowisku Scilab. Należą do nich program do obliczeń metodą elementów skończonych konstrukcji zbudowanych z elementów prętowych oraz program służący do poszukiwania rozwiązań optymalnych, wykorzystujący algorytm ławicowy. Algorytm ławicowy został opracowany dla przypadku ciągłych zmiennych optymalizacji oraz dla zmiennych dyskretnych. W dalszej części zaprezentowano testy obydwu programów. W końcowej części pracy przedstawiono wynikowe procedury służące do optymalizacji dwuwymiarowych kratownic. Zaprezentowane przykłady numeryczne dotyczą zagadnienia minimalizacji masy konstrukcji prętowych z ograniczeniami na dopuszczalne naprężenia w elementach i przemieszczenia w węzłach. Analizie poddano płaską kratownicę, model mostu oraz konstrukcji łukowej i omówiono wyniki optymale uzyskane dla różnych sformułowań problemu optymalizacji. W podsumowaniu omówiono efektywność zastosowanej procedury optymalizacyjnej, aspekty numeryczne utworzonego programu oraz możliwości jego rozbudowy i ulepszeń.

Abstract

The thesis concerns the application of the Particle Swarm Optimization (PSO) algorithm to the optimization of truss structures. The applied algorithm can be included in the class of new non-deterministic methods of global optimization inspired by nature. In the first part of the work the optimization definitions and the basic numerical methods used for searching best solutions are described. The next part presents the PSO algorithm – its history, the way how it works and its possible applications. Then basic notions of the finite element method are introduced and the truss elements used in the optimized constructions are presented in details. There are also some information concerning the FEM program created to analyze the equilibrium of the truss structures. The PSO optimization and the FEM programs written in Scilab environment are used to determine the optimal solutions. The PSO algorithm was created to work with continuous or discrete variables of the optimization problem. Next part of the thesis presents test results of both programs. In the final part the resulting procedures used to the optimization of two-dimensional truss structures are shown. Presented numerical examples concern the problem of the mass minimisation under stress and deflection constraints. The plane truss, the simple bridge model and the arch structure were analyzed and the optimal results obtained for different formulations of the optimization problem were discussed. The last part of the thesis includes the analysis of the efficiency of the procedure, the numerical aspects of the developed code, and possibilities of improvement and development of the program.

1 Cel i zakres pracy

1.1 Projektowanie w zagadnieniach inżynierskich i problem optymalizacji

W dzisiejszych czasach projektowanie konstrukcji oraz układów mechanicznych związane jest bardzo mocno z zastosowaniem komputerów. Rosnąca wciąż konkurencja przemysłowa wymaga od inżynierów ciągłego doskonalenia produktów. Optymalizacja jest obecna już we wczesnych stadiach powstawania produktu. Złożoność i skomplikowanie modeli sprawia, że nie wystarcza już intuicja i wiedza konstruktora aby znaleźć najlepsze rozwiązanie. Potrzebne są narzędzia, które wspomagają proces optymalizacji w trakcie projektowania produktu [8].

Metody optymalizacji numerycznej są cennym narzędziem, które umożliwia znalezienie rozwiązań optymalnych wspomagając proces decyzyjny inżynierów [8].

Przykładami zagadnień optymalnego projektowania konstrukcji mogą być minimalizacja masy układu, kosztu wykonania, materiałów, eksploatacji. Optymalizacji poddawany jest także kształt elementów [8].

Praktyczne zastosowanie optymalizacji wymaga opracowania metod, które będą poszukiwać rozwiązań problemów rzeczywistych, nie zaś wymyślonych problemów matematycznych. Spora ilość współczesnych zagadnień inżynierskich należy do trudnych z punktu widzenia optymalizacji. Tradycyjne metody optymalizacji wykorzystują do znalezienia ekstremum warunki analityczne (matematyczne) przy założeniu ciągłego charakteru zmiennych projektowych. Problemy i komplikacje przy ich użyciu pojawiają się, gdy chce się zastosować zmienne dyskretne (lub mieszane), gdy funkcja celu nie jest ciągła na całym obszarze poszukiwań. Trudność stanowią również zagadnienia bardzo skomplikowane, zawierające dużo liczbę zmiennych decyzyjnych. Ich uproszczenia mogą powodować otrzymanie nieprzydatnych wyników, nie mających nic wspólnego z rzeczywistością [8]. Nie bez znaczenia jest również samo skomplikowanie modelu optymalizowanego obiektu i czas potrzebny na jego rozwiązanie. Wzrost mocy obliczeniowej współczesnych komputerów umożliwia obecnie rozwiązywanie coraz bardziej skomplikowanych zagadnień. W ostatnich czasach rosnącym zainteresowaniem cieszą sie stosunkowo młode metody optymalizacji, których algorytmy inspirowane są zjawiskami obserwowanymi w naturze i przyrodzie. Metody te są bardziej efektywne niż klasyczne metody optymalizacji w przypadku trudnych zagadnień optymalizacji.

1.2 Cel pracy

Celem pracy jest stworzenie w środowisku Scilab programu wykorzystującego algorytm ławicowego do optymalizacji konstrukcji prętowych. Do obliczeń wytrzymałościowych posłużył napisany w środowisku Scilab program rozwiązujący zagadnienie równowagi statycznej konstrukcji prętowych metodą elementów skończonych. Proces optymalizacji zostanie przeprowadzony dla zmiennych ciągłych oraz dyskretnych. Efektywność procedury będzie zbadana na wybranych przykładach optymalnego wymiarowania konstrukcji inżynierskich. Analizowane przykłady numeryczne dotyczą minimalizacji masy/ciężaru konstrukcji prętowych z ograniczeniami na dopuszczalne naprężenia w elementach i przemieszczenia w węzłach.

2 Wprowadzenie do optymalizacji

2.1 Wstęp

Optymalizacja konstrukcji i układów mechanicznych wykorzystywana jest podczas projektowania, realizacji i eksploatacji rozwiązań inżynierskich [8]. Celem optymalizacji jest znalezienie najlepszego rozwiązania danego problemu względem ustalonego kryterium. Optymalnym rozwiązaniem może być minimum lub maksimum pewnej funkcji – tzw. wskaźnika jakości. Metody optymalizacji w sposób algorytmiczny poszukują rozwiązań optymalnych bez konieczności sprawdzania wszystkich możliwych przypadków [7]. Przykładem optymalnego projektowania konstrukcji może być poszukiwanie rozwiązania gwarantującego określone właściwości, parametry (np. minimalizacja masy, optymalizacja kształtu) [8].

Sformułowanie zagadnienia optymalizacji wiąże się z następującymi zadaniami [7]:

- opracowaniem modelu matematycznego analizowanego procesu lub układu,
- definicją funkcji celu optymalizacji (funkcjonału jakości, kryterium optymalizacji, kryterium jakości itp.),
- znalezieniem optymalnego rozwiązania z zastosowaniem jednej z metod optymalizacji.

W dalszej części opisu założymy, że [7]:

- funkcja celu

$$f: \mathbb{R}^n \to \mathbb{R} \tag{2.1}$$

-zbiór rozwiązań dopuszczalnych

$$X_d \in \mathbb{R}^n \tag{2.2}$$

Celem optymalizacji jest znalezienie takiego elementu $x^{opt} \in X^d$, że:

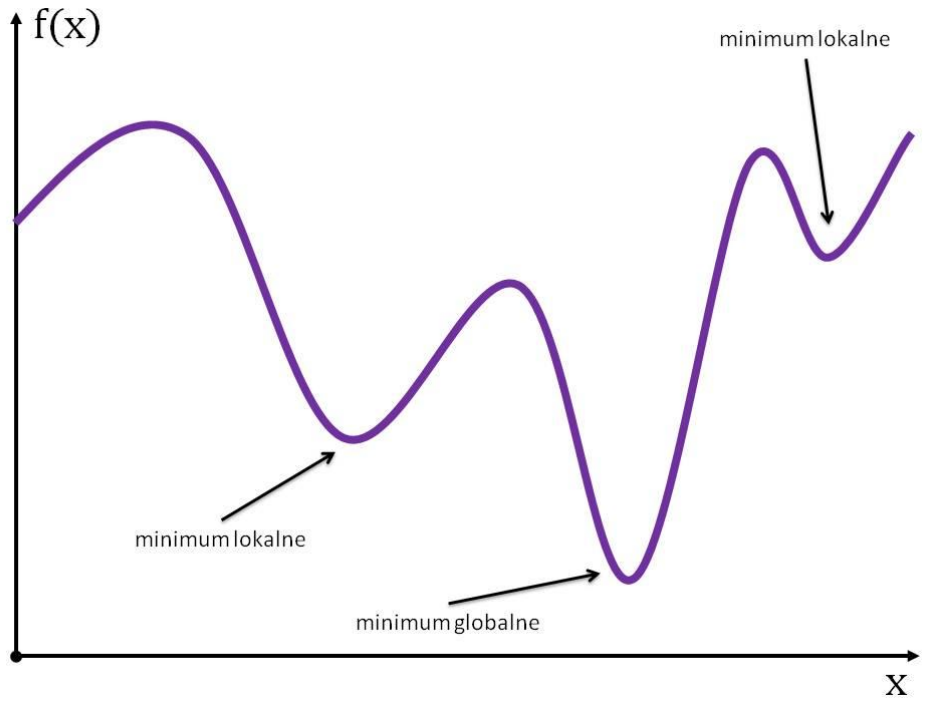
$$f(x^{opt}) \leq f(x) \qquad \forall x \in X^d \left(poszukiwanie\ minimum, x^{min} = x^{opt}\right), \tag{2.3}$$

lub

$$f(x^{opt}) \geq f(x) \qquad \forall x \in X^d \left(poszukiwanie\ maksimum, x^{max} = x^{opt}\right), \tag{2.4}$$

Wybrany algorytm optymalizacyjny wyznacza pewną wartość x*. Znalezione rozwiązanie x* ma być przybliżeniem minimum lub maksimum optymalizowanej funkcji celu. Zatem wartość f(x*) powinna być dobrym przybliżeniem wartości $f(x^{opt})$. Znalezione rozwiązanie optymalne jest minimum bądź maksimum globalnym tylko wewnątrz wcześniej zdefiniowanego zbioru dopuszczalnych rozwiązań X_d. Nie musi jednak być to ekstremum globalne w całej dziedzinie funkcji celu [7].

Trzeba pamiętać, że w analizowanym obszarze poszukiwań często występują ekstrema lokalne funkcji (minimum lub maksimum). Wartość funkcji celu może być w tych punktach odpowiednio: większa (jeśli poszukuje się minimum), mniejsza (jeśli poszukuje się maksimum) od ekstremum globalnego. Miejsca te sprawiają wiele problemów w trakcie poszukiwań optymalnego rozwiązania. Utrudniają algorytmom optymalizacyjnym znalezienie ekstremum globalnego, ponieważ algorytmy zatrzymują się w minimum (maksimum) lokalnym i nie docierają do globalnego ekstremum funkcji. Jednym z kryteriów oceny algorytmów optymalizacyjnych jest ich zdolność do znajdywania ekstremów globalnych, bez względu na napotykane ekstrema lokalne [7].



Rysunek 2.1 Wykres funkcji jednej zmiennej zawierającej minima lokalne

Rysunek 2.1 przedstawia funkcję jednej zmiennej. Zawiera ona kilka minimów. Tylko jedno z nich jest globalne.

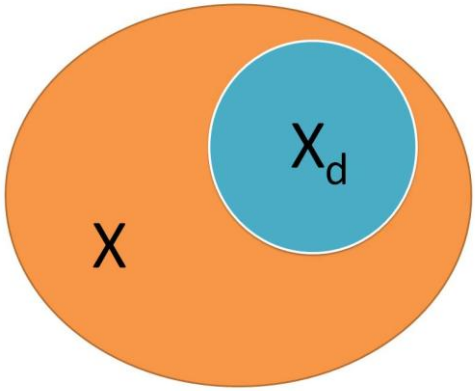
W praktyce wszystkie metody optymalizacji, może z wyjątkiem metod niedeterministycznych, znajdują najczęściej jedynie ekstrema lokalne, a w najgorszym przypadku nie znajdują niczego (algorytm kończy swoje działanie w punkcie, które nie jest nawet ekstremum lokalnym) [7].

2.2 Podstawowe pojęcia

2.2.1 Zbiór rozwiązań dopuszczalnych

Problem, który jest optymalizowany rozwiązuje się w przestrzeni zmiennych projektowych X. Zakładając, że $X = \mathbb{R}^n$, gdzie $\mathbb{R}^n$ jest n-wymiarową przestrzenią wektorów rzeczywistych $(n \in \mathbb{N})$, to zadanie optymalizacji nazywane jest statycznym [7].

W przestrzeni zmiennych optymalizacji X jest definiowany zbiór rozwiązań dopuszczalnych X_d. Prezentuje to poniższy rysunek.



Rysunek 2.2 Zbiór rozwiązań dopuszczalnych X_d w przestrzeni wszystkich rozwiązań X

Zbiór rozwiązań dopuszczalnych $X_d \in X$ jest zbiorem punktów $x \in X$, które brane są pod uwagę w procesie optymalizacji. Zbiór ten wynika ze zdefiniowanych ograniczeń, które musi spełniać wektor x, aby należeć do zbioru rozwiązań dopuszczalnych [7]:

$$X_d = \{x: g_i(x) \leq 0, h_j(x) = 0, i = 1, \dots, k; j = 1, \dots, m\} \qquad (2.5)$$

gdzie:

$g_i(x)$, $h_j(x)$ – funkcje ograniczeń (odpowiednio nierównościowych i równościowych)

Jeśli natomiast nie zostały określone żadne ograniczenia, to $X_d = X$ mówimy o zadaniu optymalizacji bez ograniczeń.

2.2.2 Funkcja celu

Funkcja celu zwana jest również kryterium optymalizacji, kryterium jakości lub funkcjonałem jakości [7].

$$f: X \to \mathbb{R} \tag{2.6}$$

Przyporządkowuje ona każdemu rozwiązaniu $x \in X$ pewną wartość liczbową $f(x) \in \mathbb{R}$.

Funkcja celu opisuje pewną właściwość układu. Właściwość musi być dobrana w taki sposób, aby można było ją opisać za pomocą liczby rzeczywistej. Dzięki temu możliwe jest porównywanie rozwiązań, a co za tym idzie znalezienie rozwiązania, które daje minimalną (lub maksymalną) wartość funkcji celu. Najczęściej zakłada się również, że optymalizowana funkcja jest ciągła [7].

Funkcjami celu mogą być w zagadnieniach inżynierskich np.:

- masa układu,
- częstość drgań własnych,
- naprężenia w konstrukcji.

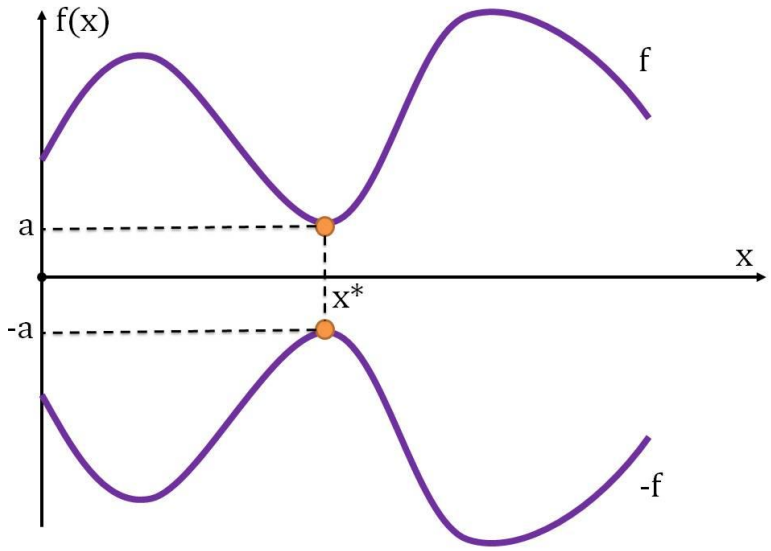
2.2.3 Rozwiązanie optymalne

Rozwiązanie x* jest nazywane minimum globalnym, gdy dla wszystkich $x \in X_d$ jest spełniony następujący warunek:

$$f(x^*) \leq f(x) \tag{2.7}$$

Jeśli warunek jest spełniony tylko w pewnym otoczeniu x*, to jest to minimum lokalne [7].

Należy również zwrócić uwagę, że problemy poszukiwania minimum oraz maksimum funkcji można w pewnym sensie traktować „wymiennie". Jeśli zostanie wyznaczone minimum funkcji f, to jest to równoznaczne z wyznaczeniem maksimum funkcji – f. Zależność tą obrazuje rysunek 2.3.

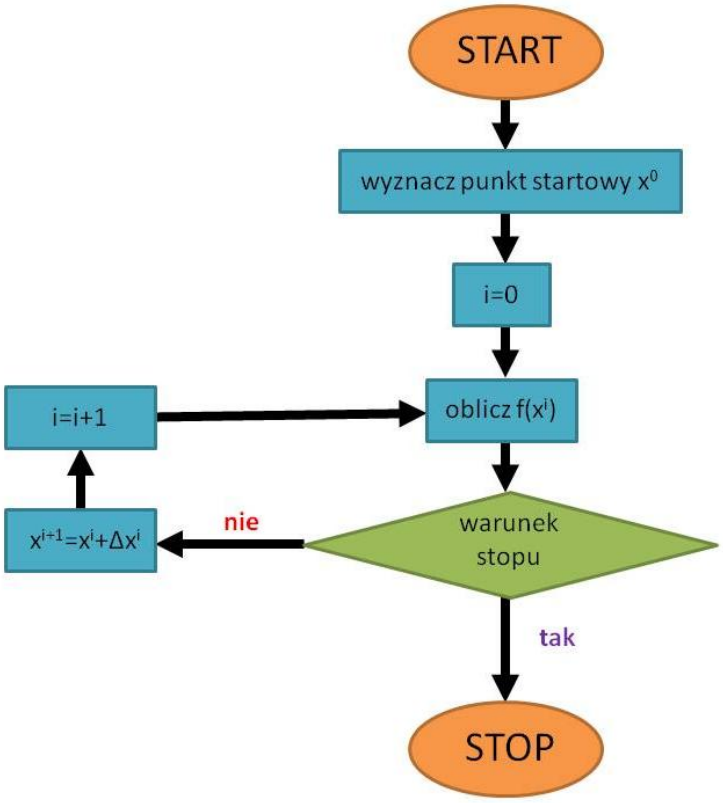


Rysunek 2.3 Poszukiwanie minimum i maksimum funkcji celu

Dysponując więc algorytmem obliczającym minimum funkcji celu, ale chcąc wyznaczyć jej maksimum można zastosować omówione tu przekształcenie w sformułowaniu problemu . Analogicznie można postąpić mając algorytm obliczający maksimum funkcji, a chcąc wyznaczyć jej minimum. Będą to rozwiązania jednoznaczne [7].

2.3 Szukanie rozwiązań optymalnych

Znacząca większość metod optymalizacji oparta jest na iteracyjnym schemacie działania. Początkowo wybierany jest dowolny punkt startowy z dziedziny dopuszczalnych rozwiązań. Następnie obliczana jest wartość funkcji celu i sprawdzany warunek stopu. Jeśli jest on spełniony, to algorytm kończy swoje działanie. Jeśli nie, generowany jest kolejny punkt i procedura jest powtarzana. Schemat działania algorytmu optymalizacyjnego przedstawia rysunek 2.4 [7].



Rysunek 2.4 Ogólny schemat blokowy strategii optymalizacji funkcji celu f [7]

W opisie algorytmu optymalizacji, przedstawionego na rysunku 2.4, przyjęto następujące oznaczenia:

x^0 –punkt startowy, wybrany ze zbioru rozwiązań dopuszczalnych,

i – numer iteracji,

x^i – wektor współrzędnych (zmiennych projektowych), obliczony w iteracji nr i,

x^{i+1} – wektor współrzędnych (zmiennych projektowych), obliczony w iteracji nr i+1,

Δx^i – wektor wyznaczany w każdej iteracji, poprawiający poprzednie rozwiązanie i umożliwiający wyznaczenie nowej wartości rozwiązania. Uwzględnia on kierunek przeszukiwań oraz długość kroku jaki wykonujemy wzdłuż kierunku poszukiwań.

Każda z metod optymalizacji ma swój własny algorytm określania wektora Δx [7].

2.4 Kryterium zatrzymania

W większości przypadków optymalizacji, znalezienie dokładnego, idealnego rozwiązania (ekstremum funkcji celu) jest praktycznie niemożliwe. Wynika to np. z dokładności obliczeń numerycznych wykonywanych przez komputery, złożoności optymalizowanego problemu, ograniczonego czasu obliczeń. Dlatego też ważne jest prawidłowe zdefiniowanie warunku zatrzymania algorytmu optymalizacyjnego. Warunek ten wprowadzany jest w celu określenia jakie przybliżenie dokładnego rozwiązania jest satysfakcjonujące.

Najczęściej stosowane warunki stopu to testy zbieżności [7].

Jest to sprawdzanie o ile różni się punkt obliczony w danej iteracji od obliczonego w poprzedniej. Jeśli różni się nieznacznie, to można sądzić, że wykonywanie kolejnych iteracji nie poprawi znacząco funkcji celu. Warunek wyrażony jest za pomocą normy euklidesowej. Obrazuje go wzór:

$$\left\| x^{i+1} - x^{i} \right\|_2 < \delta \tag{2.8}$$

gdzie:

δ>0 jest ustaloną małą liczbą (bliską 0), definiująca dokładność obliczeń,

($x^{i+1} – x^{i}$) - oznacza różnicę między rozwiązaniami obliczonymi w dwóch kolejnych iteracjach.

Poprzedni test można wyrazić również za pomocą wartości funkcji celu. Tym razem zakłada się, że gdy w kolejnych iteracjach następuje niewielka zmiana wartości funkcji celu dalsze obliczenia nie mają sensu. Oblicza się bezwzględną wartość różnicy między wartościami funkcji celu w dwóch kolejnych iteracjach:

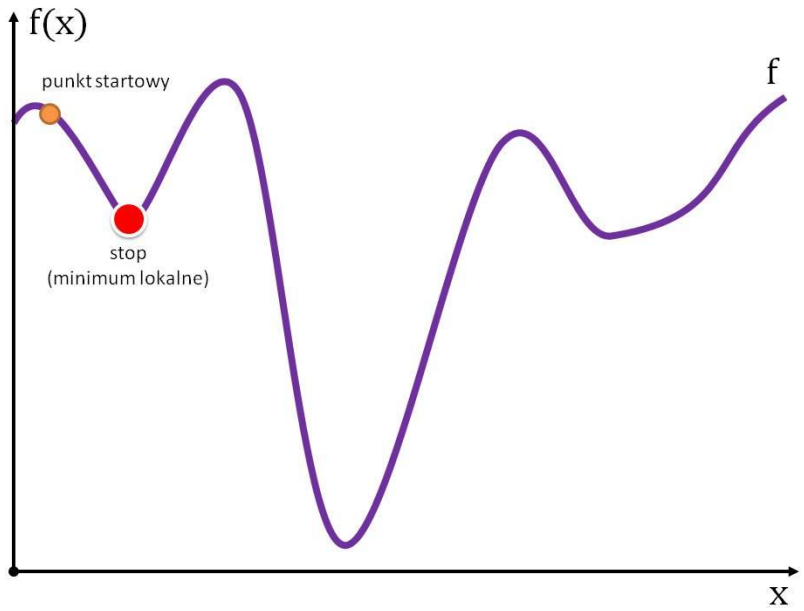
$$\left| f(x^{i+1}) - f(x^{i}) \right| < \varepsilon \tag{2.9}$$

gdzie ε>0 – stała, zazwyczaj bliska 0.

Wiele algorytmów stosuje dodatkowo jako warunek stopu maksymalną liczbę iteracji.

Należy pamiętać, że dla każdego warunku stopu istnieje funkcja, dla której będzie on niepoprawny. Przykładem może być natrafienie na obszar, w którym funkcja celu będzie nieznacznie zmieniała swoją wartość, by później nastąpił gwałtowny spadek i wystąpienie minimum. Istnieje ryzyko, że w takim wypadku warunek stopu zadziała, zatrzyma działanie algorytmu, ale nie osiągając minimum funkcji.

Wpływ na wynik obliczeń ma również wybranie punktu startowego. Źle dobrany punkt początkowy może spowodować, że algorytm „utknie” w minimum lokalnym, nie osiągając minimum globalnego [7]. Przypadek ten widoczny jest na poniższym rysunku.



Rysunek 2.5 Poszukiwanie minimum funkcji celu zakończone w minimum lokalnym

2.5 Charakterystyka głównych metod optymalizacji

W zadaniu optymalizacji poszukiwane jest najlepsze rozwiązanie. Miarą jakości rozwiązania jest funkcja celu. Zmienne, od których zależy wartość funkcji celu to zmienne decyzyjne (lub zmienne projektowe). Dodatkowo rozwiązanie musi jeszcze spełniać warunki ograniczające sformułowane w zadaniu. Szczegółowe omówienie poszczególnych elementów zostało już przedstawione w rozdziale 2.2.

Poszukiwanie rozwiązania optymalnego formułowanie jest jako zadanie poszukiwania ekstremum (minimum) funkcji. Rozwiązanie poszukiwane jest zazwyczaj za pomocą określonego algorytmu charakteryzującego daną metodę numeryczną.

Wg pozycji [7] metody optymalizacji można podzielić ze względu na:

- rodzaj rozwiązywanego zadania,
- ograniczenia,
- wymiar problemu,
- kryteria optymalizacji.

Ogólny podział ze względu na rodzaje rozwiązywanego zadania obejmuje:

- metody programowania liniowego (liniowe: funkcja celu i ograniczenia),
- metody optymalizacji nieliniowej (nieliniowa funkcja celu lub ograniczenia).

Biorąc pod uwagę ograniczenia można dokonać klasyfikacji na:

- metody optymalizacji bez ograniczeń,
- metody optymalizacji z ograniczeniami.

Biorąc pod uwagę wymiar problemu można dokonać klasyfikacji na:

- metody jednowymiarowe (jedna zmienna optymalizacji),
- metody wielowymiarowe (wiele zmiennych optymalizacji).

Biorąc pod uwagę kryteria optymalizacji można dokonać podziału na:

- metody dla pojedynczego kryterium,
- metody wielokryterialne (wiele kryteriów optymalizacji).

Inną klasyfikacją może być podział na:

- metody deterministyczne,
- metody niedeterministyczne.

Metody deterministyczne charakteryzują się tym, że uruchamiając procedurę wiele razy z tymi samymi parametrami i z tego samego punktu początkowego zawsze otrzymuje się ten sam wynik. Nie zawierają one elementu losowości [9]. Metody te można podzielić na [9]:

- bezgradientowe – przeszukiwanie zbioru rozwiązań dopuszczalnych odbywa się tylko z wykorzystaniem wartości funkcji celu,
- gradientowe – kierunek przeszukiwań określa się wykorzystując gradient funkcji celu.

Metody niedeterministyczne zawierają natomiast element losowości. Oznacza to, że procedura uruchamiana wielokrotnie z tego samego punktu początkowego i z tymi samymi parametrami może osiągać za każdym razem inne rozwiązanie, w mniejszym lub większym stopniu bliższe optimum.

2.5.1 Metody deterministyczne

Metody bezgradientowe

W tej grupie metod optymalne rozwiązanie poszukiwane jest z wykorzystaniem tylko znajomości wartości funkcji celu. Metody te nazywa się również metodami bezpośrednimi [7]. Charakteryzują się stosunkowo prostą budową. Kierunek poszukiwań i długość kroku obliczane są tylko na podstawie wartości funkcji celu. Nie wymagają więc obliczania pochodnych funkcji celu. Mogą być stosowane w tych przypadkach, gdzie funkcja celu opisana jest skomplikowanymi zależnościami. Metody te stosowane bywają również wtedy, gdy nie ma możliwości dokładnego opisania funkcji celu wzorami matematycznymi [7].

Metody gradientowe

Metody te w celu znalezienia optymalnego rozwiązania wykorzystują oprócz wartości funkcji celu jej gradient lub wielkości z nim związane. Nakłada to wymóg na funkcję celu, aby była określona i różniczkowalna w całej przestrzeni poszukiwań [7].

Gradient funkcji celu f w punkcie $x = [x_1 \quad x_2 \quad \dots \quad x_n]^T$ jest wektorem, którego składowymi są pochodne cząstkowe funkcji f względem każdej ze zmiennych (składowych wektora x). Definicję gradientu funkcji f prezentuje poniższy wzór.

$$\nabla f(x) = \left[\frac{\partial f(x)}{\partial x_1} \quad \frac{\partial f(x)}{\partial x_2} \quad \dots \quad \frac{\partial f(x)}{\partial x_n}\right]^T \tag{2.10}$$

Metody optymalizacji gradientowej analizują trendy wzrostu funkcji celu na podstawie jej gradientu. Wskazuje on bowiem kierunek największego wzrostu wartości funkcji. Własność ta jest bardzo pomocna przy poszukiwaniu ekstremum funkcji i jest wykorzystywana we wszystkich metodach gradientowych [7].

Ogólna strategia metod gradientowych wykorzystuje:

- określenie kierunku poszukiwania ekstremum przy użyciu gradientu,
- dobór kroku o odpowiedniej długości, który należy wykonać we wcześniej obliczonym kierunku.

Metody gradientowe są zazwyczaj szybciej zbieżne w porównaniu do metod bezgradientowych. Jednak ograniczeniem jest tu możliwość obliczenia gradientu funkcji celu. Jeśli nie jest to możliwe, metody te nie mogą być stosowane [7].

2.5.2 Metody niedeterministyczne

Metody niedeterministyczne zawierają w swojej budowie czynnik losowy. Zaliczane są do metod przybliżonych z powodu ich losowości. Są one łatwe do zastosowania w problemach optymalizacji ze zmiennymi dyskretnymi lub mieszanymi. Efektywność metod losowych może jednak wymagać dużej liczby obliczeń [7]. W przypadku, gdy funkcja celu posiada wiele minimów lokalnych, znalezienie minimum globalnego jest często niemożliwe stosując metody deterministyczne.

W rzeczywistych problemach optymalizacyjnych funkcje celu są bardzo skomplikowane, a liczba zmiennych może być duża. Dlatego poszukiwane są metody, które nie ograniczają się do lokalnej analizy kształtu funkcji celu. Wraz z tym wzrasta szansa na znalezienie ekstremum globalnego. Czynnik losowy zwiększa szanse na znalezienie tego ekstremum.

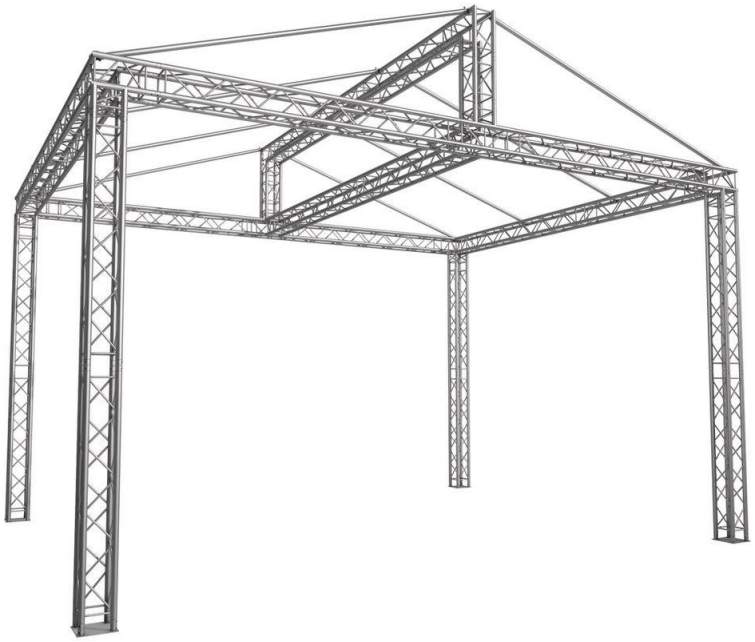
2.6 Optymalizacja konstrukcji prętowych

Wykorzystując elementy prętowe do budowy różnych konstrukcji (rys. 2.6-2.8) zazwyczaj z powodów ekonomicznych jesteśmy ograniczeni katalogiem elementów oferowanych w sprzedaży. Dostępne są więc produkty o określonych, znormalizowanych przekrojach. Optymalizacja takich konstrukcji polega najczęściej na poszukiwaniu rozwiązania o najmniejszym koszcie. Sformułowanie takie może być zazwyczaj uproszczone i zastąpione poszukiwaniem konstrukcji o najmniejszej masie lub najmniejszej objętości zużytego materiału. Dochodzą do tego ograniczenia wynikające np. z dopuszczalnych maksymalnych naprężeń bądź przemieszczeń w węzłach, częstości drgań własnych [8].

Obecność zmiennych dyskretnych sprawia, że problem ma szczególny charakter. W tej sytuacji powstaje problem optymalizacji dyskretnej. Optymalnym rozwiązaniem jest wariant wybrany spośród wszystkich możliwych kombinacji. Pociąga to za sobą przeszukiwanie nieciągłej przestrzeni projektowej. Klasyczne metody optymalizacji (wykorzystujące często gradient funkcji celu) mogą być w takim praktycznym przypadku zbyt ograniczone [8].

Rysunek 2.6 Przykład konstrukcji prętowej - most [i3]

Rysunek 2.7 Przykład konstrukcji prętowej - most [i3]



Rysunek 2.8 Przykład konstrukcji prętowej - scena [i4]

Szczegółowy opis metod optymalizacji dyskretnej zamieszczony został np. w monografii [8]. W niniejszej pracy jako metodę optymalizacji konstrukcji prętowych zaproponowano algorytm ławicowy (Particie Swarm Optimization). Poszukuje on ekstremum funkcji wykorzystując algorytm zastosowany do modelowania zachowania stadnego zwierząt. Należy jednak do metod mających zdolność poszukiwania optimum globalnego. Szczegółowo został on opisany w kolejnym rozdziale.

3 Opis algorytmu ławicowego (PSO)

3.1 Wiadomości wstępne

Algorytm ławicowy, zwany też optymalizacją rojem cząstek (PSO – Particle Swarm Optimization) został po raz pierwszy zaprezentowany przez Eberhart'a i Kennedy'ego w 1995 r. Inspiracją do jego stworzenia były zachowania zwierząt w grupach – ptaki poruszające się w kluczu (rysunek 3.1), ławice ryb [4]. Obecnie jest coraz częściej wykorzystywany do rozwiązywania zagadnień optymalizacji w problemach inżynierskich. Algorytmy ławicowe zastosowano między innymi: w klasycznym problemie optymalizacji dyskretnej układów prętowych, w optymalizacji konstrukcji z materiałów kompozytowych, biorąc pod uwagę kryteria zniszczenia i wytrzymałości w różnych kierunkach [8]. PSO należy do grupy algorytmów niedeterministycznych, tj. generujących rozwiązanie z wykorzystaniem liczb losowych.

PSO posiada wiele wspólnych cech z innymi metodami obliczeń ewolucyjnych, takich jak algorytmy genetyczne [4]. W obu przypadkach początkowa populacja generowana jest losowo. Poszukiwanie optymalnego rozwiązania realizowane jest przez ciągłe aktualizowanie populacji. Jednakże w przeciwieństwie do algorytmów genetycznych, w algorytmie ławicowym nie występuje krzyżowanie osobników oraz mutacje. W PSO potencjalne rozwiązania, zwane cząstkami, poruszają się w przestrzeni możliwych rozwiązań i podążają za cząstką, która jest aktualnym liderem roju – optymalnym rozwiązaniem [4].

Rysunek 3.1 Klucz poruszających się ptaków [i2]

3.2 Zasada działania

Algorytmy ławicowe posługują się specyficzna terminologią, którą zestawiono w tabeli 3.1 z odpowiednikami stosowanymi w zagadnieniach klasycznej optymalizacji. Pierwszym krokiem jest wygenerowanie roju cząstek o losowych położeniach w przestrzeni poszukiwań rozwiązania. Każda cząstka będąca częścią roju jest reprezentowana przez wektor w wielowymiarowej przestrzeni poszukiwań rozwiązania. Do cząstki przydzielony jest też drugi wektor, który determinuje ruch cząstki w kolejnym kroku iteracji. Jest to tzw. wektor prędkości. Wektor prędkości każdej cząstki w kolejnym kroku obliczeń jest modyfikowany na podstawie aktualnego wektora prędkości cząstki, najlepszego znalezionego dotychczas położenia oraz najlepszego położenia – rozwiązania wśród cząstek całego roju [4].

Tabela 3.1 Terminologia stosowana w algorytmach ławicowych

Algorytm ławicowy	**Optymalizacja**
Cząstka	Pojedyncze rozwiązanie
Współrzędne cząstki	Zmienne projektowe
Rój cząstek	Zbiór rozwiązań
Lider roju	Najlepsze rozwiązanie

W optymalizacji rojem cząstek algorytm jest uruchamiany określoną liczbę razy. Liczba iteracji może być ograniczona przez warunek osiągnięcia maksymalnego dopuszczalnego błędu rozwiązania lub może być po prostu określona maksymalną liczbą iteracji. Im więcej cząstek, tym przestrzeń możliwych rozwiązań jest przeszukiwana dokładniej. Należy pamiętać, że wydłuża to czas obliczeń.

Schemat działania algorytmu ławicowego przedstawia rysunek 3.2.

Rysunek 3.2 Schemat działania algorytmu ławicowego

Algorytm ławicowy działa według następującego schematu:

Początkowo cząstki są rozmieszczane losowo w n-wymiarowej przestrzeni poszukiwań. Cząstkom przypisane zostają również losowe prędkości. Następnie zostaje uruchomiona pętla warunkowa. W każdej iteracji wykonywane są następujące obliczenia [5]:

1) obliczenie wartości funkcji celu dla wszystkich cząstek. Dla każdej cząstki ustalany jest jej najlepszy dotychczasowy wynik (pozycja cząstki). Jeżeli obliczona wartość jest gorsza od zapamiętanej - nie wykorzystujemy tej informacji, jeśli lepsza – zostaje zapamiętana,

2) określenie lidera roju. Ustalenie najlepszej globalnej pozycji (wśród wszystkich cząstek) i zapamiętanie jej,

3) aktualizacja prędkości cząstek roju. Obliczany jest wektor prędkości dla każdej cząstki. Poszczególne elementy wektora, to prędkości w kierunku każdego z wymiarów przestrzeni poszukiwań (wymiarów funkcji celu). Przez pojęcie prędkości rozumie się tutaj przemieszczenie o jakie ma się zmienić położenie cząstki w danej iteracji,

4) aktualizacja położenia cząstek roju.

Prędkość i położenie cząstki są obliczane według następujących wzorów:

$$v_{k+1}^i = v_k^i + [c_1 r_{1k}^i (p_k^{li} - x_k^i) + c_2 r_{2k}^i (p_k^g - x_k^i)] \tag{3.1}$$

$$x_{k+1}^i = x_k^i + v_{k+1}^i \tag{3.2}$$

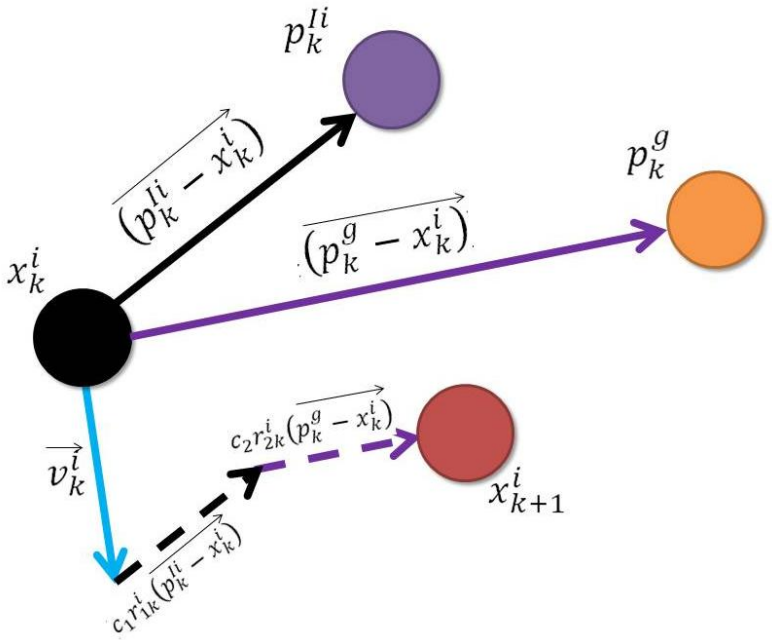
gdzie:

$x_k^i - \ wektor\ położenia\ i - tej\ cząstki,$
$v_k^i - odpowiedni\ wektor\ prędkości,$
$p_k^{li} - najlepsze\ dotąd\ znalezione\ położenie\ i - tej\ cząstki,$
$p_k^g - najlepsze\ dotąd\ znalezione\ położenie\ lidera\ roju,$
$c_1, c_2 - ustalone\ mnożniki\ wagowe,$
$r_{1k}^i, r_{2k}^i - liczby\ losowe\ z\ przedziału\ \langle 0, 1).$

Dodawanie we wzorze (3.2) obiektów reprezentujących położenie i prędkość może się wydać niespójne ze względu na zwyczajowe jednostki, jednakże wzór ten jest ogólnie prezentowany w literaturze. Możemy jednak założyć, że dotyczy on chwil k oraz k+1 odległych w czasie o jedna jednostkę i mnożyć myślowo wektor prędkości przez jednostkę czasu.

W powyższych oznaczeniach indeks k określa kolejny krok iteracji. Współczynniki c_1, c_2, określane są jako współczynniki przyspieszenia lub współczynniki uczenia się. Współczynniki r^i_{1k}, r^i_{2k}, są liczbami losowymi z przedziału <0, 1). Generowane są w kolejnych krokach iteracji dla każdej cząstki.

Rysunek 3.3 przedstawia interpretację graficzną aktualizacji ruchu cząstki.



Rysunek 3.3 Schemat obrazujący aktualizację ruchu cząstki

3.2.1 PSO z wagą inercji

Do algorytmu można dodawać różne metody służące poprawie zbieżności. Jedną z nich może być waga inercji. Została ona wprowadzona przez Y. Shi oraz R.C. Eberhart'a [5].

Metoda polega na wprowadzeniu dodatkowego parametru do podstawowego wzoru na prędkość cząstki:

$$v_{k+1}^i = w_k^i v_k^i + [c_1 r_{1k}^i (p_k^{Ii} - x_k^i) + c_2 r_{2k}^i (p_k^g - x_k^i)] \tag{3.3}$$

$w_k^i - współczynnik\ wagi\ inercji, określany\ na\ poziomie\ i - tej\ cząstki$

Współczynnik wagi inercji ma za zadanie kontrolować wpływ prędkości z poprzedniej iteracji na aktualnie obliczaną prędkość cząstki. Zaproponowano, aby ten współczynnik malał wraz z kolejnymi iteracjami z 0,9 do 0,4. Oznacza to, że wraz z czasem, wpływ prędkości z poprzedniej iteracji na ruch cząstki maleje. Kolejną ważną sprawą jest dostosowanie współczynnika tak, aby nie zakłócał działania samego algorytmu. Jeśli współczynnik wagi inercji będzie zbyt duży, nastąpi nadmierna eksploracja przestrzeni poszukiwań. Jeśli będzie zbyt mały, wtedy narasta eksploatacja [5].

Na obliczaną prędkość cząstki mają wpływ następujące składniki:

$w_k^i v_k^i$ - składnik odpowiadający za ruch cząstki w dotychczasowym kierunku,

$c_1 r_{1k}^i \left(p_k^{Ii} - x_k^i\right)$ - składnik kognitywny – odpowiada za ruch cząstki w kierunku jej dotychczas znalezionego najlepszego położenia,

$c_2 r_{2k}^i \left(p_k^g - x_k^i\right)$ - składnik socjalny – odpowiada za ruch cząstki w kierunku najlepszego znalezionego położenia wśród całego roju.

W niniejszej pracy zastosowano właśnie ten model – PSO z wagą inercji. Wzór, którego użyto do obliczeń współczynnika wagi inercji (Shi, Eberhart – 1998 r.) został zaprezentowany poniżej:

$$w_k^i = w_{max} - \left(\frac{w_{max} - w_{min}}{it_{max}}\right) \cdot k \tag{3.4}$$

w_{max} – maksymalna wartość współczynnika wagi inercji,
w_{min} – minimalna wartość współczynnika wagi inercji,
it_{max} – maksymalna liczba iteracji,
i – nr cząstki,
k – nr iteracji.

Współczynnik wagi inercji maleje od wartości początkowej w_{max} do wartości końcowej w_{min} w każdej kolejnej iteracji, aż do it_{max}.

3.2.2 Pseudokod algorytmu ławicowego

Działanie typowego algorytmu ławicowego można przedstawić w postaci pseudokodu pokazanego na rysunku 3.4.

W opisie procedur zastosowano następujące oznaczenia:

xi – współrzędne i-tej cząstki (położenie w przestrzeni poszukiwań),

pbesti – najlepsze dotychczas znalezione położenie i-tej cząstki,

gbest – najlepsze dotychczas znalezione położenie wśród całej populacji cząstek (lidera roju).

```
Inicjalizacja początkowej populacji losowej

 while nie osiągnieto kryterium stopu do
  for i=1 to rozmiar populacji do
   if f_celu(xi)<f_celu(pbesti) then
   pbesti=xi

   if f_celu(xi)<f_celu(gbest) then
   gbest=xi

   for d=1 to liczba wymiarów do
   aktualizacja prędkości
   aktualizacja położeń
```

Rysunek 3.4 Pseudokod algorytmu ławicowego [5]

3.3 Porównanie algorytmu ławicowego z algorytmami genetycznymi

Procedura obliczeniowa większości technik ewolucyjnych odbywa się następująco [5]:

- Generowana jest losowa populacja startowa.
- Obliczony zostaje współczynnik przydatności każdego osobnika (wynika to z odległości od rozwiązania optymalnego).
- Następuje reprodukcja całej populacji. Oparta jest o wartość obliczonego wcześniej współczynnika.
- Zatrzymanie algorytmu, jeśli został spełniony warunek stopu. W innym wypadku powrót do punktu oceny populacji.

PSO posiada wiele wspólnych cech z algorytmami genetycznymi. W obu przypadkach początkowe populacje generowane są w sposób losowy. Istnieje też funkcja określająca przydatność osobnika. Również oba algorytmy aktualizują populację i szukają w sposób losowy optimum.

PSO nie posiada jednak operatorów genetycznych, takich jak krzyżowanie, czy mutacja. Cząstki zmieniają swoje położenie za pomocą obliczanych prędkości.
Wymiana informacji również odbywa się w inny sposób. W algorytmach genetycznych chromosomy dzielą się między sobą informacją. W PSO tylko najlepszy osobnik (globalny lub lokalny) daje informację innym osobnikom. W algorytmie ławicowym cząstki posiadają też „pamięć” – informację o swojej najlepszej, dotychczas osiągniętej, pozycji [5].

3.4 Zastosowania PSO

Problemy, w których ma efektywne zastosowanie optymalizacja rojem cząstek charakteryzują się dyskretną, ciągłą lub mieszaną przestrzenią poszukiwań, która zwiera wiele minimów lokalnych [5].

PSO znalazło zastosowanie m.in. w następujących dziedzinach:

- optymalne projektowanie elementów konstrukcji,
- trenowanie sieci neuronowych [5],
- dynamika środowiska fizycznego i automatyka [5],
- techniki służące do kontroli pojazdów bezzałogowych [i1].

Na przestrzeni ostatnich lat powstało wiele prac, wykorzystujących algorytmy ławicowe do optymalizacji konstrukcji inżynierskich. Należą do nich między innymi podane w [8] prace, w których PSO wykorzystane zostały do:

- Kaveh i Talatahari [11], [12] stosowali algorytmy ławicowe w klasycznym problemie optymalizacji dyskretnej układów prętowych,
- Narayana Naik, Omkar, Mudigere ~~(2011)~~ [13] - wyznaczali liczbę i układ warstw w płytach kompozytowych, biorąc pod uwagę kryteria zniszczenia i wytrzymałości w różnych kierunkach,
- Gomes [10] – przedstawił efektywność PSO w zagadnieniu minimalizacji ciężaru kratownic z uwzględnieniem ograniczeń na częstości drgań własnych konstrukcji.

Opis zastosowania algorytmu PSO do optymalizacji konstrukcji prętowych można również znaleźć w pozycjach [6], [15], [16].

4 Modelowanie konstrukcji – metoda elementów skończonych

4.1 Wiadomości wstępne

Symulacja zachowania konstrukcji pod wpływem przyłożonych obciążeń wymaga numerycznego rozwiązania złożonych równań. W metodzie elementów skończonych zakłada się, że każdą wielkość (np. przemieszczenie, odkształcenie, naprężenie) opisaną za pomocą funkcji ciągłej w danym obszarze (fragmencie ciągłym modelu fizycznego) aproksymuje się modelem dyskretnym. Model dyskretny złożony jest ze zbioru funkcji ciągłych w skończonej liczbie podobszarów na jakie podzielono rozpatrywany obszar. Poszczególne funkcje ciągłe definiujemy wykorzystując wartości funkcji pierwotnej w skończonej liczbie punktów (zwanych węzłami) dla wnętrza rozpatrywanego podobszaru. Aby uzyskać żądaną dokładność należy przyjąć funkcję kształtu, która dostatecznie dokładnie odwzorowuje rzeczywiste wielkości w elemencie [1].

Metodą elementów skończonych buduje się modele dyskretne złożonych konstrukcji nośnych. Do budowy modelu stosuje się proste geometrycznie elementy skończone ułatwiające niezbędne przekształcenia i operacje obliczeniowe [1].

Przy modelowaniu rzeczywistych układów bardzo ważne jest przejście od modelu fizycznego do dyskretnego tak, aby model dyskretny dostatecznie dokładnie odzwierciedlał zjawiska zachodzące w modelu fizycznym [2].

W ogólnym sformułowaniu w metodzie elementów skończonych należy opracować model matematyczny, który opisuje tzw. zagadnienia brzegowo-początkowe dla rozwiązywanych układów równań różniczkowych cząstkowych. Do głównych zadań należą [2]:

a) opis geometrii,
b) opis materiału,
c) opis obciążenia,
d) opis warunków brzegowych.

Opis geometrii – do opisu geometrii można użyć różnego typu elementów, np.:

- elementy jednowymiarowe (prętowe, belkowe),
- elementy powierzchniowe (tarcze, płyty, powłoki),
- elementy objętościowe (czworościany, pięciościany, sześciany.

Wprowadza się również uproszczenia kształtu rozważanego obszaru, jeśli nie ma to istotnego wpływu na wyniki obliczeń. Gdy istnieje taka możliwość, zakłada się płaski stan naprężeń lub odkształceń i inne [2].

Opis materiału – przy opisie materiału przyjmuje się szereg różnych założeń, np.: jednorodność ośrodka w danym obszarze, materiał liniowy izotropowy, sprężysto-plastyczny, sztywno-plastyczny, założenie, że właściwości fizyczne są stałe w czasie i inne [2].

Opis obciążenia – przyjęcie, że obciążenia są stałe bądź zmienne w czasie, przyjmowanie obciążeń skupionych, pomijanie mało istotnych oddziaływań zewnętrznych, przybliżenie procesów stochastycznych, zachodzących w układzie rzeczywistym, procesami deterministycznymi i inne [2].

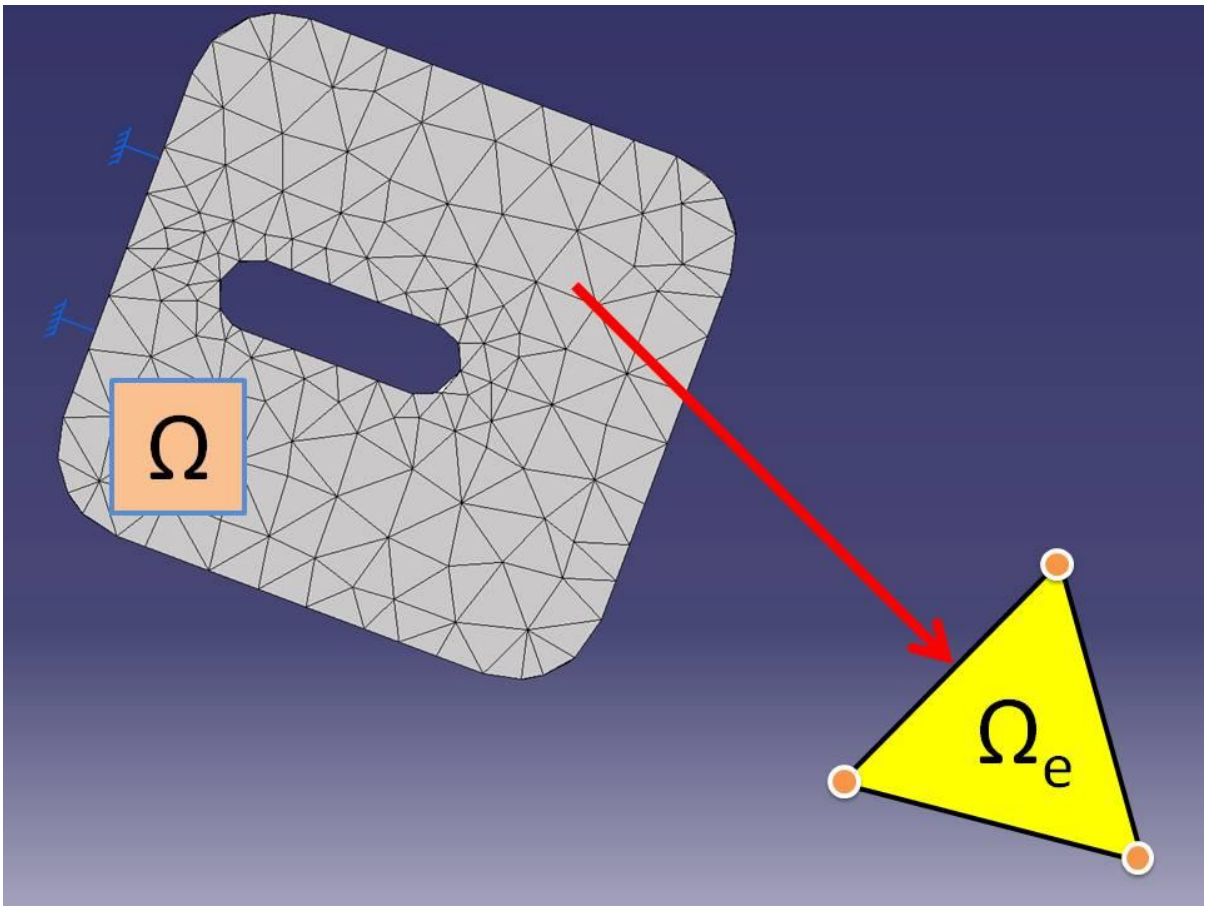
Opis warunków brzegowych – np. przyjmowanie sztywnego utwierdzenia, przyjmowanie przesuwnej podpory w określonym kierunku bez uwzględnienia tarcia, zakładanie podłoża podatnego bez tłumienia lub z tłumieniem i inne [2].

4.2 Schemat ogólny rozwiązywania problemu metodą elementów skończonych

W ogólności rozwiązywanie zagadnień mechaniki ośrodków ciągłych za pomocą metody elementów skończonych sprowadza się do następujących etapów [3]:

- dyskretyzacja geometryczna obszaru (wybór rodzaju elementu skończonego, podział na elementy skończone),
- aproksymacja poszukiwanych funkcji („wewnątrz” każdego elementu skończonego),
- sformułowanie równań „lokalnych” (dla każdego elementu skończonego),
- budowa globalnego układu równań (na podstawie równań równowagi lokalnej),
- obliczenia numeryczne.

4.2.1 Dyskretyzacja geometryczna obszaru



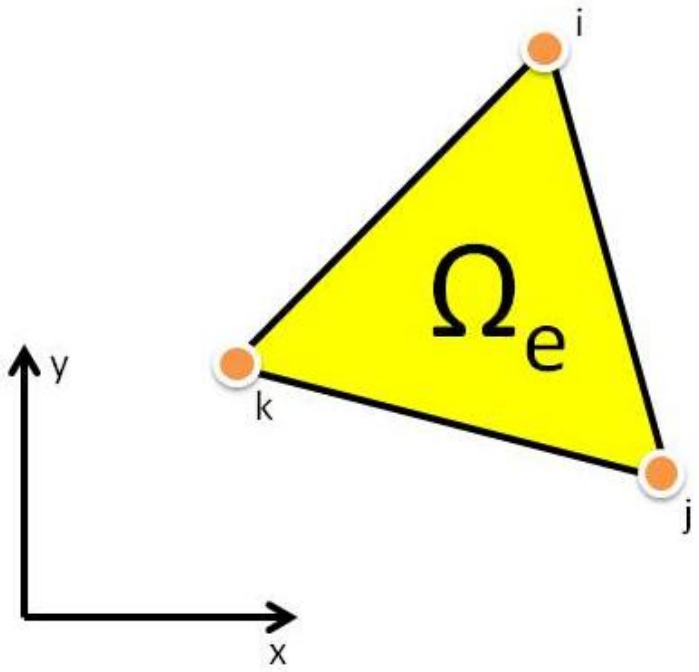
Rysunek 4.1 Dyskretyzacja geometryczna obszaru

Rozpatrywany obszar dzieli się na pewną ilość podobszarów – elementów skończonych, zdefiniowanych przez charakterystyczne punkty, zwane węzłami (rysunek 4.1).

4.2.2 Aproksymacja poszukiwanych funkcji

Wewnątrz każdego elementu skończonego (rysunek 4.2) rozkład poszukiwanych zmiennych jest aproksymowany za pomocą znanych funkcji (funkcji kształtu – najczęściej wielomianów) przez wartości węzłowe (tzw. interpolacja węzłowa).

Rozwiązanie problemu polega na obliczeniu współczynników tych aproksymacji wielomianowych (reprezentowanych przez wartości w węzłach) [3].



Rysunek 4.2 Element skończony

Rozkład poszukiwanej zmiennej:

$$u(x,y) = u_i^e N_i(x,y) + u_j^e N_j(x,y) + u_k^e N_k(x,y)$$
$$N_i(x,y), N_j(x,y), N_k(x,y) - \ funkcje\ kształtu \tag{4.1}$$

Parametry aproksymacji – wartości niewiadomych w węzłach: u_i^e, u_j^e, u_k^e.

W ogólności można zapisać:

$$u(x,y) = \sum_{n=1}^{IE} N_n(x,y)\, u_n \tag{4.2}$$

W formie macierzowej:

$$\langle u \rangle = [N_e]\langle V_e \rangle \tag{4.3}$$

gdzie:

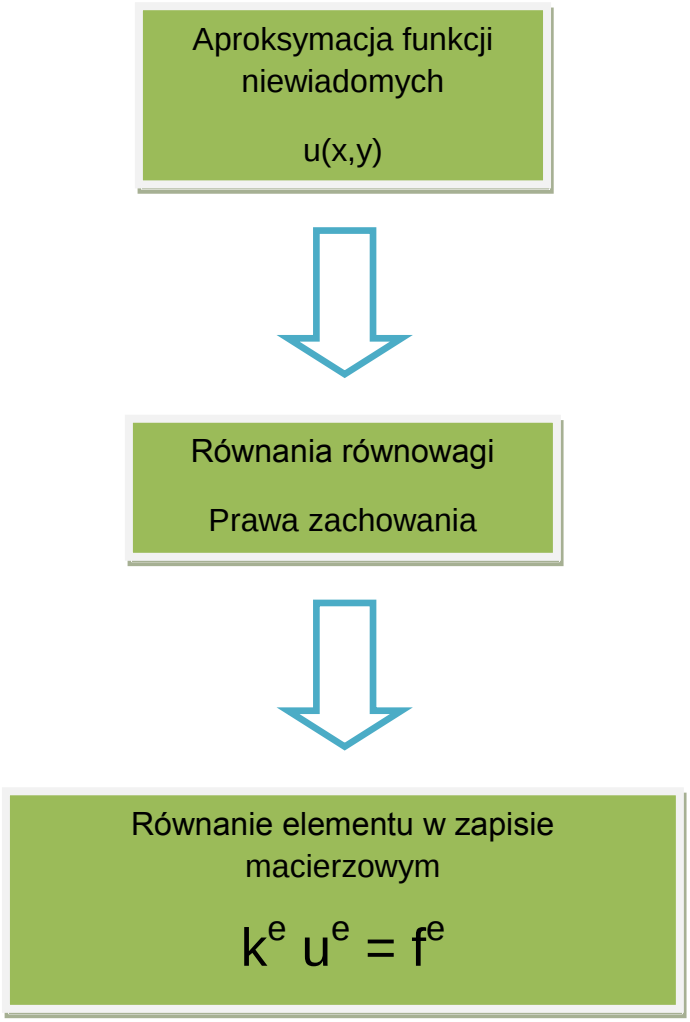
$IE - liczba\ węzłów\ opisujących\ element\ skończony,$

$[N_e] = [N_1,\ N_2, \dots, N_{IE}\ ] - macierz\ funkcji\ kształtu\ elementu,$

$\langle V_e \rangle = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{IE} \end{bmatrix} - wektor\ wielkości\ węzłowych\ elementu (np. przemieszczeń).$

4.2.3 Sformułowanie równań lokalnych

W każdym elemencie skończonym równania problemu (ciągłe) przekształcone zostają do postaci dyskretnej dzięki zastosowaniu odpowiedniej aproksymacji niewiadomych. Równania można otrzymać ze sformułowań energetycznych lub całkowych [3].



4.2.4 Budowa globalnego układu równań

Na podstawie równań równowagi poszczególnych elementów buduje się równania równowagi we wszystkich węzłach układu. Procedura ta nazywana jest agregacją (składaniem) [3].

$$\left.\begin{array}{c} k^1u^1 = f^1 \\ k^2u^2 = f^2 \\ k^3u^3 = f^3 \\ \vdots \\ k^Nu^N = f^N \end{array}\right\} \rightarrow KU = F \tag{4.4}$$

$$KU = F - globalny\ układ\ równowagi \tag{4.5}$$

U - wektor nieznanych wartości w węzłach
N - liczba elementów skończonych
k - macierz sztywności elementu skończonego
K - globalna macierz sztywności układu
U- wektor nieznanych wartości w węzłach
f - wektor sił uogólnionych występujących w elemencie skończonym
F - globalny wektor sił uogólnionych

W układzie równań należy uwzględnić również warunki brzegowe i początkowe.

4.2.5 Obliczenia numeryczne

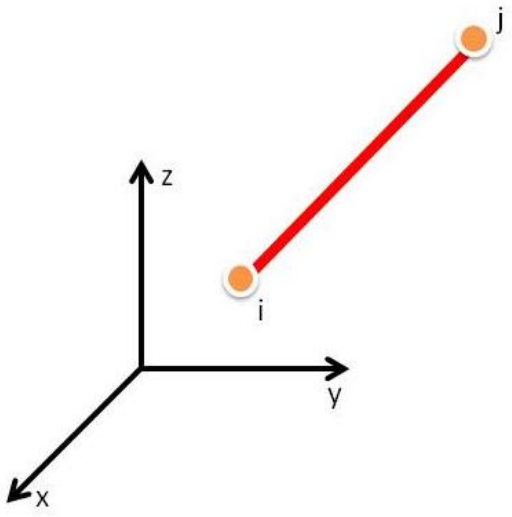
Rozwiązywany jest globalny układ równań w celu wyznaczenia wartości węzłowych.

$$KU = F \rightarrow U \tag{4.6}$$

Na podstawie wyliczonych wartości węzłowych obliczane są szukane niewiadome w całym obszarze. Obliczenia przeprowadza się oddzielnie dla każdego elementu skończonego wykorzystując zastosowane aproksymacje niewiadomych podane zależnościami (4.1).

4.3 Modelowanie konstrukcji prętowych

Element prętowy jest często wykorzystywany do modelowania bardzo sztywnych sztab, pełnych litych belek o bardzo dużej sztywności [2]. Używany jest do modelowania kratownic oraz elementów przenoszących głównie obciążenia osiowe (liny, łańcuchy). W niniejszej pracy analizowane będą konstrukcje kratownicowe.



Rysunek 4.3 Element prętowy

Podstawowe informacje dotyczące rozważanego elementu prętowego (rysunek 4.3) są następujące:

- dwa węzły w elemencie (w każdym węźle maksymalnie 3 stopnie swobody – u_x, u_y, u_z),
- węzły elementu nie przenoszą momentów gnących ani skręcających, element podlega tylko ściskaniu lub rozciąganiu,
- obciążenie poniżej sił krytycznych.

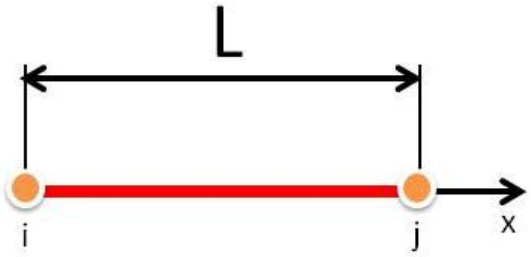
Wymagane dane:

- pole przekroju,
- długość,
- materiał.

Poszukiwane:

- przemieszczenia punktów pręta,
- siły osiowe,
- odkształcenia,
- naprężenia w przekroju pręta.

4.3.1 Wybór niewiadomych i funkcji kształtu



Rysunek 4.4 Element prętowy

Przemieszczenia (aproksymacja liniowa) – w lokalnym układzie współrzędnych mają postać:

$$u(x) = N_i(x)u_i^e + N_j(x)u_j^e = \langle N(x)\rangle\{u^e\} \tag{4.7}$$

gdzie:

$$\begin{gathered}\langle N(x)\rangle = [N_i(x) \quad N_j(x)] \\ \{u^e\} = \left[u_i^e \quad u_j^e\right]^T\end{gathered} \tag{4.8}$$

$$N_i(x), N_i(x) - funkcje\ kształtu$$

$$u_i^e, u_j^e - przemieszczenia\ w\ węzłach$$

Funkcje kształtu oblicza się następująco:

założenie:

$$u(x) = ax + b - funkcja\ liniowa$$

$$dla\ x = 0 \quad u = u_i \quad \rightarrow \quad u_i = a \cdot 0 + b \quad \rightarrow \quad b = u_i \tag{4.9}$$

$$dla\ x = L \quad u = u_j \quad \rightarrow \quad u_j = a \cdot L + b \quad \rightarrow \quad u_j = aL + u_i \quad \rightarrow \quad a = \frac{u_j - u_i}{L}$$

więc:

$$u = \frac{u_j - u_i}{L}x + u_i = \frac{u_j}{L}x - \frac{u_i}{L}x + u_i = u_i\left(\mathbf{1} - \frac{\boldsymbol{x}}{\boldsymbol{L}}\right) + u_j\frac{\boldsymbol{x}}{\boldsymbol{L}} \tag{4.10}$$

funkcje kształtu:

$$N_i(x) = 1 - \frac{x}{L}; \quad N_j(x) = \frac{x}{L} \tag{4.11}$$

4.3.2 Macierz sztywności elementu prętowego (w układzie lokalnym)

Odkształcenie względne dowolnego punktu elementu prętowego ma postać:

$$\varepsilon_{xx} = \frac{du(x)}{dx} \tag{4.12}$$

W zapisie macierzowym:

$$\varepsilon_{xx} = \langle \frac{dN(x)}{dx} \rangle \{u^e\} = \langle B(x) \rangle \{u^e\} \tag{4.13}$$

gdzie:

$$\langle \frac{dN(x)}{dx} \rangle = \left[-\frac{1}{L} \quad \frac{1}{L} \right] = \langle B(x) \rangle \tag{4.14}$$

Zachowanie:

$$\sigma_{xx} = E\varepsilon_{xx} = E\langle B(x) \rangle \{u^e\} \tag{4.15}$$

σ_{xx} – naprężenia w elemencie

E – moduł Young'a

Równania równowagi (minimalizacja całkowitej energii potencjalnej) można zapisać następująco:

$$\pi_t = U_d - T_{ext} \rightarrow min, \tag{4.16}$$

gdzie: π_t – całkowita energia potencjalna,

U_d – energia sprężysta

$$U_d = \frac{1}{2} \int_{V_e} \sigma_{xx} \varepsilon_{xx} dV, \tag{4.17}$$

V_e – objętość elementu,

T_{ext} – praca sił zewnętrznych,

$$T_{ext} = \langle u^e \rangle \{F^e\}, \tag{4.18}$$

$\{F_e\} = [F_i \quad F_j]^T$ – wektor kolumnowy sił działających w węzłach elementu skończonego.

Zatem otrzymujemy

$$\pi_t = \frac{1}{2}\int_{V_e} \sigma_{xx}\varepsilon_{xx} dV - \langle u^e\rangle\{F^e\}. \tag{4.19}$$

Podstawiając do (4.19) wyrażenia na σ_{xx} oraz ε_{xx} otrzymuje się:

$$\begin{aligned} \pi_t &= \frac{1}{2}\int_{V_e} E\langle B(x)\rangle\{u^e\}\langle B(x)\rangle\{u^e\} dV - \langle u^e\rangle\{F^e\}, \\ \pi_t &= \frac{1}{2}\langle u^e\rangle \int_{V_e} \{\boldsymbol{B}(\boldsymbol{x})\}\boldsymbol{E}\langle \boldsymbol{B}(\boldsymbol{x})\rangle \boldsymbol{dV}\,\{u^e\} - \langle u^e\rangle\{F^e\}, \end{aligned} \tag{4.20}$$

gdzie

$$[k^e] = \int_{V_e}\{B(x)\}E\langle B(x)\rangle dV - definicja\ macierzy\ sztywności\ elementu. \tag{4.21}$$

Po kolejnych przekształceniach można zapisać

$$\begin{aligned} \pi_t &= \frac{1}{2}\langle u^e\rangle[k^e]\{u^e\} - \langle u^e\rangle\{F^e\}, \\ min(\pi_t) &\Leftrightarrow \frac{d\pi_t}{du} = 0, \\ \frac{d\pi_t}{du} &= [k^e]\{u^e\} - \{F^e\} = 0. \end{aligned} \tag{4.22}$$

Zależność

$$[k^e]\{u^e\} = \{F^e\} \tag{4.23}$$

przedstawia równanie równowagi elementu prętowego.

Macierz sztywności elementu prętowego (w układzie lokalnym, pokrywającym się z osią pręta), sformułowana przyjmując oznaczenia

L – długość,
A – pole przekroju,
E – moduł Young'a,

ostatecznie można wyrazić jako

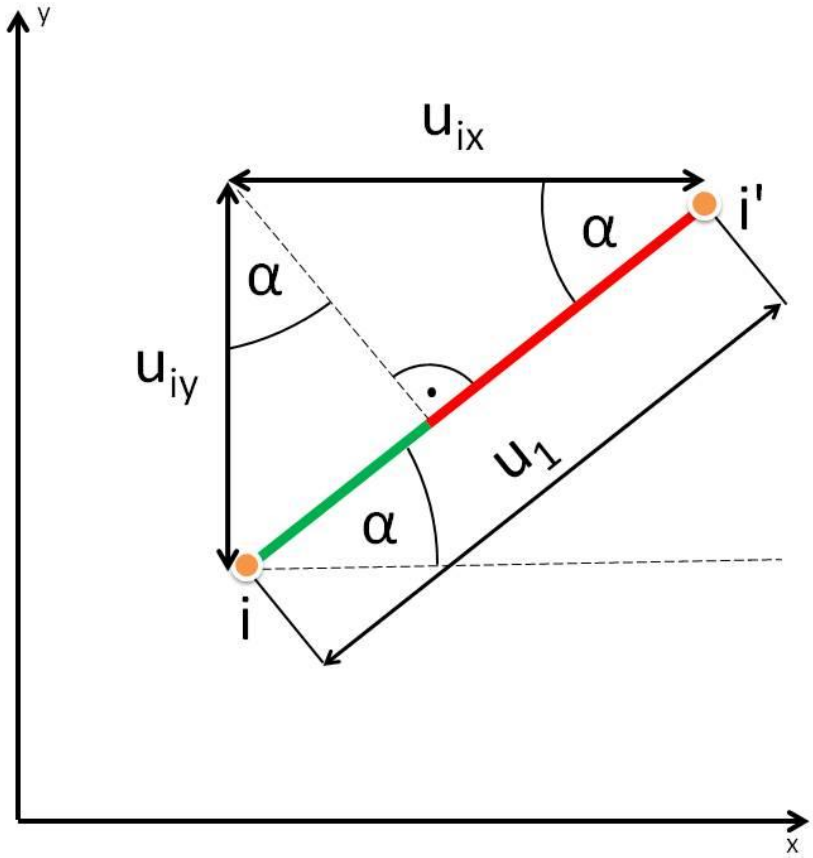
$$[k^e] = \int_{V_e}\{B(x)\}E\langle B(x)\rangle dV = \int_0^L \begin{Bmatrix} -\frac{1}{L} \\ \frac{1}{L} \end{Bmatrix} E \langle -\frac{1}{L} \quad \frac{1}{L}\rangle A dx. \tag{4.24}$$

Po obliczeniach otrzymujemy jej następującą postac:

$$[k^e] = \frac{EA}{L}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}. \tag{4.25}$$

4.3.3 Macierz sztywności elementu prętowego (w układzie globalnym)

W dalszej części przedstawiono sposób wyznaczenia macierzy sztywności elementu prętowego w układzie globalnym (rysunek 4.5), w przestrzeni dwuwymiarowej.



Rysunek 4.5 Przemieszczenia w układzie globalnym

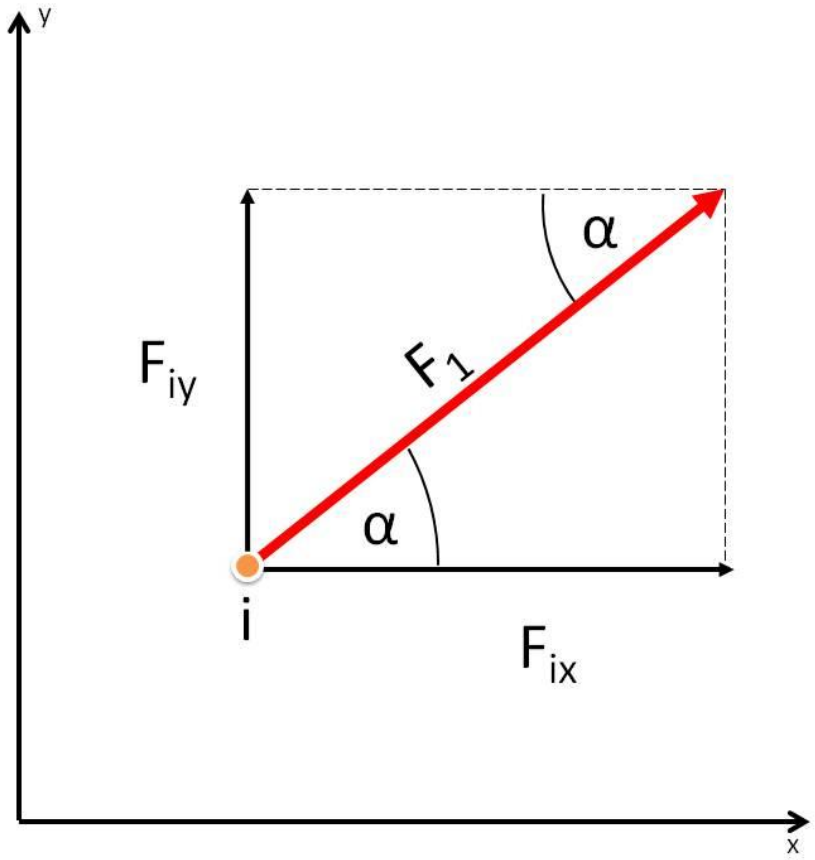
Przemieszczenia węzłów można wyrazić w postaci:

$$\begin{aligned} u_1 &= cos\alpha \cdot u_{ix} + sin\alpha \cdot u_{iy} \\ u_2 &= cos\alpha \cdot u_{jx} + sin\alpha \cdot u_{jy} \end{aligned} \tag{4.26}$$

lub w zapisie macierzowym:

$$\begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix} = \begin{bmatrix} cos\alpha & sin\alpha & 0 & 0 \\ 0 & 0 & cos\alpha & sin\alpha \end{bmatrix} \cdot \begin{Bmatrix} u_{ix} \\ u_{iy} \\ u_{jx} \\ u_{jy} \end{Bmatrix} \tag{4.27}$$

Siły węzłowe również przedstawiamy w układzie globalnym x, y (rysunek 4.6):



Rysunek 4.6 Siły w układzie globalnym

$$\begin{aligned} F_1 cos\alpha &= F_{ix} \\ F_1 sin\alpha &= F_{iy} \\ F_2 cos\alpha &= F_{jx} \\ F_2 sin\alpha &= F_{jy} \end{aligned} \tag{4.28}$$

lub w zapisie macierzowym:

$$\begin{Bmatrix} F_{ix} \\ F_{iy} \\ F_{jx} \\ F_{jy} \end{Bmatrix} = \begin{bmatrix} cos\alpha & 0 \\ sin\alpha & 0 \\ 0 & cos\alpha \\ 0 & sin\alpha \end{bmatrix} \cdot \begin{Bmatrix} F_1 \\ F_2 \end{Bmatrix} \tag{4.29}$$

Wykorzystując macierz przejścia zdefiniowaną jako:

$$[\lambda] = \begin{bmatrix} cos\alpha & sin\alpha & 0 & 0 \\ 0 & 0 & cos\alpha & sin\alpha \end{bmatrix} \tag{4.30}$$

można zapisać:

$$\{u^{lok}\} = [\lambda]\{u^{glob}\}$$
$$\{F^{glob}\} = [\lambda]^T\{F^{lok}\} \tag{4.31}$$

Podstawiając wyrażenia (4.31) do równania (4.23) otrzymujemy:

$$[k^{e\,lok}]\{u^{e\,lok}\} = \{F^{e\,lok}\}$$
$$[k^{e\,lok}][\lambda]\{u^{e\,glob}\} = \{F^{e\,lok}\} \tag{4.32}$$
$$[\boldsymbol{\lambda}]^{\boldsymbol{T}}\left[\boldsymbol{k^{e\,lok}}\right][\boldsymbol{\lambda}]\{u^{e\,glob}\} = [\boldsymbol{\lambda}]^{\boldsymbol{T}}\left\{\boldsymbol{F^{e\,lok}}\right\}$$

$$[k^{e\,glob}] = [\boldsymbol{\lambda}]^{\boldsymbol{T}}\left[\boldsymbol{k^{e\,lok}}\right][\boldsymbol{\lambda}] \tag{4.33}$$

$$\{F^{e\,glob}\} = [\boldsymbol{\lambda}]^{\boldsymbol{T}}\left\{\boldsymbol{F^{e\,lok}}\right\} \tag{4.34}$$

Ostatcznie otrzymujemy równanie równowagi w układzie globalnym:

$$[k^{e\,glob}]\{u^{e\,glob}\} = \{F^{e\,glob}\}. \tag{4.35}$$

Wyznaczenie elementów macierzy sztywności elementu prętowego w układzie globalnym przeprowadzono przyjmując następujące oznaczenia:

- c – cosα
- s – sinα

Podstawiając odpowiednie macierze do wzoru (4.33) na globalną macierz sztywności otrzymuje się:

$$[k^{e\,glob}] = \begin{bmatrix} c & 0 \\ s & 0 \\ 0 & c \\ 0 & s \end{bmatrix} \cdot \frac{EA}{L} \cdot \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \cdot \begin{bmatrix} c & s & 0 & 0 \\ 0 & 0 & c & s \end{bmatrix} \tag{4.36}$$

$$[k^{e\,glob}] = \frac{EA}{L}\begin{bmatrix} c^2 & cs & -c^2 & -cs \\ cs & s^2 & -cs & -s^2 \\ -c^2 & -cs & c^2 & cs \\ -cs & -s^2 & cs & s^2 \end{bmatrix} \tag{4.37}$$

5 Program MES w środowisku Scilab

5.1 Kod programu oraz sposób definicji zmiennych

W środowisku Scilab został napisany kod programu wykonującego obliczenia metodą elementów skończonych. Program posłuży do modelowania zagadnień statyki konstrukcji zbudowanych z elementów prętowych, w przestrzeni dwuwymiarowej.

W załączniku nr 1 przedstawiono przykładowo sposób definicji zmiennych – parametrów konstrukcji, na podstawie których program MES dokonuje obliczeń. Bardziej szczegółowo zostanie omówione to rozdziale 5.2 opisującym główne procedury programu.

Na podstawie podanych danych moduł MES dokonuje dyskretyzacji konstrukcji na elementy prętowe, obliczenia macierzy sztywności wszystkich elementów zgodnie z zależnością 4.37, budowy macierzy sztywności globalnej całej konstrukcji, określenia wektora obciążeń, wprowadzenia warunków brzegowych i rozwiązania uzyskanego układu równań liniowych. Kod modułu MES wykonany w programie Scilab, realizujący wymienione etapy zamieszczony został w załączniku nr 2.

5.2 Opis głównych procedur

5.2.1 Definicja zmiennych i struktura danych

Program MES został napisany jako funkcja, którą można wywołać w innym (pod)programie. Dlatego przed uruchomieniem obliczeń muszą zostać wczytane do pamięci odpowiednie dane konstrukcji - zmienne, które wykorzystuje funkcja – program MES.

Definicja zmiennych, które są skalarami, takich jak np. liczba węzłów, liczba elementów skończonych, moduł Young’a nie wymaga specjalnych objaśnień. Wartości są zapisywane w zmiennych o odpowiednich nazwach – komentarze w kodzie opisują co dokładnie jest przechowywane pod daną nazwą.

Część zmiennych ma postać wektora lub macierzy. Zmienne te zostały opisane poniżej.

5.2.1.1. Macierz współrzędnych węzłów

$$nc = \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \vdots & \vdots \\ x_{nw} & y_{nw} \end{bmatrix} \tag{5.1}$$

nw – liczba węzłów

Macierz „**nc”** jest wymiaru nw x 2. Pierwsza kolumna zawiera współrzędne „x” węzłów, a kolumna druga współrzędne „y”. Przyjęto, że numer wiersza macierzy odpowiada numerowi węzła w konstrukcji.

5.2.1.2. Macierz połączeń węzłów

W macierzy tej definiuje się elementy skończone.

$$mp = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ \vdots & \vdots \\ w_{ne1} & w_{ne2} \end{bmatrix} \tag{5.2}$$

ne – liczba elementów skończonych, z których składa się konstrukcja

Macierz „mp” jest wymiaru ne x 2. Każdy kolejny wiersz macierzy zawiera informację z jakich dwóch węzłów składa się dany element skończony. Wartości w_{i1} oraz w_{i2} muszą należeć do przedziału <1:nw> a także $w_{i1} \neq w_{i2}$.

Przykład:

$$mp = \begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix} \tag{5.3}$$

Oznacza to, że w macierzy „mp” zostały zdefiniowane dwa elementy skończone. Element nr 1 (pierwszy wiersz macierzy) składa się z węzła nr 2 oraz węzła nr 1. Numeracja ta odnosi się, do macierzy współrzędnych węzłów „nc”, zatem węzłowi nr 1 odpowiadają współrzędne z pierwszego wiersza macierzy „nc”, a węzłowi nr 2 współrzędne z drugiego wiersza macierzy „nc”. Analogicznie wygląda sytuacja z elementem nr 2 (drugi wiersz macierzy „mp”). Składa się on z węzła nr 1 oraz węzła nr 3. Nie ma znaczenia kolejność przy podawaniu numerów węzłów, z których składa się element skończony (np. 2-1 lub 1-2).

5.2.1.3. Pola przekroju elementów

$$A = [A_1 \quad A_2 \quad \dots \quad A_{ne}] \tag{5.4}$$

Wektor zawiera wartości pól przekrojów elementów skończonych. Nie jest on bezpośrednio definiowany w pliku z parametrami, lecz jest zmienną funkcji modułu MES.

Dodatkowo definiowane są takie zmienne jak Amin, Amax, katalogA. Nie są one bezpośrednio wykorzystywane przez program MES. Wykorzystuje je natomiast algorytm PSO podczas generowania przekrojów elementów prętowych. Wartości Amin, Amax definiują przedział, z którego może być dobrany przekrój (dla problemu optymalizacji rozwiązywanego dla zmiennych ciągłych). Z kolei wektor `katalogA` zawiera dopuszczalne wartości pól przekroju, które program PSO może wykorzystać (wykorzytywany w przypadku rozwiązywania problemu optymalizcji w którym występują zmienne dyskretne).

5.2.1.4. Definicja obciążenia

Obciążenie (siły zewnętrzne przyłożone w węzłach) definiuje następująca macierz:

$$FP = \begin{bmatrix} F_1 & p_1 \\ F_2 & p_2 \\ \vdots & \vdots \\ F_n & p_n \end{bmatrix} \tag{5.5}$$

Macierz „FP” składa się z dwóch kolumn. Liczba wierszy zależy od ilości sił zewnętrznych przyłożonych w węzłach. W kolumnie pierwszej zapisane są wartości działających sił, w kolumnie drugiej numery kierunków przemieszczeń, w których działają siły. Numeracja musi być zgodna z definicją wektora przemieszczeń „U”.

$$U = \begin{bmatrix} u_{1x} \\ u_{1y} \\ u_{2x} \\ u_{2y} \\ \vdots \\ u_{nw\,x} \\ u_{nw\,y} \end{bmatrix} \tag{5.6}$$

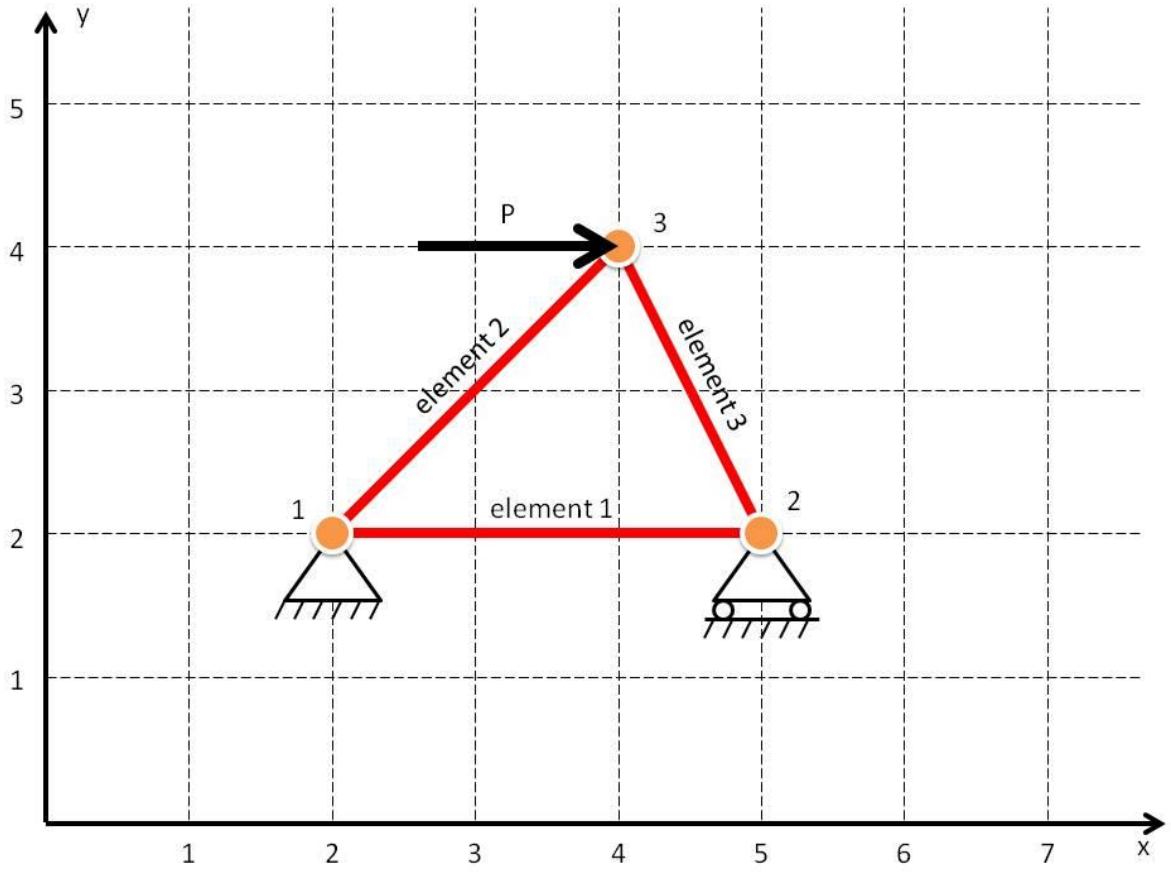
Wektor „U” składa się z *2 x nw* elementów – pod dwa przemieszczenia dla każdego węzła (w kierunku osi „x” oraz w kierunku osi „y”). Wartości p_1, p_2, …,p_n są numerami elementów wektora „U” – przemieszczeń, w kierunku których działa siła.

5.2.1.5. Definicja warunków brzegowych

$$wb = [q_1 \quad q_2 \quad \cdots \quad q_m] \tag{5.7}$$

Definicja warunków brzegowych polega na wpisaniu do wektora „wb” tych numerów przemieszczeń węzłów, w których przemieszczenia mają być równe 0. Wartości q_1, q_2, …, q_m są numerami elementów wektora „U”. Ważne jest to, aby wpisane wartości były w kolejności od największej do najmniejszej.

Na rysunku 5.1 przedstawiony został prosty przykład konstrukcji złożonej z elementów prętowych, dla której zdefiniowane zostały wybrane zmienne, które reprezentują konstrukcję przedstawioną na rysunku.



Rysunek 5.1 Konstrukcja złożona z elementów prętowych

W celu zdefiniowania konstrukcji z rysunku 5.1 należy zatem podać:

$$
\begin{aligned}
U &= \begin{bmatrix} u_{1x} \\ u_{1y} \\ u_{2x} \\ u_{2y} \\ u_{3x} \\ u_{3y} \end{bmatrix} - wektor\ przemieszczeń \\
nc &= \begin{bmatrix} 2 & 2 \\ 5 & 2 \\ 4 & 4 \end{bmatrix} - współrzędne\ węzłów \\
mp &= \begin{bmatrix} 1 & 2 \\ 1 & 3 \\ 3 & 2 \end{bmatrix} - macierz\ połączeń \\
FP &= [P \quad 5] - siły\ zewnętrzne \\
wb &= [4 \quad 2 \quad 1] - warunki\ brzegowe
\end{aligned}
\tag{5.8}
$$

5.2.2 Obliczanie globalnej macierzy sztywności układu

W pierwszej kolejności program oblicza macierz sztywności (w układzie globalnym) każdego z elementów. Macierz obliczana jest zgodnie ze wzorem (4.37). W tym celu potrzebna jest informacja o polu przekroju elementu, jego długości oraz kącie nachylenia do osi X.

Pole przekroju znajduje się w wektorze A (5.4). Długość elementu oraz kąt nachylenia do osi X obliczane są na podstawie współrzędnych węzłów, z których składa się dany element. Jest to po prostu obliczenie długości i kąta nachylenia wektora zdefiniowanego przez dwa punkty - węzły (jego początek i koniec). Procedurę przedstawia poniższy fragment kodu (załącznik nr 2 zawiera pełny kod analizy MES, w której jest ona wykorzystana).

```
    //Obliczenie dlugosci oraz katow nachylenia poszczegolnych
elementow w globalnym ukladzie wspolrzednych

    L=zeros([1:ne]); //wektor dlugosci elementow [L1, L2, ...
, Lne]
    alfa=zeros([1:ne]); //wektor katow nachylenia elementow w
globalnym ukladzie wspolrzednych [rad]

    for i=1:ne,
            //dlugosc elementu
                //wspolrzedne wektora i-tego elementu [x,y]
                wspx=nc(mp(i,2),1)-nc(mp(i,1),1);
//wspolrzedna x
                wspy=nc(mp(i,2),2)-nc(mp(i,1),2);
//wspolrzedna y
            L(i)=sqrt((wspx^2)+(wspy^2)); //dlugosc i-tego
elementu
```

```
            //kat nachylenia
            if wspy>=0 then
                alfa(i)=acos(wspx/L(i));
            else
                alfa(i)=2*%pi-acos(wspx/L(i));
            end
end
```

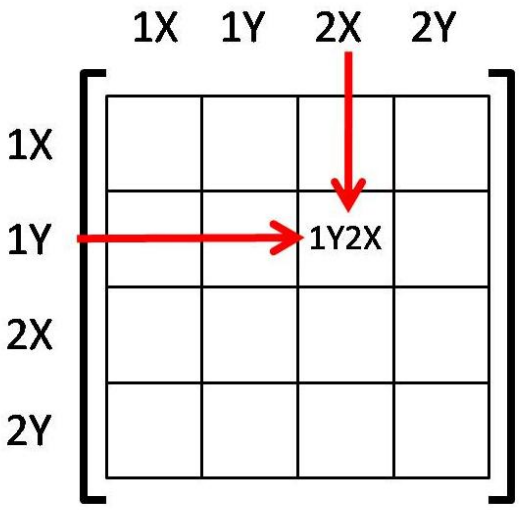
Tak obliczone elementy macierzy sztywności elementu prętowego muszą zostać zapisane na odpowiednich pozycjach globalnej macierzy sztywności układu.

Macierz sztywności elementu prętowego w globalnym układzie współrzędnych ma wymiar 4x4. Każdy element składa się z dwóch węzłów, które mają możliwość przemieszczania się w dwóch kierunkach – wzdłuż osi X oraz osi Y.

Równanie równowagi elementu wygląda następująco:

$$[k^{e\ glob}] \cdot \begin{bmatrix} u_{1x} \\ u_{1y} \\ u_{2x} \\ u_{2y} \end{bmatrix} = \{F^{e\ glob}\} \tag{5.9}$$

Sposób indeksowania elementów macierzy sztywności przedstawia rysunek 5.2.

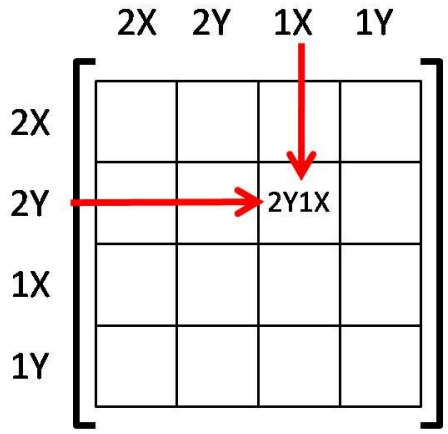


Rysunek 5.2 Macierz sztywności elementu prętowego w globalnym układzie współrzędnych – indeksowanie

Należy zwrócić uwagę, że istotny jest sposób definicji elementu skończonego. Znaczenie ma to, czy element prętowy został zdefiniowany jako składający się z elementów 1-2, czy 2-1. Powyższe równanie równowagi i macierz sztywności odpowiadają definicji elementu składającego się z elementów 1-2. Dla definicji

elementu składającego się z elementów 2-1 równanie równowagi i macierz sztywności prezentują się następująco:

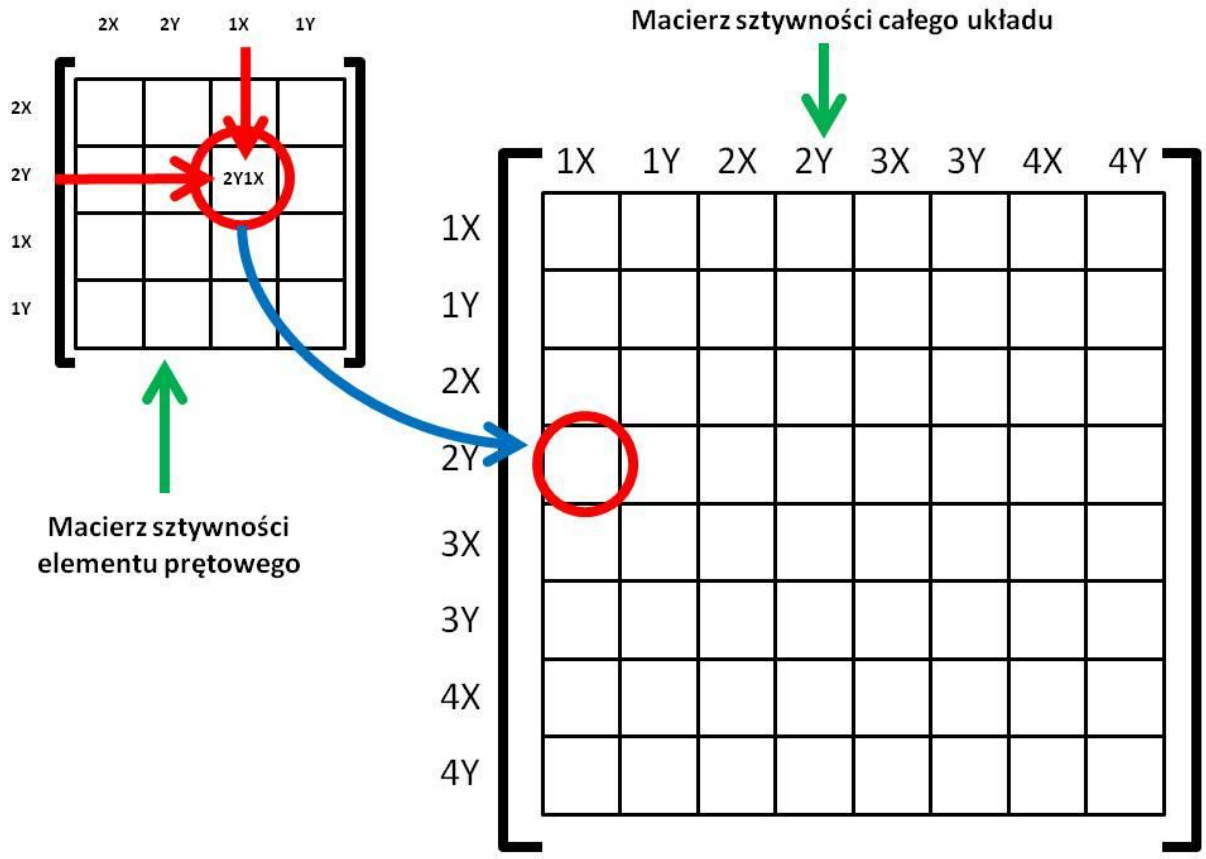
$$[k^{e\ glob}] \cdot \begin{bmatrix} u_{2x} \\ u_{2y} \\ u_{1x} \\ u_{1y} \end{bmatrix} = \{F^{e\ glob}\} \tag{5.10}$$



Rysunek 5.3 Macierz sztywności elementu prętowego w globalnym układzie współrzędnych – indeksowanie

Z punktu widzenia wyników obliczeń nie ma żadnego znaczenia w jakiej kolejności zostały podane węzły definiujące element. Jest to natomiast ważne przy aktualizowaniu elementów globalnej macierzy sztywności układu, aby konkretne wartości zostały zapisane w odpowiednich miejscach globalnej macierzy sztywności układu.

Globalna macierz sztywności układu jest zawsze wymiaru **2nw x 2nw** (nw – liczba węzłów). Rysunek 5.4 pokazuje w jaki sposób zawartość macierzy sztywności elementu zapisywana jest w globalnej macierzy sztywności układu.



Rysunek 5.4 Aktualizowanie zawartości globalnej macierzy sztywności układu

W dalszej części podano fragment kodu programu (załącznik nr 2), zawierającego opisywaną procedurę.

```
for i=1:ne,
    kat=alfa(i);

    K=[cos(kat)^2, cos(kat)*sin(kat), -(cos(kat)^2), -
(cos(kat)*sin(kat));
    cos(kat)*sin(kat), sin(kat)^2, -(cos(kat)*sin(kat)), -
(sin(kat)^2);
    -(cos(kat)^2), -(cos(kat)*sin(kat)), cos(kat)^2,
cos(kat)*sin(kat);
    -(cos(kat)*sin(kat)), -(sin(kat)^2), cos(kat)*sin(kat),
sin(kat)^2];

    KE=(E*A(i)/L(i))*K;
    //mprintf("Macierz sztywnosci elementu nr: %i",i)
    //disp(KE)
    //disp("***")

    //Zapisanie wartosci w globalnej macierzy sztywnosci
ukladu
```

```
d1=(mp(i,1)-1)*2; //przyrost dla wspolrzednych 1. wezla
d2=(mp(i,2)-2)*2; //przyrost dla wspolrzednych 2. wezla

KP=zeros(2*nw,2*nw); //pomocnicza macierz sztywnosci

KP(1+d1,1+d1)=KE(1,1);
KP(1+d1,2+d1)=KE(1,2);
KP(1+d1,3+d2)=KE(1,3);
KP(1+d1,4+d2)=KE(1,4);

KP(2+d1,1+d1)=KE(2,1);
KP(2+d1,2+d1)=KE(2,2);
KP(2+d1,3+d2)=KE(2,3);
KP(2+d1,4+d2)=KE(2,4);

KP(3+d2,1+d1)=KE(3,1);
KP(3+d2,2+d1)=KE(3,2);
KP(3+d2,3+d2)=KE(3,3);
KP(3+d2,4+d2)=KE(3,4);

KP(4+d2,1+d1)=KE(4,1);
KP(4+d2,2+d1)=KE(4,2);
KP(4+d2,3+d2)=KE(4,3);
KP(4+d2,4+d2)=KE(4,4);


KG=KG+KP; //globalna macierz sztywnosci


End
```

Procedura aktualizowania zawartości globalnej macierzy sztywności realizowana w programie

Cała procedura jest wywoływana w pętli ***for*** określoną liczbę razy, zależną od ilości elementów skończonych w konstrukcji.

Zostały również zdefiniowane pomocnicze zmienne d1 oraz d2. Są to wartości, o które trzeba zmienić współrzędne (pozycje) elementów w macierzy sztywności elementu, tak aby otrzymać pozycję w globalnej macierzy sztywności układu, do której ma być zapisana dana wartość.

$$d1 = (mp(i,1) - 1) \cdot 2$$
$$d2 = (mp(i,2) - 2) \cdot 2 \qquad (5.11)$$

i – numer elementu skończonego, dla którego obliczane są wartości zmiennych d1, d2

mp(i,1) – wyrażenie, które zwraca wartość numeru pierwszego węzła, z którego składa się i-ty element prętowy

mp(i,2) – wyrażenie, które zwraca wartość numeru drugiego węzła, z którego składa się i-ty element prętowy

Działanie procedury ilustruje przykład odpowiadający konstrukcji z rysunku 5.4.

Element prętowy składa się z węzłów nr 2 oraz nr 1. Celem jest zapisanie wartości z pozycji „2Y1X” (wg przyjętego w pracy indeksowania) w globalnej macierzy sztywności. Pozycja „2Y1X”, to po prostu element znajdujący się na przecięciu wiersza nr 2 z kolumną nr 3 – KE(2,3).

Wartości d1, d2:

$$\begin{gathered} d1 = (2-1)\cdot 2 = 2 \\ d2 = (1-2)\cdot 2 = -2 \end{gathered} \tag{5.12}$$

Zatem zgodnie z kodem programu:

$$\begin{gathered} KP(2+d1, 3+d2) = KE(2,3) \\ KP(2+2, 3-2) = KE(2,3) \\ KP(4,1) = KE(2,3) \end{gathered} \tag{5.13}$$

Powyższe obliczenia potwierdzają poprawność procedury.

W analogiczny sposób zapisywane są pozostałe elementy macierzy.

5.2.3 Rozwiązywanie globalnego układu równań równowagi

Program MES ma do rozwiązania następujące wynikowe równanie macierzowe przedstawiające układ liniowych rownań algebraicznych:

$$KP \cdot UP = FP \tag{5.14}$$

KP – globalna macierz sztywności układu

UP – wektor przemieszczeń w węzłach

FP – wektor sił działających w węzłach

Celem jest wyznaczenie wektora przemieszczeń.

$$KP \cdot UP = FP \rightarrow UP \tag{5.15}$$

Poniższy fragment kodu z załącznika nr 2 przedstawia sposób rozwiązania równania macierzowego (5.14) w programie Scilab.

```
//Rozwiazanie ukladu rownan

    //K*U=F --. U=K/F
    UP=KP\FP;
```

Jednak przed przystąpieniem do wyznaczenia wektora przemieszczeń należy uwzględnić warunki brzegowe, zdefiniowane w wektorze wb. Sprowadza się to do usunięcia tych numerów wierzy oraz kolumn z macierzy oraz wektorów równania macierzowego, dla których przemieszczenia są równe 0.

Program na podstawie danych zawartych w wektorze wb sam automatycznie usuwa odpowiednie wiersze i kolumny. W dalszej czesci pokazano fragment kodu programu (załącznik nr 2), realizującego opisaną procedurę.

```
    //zdefiniowanie warunkow brzegowych
     [lw, lc]=size(wb); //lw - liczba wierszy, lc - liczba
kolumn

     KP=KG; //pomocnicza maciez KP, zeby nie stracic danych
przy usuwaniu wierszy z macierzy KG
     FP=F; //pomocniczy wektor FP, zeby nie stracic danych
przy usuwaniu elementow z wektora F
     poz=(1:2*nw); //pomocniczy wektor - pozycje
sil/przemieszczen - zeby bylo wiadomo w ktorych wezlach
wyznaczono przemieszczenia (w pelnym ukladzie rownan)

     for i=1:lc,     //kasowanie wierszy i kolumn tam,
gdzie przemieszczenia sa rowne "0"

         KP(wb(i),:)=[];      //usuwanie kolumn
         KP(:,wb(i))=[];      //usuwanie wierszy
         FP(wb(i),:)=[];
         poz(:,wb(i))=[];
      end
```

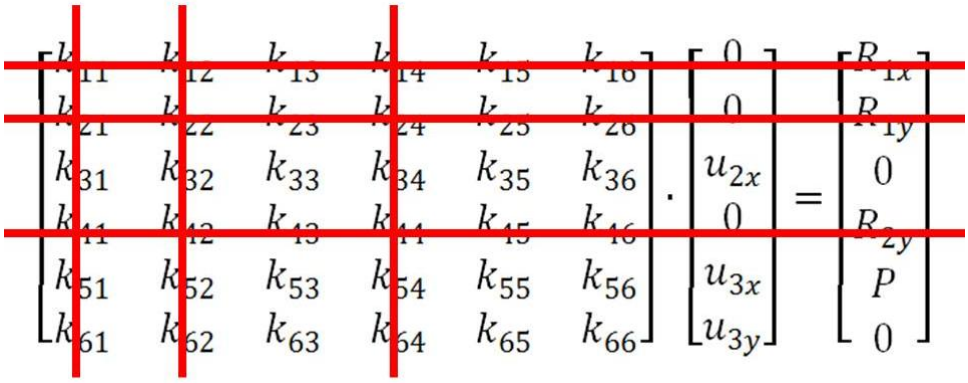
W przypadku przykładu z rysunku 5.1 otrzymujemy:

$$wb = [4 \quad 2 \quad 1] - warunki\ brzegowe$$

$$\begin{bmatrix} k_{11} & k_{12} & k_{13} & k_{14} & k_{15} & k_{16} \\ k_{21} & k_{22} & k_{23} & k_{24} & k_{25} & k_{26} \\ k_{31} & k_{32} & k_{33} & k_{34} & k_{35} & k_{36} \\ k_{41} & k_{42} & k_{43} & k_{44} & k_{45} & k_{46} \\ k_{51} & k_{52} & k_{53} & k_{54} & k_{55} & k_{56} \\ k_{61} & k_{62} & k_{63} & k_{64} & k_{65} & k_{66} \end{bmatrix} \cdot \begin{bmatrix} u_{1x} \\ u_{1y} \\ u_{2x} \\ u_{2y} \\ u_{3x} \\ u_{3y} \end{bmatrix} = \begin{bmatrix} R_{1x} \\ R_{1y} \\ 0 \\ R_{2y} \\ P \\ 0 \end{bmatrix} - równanie\ równowagi \tag{5.16}$$

Zgodnie z wektorem wb, następujące przemieszczenia mają wartość równą 0: u_{1x}, u_{1y}, u_{2y}.

Usuwanie odpowiednich kolumn i wierszy przedstawia rysunek 5.5:



Rysunek 5.5 Usuwanie wierszy i kolumn – realizacja warunków brzegowych

Po usunięciu odpowiednich elementów równanie przyjmuje postać:

$$\begin{bmatrix} k_{33} & k_{35} & k_{36} \\ k_{53} & k_{55} & k_{56} \\ k_{63} & k_{65} & k_{66} \end{bmatrix} \cdot \begin{bmatrix} u_{2x} \\ u_{3x} \\ u_{3y} \end{bmatrix} = \begin{bmatrix} 0 \\ P \\ 0 \end{bmatrix} \tag{5.17}$$

Powyższe równanie zostaje rozwiązane przez program. Wynikami są wartości przemieszczeń w węzłach.

5.2.4 Obliczanie naprężeń w poszczególnych elementach

Naprężenia występujące w elemencie prętowym wyrażone w układzie lokalnym wyrażają się zależnością:

$$\sigma = E\varepsilon = E\frac{(u_2 - u_1)}{l} \tag{5.18}$$

W zapisie macierzowym można je przedstawic jako:

$$\sigma = \frac{E}{l}[-1 \quad 1][u_1 \quad u_2]^T \tag{5.19}$$

Aby przejść do globalnego układu współrzędnych wykorzystano zależność (4.31) tj. związek między przemieszczeniami w lokalnym i globalnym układzie współrzędnych).

$$\sigma = \frac{E}{l}[-1 \quad 1][\lambda]\begin{bmatrix} u_{1x} \\ u_{1y} \\ u_{2x} \\ u_{2y} \end{bmatrix} = \frac{E}{l}[-\cos\alpha \quad -\sin\alpha \quad \cos\alpha \quad \sin\alpha]\begin{bmatrix} u_{1x} \\ u_{1y} \\ u_{2x} \\ u_{2y} \end{bmatrix} \tag{5.20}$$

Dalej zaprezentowano fragment kodu programu, który odpowiada za obliczanie naprężeń w elementach (załącznik nr 2).

```
//Naprezenia w poszczegolnych elementach

    for i=1:ne,
        US=[U(2*(mp(i,1))-1);       //wektor przemieszczen w
wezlach poszczegolnego elementu
            U(2*(mp(i,1)));
            U(2*(mp(i,2))-1);
            U(2*(mp(i,2)))]

        sigma_e=(E/L(i))*[-cos(alfa(i)), -sin(alfa(i)),
cos(alfa(i)), sin(alfa(i))]*US; //naprezenie w i-tym elemencie

        sigma(i)=sigma_e; //wektor zawierajacy wartosci
naprezen w elementach

    end
```

Obliczenia wykonywane są w pętli **`for`**, a wyniki zapisywane są w wektorze oznaczonym jako **`sigma`**.

5.2.5 Dodatkowe funkcje

Program do obliczeń konstrukcji prętowych metodą elementów skończonych zawiera też inne, dodatkowe funkcje.

- Oblicza sumy rzutów sił na osie X i Y oraz sumę momentów względem początku układu współrzędnych – jest to pomocne przy sprawdzaniu poprawności obliczeń i szukaniu błędów.
- Oblicza masę konstrukcji oraz funkcję celu odpowiadającą rozwiązywanemu w dalszej części problemowi optymalizacji. Funkcję celu wykorzystuje algorytm ławicowy zastosowany w procesie optymalizacji konstrukcji. Jej budowa szczegółowo zostanie opisana w następnych rozdziałach.
- Może generować schematyczny rysunek konstrukcji przed i po obciążeniu.

Funkcja celu (fragment kodu z załącznika nr 2):

Z poniższego kodu można odczytać jaka została przyjęta postać funkcji celu.

```
//-------------funkcja celu - minimalna masa
//funkcja kary - przekroczenie naprezen dopuszczalnych
        g1=0;     //funkcja kary
        for i=1:ne,
            ge=max([0,(abs(sigma(i))-sigmadop)/sigmadop]);
//funkcja kary dla i-tego elementu
            //ge=max([0,(abs(sigma(i))-sigmadop)]);
            g1=g1+ge;
        end
//funkcja kary - przekroczenie przemieszczen dopuszczalnych
        g2=0;     //funkcja kary
        for i=1:2*nw,
            ge=max([0,(abs(U(i))-u_dop)/u_dop]); //funkcja
kary dla i-tego elementu

            g2=g2+ge;
        end

//maksymalna mozliwa masa konstrukcji
        M_max=0; //maksymalna masa

        for i=1:ne,
            M_max_p=L(i)*Amax*ro;
            M_max=M_max+M_max_p; //maksymalna masa i-tego
elementu
        end

//funkcja celu
        fcelu=(M/M_max)+(kara*g1)+(kara2*g2);
```

Funkcja celu składa się z trzech członów:

`M/Mmax` – jest to stosunek obliczonej masy konstrukcji, do masy maksymalnej. Przez masę maksymalną rozumiana jest masa konstrukcji zbudowanej tylko z elementów o dopuszczalnym maksymalnym przekroju Amax.

`kara*g1` – jest to funkcja kary g1 pomnożona przez odpowiedni współczynnik. Funkcja g1 odpowiada za przekroczenie dopuszczalnych naprężeń. Gdy naprężenia dopuszczalne w którymś elemencie zostają przekroczone, wartość funkcji wzrasta i zwiększa tym samym wartość funkcji celu.

`kara*g2` – jest to funkcja kary g2 pomnożona przez odpowiedni współczynnik. Funkcja g2 odpowiada za przekroczenie dopuszczalnych przemieszczeń. Gdy przemieszczenia dopuszczalne któregoś węzła zostają przekroczone, wartość funkcji wzrasta i zwiększa tym samym wartość funkcji celu.

Rysunek konstrukcji przed i po obciążeniu

Dodając odpowiednie komendy do kodu modułu MES (który znajduje się w załączniku nr 2) można tworzyć schematyczne rysunki konstrukcji przed i po obciążeniu. Docelowo polecenie te jest zamieszczone w kodzie algorytmu PSO (załącznik nr 5). Fragment kodu poniżej.

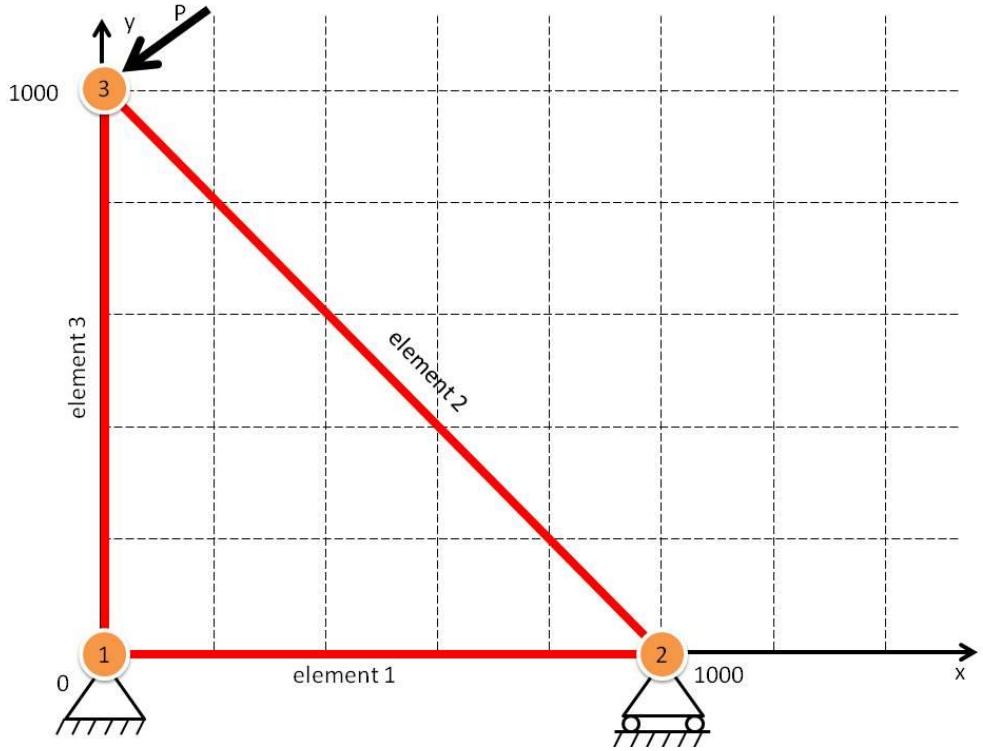
```
//Rysunek konstrukcji przed obciazeniem
clf();
        xset("window",0)
        for i=1:ne,
            x=[nc(mp(i,1),1),nc(mp(i,2),1)];
            y=[nc(mp(i,1),2),nc(mp(i,2),2)];
            plot2d(x,y,style=3,axesflag=0, frameflag=1,
rect=[-200,-200,max(nc)+200,max(nc)+200]);
            plot2d(x,y,style=-4, frameflag=1, rect=[-200,-
200,max(nc)+200,max(nc)+200]);
            xtitle("Schemat konstrukcji:")

        end
//Rysynek konstrukcji po odkszztalceniu
        ncp=nc; //macierz wspolrzednych wezlow po obciazeniu
konstrukcji
        UX=U; //wektor przemieszczen w kierunku X
        UY=U; //wektor przemieszczen w kierunku Y
        [lw, lc]=size(U); //lw - liczba wierszy, lc - liczba
kolumn
```

```
        for i=lw:-1:1,

                if modulo(i,2)==0 then
                    UX(i)=[];
                else
                    UY(i)=[];
                end
            end

        ncp=nc+[UX,UY]; //aktualizacja wspolrzednych

        xset("window",0)
        for i=1:ne,
            x=[ncp(mp(i,1),1),ncp(mp(i,2),1)];
            y=[ncp(mp(i,1),2),ncp(mp(i,2),2)];
            plot2d(x,y,style=26,axesflag=0, frameflag=1,
rect=[-200,-200,max(nc)+200,max(nc)+200]);
            plot2d(x,y,style=-3, frameflag=1, rect=[-200,-
200,max(nc)+200,max(nc)+200]);
            xtitle("Schemat konstrukcji:")

        end
```

Tworzenie rysunku zostało zamieszczone w kodzie algorytmu PSO aby nie tworzyć niepotrzebnie rysunków w każdej iteracji pętli optymalizacyjnej. Dzięki temu rysunki są tworzone raz. Rysunek konstrukcji przed obciążeniem i na koniec rysunek konstrukcji optymalnej po obciążeniu. Jednak jeśli nie używamy modułu MES wraz z algorytmem PSO można powyższe fragmenty kodu przenieść do modułu MES.

5.3 Przykład obliczeniowy

Działanie procedury zilustrowano na przykładzie prostej konstrukcji przedstawionej na rysunku 5.6.

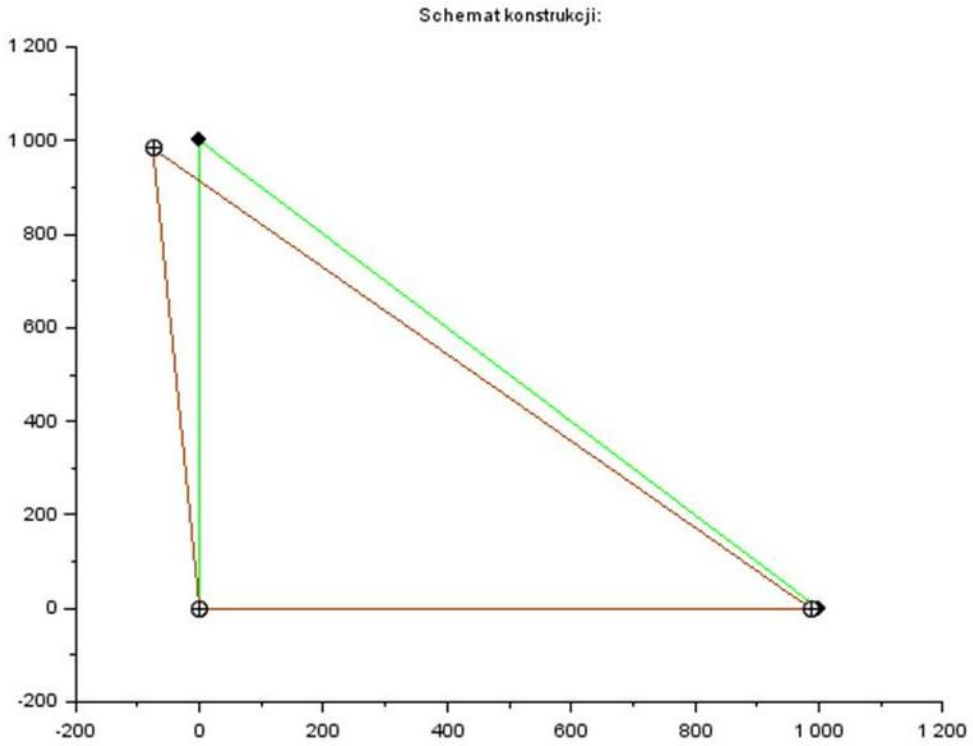


Rysunek 5.6 Przykład obliczeniowy

Do pliku z parametrami konstrukcji wprowadzono dane opisujące przykład. Kod pliku został podany w załączniku nr 3.

Ponieważ program wykonujący obliczenia MES został napisany jako funkcja, potrzebne było napisanie prostego skryptu wczytującego dane oraz uruchamiającego moduł MES. Kod skryptu zawiera załącznik nr 4.

Wyniki przeprowadzonych obliczeń przedstawiono graficznie na rys. 5.7.



Rysunek 5.7 Schemat konstrukcji przed i po obciążeniu – wygenerowany przez program

Rysunek 5.7 przedstawia schemat konstrukcji przed obciążeniem (kolor zielony) oraz schemat konstrukcji po obciążeniu (kolor brązowy).

Numeracja węzłów oraz elementów skończonych przedstawiona została na rysunku 5.6.

Rozwiązanie równania (5.14) przedstawiono poniżej. Struktura macierzy i wektorów wynika z przyjętych oznaczeń konstrukcji (rysunek 5.6).

Globalna macierz sztywności:

$$KP = \begin{bmatrix} 65{,}60 & 0 & -65{,}60 & 0 & 0 & 0 \\ 0 & 395{,}93 & 0 & 0 & 0 & -395{,}93 \\ -65{,}60 & 0 & 321{,}14 & -255{,}54 & -255{,}54 & 255{,}54 \\ 0 & 0 & -255{,}54 & 255{,}54 & 255{,}54 & -255{,}54 \\ 0 & 0 & -255{,}54 & 255{,}54 & -255{,}54 & -255{,}54 \\ 0 & -395{,}93 & 255{,}54 & -255{,}54 & -255{,}54 & 651{,}47 \end{bmatrix} \tag{5.21}$$

Wektor przemieszczeń w węzłach:

$$UP = \begin{bmatrix} 0 \\ 0 \\ -107{,}77 \\ 0 \\ -171{,}15 \\ -35{,}71 \end{bmatrix} \tag{5.22}$$

Wektor sił (składowe na poszczególnych kierunkach przemieszczeń – oznaczenia kierunków jak w wektorze UP):

$$FP = \begin{bmatrix} 7070 \\ 14140 \\ 0 \\ -7070 \\ -7070 \\ -7070 \end{bmatrix} \tag{5.23}$$

Wektor naprężeń normalnych w prętach (nr wiersza to nr elementu):

$$\sigma = \begin{bmatrix} -1077{,}75 \\ 138{,}33 \\ -357{,}13 \end{bmatrix} \tag{5.24}$$

Sprawdzenie poprawności obliczeń

Suma rzutów sił na oś X:

$$\sum F_x = -1{,}4 \cdot 10^{-10} \approx 0 \tag{5.25}$$

Suma rzutów sil na oś Y:

$$\sum F_x = -2{,}37 \cdot 10^{-11} \approx 0 \tag{5.26}$$

Suma momentów (względem początku układu współrzędnych:

$$\sum M_0 \approx 0 \tag{5.27}$$

Masa konstrukcji:

$$M = 1{,}16 \tag{5.28}$$

Wyniki otrzymane w tym przykładzie celowo zostały podane bez jednostek. Program wykonuje obliczenia na liczbach, jednostki nie mają znaczenia podczas wykonywania operacji matematycznych. Jednostki wyników obliczeń zależą tylko od sposobu definicji zmiennych – od tego jakie jednostki definiowanych zmiennych przyjmie użytkownik. Np. zapisanie sił w [N], pól przekroju w [mm^2] powoduje, że naprężenia będą wyrażone w [MPa] itd. Należy jednak uważać aby zastosowany układ jednostek był spójny dla całej konstrukcji i całości obliczeń.

Zaprezentowany tu przykład oraz inne zostały sprawdzone „ręcznie” przez autora pracy. Wszystkie obliczenia zostały wykonane samodzielnie i porównane z wynikami otrzymanymi przez program MES w Scilabie. We wszystkich przypadkach wyniki obliczeń zgadzały się. Na tej podstawie stwierdzono, że program działa poprawnie.

6 Program PSO w Scilab'ie

Kod algorytmu ławicowego (PSO) został napisany w środowisku Scilab. Można go wykorzystywać do optymalizacji dowolnej funkcji celu. Docelowo program PSO zbudowano tak, aby pracował w połączeniu z modułem MES (opisanym w poprzednim rozdziale). W dalszej części rozdziału algorytm zostanie dokładniej opisany oraz zostanie zaprezentowane jego działanie na wybranym przykładzie. Kod programu został zamieszczony w załączniku nr 5.

6.1 Definicja zmiennych i struktura danych

6.1.1 Parametry algorytmu PSO

Poniżej został przedstawiony fragment kodu (załącznik nr 5), zawierający definicję zmiennych algorytmu PSO.

Komentarze w kodzie opisują każdy z parametrów. Dokładne znaczenie i rola poszczególnych parametrów przedstawione zostały w rozdziale 3, poświęconym algorytmowi PSO.

```
//parametry PSO
    wmax=0.9; // maksymalna (poczatkowa) wartość współczynnika
wagi inercji
    wmin=0.4; // minimalna (koncowa) wartość współczynnika
wagi inercji
    itmax=10; // maksymalna liczba iteracji
    c1=1.3; // ustalony mnoznik wagowy (przy czlonie z
najlepszym dotychczas znalezionym polozeniem czastki)
    c2=2.8; // ustalony mnoznik wagowy (przy czlonie z
najlepszym dotychczas znalezionym polozeniem lidera roju)
    N=10; // liczba czastek
    D=ne; // wymiar problemu (musi byc zgodny z definicja
optymalizowanej funkcji)
    x_sup=ones([1:D])
    x_inf=ones([1:D])
    x_sup=Amin*x_sup'; // minimalna wartosc polozenia (uzywana
do generowania poczatkowej populacji czastek)
    x_inf=Amax*x_inf'; // maksymalna wartosc polozenia
(uzywana do generowania poczatkowej populacji czastek)
    v_max=ones([1:D])
    v_max=500*v_max'; // maksymalna wartosc predkosci (uzywana
do generowania poczatkowych predkosci czastek)
    v_min=-1*v_max; // minimalna wartosc predkosci (uzywana do
generowania poczatkowych predkosci czastek)
```

6.1.2 Struktura danych

Poniższy kod (fragment załącznika nr 5) przedstawia zmienne, które wykorzystuje algorytm PSO.

```
//deklaracje jawne zmiennych
    W=zeros([1:itmax])  // wektor wspolczynnikow wag
    F0=zeros([1:N])   // pierwsze obliczone wartosci funkcji
celu
    x1=zeros(N,D,itmax)   //  obliczone wektory pozycji
(ostatni wymiar to nr iteracji)
    v1=zeros(N,D,itmax)   // obliczone wektory predkosci
(ostatni wymiar to nr iteracji)
    F1=zeros(N,itmax)   // obliczone wartosci funkcji celu
(ostatni wymiar to nr iteracji)

    gbest1=zeros(D,itmax)  // pomocnicza do wartosci x
    G1=zeros(N,D,itmax)   // najlepsze globalne (wartosci x)
    Fbest1=zeros([1:itmax])   // najlepsze globalne (wartosci
funckji)
    pbest1=zeros(N,D,itmax)   // najlepsze rozwiazanie
pojedynczego osobnika (wartosci x)
    Fb1=zeros([1:itmax])   // najlepsze rozwiazanie w danej
iteracji (wartosc funkcji)

//  obliczenie wartosci wektora wspolczynnikow wagi inercji
dla wszystkich iteracji (shi & eberhart 1998)

    for j=1:itmax
        W(j)=wmax-((wmax-wmin)/itmax)*j;
    end
```

W programie wykorzystano następującą strukturę danych:

`W` – jest wektorem wymiaru `1 x itmax`. Jego elementami są współczynniki wagi, wykorzystywane w kolejnych iteracjach do obliczenia nowego położenia. Wartości wektora `W` obliczana jest według wzoru 3.4 (rozdział 3.2.1).

`F0` – jest to pierwsza obliczona wartość funkcji celu dla każdej cząstki. Jest to wektor wymiaru `1 x N`.

`x1` – jest to obiekt trójwymiarowy `N x D x itmax`. Składa się z macierzy (o wymiarze `N x D`) zawierających wektory pozycji każdej z cząstek. Liczba macierzy wynosi `itmax`.

`v1` – jest to obiekt analogiczny do `x1`, tylko zamiast położenia cząstek zawiera ich prędkości.

F1 – jest macierzą o wymiarach N x itmax. Zawiera wartości funkcji celu każdej z cząstek w poszczególnych iteracjach.

gbest1 – jest to pomocnicza wartość – macierz wymiaru D x itmax. Zapisywane są w niej współrzędne lidera roju dla każdej iteracji.

G1 – obiekt wymiaru N x D x itmax. Składa się z macierzy wymiaru N x D zawierających współrzędne cząstki dla najlepszego globalnego rozwiązania. Liczba macierzy wynosi itmax.

Fbest1 – wektor wymiaru 1 x itmax. Zawiera najlepsze globalne wartości funkcji celu w poszczególnych iteracjach.

pbest1 – jest obiektem trójwymiarowym, wymiaru N x D x itmax. Zawiera on współrzędne najlepszego dotychczas znalezionego położenia poszczególnych cząstek.

Fb1 –zmienna, w której zapisywana jest najlepsza wartość funkcji celu w danej iteracji. Jest to wektor wymiaru 1 x itmax.

6.2 Opis głównych procedur

6.2.1 Tryb programu – zmienne ciągłe, zmienne dyskretne

W pierwszej kolejności zostanie opisany sposób, w jaki program realizuje tryb optymalizacji stosujący „zmienne ciągłe” oraz „zmienne dyskretne”, nie wnikając na razie w szczegóły jak program oblicza wartości współrzędnych cząstek.

Program PSO wykorzystuje wartość zmiennej tryb do określenia, czy będzie pracował na zmiennych ciągłych czy zmiennych dyskretnych. Przed uruchomieniem procedury należy określić wartość tej zmiennej (opis w kodzie).

Zmienna tryb służy do sterowania instrukcją warunkową if. Instrukcja warunkowa if została zastosowana w tych miejscach kodu, gdzie obliczane są wartości współrzędnych cząstek.

Poniższy kod jest fragmentem kodu programu algorytmu PSO (załącznik nr 5).

```
//-------------------------------
//Okreslenie trybu optymalizacji
//---do wyboru
//------zmienne ciagle (zmienne przyjmuja wartosc miedzy
zadeklarowanym minimum i maksimum)
//------zmienne dyskretne (zmienne przyjmuja tylko wybrane
wartosci zadeklarowane w wektorze - katalogu zmiennych)
    tryb=0; //zmienne ciagle
    //tryb=1; //zmienne dyskretne
```

W przypadku rozwiązywania zadania optymalizacji zdefiniowanego dla zmiennych ciągłych program oblicza wartości współrzędnych według określonego wzoru algorytmu PSO. Dodatkowo na zmienne ciągłe nałożone są ograniczenia w postaci ich wartości minimalnej i maksymalnej. W kolejnej instrukcji warunkowej `if` sprawdzane jest, czy wartość współrzędnej mieści się w zadanych granicach. Jeśli wartość przekracza wartość maksymalną, to przypisywana jest jej zdefiniowana wartość maksymalna. Analogicznie sprawa ma się z wartością minimalną. Jeśli obliczona wartość jest mniejsza od minimalnej, to staje się wartością minimalną.

W przypadku zadania optymalizacji sformułowanego dla zmiennych dyskretnych procedura wygląda nieco inaczej. W odpowiednim wektorze znajdującym się w pliku z parametrami konstrukcji (w tym przypadku nosi on nazwę `katalogA`) zdefiniowane są dopuszczalne wartości przekrojów (skończona liczba) pełniące tutaj rolę współrzędnych. Po wyznaczeniu wartości współrzędnej przez algorytm PSO program zaczyna sprawdzać, od którego pierwszego i najmniejszego elementu z katalogu jest większa obliczona współrzędna wyznaczona przez PSO. Realizowane jest to w pętli `while`. Dzięki temu program ma informację między którymi dwoma sąsiednimi elementami z katalogu znajduje się obliczona współrzędna. Następnie obliczana jest różnica między każdą aktualną współrzędną wektora pozycji a pierwszym większym i mniejszym elementem z katalogu. Przyjęta jest ta wartość pozycji z katalogu, dla którego różnica jest mniejsza. Odpowiada to zaokrągleniu wygenerowanej współrzędnej do najbliższej dopuszczalnej wartości z katalogu.

Poniżej przedstawiono fragment kodu prezentujący procedurę wyboru obliczeń dla zmiennych ciągłych i dyskretnych (załącznik nr 5).

```
//Obliczenie predkosci i polozenia dla nastepnych iteracji
        if tryb==0 then //dla zmiennych ciaglych

v1(1:N,1:D,j+1)=W(j)*v1(1:N,1:D,j)+c1*rand()*(pbest1(1:N,1:D,j
)-x1(1:N,1:D,j))+c2*rand()*(G1(1:N,1:D,j)-x1(1:N,1:D,j));
//nowa predkosc
               x1(1:N,1:D,j+1)=x1(1:N,1:D,j)+v1(1:N,1:D,j+1);
//nowe polozenie

                    for q=1:N //uwzglednienie zdefiniowanego
maksymalnego i minimalnego przekroju
                        for r=1:D
                            if x1(q,r,j+1)>Amax then
                                x1(q,r,j+1)=Amax; //jesli
obliczone x1 wieksze od Amax, to staje sie Amax
                            elseif x1(q,r,j+1)<Amin then
                                x1(q,r,j+1)=Amin; //jesli
obliczone x1 mniejsze od Amin, to staje sie Amin
                            else //jesli nie spelnione
poprzednie warunki to nic nie robimy
                            end
                        end
                    end

        else //dla zmiennych dyskretnych

v1(1:N,1:D,j+1)=W(j)*v1(1:N,1:D,j)+c1*rand()*(pbest1(1:N,1:D,j
)-x1(1:N,1:D,j))+c2*rand()*(G1(1:N,1:D,j)-x1(1:N,1:D,j));
//nowa predkosc
        x1(1:N,1:D,j+1)=x1(1:N,1:D,j)+v1(1:N,1:D,j+1); //
nowe polozenie

       dimA=size(katalogA);
       dimA=dimA(1,2);
        for q=1:N //dobor wartosci z katalogu
            for r=1:D
                f=1;
                if x1(q,r,j+1)>katalogA(1)
                    if x1(q,r,j+1)>katalogA(dimA)

                        x1(q,r,j+1)=katalogA(dimA);
                    else
                        while katalogA(f)<x1(q,r,j+1)
                            f=f+1;
                        end
                        a1=abs(x1(q,r,j+1)-katalogA(f));
                        a2=abs(x1(q,r,j+1)-katalogA(f-1));
```

```
                                if a1>a2
                                    x1(q,r,j+1)=katalogA(f-1);
//podstawienie wartosci z katalogu
                                else
                                    x1(q,r,j+1)=katalogA(f);
//podstawienie wartosci z katalogu
                                end
                            end

                        else
                            x1(q,r,j+1)=katalogA(f); //podstawienie
wartosci z katalogu
                        end
                    end
                end

            end
```

6.2.2 Generowanie początkowego roju cząstek

Wzór 6.1 przedstawia ideę obliczania wartości współrzędnych z przedziału <x_sup:x_inf>. Wartość `rand` to liczba losowa z przedziału <0,1>. Zatem dla `rand=0` x1=x_inf, dla `rand=1`, x1=x_sup.

$$x1 = x\_inf + rand(x\_sup - x\_inf) \tag{6.1}$$

Fragment kodu (załącznik nr 5) przedstawia sposób generowania początkowych pozycji oraz prędkości każdej z cząstek.

```
x1(1:N,i,j) =x_inf(i) +rand(N,1) * ( x_sup(i) - x_inf(i) );
// pierwszy wektor pozycji

v1(1:N,i,j)=v_min(i)+(v_max(i)-v_min(i))*rand(N,1);
// pierwszy wektor predkosci
```

Według analogicznego wzoru obliczane są wartości prędkości początkowych cząstek.

6.2.3 Pętla optymalizacyjna

Proces optymalizacji realizowany jest w pętli `while`. Pętla `while` pozwala na powtarzanie ciągu instrukcji w niej zawartych do momentu aż prawdziwy jest warunek zakończenia iteracji. W tym przypadku jest to osiągnięcie maksymalnej liczby iteracji.

Poniżej zostaną opisane procedury zawarte w pętli optymalizacji. Wszystkie zamieszczone fragmenty kodu pochodzą z załącznika nr 5.

Pierwszym krokiem jest obliczenie wartości funkcji celu.

Wartość funkcji celu obliczana jest dla każdej z cząstek przy wykorzystaniu zewnętrznej funkcji, która zwraca odpowiednią wartość. W pokazanym poniżej przykładzie wywoływany jest program do obliczeń konstrukcji prętowych metodą elementów skończonych, jednak może być to inna odpowiednio zaprogramowana funkcja.

```
//Obliczenie funkcji celu
    for i=1:N
        y1=x1(i,1:D,j);
        [fcelu, U, M, sigma, F, FX, FY,
T]=mes_module(y1,nw,ne,nc,mp,E,ro,FP,wb,sigmadop,kara,kara2,u_
dop);
        F1(i,j)=fcelu //F i-tej czastki w j-tej iteracji
    end
```

W dalszej kolejności (fragment kodu poniżej) następuje poszukiwanie minimum funkcji celu wśród osobników całego roju cząstek. Sprowadza się to do znalezienia w macierzy `F1` wartości minimalnej oraz pozycji, na której się ona znajduje. Do zmiennej `gbest1` zapisane zostają wartości współrzędnych, dla których została otrzymana wartość minimalna funkcji celu. W wektorze `Fbest1` zostaje zapisana wartość funkcji celu.

```
//Poszukiwanie minimum wsrod roju czastek
    [fmin, pos]=min(F1(1:N,j)) ; //  pos - to pozycja gdzie
minimum znalezione


//Poszukiwanie globalnego minimum

//zakladamy ze mamy lepszy wynik i go zapisujemy *
    gbest1(1:D,j)=x1(pos,1:D,j)'; // w pomocniczej global best
(wartosci x) *
    Fb1(j)=F1(pos,j); // albo =fmin
    Fbest1(j)=Fb1(j); // Fbest1 to najlepszy znany  *
```

Kolejnym krokiem jest sprawdzenie (kod zamieszczono poniżej), czy znalezione w bieżącej iteracji globalne rozwiązanie jest lepsze od wyniku z poprzednich iteracji. Procedura wykorzystuje instrukcję warunkową `if`. Jeśli otrzymany wynik (lidera roju) jest lepszy od poprzedniego, to zostaje zapamiętany jako aktualne najlepsze dotychczas znalezione położenie lidera roju. Jeśli jest gorszy, to najlepszym globalnym rozwiązaniem pozostaje wynik z poprzedniej iteracji (gbest1 – współrzędne cząstki, Fbest1 – wartość funkcji celu dla najlepszego rozwiązania).

```
//poprawiane zmienne jesli zalozenie nieprawdziwe

    if Fbest1(j)<Fbest1(j-1) //zalozenie spelnione - nic nie
zmianiamy
    else  // w przeciwnym wypadku
        gbest1(1:D,j)=gbest1(1:D,j-1); //zamiana wartosci
wspolrzednych (najlepsze globalne rozwiazanie)
        Fbest1(j)=Fbest1(j-1); // Fbest1 nie zostalo tym razem
znalezione
    end
```

Aby wyznaczyć nowe położenie cząstki program potrzebuje jeszcze informacji o jej dotychczasowym najlepszym położeniu. Ustalanie wartości `pbest1` realizowane jest w instrukcji warunkowej `if`. W pierwszej kolejności sprawdzany jest warunek, czy wartość funkcji celu danej cząstki w jej aktualnym położeniu (`F1`) jest lepsza od wartości `f_pbest`. W macierzy `f_pbest` są zapisywane wartości funkcji celu w punkcie `pbest1` w każdej z iteracji dla każdej z cząstek. Jeśli warunek jest spełniony, to `pbest1` staje się aktualnym położeniem cząstki `x1`. Wartość `f_pbest` staje się wartością funkcji celu dla danej cząstki w bieżącej iteracji. Jeśli warunek nie jest spełniony, to aktualnymi wartościami `pbest1` oraz `f_pbest` zostają wartości z poprzedniej iteracji. Poniższy fragment kodu prezentuje sposób określania tej wartości.

```
//Obliczanie najlepszego rozwiazania dla danej czastki
(personal best)
for i=1:N

        if F1(i,j)<f_pbest(i,j-1)
            pbest1(i,1:D,j)=x1(i,1:D,j);
            f_pbest(i,j)=F1(i,j);
        else
            pbest1(i,1:D,j)=pbest1(i,1:D,j-1);
            f_pbest(i,j)=f_pbest(i,j-1);
        end

end
```

W ostatniej kolejności obliczana jest nowa pozycja każdej z cząstek oraz poszczególne prędkości (wg wzorów 3.1 oraz 3.2).

```
v1(1:N,1:D,j+1)=W(j)*v1(1:N,1:D,j)+c1*rand()*(pbest1(1:N,1:D,j
)-x1(1:N,1:D,j))+c2*rand()*(G1(1:N,1:D,j)-x1(1:N,1:D,j));
//nowa predkosc
x1(1:N,1:D,j+1)=x1(1:N,1:D,j)+v1(1:N,1:D,j+1); //nowe
polozenie
```

Wszystkie opisane powyżej procedury wykonywane są aż do momentu, kiedy zmienna `j` osiągnie wartość `itmax`.

Wynikiem optymalizacji jest obliczona wartość funkcji celu oraz komplet zmiennych (współrzędnych), dla których uzyskano najlepsze rozwiązanie.

6.2.4 Dodatkowe funkcje

Do kodu algorytmu PSO zostało wprowadzone kilka dodatkowych procedur, zwiększających funkcjonalność i użyteczność programu.

Aby przyspieszyć działanie procesu optymalizacji, tworzenie rysunku optymalizowanej konstrukcji prętowej (obliczanej przez moduł MES) zostało przeniesione do algorytmu PSO. Rysunki nie są zatem tworzone za każdym razem kiedy uruchamiany jest moduł MES. Tworzone są tylko jeden raz i przedstawiają rysunek konstrukcji przed obciążeniem oraz po zakończeniu procesu optymalizacji rysunek po obciążeniu (konstrukcji optymalnej).

Wyświetlane są również niektóre wartości obliczane przez moduł MES, takie jak np. sumy rzutów sił, momentów, wartości przemieszczeń, wartości naprężeń w elementach itp.

Program wyświetla też komunikaty ostrzegawcze o przekroczeniu naprężeń w elementach i przemieszczeń w węzłach (w razie zakończenia procesu optymalizacji niepowodzeniem).

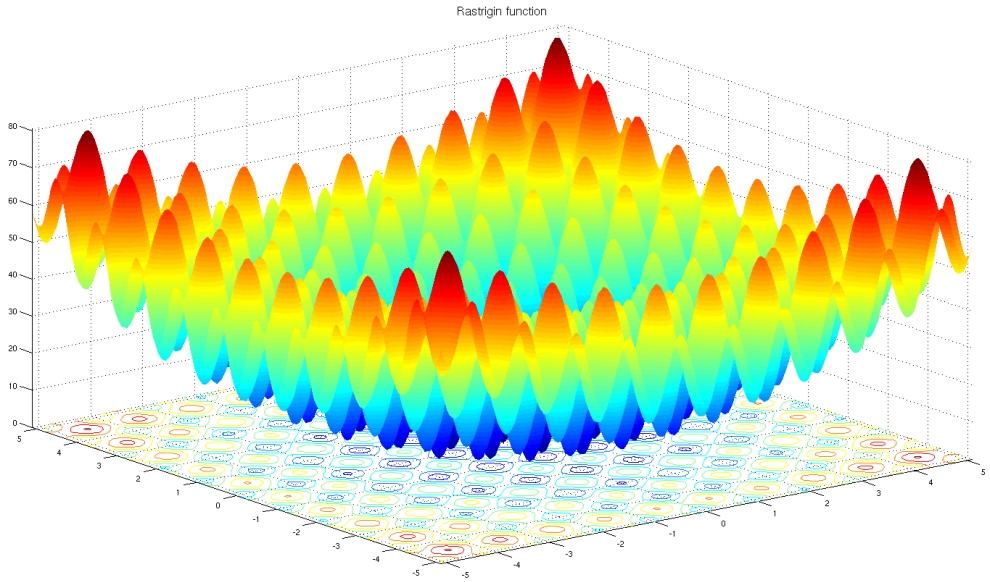
Dodatkowo wybrane wyniki, użyteczne do dalszych analiz zapisywane są w jednej macierzy. Ułatwia to eksport danych z Scilaba. Wszystkie wymagane wyniki znajdują się wtedy w jednym pliku. Fragment kodu poniżej (załącznik nr 5) prezentuję strukturę macierzy `wyniki`.

```
//zapisanie potrzebnych do dalszej analizy wynikow w jednej
macierzy

    wyniki=zeros(200,5);
    wyniki(1:ne,1)=gbest1(:,j); //1. kolumna - przekroje
    wyniki(1:ne,2)=sigma;       //2. kolumna - naprezenia
    wyniki(1:2*nw,3)=U;         //3. kolumna - przemieszczenia
    wyniki(1:itmax,4)=Fbest1';  //4. kolumna - wartosci
funkcji celu w poszczegolnych iteracjach
    wyniki(1,5)=M;              //5. kolumna - masa
konstrukcji
```

Program tworzy również wykres funkcji celu podając jej wartości w poszczególnych iteracjach oraz mierzy czas trwania obliczeń całego procesu optymalizacji.

6.3 Przykład obliczeniowy – test algorytmu PSO

Do przetestowania działania algorytmu ławicowego, wpływu parametrów na wyniki obliczeń użyto funkcji Rastrigin'a o dwóch zmiennych. Jest to typowy przykład testowy z optymalizacji globalnej, niezwykle trudny do rozwiązania za pomocą klasycznych metod optymalizacji ze względu na wielomodalny (wiele mimimuów lokalnych) charakter funkcji. Rysunek 6.1 przedstawia wykres funkcji Rastrigin'a o dwóch zmiennych.



Rysunek 6.1 Wykres funkcji Rastrigin'a o dwóch zmiennych [i5]

Ogólny wzór funkcji Rastrigin'a przedstawia się następująco:

$$f(x) = An + \sum_{i-1}^{n} \left[x_i^2 - A\cos(2\pi x_i)\right] \tag{6.2}$$

gdzie:

$$\begin{gathered} A = 10 \\ x_i \in [-5{,}12 : 5{,}12] \\ minimum\ lokalne - x = 0, f(x) = 0 \end{gathered} \tag{6.3}$$

Dla funkcji Rastrigin'a o dwóch zmiennych wartość n=2.

Definicję funkcji w programie Scilab prezentuje poniższy kod.

```
// Test: funkcja Rastrigina
function f=fun1(x)
f=20+sum((x.^2)-10*cos(2*%pi.*x));
endfunction;
```

Powyższa funkcja wywoływana jest w programie PSO w miejscach, gdzie wyznaczana jest wartość funkcji celu. W kolejnych podrozdziałach zaprezentowano wyniki przeprowadzonych obliczeń.

6.3.1 Wpływ liczby cząstek roju N

W niniejszym rozdziale zaprezentowano wpływ liczby cząstek na otrzymany wynik. Tabela 6.1 przedstawia parametry z jakimi uruchamiany był program PSO.

Tabela 6.1 Parametry algorytmu PSO

Parametry	wmax	wmin	itmax	c1	c2	N	D	x_sup	x_inf	v_max	v_min
Wartość	0,9	0,4	100	2	2	zmienna	2	10	200	10	-10

Obliczenia zostały wykonane dla roju składającego się z 10, 50 oraz 250 cząstek.

Wyniki obliczeń dla N=10 przedstawiono w tabeli 6.2.

Tabela 6.2 Wyniki obliczeń dla N=10

	Wartość funkcji celu	Współrzędne	
Nr uruchomienia	Fbest1	x1	x2
1	7,959664800	-1,989894700	1,989803200
2	1,989926300	-0,994764000	0,994899600
3	4,017828200	-2,003493200	0,002689000
4	1,989941500	-0,994633200	-0,995067400
5	0,994987200	-0,000062400	-0,994586800
6	0,994960100	0,000055100	0,994909900
7	0,995029100	-0,000066500	-0,994368000
8	1,990061100	-0,995542700	-0,995574900
9	1,989918100	-0,994958600	-0,994959000
10	0,994967700	-0,000200800	0,994903400
Wartość średnia	2,391728410	-0,797356100	0,000264900

Wyniki obliczeń dla N=50 przedstawiono w tabeli 6.3.

Tabela 6.3 Wyniki obliczeń dla N=50

	Wartość funkcji celu	Współrzędne	
Nr uruchomienia	Fbest1	x1	x2
1	0,994959200	-0,994976000	0,000013200
2	1,989918100	0,994951900	0,994950300
3	0,000000400	0,000024100	0,000038100
4	4,974790300	-1,989926100	0,994964600
5	0,994959100	-0,000000022	-0,994958000
6	0,000000000	0,000000005	0,000000007
7	0,994959100	0,000000077	-0,994956900
8	0,994967600	0,000131900	-0,994799000
9	0,994995900	0,000332600	0,994684300
10	0,994959100	0,000000300	0,994959200
Wartość średnia	1,293450880	-0,198946124	0,099489581

Wyniki obliczeń dla N=250 przedstawiono w tabeli 6.4.

Tabela 6.4 Wyniki obliczeń dla N=250

	Wartość funkcji celu	Współrzędne	
Nr uruchomienia	Fbest1	x1	x2
1	0,000005900	-0,000138600	-0,000103200
2	0,994959100	-0,994950500	-0,000006900
3	0,994959900	-0,000018600	-0,994897200
4	0,000000000	-0,000000200	0,000000093
5	0,000000078	0,000013500	0,000014600
6	0,000004000	-0,000081700	0,000115300
7	0,000000000	0,000000053	-0,000000100
8	0,000000008	-0,000004400	-0,000004800
9	0,000000002	0,000001000	-0,000002600
10	0,000000000	-0,000000027	-0,000000200
Wartość średnia	0,198992899	-0,099517947	-0,099488501

Można zauważyć, że liczba cząstek N wpływa na dokładność rozwiązania.

Im większa liczba cząstek, tym dokładniejsze, bliższe analitycznemu rozwiązanie (średnia) można uzyskać.

Ponieważ w algorytmie występuje pewna losowość, wyniki obliczeń dla tego samego zestawu parametrów przy każdym kolejnym uruchomieniu algorytmu mogą się od siebie różnić, czasem nawet znacznie.

Zwiększanie liczby cząstek roju wydłuża czas obliczeń. Przy analizowanym tu prostym zadaniu nie jest to zauważalne, jednak przy bardziej skomplikowanych problemach może znacznie wydłużać czas obliczeń.

6.3.2 Wpływ liczby iteracji

W rozdziale tym zaprezentowano wpływ liczby iteracji na otrzymany wynik. Tabela 6.5 przedstawia parametry z jakimi uruchamiany był program PSO.

Tabela 6.5 Parametry algorytmu PSO

Parametry	wmax	wmin	itmax	c1	c2	N	D	x_sup	x_inf	v_max	v_min
Wartość	0,9	0,4	zmienna	2	2	100	2	10	200	10	-10

Obliczenia zostały wykonane dla maksymalnej liczby iteracji równej 50, 100 oraz 150. Wyniki obliczeń dla itmax=50 przedstawiono w tabeli 6.6.

Tabela 6.6 Wyniki obliczeń dla itmax=50

	Wartość funkcji celu	Współrzędne	
Nr uruchomienia	Fbest1	x1	x2
1	1,016212200	-0,010294700	0,993870400
2	0,079073900	0,002952100	0,019757500
3	1,746865500	0,001361900	-1,056782000
4	0,995146600	0,000819500	0,994435500
5	1,537900100	0,026650200	-0,949693400
6	0,020053500	0,008519300	0,005340500
7	0,015215700	0,008444500	-0,002324300
8	0,006858100	-0,001940800	0,005550200
9	0,136063000	0,025399800	0,006484200
10	1,025799300	-0,995306200	0,012466300
Wartość średnia	0,657918790	-0,093339440	0,002910490

Wyniki obliczeń dla itmax=100 przedstawiono w tabeli 6.7.

Tabela 6.7 Wyniki obliczeń dla itmax=100

	Wartość funkcji celu	Współrzędne	
Nr uruchomienia	Fbest1	x1	x2
1	0,000545800	0,000417500	0,001605200
2	0,000001300	0,000036500	0,000073600
3	0,994959100	0,000003000	0,994947900
4	0,000000028	0,000007800	0,000008900
5	0,000000000	-0,000000049	-0,000000010
6	0,000000014	-0,000005200	-0,000006600
7	0,007766700	0,005181100	0,003508200
8	0,994959100	0,994958100	-0,000000100
9	0,000000000	-0,000000300	0,000000700
10	0,000000000	0,000001200	0,000000400
Wartość średnia	0,199823204	0,100059965	0,100013819

Wyniki obliczeń dla itmax=150 przedstawiono w tabeli 6.8.

Tabela 6.8 Wyniki obliczeń dla itmax=150

	Wartość funkcji celu	Współrzędne	
Nr uruchomienia	Fbest1	x1	x2
1	0,000000000	-0,000000001	-0,000000006
2	0,000000000	0,000000032	-0,000000003
3	0,000000000	0,000000000	0,000000004
4	0,000000000	-0,000000100	0,000000048
5	0,994959100	0,000000002	0,994958600
6	0,000000000	0,000000000	-0,000000004
7	0,000000000	0,000000031	0,000000045
8	0,000000000	0,000000000	-0,000000002
9	4,974790400	1,989885100	-0,994951500
10	0,994959100	0,994958700	-0,000000100
Wartość średnia	0,696470860	0,298484376	0,000000708

Na podstawie wartości średniej z 10 uruchomień można sądzić, że zwiększenie liczby iteracji pozwala uzyskać dokładniejsze wyniki.

Jednak dla itmax=150, uruchomienie nr 9, otrzymany wynik odbiega od reszty o kilka rzędów. Powodem jest losowość występująca w algorytmie. Można go odrzucić przy obliczaniu wartości średniej Fbest1.

Zwiększanie liczby iteracji prowadzi do uzyskiwania lepszych wyników, ale również wydłuża czas obliczeń. Można również przekroczyć jakieś graniczne itmax,

przy którym dalsze iteracje nie będą prowadziły do uzyskania jeszcze dokładniejszego wyniku.

6.3.3 Wpływ parametrów c1, c2

Zaprezentowane zostaną tu wyniki wpływu parametrów c1 oraz c2 na działanie algorytmu.

Współczynnik wagowy c1 - stoi przy członie z najlepszym dotychczasowym wynikiem pojedynczej cząstki. Współczynnik c2 stoi przy członie z najlepszym dotychczasowym wynikiem globalnym roju. Współczynniki określają więc jaki wpływ na dalsze zachowanie się cząstki (w którym kierunku podąży) ma jej dotychczasowy najlepszy wynik i wynik lidera roju.

Tabela 6.9 przedstawia parametry z jakimi uruchamiany był program PSO.

Tabela 6.9 Parametry algorytmu PSO

Parametry	wmax	wmin	itmax	c1	c2	N	D	x_sup	x_inf	v_max	v_min
Wartość	0,9	0,4	100	zmienna	zmienna	100	2	10	200	10	-10

Wyniki obliczeń dla c1=c2=2 przedstawiono w tabeli 6.10.

Tabela 6.10 Wyniki obliczeń dla c1=c2=2

	Wartość funkcji celu	Współrzędne	
Nr uruchomienia	Fbest1	x1	x2
1	0,000545800	0,000417500	0,001605200
2	0,000001300	0,000036500	0,000073600
3	0,994959100	0,000003000	0,994947900
4	0,000000028	0,000007800	0,000008900
5	0,000000000	-0,000000049	-0,000000010
6	0,000000014	-0,000005200	-0,000006600
7	0,007766700	0,005181100	0,003508200
8	0,994959100	0,994958100	-0,000000100
9	0,000000000	-0,000000300	0,000000700
10	0,000000000	0,000001200	0,000000400
Wartość średnia	0,199823204	0,100059965	0,100013819

Wyniki obliczeń dla c1=1, c2=3 przedstawiono w tabeli 6.11.

Tabela 6.11 Wyniki obliczeń dla c1=1, c2=3

	Wartość funkcji celu	Współrzędne	
Nr uruchomienia	Fbest1	x1	x2
1	0,000028500	0,000164300	-0,000341300
2	0,015349900	0,007493400	-0,004607900
3	36,812951000	-5,969590700	0,994946600
4	0,000156000	-0,000325100	0,000825100
5	0,000132800	0,000488600	-0,000656300
6	0,107463000	0,018520900	-0,014112400
7	2,041803900	-0,038637400	0,932852900
8	27,474038000	-4,030684100	2,878993600
9	2,202523000	-0,964737700	1,007740200
10	0,996042800	0,002155100	0,995863300
Wartość średnia	6,965048890	-1,097515270	0,679150380

W powyższym przypadku większy wpływ na ruch cząstki ma globalny wynik w roju cząstek.

Widać, że rozrzut otrzymanych wyników w porównaniu, gdy c1 i c2 były sobie równe jest większy.

Gdy c1 i c2 były sobie równe, otrzymane wyniki były do siebie zbliżone. Tu zdarzają się bardzo duże błędy (nr 3 i 8).

Wyniki obliczeń dla c1=3, c2=1 przedstawiono w tabeli 6.12.

Tabela 6.12 Wyniki obliczeń dla c1=3, c2=1

	Wartość funkcji celu	Współrzędne	
Nr uruchomienia	Fbest1	x1	x2
1	0,994959100	-0,000000001	0,994958600
2	0,000000000	-0,000000026	-0,000000052
3	0,994959100	0,994958600	-0,000000021
4	0,000004700	-0,000009600	0,000153600
5	0,000064700	-0,000027600	-0,000570500
6	0,994959100	0,994958600	0,000000002
7	0,000000000	-0,000000001	0,000000007
8	0,000000000	-0,000000200	-0,000000001
9	0,000000000	0,000000007	0,000000003
10	0,000000000	0,000000095	-0,000000068
Wartość średnia	0,298494670	0,198987987	0,099454157

W tym przypadku rozrzut wyników jest mniejszy.

6.3.4 Wpływ współczynnika inercji

Współczynnik inercji odpowiada za wpływ poprzedniej prędkości cząstki na kolejną obliczaną jej prędkość.

Tabela 6.13 przedstawia parametry z jakimi uruchamiany był program PSO.

Tabela 6.13 Parametry algorytmu PSO

Parametry	wmax	wmin	itmax	c1	c2	N	D	x_sup	x_inf	v_max	v_min
Wartość	zmienna	zmienna	100	2	2	100	2	10	200	10	-10

Wyniki obliczeń dla wmax=1,4, wmin=1 przedstawiono w tabeli 6.14.

Tabela 6.14 Wyniki obliczeń dla wmax=1,4, wmin=1

	Wartość funkcji celu	Współrzędne	
Nr uruchomienia	Fbest1	x1	x2
1	180,779000000	-4,718709800	-11,376459000
2	50,075790000	4,465908900	0,753302400
3	33,126267000	0,230230500	3,305027600
4	47,324034000	-5,090857200	4,122649300
5	30,647783000	-0,751530800	2,328875500
6	123,023370000	4,789404900	-8,565145000
7	17,200065000	-2,194307800	0,831088600
8	2,842234200	-1,091581300	0,014259100
9	23,148464000	0,111447400	-3,811024500
10	42,403088000	-1,799886000	4,755920800
Wartość średnia	55,057009520	-0,604988120	-0,764150520

Widać, że gdy współczynnik inercji jest większy od jedności, wyniki obliczeń są praktycznie nieprzydatne, są obarczone bardzo dużym błędem. Czasami trafia się dokładniejsze rozwiązanie (np. nr 8), ale wynika to z losowości, którą zawiera PSO.

Wyniki obliczeń dla wmax=0,9, wmin=0,4 przedstawiono w tabeli 6.15.

Tabela 6.15 Wyniki obliczeń dla wmax=0,9, wmin=0,4

	Wartość funkcji celu	Współrzędne	
Nr uruchomienia	Fbest1	x1	x2
1	0,000545800	0,000417500	0,001605200
2	0,000001300	0,000036500	0,000073600
3	0,994959100	0,000003000	0,994947900
4	0,000000028	0,000007800	0,000008900
5	0,000000000	-0,000000049	-0,000000010
6	0,000000014	-0,000005200	-0,000006600
7	0,007766700	0,005181100	0,003508200
8	0,994959100	0,994958100	-0,000000100
9	0,000000000	-0,000000300	0,000000700
10	0,000000000	0,000001200	0,000000400
Wartość średnia	0,199823204	0,100059965	0,100013819

Wyniki obliczeń dla wmax=0,75, wmin=0,75 (w=const) przedstawiono w tabeli 6.16.

Tabela 6.16 Wyniki obliczeń dla wmax=0,75, wmin=0,75 (w=const)

	Wartość funkcji celu	Współrzędne	
Nr uruchomienia	Fbest1	x1	x2
1	0,128794500	0,010621300	-0,023181200
2	2,049379700	0,982367700	-1,006854200
3	0,000005700	-0,000169100	-0,000000600
4	0,000001200	-0,000039200	-0,000066200
5	0,000000200	0,000030100	0,000007200
6	0,996126000	0,001584600	-0,993122000
7	0,997610600	-0,992576800	0,002774200
8	0,000076300	-0,000072800	-0,000615900
9	0,001553000	-0,001431100	0,002404100
10	2,052268300	0,977287100	-0,993274800
Wartość średnia	0,622581550	0,097760180	-0,301192940

Porównując wartości średnie Fbest1 widać, że najlepsze wyniki otrzymano gdy wartość "w" nie przekraczała jedności.

Dla w=const otrzymane mogą być zadowalające. Nie są najlepsze wśród porównywanych przypadków, ale mając na uwadze, że na wyniki mają wpływ pozostałe parametry, można próbować je dobrać tak, aby poprawić dokładność algorytmu.

6.3.5 Podsumowanie

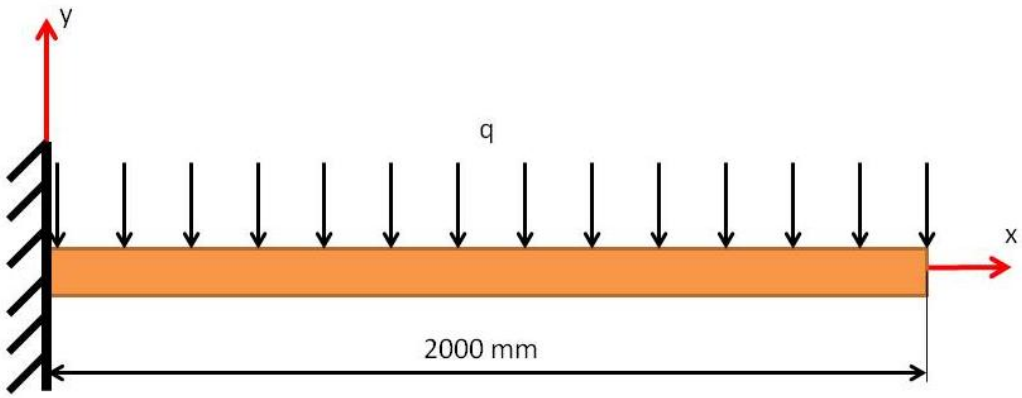
Wpływ na działanie algorytmu PSO ma wiele jego parametrów. Na przykład zwiększanie liczby cząstek, maksymalnej liczby iteracji zwiększa dokładność, lecz wydłuża czas obliczeń. Prawidłowo działający algorytm PSO powinien mieć więc tak dobrane parametry, aby przy możliwie jak najmniejszej liczbie obliczeń funkcji celu uzyskiwać satysfakcjonujące nas wyniki (dokładność rozwiązania, czas obliczeń, eliminacja pojawiania się wyników obarczonych bardzo dużym błędem).

Mimo, iż w algorytmie ławicowym występuje stosunkowo mało parametrów, ich dobór jest niezwykle ważny. Nie ma jednego, uniwersalnego zestawu parametrów, jednak można wyróżnić niektóre tendencje. Autor pozycji [17] przedstawia je w następujący sposób.

Zauważa, że liczebność roju nie powinna być zbyt duża. Od pewnej wartości ilości cząstek zdolność roju do przeszukiwań przestaje wyraźnie zwiększać się. Natomiast czas obliczeń wyraźnie się wydłuża. Autor podkreśla również, że bardzo istotna jest wartość współczynników c1 oraz c2. Wysoka wartość współczynnika c2 może powodować niestabilne działanie algorytmu. Rozumiany jest przez to fakt, że wyniki otrzymywane w kolejnych uruchomieniach algorytmu nie są powtarzalne, pojawiają się duże rozbieżności. Potwierdzają to wyniki obliczeń przeprowadzonych w tej pracy (tabela 6.11). Autor pracy [17] twierdzi, że najlepsze wyniki otrzymuje się przy wartościach c1, c2 zbliżonych do siebie (jednakże poza pewnymi wyjątkami). Wyniki uzyskane w niniejszej pracy również to potwierdzają.

6.4 Przykład obliczeniowy – belka zginana

Działanie algorytmu PSO zostało przetestowane na bardzo prostym przykładzie belki zginanej.



Rysunek 6.2 Belka zginana – schemat obliczeniowy

Rysunek 6.2 przedstawia schemat belki zginanej, obciążonej obciążeniem ciągłym.

Zmiennymi projektowymi są wymiary przekroju poprzecznego belki. Przyjęto przekrój kwadratowy. Celem optymalizacji jest minimalizacja masy - dobranie przekroju tak, aby spełnić zadane ograniczenia (w tym przypadku ograniczenie naprężeń dopuszczalnych).

Dane przyjęte do obliczeń to:

- długość L=2000 mm
- obciążenie q równomiernie rozłożone q= 1 N/mm
- moduł Young'a - E = 220 000 N/mm^2
- gęstość - ro = 7850 kg/m^3 = 7850 * 10 $^{-9}$ kg/mm^3

Zmienne projektowe: wymiary przekroju poprzecznego
- przekrój kwadratowy
- długość boku: x
- pole powierzchni P=x^2
- moment bezwładności I=x^4/12

Ograniczenia: dopuszczalne naprężenia = 50 N/mm^2 oraz x>0

Kod zamieszczony poniżej prezentuje utworzoną funkcję celu, uwzględniającą ograniczenia.

```
function f=fun1(x)

//parametry
dlugosc=2000; //dlugosc belki [mm]
ro=7850*10^-9; //gestosc [kg/mm3]
q=1; //obciazenie [N/mm]
kara=10; //wspolczynnik kary
sigmadop=50; //naprezenia dopuszczalne [MPa]
kara2=-1000;

f=(x.^2)*dlugosc*ro+(kara*max([0,(3*q*(dlugosc^2))/(x.^3)-
sigmadop]))+kara2*min([0,x]);

endfunction;
```

Konstrukcja funkcji celu wykorzystuje metodę funkcji kary:

$$f = M(x) + r_1 g(x) + r_2 h(x) \tag{6.4}$$

gdzie:

M(x) – masa konstrukcji,
g(x) – funkcja kary (przekroczenie naprężeń dopuszczalnych),
h(x) – funkcja kary (eliminacja wartości ujemnych),
r_1, r_2 – odpowiednie współczynniki kary.

Funkcja kary zbudowana została tak aby jej wartość wynosiła 0 w przypadku spełnienia ograniczeń lub dodatnią wartość równą przekroczeniu graniczenia narzuconego na maksymalne dopuszczlne naprężenia.

$$g(x) = max[0, \sigma_{max} - \sigma_{dop}] \tag{6.5}$$

$$\sigma_{max} = \frac{M_g z}{I} \tag{6.6}$$

gdzie:
σ_{dop} – naprężenia dopuszczalne,
M_g – moment gnący,
z – odległość skrajnych włókien od osi obojętnej,

W analogiczny sposób generowane są wartości określające zachowanie dodatniej wartości zmiennej decyzyjnej x (wartość 0 gdy x>=0 lub wartość ujemna, która pomnożona będzie przez ujemny w tym wypadku współczynnki kary).

$$h(x) = min[0, x] \tag{6.7}$$

Funkcja celu (równanie 6.4) została wprowadzona do programu PSO. Algorytm PSO uruchomiono z parametrami przedstawionymi w tabeli 6.17.

Tabela 6.17 Parametry algorytmu PSO

Parametry	wmax	wmin	itmax	c1	c2	N	D	x_sup	x_inf	v_max	v_min
Wartość	0,9	0,4	15	1,6	2,5	15	1	1	100	1	-1

Proces optymalizacji został uruchomiony kilkukrotnie. Poniżej zaprezentowano najlepszy otrzymany wynik.

Wyniki obliczeń za pomocą algorytmu PSO:

- masa: M=60,71 kg
- długość boku: x=62,18 mm

Dokładne rozwiązanie analityczne:

- masa: M=60,62 kg
- długość boku: x=62,14 mm

Jak widać, otrzymany wynik za pomocą algorytmu PSO jest bardzo bliski rozwiązaniu dokładnemu. Można więc sądzić, że algorytm działa poprawnie.

7 Przykłady optymalizacji konstrukcji prętowych za pomocą algorytmu PSO

W niniejszym rozdziale zostały zaprezentowane przykłady optymalizacji konstrukcji prętowych. W rozważanych przypadkach celem była minimalizacja masy konstrukcji z uwzględnieniem określonych ograniczeń, a zmiennymi projektowymi przekroje elementów skończonych - prętów. Porównane zostały wyniki optymalizacji ciągłej i dyskretnej z konstrukcjami nieoptymalizowanymi.

7.1 Kratownica płaska

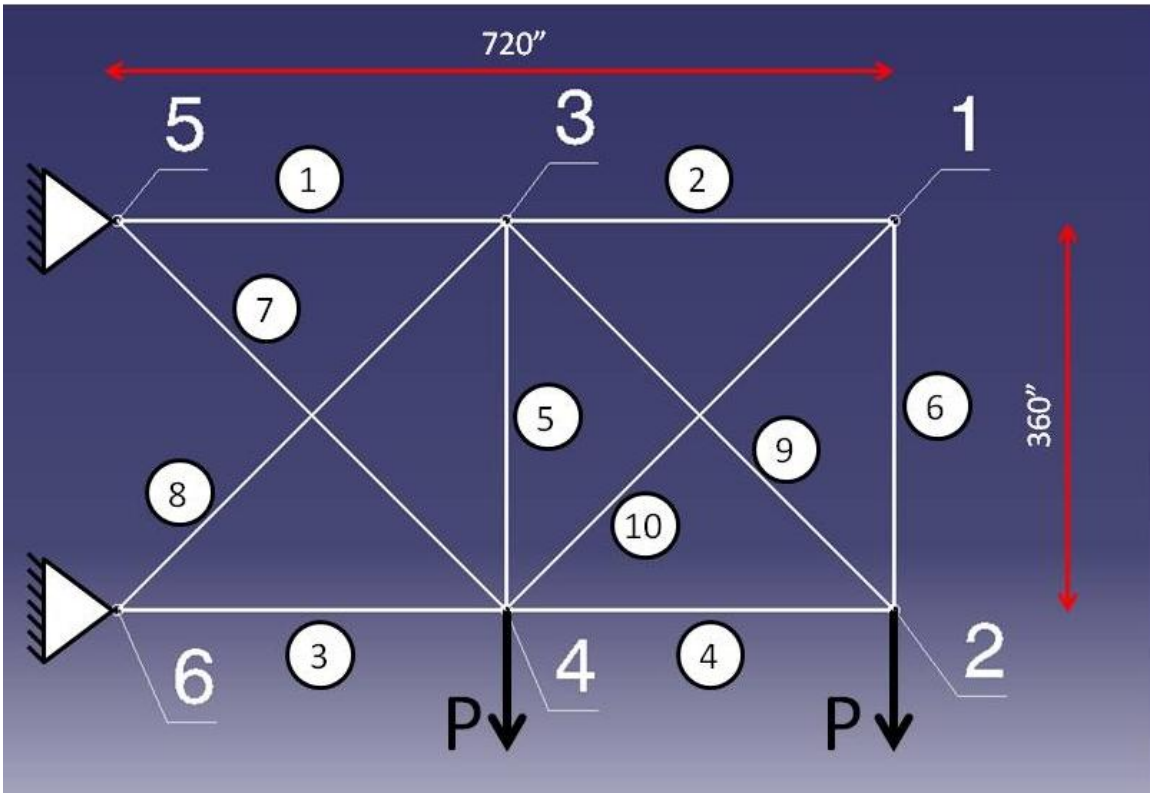
Jako pierwszy przykład wybrano konstrukcję z pozycji [8] (s. 93-95) w celu potwierdzenia poprawności działania algorytmu PSO oraz porównania go z innymi metodami optymalizacji (np. z algorytmami ewolucyjnymi). Jest to klasyczny problem testowy wykorzystywany w literaturze [8].

Zachowano również jednostki systemu imperialnego. Poniższa tabela prezentuje jednostki systemu imperialnego oraz odpowiadające im wartości w układzie SI.

Tabela 7.1 Jednostki systemu imperialnego oraz w układzie SI

Jednostki		
	systemu imperialnego	SI
Długość:	**cal [in]**	**[mm]**
	1	25,4
Masa:	**funt [lb]**	**[kg]**
	1	0,45
Siła	**funt-siła [lbf]**	**[N]**
	1	4,45
Naprężenia:	**lbf/in^2 [PSI]**	**[MPa]**
	1	0,007
Pole powierzchni:	**[in^2]**	**[mm^2]**
	1	645,16
Gęstość:	**[lb/in^3]**	**[kg/mm^3]**
	1	0,0000275

Przykład optymalizowanej kratownicy prezentuje rysunek 7.1 (wymiary w calach).



Rysunek 7.1 Kratownica składająca się z 10 prętów

Celem optymalizacji była minimalizacja masy kratownicy. Zmiennymi projektowymi są pola przekrojów poszczególnych elementów prętowych. Model został zdefiniowany w odpowiednim pliku z parametrami. Prezentuje go załącznik nr 6. Początek układu współrzędnych (niezbędny do definicji węzłów) przyjęto w węźle nr 6. Struktura pliku z danymi została już opisana rozdziale 5.2.1. Należy zaznaczyć, że wszystkie parametry konstrukcji, takie jak współrzędne węzłów, obciążenie, moduł Young’a, naprężenia dopuszczalne, przemieszczenia dopuszczalne, dopuszczalne wartości przekrojów zostały przyjęte takie same jak w pozycji [8]. Dzięki temu możliwe będzie porównanie wyników optymalizacji algorytmem PSO z wynikami optymalizacji algorytmem ewolucyjnym.

W analizowanym przykładzie przyjęto następujące dane:

$$P = 10^5\ lbf$$

$$E = 10^7\ \frac{lbf}{in^2} - moduł\ Young'a$$

$$\rho = 0{,}1 \cdot 10^{-3}\ \frac{lb}{in^3} - gęstość$$

$$\sigma_{dop} = 25000\ \frac{lbf}{in^2} - naprężenia\ dopuszczalne$$

$$u_{dop} = 2\ in - przemieszczenia\ dopuszczalne$$

Zmienne projektowe (przekroje prętów):

- zmienne ciągłe mogą przyjmować wartość z przedziału <Amin:Amax>

$$Amin = 0{,}1\ in^2$$

$$Amax = 36\ in^2$$

- zmienne dyskretne mogą przyjmować tylko wartości znajdujące się w wektorze **katalogA** (wartości w calach).

katalogA = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36]

W pierwszej kolejności został wyznaczony jeden wspólny przekrój dla wszystkich elementów prętowych, tak aby konstrukcja spełniała nałożone ograniczenia. Jest to konstrukcja spełniająca zadane ograniczenia, jednak ze względu na sformułowane kryteria optymalizacji jest ona nieoptymalna. Następnie proces optymalizacji został uruchomiony kilkukrotnie dla różnych zestawów parametrów PSO, dzięki czemu możliwa była ocena wpływu parametrów na jakość działania algorytmu oraz ocena zaoszczędzonej masy konstrukcji .

7.1.1 Wyniki obliczeń bez optymalizacji

Poniżej zostały zaprezentowane wyniki obliczeń dla kratownicy płaskiej. Dobór przekroju został wykonany przy pomocy skryptu zamieszczonego w załączniku nr 9.

Pole przekroju pręta konstrukcji o jednakowych przekrojach spełniających warunki ograniczające (dopuszczalne przemieszczenia i naprężenia) wynosi:

$$A = 19{,}7\ in^2.$$

Masa konstrukcji wynosi wówczas:

$$M = 8{,}27\ lb.$$

7.1.2 Zestaw parametrów nr 1

Optymalizacja karatownicy dla zmiennych ciągłych

Algorytm PSO został uruchomiony z następującymi parametrami zawartymi w poniższej tabeli.

Tabela 7.2 Parametry algorytmu PSO

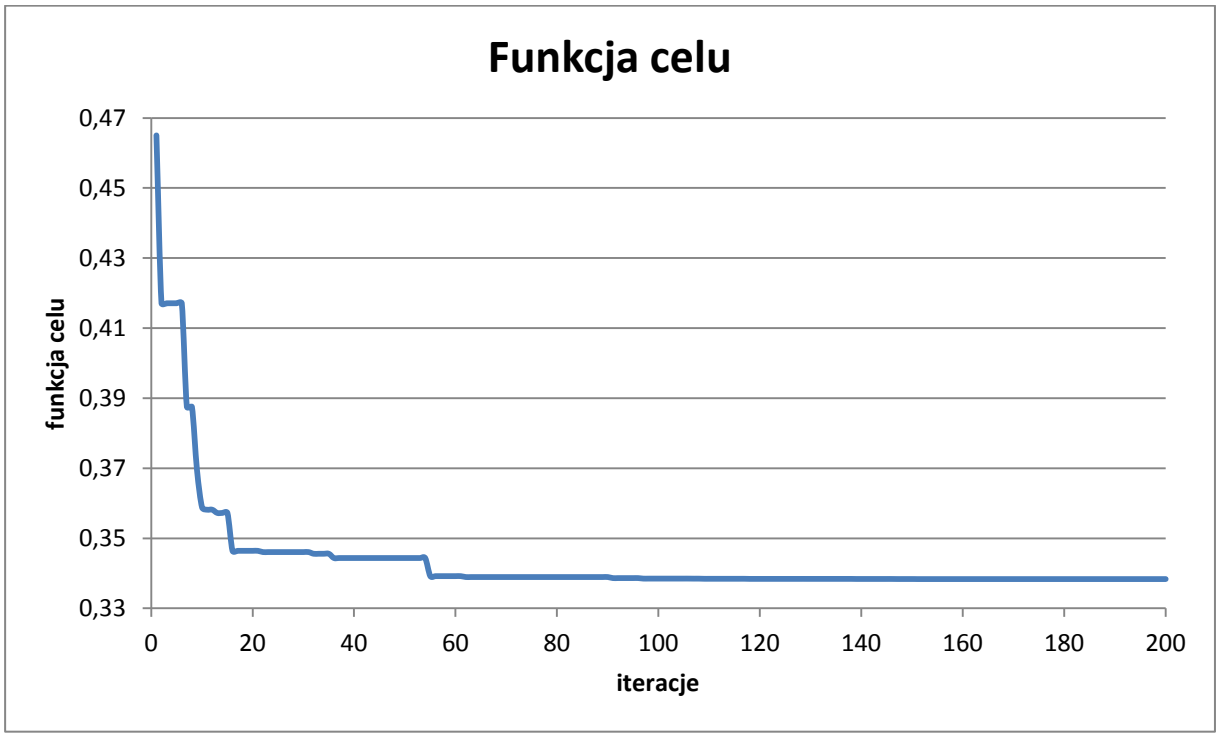
wmax	wmin	itmax	c1	c2	N	zmienne	vmax	vmin	ograniczenia
0,9	0,4	200	1,3	2,8	100	ciągłe	20	-20	naprężenia
									przemieszczenia

Dobór parametrów N (liczba cząstek) oraz itmax (maksymalna liczba iteracji) nie był przypadkowy. Algorytm ewolucyjny, który posłużył do optymalizacji kratownicy w pozycji [8] wykorzystywał populację 100 osobników (wartość analogiczna do liczby cząstek N w algorytmie PSO) natomiast liczba generacji wynosiła 200 (analogiczna wartość do itmax w algorytmie PSO). W pierwszej kolejności obliczenia zostały wykonane dla zmiennych ciągłych z zakresu Amin:Amax zdefiniowanego przez użytkownika (patrz załącznik nr 6). Proces optymalizacji był uruchamiany kilkukrotnie, poniżej zaprezentowano najlepszy wynik uzyskanych w uruchomieniach.

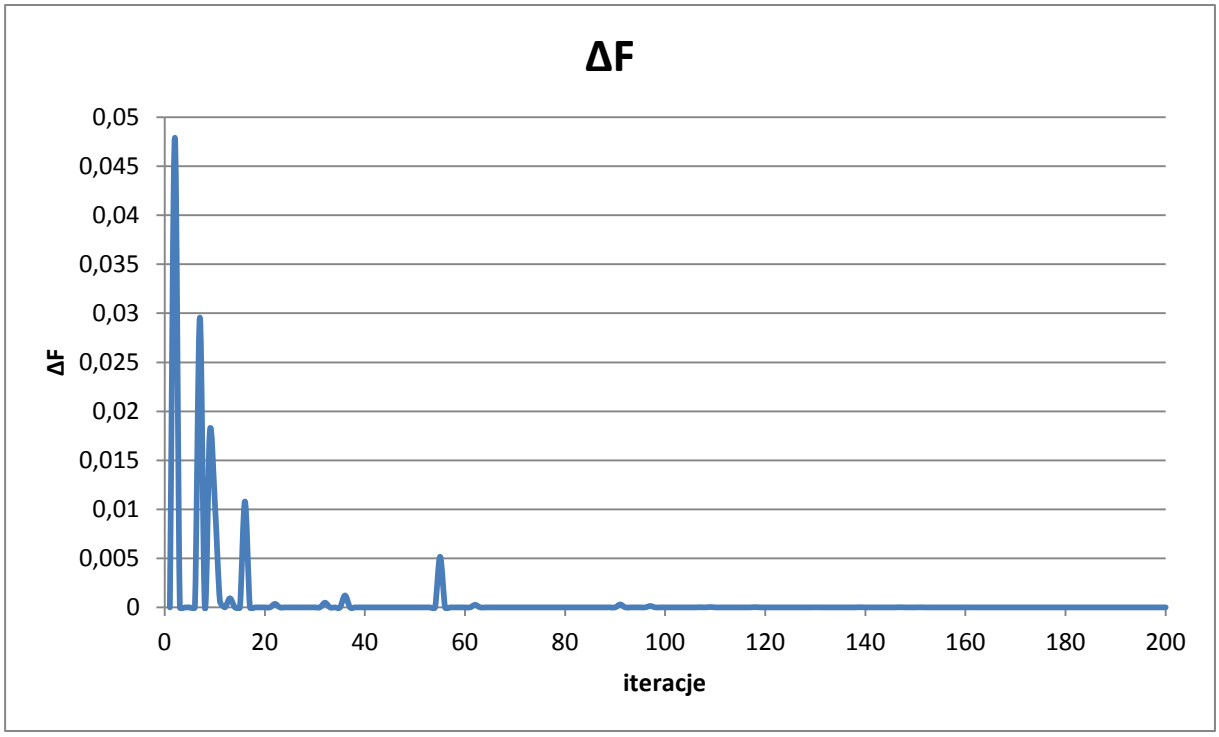
Tabela 7.3 przedstawia najlepsze uzyskane wyniki optymalizacji.

Tabela 7.3 Wyniki optymalizacji (zmienne ciągłe)

Nr elementu	Pole przekroju [in^2]
1	36,000
2	0,100
3	22,615
4	14,130
5	0,100
6	0,100
7	8,533
8	20,172
9	19,947
10	0,100
Masa [lb]	5,112
Funkcja celu	0,33836



Rysunek 7.2 Wykres wartości funkcji celu w poszczególnych iteracjach



Rysunek 7.3 Wykres zmian wartości funkcji celu w poszczególnych iteracjach

Jak widać na rysunkach 7.2 oraz 7.3 od około iteracji nr 100, wartość funkcji celu przestaje zmniejszać swoją wartość i dalsze obliczenia nie przyczyniają się do uzyskania lepszego wyniku.

Optymalizacja karatownicy dla zmiennych dyskretnych

Tabela 7.4 przedstawia parametry zastosowanego algorytmu PSO.

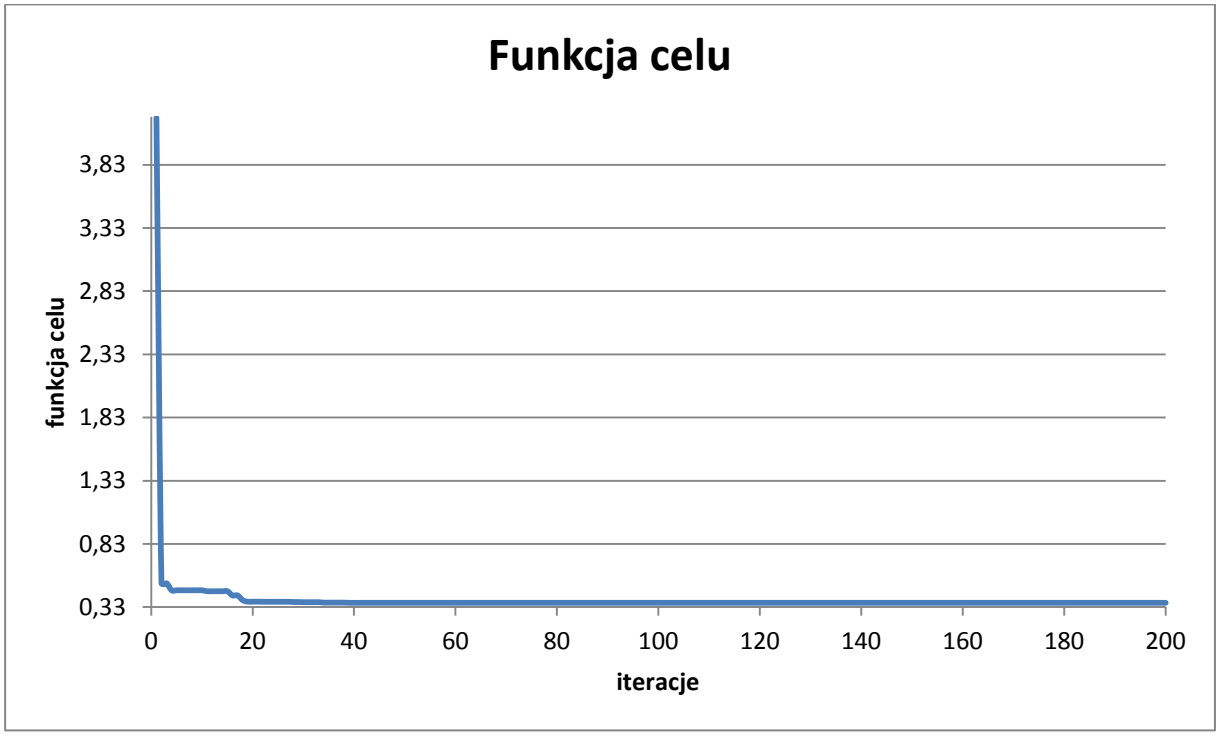
Tabela 7.4 Parametry algorytmu PSO

wmax	wmin	itmax	c1	c2	N	zmienne	vmax	vmin	ograniczenia
0,9	0,4	200	1,3	2,8	100	dyskretne	20	-20	naprężenia
									przemieszczenia

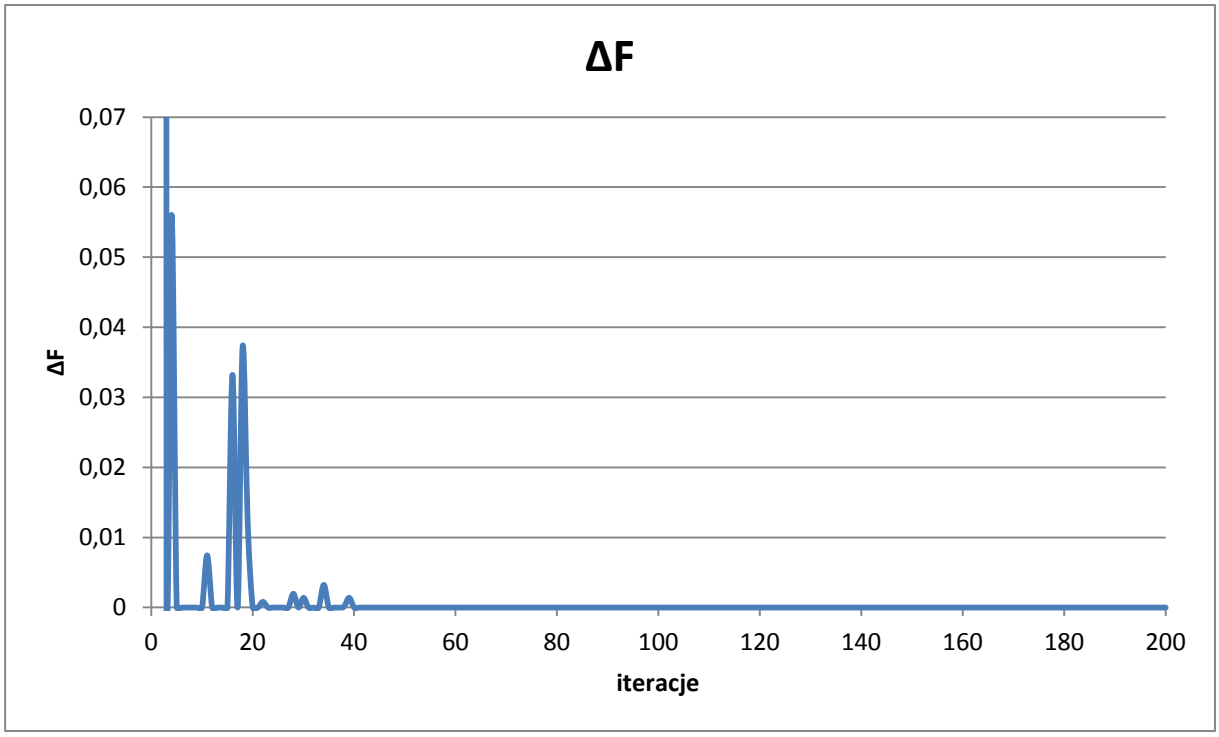
W tym przypadku proces optymalizacji został uruchomiony dla identycznych zmiennych jak w poprzednich przypadkach, ale ze zmiennymi dyskretnymi (dopuszczalne wartości pól przekrojów ze zdefiniowanego katalogu o skończonej liczbie elementów). Tabela 7.5 oraz rysunki 7.4 i 7.5 prezentują wyniki obliczeń.

Tabela 7.5 Wyniki optymalizacji (zmienne dyskretne)

Nr elementu	Pole przekroju [in^2]
1	28
2	0,1
3	20
4	12
5	0,1
6	0,1
7	9
8	20
9	36
10	0,1
Masa [lb]	5,48515
Funkcja celu	0,36308



Rysunek 7.4 Wykres wartości funkcji celu w poszczególnych iteracjach



Rysunek 7.5 Wykres zmian wartości funkcji celu w poszczególnych iteracjach

7.1.3 Zestaw parametrów nr 2

W tym przypadku zmieniono wartości N (liczba cząstek) oraz itmax (maksymalna liczba iteracji).

Optymalizacja karatownicy dla zmiennych ciągłych

Algorytm PSO został uruchomiony z następującymi parametrami zawartymi w tabeli 7.6.

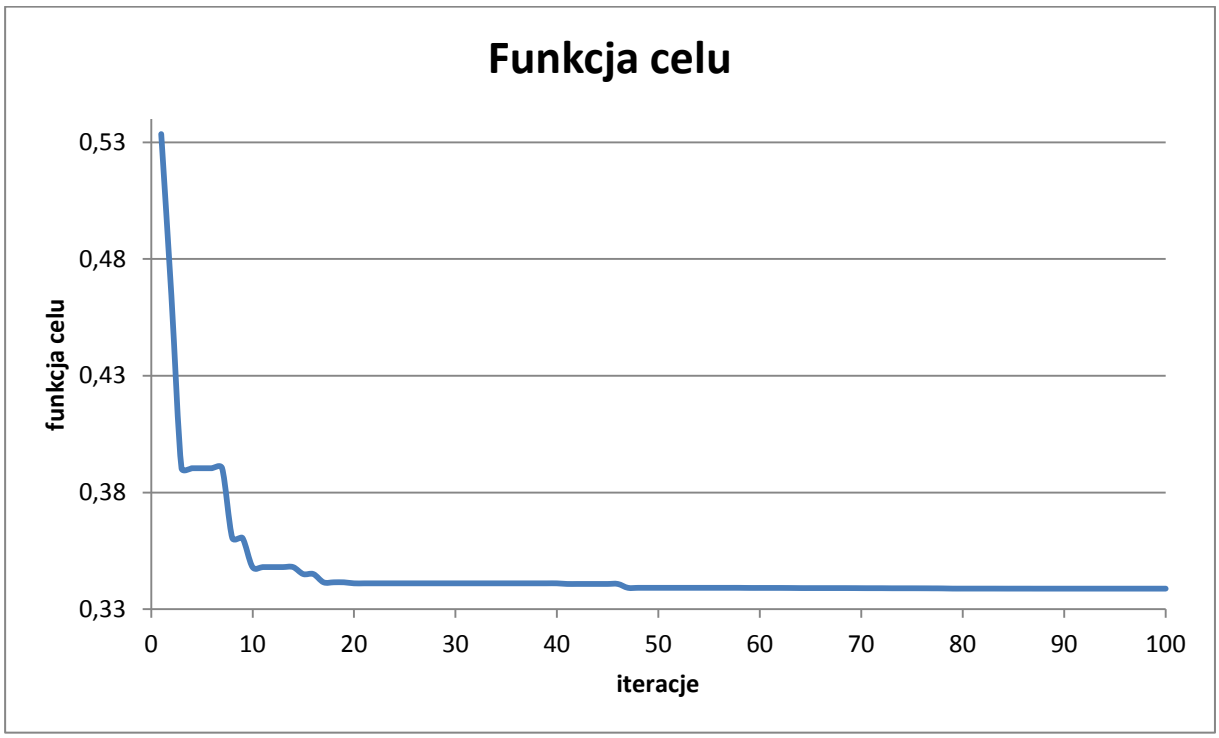
Tabela 7.6 Parametry algorytmu PSO

wmax	wmin	itmax	c1	c2	N	zmienne	vmax	vmin	ograniczenia
0,9	0,4	100	1,3	2,8	200	ciągłe	20	-20	naprężenia
									przemieszczenia

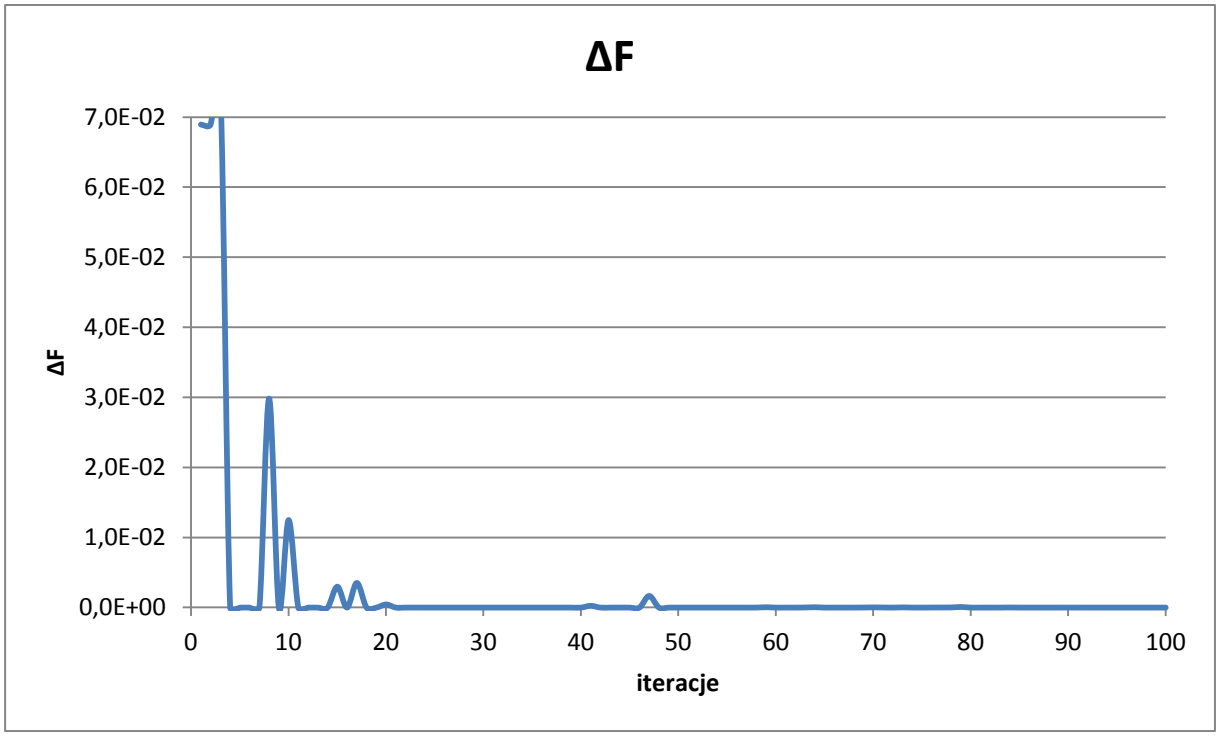
Wyniki optymalizacji prezentuje tabela 7.7 oraz rysunki 7.6 i 7.7.

Tabela 7.7 Wyniki optymalizacji (zmienne ciągłe)

Nr elementu	Pole przekroju [in^2]
1	36
2	0,1
3	24,9294
4	14,0679
5	0,1
6	0,1
7	8,12472
8	19,2445
9	19,823
10	0,1
Masa [lb]	5,11843
Funkcja celu	0,36308



Rysunek 7.6 Wykres wartości funkcji celu w poszczególnych iteracjach



Rysunek 7.7 Wykres zmian wartości funkcji celu w poszczególnych iteracjach

Optymalizacja karatownicy dla zmiennych dyskretnych

Algorytm PSO został uruchomiony z następującymi parametrami zawartymi w poniższej tabeli.

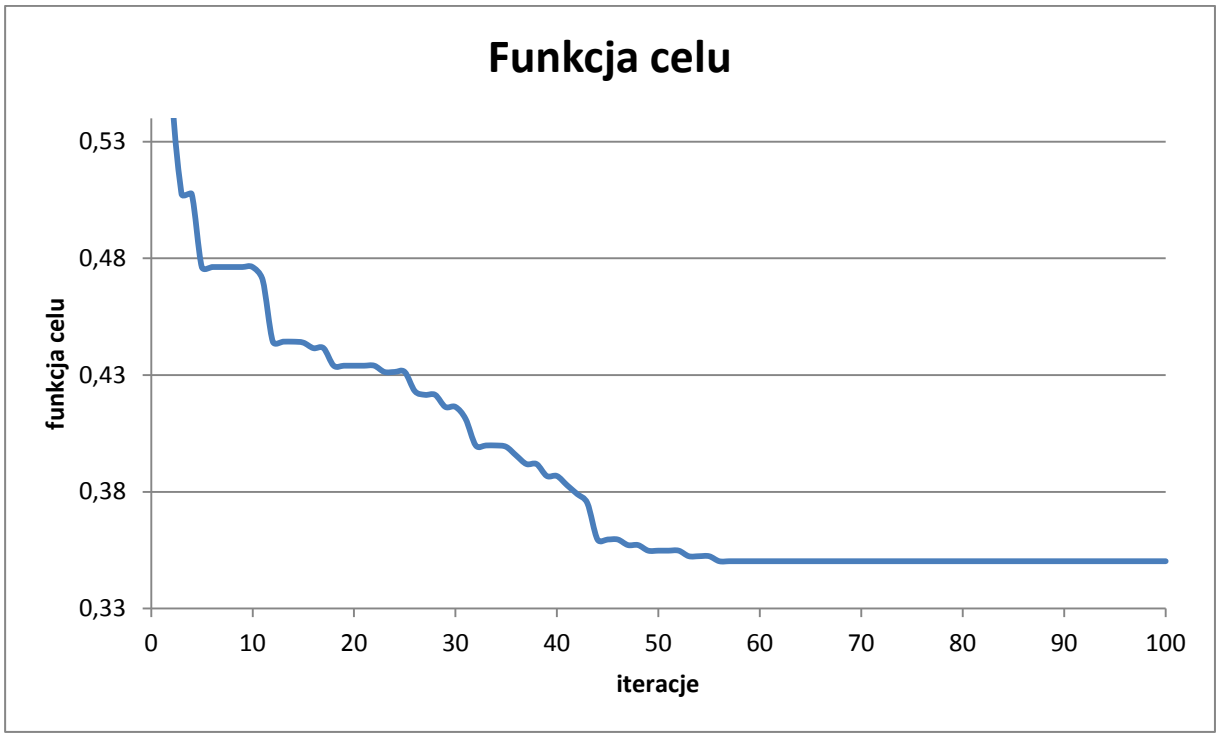
Tabela 7.8 Parametry algorytmu PSO

wmax	wmin	itmax	c1	c2	N	zmienne	vmax	vmin	ograniczenia
0,9	0,4	100	1,3	2,8	200	dyskretne	20	-20	naprężenia
									przemieszczenia

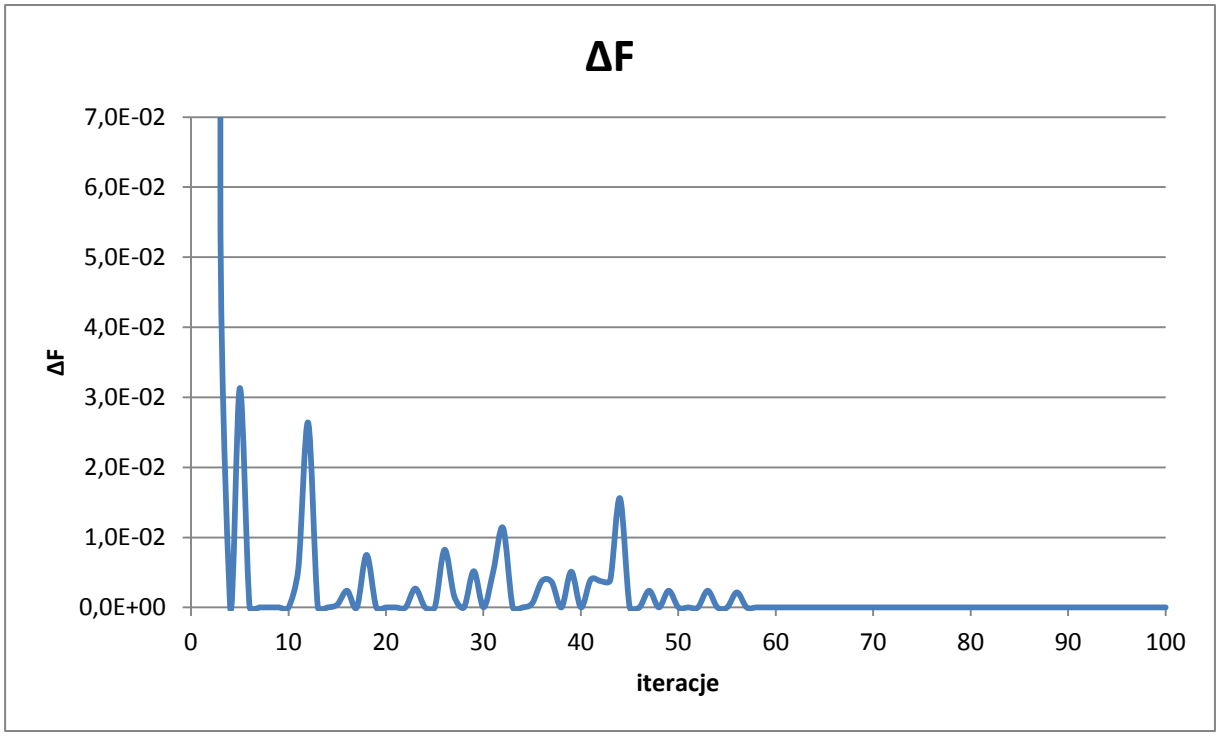
Wyniki optymalizacji prezentuje tabela 7.9 oraz rysunki 7.8 i 7.9.

Tabela 7.9 Wyniki optymalizacji (zmienne dyskretne)

Nr elementu	Pole przekroju [in^2]
1	32
2	0,1
3	36
4	14
5	0,1
6	1
7	7
8	20
9	18
10	0,1
Masa [lb]	5,29132
Funkcja celu	0,35025



Rysunek 7.8 Wykres wartości funkcji celu w poszczególnych iteracjach



Rysunek 7.9 Wykres zmian wartości funkcji celu w poszczególnych iteracjach

7.1.4 Zestaw parametrów nr 3

W tym przypadku liczbę cząstek przyjęto N=150 oraz maksymalną liczbę iteracji itmax=60. Pozostałe parametry algorytmu PSO pozostały bez zmian.

Optymalizacja karatownicy dla zmiennych ciągłych

Algorytm PSO został uruchomiony z następującymi parametrami zawartymi w poniższej tabeli.

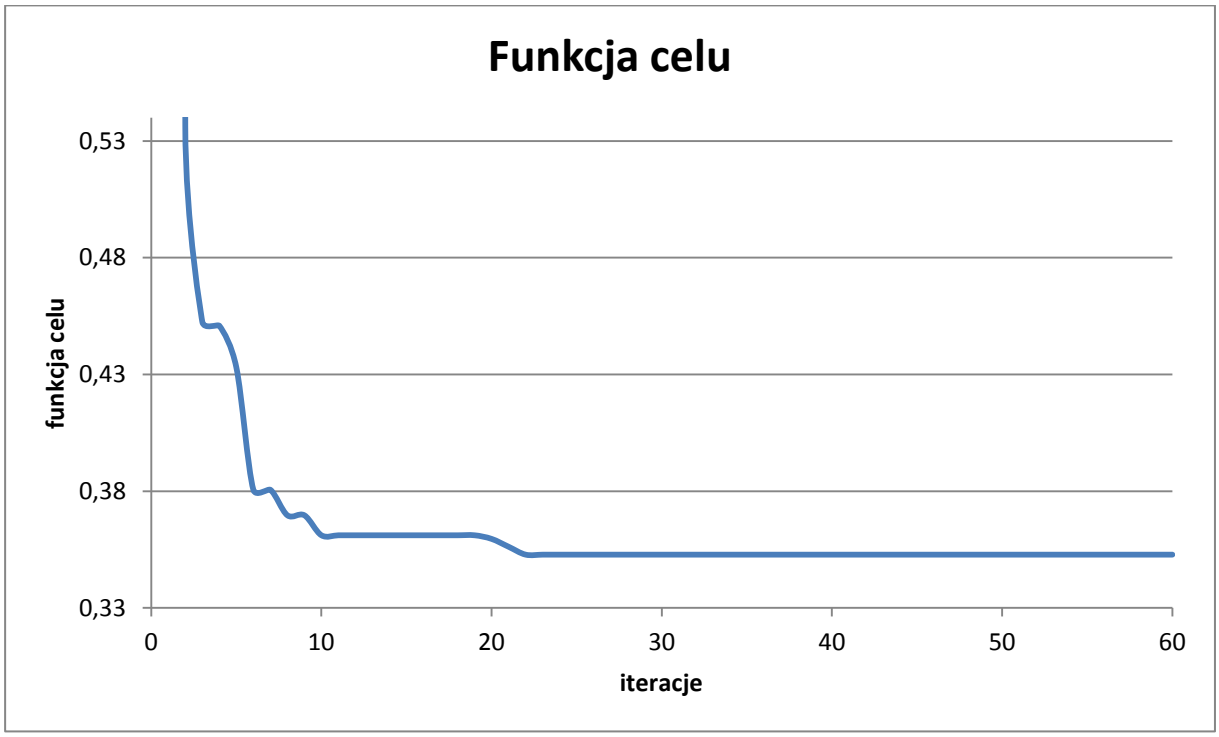
Tabela 7.10 Parametry algorytmu PSO

wmax	wmin	itmax	c1	c2	N	zmienne	vmax	vmin	ograniczenia
0,9	0,4	60	1,3	2,8	150	ciągłe	20	-20	naprężenia
									przemieszczenia

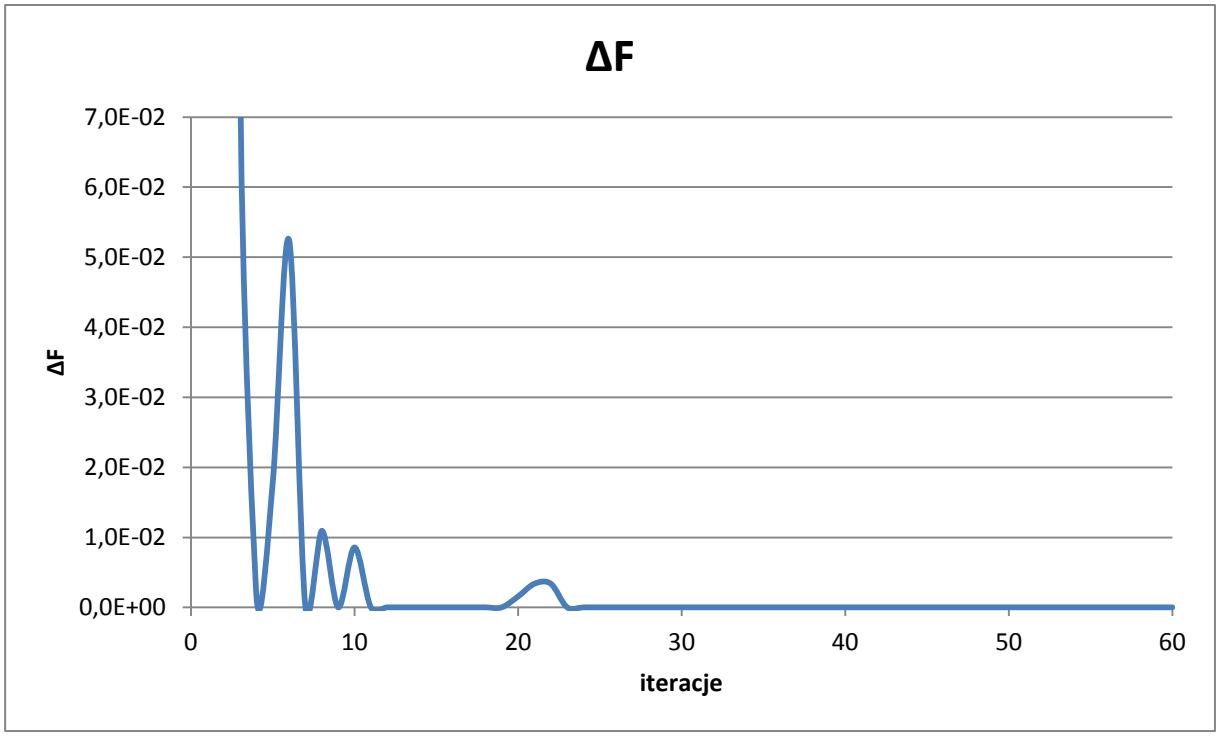
Wyniki optymalizacji prezentuje tabela 7.11 oraz rysunki 7.10 i 7.11.

Tabela 7.11 Wyniki optymalizacji (zmienne ciągłe)

Nr elementu	Pole przekroju [in^2]
1	36
2	0,1
3	24,9294
4	14,0679
5	0,1
6	0,1
7	8,12472
8	19,2445
9	19,823
10	0,1
Masa [lb]	5,11843
Funkcja celu	0,36308



Rysunek 7.10 Wykres wartości funkcji celu w poszczególnych iteracjach



Rysunek 7.11 Wykres zmian wartości funkcji celu w poszczególnych iteracjach

Optymalizacja karatownicy dla zmiennych dyskretnych

Algorytm PSO został uruchomiony z następującymi parametrami zawartymi w poniższej tabeli.

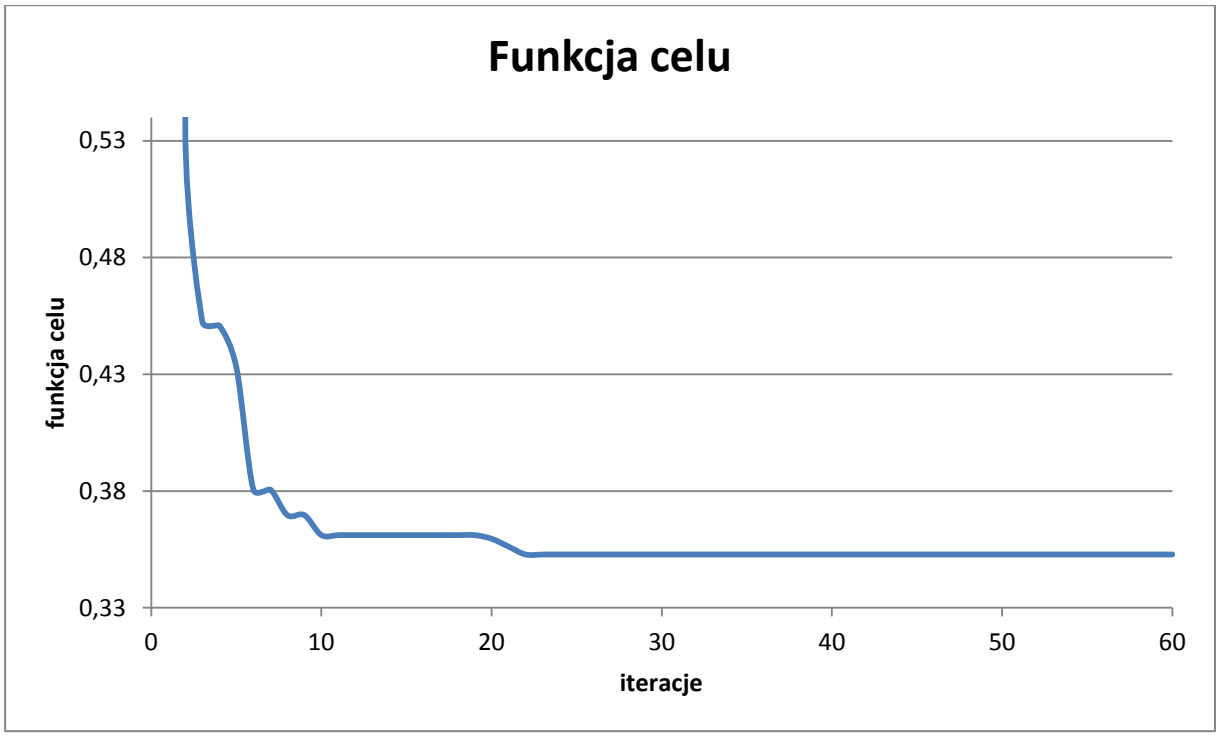
Tabela 7.12 Parametry algorytmu PSO

wmax	wmin	itmax	c1	c2	N	zmienne	vmax	vmin	ograniczenia
0,9	0,4	60	1,3	2,8	150	dyskretne	20	-20	naprężenia
									przemieszczenia

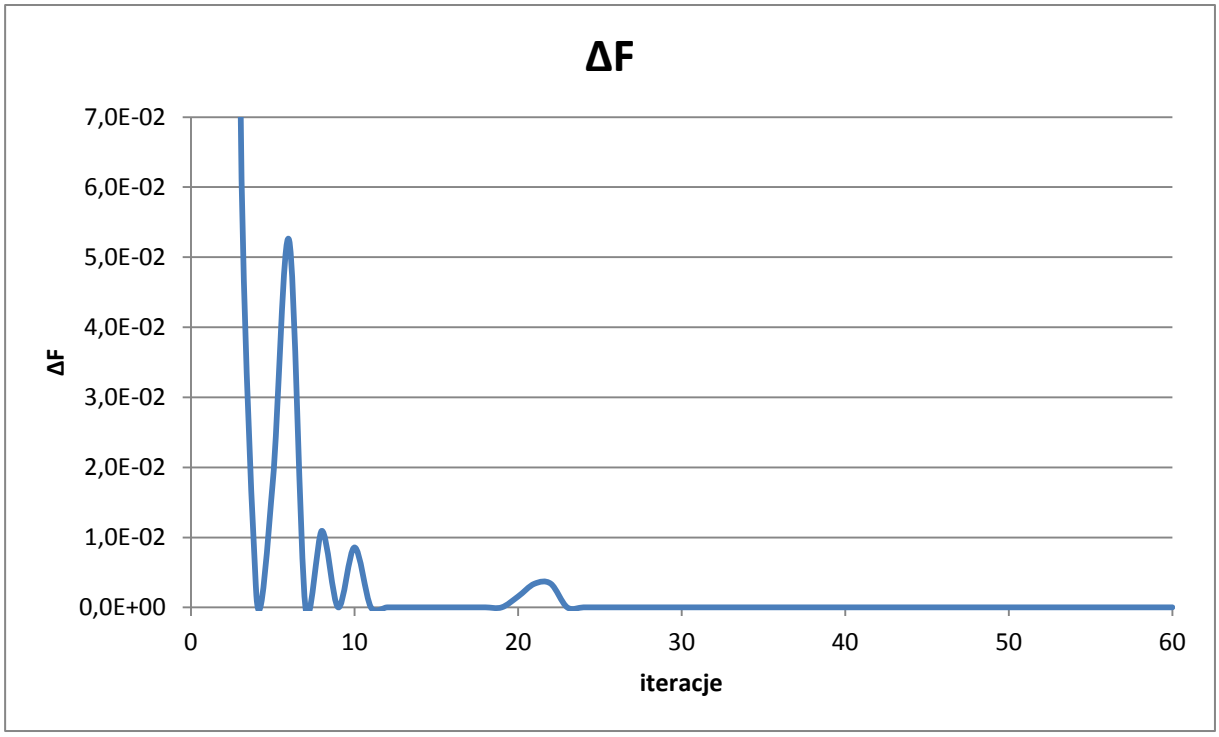
Wyniki optymalizacji prezentuje tabela 7.13 oraz rysunki 7.12 i 7.13.

Tabela 7.13 Wyniki optymalizacji (zmienne dyskretne)

Nr elementu	Pole przekroju [in^2]
1	36
2	0,1
3	36
4	12
5	0,1
6	0,1
7	7
8	18
9	20
10	0,1
Masa [lb]	5,33092
Funkcja celu	0,35287



Rysunek 7.12 Wykres wartości funkcji celu w poszczególnych iteracjach



Rysunek 7.13 Wykres zmian wartości funkcji celu w poszczególnych iteracjach

7.1.5 Zestaw parametrów nr 4

W kolejnym przypadku użyto parametrów jak w zestawie nr 2, ale zmieniono ograniczenia. Ograniczeniem były tylko naprężenia dopuszczalne.

Optymalizacja karatownicy dla zmiennych ciągłych

Algorytm PSO został uruchomiony z następującymi parametrami zawartymi w poniższej tabeli.

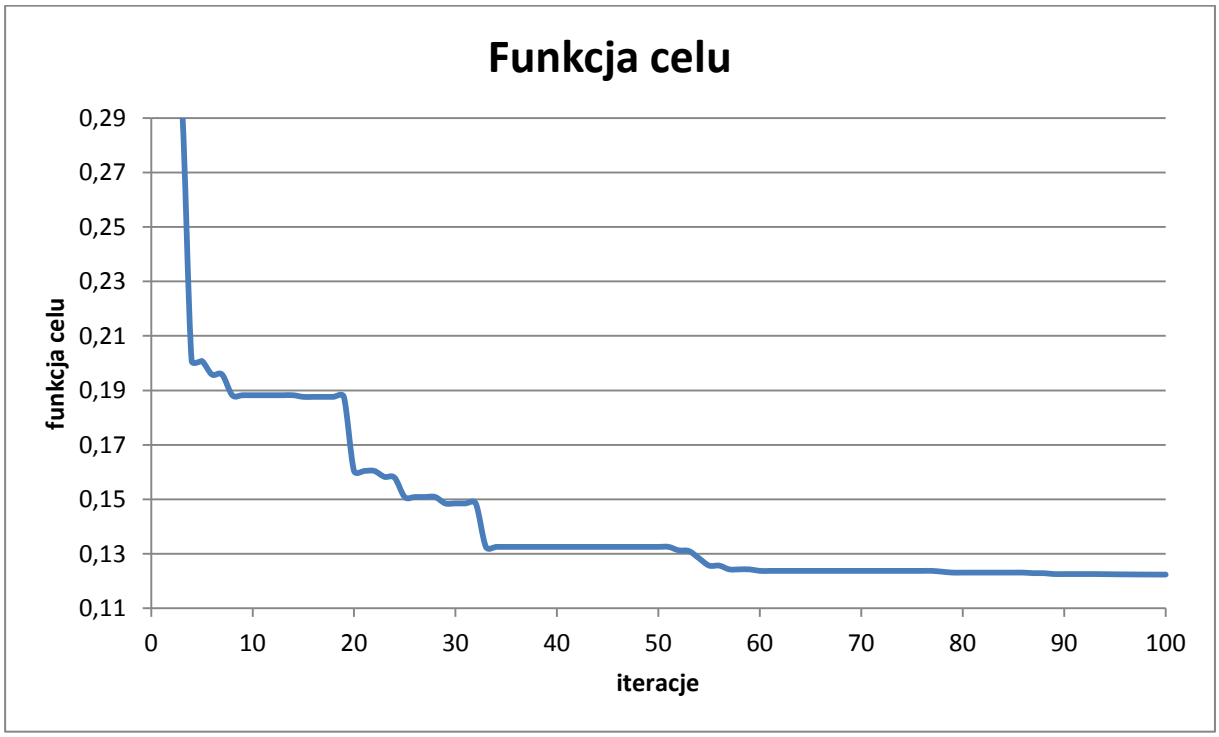
Tabela 7.14 Parametry algorytmu PSO

wmax	wmin	itmax	c1	c2	N	zmienne	vmax	vmin	ograniczenia
0,9	0,4	100	1,3	2,8	200	ciągłe	20	-20	naprężenia

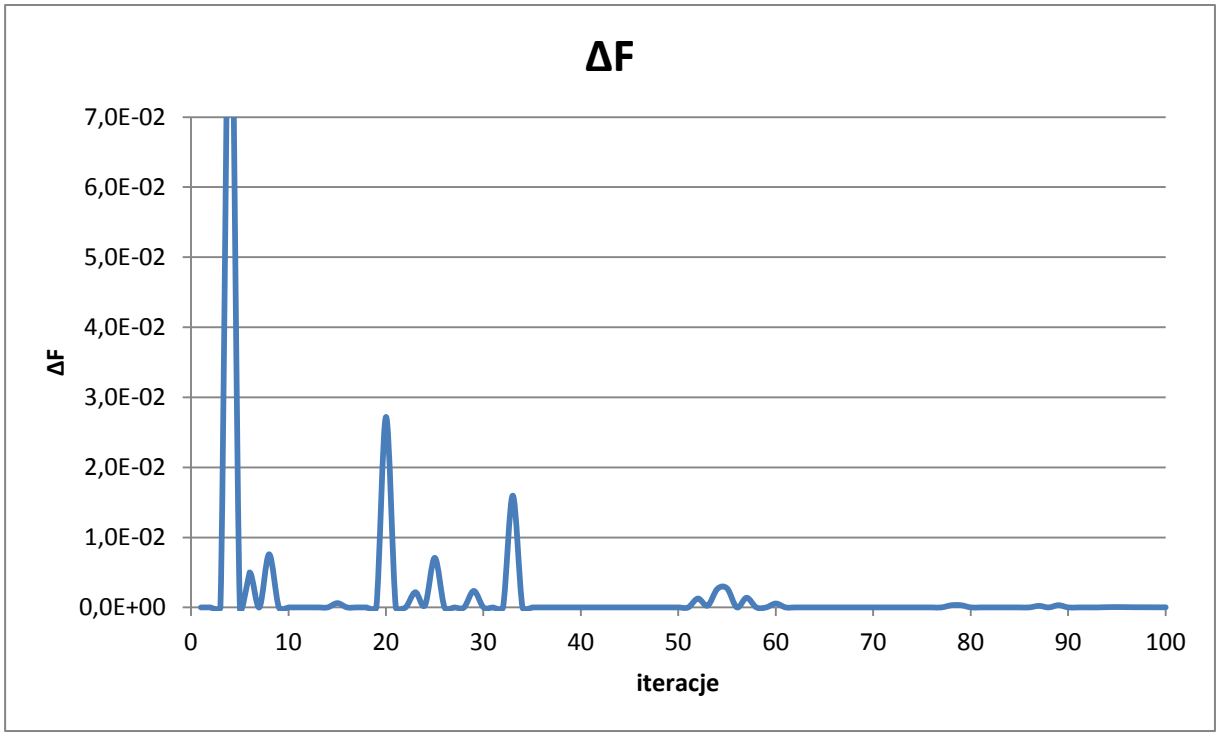
Wyniki optymalizacji prezentuje tabela 7.15 oraz rysunki 7.14 i 7.15.

Tabela 7.15 Wyniki optymalizacji (zmienne ciągłe)

Nr elementu	Pole przekroju [in^2]
1	8,11255
2	0,1
3	8,29746
4	3,96206
5	0,1
6	0,1
7	5,64276
8	7,61238
9	8,33117
10	0,1
Masa [lb]	1,84828
Funkcja celu	0,12234



Rysunek 7.14 Wykres wartości funkcji celu w poszczególnych iteracjach



Rysunek 7.15 Wykres zmian wartości funkcji celu w poszczególnych iteracjach

Optymalizacja karatownicy dla zmiennych dyskretnych

Algorytm PSO został uruchomiony z następującymi parametrami zawartymi w poniższej tabeli.

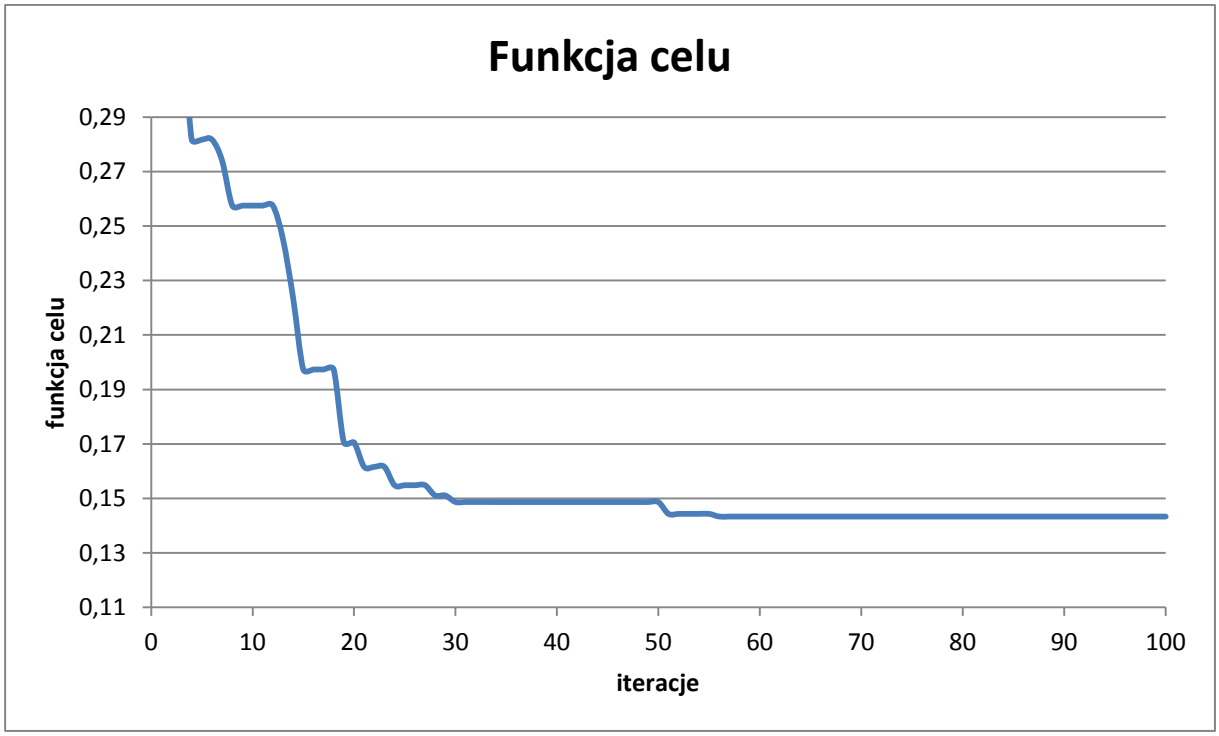
Tabela 7.16 Parametry algorytmu PSO

wmax	wmin	itmax	c1	c2	N	zmienne	vmax	vmin	ograniczenia
0,9	0,4	100	1,3	2,8	200	dyskretne	20	-20	naprężenia

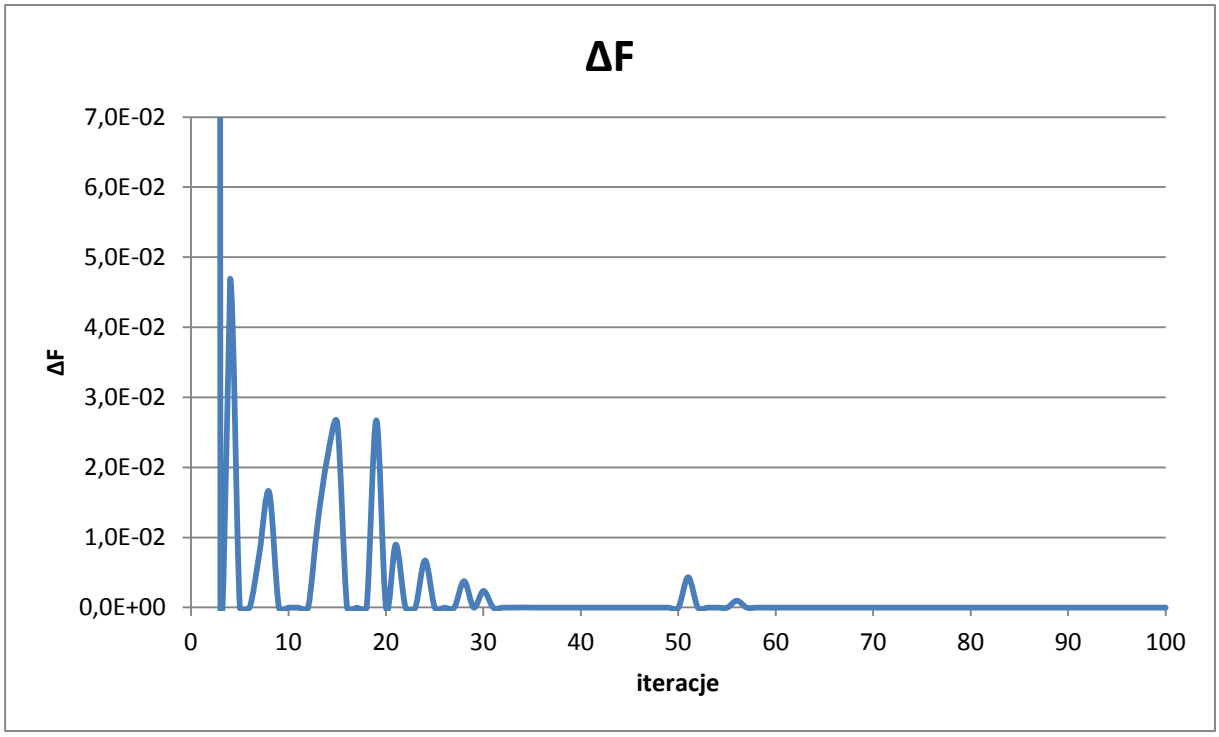
Wyniki optymalizacji prezentuje tabela 7.17 oraz rysunki 7.16 i 7.17.

Tabela 7.17 Wyniki optymalizacji (zmienne dyskretne)

Nr elementu	Pole przekroju [in^2]
1	5
2	6
3	12
4	0,1
5	0,1
6	7
7	12
8	0,1
9	0,1
10	9
Masa [lb]	2,16653
Funkcja celu	0,14341



Rysunek 7.16 Wykres wartości funkcji celu w poszczególnych iteracjach



Rysunek 7.17 Wykres zmian wartości funkcji celu w poszczególnych iteracjach

7.1.6 Podsumowanie wyników

Wyniki optymalne uzyskane w przykładach zaprezentowanych w podrozdziałach 7.1.1 do 7.1.5 zestawiono w tabeli 7.18.

Tabela 7.18 Wyznaczona masa dla wszystkich zestawów parametrów

	Zestaw nr 1		**Zestaw nr 2**		**Zestaw nr 3**		**Zestaw nr 4**		**Bez optymalizacji**
	N=100	itmax=200	N=200	itmax=100	N=150	itmax=60	N=200	itmax=100	
ograniczenia	naprężenia, przemieszczenia		naprężenia, przemieszczenia		naprężenia, przemieszczenia		naprężenia		
zmienne	ciągłe	dyskretne	ciągłe	dyskretne	ciągłe	dyskretne	ciągłe	dyskretne	
masa [lb]	**5,112**	**5,485**	**5,118**	**5,291**	**5,118**	**5,331**	**1,848**	**2,167**	**8,27**

Jak można zauważyć w powyższych tabelach masa konstrukcji wyznaczona dla każdego zestawu parametrów jest zawsze mniejsza w przypadku optymalizacji dla zmiennych ciągłych.

Widać również, że masy uzyskane w każdym z przypadków mają zbliżone do siebie wartości. Można więc sądzić, że w przypadku nr 1 oraz nr 2 maksymalna liczba iteracji była zbyt duża. Program wykonywał więcej obliczeń, jednak nie przyczyniały się one do uzyskania lepszego wyniku. Bardzo ważne jest zatem aby w procesie optymalizacji konstrukcji za pomocą PSO odpowiednio dobrać parametr itmax, ponieważ niepotrzebne iteracje wydłużają obliczenia. Dzięki procesowi optymalizacji udało się zredukować masę konstrukcji o około 38% w porównaniu z rozwiązaniem uzyskanym przyjmując jednakowe przekroje wszystkich prętow.

Tabele 7.19 oraz 7.20 prezentują wyniki obliczeń optymalizacji kratownicy uzyskane za pomocą Algorytmu Ewolucyjnego, podane w [8] (oznaczenia: p – liczba populacji, e – liczba ewolucji). Zamieszczone tu wyniki są najlepszym rozwiązaniem otrzymanym wśród 100 uruchomień procesu optymalizacji.

Tabela 7.19 Wyznaczona masa za pomocą algorytmu ewolucyjnego [8]

	p=100	e=200
ograniczenia	naprężenia	
zmienne	ciągłe	dyskretne
Masa [lb]	**1,593**	**1,688**

Tabela 7.20 Wyznaczona masa za pomocą algorytmu ewolucyjnego [8]

	p=100	e=200
ograniczenia	naprężenia, przemieszczenia	
zmienne	ciągłe	dyskretne
Masa [lb]	**5,061**	**5,111**

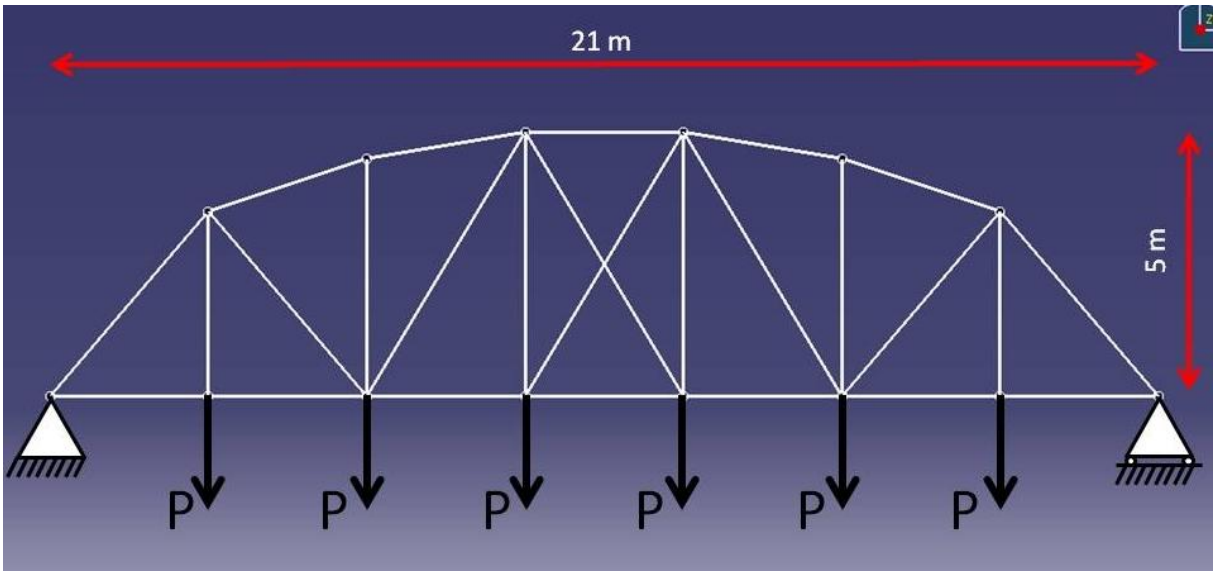
Wyniki otrzymane przy pomocy zastosowanego algorytmu PSO nie są tak dobre jak te uzyskane przy pomocy algorytmu ewolucyjnego, jednak nie odbiegają od nich znacząco. Algorytm PSO uruchamiany był od 5 do 10 razy, natomiast algorytm ewolucyjny w pracy [8] aż 100 razy. W metodach niedeterministycznych wynik zależy zatem w pewnym stopniu od szczęścia czyli w naszym przypadku od generatora liczb losowych. Zawierają one element losowości i każde uruchomienie daje nieco inny wynik. Można więc za pierwszym razem trafić na bardzo dobry wynik, ale czasem potrzeba wielu uruchomień na uzyskanie zadowalających wyników. Sytuację tą można poprawić odpowiednio dopierając parametry algorytmu (takie jak wmax, wmin, c1, c2, N) jednak w ogólnym przypadku nie jest to łatwe i wymaga szeregu prób.

Warto również zauważyć różnicę w wynikach dla zadania z ograniczeniami naprężeń i przemieszczeń dopuszczalnych oraz zadania sformułowanego tylko dla ograniczeń naprężeń. W drugim przypadku obliczona masa jest ponad dwukrotnie mniejsza. Oznacza to, że tzw. warunkiem aktywnym są dopuszczalne przemieszczenia. Spełnienie warunku na dopuszczalne przemieszczenia spełnia równocześnie warunek naprężeń dopuszczalnych z dużym zapasem. W analizowanym przykładzie testowym usunięcie ograniczeń naprężeń, a pozostawienie tylko ograniczeń przemieszczeń nie powinno wpływać na wyniki w porównaniu do ograniczeń tylko przemieszczeń dopuszczalnych.

7.2 Model płaski mostu

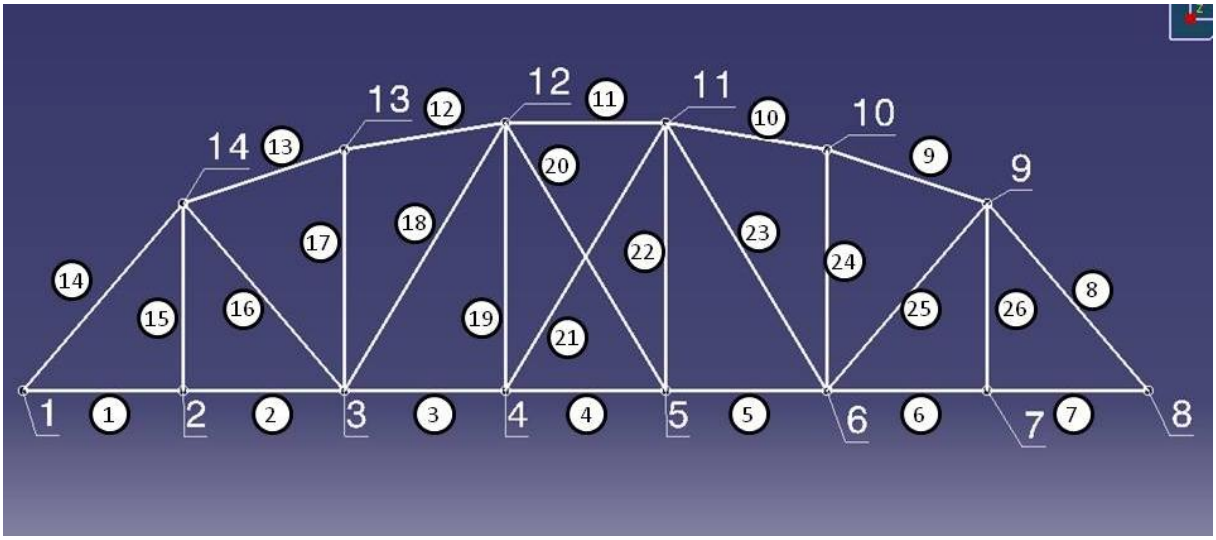
Kolejną optymalizowaną konstrukcją jest fragment konstrukcji mostu zbudowanego z prętów, poddany obciążeniu statycznemu.

Schemat konstrukcji, obciążenie oraz wymiary główne przedstawia rysunek 7.18.



Rysunek 7.18 Model mostu zbudowanego z elementów prętowych – schemat konstrukcji, obciążenie oraz wymiary główne

Rysunek 7.19 przedstawia numerację węzłów oraz numerację elementów skończonych.



Rysunek 7.19 Model mostu zbudowanego z elementów prętowych – oznaczenie węzłów oraz elementów skończonych

W analizowanym przykadzei przyjęto następujące dane:

$$P = 115000\ N$$
$$E = 210000\ MPa - moduł\ Young'a$$
$$\rho = 7{,}86 \cdot 10^{-6}\ \frac{kg}{mm^3} - gęstość$$
$$\sigma_{dop} = 350\ MPa - naprężenia\ dopuszczalne$$
$$u_{dop} = 30\ mm - przemieszczenia\ dopuszczalne$$

Zmienne projektowe (przekroje prętów):
- zmienne ciągłe mogą przyjmować wartość z przedziału <Amin:Amax>

$$Amin = 20\ mm^2$$
$$Amax = 4000\ mm^2$$

- zmienne dyskretne mogą przyjmować tylko wartości znajdujące się w wektorze **`katalogA`** (wartości w mm^2).

`katalogA` = [20, 120, 240, 320, 440, 560, 680, 760, 880, 1000, 1120, 1320, 1560, 1760, 2000, 2240, 2440, 2680, 2880, 3120, 3320, 3560, 3760, 4000]

Model konstrukcji, wymagane parametry, dane materiałowe, obciążenie, warunki brzegowe zostały zapisane w pliku danych o odpowiedniej strukturze, który wykorzystuje program PSO. Kod pliku został zamieszczony w załączniku nr 7.

Ponieważ analizowany przypadek jest bardziej skomplikowany w stosunku do kratownicy z rozdziału 7.1 w wykonywanych obliczeniach został zaproponowany następujący schemat optymalizacji. W pierwszym kroku uruchomiony został algorytm standardowy PSO (bez wprowadzania żadnych zmian). W kroku drugim wprowadzono w nim pewną modyfikację. Polegała ona na tym, że wynik otrzymany z pierwszego uruchomienia dodatkowo wprowadzono do algorytmu. Optymalne rozwiązanie z poprzedniego uruchomienia – pozycję cząstki, nadano jednej z cząstek w momencie losowego generowania roju w kroku drugim. Taka modyfikacja sprawia, że od początku procesu optymalizacji algorytm PSO ma informację o rozwiązaniu z poprzedniego uruchomienia i stara się je poprawić. Taką samą procedurę zastosowano w trzecim kroku, wprowadzając informację o wyniku optymalizacji z kroku drugiego.

Procedurę optymalizacji dla każdego z kroków uruchamiano kilkukrotnie zarówno dla zmiennych ciągłych jak i dyskretnych. Najlepsze otrzymane wyniki zaprezentowano w dalszej części rozdziału.

W pierwszej kolejności został wyznaczony jeden wspólny przekrój dla wszystkich elementów prętowych, tak aby konstrukcja spełniała nałożone

ograniczenia. Jest to konstrukcja spełniająca zadane ograniczenia, jednak nieoptymalna w sensie osiągalnych rozwiązań. Następnie został uruchomiony proces optymalizacji dzięki czemu możliwa była ocena zaoszczędzonej masy konstrukcji w wyniku optymalizacji.

Wyniki obliczeń bez optymalizacji

Poniżej zostały zaprezentowane wyniki obliczeń bez optymalizacji. Dobór przekroju został wykonany przy pomocy skryptu znajdującego się w załączniku nr 9.

Pole przekroju pręta wynosi:

$$A = 2190\ mm^2.$$

Masa konstrukcji wynosi:

$$M = 1793{,}1\ kg.$$

Optymalizacja konstrukcji dla zmiennych ciągłych

Algorytm PSO został uruchomiony z następującymi parametrami zawartymi w tabeli 7.21.

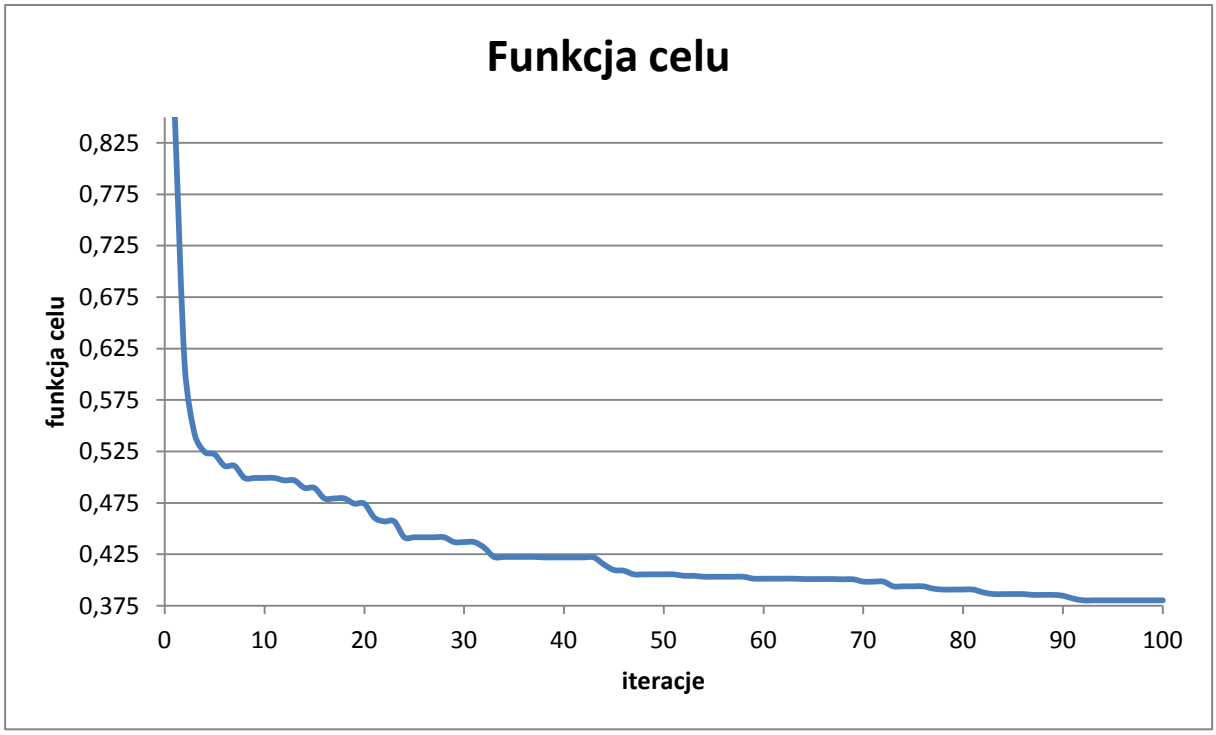
Tabela 7.21 Parametry algorytmu PSO

wmax	wmin	itmax	c1	c2	N	zmienne	vmax	vmin	ograniczenia
0,9	0,4	100	1,8	2,3	200	ciągłe	200	-200	naprężenia
									przemieszczenia

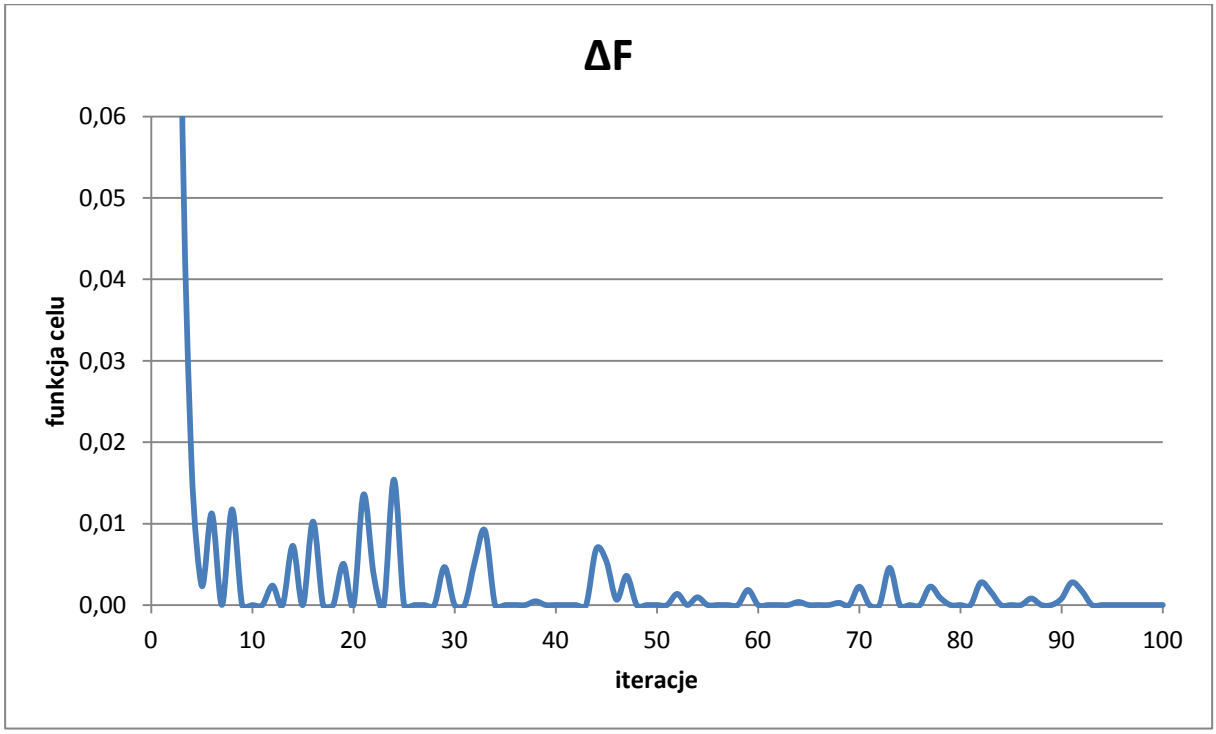
Wyniki optymalizacji prezentuje tabela 7.22 oraz rysunki 7.20-7.25.

Tabela 7.22 Wyniki optymalizacji (zmienne ciągłe)

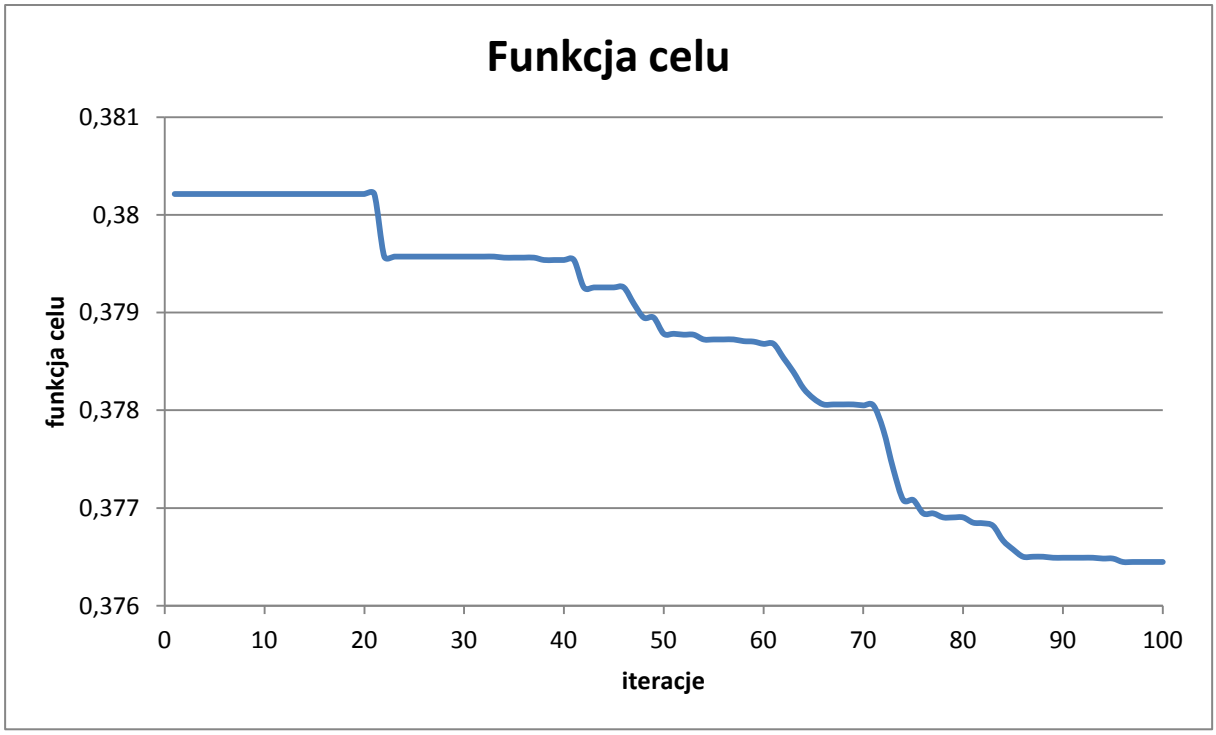
	Przekroje [mm^2]		
Nr elementu	**Krok 1**	**Krok 2**	**Krok 3**
1	1696,01	1798,60	1959,45
2	1992,21	1582,18	1600,45
3	2992,39	2951,20	3058,93
4	4000,00	4000,00	4000,00
5	2806,32	2603,42	2605,74
6	4000,00	4000,00	4000,00
7	3902,53	3681,72	1852,88
8	2526,84	2536,64	2565,32
9	2266,15	2393,22	2318,91
10	2331,14	2199,05	2122,08
11	2589,06	2476,74	2780,51
12	2221,93	2297,36	2126,48
13	2744,33	2694,04	2615,25
14	2590,65	2521,25	2737,00
15	346,81	328,58	331,19
16	932,30	1043,84	983,09
17	404,17	420,19	402,06
18	571,13	686,24	688,56
19	1154,60	1106,96	1153,73
20	20,00	20,00	20,00
21	20,00	20,00	20,00
22	1004,86	969,50	1159,21
23	521,50	628,35	709,14
24	504,23	514,79	416,09
25	1091,27	1062,11	959,16
26	336,48	329,28	328,91
Masa [kg]	1245,2	1232,9	1206,0
Funkcja celu	0,38021	0,37645	0,36823
Czas obliczeń [s]	82	82	82



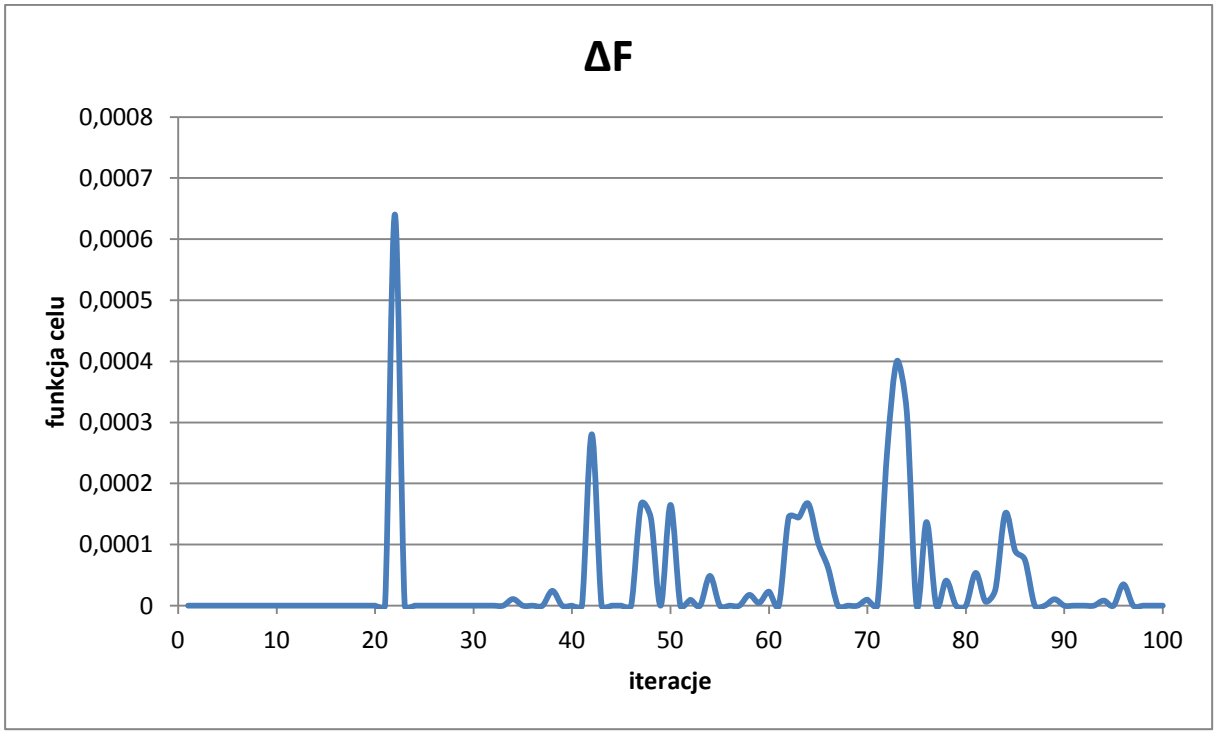
Rysunek 7.20 Wykres wartości funkcji celu w poszczególnych iteracjach – krok 1



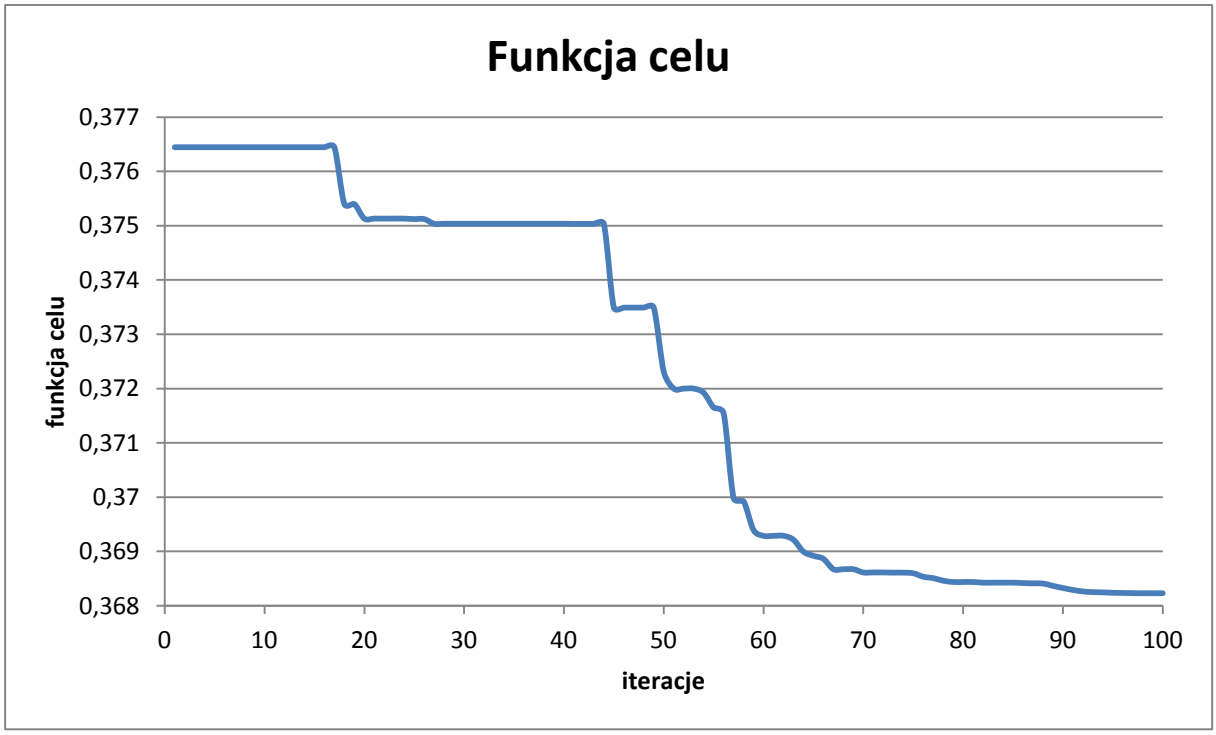
Rysunek 7.21 Wykres zmian wartości funkcji celu w poszczególnych iteracjach – krok 1



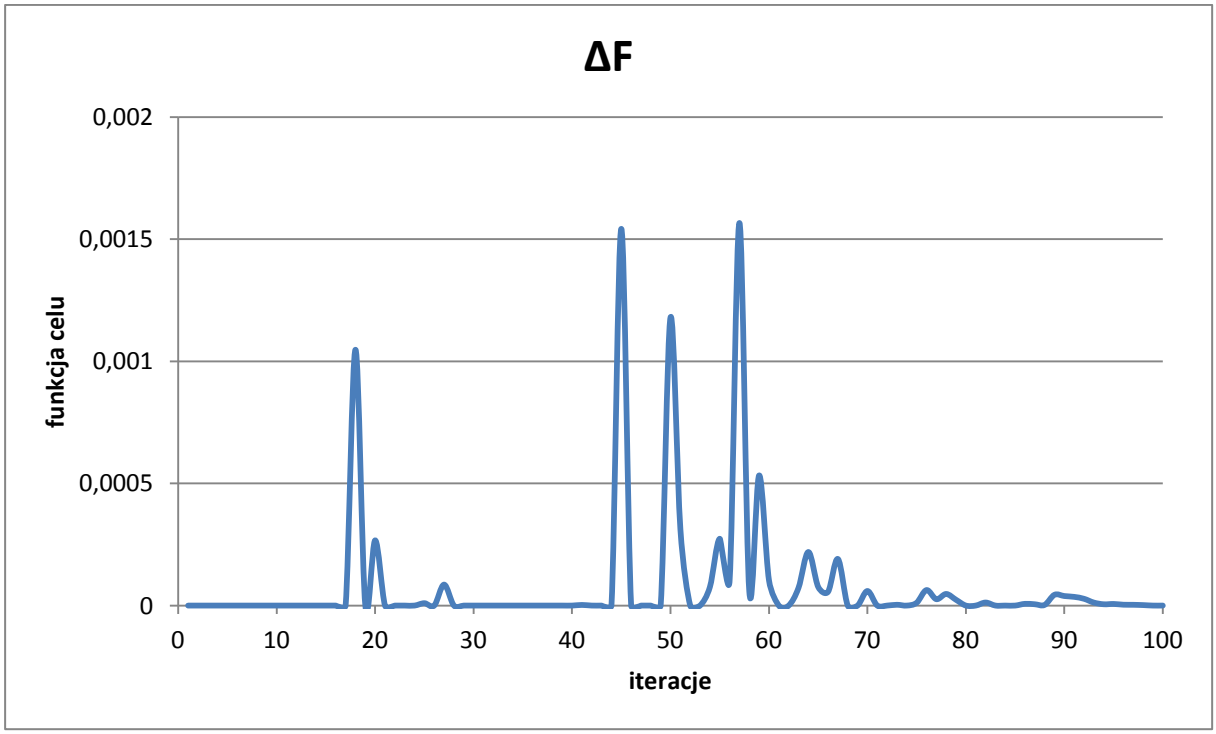
Rysunek 7.22 Wykres wartości funkcji celu w poszczególnych iteracjach – krok 2



Rysunek 7.23 Wykres zmian wartości funkcji celu w poszczególnych iteracjach – krok 2



Rysunek 7.24 Wykres wartości funkcji celu w poszczególnych iteracjach – krok 3



Rysunek 7.25 Wykres zmian wartości funkcji celu w poszczególnych iteracjach – krok 3

Optymalizacja konstrukcji dla zmiennych dyskretnych

Algorytm PSO został uruchomiony z następującymi parametrami zawartymi w tabeli 7.23. Dopuszczalne wartości zmiennych dyskretnych podano na początku niniejszego rozdziału.

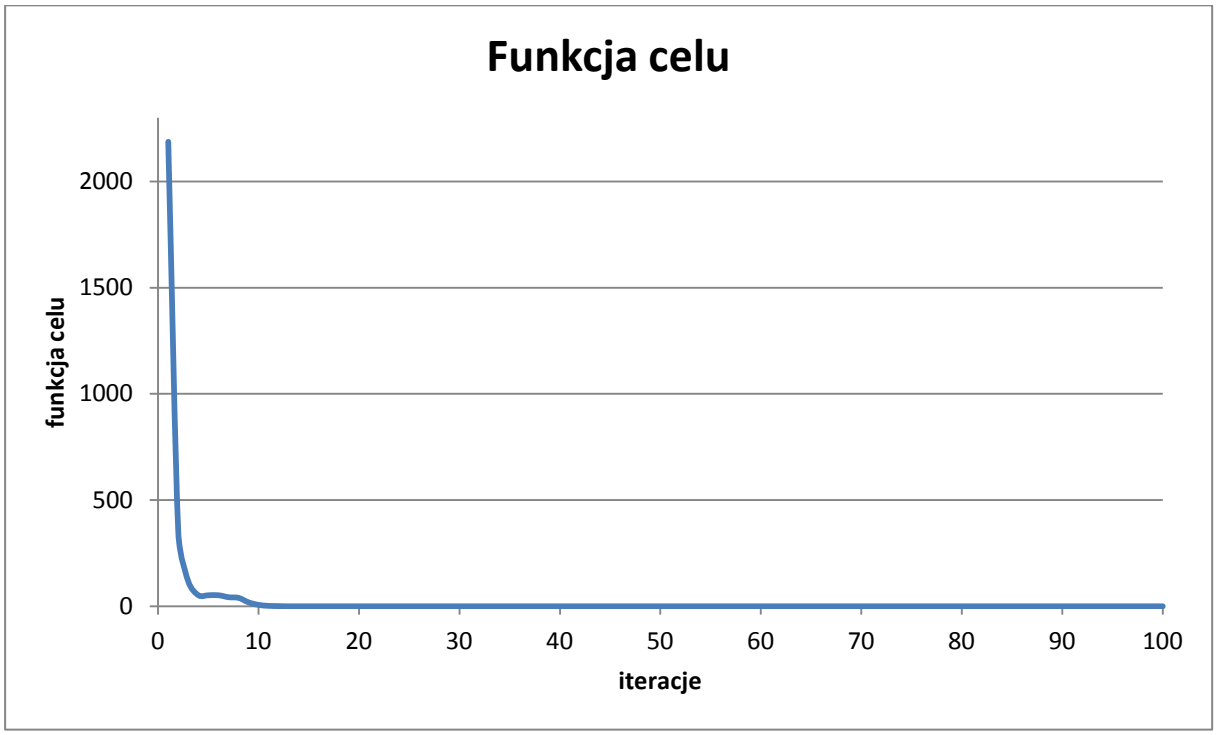
Tabela 7.23 Parametry algorytmu PSO

wmax	wmin	itmax	c1	c2	N	zmienne	vmax	vmin	ograniczenia
0,9	0,4	100	1,8	2,3	200	dyskretne	200	-200	naprężenia
									przemieszczenia

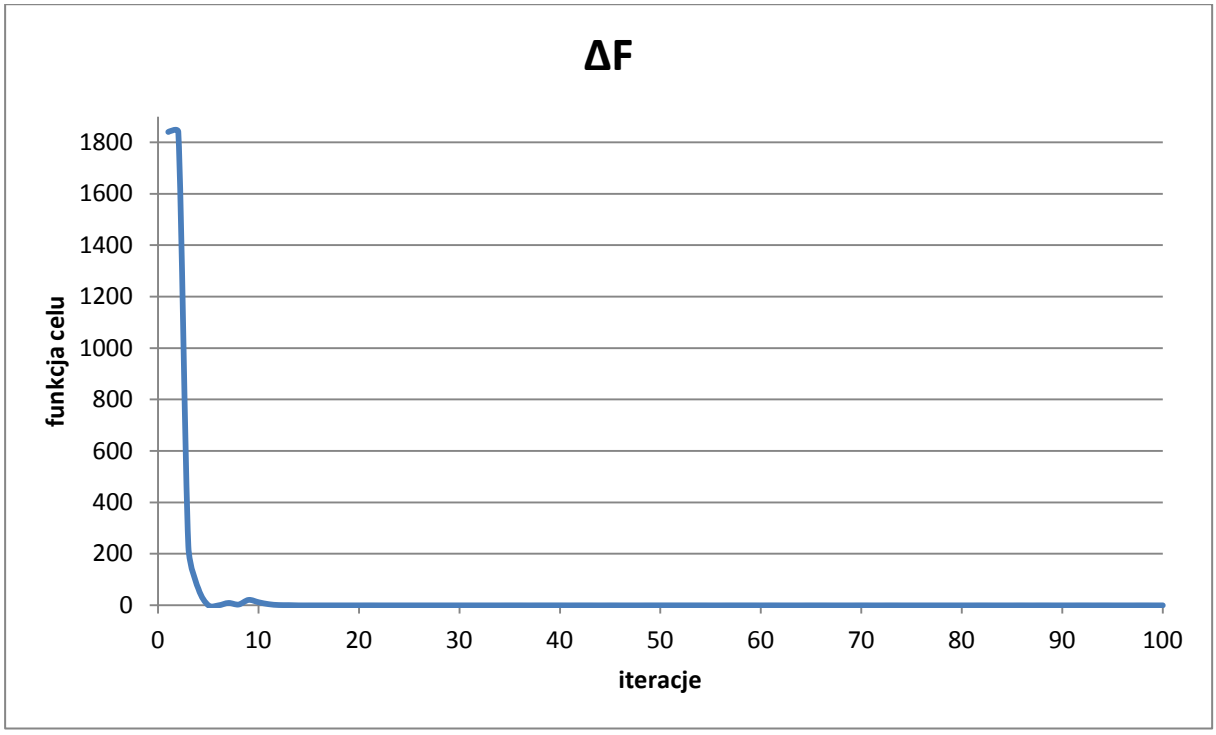
Wyniki optymalizacji prezentuje tabela 7.24 oraz rysunki 7.26-7.31.

Tabela 7.24 Wyniki optymalizacji (zmienne dyskretne)

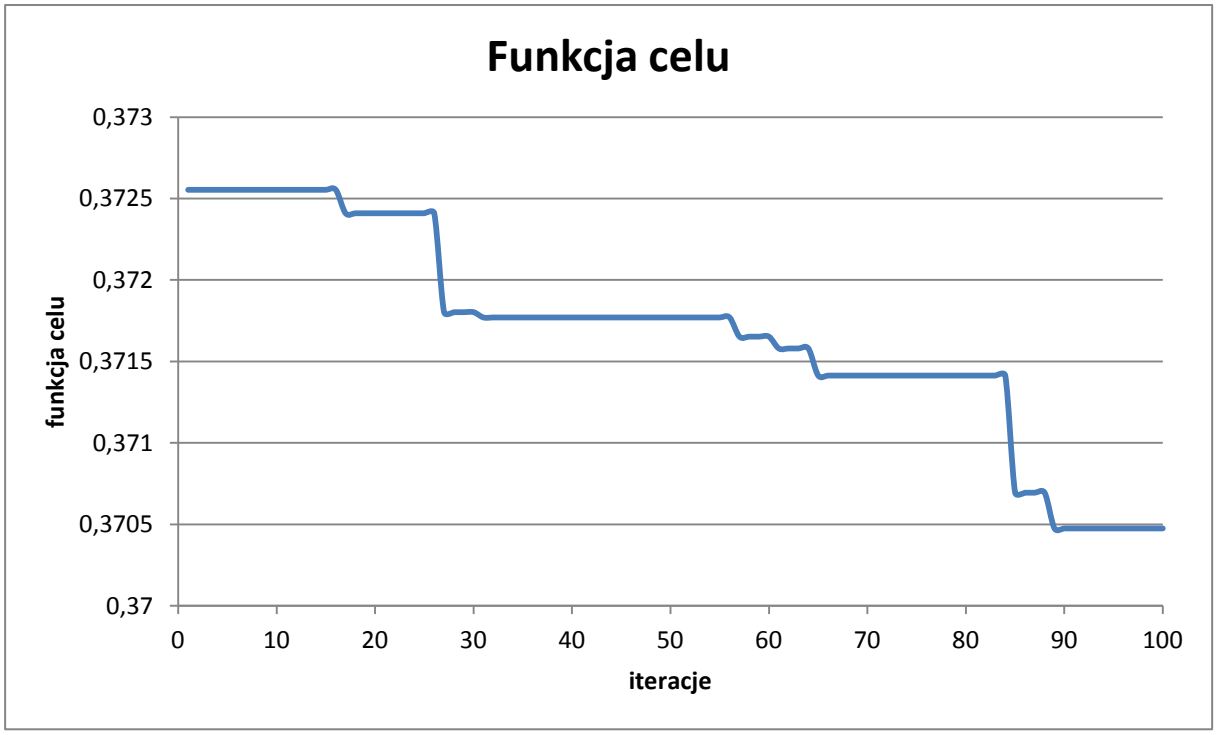
	Przekroje [mm^2]		
Nr elementu	**Krok 1**	**Krok 2**	**Krok 3**
1	1560	1560	1760
2	1560	1560	1560
3	3120	2880	2880
4	2680	2680	2880
5	4000	4000	4000
6	1760	1560	1560
7	1760	1760	1560
8	2440	2440	2440
9	2440	2240	2240
10	4000	4000	4000
11	2680	2680	2680
12	2000	2440	2240
13	2440	2240	2680
14	4000	4000	2680
15	440	440	440
16	880	1000	1120
17	320	440	440
18	880	760	760
19	1000	1000	1120
20	20	20	20
21	20	20	20
22	880	1000	1000
23	880	680	760
24	440	440	440
25	880	1000	1120
26	440	440	440
Masa [kg]	1220,2	1213,3	1193,5
Funkcja celu	0,37255	0,37047	0,364406
Czas obliczeń [s]	220	210	227



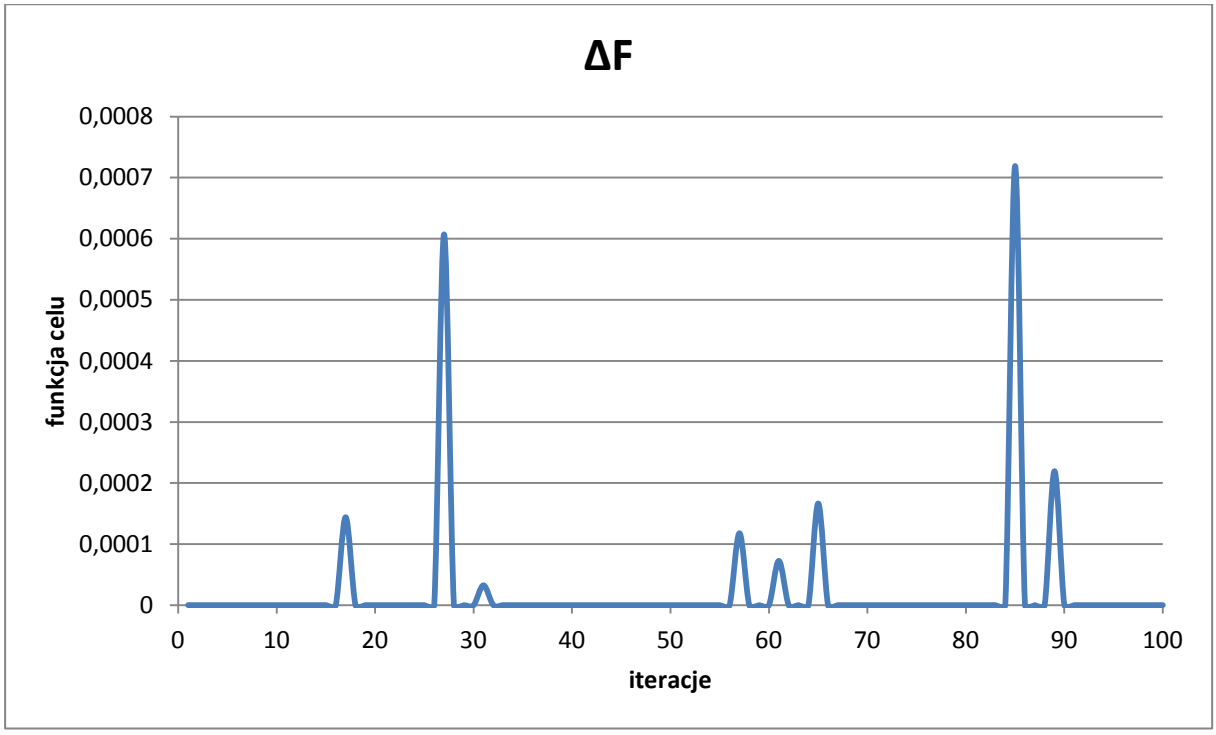
Rysunek 7.26 Wykres wartości funkcji celu w poszczególnych iteracjach – krok 1



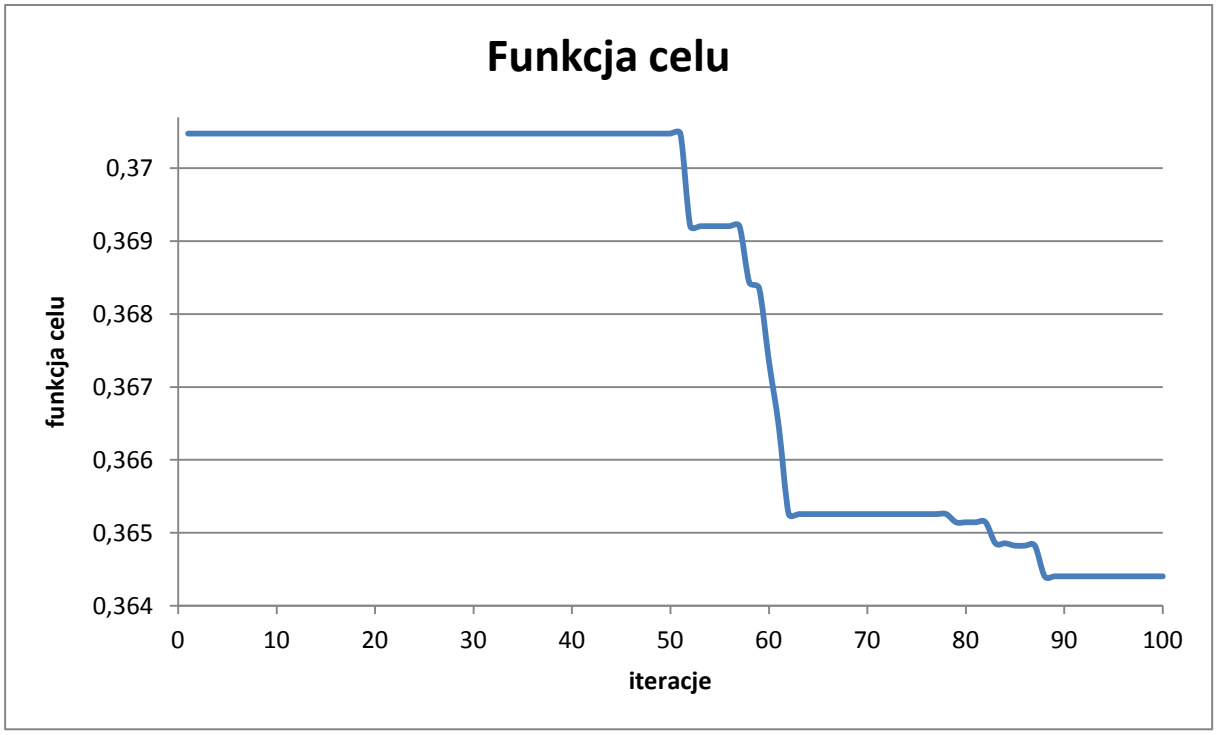
Rysunek 7.27 Wykres zmian wartości funkcji celu w poszczególnych iteracjach – krok 1



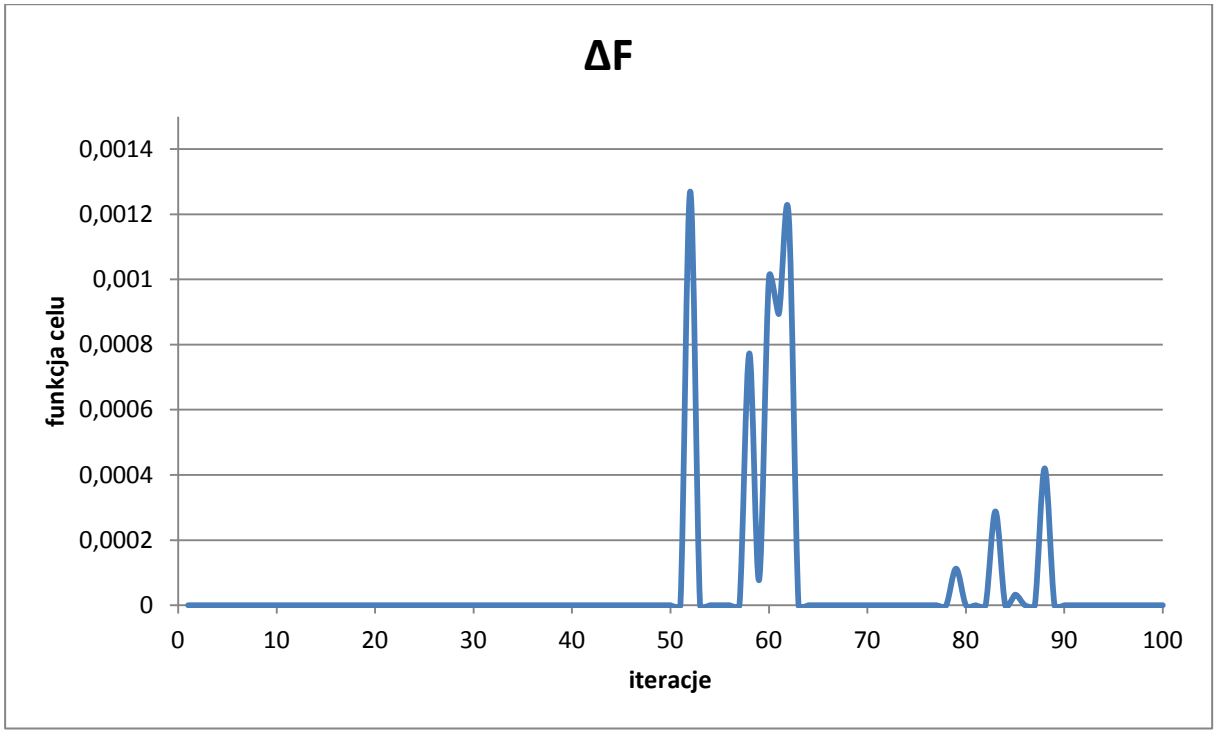
Rysunek 7.28 Wykres wartości funkcji celu w poszczególnych iteracjach – krok 2



Rysunek 7.29 Wykres zmian wartości funkcji celu w poszczególnych iteracjach – krok 2



Rysunek 7.30 Wykres wartości funkcji celu w poszczególnych iteracjach – krok 3



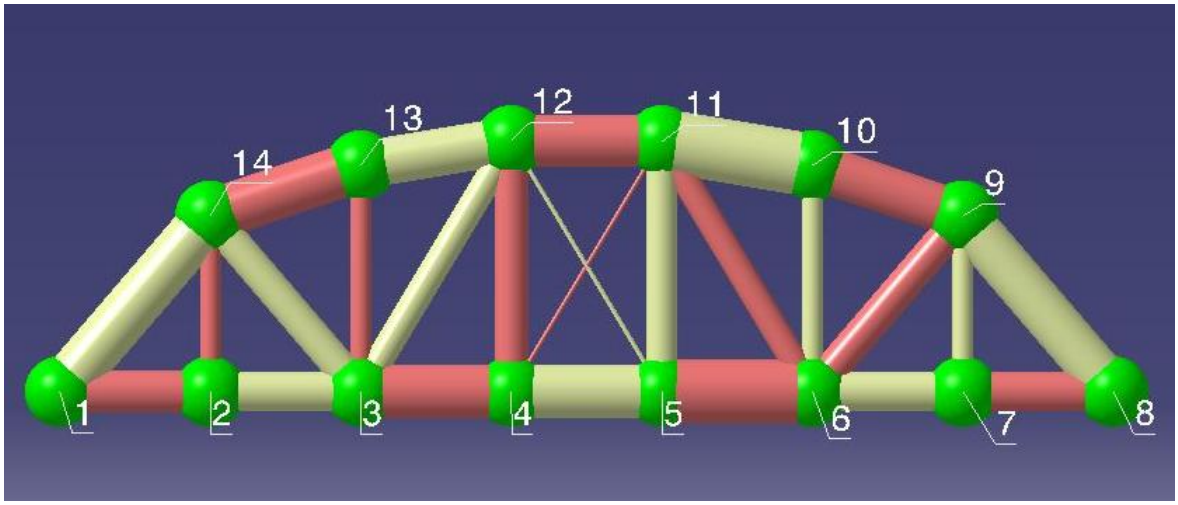
Rysunek 7.31 Wykres zmian wartości funkcji celu w poszczególnych iteracjach – krok 3

Porównanie otrzymanych wyników optymalizacji dla zmiennych ciągłych, dyskretnych oraz konstrukcji bez optymalizacji prezentuje tabela 7.25.

Tabela 7.25 Wyniki optymalizacji konstrukcji fragmentu mostu

	Zmienne ciągłe			Zmienne dyskretne			
	Krok 1	**Krok 2**	**Krok 3**	**Krok 1**	**Krok 2**	**Krok 3**	**Bez optymalizacji**
Masa [kg]	1245,2	1232,9	1206	1220,2	1213,3	1193,2	1793,1
Czas obliczeń [s]	82	82	82	220	210	227	-

Najlepszy otrzymany wynik przedstawiono w formie graficznej na rysunku 7.32. Wartości pól przekrojów zostały odpowiednio przeskalowane w stosunku do wymiarów konstrukcji, aby na rysunku były widoczne różnice pomiędzy prętami.



Rysunek 7.32 Graficzne przedstawienie wyników optymalizacji – najlepszy wynik (dla zmiennych dyskretnych)

Najlepszy otrzymany wynik (dla zmiennych dyskretnych) pozwala na zaoszczędzenie około 33% masy w stosunku do konstrukcji zaprojektowanej bez optymalizacji. Można również zauważyć, że optymalizacja dla zmiennych dyskretnych trwa dłużej od optymalizacji dla zmiennych ciągłych (w rozważanym przypadku ponad 2,5 raza dłużej). Spowodowane jest to tym, że za każdym razem w pętli optymalizacji, po obliczeniu przez PSO nowych współrzędnych cząstek program musi przeprowadzić dodatkowe obliczenia, aby dobrać wartości z katalogu dopuszczalnych przekrojów. W algorytmie opracowanym dla zmiennych ciągłych te obliczenia nie są przeprowadzane.

Należy również zwrócić uwagę na fakt, że lepsze wyniki uzyskano dla zmiennych dyskretnych, a nie ciągłych, jak można byłoby się spodziewać. Może być to spowodowane tym, że w pracy zastosowano prostą wersję algorytmu, bez żadnych dodatkowych modyfikacji poprawiających zbieżność algorytmu [5]. Dobór

przekrojów odbywa się przez zaokrąglanie obliczonych wartości pól przekroju, do najbliższej wartości z katalogu. Zatem czasem takie zaokrąglenie może powodować przekroczenie nałożonych na konstrukcję ograniczeń, a czasem może pomóc algorytmowi w znalezieniu lepszego rozwiązania.

Można więc sądzić, że użyty w niniejszej pracy algorytm sprawdza się dobrze dla zmiennych dyskretnych. Dla zmiennych ciągłych nie zawsze tak jest zwłaszcza, że widać wyraźnie iż dla ostatnich iteracji algorytm jest wolno zbieżny.

Jak widać na rysunkach 7.26-7.31 stopniowo z każdym krokiem procesu optymalizacji różnica w zmianie wartości funkcji zmniejsza się. Jednak nawet niewielkie jej zmiany mają wpływ na spadek masy konstrukcji. W pierwszych iteracjach funkcja celu przyjmowała duże wartości, ze względu na losowy dobór pól przekrojów. Konstrukcje takie nie spełniały nałożonych ograniczeń i wartość funkcji celu sztucznie była zwiększana przez funkcję kary. Można też sądzić, że zwiększenie ilości iteracji oraz liczby cząstek spowodowałoby uzyskanie lepszych wyników. Wraz ze wzrostem poziomu skomplikowania konstrukcji, liczby węzłów, liczby elementów czas obliczeń znacząco się wydłuża, szczególnie przy optymalizacji dyskretnej. Ze względu na dużą czasochłonność procesu optymalizacji algorytm uruchamiany był zaledwie kilka razy. Niewykluczone jest, że gdyby wykonać większą ilość uruchomień być może udałoby się uzyskać lepsze wyniki.

Warto też zwrócić uwagę na wpływ wartości parametru vmax/vmin na działanie algorytmu PSO. Zdefiniowany katalog dopuszczalnych wartości przekrojów zawiera skończoną liczbę elementów między Amin (20 mm^2) a Amax (4000 mm^2).

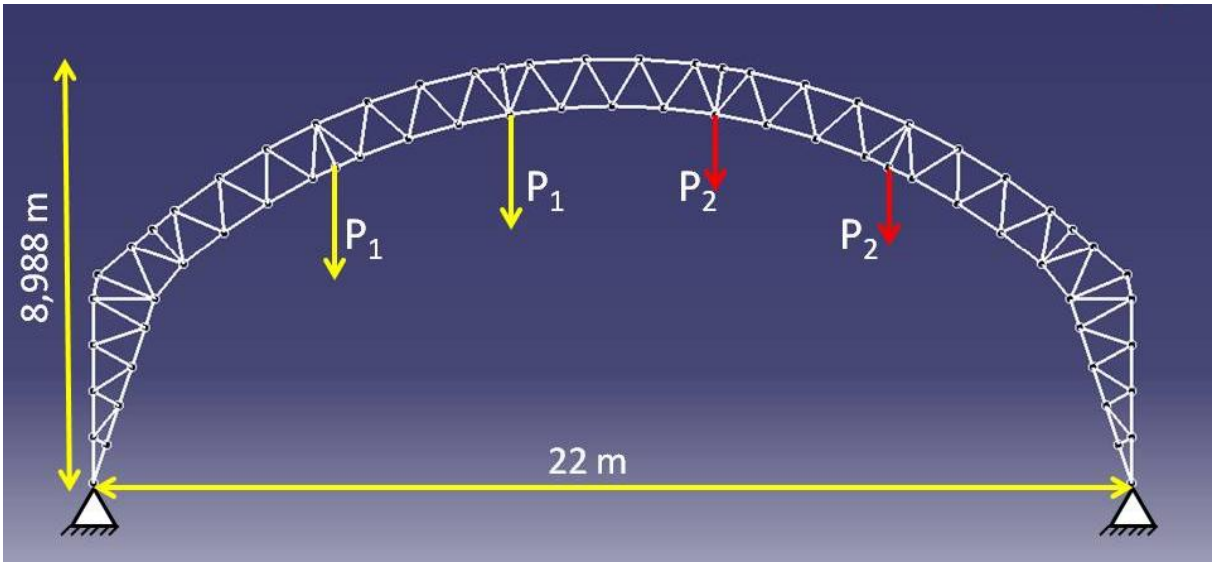
Podczas optymalizacji dyskretnej po wyliczeniu przez algorytm nowych współrzędnych cząstek dobierane są przekroje z katalogu. Przekrój dobierany jest na podstawie wartości różnicy między obliczonym przekrojem a elementami katalogu. Element, dla którego różnica ta jest najmniejsza staje się obliczonym przekrojem.

Jeśli więc ustawi się w PSO za małą wartość prędkości początkowych (dla pierwszej iteracji) – zbyt małą w stosunku do różnicy między elementami katalogu, proces optymalizacji nie będzie przebiegał poprawnie. W optymalizowanym w tym rozdziale przypadku największa różnica między dwoma sąsiednimi elementami wynosi ponad 200 mm^2. Zdefiniowana vmax=200 mm^2. Gdyby ustawić vmax np. na poziomie 10 mm^2 albo mniejszej, optymalizacja dyskretna nie zadziała poprawnie. Zostało to sprawdzone na podstawie testowych uruchomień programu. Należy więc zwrócić uwagę na dobór katalogu dopuszczalnych zmiennych dyskretnych (tu przekrojów) oraz na przedział z jakiego generowane są początkowe prędkości cząstek.

7.3 Model płaski konstrukcji łukowej

Ostatnim badanym przykładem jest wykonana z elementów prętowych konstrukcja łukowa.

Schemat, obciążenie oraz wymiary główne prezentuje rysunek 7.33.



Rysunek 7.33 Model konstrukcji łukowej zbudowanej z elementów prętowych – schemat konstrukcji, obciążenie oraz wymiary główne

W analizowanym przykadzie przyjęto następujące dane:

$$P_1 = 3000\ N$$
$$P_2 = 2000\ N$$
$$E = 70\ MPa - moduł\ Young'a$$
$$\rho = 2{,}8 \cdot 10^{-6}\ \frac{kg}{mm^3} - gęstość$$
$$\sigma_{dop} = 250\ MPa - naprężenia\ dopuszczalne$$
$$u_{dop} = 50\ mm - przemieszczenia\ dopuszczalne$$

Zmienne projektowe (przekroje prętów):
- zmienne ciągłe mogą przyjmować wartość z przedziału <Amin:Amax>

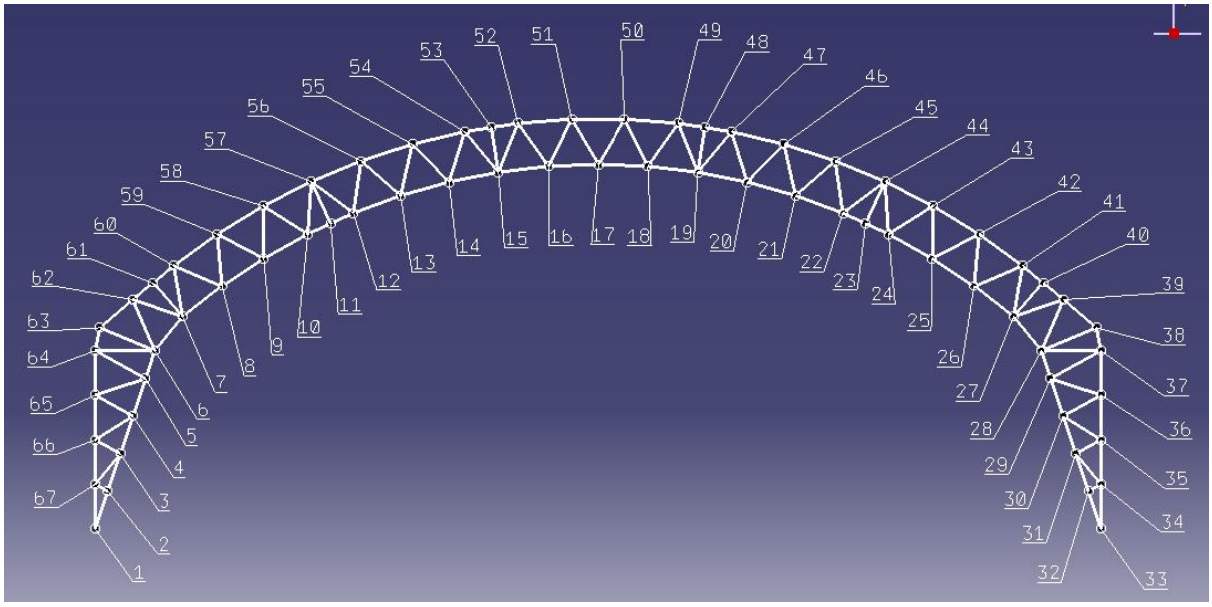
$$Amin = 10\ mm^2$$
$$Amax = 230\ mm^2$$

- zmienne dyskretne mogą przyjmować tylko wartości znajdujące się w wektorze **katalogA**.

katalogA = [10, 12, 14, 18, 25, 32, 39, 44, 51, 58, 64, 76, 90, 101, 115, 129, 140, 154, 166, 179, 191, 205, 216, 230]

Model konstrukcji, wymagane parametry, dane materiałowe, obciążenie, warunki brzegowe zostały zapisane w odpowiednim pliku, który wykorzystuje program PSO. Plik z danymi został zamieszczony w załączniku nr 8.

Rysunek 7.34 przedstawia przyjętą numerację węzłów. Numery elementów nie zostały naniesione na rysunek aby zachować jego czytelność. Definicja i numeracja elementów podana została w załączniku nr 8.



Rysunek 7.34 Model konstrukcji łukowej zbudowanej z elementów prętowych – oznaczenie węzłów

Analizowana konstrukcja składa się z 67 węzłów oraz 131 elementów. Liczba zmiennych problemu optymalizacji wynosi 131.

W przypadku konstrukcji łuku przyjęto taki sam schemat procesu optymalizacji jak w przypadku konstrukcji fragmentu mostu (opisany w rozdziale 7.2). Analogicznie do przykładu obliczanego w rozadziale 7.2 składa się on z trzech kroków.

Procedurę optymalizacji dla każdego z kroków uruchamiano kilkukrotnie zarówno dla zmiennych ciągłych jak i dyskretnych. Najlepsze otrzymane wyniki zaprezentowano w dalszej części rozdziału.

W pierwszej kolejności został wyznaczony jeden wspólny przekrój dla wszystkich elementów prętowych, tak aby konstrukcja spełniała nałożone ograniczenia. Jest to konstrukcja spełniająca zadane ograniczenia, jednak w kontekście sformułowango zadania optymalizacji nie jest optymalna.

Wyniki obliczeń bez optymalizacji

Poniżej zostały zaprezentowane wyniki obliczeń bez optymalizacji. Dobór przekroju został wykonany przy pomocy skryptu znajdującego się w załączniku nr 9.

Pole przekroju pręta wynosi:

$$A = 127{,}6\ mm^2.$$

Masa konstrukcji wynosi:

$$M = 47{,}4\ kg.$$

Optymalizacja konstrukcji dla zmiennych ciągłych

Algorytm PSO został uruchomiony z następującymi parametrami zawartymi w tabeli 7.26.

Tabela 7.26 Parametry algorytmu PSO

wmax	wmin	itmax	c1	c2	N	zmienne	vmax	vmin	ograniczenia
0,9	0,4	100	1,8	2,3	200	ciągłe	15	-15	naprężenia
									przemieszczenia

Wyniki uzyskane w poszczególnych krokach optymalizacji prezentują tabele 7.27-7.29 oraz rysunki 7.35-7.40.

Tabela 7.27 Wyniki optymalizacji (zmienne ciągłe) – krok 1.

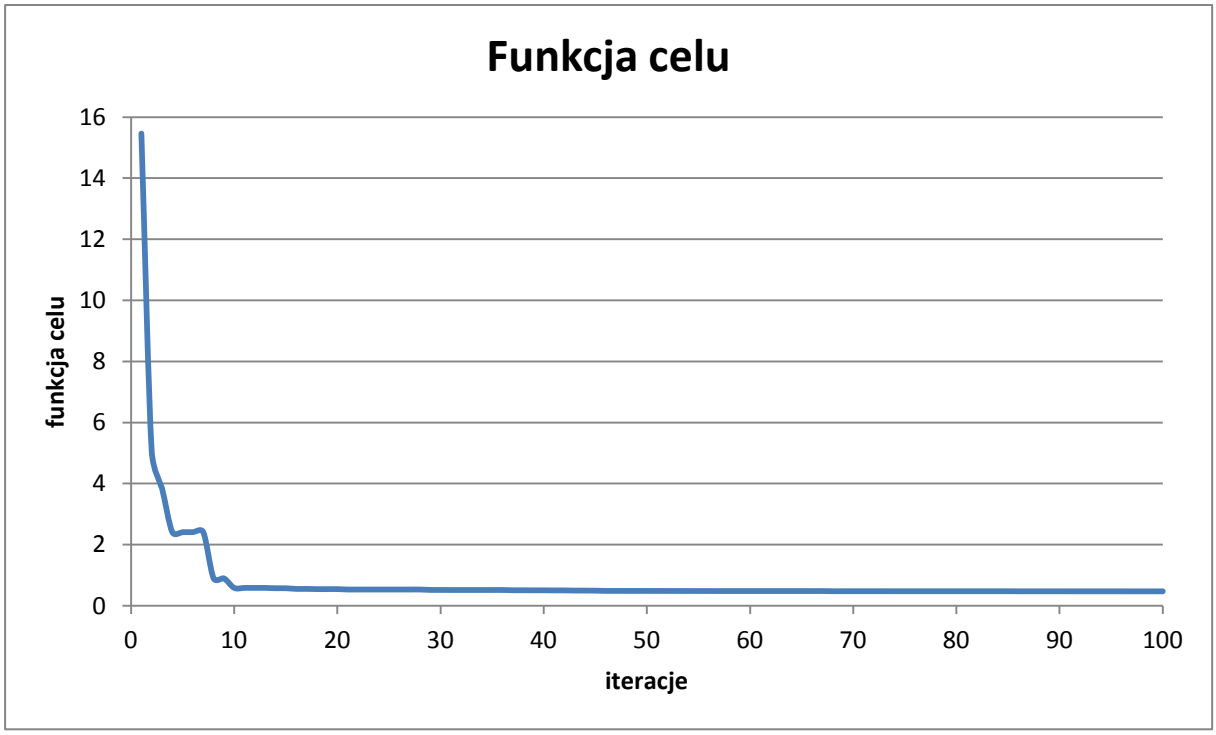
	Pole przekroju		Pole przekroju		Pole przekroju		Pole przekroju
Nr el.	[mm^2]	Nr el.	[mm^2]	Nr el.	[mm^2]	Nr el.	[mm^2]
1	124,61	41	144,81	81	41,50	121	10,00
2	115,87	42	52,85	82	52,50	122	39,02
3	162,82	43	83,84	83	89,41	123	47,16
4	215,12	44	183,75	84	132,13	124	151,50
5	206,11	45	171,51	85	145,78	125	10,28
6	176,07	46	103,39	86	55,64	126	118,82
7	134,14	47	138,69	87	117,47	127	171,68
8	225,88	48	149,75	88	200,60	128	92,88
9	216,58	49	128,25	89	60,95	129	135,39
10	16,33	50	143,27	90	40,29	130	70,97
11	10,00	51	159,62	91	186,65	131	143,95
12	16,20	52	168,94	92	144,09	**Masa [kg]**	40,56
13	146,48	53	191,59	93	48,38	**Funkcja celu**	0,474896
14	185,67	54	161,94	94	124,70	**Czas obliczeń [s]**	684
15	157,02	55	107,30	95	140,88		
16	88,67	56	116,29	96	20,42		
17	50,04	57	148,69	97	91,66		
18	47,63	58	85,74	98	26,27		
19	51,74	59	99,09	99	35,68		
20	61,47	60	181,90	100	90,85		
21	215,25	61	63,72	101	31,65		
22	134,27	62	69,89	102	18,06		
23	89,31	63	176,62	103	224,49		
24	218,25	64	71,36	104	28,44		
25	177,69	65	182,54	105	139,07		
26	120,71	66	80,46	106	47,52		
27	203,85	67	118,24	107	48,73		
28	187,70	68	93,78	108	230,00		
29	191,57	69	32,77	109	96,28		
30	203,02	70	62,68	110	19,37		
31	172,26	71	188,57	111	20,77		
32	161,40	72	75,92	112	57,84		
33	80,49	73	99,71	113	146,51		
34	193,16	74	86,56	114	154,55		
35	91,88	75	10,57	115	60,33		
36	74,42	76	51,40	116	16,36		
37	126,24	77	38,46	117	213,06		
38	187,05	78	10,00	118	216,10		
39	94,72	79	90,50	119	73,74		
40	132,47	80	52,11	120	117,80		

Tabela 7.28 Wyniki optymalizacji (zmienne ciągłe) – krok 2

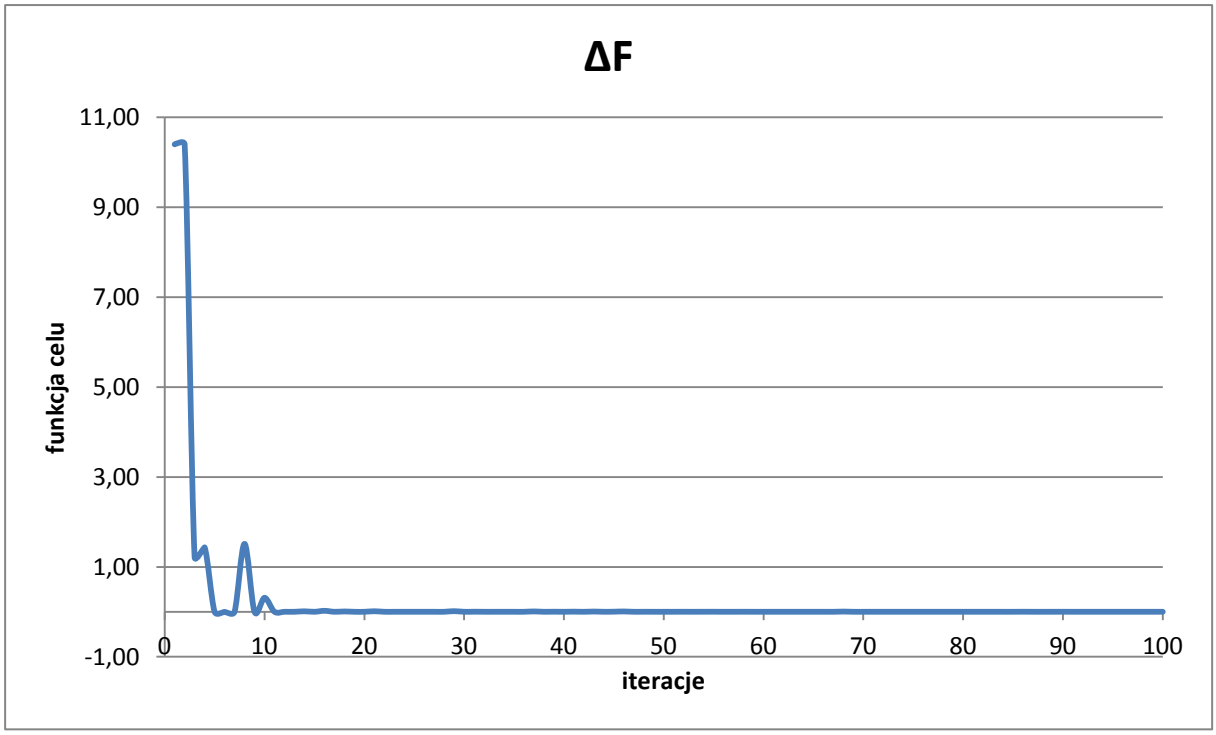
	Pole przekroju		Pole przekroju		Pole przekroju		Pole przekroju
Nr el.	[mm^2]	Nr el.	[mm^2]	Nr el.	[mm^2]	Nr el.	[mm^2]
1	201,18	41	79,69	81	24,91	121	10,00
2	162,69	42	107,80	82	56,42	122	42,45
3	193,16	43	76,56	83	29,12	123	70,33
4	213,74	44	157,55	84	70,10	124	49,84
5	155,30	45	143,58	85	73,34	125	10,00
6	178,29	46	138,91	86	100,09	126	95,70
7	142,45	47	79,35	87	117,73	127	176,70
8	230,00	48	100,14	88	213,36	128	121,75
9	200,85	49	160,64	89	45,00	129	79,20
10	10,17	50	189,88	90	42,18	130	39,02
11	10,00	51	163,12	91	150,48	131	84,49
12	10,00	52	156,83	92	115,02	**Masa [kg]**	37,47
13	125,06	53	161,65	93	49,85	**Funkcja celu**	0,438610
14	143,12	54	165,44	94	77,49	**Czas obliczeń [s]**	668
15	157,64	55	136,97	95	10,00		
16	115,65	56	89,85	96	23,75		
17	107,71	57	86,40	97	56,64		
18	44,71	58	29,23	98	41,34		
19	73,37	59	69,70	99	25,38		
20	101,03	60	172,39	100	53,95		
21	205,06	61	42,46	101	24,27		
22	175,76	62	68,74	102	10,79		
23	86,82	63	173,05	103	229,99		
24	213,80	64	61,16	104	43,23		
25	152,86	65	193,31	105	156,03		
26	159,64	66	62,34	106	47,02		
27	183,73	67	100,99	107	50,45		
28	196,63	68	109,41	108	230,00		
29	155,23	69	68,24	109	45,80		
30	186,77	70	44,71	110	31,03		
31	155,22	71	201,19	111	25,27		
32	180,66	72	123,76	112	30,97		
33	115,14	73	32,03	113	123,05		
34	167,77	74	36,78	114	41,36		
35	91,30	75	10,00	115	59,79		
36	73,26	76	65,85	116	10,00		
37	79,93	77	22,97	117	183,49		
38	171,65	78	10,00	118	161,60		
39	128,62	79	172,94	119	56,86		
40	131,06	80	69,93	120	113,67		

Tabela 7.29 Wyniki optymalizacji (zmienne ciągłe) – krok 3

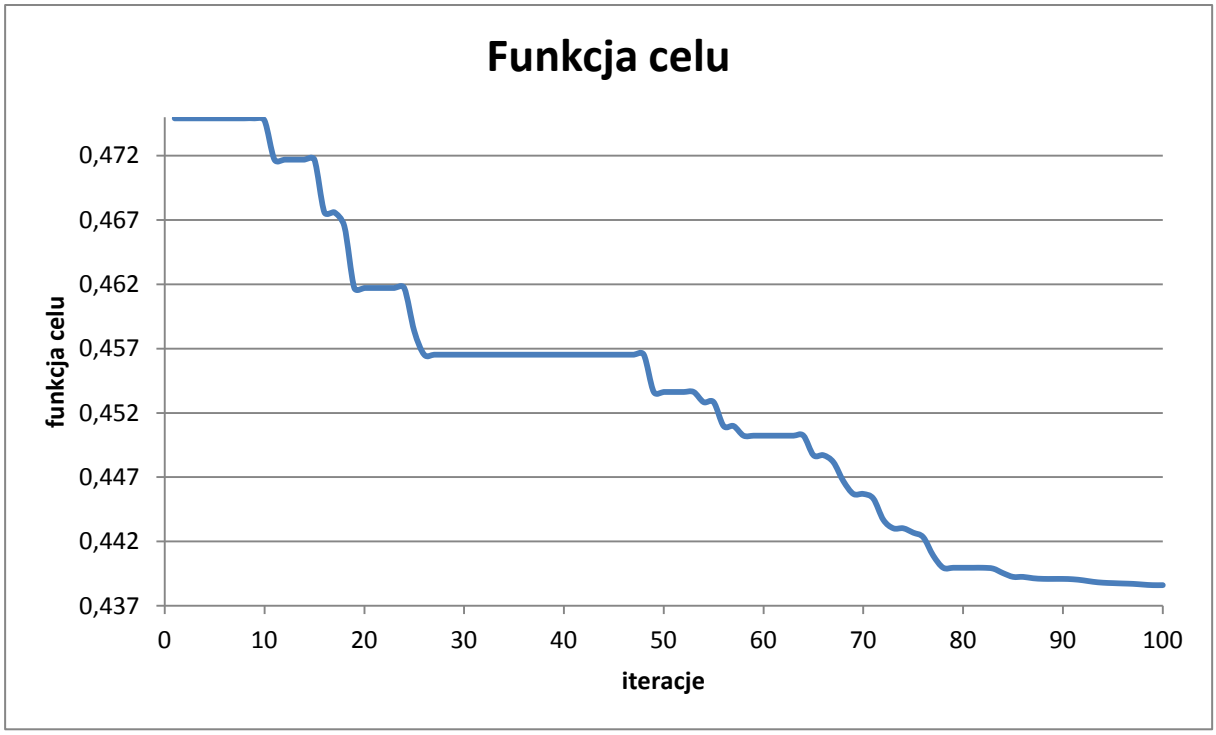
	Pole przekroju		Pole przekroju		Pole przekroju		Pole przekroju
Nr el.	[mm^2]	Nr el.	[mm^2]	Nr el.	[mm^2]	Nr el.	[mm^2]
1	173,22	41	110,74	81	30,57	121	10,00
2	180,05	42	79,79	82	62,82	122	45,13
3	186,12	43	13,53	83	38,07	123	61,07
4	209,84	44	10,00	84	73,72	124	46,13
5	167,79	45	34,18	85	69,47	125	10,00
6	170,08	46	114,82	86	78,05	126	65,75
7	131,11	47	127,90	87	72,07	127	144,90
8	230,00	48	94,87	88	199,08	128	32,70
9	176,88	49	150,76	89	61,61	129	81,76
10	10,00	50	201,96	90	34,33	130	17,26
11	10,00	51	180,32	91	30,84	131	38,36
12	10,00	52	170,85	92	39,74	**Masa [kg]**	34,55
13	158,22	53	214,36	93	63,22	**Funkcja celu**	0,404468
14	142,55	54	164,28	94	73,45	**Czas obliczeń [s]**	668
15	145,96	55	183,00	95	10,00		
16	92,55	56	92,66	96	26,50		
17	108,15	57	28,75	97	71,29		
18	73,91	58	11,21	98	38,82		
19	59,75	59	53,10	99	21,54		
20	10,00	60	207,56	100	33,87		
21	209,91	61	62,41	101	30,23		
22	188,42	62	80,10	102	10,18		
23	136,31	63	162,13	103	230,00		
24	209,84	64	65,56	104	50,13		
25	166,81	65	227,65	105	99,79		
26	161,47	66	110,71	106	47,55		
27	169,76	67	65,24	107	43,48		
28	178,78	68	98,87	108	230,00		
29	126,07	69	95,66	109	44,27		
30	229,92	70	17,87	110	36,78		
31	156,21	71	194,15	111	10,00		
32	160,44	72	76,48	112	32,44		
33	107,82	73	40,60	113	81,18		
34	159,54	74	51,32	114	70,05		
35	89,92	75	10,00	115	76,40		
36	75,16	76	60,19	116	10,00		
37	76,08	77	23,78	117	195,15		
38	152,11	78	10,00	118	148,48		
39	159,72	79	82,02	119	64,98		
40	172,86	80	79,29	120	100,89		



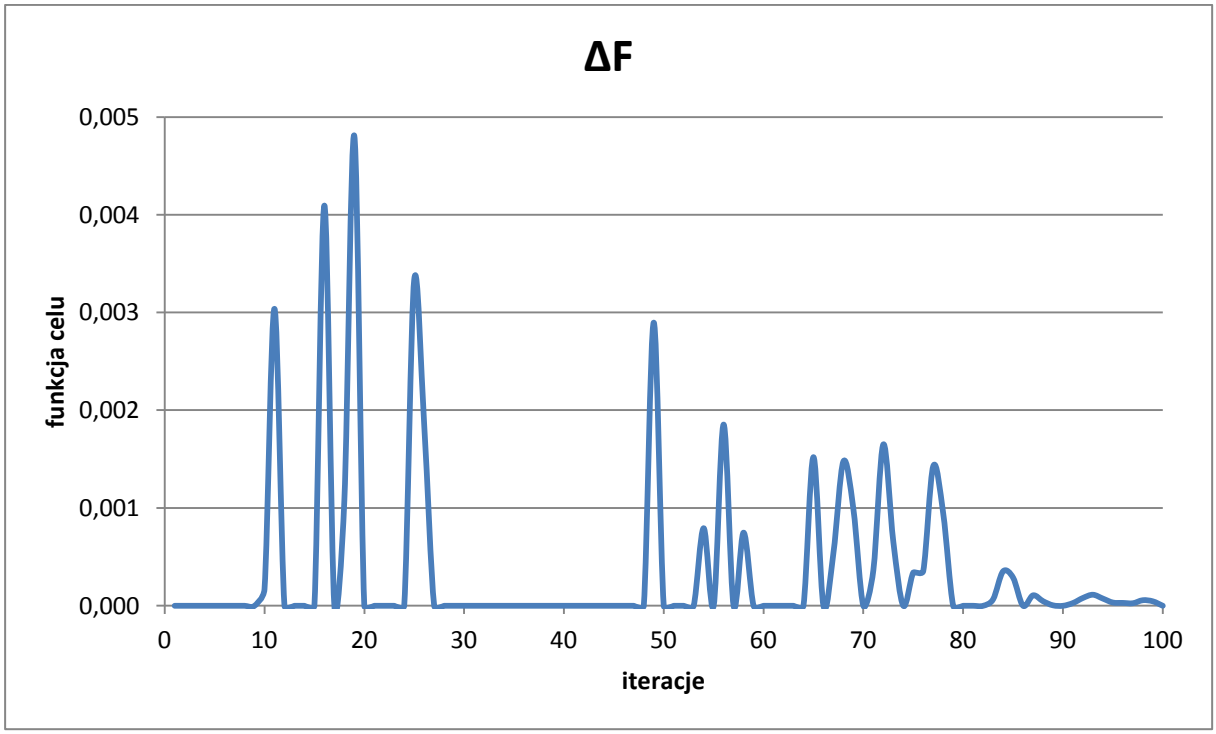
Rysunek 7.35 Wykres wartości funkcji celu w poszczególnych iteracjach – krok 1



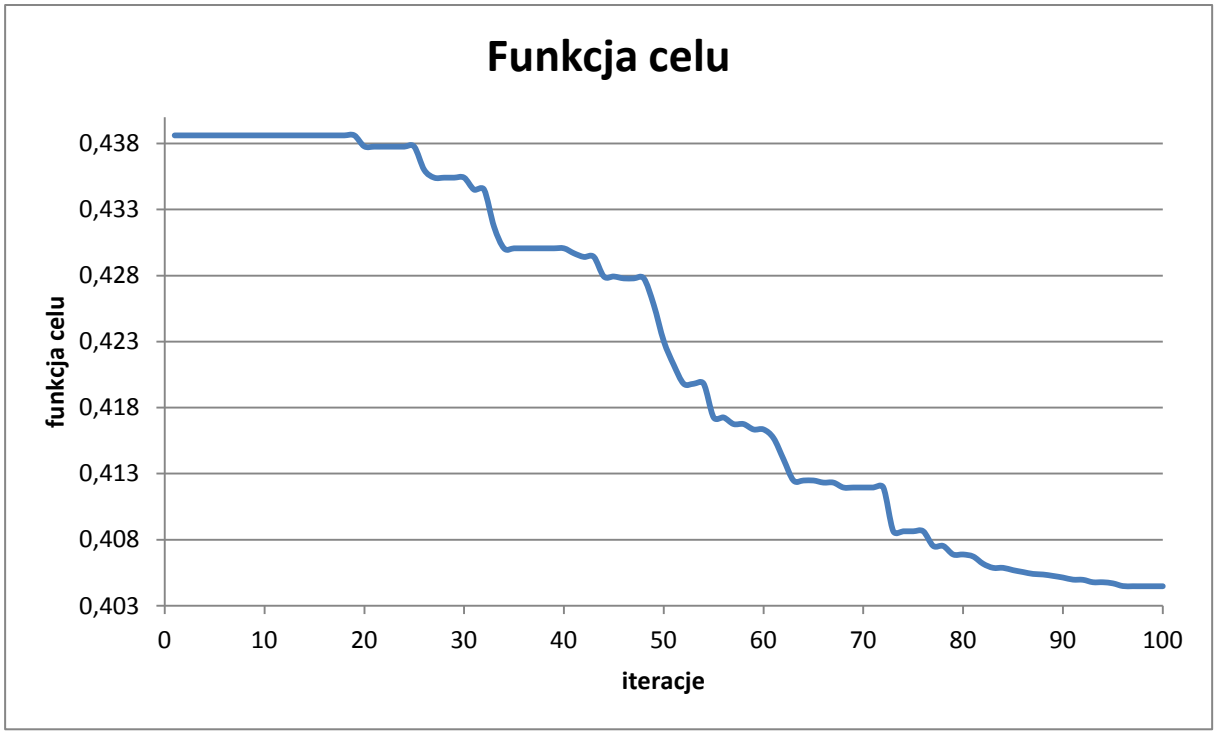
Rysunek 7.36 Wykres zmian wartości funkcji celu w poszczególnych iteracjach – krok 1



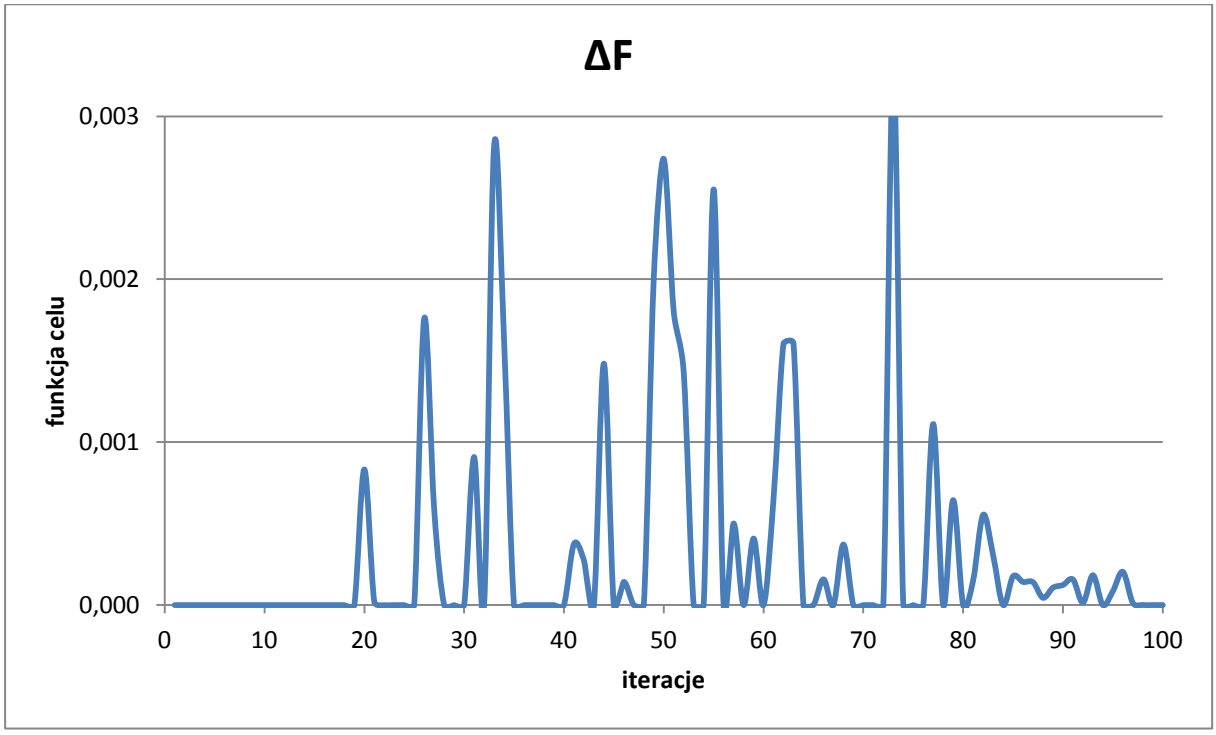
Rysunek 7.37 Wykres wartości funkcji celu w poszczególnych iteracjach – krok 2



Rysunek 7.38 Wykres zmian wartości funkcji celu w poszczególnych iteracjach – krok 2



Rysunek 7.39 Wykres wartości funkcji celu w poszczególnych iteracjach – krok 2



Rysunek 7.40 Wykres zmian wartości funkcji celu w poszczególnych iteracjach – krok 3

Optymalizacja konstrukcji dla zmiennych dyskretnych

Algorytm PSO został uruchomiony z następującymi parametrami zawartymi w tabeli 7.30.

Tabela 7.30 Parametry algorytmu PSO

wmax	wmin	itmax	c1	c2	N	zmienne	vmax	vmin	ograniczenia
0,9	0,4	100	1,8	2,3	200	dyskretne	15	-15	naprężenia
									przemieszczenia

Wyniki uzyskane w poszczególnych krokach optymalizacji prezentują tabele 7.31-7.33 oraz rysunki 7.41-7.46.

Tabela 7.31 Wyniki optymalizacji (zmienne dyskretne) – krok 1.

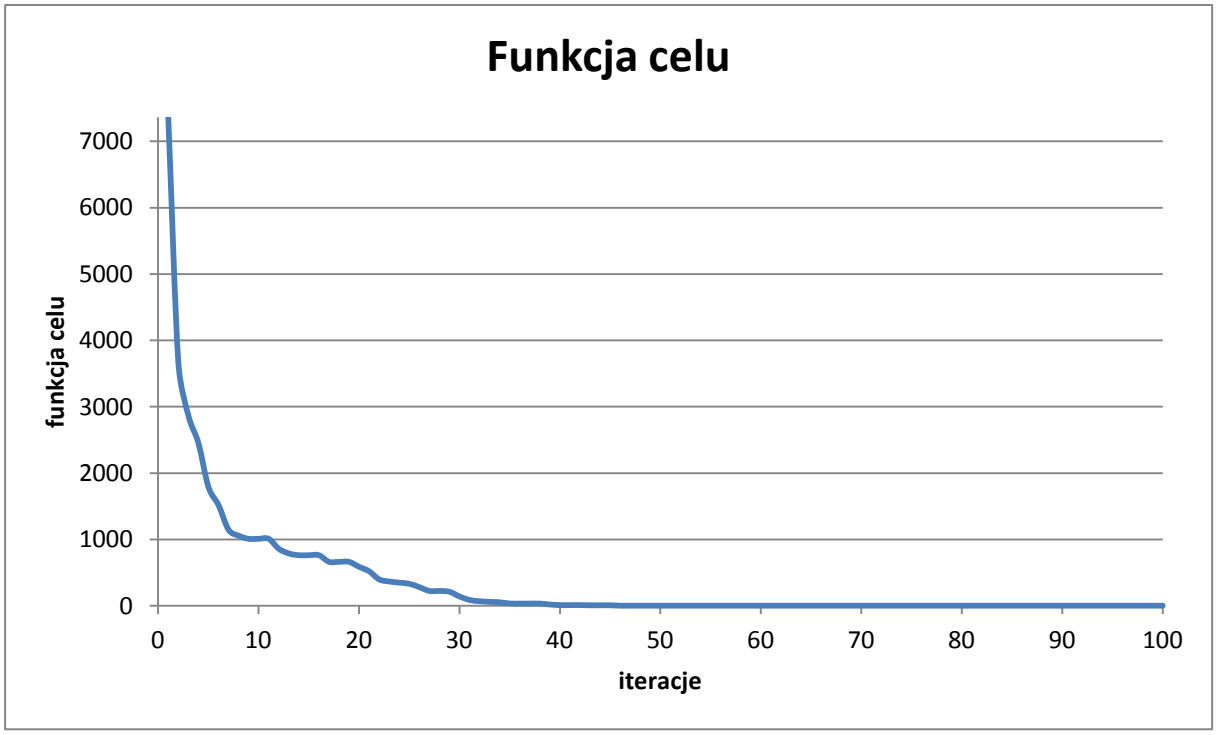
	Pole przekroju		Pole przekroju		Pole przekroju		Pole przekroju
Nr el.	**[mm^2]**	**Nr el.**	**[mm^2]**	**Nr el.**	**[mm^2]**	**Nr el.**	**[mm^2]**
1	101	41	140	81	10	121	10
2	230	42	230	82	58	122	18
3	205	43	44	83	44	123	101
4	230	44	10	84	25	124	10
5	230	45	230	85	25	125	10
6	216	46	140	86	44	126	10
7	230	47	191	87	230	127	44
8	230	48	90	88	14	128	10
9	230	49	230	89	101	129	10
10	230	50	230	90	10	130	10
11	25	51	230	91	51	131	10
12	51	52	205	92	10	**Masa [kg]**	44,51
13	90	53	230	93	44	**Funkcja celu**	0,521066
14	129	54	230	94	10	**Czas obliczeń [s]**	1401
15	230	55	230	95	10		
16	140	56	230	96	10		
17	166	57	25	97	230		
18	129	58	10	98	64		
19	10	59	230	99	230		
20	230	60	179	100	18		
21	129	61	58	101	44		
22	230	62	129	102	230		
23	140	63	230	103	10		
24	230	64	140	104	230		
25	129	65	230	105	10		
26	230	66	230	106	64		
27	230	67	230	107	205		
28	230	68	44	108	58		
29	230	69	230	109	115		
30	216	70	10	110	25		
31	179	71	115	111	14		
32	230	72	10	112	10		
33	90	73	230	113	230		
34	179	74	25	114	39		
35	129	75	14	115	44		
36	230	76	230	116	64		
37	230	77	64	117	10		
38	230	78	230	118	230		
39	230	79	10	119	101		
40	64	80	39	120	10		

Tabela 7.32 Wyniki optymalizacji (zmienne dyskretne) – krok 2.

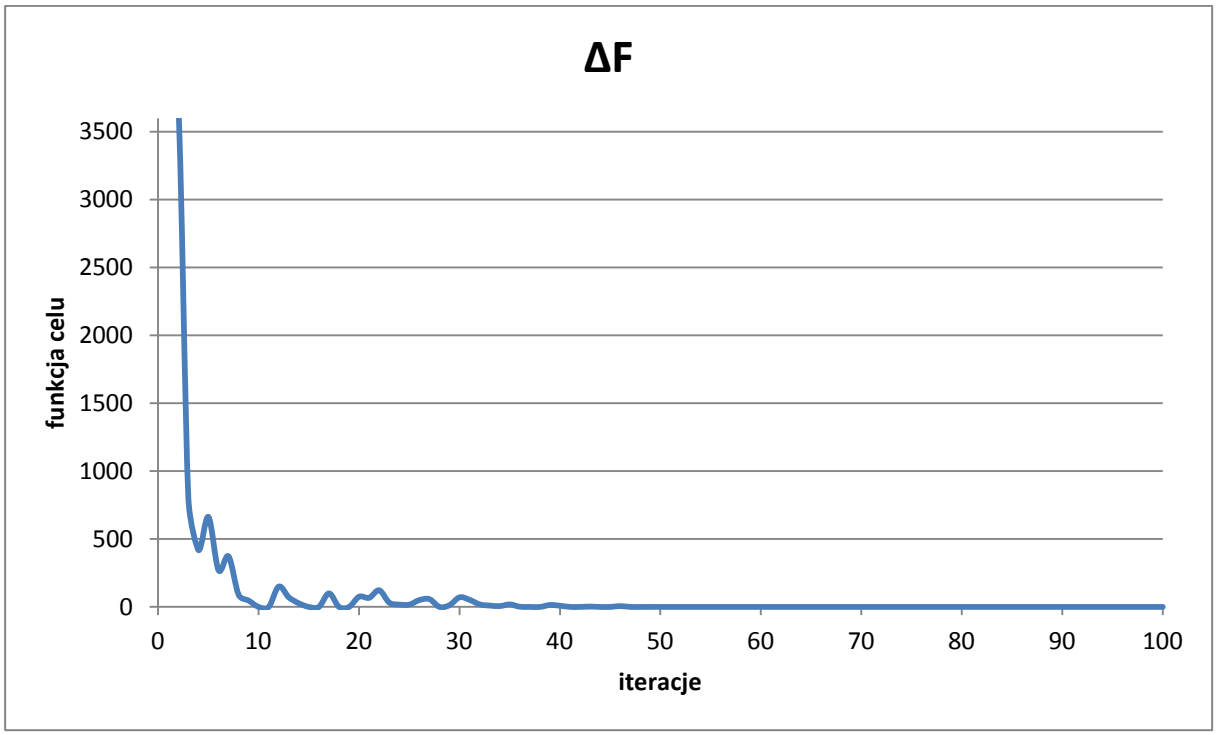
	Pole przekroju		Pole przekroju		Pole przekroju		Pole przekroju
Nr el.	**[mm^2]**	**Nr el.**	**[mm^2]**	**Nr el.**	**[mm^2]**	**Nr el.**	**[mm^2]**
1	166	41	140	81	10	121	10
2	230	42	230	82	64	122	14
3	205	43	44	83	64	123	64
4	230	44	10	84	44	124	10
5	230	45	230	85	39	125	10
6	205	46	64	86	64	126	10
7	230	47	115	87	230	127	44
8	230	48	154	88	44	128	10
9	230	49	230	89	51	129	10
10	230	50	230	90	10	130	10
11	25	51	230	91	39	131	10
12	18	52	230	92	10	**Masa [kg]**	41,44
13	58	53	230	93	44	**Funkcja celu**	0,485124
14	129	54	230	94	10	**Czas obliczeń [s]**	1634
15	230	55	230	95	10		
16	90	56	230	96	10		
17	39	57	44	97	230		
18	44	58	10	98	39		
19	10	59	230	99	230		
20	230	60	205	100	10		
21	51	61	101	101	14		
22	230	62	101	102	230		
23	140	63	230	103	10		
24	230	64	90	104	230		
25	205	65	230	105	14		
26	230	66	230	106	39		
27	230	67	230	107	101		
28	230	68	39	108	58		
29	230	69	230	109	64		
30	166	70	10	110	32		
31	179	71	10	111	25		
32	179	72	10	112	10		
33	115	73	230	113	230		
34	115	74	10	114	32		
35	115	75	10	115	32		
36	230	76	230	116	32		
37	230	77	18	117	10		
38	230	78	230	118	230		
39	230	79	10	119	25		
40	115	80	44	120	10		

Tabela 7.33 Wyniki optymalizacji (zmienne dyskretne) – krok 3.

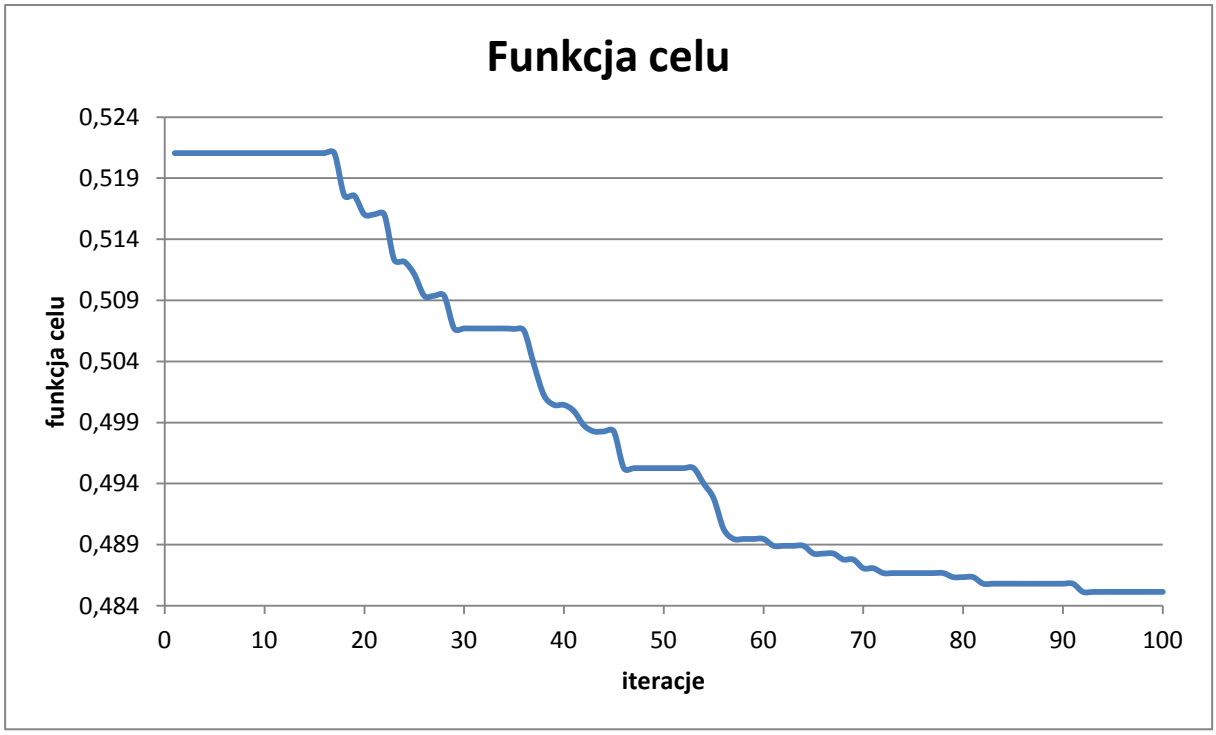
	Pole przekroju		Pole przekroju		Pole przekroju		Pole przekroju
Nr el.	**[mm^2]**	**Nr el.**	**[mm^2]**	**Nr el.**	**[mm^2]**	**Nr el.**	**[mm^2]**
1	115	41	115	81	25	121	39
2	230	42	90	82	32	122	39
3	140	43	39	83	58	123	51
4	216	44	10	84	44	124	10
5	179	45	64	85	51	125	10
6	230	46	51	86	64	126	10
7	191	47	129	87	230	127	10
8	216	48	90	88	39	128	64
9	154	49	101	89	44	129	10
10	129	50	230	90	44	130	14
11	44	51	191	91	58	131	10
12	32	52	179	92	39	**Masa [kg]**	32,86
13	32	53	179	93	44	**Funkcja celu**	0,384655
14	101	54	230	94	39	**Czas obliczeń [s]**	1607
15	129	55	230	95	10		
16	115	56	90	96	18		
17	39	57	58	97	154		
18	39	58	32	98	32		
19	32	59	230	99	230		
20	51	60	140	100	39		
21	101	61	101	101	14		
22	216	62	58	102	205		
23	129	63	166	103	10		
24	230	64	58	104	10		
25	191	65	154	105	32		
26	191	66	154	106	25		
27	179	67	154	107	58		
28	179	68	64	108	39		
29	191	69	205	109	39		
30	205	70	58	110	32		
31	154	71	10	111	39		
32	191	72	12	112	12		
33	115	73	179	113	115		
34	90	74	10	114	39		
35	101	75	10	115	32		
36	115	76	129	116	32		
37	230	77	10	117	58		
38	154	78	51	118	44		
39	140	79	10	119	44		
40	90	80	32	120	12		



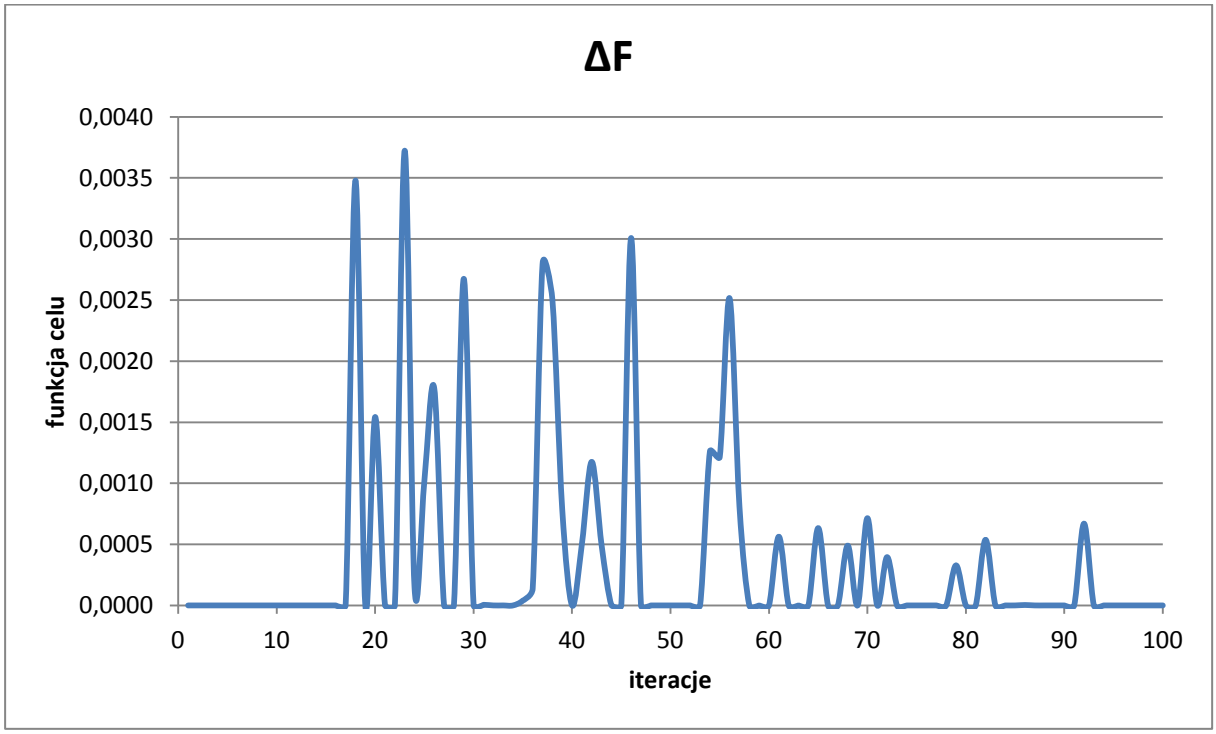
Rysunek 7.41 Wykres wartości funkcji celu w poszczególnych iteracjach – krok 1.



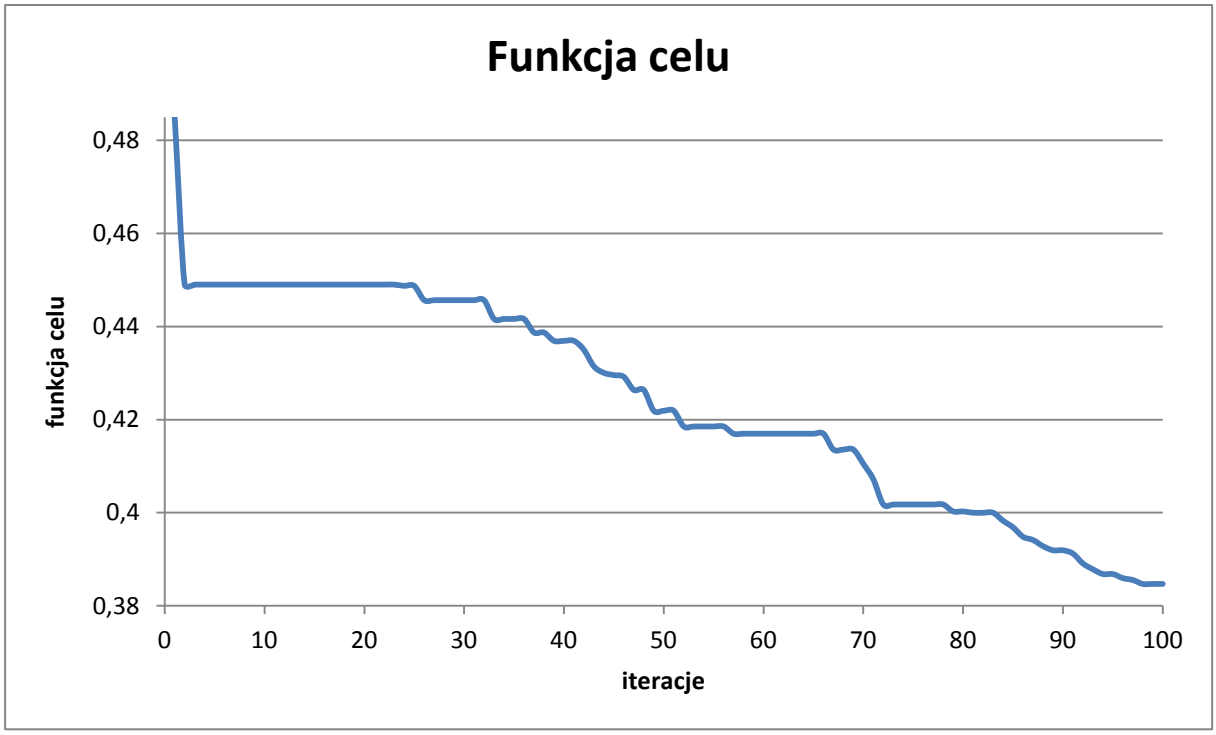
Rysunek 7.42 Wykres zmian wartości funkcji celu w poszczególnych iteracjach – krok 1.



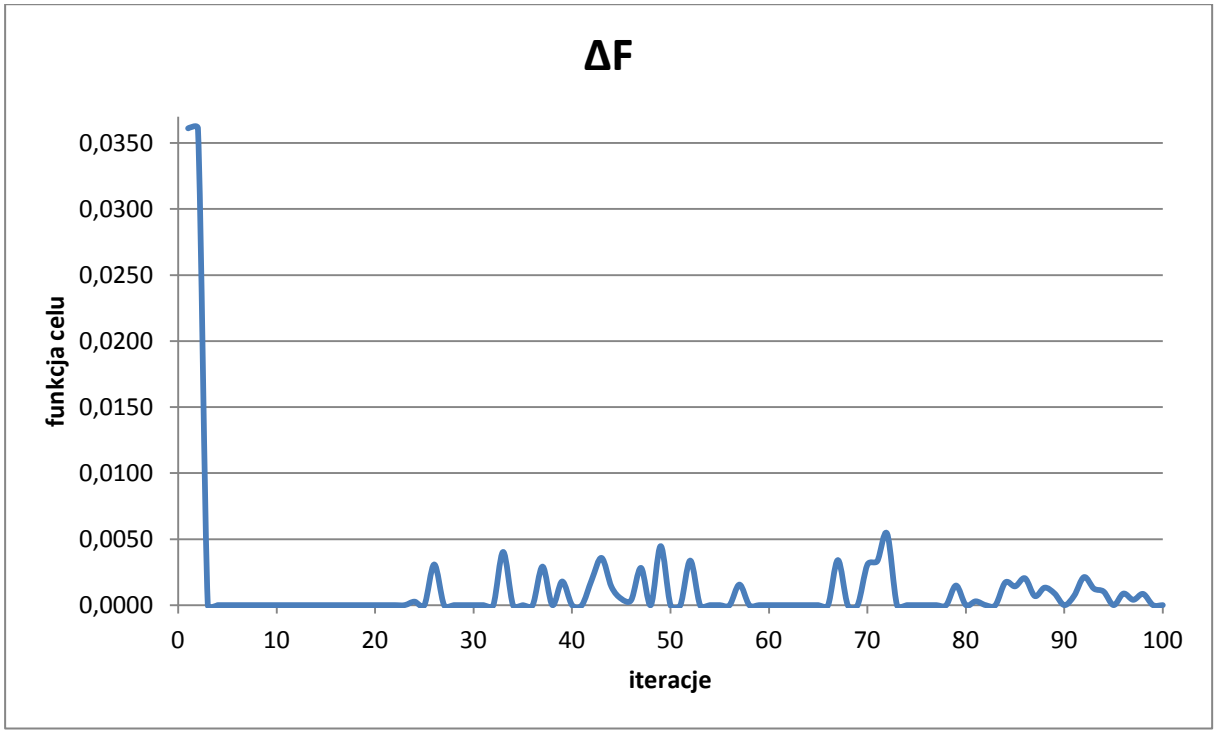
Rysunek 7.43 Wykres wartości funkcji celu w poszczególnych iteracjach – krok 2.



Rysunek 7.44 Wykres zmian wartości funkcji celu w poszczególnych iteracjach – krok 2.



Rysunek 7.45 Wykres wartości funkcji celu w poszczególnych iteracjach – krok 3.



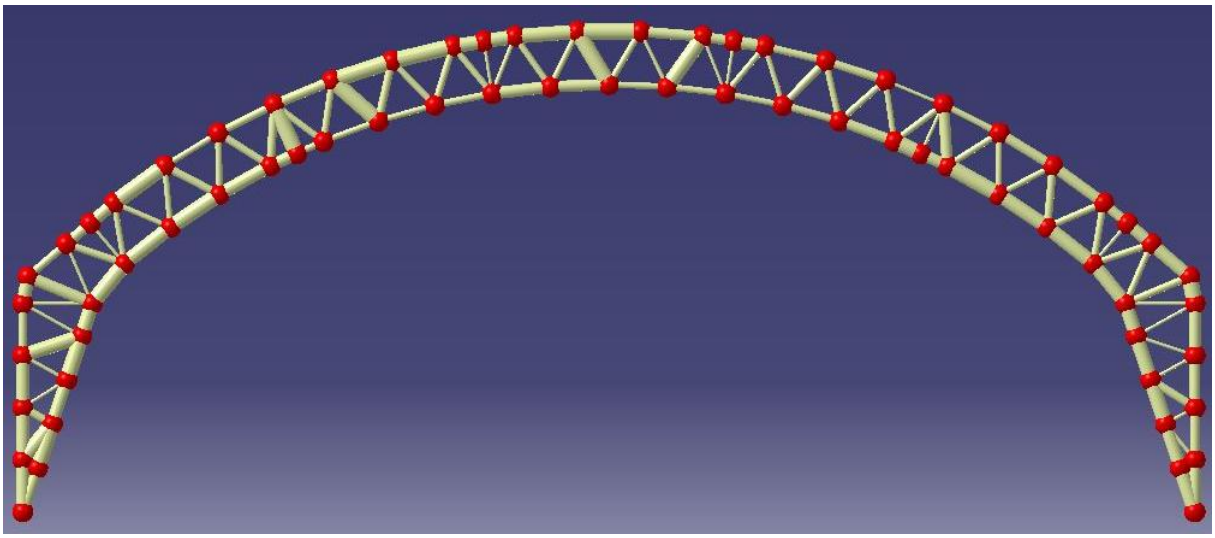
Rysunek 7.46 Wykres zmian wartości funkcji celu w poszczególnych iteracjach – krok 3.

Porównanie otrzymanych wyników optymalizacji dla zmiennych ciągłych, dyskretnych oraz konstrukcji bez optymalizacji prezentuje tabela 7.34.

Tabela 7.34 Porównanie otrzymanych wyników optymalizacji konstrukcji łukowej

	Zmienne ciągłe			Zmienne dyskretne			
	Krok 1	**Krok 2**	**Krok 3**	**Krok 1**	**Krok 2**	**Krok 3**	**Bez optymalizacji**
Masa [kg]	40,56	37,47	34,55	44,51	41,44	32,86	47,4
Czas obliczeń [s]	684	668	668	1401	1634	1607	-

Najlepszy otrzymany wynik przedstawiono w formie graficznej na rysunku 7.47. Wartości pól przekrojów zostały odpowiednio przeskalowane w stosunku do wymiarów konstrukcji, aby na rysunku były widoczne różnice pomiędzy prętami.



Rysunek 7.47 Graficzne przedstawienie wyników optymalizacji – najlepszy wynik (dla zmiennych dyskretnych)

W wyniku optymalizacji dla zmiennych ciągłych ostatecznie udało się zredukować masę o około 27% w stosunku do konstrukcji zaprojektowanej bez optymalizacji. Stosując zmienne dyskretne udało się oszczędzić na masie konstrukcji niecałe 31%. Otrzymana w wyniku optymalizacji konstrukcja nie jest symetryczna, jednak można w niej wyróżnić symetryczne podobszary. Wynika to z aktu, iż zdefiniowane obciążenie również nie było symetryczne (rysunek 7.33).

W tym jak i również w poprzednich przypadkach (fragment mostu, kratownica) czas obliczeń dla zmiennych dyskretnych był dłuższy od czasu obliczeń dla zmiennych ciągłych. Analizowany w tym rozdziale przypadek konstrukcji jest na tyle skomplikowany, że różnica w czasie obliczeń jest wyraźnie zauważalna. Dla zmiennych ciągłych jedna procedura optymalizacji trwała nieco ponad 11 minut, a dla zmiennych dyskretnych ponad 25 minut.

W tym przykładzie jak i w przykładzie optymalizacji fragmentu mostu z rozdziału 7.2 najlepszy wynik otrzymano w optymalizacji dla zmiennych dyskretnych, a nie jak możnaby się spodziewać dla zmiennych ciągłych. Powody tego zostały opisane już w rozdziale 7.2. Można jednak zauważyć (tabela 7.34), że w kroku 1. oraz w kroku 2. znacznie lepsze wyniki uzyskano w optymalizacji dla zmiennych ciągłych. Dopiero w ostatnim kroku nastąpiła znaczna poprawa wyniku w przypadku zmiennych dyskretnych. Należy pamiętać, że dla każdego kroku procedura optymalizacji uruchamiana była zaledwie kilka razy, a algorytm PSO zawiera elementy losowości. Żeby dokładniej móc ocenić wyniki i działanie algorytmu należałoby wykonać analizę większej ilości uruchomień. Tak samo ze względu na czas obliczeń i zasoby pamięci liczba cząstek oraz liczba iteracji były ograniczone. Można więc przypuszczać, że PSO w testowanej tu wersji dla skomplikowanych przypadków lepiej działa na zmiennych dyskretnych, a podczas pracy na zmiennych ciągłych ma problemy ze zbieżnością wyniku. Jednak aby być całkowicie tego pewnym należy wykonać większą ilość obliczeń i je ze sobą porównać. Ponadto uogólnianie uzyskanych wyników na inne przyklady powinno być dokonywane z dużą ostrożnością i rozwagą.

8 Wnioski

Na podstawie analizy wyników otrzymanych w trakcie optymalizacji kratownicy płaskiej (przykładu z pozycji [8]) można stwierdzić, że algorytm PSO daje zadowalające wyniki. Są one nieco gorsze niż w przypadku optymalizacji algorytmem ewolucyjnym [8] jednak nie odbiegają znacząco od wyników otrzymanych w [8]. Należy pamiętać, że obie metody są metodami niedeterministycznymi. Przez to, że zawierają w sobie elementy losowości, każde uruchomienie daje nieco inny wyniki. Dlatego wskazane jest (o ile możliwe) wielokrotne uruchamianie procesu optymalizacji, aby móc wyciągnąć konstruktywne wnioski służące dalszym obliczeniom lub też odpowiednie dobranie parametrów algorytmu aby zawsze uzsykać jak najlepsze rozwiązanie.

Przeprowadzone procesy optymalizacji dla kratownicy, fragmentu mostu oraz konstrukcji łukowej pokazują, że w wyniku optymalizacji można zredukować masę o nawet ponad 30% w stosunku do konstrukcji nie optymalizowanej, spełniającej tylko zadane ograniczenia dla jednakowych elementów składowych.

W niniejszej pracy w związku ze stosunkowo długim czasem obliczeń (wynikającym między innymi z faktu stosowania środowiska Scilab), algorytm PSO był uruchamiany o wiele mniej razy niż algorytm ewolucyjny w pozycji [8]. Zatem przedstawione "gorsze" wyniki mogą być spowodowane nie tylko doborem parametrów metody ale i niedostateczną liczbą uruchomień algorytmu PSO. Należy również zaznaczyć, że testowany w pracy algorytm ławicowy był użyty w bardzo prostej wersji. W literaturze proponowanych jest kilka modyfikacji pozwalających poprawić zbieżność algorytmu, takich jak np. PSO z wagą inercji (zastosowany w niniejszej pracy), PSO z wagą ścisku, PSO wykorzystujący różne topologie komunikacyjne [5]. W przypadku zastosowanego w pracy algorytmu, cząstka miała informację tylko o swoim najlepszym położeniu i położeniu lidera roju. Można to rozszerzyć, dokonać odpowiednich modyfikacji we wzorach i sprawić, że na ruch cząstki może oddziaływać dodatkowo lider, ale nie wybierany z całego roju, a tylko spośród sąsiadów z najbliższego otoczenia.

W samym modelu PSO z wagą inercji jest bardzo dużo sposobów na określenie wartości współczynnika inercji w poszczególnych iteracjach. W niniejszej pracy zastosowano tylko jeden sposób. Pozycja [14] prezentuje 15 różnych wzorów na określanie współczynnika inercji. Analiza wynikow dla poszczególnych zależnosci mogłaby przyczynić się do poprawy zbieżności wyniku dla analizowanych tu konstrukcji prętowych. Nie ma zatem jednego, najlepszego zestawu parametrów algorytmu PSO, uniwersalnego dla każdego zagadnienia optymalizacji.

Zastosowany w pracy algorytm PSO oferuje szereg możliwości rozbudowy. Algorytm PSO może zostać rozszerzony o dodatkowe funkcje poprawiające zbieżność wyników. Jeśli chodzi natomiast o rozbudowę modułu wykonującego

obliczenia z zakresu statyki konstrukcji prętowych, pojawia się wiele opcji. Pierwszą może być rozszerzenie analizy MES do przypadku obliczeń konstrukcji trójwymiarowych. Kolejnym krokiem może być uzupełnienie obliczeń o analizę stateczności, to jest analizę z uwzględnieniem warunków wyboczenia prętów. Użyteczną funkcją byłaby również możliwość generowania wynikowej (lub pośrednich) konstrukcji i w zależności od występujących w pręcie naprężeń, dołączenie procedur graficznych umozliwiajacych kolorowanie elementów prętowych według przyjętej skali. Udogodnieniem mógłby być też interaktywny interfejs ułatwiający przygotowywanie danych do obliczeń.

Jak pokazały obliczenia, szczególnie analiza bardziej skomplikowanych przypadków (most, konstrukcja łukowa) przy zwiększającej się ilości węzłów i elementów skończonych znacznie wydłuża się czas obliczeń (szczególnie przy pracy na zmiennych dyskretnych). Z kolei chęć uzyskania zadowalających wyników wymusza stosowanie dużej liczby cząstek oraz liczby iteracji. Niestety Scilab dość wolno wykonuje zaprogramowane obliczenia, co w połączeniu z koniecznością wielokrotnego uruchamiania procedury optymalizacji wiąże się z długim czasem oczekiwania na wyniki. Można więc pokusić się o przeniesienie całego programu do innego środowiska programowania, które posiada możliwość kompilacji kodu i stworzenie programu wykonywalnego typu exe, który w znacznym stopniu przyspieszy obliczenia.

Bibliografia

[1] Rusiński E., Czmochowski J., Smolnicki T., *Zaawansowana metoda elementów skończonych w konstrukcjach nośnych*, Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław, 2000

[2] Rusiński E., *Zasady projektowania konstrukcji nośnych pojazdów samochodowych*, Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław, 2002

[3] Pyrz M., *Wprowadzenie do Metody Elementów Skończonych*, PW – SiMR – IP, 2014 (materiały z wykładu)

[4] Lazinica A., *Particle Swarm Optimization*, In-tech, Croatia, 2009

[5] Uniwersytet Śląski w Katowicach, Wydział Informatyki i Nauki o Materiałach, Instytut Informatyki, *Skrypt do przedmiotu Algorytmy ewolucyjne*, Sosnowiec, 2009

[6] Guan-Chun Luh , Chun-Yi Lin, 2011. Optimal design of truss-structures using particle swarm optimization, *Computers and Structures,* 89, 2221–2232.

[7] Kusiak J., Danielewska-Tułecka A., Oprocha P., *Optymalizacja – Wybrane metody z przykładami zastosowań*, Wydawnictwo Naukowe PWN, Warszawa, 2009

[8] Pyrz M., *Algorytmy ewolucyjne w optymalnym projektowaniu konstrukcji i identyfikacji parametrów materiałowych*, Wydawnictwo Naukowe Instytutu Technologii Eksploatacji – PIB, Warszawa, 2011

[9] Ptaszny J., *Metody optymalizacji. Wykład I. Wiadomości podstawowe*, Katedra Wytrzymałości Materiałów i Metod Komputerowych Mechaniki, Gliwice, 2009

[10] Gomes, H.M., 2011. Truss optimization with dynamic constraints using a particle swart algorithm, *Expert Systems with Applications*, 38, 957-968.

[11] Kaveh A., Talatahari S., 2009a. Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures, *Computer and Structures*, 87, 267-283.

[12] Kaveh A., Talatahari S., 2009b. A particie swarm ant colony optimization for truss structures with discrete variables, *Journal of Constructional Steel Research*, 65, 1558-1568.

[13] Narayana Naik, G., Omkar, S.N., Mudigere, D., Gopalakrishnan, S., 2011. Nature inspired optimization techniques for the design optimization of laminated composite structures using failure criteria, *Expert Systems with Applications*, 38,3,2489-2499.

[14] J. C. Bansal, P. K. Singh, Mukesh Saraswat, Abhishek Verma, Shimpi Singh Jadon, Ajith Abraham, Inertia *Weight Strategies in Particle Swarm Optimization*, 1ABV-Indian Institute of Information Technology & Management, Gwalior, India; Machine Intelligence Research Labs (MIR Labs), USA; VSB Technical University of Ostrava, Czech Republic

[15] Saeed Gholizadeh, 2013. Layout optimization of truss structures by hybridizing cellular automata and particle swarm optimization, *Computers and Structures*, 125, 86–99.

[16] L.J. Li, Z.B. Huang, F. Liu, 2009. A heuristic particle swarm optimization method for truss structures with discrete variables, *Computers and Structures*, 87, 435–443.

[17] Foryś P., *Nowy algorytm optymalizacji rojem cząstek i jego zastosowanie w kształtowaniu elementów konstrukcji – praca doktorska*, Instytut Mechaniki Stosowanej Politechniki Krakowskiej, Kraków, 2007

[i1] http://zsi.ii.us.edu.pl/~mboryczka/IntStad/is_zastosowanie.php

[i2] http://zsi.ii.us.edu.pl/~mboryczka/IntStad/pso_inspiracje.php

[i3] http://okbridges.wkinsler.com/builders/oklahoma.html

[i4] http://www.naxpro-truss.co.uk/trusses/stages/stages-with-gabled-roof/

[i5] http://en.wikipedia.org/wiki/Rastrigin_function

Załączniki

Załącznik nr 1

Kod w środowisku Scilab - sposób definicji zmiennych – parametrów konstrukcji, na podstawie których program MES dokonuje obliczeń.

mes_parametry_przyklad.sce

```
// Metoda elementow skonczonych - (elementy pretowe)

//Definicja parametrow konstrukcji

//-----Przyklad-----

    //dane geometryczne konstrukcji
    nw=12;    //liczba wezlow
    ne=21;   //liczba elementow skonczonych

    //definiowanie wspolrzednych wezlow
    nc=[0,0;
        300,200;
        500,200;
        700,200;
        900,200;
        1100,200;
        1400,0;
        1100,400;
        900,400;
        700,400;
        500,400;
        300,400];      //macierz wspolrzednych poszczegolnych
wezlow (nw x 2) [x1, y1; x2, y2; ...]


    //definicja polaczen wezlow - elementow skonczonych oraz
ich parametrow
    mp=[1,2;
        2,3;
        3,4;
        4,5;
        5,6;
        6,7;
        7,8;
        6,8;
```

```
        5,8;
        5,9;
        5,10;
        4,10;
        3,10;
        3,11;
        3,12;
        2,12;
        1,12;
        11,12;
        11,10;
        10,9;
        9,8];      //macierz polaczen (wymiar - ne x 2)

    //dane dotyczace przekrojow
    //zmienne ciagle
    Amin=0.1;   //minimalna dopuszalna wartosc pola przekroju
preta
    Amax=500;   //maksymalna dopuszczalna wartosc pola
przekroju preta

    //zmienne dyskretne

katalogA=[1,2,3,4,5,6,7,8,9,11,15,22,34,45,57,71,92,113,150,22
0];    //dopuszczalne wartosci pol przekrojow preta


    //dane materialowe
    E=0.1*10^5;   //modul Younga [MPa]
    ro=7850*10^-9;   //gestosc [kg/mm^3]

    //definicja obciazenia - sily dzialajace w wezlach (poza
reakcjami) [N]
    FP=[-10000,8;10000,15];   //pary liczb: 'wartosc sily -
kierunek przemieszczenia , w ktorym dziala (numeracja
przemieszczen jak w wektorze U)'

    //definicja warunkow brzegowych
    wb=[14,13,2,1];   //w wektorze zapisane sa nr
przemieszczen, ktore przyjmuja wartosc "0" (numeracja musi byc
zgodna z definicja wektora U),
                  //wazne zeby byly zapisane w kolejnosci
malejacej

    //parametry wykorzystywane do obliczenia funkcji celu
    sigmadop=300;    //naprezenia dopuszczalne [MPa]
```

```
    kara=10;   //wspolczynnik kary (przekroczenie naprezen
dopuszczalnych)
    kara2=-1000;   //wspolczynnik kary (eliminacja wartosci
ujemnych)
```

Załącznik nr 2

Plik z kodem (w środowisku Scilab) modułu MES.

mes_modul_funkcja.sci

```
// Metoda elementow skonczonych - (elementy pretowe)

function [fcelu, U, M, sigma, F, FX, FY, T, TX,
TY]=mes_module(A, nw, ne, nc, mp, E, ro, FP, wb, sigmadop,
kara, kara2, u_dop)

    //Obliczenie dlugosci oraz katow nachylenia poszczegolnych
elementow w globalnym ukladzie wspolrzednych

    L=zeros([1:ne]); //wektor dlugosci elementow [L1, L2, ...
, Lne]
    alfa=zeros([1:ne]); //wektor katow nachylenia elementow w
globalnym ukladzie wspolrzednych [rad]

    for i=1:ne,
            //dlugosc elementu
                //wspolrzedne wektora i-tego elementu [x,y]
                wspx=nc(mp(i,2),1)-nc(mp(i,1),1);
//wspolrzedna x
                wspy=nc(mp(i,2),2)-nc(mp(i,1),2);
//wspolrzedna y
            L(i)=sqrt((wspx^2)+(wspy^2)); //dlugosc i-tego
elementu

            //kat nachylenia
            if wspy>=0 then
                alfa(i)=acos(wspx/L(i));
            else
                alfa(i)=2*%pi-acos(wspx/L(i));
            end
    end


    //macierze sztywnoosci elementu w globalnym ukladzie
wspolrzednych

    KG=zeros(2*nw,2*nw);

    for i=1:ne,
        kat=alfa(i);

        K=[cos(kat)^2, cos(kat)*sin(kat), -(cos(kat)^2), -
(cos(kat)*sin(kat));
```

```
        cos(kat)*sin(kat), sin(kat)^2, -(cos(kat)*sin(kat)), -
(sin(kat)^2);
        -(cos(kat)^2), -(cos(kat)*sin(kat)), cos(kat)^2,
cos(kat)*sin(kat);
        -(cos(kat)*sin(kat)), -(sin(kat)^2),
cos(kat)*sin(kat), sin(kat)^2];

        KE=(E*A(i)/L(i))*K;

        //Zapisanie wartosci w globalnej macierzy sztywnosci
ukladu

        d1=(mp(i,1)-1)*2; //przyrost dla wspolrzednych 1.
wezla
        d2=(mp(i,2)-2)*2; //przyrost dla wspolrzednych 2.
wezla

        KP=zeros(2*nw,2*nw); //pomocnicza macierz sztywnosci

        KP(1+d1,1+d1)=KE(1,1);
        KP(1+d1,2+d1)=KE(1,2);
        KP(1+d1,3+d2)=KE(1,3);
        KP(1+d1,4+d2)=KE(1,4);

        KP(2+d1,1+d1)=KE(2,1);
        KP(2+d1,2+d1)=KE(2,2);
        KP(2+d1,3+d2)=KE(2,3);
        KP(2+d1,4+d2)=KE(2,4);

        KP(3+d2,1+d1)=KE(3,1);
        KP(3+d2,2+d1)=KE(3,2);
        KP(3+d2,3+d2)=KE(3,3);
        KP(3+d2,4+d2)=KE(3,4);

        KP(4+d2,1+d1)=KE(4,1);
        KP(4+d2,2+d1)=KE(4,2);
        KP(4+d2,3+d2)=KE(4,3);
        KP(4+d2,4+d2)=KE(4,4);


        KG=KG+KP; //globalna macierz sztywnosci


    end


        U=zeros([1:2*nw])'; //wektor kolumnowy przemieszczen w
poszczegolnych wezlach [u1x, u1y, u2x, u2y ...]'
```

```
        F=zeros([1:2*nw])'; //wektor kolumnowy sił w
poszczególnych wezlach

        [lw, lc]=size(FP); //lw - liczba wierszy, lc - liczba
kolumn

        //wprowadzenie wartosci sil do wektora F
        for i=1:lw,
            F(FP(i,2))=FP(i,1);
        end


        //zdefiniowanie warunkow brzegowych
        [lw, lc]=size(wb); //lw - liczba wierszy, lc - liczba
kolumn

        KP=KG; //pomocnicza maciez KP, zeby nie stracic danych
przy usuwaniu wierszy z macierzy KG
        FP=F; //pomocniczy wektor FP, zeby nie stracic danych
przy usuwaniu elementow z wektora F
        poz=(1:2*nw); //pomocniczy wektor - pozycje
sil/przemieszczen - zeby bylo wiadomo w ktorych wezlach
wyznaczono przemieszczenia (w pelnym ukladzie rownan)

        for i=1:lc,     //kasowanie wierszy i kolumn tam,
gdzie przemieszczenia sa rowne "0"

            KP(wb(i),:)=[];     //usuwanie kolumn
            KP(:,wb(i))=[];     //usuwanie wierszy
            FP(wb(i),:)=[];
            poz(:,wb(i))=[];

        end

    //Rozwiazanie ukladu rownan


        //K*U=F --. U=K/F
        UP=KP\FP;


        [lw, lc]=size(UP); //lw - liczba wierszy, lc - liczba
kolumn

        for i=1:lw,
            U(poz(i),1)=UP(i);  //aktualizacja wektora
przemieszczen
        end

    //Reakcje
```

```
        KP=KG; //pomocnicza maciez KP, zeby nie stracic danych
przy usuwaniu wierszy z macierzy KG
        [lw, lc]=size(wb); //lw - liczba wierszy, lc - liczba
kolumn

        for i=1:lc,     //kasowanie  kolumn tam, gdzie
przemieszczenia sa rowne "0"

            KP(:,wb(i))=[];     //usuwanie kolumn

        end

        poz=gsort(poz); //sortowanie pozycji malejaco (zeby
poprawnie dzialalo usuwanie wierszy)
        [lw, lc]=size(poz); //lw - liczba wierszy, lc - liczba
kolumn

         for i=1:lc,        //kasowanie wierszy (wszystkie
poza tymi, gdzie sa reakcje)

             KP(poz(i),:)=[];

         end

        //wektor reakcji
        R=KP*UP;

        //wstawienie obliczonych wartosci reakcji do wektora
sil "F"

        wb=gsort(wb,'g','i'); //sortowanie pozycji rosnaco
(zeby poprawnie wstawic wartosci reakcji do wektora F)
        [lw, lc]=size(wb); //lw - liczba wierszy, lc - liczba
kolumn

            for i=1:lc,
                F(wb(i),1)=R(i);        //aktualizacja wektora
sil (wartosci reakcji)
            end

        //Suma rzutow sil na osie OX i OY - sprawdzenie
            //Sily w kierunku X - nr pozycji nieparzysty w
wektorze sil F
            //Sily w kierunku Y - nr pozycji parzysty w
wektorze sil F
            [lw, lc]=size(U); //lw - liczba wierszy, lc -
liczba kolumn

            FX=0;   //Suma rzutow sil na os OX
            FY=0;   //Suma rzutow sil na os OY
```

```
            for i=1:lw,

                if modulo(i,2)==0 then
                    FY=FY+F(i);
                else
                    FX=FX+F(i);
                end

            end

        //Suma momentow wzgledem poczatku ukladu wspolrzednych
- sprawdzenie

            //Moment od sil dzialajacych w kierunku OX
                wx=1; //pomocnicza zmienna sterujaca
                TX=0; //moment sil dzialajacych w kierunku OX

                for z=1:2:(2*nw)
                    TX=TX+F(z)*nc(wx,2);
                    wx=wx+1;
                end

            //Moment od sil dzialajacych w kierunku OY
                wy=1; //pomocnicza zmienna sterujaca
                TY=0; //moment sil dzialajacych w kierunku OY

                for z=2:2:(2*nw)
                    TY=TY-(F(z)*nc(wy,1)); //znak "-" bo dla
Y>0 M<0
                    wy=wy+1;
                end

             T=TX+TY; //suma moementow wzgledem poczatku
ukladu wspolrzednych



        //wyswietlenie obliczonych wartosci      //---
dezaktywowane
        //disp("Maciez sztywnosci K: ")
        //disp(KG)
        //disp("***")

        //disp("Wektor przemieszczen U: ")
        //disp(U)
        //disp("***")

        //disp("Wektor sil F: ")
        //disp(F)
        //disp("***")
```

```
        //disp("Moment T wzgledem poczatku ukladu
wspolrzednych: ")
        //disp(T)
        //disp("***")

        //disp("Suma rzutow sil na os OX: ")
        //disp(FX)
        //disp("***")

        //disp("Suma rzutow sil na os OY: ")
        //disp(FY)
        //disp("***")


    //Naprezenia w poszczegolnych elementach

    for i=1:ne,
         US=[U(2*(mp(i,1))-1);       //wektor przemieszczen w
wezlach poszczegolnego elementu
             U(2*(mp(i,1)));
             U(2*(mp(i,2))-1);
             U(2*(mp(i,2)))]

         sigma_e=(E/L(i))*[-cos(alfa(i)), -sin(alfa(i)),
cos(alfa(i)), sin(alfa(i))]*US; //naprezenie w i-tym elemencie

         sigma(i)=sigma_e; //wektor zawierajacy wartosci
naprezen w elementach

    end


        //disp("Naprezenia: ") //---dezaktywowane
        //disp(sigma)
        //disp("***")


    //Masa konstrukcji

    M=0; //masa calej konstrukcji
    for i=1:ne,
        me=L(i)*A(i)*ro;    //masa i-tego elementu
        M=M+me;
    end

        //disp("Masa konstrukcji: ") //---dezaktywowane
        //disp(M)
        //disp("***")
```

```
//-------------------------------------------------------------
-
//-------------funkcja celu - minimalna masa
//-------------------------------------------------------------
-
//funkcja kary - przekroczenie naprezen dopuszczalnych
        g1=0;    //funkcja kary

        for i=1:ne,
            ge=max([0,(abs(sigma(i))-sigmadop)/sigmadop]);
//funkcja kary dla i-tego elementu
            //ge=max([0,(abs(sigma(i))-sigmadop)]);
            g1=g1+ge;
        end

//funkcja kary - przekroczenie przemieszczen dopuszczalnych
        g2=0;    //funkcja kary

        for i=1:2*nw,

            ge=max([0,(abs(U(i))-u_dop)/u_dop]); //funkcja
kary dla i-tego elementu

            g2=g2+ge;
        end


//maksymalna mozliwa masa konstrukcji
        M_max=0; //maksymalna masa

        for i=1:ne,
            M_max_p=L(i)*Amax*ro;
            M_max=M_max+M_max_p; //maksymalna masa i-tego
elementu
        end

//funkcja celu

        fcelu=(M/M_max)+(kara*g1)+(kara2*g2);


    endfunction
```

Załącznik nr 3

Plik z kodem (w środowisku Scilab) – definicja zmiennych – trójkąt (przykład testowy).

mes_parametry_trojkat.sce

```
// Metoda elementow skonczonych - (elementy pretowe)

//Definicja parametrow konstrukcji

//-----Przyklad 1 - 'Trojkat'-----

    //dane geometryczne konstrukcji
    nw=3;    //liczba wezlow
    ne=3;   //liczba elementow skonczonych

    //definiowanie wspolrzednych wezlow
    nc=[0,0;1000,0;0,1000];   //macierz wspolrzednych
poszczegolnych wezlow (nw x 2) [x1, y1; x2, y2; ...]

    //definicja polaczen wezlow - elementow skonczonych oraz
ich parametrow
    mp=[1,2;2,3;1,3];   //macierz polaczen (wymiar - ne x 2)

    //dane materialowe
    E=0.1*10^5;   //modul Younga
    ro=7850*10^-9;   //gestosc

    //definicja obciazenia - sily dzialajace w wezlach (poza
reakcjami)
    FP=[-7070,5; -7070,6];   //pary liczb: 'wartosc sily -
kierunek przemieszczenia , w ktorym dziala (numeracja
przemieszczen jak w wektorze U)'

    //definicja warunkow brzegowych
    wb=[4,2,1];   //w wektorze zapisane sa nr przemieszczen,
ktore przyjmuja wartosc "0" (numeracja musi byc zgodna z
definicja wektora U),
                  //wazne zeby byly zapisane w kolejnosci
malejacej

    //parametry wykorzystywane do obliczenia funkcji celu
    sigmadop=500;    //naprezenia dopuszczalne
    kara=10;   //wspolczynnik kary (przekroczenie naprezen
dopuszczalnych)
    kara2=-1000;   //wspolczynnik kary (eliminacja wartosci
ujemnych)
```

Załącznik nr 4

Skrypt testujący działanie modułu MES.

test_modulu_mes.sce

```
//Skrypt testujacy dzialanie modulu MES

clear

exec('mes_modul_funkcja.sci');

exec('mes_parametry_trojkat.sce');   //wczytanie danych


wsp=150; //wspolczynnik uzywany przy generowaniu losowych
przekrojow


    A=wsp*rand(1,ne); //generowanie losowych przekrojow
elementow skonczonych
    [fcelu, U,
M]=mes_module(A,nw,ne,nc,mp,E,ro,FP,wb,sigmadop,kara,kara2);


    disp('***')
    disp('Masa konstrukcji : ')
    disp(M)
    disp('***')
```

Załącznik nr 5

Plik z kodem (w środowisku Scilab) algorytmu PSO.

PSO_inetrial.sci

```
//Algorytm PSO z waga inercji

    mode(-1) //usuniecie echa

//jawne deklaracje wektorow i macierzy

clear //dezaktywowac w przypadku wywolywania PSO w innym
skrypcie
stacksize('max')

//wczytanie parametrow konstrukcji - potrzebne modulowi MES do
obliczenia funkcji celu
exec('mes_parametry_bridge.sce');

//wczytanie modulu MES
exec('mes_modul_funkcja.sci');

//--------------------------------
//Okreslenie trybu optymalizacji
//---do wyboru
//------zmienne ciagle (zmienne przyjmuja wartosc miedzy
zadeklarowanym minimum i maksimum)
//------zmienne dyskretne (zmienne przyjmuja tylko wybrane
wartosci zadeklarowane w wektorze - katalogu zmiennych)
    tryb=0; //zmienne ciagle
    //tryb=1; //zmienne dyskretne

//Rysunek konstrukcji przed obciazeniem
clf();
        xset("window",0)
        for i=1:ne,

            x=[nc(mp(i,1),1),nc(mp(i,2),1)];
            y=[nc(mp(i,1),2),nc(mp(i,2),2)];
            plot2d(x,y,style=3,axesflag=0, frameflag=1,
rect=[-200,-200,max(nc)+200,max(nc)+200]);
            plot2d(x,y,style=-4, frameflag=1, rect=[-200,-
200,max(nc)+200,max(nc)+200]);
            xtitle("Schemat konstrukcji:")
```

```
        end


lines(0)  // disables vertical paging

//parametry PSO
    wmax=0.9; // maksymalna (poczatkowa) wartość współczynnika
wagi inercji
    wmin=0.4; // minimalna (koncowa) wartość współczynnika
wagi inercji
    itmax=10; // maksymalna liczba iteracji
    c1=1.3; // ustalony mnoznik wagowy (przy czlonie z
najlepszym dotychczas znalezionym polozeniem czastki)
    c2=2.8; // ustalony mnoznik wagowy (przy czlonie z
najlepszym dotychczas znalezionym polozeniem lidera roju)
    N=10; // liczba czastek
    D=ne; // wymiar problemu D zdefinowany ponizej (musi byc
zgodny z definicja optymalizowanej funkcji)
    x_sup=ones([1:D])
    x_inf=ones([1:D])
    x_sup=Amin*x_sup'; // minimalna wartosc polozenia (uzywana
do generowania poczatkowej populacji czastek)
    x_inf=Amax*x_inf'; // maksymalna wartosc polozenia
(uzywana do generowania poczatkowej populacji czastek)
    v_max=ones([1:D])
    v_max=500*v_max'; // maksymalna wartosc predkosci (uzywana
do generowania poczatkowych predkosci czastek)
    v_min=-1*v_max; // minimalna wartosc predkosci (uzywana do
generowania poczatkowych predkosci czastek)

//deklaracje jawne zmiennych
    W=zeros([1:itmax])  // wektor wspolczynnikow wag
    F0=zeros([1:N])   // pierwsze obliczone wartosci funkcji
celu
    x1=zeros(N,D,itmax)   //  obliczone wektory pozycji
(ostatni wymiar to nr iteracji)
    v1=zeros(N,D,itmax)   // obliczone wektory predkosci
(ostatni wymiar to nr iteracji)
    F1=zeros(N,itmax)   // obliczone wartosci funkcji celu
(ostatni wymiar to nr iteracji)

    gbest1=zeros(D,itmax)  // pomocnicza do wartosci x
```

```
    G1=zeros(N,D,itmax)    // najlepsze globalne (wartosci x)
    Fbest1=zeros([1:itmax])    // najlepsze globalne (wartosci
funckji)
    pbest1=zeros(N,D,itmax)    // najlepsze rozwiazanie
pojedynczego osobnika (wartosci x)
    Fb1=zeros([1:itmax])    // najlepsze rozwiazanie w danej
iteracji (wartosc funkcji)

//  obliczenie wartosci wektora wspolczynnikow wagi inercji
dla wszystkich iteracji (shi & eberhart 1998)

    for j=1:itmax
        W(j)=wmax-((wmax-wmin)/itmax)*j;
    end

//obliczanie polozenia i predkosci wszystkich czastek
    //losowe wygenerowanie pozycji i predkosci wewnatrz
zadanych granic
    // aktualny numer iteracji
        j=1;

    for i=1:D

        if tryb==0 then //dla zmiennych ciaglych

            x1(1:N,i,j) =x_inf(i) +rand(N,1) * ( x_sup(i) -
x_inf(i) ); // pierwszy wektor pozycji
            v1(1:N,i,j)=v_min(i)+(v_max(i)-
v_min(i))*rand(N,1); // pierwszy wektor predkosci

        else //dla zmiennych dyskretnych

            v1(1:N,i,j)=v_min(i)+(v_max(i)-
v_min(i))*rand(N,1); // pierwszy wektor predkosci

            dimA=size(katalogA);
            dimA=dimA(1,2); //zapisanie wartosci o ilosci
dostepnych przekrojow znajdujacysch sie w wektorze katalogA

            k=rand()*dimA;  //losowy wybor przekroju
            k=round(k);     //losowy wybor przekroju -
zaokraglenie do liczny calkowitej (pozycja z katalogA)
```

```
            if k==0 then //zabezpieczenie przed k=0 po
zaokragleniu
                k=1;
            end

            x1(1:N,i,j)=katalogA(1,k); // pierwszy wektor
pozycji //zapisanie przekroju elementu w wektorze pol
przekroju A z katalogu

        end

    end

//pierwsze obliczenie wartosci funkcji celu

    for i=1:N
        y1=x1(i,1:D,j)
        [fcelu, U, M, sigma, F, FX, FY, T, TX,
TY]=mes_module(y1,nw,ne,nc,mp,E,ro,FP,wb,sigmadop,kara,kara2,u
_dop);
        F0(i)=fcelu //F i-tej czastki
        F1(i,j)=fcelu //F i-tej czastki w j-tej iteracji
    end

//poszukiwanie minimum wsrod czastek roju

    [fmin, pos]=min(F1(1:N,j)) ; //  pos - pozycja, gdzie
znaleziono minimum

//pierwsze minimum jest tez globalnym minimum ponieważ jest to
pierwsza iteracja
    gbest1(1:D,j)=x1(pos,1:D,j)';  // moja wersja *

    for i=1:N
        G1(i,1:D,j)=gbest1(1:D,j); // macierz z wektorami
rozwiazan (x)
    end

//pierwsze minimum jest najlepszym wynikiem poniewaz jest to
pierwsza iteracja
    Fbest1(j)= fmin;     // najlepsze rozwiazanie globalne
    Fb1(j)=fmin; // najlepsze w iteracji nr j
```

```
//kazde rozwiazanie - pozycja czastki jest jej najlepszym
rozwiazaniem poniewaz jest to pierwsza iteracja

    for i=1:N
        pbest1(i,1:D,j)=x1(i,1:D,j)    // personal best dla
pierwszej iteracji (tj pierwsze x)
    end

//predkosci i polozenie dla nastepnych iteracji

        if tryb==0 then //dla zmiennych ciaglych

v1(1:N,1:D,j+1)=W(j)*v1(1:N,1:D,j)+c1*rand()*(pbest1(1:N,1:D,j
)-x1(1:N,1:D,j))+c2*rand()*(G1(1:N,1:D,j)-x1(1:N,1:D,j));
//nowa predkosc
            x1(1:N,1:D,j+1)=x1(1:N,1:D,j)+v1(1:N,1:D,j+1); //
nowe polozenie

            for q=1:N //uwzglednienie zdefiniowanego
maksymalnego i minimalnego przekroju
                for r=1:D
                    if x1(q,r,j+1)>Amax then
                        x1(q,r,j+1)=Amax; //jesli obliczone x1
wieksze od Amax, to staje sie Amax

                    elseif x1(q,r,j+1)<Amin then
                        x1(q,r,j+1)=Amin; //jesli obliczone x1
mniejsze od Amin, to staje sie Amin

                    else //jesli nie spelnione poprzednie
warunki to nic nie robimy
                    end
                end
            end

        else //dla zmiennych dyskretnych


v1(1:N,1:D,j+1)=W(j)*v1(1:N,1:D,j)+c1*rand()*(pbest1(1:N,1:D,j
)-x1(1:N,1:D,j))+c2*rand()*(G1(1:N,1:D,j)-x1(1:N,1:D,j));
//nowa predkosc
            x1(1:N,1:D,j+1)=x1(1:N,1:D,j)+v1(1:N,1:D,j+1); //
nowe polozenie
```

```
            dimA=size(katalogA);
            dimA=dimA(1,2);

             for q=1:N //dobor wartosci z katalogu
                 for r=1:D
                                 for t=1:dimA
                                     delta(t)=abs(x1(q,r,j+1)-
katalogA(t));    //obliczenie roznic miedzy kazdym z elementow
a aktualnym x
                                 end

                                 [xmin imin]=min(delta);
//znalezienie min wartosci i jej pozycji imin

                     x1(q,r,j+1)=katalogA(imin); //podstawienie
wartosci z katalogu

                  end
              end
             end


//-----------------------------------------------------
//Rozpoczecie petli optymalizacyjnej
//-----------------------------------------------------
    while (j<=itmax-1)
        j=j+1

        //Obliczenie funkcji celu
            for i=1:N
                y1=x1(i,1:D,j);
                [fcelu, U, M, sigma, F, FX, FY,
T]=mes_module(y1,nw,ne,nc,mp,E,ro,FP,wb,sigmadop,kara,kara2,u_
dop);
                F1(i,j)=fcelu //F i-tej czastki w j-tej
iteracji
            end

        //Poszukiwanie minimum wsrod roju czastek
            [fmin, pos]=min(F1(1:N,j)) ; //  pos - to pozycja
gdzie minimum znalezione


        //Poszukiwanie globalnego minimum
```

```
        //zakladamy ze mamy lepszy wynik i go zapisujemy *
            gbest1(1:D,j)=x1(pos,1:D,j)'; // w pomocniczej
global best (wartosci x) *
            Fb1(j)=F1(pos,j); // albo =fmin poszukiwanie
najlepszego rozwiazania *
            Fbest1(j)=Fb1(j); // Fbest1 to najlepszy znany  *


        //poprawiane zmienne jesli zalozenie nieprawdziwe

            if Fbest1(j)<Fbest1(j-1) //zalozenie spelnione -
nic nie zmianiamy
            else  // w przeciwnym wypadku
                gbest1(1:D,j)=gbest1(1:D,j-1); // zmiana
wartości współrzędnych (najlepsze globalne rozwiazanie)
                Fbest1(j)=Fbest1(j-1); // Fbest1 nie zostalo
tym razem znalezione
            end

        //MP moja wersja *
            for i=1:N
                G1(i,1:D,j)=gbest1(1:D,j); //macierz z
wektorami rozwiazan x
            end


        //Obliczanie najlepszego rozwiazania dla danej czastki
(personal best)
             for i=1:N

                    if F1(i,j)<f_pbest(i,j-1)
                        pbest1(i,1:D,j)=x1(i,1:D,j);
                        f_pbest(i,j)=F1(i,j);
                    else
                        pbest1(i,1:D,j)=pbest1(i,1:D,j-1);
                        f_pbest(i,j)=f_pbest(i,j-1);
                    end

             end


        //Ponowne obliczenie Fbest1 (wartosc funkcji celu w
global best)
            y1=gbest1(1:D,j);
```

```
            [fcelu, U, M, sigma, F, FX, FY,
T]=mes_module(y1,nw,ne,nc,mp,E,ro,FP,wb,sigmadop,kara,kara2,u_
dop);
            Fbest1(j)=fcelu //funkcja celu i-tej czastki w j-
tej iteracji


        //Obliczenie predkosci i polozenia dla nastepnych
iteracji

            if tryb==0 then //dla zmiennych ciaglych

v1(1:N,1:D,j+1)=W(j)*v1(1:N,1:D,j)+c1*rand()*(pbest1(1:N,1:D,j
)-x1(1:N,1:D,j))+c2*rand()*(G1(1:N,1:D,j)-x1(1:N,1:D,j));
//nowa predkosc

x1(1:N,1:D,j+1)=x1(1:N,1:D,j)+v1(1:N,1:D,j+1); //nowe
polozenie

                        for q=1:N //uwzglednienie
zdefiniowanego maksymalnego i minimalnego przekroju
                            for r=1:D
                                if x1(q,r,j+1)>Amax then
                                    x1(q,r,j+1)=Amax; //jesli
obliczone x1 wieksze od Amax, to staje sie Amax

                                elseif x1(q,r,j+1)<Amin then
                                    x1(q,r,j+1)=Amin; //jesli
obliczone x1 mniejsze od Amin, to staje sie Amin

                                else //jesli nie spelnione
poprzednie warunki to nic nie robimy
                                end
                            end
                        end

            else //dla zmiennych dyskretnych


v1(1:N,1:D,j+1)=W(j)*v1(1:N,1:D,j)+c1*rand()*(pbest1(1:N,1:D,j
)-x1(1:N,1:D,j))+c2*rand()*(G1(1:N,1:D,j)-x1(1:N,1:D,j));
//nowa predkosc
            x1(1:N,1:D,j+1)=x1(1:N,1:D,j)+v1(1:N,1:D,j+1); //
nowe polozenie
```

```
dimA=size(katalogA);
dimA=dimA(1,2);

 for q=1:N //dobor wartosci z katalogu
     for r=1:D
 f=1;
 if x1(q,r,j+1)>katalogA(1)
     if x1(q,r,j+1)>katalogA(dimA)

         x1(q,r,j+1)=katalogA(dimA);
     else
         while katalogA(f)<x1(q,r,j+1)
             f=f+1;
         end


         a1=abs(x1(q,r,j+1)-katalogA(f));
         a2=abs(x1(q,r,j+1)-katalogA(f-1));

             if a1>a2
                 x1(q,r,j+1)=katalogA(f-1);
 //podstawienie wartosci z katalogu
             else
                 x1(q,r,j+1)=katalogA(f);
 //podstawienie wartosci z katalogu
             end
      end

 else
     x1(q,r,j+1)=katalogA(f); //podstawienie
 wartosci z katalogu
 end

       end
   end

 end


//wyswietlanie numeru iteracji
disp('Iteracja numer    :  ' + string(j))
disp('***')


end   //Koniec pentli optymalizacji

//Rysynek konstrukcji po odkszztalceniu
```

```
        ncp=nc; //macierz wspolrzednych wezlow po obciazeniu
konstrukcji
        UX=U; //wektor przemieszczen w kierunku X
        UY=U; //wektor przemieszczen w kierunku Y
        [lw, lc]=size(U); //lw - liczba wierszy, lc - liczba
kolumn

        for i=lw:-1:1,

            if modulo(i,2)==0 then

                UX(i)=[];
            else

                UY(i)=[];
            end

        end

        ncp=nc+[UX,UY]; //aktualizacja wspolrzednych

        xset("window",0)
        for i=1:ne,

            x=[ncp(mp(i,1),1),ncp(mp(i,2),1)];
            y=[ncp(mp(i,1),2),ncp(mp(i,2),2)];
            plot2d(x,y,style=26,axesflag=0, frameflag=1,
rect=[-200,-200,max(nc)+200,max(nc)+200]);
            plot2d(x,y,style=-3, frameflag=1, rect=[-200,-
200,max(nc)+200,max(nc)+200]);
            xtitle("Schemat konstrukcji:")

        end

disp('********************************')
disp('Fbest1    :   ' + string(Fbest1(:,j)))
disp('Gbest1   :   ' + string(gbest1(:,j)))
disp('********************************')

//wykres funkcji celu


            x=[1:itmax];
            y=Fbest1;
```

```
            xset("window",1)
            clf()
            plot2d(x,y, style=5);
            xtitle("Funkcja celu:")


//---wyswietlenie ostatecznych wynikow
    fcelu=0;
    U=0;
    M=0;
    sigma=0;
    F=0;
    FX=0;
    FY=0;
    T=0;

    [fcelu, U, M, sigma, F, FX, FY,
T]=mes_module(gbest1(:,j),nw,ne,nc,mp,E,ro,FP,wb,sigmadop,kara
,kara2,u_dop);

    disp('*******************************')
    disp('Masa konstrukcji    :  ')
    disp(M)
    disp('Przemieszczenia w wezlach    :  ')
    disp(U)
    disp('Naprezenia w elementach    :  ')
    disp(sigma)
    disp('Sily w wezlach    :  ')
    disp(F)
    disp('Suma rzutow sil w kierunku X    :  ')
    disp(FX)
    disp('Suma rzutow sil w kierunku Y    :  ')
    disp(FY)
    disp('Suma momentow    :  ')
    disp(T)
    disp('Funkcja celu    :  ')
    disp(fcelu)
    disp('*******************************')

//-------------------------
//informacja o przekroczonych naprezeniach dopuszczalnych
    for i=1:ne
        if abs(sigma(i))>sigmadop then
            h1(i)=1;
```

```
        else
            h1(i)=0;
        end
    end

    if max(h1)==1 then
        disp('********************************')
        disp('Uwaga! Naprezenia dopuszczalne przekroczone!')
        disp('********************************')
        else
    end

//--------------------------
//informacja o przekroczonych przemieszczeniach dopuszczalnych
    for i=1:2*nw
        if abs(U(i))>u_dop then
            h2(i)=1;
        else
            h2(i)=0;
        end
    end

    if max(h2)==1 then
        disp('********************************')
        disp('Uwaga! Przemieszczenia dopuszczalne
przekroczone!')
        disp('********************************')
        else
    end


//zapisanie potrzebnych do dalszej analizy wynikow w jednej
macierzy

    wyniki=zeros(200,6);
    wyniki(1:ne,1)=gbest1(:,j); //1. kolumna - przekroje
    wyniki(1:ne,2)=sigma;       //2. kolumna - naprezenia
    wyniki(1:2*nw,3)=U;         //3. kolumna - przemieszczenia
    wyniki(1:itmax,4)=Fbest1';  //4. kolumna - wartosci
funkcji celu w poszczegolnych iteracjach
    wyniki(1,5)=M;              //5. kolumna - masa
konstrukcji
    wyniki(1,6)=czas;           //6. kolumna - czas obliczeń

    //zapisanie do pliku wynikow
        plik=file('open','wyniki_obliczen.txt','unknow');
```

```
            write(plik,'---przekroje---')
            write(plik,wyniki(1:ne,1))
            write(plik,'---naprezenia---')
            write(plik,wyniki(1:ne,2))
            write(plik,'---przemieszczenia---')
            write(plik,wyniki(1:2*nw,3))
            write(plik,'---wartosci funkcji celu w
poszczegolnych iteracjach---')
            write(plik,wyniki(1:itmax,4))
            write(plik,'---masa konstrukcji---')
            write(plik,wyniki(1,5))
            write(plik,'---czas obliczeń---')
            write(plik,wyniki(1,6))
        file('close',plik);
```

Załącznik nr 6

Plik z kodem (w środowisku Scilab) zawierający dane konstrukcji oraz parametry optymalizacji kratownicy płaskiej - przykład z pozycji [8].

mes_parametry_kratownica_mp.sce

```
// Metoda elementow skonczonych - (elementy pretowe)

//Definicja parametrow konstrukcji

//-----Przyklad z pozycji ALGORYTMY EWOLUCYJNE (M. Pyrz)-----

    //dane geometryczne konstrukcji
    nw=6;    //liczba wezlow
    ne=10;   //liczba elementow skonczonych

    //definiowanie wspolrzednych wezlow [in]
    nc=[720,360;
        720,0;
        360,360;
        360,0;
        0,360;
        0,0];      //macierz wspolrzednych poszczegolnych
wezlow (nw x 2) [x1, y1; x2, y2; ...]

    //definicja polaczen wezlow - elementow skonczonych oraz
ich parametrow
    mp=[5,3;
        3,1;
        6,4;
        4,2;
        4,3;
        2,1;
        5,4;
        6,3;
        3,2;
        4,1];      //macierz polaczen (wymiar - ne x 2)

    //dane dotyczace przekrojow [in^2]
        //zmienne ciagle
        Amin=0.1;   //minimalna dopuszalna wartosc pola
przekroju preta
        Amax=36;   //maksymalna dopuszczalna wartosc pola
przekroju preta
```

```
        //zmienne dyskretne

katalogA=[0.1,1,2,3,4,5,6,7,8,9,10,12,14,16,18,20,22,24,26,28,
30,32,34,36];    //dopuszczalne wartosci pol przekrojow preta

    //dane materialowe
    E=10^7;   //modul Younga [lbf/in^2]
    ro=0.1*10^-3;   //gestosc [lb/in^3]

    //definicja obciazenia - sily dzialajace w wezlach (poza
reakcjami) [lbf]
    FP=[-10^5,4; -10^5,8];   //pary liczb: 'wartosc sily -
kierunek przemieszczenia , w ktorym dziala (numeracja
przemieszczen jak w wektorze U)'

    //definicja warunkow brzegowych
    wb=[12,11,10,9];   //w wektorze zapisane sa nr
przemieszczen, ktore przyjmuja wartosc "0" (numeracja musi byc
zgodna z definicja wektora U),
                  //wazne zeby byly zapisane w kolejnosci
malejacej

    //parametry wykorzystywane do obliczenia funkcji celu
    sigmadop=25000;    //naprezenia dopuszczalne [lbf/in^2]
    u_dop=2;    //dopuszczalne przemieszczenia [in]]
    kara=1;   //wspolczynnik kary (przekroczenie naprezen
dopuszczalnych)
    kara2=1;   //wspolczynnik kary (przekroczenie
przemieszczen dopuszczalnych)
```

Załącznik nr 7

Plik z kodem (w środowisku Scilab) zawierający dane konstrukcji oraz parametry optymalizacji modelu mostu (2D) – rozdział 7.2.

mes_parametry_bridge.sce

```
// Metoda elementow skonczonych - (elementy pretowe)

//Definicja parametrow konstrukcji

//------------------Most-------------------------

    //dane geometryczne konstrukcji
    nw=14;    //liczba wezlow
    ne=26;   //liczba elementow skonczonych

    //definiowanie wspolrzednych wezlow [mm]
    nc=[0,0;
        3000,0;
        6000,0;
        9000,0;
        12000,0;
        15000,0;
        18000,0;
        21000,0;
        18000,3500;
        15000,4500;
        12000,5000;
        9000,5000;
        6000,4500;
        3000,3500];      //macierz wspolrzednych
poszczegolnych wezlow (nw x 2) [x1, y1; x2, y2; ...]

    //definicja polaczen wezlow - elementow skonczonych oraz
ich parametrow
    mp=[1,2;
        2,3;
        3,4;
        4,5;
        5,6;
        6,7;
        7,8;
        8,9;
        9,10;
        10,11;
        11,12;
        12,13;
        13,14;
        14,1;
        2,14;
```

```
        14,3;
        3,13;
        3,12;
        4,12;
        12,5;
        4,11;
        5,11;
        11,6;
        6,10;
        6,9;
        9,7];      //macierz polaczen (wymiar - ne x 2)

    //dane dotyczace przekrojow [mm2]
        //zmienne ciagle
        Amin=20;   //minimalna dopuszalna wartosc pola
przekroju preta
        Amax=4000;   //maksymalna dopuszczalna wartosc pola
przekroju preta

        //zmienne dyskretne
            katalogA=[20 120 240 320 440 560 680 760 880 1000
1120 1320 1560 1760 2000 2240 2440 2680 2880 3120 3320 3560
3760 4000]; //dopuszczalne wartosci pol przekrojow preta


    //dane materialowe
    E=210000;   //modul Younga [MPa]
    ro=7.86*10^-6;   //gestosc [kg/mm3]

    //definicja obciazenia - sily dzialajace w wezlach (poza
reakcjami) [N]
    FP=[-115000,4; -115000,6; -115000,8; -115000,10; -
115000,12; -115000,14;];   //pary liczb: 'wartosc sily -
kierunek przemieszczenia , w ktorym dziala (numeracja
przemieszczen jak w wektorze U)'

    //definicja warunkow brzegowych
    wb=[16,2,1];   //w wektorze zapisane sa nr przemieszczen,
ktore przyjmuja wartosc "0" (numeracja musi byc zgodna z
definicja wektora U),
                  //wazne zeby byly zapisane w kolejnosci
malejacej

    //parametry wykorzystywane do obliczenia funkcji celu
    sigmadop=350;    //naprezenia dopuszczalne [MPa]
    u_dop=30;    //dopuszczalne przemieszczenia [mm]
    kara=10;   //wspolczynnik kary (przekroczenie naprezen
dopuszczalnych)
    kara2=10;   //wspolczynnik kary (przekroczenie
przemieszczen dopuszczalnych)
```

Załącznik nr 8

Plik z kodem (w środowisku Scilab) zawierający dane konstrukcji oraz parametry optymalizacji modelu konstrukcji łukowej (2D) – rozdział 7.3.

mes_parametry_truss_arch.sce

```
// Metoda elementow skonczonych - (elementy pretowe)

//Definicja parametrow konstrukcji

//-----Truss arch-----

    //dane geometryczne konstrukcji
    nw=67;    //liczba wezlow
    ne=131;   //liczba elementow skonczonych

    //definiowanie wspolrzednych wezlow [mm]
    nc=[0,0;
        275,822;
        550,1645;
        825,2467;
        1101,3289;
        1300,3900;
        1908,4646;
        2771,5326;
        3685,5937;
        4644,6474;
        5143,6704;
        5642,6934;
        6674,7315;
        7732,7613;
        8810,7828;
        9902,7957;
        11000,8000;
        12098,7957;
        13190,7828;
        14268,7613;
        15326,7315;
        16358,6934;
        16857,6704;
        17356,6474;
        18315,5937;
        19229,5326;
        20092,4646;
        20700,3900;
        20899,3289;
        21175,2467;
        21450,1645;
        21725,822;
        22000,0;
```

```
        22000,975;
        22000,1950;
        22000,2925;
        22000,3899;
        21900,4400;
        21182,5015;
        20734,5397;
        20286,5780;
        19334,6472;
        18329,7087;
        17280,7622;
        16192,8073;
        15072,8437;
        13926,8712;
        13345,8804;
        12763,8896;
        11589,8988;
        10411,8988;
        9237,8896;
        8655,8804;
        8074,8712;
        6928,8437;
        5808,8073;
        4720,7622;
        3671,7087;
        2666,6472;
        1714,5780;
        1266,5397;
        818,5015;
        100,4400;
        0,3899;
        0,2925;
        0,1950;
        0,975];      //macierz wspolrzednych poszczegolnych
wezlow (nw x 2) [x1, y1; x2, y2; ...]

    //definicja polaczen wezlow - elementow skonczonych oraz
ich parametrow
    mp=[1,2;
        2,3;
        3,4;
        4,5;
        5,6;
        6,7;
        7,8;
        8,9;
        9,10;
        10,11;
        11,12;
        12,13;
        13,14;
```

```
14,15;
15,16;
16,17;
17,18;
18,19;
19,20;
20,21;
21,22;
22,23;
23,24;
24,25;
25,26;
26,27;
27,28;
28,29;
29,30;
30,31;
31,32;
32,33;
33,34;
34,35;
35,36;
36,37;
37,38;
38,39;
39,40;
40,41;
41,42;
42,43;
43,44;
44,45;
45,46;
46,47;
47,48;
48,49;
49,50;
50,51;
51,52;
52,53;
53,54;
54,55;
55,56;
56,57;
57,58;
58,59;
59,60;
60,61;
61,62;
62,63;
63,64;
64,65;
```

```
65,66;
66,67;
67,1;
2,67;
67,3;
3,66;
66,4;
4,65;
65,5;
5,64;
64,6;
6,63;
6,62;
62,7;
7,61;
7,60;
60,8;
8,59;
59,9;
9,58;
58,10;
10,57;
57,11;
57,12;
12,56;
56,13;
13,55;
55,14;
14,54;
54,15;
15,53;
15,52;
52,16;
16,51;
51,17;
17,50;
50,18;
18,49;
49,19;
19,48;
19,47;
47,20;
20,46;
46,21;
21,45;
45,22;
22,44;
44,23;
44,24;
24,43;
43,25;
```

```
        25,42;
        42,26;
        26,41;
        41,27;
        27,40;
        27,39;
        39,28;
        28,38;
        28,37;
        37,29;
        29,36;
        36,30;
        30,35;
        35,31;
        31,34;
        34,32];      //macierz polaczen (wymiar - ne x 2)

    //dane dotyczace przekrojow [mm2]
        //zmienne ciagle
        Amin=10;   //minimalna dopuszalna wartosc pola
przekroju preta
        Amax=230;   //maksymalna dopuszczalna wartosc pola
przekroju preta

        //zmienne dyskretne
            katalogA=[10 12 14 18 25 32 39 44 51 58 64 76 90
101 115 129 140 154 166 179 191 205 216 230]; //dopuszczalne
wartosci pol przekrojow preta


    //dane materialowe
    E=70000;   //modul Younga [MPa]
    ro=2.8*10^-6;   //gestosc [kg/mm3]

    //definicja obciazenia - sily dzialajace w wezlach (poza
reakcjami) [N]
    FP=[-3000,22; -3000,30; -2000,38; -2000,46];   //pary
liczb: 'wartosc sily - kierunek przemieszczenia , w ktorym
dziala (numeracja przemieszczen jak w wektorze U)'

    //definicja warunkow brzegowych
    wb=[66,65,2,1];   //w wektorze zapisane sa nr
przemieszczen, ktore przyjmuja wartosc "0" (numeracja musi byc
zgodna z definicja wektora U),
                  //wazne zeby byly zapisane w kolejnosci
malejacej

    //parametry wykorzystywane do obliczenia funkcji celu
    sigmadop=250;    //naprezenia dopuszczalne [MPa]
    u_dop=50;    //dopuszczalne przemieszczenia [mm]
```

```
    kara=100;   //wspolczynnik kary (przekroczenie naprezen
dopuszczalnych)
    kara2=100;   //wspolczynnik kary (przekroczenie
przemieszczen dopuszczalnych)
```

Załącznik nr 9

Plik z kodem (w środowisku Scilab) zawierający kod programu służącego do wyznaczenia jednego, wspólnego dla wszystkich prętów przekroju, tak aby konstrukcja spełniała nałożone na nią ograniczenia.

masa_konstrukcji_bez_optymalizacji.sce

```
//Wyznaczanie masy konstrukcji
//bez optymalizacji, przy zalozeniu spelnienia ograniczen

clear //dezaktywowac w przypadku wywolywania PSO w innym
skrypcie
stacksize('max')

//exec('mes_parametry_kratownica_mp.sce');
//exec('mes_parametry_bridge.sce');
exec('mes_parametry_truss_arch.sce');

//wczytanie modulu MES
exec('mes_modul_funkcja.sci');

//-------------------------------

    //pomocnicze wartosci dla pierwszej iteracji (zeby
spelniony byl warunek w petli while)
        u_max=2*u_dop;
        sigma_max=2*sigmadop;

    //pozostale wartosci poczatkowe
        A=Amin*ones([1:ne]); //przekroj

        dA=((Amax-Amin)/500)*ones([1:ne]);


            while (sigma_max>sigmadop | u_max>u_dop)
                A=A+dA;
                [fcelu, U, M, sigma, F, FX, FY, T, TX,
TY]=mes_module(A,nw,ne,nc,mp,E,ro,FP,wb,sigmadop,kara,kara2,u_
dop);
                u_max=max(abs(U));
                sigma_max=max(abs(sigma));
            end


            disp('*******************************')
            disp('Masa konstrukcji    :  ')
            disp(M)
            disp('Przemieszczenia w wezlach    :  ')
            disp(U)
            disp('Naprezenia w elementach    :  ')
```

```
            disp(sigma)
            disp('Sily w wezlach    :  ')
            disp(F)
            disp('Suma rzutow sil w kierunku X    :  ')
            disp(FX)
            disp('Suma rzutow sil w kierunku Y    :  ')
            disp(FY)
            disp('Suma momentow    :  ')
            disp(T)
            disp('Funkcja celu    :  ')
            disp(fcelu)
            disp('Pole przekroju    :  ')
            disp(A)
            disp('********************************')

            //--------------------------
            //informacja o przekroczonych naprezeniach
dopuszczalnych

                    if sigma_max>sigmadop then
                        h1=1;
                    else
                        h1=0;
                    end


                if max(h1)==1 then
                    disp('********************************')
                    disp('Uwaga! Naprezenia dopuszczalne
przekroczone!')
                    disp('********************************')
                    else
                end
                        //--------------------------
            //informacja o przekroczonych przemieszczeniach
dopuszczalnych

                    if u_max>u_dop then
                        h2=1;
                    else
                        h2=0;
                    end


                if max(h2)==1 then
                    disp('********************************')
                    disp('Uwaga! Przemieszczenia dopuszczalne
przekroczone!')
                    disp('********************************')
                    else
                end
```