AKADEMIA GÓRNICZO-HUTNICZA

IM. STANISŁAWA STASZICA W KRAKOWIE



Wydział Inżynierii Metali i Informatyki Przemysłowej



PRACA DYPLOMOWA MAGISTERSKA

pt.

„Implementacja systemu komputerowego do optymalizacji procesów i cykli produkcyjnych”

Imię i nazwisko dyplomanta: **Mateusz Skiba**

Kierunek studiów: **Informatyka Stosowana**

Specjalność:  **Modelowanie i Technologie Informacyjne**

Nr albumu: **223154**

Promotor: dr inż. Łukasz Rauch

Recenzent: prof. dr hab. inż. Maciej Pietrzyk

Podpis dyplomanta: Podpis promotora:

Kraków 2013

***(Oświadczenia zgodnie z Regulaminu studiów w AGH):***

***Oświadczam, świadomy(-a) odpowiedzialności karnej za poświadczenie nieprawdy, że niniejszą pracę dyplomową wykonałem(-am) osobiście i samodzielnie i że nie korzystałem(-am) ze źródeł innych niż wymienione w pracy.***

Kraków, dnia …… Podpis dyplomanta…………….

**Spis treści**

[1 Wstęp. 4](#_Toc370937438)

[2 Przegląd literatury. 5](#_Toc370937439)

[2.1 Optymalizacja. 5](#_Toc370937440)

[2.1.1 Podstawowe pojęcia związane z optymalizacją. 6](#_Toc370937441)

[2.1.2 Ogólne, teoretyczne sposoby szukania rozwiązań optymalnych. 7](#_Toc370937442)

[2.1.3 Warunki stopu. 9](#_Toc370937443)

[2.1.4 Ogólne klasyfikacja metod optymalizacji. 9](#_Toc370937444)

[2.2 Symulacje numeryczne. 10](#_Toc370937445)

[2.3 Metamodelowanie. 14](#_Toc370937446)

[2.3.1 Idea i charakterystyka sztucznych sieci neuronowych. 14](#_Toc370937447)

[2.3.2 Dostępne rozwiązania. 19](#_Toc370937448)

[2.4 Podsumowanie. 20](#_Toc370937449)

[3 Implementacja systemu. 21](#_Toc370937450)

[3.1 Zarządzanie bazą danych. 22](#_Toc370937451)

[3.1.1 Struktura i schemat bazy lokalnej. 22](#_Toc370937452)

[3.1.2 Wykorzystanie istniejącej bazy materiałowej. 26](#_Toc370937453)

[3.2 Tworzenie i parametryzacja różnych cykli produkcyjnych. 28](#_Toc370937454)

[3.3 Symulacje i metamodelowanie. 35](#_Toc370937455)

[3.4 Optymalizacja. 37](#_Toc370937456)

[4 Analiza przypadków użycia. 42](#_Toc370937457)

[4.1 Proces walcowania prętów jako przykład symulacji z wykorzystaniem zdalnego programu zewnętrznego. 42](#_Toc370937458)

[4.2 Optymalizacja procesu chłodzenia prętów. 46](#_Toc370937459)

[4.3 Chłodzenie szyn przykładem cyklu produkcyjnego wykorzystującego lokalne oprogramowanie symulujące. 51](#_Toc370937460)

[5 Podsumowanie i kierunek rozwoju. 55](#_Toc370937461)

[6 Bibliografia 56](#_Toc370937462)

# **Wstęp.**

Obniżenie kosztów, szybkość produkcji, przy jednoczesnym spełnieniu wszystkich założeń technologicznych oraz braku ingerencji w jakość wykonywanych produktów są najważniejszymi czynnikami, z jakimi musi zmierzyć się współczesny przemysł. Znalezienie optymalnego rozwiązania w tradycyjny sposób poprzez próby przemysłowe jest skomplikowane, ze względu na złożoność postawionych kryteriów oraz niesamowicie kosztowne i niestabilne, ponieważ wymaga zastosowania metody prób i błędów. Z pomocą przychodzą symulacje komputerowe oraz oprogramowanie umożliwiające dokonanie optymalizacji jedno lub wielokryterialnej,   
a zastosowanie analizy odwrotnej pozwala otrzymać produkt, który spełnia wszelkie postawione wymagania bez dodatkowych kosztów. Jednakże, samo oprogramowanie, które w chwili obecnej znajduje się na rynku nie jest idealne. W projekcie inżynierskim dokonano dokładnego przeglądu dostępnych aplikacji, które w części swojej funkcjonalności i idei umożliwiają zarządzanie   
i integrację obliczeń numerycznych oraz przeprowadzanie samej optymalizacji otrzymanych wyników. Można wywnioskować z niego, że aktualny rynek nie jest obfity w takie rozwiązania,   
a dostęp do informacji na temat istniejących systemów nie jest łatwy. Większość produktów jest płatna, a udostępniona darmowa dokumentacja jest niekompletna. Dodatkowo wiele zastrzeżeń można mieć do uniwersalizmu poszczególnych rozwiązań. Większość z nich umożliwia dokonanie optymalizacji konkretnego procesu.

Wszystkie te czynniki miały ogromny wpływ na napisanie przedstawianej pracy oraz okazały się główną motywacją do stworzenia alternatywnego do obecnych na rynku oprogramowania. Biorąc pod uwagę te aspekty uformował się cel projektu, jakim była implementacja systemu, który umożliwia przeprowadzenie optymalizacji procesów i cykli produkcyjnych.

W pierwszej części pracy przybliżone zostanie teoretyczne podejście do problemu optymalizacji. Następnie opisane zostaną implementacyjne szczegóły systemu, poczynając od narzędzi do zarządzania bazy danych oraz bazy materiałowej. Pokazany zostanie etap tworzenia   
i parametryzacji cyklu produkcyjnego oraz dokonywanie jego symulacji za pomocą obliczeń numerycznych lub nauczonego metamodelu. Końcowym punktem będzie zaprezentowanie funkcjonalności przeprowadzania optymalizacji. Kolejna część pracy poświęcona jest działaniu systemu na konkretnych cyklach produkcyjnych. Analiza przeprowadzona zostanie dla cyklu kucia kotw. Podczas podsumowywania przybliżone zostaną również możliwe kierunki dalszego rozwoju oprogramowania.

# **Przegląd literatury.**

Rozdział ten poświęcony jest teoretycznemu podejściu do tematu optymalizacji. Przybliżone zostaną podstawowe pojęcia oraz ogólny sposób rozwiązywania problemów związanych   
z optymalizacją, co pozwoli na zrozumienie działania aplikacji. Sklasyfikowane zostaną same metody optymalizacji oraz opisane zostaną warunki stopu. W kolejnej części rozdziału przedstawione zostaną teoretyczne podstawy sposobów rozwiązywania problemów symulacyjnych w dwojaki sposób tj. za pomocą samych symulacji numerycznych oraz wykorzystania metamodeli opartych na sztucznych sieciach neuronowych. Zakończeniem rozdziału będzie krótkie podsumowanie dokonanego wstępu teoretycznego.

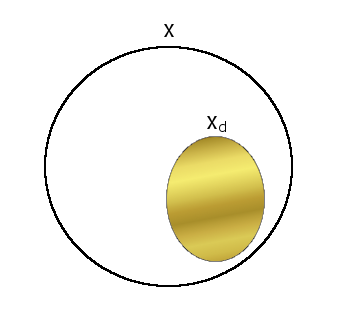
## Optymalizacja.

Zagadnienia optymalizacji wykorzystywane są praktycznie we wszystkich dziedzinach życia, zarówno w nauce, ekonomii czy przemyśle. Ich rozwiązywanie nie może odbyć się bez znajomości metod pozwalających na znalezienie optymalnego rozwiązania. Ze względu na brak uniwersalnego sposobu w tym zakresie, spektrum dostępnych algorytmów jest bardzo obszerne. Efektywność oraz czas obliczeń danej metody zależy od optymalizowanej funkcji. Metoda, która jest idealna do rozwiązania jednego problemu, może nie sprawdzić się podczas rozpatrywania innego problemu. Uogólniając, celem całego procesu optymalizacji jest znalezienie najlepszego (minimalnego lub maksymalnego) rozwiązania danego problemu, które mieści się w granicach postawionych kryteriów oraz ograniczeń początkowych. Zastosowanie dostępnych metod optymalizacji pozwala uniknąć rozwiązywania problemu metodą prób i błędów, która jest nieefektywna oraz czasochłonna. Wybór odpowiedniego algorytmu pozwala na drodze numerycznej bądź analitycznej znaleźć optymalne rozwiązania bez konieczności przeszukiwania i analizowania wszystkich możliwych sposobów. Samo podejście do rozwiązywania problemu optymalizacji musi rozpoczynać się od odpowiedniego sformułowania analizowanego problemu optymalizacji, zbioru poszukiwań oraz funkcji celu. Poszerzając, poszukiwanie optimum związane jest z następującymi zadaniami:

* opracowaniem modelu matematycznego analizowanego procesu,
* zdefiniowaniem funkcji celu optymalizacji (kryterium optymalizacji, kryterium jakości itp.),
* poszukiwaniem optymalnego rozwiązania z zastosowaniem jednej z metod optymalizacji, spełniającego przyjęte kryterium optymalizacji.

### Podstawowe pojęcia związane z optymalizacją.

Przybliżenie tematu optymalizacji nie może odbyć się bez wyjaśnienia podstawowych definicji i terminów z nim związanych takich, jak zbiór rozwiązań dopuszczalnych, funkcja celu czy rozwiązanie optymalne. Każde zadanie optymalizacyjne rozwiązuje się na przestrzeni zmiennych optymalizacji (zmiennych decyzyjnych) X, w której definiuje się zbiór rozwiązań dopuszczalnych Xd (rysunek 1).



Rys. 1 Zbiór rozwiązań dopuszczalnych Xd w przestrzeni wszystkich rozwiązań X   
[Źródło: Opracowanie własne]

Według definicji zbiór rozwiązań dopuszczalnych Xd jest zbiorem punktów **x** ϵ X, które są brane w procesie optymalizacji. Definiowane są one przeważnie poprzez podanie pewnych ograniczających warunków, które muszę zostać spełnione, aby wektor **x** mógł należeć do zbioru Xd. Jeśli żadne ograniczenia nie są określone przyjmujemy, że zbiór rozwiązań dopuszczalnych jest równy przestrzeni wszystkich rozwiązań, a taki przypadek nazywany jest optymalizacją bez ograniczeń.

Kolejnym nierozłącznie związanym z dokonywaniem optymalizacji pojęciem jest funkcja celu, czasem nazywanej również kryterium optymalizacji, kryterium jakości czy funkcjonałem jakości. Funkcja celu przyporządkowuje poszczególnym rozwiązaniom ze zbioru rozwiązań dopuszczalnych pewną wartość liczbową, co umożliwia porównywanie rozwiązań, które ma wpływ na późniejszy wybór rozwiązania dopuszczalnego dającego minimalną lub maksymalną wartość funkcji celu. Zazwyczaj zakłada się, że funkcja celu jest funkcją ciągłą, ponieważ znalezienie minimum funkcji nieciągłej jest trudne i wymaga zastosowania skomplikowanych algorytmów.

Najważniejszym terminem, który przewijać się będzie w przedstawianej pracy jest rozwiązanie optymalne **x\***, czyliminimum globalne, które dla wszystkich **x** ϵ Xd spełnia   
warunek 1:

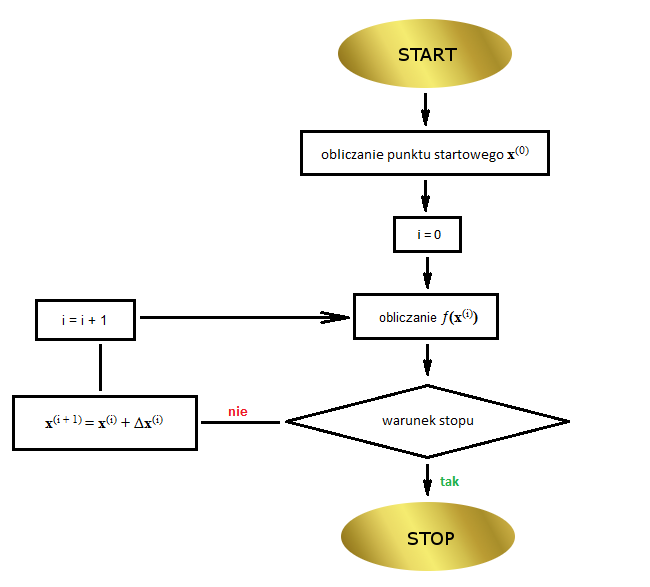
ƒ(**x\***) ≤ ƒ(**x**) *1*

Jeżeli warunek ten jest spełniony tylko w pewnym otoczeniu punktu **x\***, to mówimy o minimum lokalnym. Warto ponownie zwrócić uwagę, że optymalne rozwiązania (minimalne lub maksymalne) funkcji celu znajdują się w zbiorze rozwiązań dopuszczalnych. Istotnym elementem jest spostrzeżenie, że pojęcia rozwiązanie minimalne i maksymalne są w pewnym stopniu stosowane wymiennie. Dzieje się tak dlatego, że każde zadanie polegające na poszukiwaniu maksimum funkcji celu może zostać zastąpione zadaniem odwrotnym (poszukiwania minimum). Rozwiązanie maksymalne **x\*** funkcji ƒ jest jednocześnie rozwiązaniem minimalnym funkcji –ƒ i odwrotnie, co znaczy, że znając algorytm maksimum dowolnej funkcji celu, jesteśmy w stanie określić sposób rozwiązania minimalnego.

### Ogólne, teoretyczne sposoby szukania rozwiązań optymalnych.

Kolejnym bardzo ważnym elementem, który musi znaleźć się w podstawach teoretycznych dotyczących optymalizacji jest ogólne rozwiązywanie zadań polegających na szukaniu rozwiązań optymalnych. Każde zadanie optymalizacyjne nawiązuje do znalezienia maksymalnej lub minimalnej wartości funkcji celu przy założonych ograniczeniach, czyli takie które zawiera się w zbiorze rozwiązań dopuszczalnych.

Metody optymalizacyjne zazwyczaj sprowadzają się do iteracyjnego poszukiwania odpowiedniej wartości funkcji celu [8]. Początkowo wybierany jest punkt startowy **x**(0) należący do zbioru rozwiązań dopuszczalnych, a następnie obliczana jest dla niego początkowa wartość funkcji optymalizowanej. Kolejnym krokiem jest sprawdzenie warunku stopu optymalizacji, szerzej opisanego w podrozdziale 2.1.3 Jeśli warunek jest spełniony następuje zakończenie procesu optymalizacji, w innym przypadku wybierany jest kolejny punkt **x**(i + 1) = **x**(i) **+** Δ**x**(i), dla którego ponownie obliczana jest funkcja celu. Cała procedura powtarzana jest, aż do otrzymania pozytywnego wyniku ze sprawdzania warunku stopu. Ogólna strategia rozwiązywania zadań optymalizacyjnych została przedstawiona na rysunku 2.



Rys. 2 Schemat blokowy ogólnego sposobu szukania rozwiązania problem optymalizacyjnego [Źródło: opracowanie własne]

Każdą metodę optymalizacji wyróżnia algorytm określania wektoraΔ**x**, który zazwyczaj zależy od wyznaczania:

* kierunku poszukiwań,
* długości kroku, wykonywanego w kierunku poszukiwań.

Powyższe stwierdzenie można zapisać za pomocą równania Δ**x** = s**d**, przyjmują **d** jako kierunek poszukiwań oraz s jako długość kroku. Podstawiając do równania na kolejny krok iteracji otrzymujemy ogólną postać równania na nowy punkt poszukiwań rozwiązania optymalnego [2]:

**x**(i + 1) = **x**(i) **+** Δ**x**(i) = **x**(i) + s(i)**d**(i) *2*

### Warunki stopu.

Obecne metody optymalizacji nie dają idealnego rozwiązania optymalnego, ponieważ prawdopodobieństwo, że algorytm natrafi na rzeczywiste minimum funkcji optymalizacyjnej   
ƒ jest niewielkie. Dzieje się tak choćby dlatego, że numeryczna reprezentacja liczb ma ograniczoną dokładność. W praktyce rozwiązanie algorytmu jest bliskie rzeczywistemu,   
a warunki stopu nałożone na metodę mają określić dokładność, która będzie zadowalająca.

Najczęściej stosowanymi warunkami stopu są testy zbieżności.

* Otrzymana w kolejnych iteracjach wartość funkcji celu jest bliska wartości otrzymanej   
  w poprzedniej iteracji, może świadczyć o braku podstaw do dalszego wykonywania algorytmu, ponieważ najprawdopodobniej nie będzie miało to znacznego wpływu na poprawę wyniku, nawet przy dużej ilości kroków iteracyjnych.
* Otrzymany w kolejnych iteracjach punkt niewiele różniący się od punktu z poprzedniej iteracji, w analogiczny sposób jak w teście powyższym może świadczyć o zbieżności rozwiązania.

Część algorytmów dopuszcza jako warunek stopu ilość iteracji jaka ma zostać wykonana. Można również spotkać warunki wynikające ze specyfiki danej metody. Warto wspomnieć, że dla każdego warunku stopu może nastąpić zatrzymanie wykonywania algorytmu w miejscu,   
w którym nie powinno, ponieważ akurat dla danej funkcji celu będzie on niepoprawny. Przykładowo, kolejne iteracje algorytmu wskażą ekstremum lokalne funkcji celu, a nie poszukiwane ekstremum globalne. Istotny wpływ związany z tym faktem na działanie algorytmu, jego efektywność oraz efekt końcowy ma wybór punktu startowego. Zły dobór tego punktu może spowodować „zapętlenie” się algorytmu w minimum lokalnym. Niestety nie ma złotego środka na dobre rozpoczęcie wykonywania algorytmu, dlatego, aby uniknąć tego problemu, algorytm startuje z różnymi punktami. Następnie porównywane są wyniki.

### Ogólne klasyfikacja metod optymalizacji.

Metody optymalizacji mogą zostać zakwalifikowane na wiele sposobów. Głównym kryterium branym pod uwagę podczas dobierania właściwej metody optymalizacji jest rodzaj rozwiązywanego zadania. Następnie wpływ na metodę mają ograniczenia, wymiar problemu, czy kryteria optymalizacji. Ogólnym podział metod optymalizacji został zamieszczany w tabeli 1.

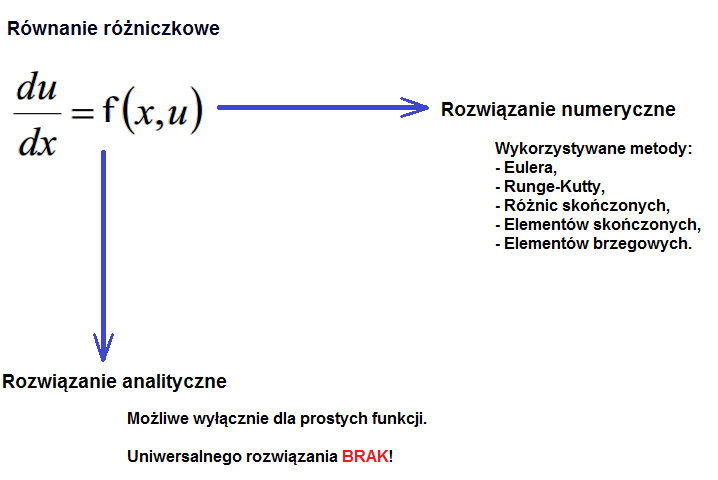
|  |  |  |  |
| --- | --- | --- | --- |
| **Kryterium klasyfikacji metody** | | | |
| Rodzaj rozwiązywanego zadania | Ograniczenia | Wymiar problemu  (liczba zmiennych optymalizacji) | Kryteria optymalizacji |
| - Metody programowania liniowego  - Metody optymalizacji nieliniowej | - Metody optymalizacji bez ograniczeń  - Metody optymalizacji z ograniczeniami | - Metody jednowymiarowe  - Metody wielowymiarowe | - Metody dla pojedynczego kryterium  - Metody wielokryterialne |

Tab. 1 Ogólny podział metod optymalizacji [Źródło: 1]

## Symulacje numeryczne.

Złożoność zjawisk zachodzących nie tylko w procesach przetwórstwa metali, ale we wszystkich zagadnieniach technologicznych jest ogromna. Większość z nich może zostać opisana za pomocą równań, których nie sposób jest rozwiązać w sposób analityczny. W rezultacie takie równania wymagają numerycznego traktowania. Generalną ideą eksperymentów numerycznych jest symulowanie procesu poprzez rozwiązanie odpowiedniego układu równań matematycznych, które dany przypadek opisują. Typowym sposobem patrzenia na symulacje numeryczne jest konstrukcja modelu matematycznego, zazwyczaj w formie układu równań różniczkowych,   
a następnie transformacja tych równań do postaci dyskretnej, która może już być rozwiązana numerycznie. W konsekwencji symulacje numeryczne polegają na zadaniu warunków początkowych i śledzeniu ich ewolucji w czasie. Schemat pokazujący niezbędność przejścia   
z rozwiązania analitycznego na rozwiązanie numeryczne pokazany został na rysunku 3.

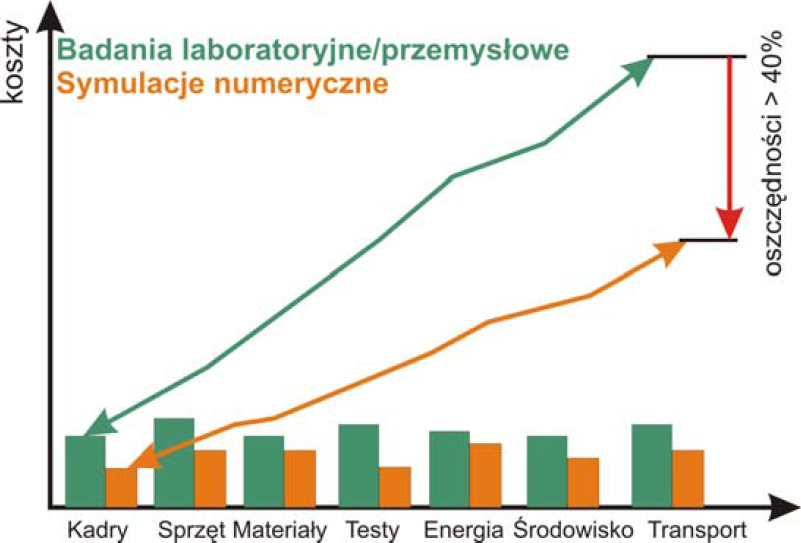
Jako nowa dziedzina symulacje numeryczne pojawiły się uzupełniając zarówno teorię jak   
i eksperyment. Można przypuszczać, że mogłyby zastąpić wyniki otrzymywane doświadczalnie, jednak na ogół symulacje i eksperyment uzupełniają się. Symulacje są zazwyczaj kalibrowane wynikami z badań, natomiast wyniki wyjaśniane są symulacjami. Na rozwój symulacji niebagatelny wpływ miał również rozwój dostępnych technologii komputerowych oraz wyższa moc obliczeniowa, natomiast na jakości otrzymanych wyników znacząco odcisnęła się polepszająca się jakość algorytmów. Te czynniki przyczyniły się do zainteresowania   
i popularności, jaką cieszą się obecnie symulacje numeryczne [10].



Rys. 3 Schemat pokazujący zalety przejścia z rozwiązywania analitycznego na rozwiązanie numeryczne [Źródło: wykłady dr hab. inż. Łukasza Madeja z zakresu modelowania wielkoskalowego udostępnione na stronie: http://home.agh.edu.pl/~lmadej/]

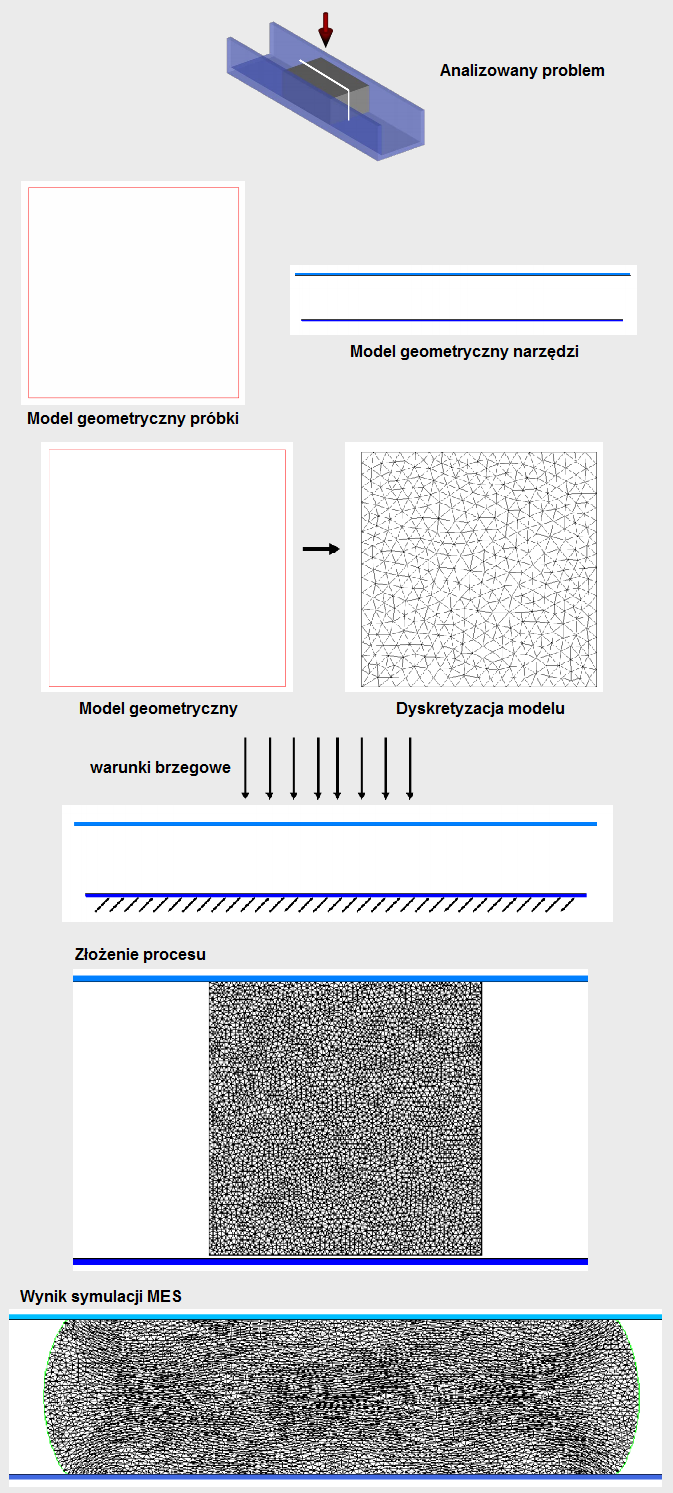
Rozpoczęcie stosowania symulacji numerycznych jest niezbędne we współczesnym przemyśle głównie ze względu na generowane dzięki nim oszczędności. Stosowane wcześniej próby przemysłowe wymagały od przedsiębiorstwa każdorazowego generowania produktu   
w celu sprawdzenia poprawności zmienionych własności. Poprzez zastosowanie symulacji numerycznych koszty zostały ograniczone o około 40% w każdej gałęzi generowania kosztów, co zostało zilustrowane na rysunku 4.

Wyniki przeprowadzonej symulacji numerycznej nierozłącznie zależą od założeń początkowych przyjętych w modelu. Jeśli podczas przeprowadzanej symulacji wynik różni się od tego otrzymanego doświadczalnie przyjmuje się, że model problemu jest źle skonstruowany. Doskonalenie modelu polega na otrzymaniu podobnych wyników jak w przypadku badań, jeśli symulowane jest rzeczywiste zdarzenie [9].



Rys. 4 Wykres ilustrujący zysk podczas wykorzystania symulacji numerycznych [Zródło: wykłady dr hab. inż. Łukasza Madeja z zakresu modelowania wielkoskalowego udostępnione na stronie: http://home.agh.edu.pl/~lmadej/]

Budowa samego modelu procesu przebiega przez kilka etapów, podobnych dla każdej metody. Krokiem wspólnym jest definicja analizowanego problemu fizycznego. Jak wspomniano wyżej niezbędne jest ustanowienie równań oraz warunków początkowych dla wczytanej geometrii, która może zostać przygotowana w programie CAD/CAM. Na przykładzie metody elementów skończonych dalsza symulacja polega na dyskretyzacji regionu na skończoną ilość podregionów oraz wprowadzenie równań odpowiednich dla zastosowanego typu podregionu. Elementem końcowym jest złożenie procesu oraz rozwiązanie uzyskanego układu równań. Po tych krokach można przejść do analizy otrzymanych wyników. Schemat budowy prostego procesu za pomocą metody elementów skończonych został zaprezentowany na rysunku 5.



Rys. 5 Schemat budowy prostego procesu do symulacji za pomocą MES   
[Źródło: Opracowanie własne].

## Metamodelowanie.

Tradycyjny model analizowanego procesu bardzo często wymaga znacznych nakładów finansowych w celu przeprowadzenia wymaganej ilości testów, przez co staje się bardzo czasochłonny, dlatego wykorzystuje się zdefiniowany i nauczony wcześniej metamodel. Metamodel postrzegany jest jako pewna abstrakcja stworzona na bazie modelu analizy procesu rzeczywistego zbudowana z wykorzystaniem metod modelowania matematycznego, która   
w wypadku braku matematycznych aproksymacji może w prosty sposób je zastąpić. Prawidłowo opracowany metamodel umożliwia natychmiastowe obliczenie wartości funkcji celu, przez co czas optymalizacji znacznie się skraca. Ważnym aspektem jest to, że metamodel jest tylko przybliżeniem obliczanej wartości funkcji, przez co niezbędne jest określenie zadowalającej dokładności podczas jego nauczania. W celu zbudowania metamodelu wykorzystana może być dowolna metoda aproksymacji, która zapewni skrócenie obliczeń jednocześnie nie wpływając na dokładność porównywaną z oryginalnym modelem.

Metodą, która została użyta w przedstawianej pracy do budowy metamodeli jest tą najbardziej popularną – sztuczne sieci neuronowe. Istnieje jednak wiele innych metod dość popularnych, których przykładem może być Metoda Powierzchni Odpowiedzi (RSM – z ang. Response Surface Methodology), która pomaga określić zależności pomiędzy kilkoma zmiennymi objaśniającymi a jedną lub kilkoma zmiennymi odpowiedzi. Kolejnym sposobem na budowanie metamodelu może być metoda Kriging, dzięki której otrzymuje się jedne   
z najlepszych nieobciążonych liniowo oszacowań wartości analizowanej zmiennej zregionalizowanej. Danym punktom pomiarowym zawierających się w obszarze estymacji przydziela się wagi (współczynniki krigingu) w odpowiedni sposób. Wykonuje się to w celu minimalizacji średniokwadratowego błędu estymacji (wariancji krigingu). W opisywanej pracy przybliżona została ta najbardziej popularna i wykorzystana w projekcie metoda sztucznych sieci neuronowych.

### Idea i charakterystyka sztucznych sieci neuronowych.

Esencją budowania sztucznych sieci neuronowych jest próba symulacji działania ludzkiego mózgu. Idea sieci neuronowej zakłada, że część kluczowych właściwości biologicznych neuronów może być zastosowana w symulacjach, tworząc tym samym uproszczony „mózg”. Jedna z najistotniejszych rzeczy, która musi zostać wspomniana dotyczy komponentów sztucznej sieci neuronowej, które w założeniu są próbą odtworzenia potencjału obliczeniowego mózgu. Do tej pory niestety nie udało się wykonać aż tak złożonej symulacji. Patrząc na ludzki mózg posiadający dziesiątki, setki miliardów neuronów, typowa sztuczna sieć neuronowa posiadająca ich nie więcej niż tysiąc wypada miernie. Metoda znajduje jednak szerokie zastosowanie podczas zadań opierających się na rozpoznawaniu pisma, obrazu, podczas dokonywania prognoz giełdowych, syntezy mowy, czy właśnie optymalizacji.

Zrozumienie sztucznych sieci neuronowych nie może obyć się bez zapoznania się   
z działaniem prawdziwego mózgu, ponieważ to na nim cała metoda jest wzorowana. Najważniejszym aspektem jest połączenie między neuronami i wpływ sąsiadujących neuronów na te połączenia. Rozkład sił oddziałujących na dany neuron powoduje utrwalenie informacji   
w mózgu. Wartości danych sił ulegają zmianie, jedne słabną, a nawet zanikają, podczas gdy drugie zostają wzmocnione, poprzez uczenie się, powtarzanie danej czynności czy zdobywanie doświadczenia. Bardzo istotne jest również rozróżnienie neuronów na pobudzające i hamujące. W rzeczywistości aktywacja jednego neuronu powoduje zwiększenie lub zmniejszenie wartości neuronów sąsiadujących, a wielkość zmiany uzależniona jest od siły połączenia. Trwalsze połączenia powodują większe oddziaływania, natomiast słabsze odpowiednio mniejsze. Ostatnią najistotniejszą cechą jest funkcja przenoszenia opisująca reakcję danego neuronu w zależności od otrzymywanych danych wejściowych. Każdy neuron posiada próg ilościowy, który determinuje stopień jego aktywności. Matematyczny zapis takiego zachowania nazywany jest funkcją aktywacji.

Konstruowanie prawidłowej sieci neuronowej nie może się zatem odbyć bez określenia trzech cech opisanych powyżej:

* siły połączeń,
* klasyfikacji neuronów względem pobudzania/hamowania,
* funkcji aktywacji.

Teoretycznie, właściwości sztucznego neuronu nie powinny odróżniać się od jego biologicznego odpowiednika. Siła połączenia między neuronami mieści się w przedziale od -1.0, oznaczając tą najbardziej hamującą, do +1.0, czyli największego pobudzenia. Funkcja aktywacji podczas tworzenia sieci neuronowej zazwyczaj jest predefiniowana i zależy od rodzaju rozwiązywanego problemu. Możemy wyróżnić trzy główne rodzaje funkcji aktywacji: funkcję progową (skoku jednostkowego), liniową i nieliniową, która charakteryzuje się największą zdolnością nauki,   
a wykorzystywana jest podczas budowy siedzi wielowarstwowych. Najczęściej jednak stosowana jest funkcja sigmoidalna (krzywa logistyczna), przyjmująca wartości pomiędzy 0, a 1.

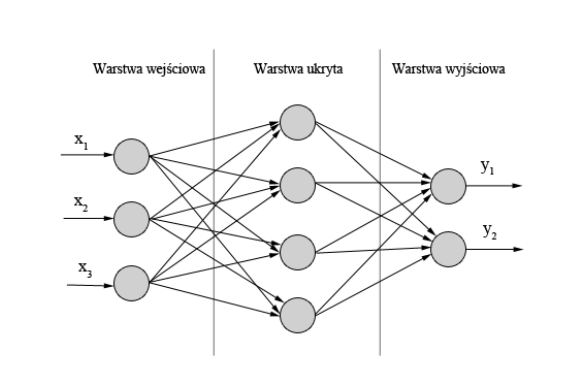
Organizacja neuronów jest jedną z najbardziej zauważalnych różnic pomiędzy biologicznymi, a sztucznymi sieciami neuronowymi, w których możliwe jest utworzenie wielu różnych architektur sieci neuronowej, jednak sama jej organizacja opiera się na tej samej, podstawowej strukturze. Istnieje jednak ryzyko złego doboru odpowiedniej budowy sieci, czego następstwem są błędne wyniki, a zależeć to może od ilości i rodzaju danych uczących (ilość neuronów na warstwie wejściowej oraz wyjściowej). Prawidłowy dobór architektury sprowadza się do wykonania szeregu testów w celu sprawdzenia szybkości uczenia się oraz uzyskanego błędu. Za każdym razem modyfikacji ulega ilość neuronów i parametry sieci. W praktyce zaleca się używanie maksymalnie dwóch warstw ukrytych sieci, natomiast sieć posiadająca jedną warstwę ukrytą powinna nauczyć się i rozwiązać większość podstawowych problemów.

Architektura każdej sieci neuronowej zawiera trzy komponenty:

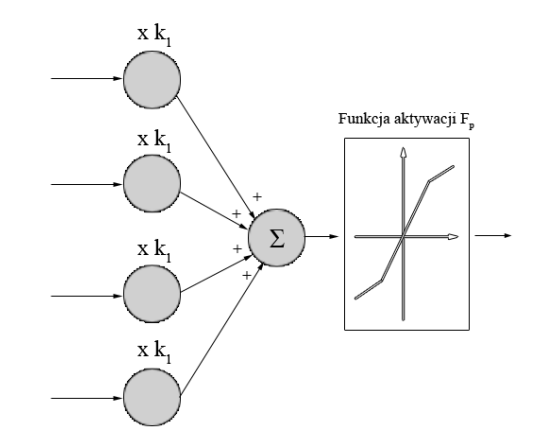
* zestaw neuronów wejściowych (warstwa wejściowa),
* jedną lub więcej warstwę ukrytą posiadającą dowolną ilość neuronów ukrytych,
* warstwę wyjściową (liczba neuronów nie musi odpowiadać liczbie neuronów w warstwie wejściowej).

Schemat architektury sztucznej sieci neuronowej został przedstawiony na rysunku 6. Węzły wejściowe przyjmują dane do nauki, mogą być one dowolne, które można przedstawić w formie numerycznej (cyfrowy obraz, historia kursu walut, dowolna funkcja). Każda informacja wejściowa otrzymuje swoją wartość aktywacyjną, która następnie jest przekazywana do połączonych ze sobą neuronów.

Siła połączenia, zmienne pobudzające lub hamujące oraz funkcja aktywacji określają, jaka ilość wartości początkowej zostanie przekazana do kolejnego węzła. Zadaniem każdego   
z neuronów jest zsumowanie wartości, które otrzymał i analogicznie, stosując ten sam proces, przekazanie zmodyfikowanej wartości do każdego węzła, z którym jest połączony, w głąb sieci. Informacja przekazywana jest do momentu aktywacji warstwy wyjściowej (rysunek 7). Prawidłowo zbudowana sieć powinna zwrócić wartość znacznie nieodbiegającą od tej otrzymanej podczas testów [2].



Rys.6 Schemat budowy sztucznej sieci neuronowej [Źródło: 3].



Rys.7 Schemat działania pojedynczego neuronu [Źródło: 3].

Korzystając z powyższej charakterystyki łatwo można wywnioskować, że w celu prawidłowego nauczenia sieci wystarczy w odpowiedni sposób dostosować wartości połączeń pomiędzy neuronami, ponieważ siły połączeń mają kluczowy wpływ na ilość przekazanych danych, a finalnie na otrzymany wynik. Jedną z najpopularniejszą metodą uczenia jest metoda Wstecznej Propagacji Błędu, czyli tak zwane uczenie z nadzorem lub nauczycielem. Polega ona na wykorzystaniu dwóch zbiorów danych: uczącego i weryfikującego. Zestaw uczący powinien dokładnie odzwierciedlać rozpatrywany problem, a pojedyncza porcja danych nazywana jest wektorem uczącym, który składa się z danych wejściowych oraz danych, które powinna zwrócić sieć na wyjściu. Po pierwszej epoce (iteracji), czyli przetworzeniu całego ciągu uczącego, otrzymujemy wynik, który zazwyczaj znacząco różni się od szukanej, prawidłowej wartości. Następnie liczony jest błąd dla danej iteracji i rozpoczyna się kolejny cykl. Schemat ten jest powtarzany do momentu uzyskania zadowalającego błędu minimalnego, a następnie uzyskana wartość zostaje porównana z właściwą odpowiedzią, wykorzystana zostaje wsteczna propagacja błędu. Analizując przebieg aż do węzłów wyjściowych, modyfikowana jest wartość każdego połączenia, w celu zwrócenia odpowiedzi bliższej prawidłowej w kolejnej epoce. Sieć otrzymuje wtedy kolejny zestaw danych wejściowych i ponownie stara się odpowiednio dostosować wagi połączeń. Proces uczenia, w zależności od wybranych danych wejściowych może zająć od kilkuset do kilkudziesięciu tysięcy iteracji. Wartym zauważenia jest fakt, że nie zawsze zastosowanie większej ilości iteracji skutkuje lepiej nauczoną siecią neuronową. Zbudowane sztuczne sieci neuronowe posiadają tolerancję na nieciągłość czy niewielkie braki w zbiorze uczącym, dzięki uogólniania wiedzy. Złożoność obliczeniowa sieci oraz czas nauczania zależy głównie od wykorzystywanej architektury sieci i rozmiaru zbiorów uczących (im bardziej rozległa architektura oraz duży rozmiar zbioru uczącego tym większy czas nauczania oraz złożoność). Dodatkowo, rzadko spotykana jest sytuacja uzyskania optymalnej sieci już za pierwszym razem. Ponadto, brak jest pewności, że nie utknie ona w minimum lokalnym podczas poszukiwania ekstremum globalnego. W następstwie tego istnieje możliwość regulacji stromości funkcji aktywacji oraz wpływu zmian wag na proces uczenia sieci. Za pomocą współczynników momentum, które kontrolują różnicę pomiędzy wzorcem, a obrazem uzyskanym podczas ostatniej iteracji oraz samego uczenia można w łatwy sposób wpłynąć na jakość oraz szybkość procesu nauczania sieci. Dobór odpowiednich parametrów oraz architektury sieci ma kluczowy wpływ na prawidłowość otrzymanych wyników.

### Dostępne rozwiązania.

Istnieje szereg rozwiązań wspomagających budowę i nauczanie sztucznych sieci neuronowych. Warto jednak zapoznać się z tymi najbardziej popularnymi i dającymi najbardziej optymalne rozwiązania.

Rozważając budowanie sztucznych sieci neuronowych nie można pominąć środowiska MATLAB, które stało się standardem we współczesnych obliczeniach naukowo-technicznych. Sam program jest interpretatorem skryptów napisanych za pomocą języka wysokiego poziomu, który jest dostarczany wraz z całym środowiskiem. Głównymi zadaniami oprogramowania są obliczenia inżynierskie, jednak zauważając mnogość jego funkcji znajduje zastosowanie   
w bardzo różnych dziedzinach. Do budowy sieci neuronowej wykorzystywane jest rozszerzenie Neural Network Toolbox, które wzbogaca całe środowisko o funkcje projektowania, implementacji, wizualizacji oraz symulacji sieci neuronowych. Główną zaletą rozszerzenia jest jego graficzny interfejs, dzięki któremu w łatwy sposób można skorzystać z opisanych wyżej funkcjonalności. W sam komponent wbudowany jest zestaw funkcji uczących sieć, wspierających różne architektury (bez nauczyciela oraz z nauczycielem). Użytkownik ma możliwość definiowania ilości neuronów na każdej z warstw - wejściowej, ukrytej oraz wyjściowej. W środowisku MATLAB zautomatyzowane jest skalowanie danych wejściowych do zakresu funkcji aktywacji. Dodatkowo możliwa jest wizualizacja nauczonej sieci jak i danych zwróconych przez symulację.

Kolejnym oprogramowaniem wspierającym budowanie metamodeli jest SNNS (Stuttgart Neural Network Simulator), oprogramowanie stworzone przez Institute for Parallel and Distributed High Performance Systems na Uniwersytecie w Stuttgard’zie. Celem projektu była budowa środowiska umożliwiającego użytkownikowi rozbudowaną symulację sieci neuronowych w celach badawczych. Symulator SNNS składa się z dwóch elementów – jądra napisanego w języku C oraz interfejsu graficznego. Jądro systemu operuje na wewnętrznej strukturze i jest odpowiedzialny za wykonywanie wszelkich wymaganych operacji. Natomiast graficzny interfejs kontroluje całą symulacją oraz umożliwia wizualizację sztucznej sieci neuronowej. Dodatkowo, interfejs może być wykorzystany w celu bezpośredniego tworzenia, edycji oraz wizualizacji sieci na wiele sposobów. Dzięki tej funkcjonalności możliwe jest tworzenie nawet najbardziej skomplikowanych sieci. Istnieje możliwość rozszerzenia środowiska przez użytkownika o dodatkowe funkcje aktywacji, funkcje wyjściowe czy nawet metody uczenia, które dołączane są do jądra w postaci prostych programów w języku C. SNNS   
w pełni wspiera metodę wstecznej propagacji błędów.

Istnieje również szereg bibliotek umożliwiających szybką budowę sztucznych sieci neuronowych przy pomocy samodzielnie zaimplementowanego oprogramowania. Można doszukać się wiele witryn udostępniających samo oprogramowanie oraz obszerną bazę wiedzy na ten temat. Kilka z najpopularniejszych rozwiązań to:

* FANN (C),
* Annie (C++),
* Neuroph (Java),
* Flood (Objective-C),
* AForge (C#).

## Podsumowanie.

Przedstawiony powyżej wstęp teoretyczny jest niezbędny, aby zrozumieć opisane w dalszej części pracy działanie zaimplementowanej aplikacji. Napisany system do optymalizacji procesów i cykli produkcyjnych wspiera przeprowadzanie symulacji za pomocą symulacji numerycznych oraz metamodeli, z możliwością ich importowania. Korzystając z symulacji otrzymujemy wyniki dla dokonanej wcześniej parametryzacji procesu czy cyklu. Przeprowadzona później optymalizacja jest zautomatyzowana i tak reguluje zdefiniowanymi zmiennymi, aby spełniona była założona funkcja celu. Po przeprowadzonym procesie optymalizacji dokonana może być ponowna symulacja z już zmodyfikowanymi parametrami.

Optymalizacja może odbywać się w zakresie całego cyklu lub pojedynczego procesu. Podczas symulacji parametry przekazywane są z jednego procesu na kolejny. Szczegóły implementacyjne systemu z detalami dotyczącymi zarządzania bazą danych oraz tworzenia i parametryzacji procesów opisane zostały w rozdziale 3.

# Implementacja systemu.

W przedstawianym rozdziale opisane zostaną wszelkie szczegóły implementacyjne systemu przedstawiony zostanie etap projektowania oraz dostępne funkcjonalności. Skonstruowaniu uległa cała baza danych, dlatego niezbędny będzie opis nowopowstałego schematu. Do projektu została dołączona również ogólnodostępna baza materiałowa, która jest wspólna dla kilku projektów realizowanych na Akademii Górniczo-Hutniczej.

Przeprojektowanie bazy danych systemu wymusiło zmiany w interfejsie aplikacji oraz skłoniło do zaprogramowania nowych funkcjonalności. W tej części pracy pokazana zostanie również możliwość tworzenia cykli produkcyjnych oraz ich parametryzacji, a co za tym idzie opisany zostanie etap definiowania transformatorów – zewnętrznych lub zintegrowanych komponentów umożliwiających łączenie procesów w cykl produkcyjny, jak również interpretujących wyniki z danej symulacji czy optymalizacji.

Kolejnym krokiem będzie zwrócenie uwagi na przeprowadzanie samych obliczeń dla danego, sparametryzowanego procesu za pomocą symulacji numerycznych lub metamodeli.   
W systemie zaimplementowana została struktura odpowiadająca za import wcześniej zdefiniowanych modeli. Ostatnim istotnym członem tego rozdziału będzie przybliżenie samego etapu przeprowadzanej optymalizacji. System ma możliwość wyboru metody, przy użyciu której ma zostać zoptymalizowany proces lub cykl produkcyjny.

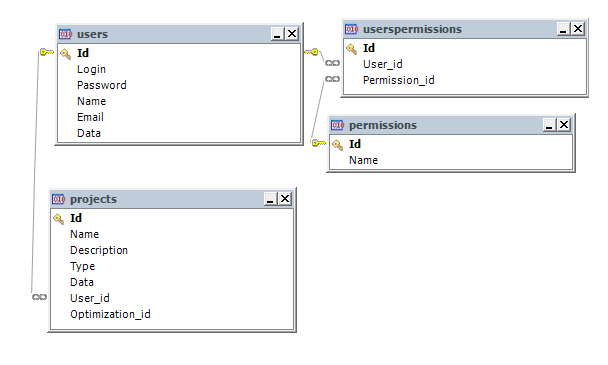
Prezentowana aplikacja napisana jest w języku C#, korzystając z frameworku .NET 4. Podczas realizowania projektu wykorzystana została baza danych MySQL oraz system mapowania obiektowo-relacyjnego – Fluent NHibernate. Wygląd aplikacji, podobnie jak kod źródłowy, wykonany został za pomocą środowiska Microsoft Visual Studio Ultimate   
w technologii WPF. Cały zaimplementowany system jest klientem, który nawiązuje połączenie   
z serwerem bazy danych, co spełnia wymogi architektury klient-serwer i daje możliwość korzystania z aplikacji jednocześnie przez większą ilość użytkowników. Aplikacja jest wielojęzyczna, a jej interfejs może być prezentowany w języku polskim i angielskim.

## Zarządzanie bazą danych.

### Struktura i schemat bazy lokalnej.

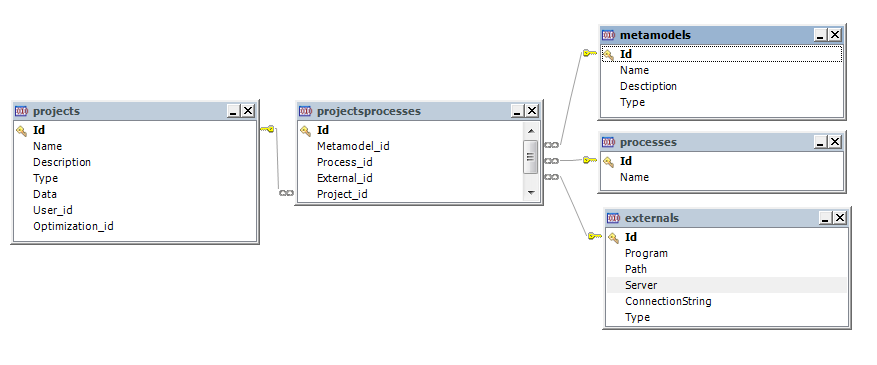
Porównując bazę danych prezentowaną w ramach projektu inżynierskiego z obecną architekturą można stwierdzić, że uległa uproszczeniu. Schemat jest bardziej przejrzysty   
i prezentuje czytelniejszy podział na fragmenty odpowiedzialne odpowiednio za symulacje, metamodelowanie oraz optymalizacje cyklów, bądź procesów przemysłowych.

Główną tabelą zaprojektowanej bazy jest tabela *projects*. To w niej zapisywane są podstawowe informacje dotyczące nazwy czy opisu tworzonego projektu. Spełnia ona również zadanie przypisujące do projektu użytkownika, który go stworzył. Każdy użytkownik dostaje rangę, według której ma dostęp do projektów innych użytkowników aplikacji. Domyślnie,   
w celach testowych, każdy nowo stworzony użytkownik ma rangę administratora, czyli posiada dostęp do wszystkich projektów w bazie danych. Schemat bazy pokazujący relację pomiędzy tabelą *projects*, a tabelą *users* i dalszy sposób przechowywania rang pokazany został na   
rysunku 8.



*Rys.8 Fragment bazy danych prezentujący relacje pomiędzy tabelami projects i users oraz dalszym sposobem przechowywania rang użytkowników [Źródło: Opracowanie własne].*

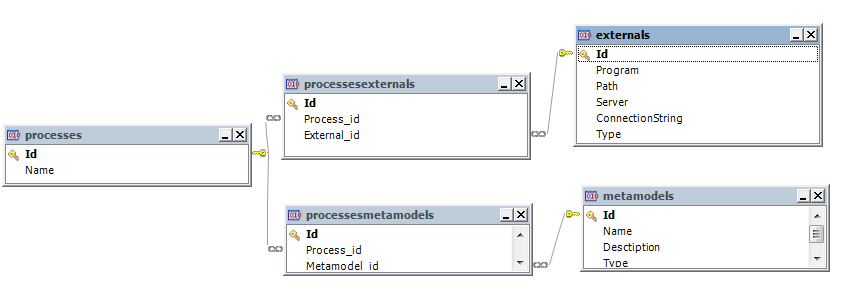
Opisane wyżej zadanie tabeli *projects* można nazwać pobocznym, ponieważ jest ona połączona relacją wiele do wielu z tabelą *processes*. To połączenie gwarantuje sprawne tworzenie cyklu produkcyjnego, a korzystanie z tych tabel odbywa się praktycznie podczas każdego uruchomienia zaimplementowanej aplikacji. Stwierdzenie to nie jest bezpodstawne, co bardzo przejrzyście widać na rysunku 9 przedstawiającym relacje z tabelą *projects*, ale również relacje   
z tabelą *metamodels* oraz *externals*. To w tym miejscu przechowywana jest informacja na temat użytej w projekcie metody dokonywania obliczeń sparametryzowanego procesu (metamodel – symulacja numeryczna).



*Rys.9 Fragment bazy danych pokazujący sposób przechowywania wybranej metody obliczeniowej dla danego procesu w cyklu przemysłowym [Źródło: Opracowanie własne].*

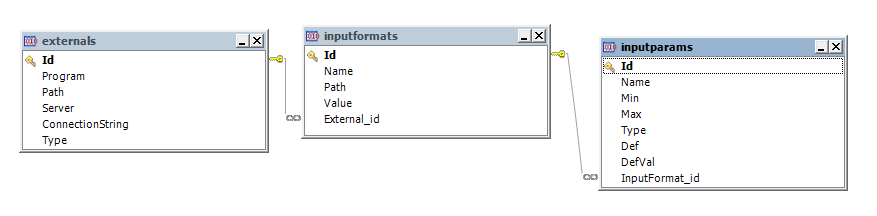
Na przedstawionym rysunku widać również wyraźnie, że baza danych jest przystosowana do akceptacji programów wykonujących obliczenia numeryczne znajdujących się na zewnętrznych serwerach. Pola *server,* czy *ConnectionString* zapewniają dostęp do niezbędnych informacji w celu wykonania skryptu nie tylko po stronie lokalnej aplikacji. Ta funkcjonalność zostanie bliżej opisana w rozdziale 3.3.

Powyższy opis odnosi się do przechowywania już konkretnego wyboru użytkownika dotyczącego przypisanego programu symulującego lub metamodelu przy danym procesie.   
W bazie danych nie może jednak zabraknąć informacji, które sposoby obliczeń (programy zewnętrzne, metamodele) są dostępne dla danego procesu. Stąd relacja przedstawiona na rysunku 10.



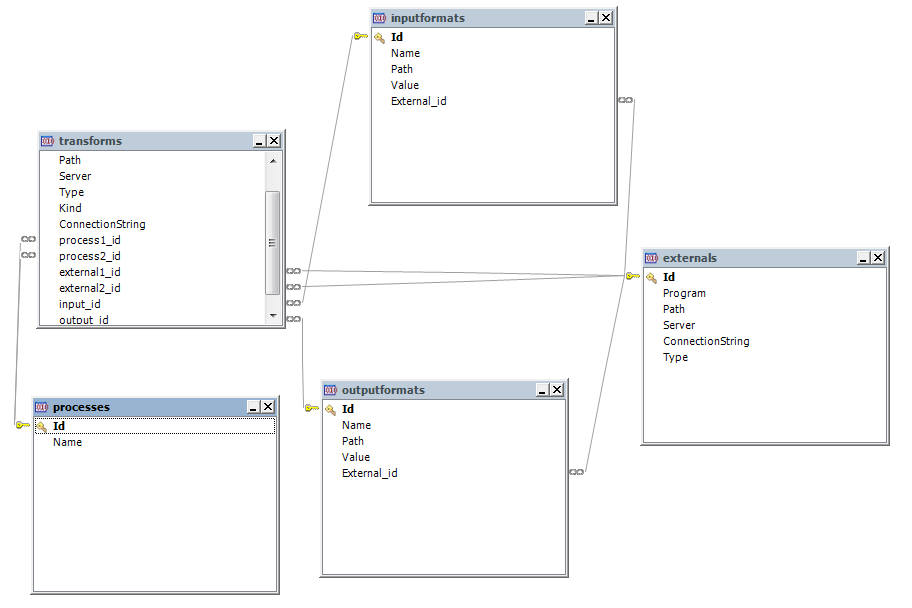
*Rys.10 Fragment bazy danych odpowiedzialny za zapis informacji dotyczących dostępnych sposobów obliczeniowych dla danego procesu [Źródło: Opracowanie własne]..*

Kończąc rozważania dotyczące lokalnej bazy danych warto jeszcze wspomnieć o dwóch fragmentach, bez których nie byłoby możliwe prawidłowe działanie systemu. Jednym z nim jest część bazy odpowiedzialna za przechowywanie parametrów dla danego procesu. Parametry ściśle związane są ze zdefiniowanym szablonem pliku wejściowego, który przypisany jest do konkretnego programu zewnętrznego. Dzięki temu aplikacja potrafi wygenerować odpowiedni, zgodny z obsługiwanym formatem dla danego programu zewnętrznego plik wejściowy, który zawiera parametry zdefiniowane przez zalogowanego do systemu użytkownika. Fragment bazy odpowiedzialny za powyższą funkcjonalność przedstawiony jest poprzez diagram na rysunku 11.



*Rys.11 Fragment bazy danych przechowujący informacje o parametryzacji danego procesu.*

Ostatnim wartym zwrócenia uwagi fragmentem schematu bazy jest miejsce, w którym przechowywane są dane dotyczące transformatorów, czyli aplikacji odpowiedzialnych za interpretowanie wyników z programów zewnętrznych, łączenie procesów w cykl, czy przygotowywanie niezbędnych plików do optymalizacji. Działania, które dostarczają transformatory są bardzo złożone, dlatego będą często wspominane i dokładnie opisywane   
w dalszej pracy. Schemat przechowywania transformatorów jest minimalnie bardziej skomplikowany, ponieważ dodając aplikacje transformującą użytkownik musi określić proces wejściowy oraz wyjściowy, zdefiniować programy zewnętrzne, którymi dane procesy mogą zostać zasymulowane oraz dodatkowo określić pliki wejściowe i wyjściowe dla wybranych programów zewnętrznych. Fragment odpowiadający za przechowywanie opisanych wyżej danych został przedstawiony na rysunku 12.

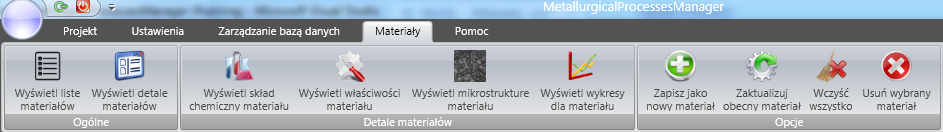


*Rys.12 Sposób, w jaki przechowywane w bazie danych są programy transformujące wraz z niezbędnymi relacjami [Źródło: Opracowanie własne]*

Podsumowując podrozdział 3.1.1 warto zauważyć, że baza danych została zaprojektowana na potrzeby aplikacji w taki sposób, aby łatwo i szybko dostarczać dane. Przedstawione w podrozdziale zostały wyłącznie fragmenty schematu pełnej bazy danych w celu lepszej ilustracji. Istotnym faktem jest, że projekt bazy danych przekłada się na zaprezentowane w dalszej części pracy funkcjonalności oraz interfejs użytkownika.

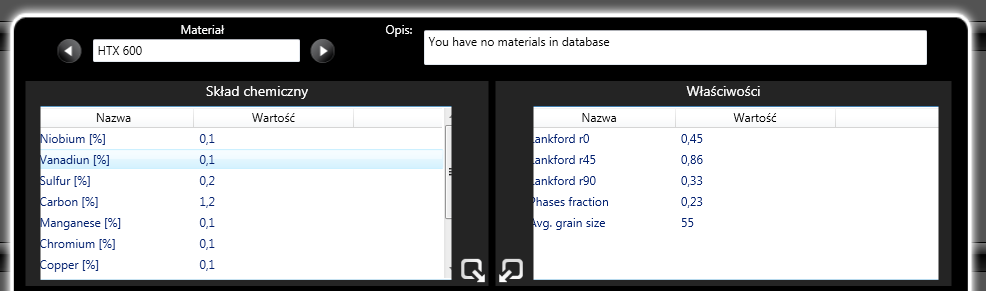
### Wykorzystanie istniejącej bazy materiałowej.

Prezentowana w zaimplementowanym systemie baza materiałowa została zaprojektowana, jak wspomniano we wstępie, na potrzeby różnych projektów realizowanych na Akademii Górniczo-Hutniczej. Biorąc pod uwagę ten fakt nie będzie przedstawiany jej diagram, lecz uwaga zostanie skierowana głównie na funkcjonalności dostępne z poziomu użytkownika. Czytelny interfejs oraz wykorzystanie komponentu Ribbon sprawia, że wprowadzanie zmian czy definiowanie nowych materiałów jest proste i intuicyjne. Główne menu zostało przedstawione na rysunku 13.



*Rys.13 Fragment interfejsu użytkownika odpowiedzialny za zarządzanie bazą materiałową [Źródło: Opracowanie własne]*

Pierwszą udostępnioną użytkownikowi funkcjonalnością jest spis wszystkich modeli materiałowych, które dostępne są podczas tworzenia. W tabeli zamieszczone są krótkie informacje dotyczące nazwy, opisu oraz ilości dostępnych detali dla danego materiału – danych chemicznych, mechanicznych, zdjęć czy wykresów. Przechodząc w edycję danego materiału lub tworzenie nowego użytkownik zostaje zapoznany z czterema panelami, z których każdy prezentuje inne własności – skład chemiczny, zdjęcia mikrostruktur, przypisane dla danego materiału wykresy oraz właściwości mechaniczne (rysunek 14). Przedstawione panele dają ogólny pogląd na prezentowany model materiałowy, ponieważ każdy z nich może zostać powiększony i udostępnić dokładne informacje dotyczące wybranej charakterystyki.



*Rys.14 Panele prezentujące własności chemiczne oraz mechaniczne wybranego materiału [Źródło: Opracowanie własne]*

Przyglądając się detalom w poszczególnych panelach udostępnionych użytkownikowi łatwo można edytować skład chemiczny czy dodać nową własność mechaniczną prezentowanego materiału. Interesującą funkcjonalnością jest również graficzne prezentowanie mikrostruktur oraz tworzenie wykresów. Pozwala to na intuicyjne interpretowanie własności, jakie posiada przeglądany materiał. Ostatnia funkcjonalność pozwala na importowanie danych z popularnych formatów, na podstawie których generowany jest wykres prezentujący te dane. Przykładowy wykres w aplikacji znajduje się na rysunku 15.



*Rys.15 Wykres wygenerowany w interfejsie aplikacji z zaimportowanych danych z pliku .xls*

*[Źródło: Opracowanie własne]*

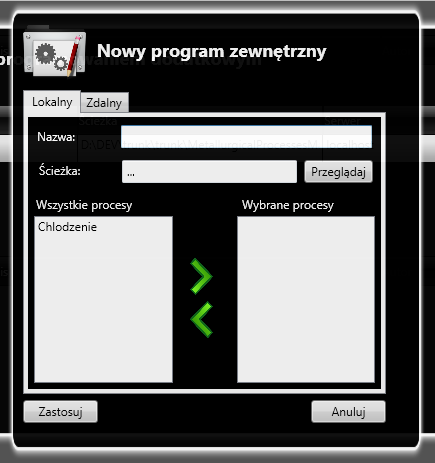
Wszystkie pliki graficzne prezentujące zdjęcia mikrostruktur zamieszczone są na zewnętrznym serwerze ftp. Daje to uniwersalność w dostępie do danych przez wielu użytkowników systemu. Tak skonstruowany interfejs charakteryzuje się możliwością intuicyjnej   
i prostej edycji oraz utworzenia nowego modelu materiałowego, który widoczny jest we wszystkich projektach korzystających z opisywanej bazy danych. Wszystkie materiały zdefiniowane w tej zakładce dostępne są podczas tworzenia nowego projektu, a docelowo ich własności chemiczne oraz mechaniczne mają być brane pod uwagę podczas przeprowadzania obliczeń za pomocą symulacji numerycznych bądź metamodelu, jak i w dokonywanej późniejszej optymalizacji cyklu produkcyjnego.

## Tworzenie i parametryzacja różnych cykli produkcyjnych.

Każdy cykl produkcyjny utworzony przez użytkownika w zaimplementowanym systemie składa się z pojedynczych, wcześniej zdefiniowanych procesów, do których przypisana jest grupa parametrów. Opis samej parametryzacji musi zostać rozpoczęty od przybliżenia struktury szablonów plików wejściowych, która wymusza ilość parametrów dla danego procesu.

Rozpoczynając od definicji samego procesu, nadaniu mu nazwy i skategoryzowaniu go użytkownik przechodzi do określenia programu zewnętrznego, który będzie odpowiedzialny za wykonanie symulacji numerycznej. Program może mieścić się zarówno na serwerze i wtedy to na nim dokonywane są wszelkie obliczenia, a zwracany wynik jest tylko lokalną prezentacją tych obliczeń oraz lokalnie z dysku użytkownika. Podczas definicji lokalnej użytkownik zobowiązany jest określić ścieżkę do wykonywalnego pliku, a także wyszczególnić wcześniej sprecyzowane procesy, które mogą być symulowane przez aktualnie dodawaną aplikację zewnętrzną   
(rysunek 16). Sprawa jest trochę bardziej skomplikowana, gdy oprogramowanie symulacyjne ma być zamieszczone na zdalnym serwerze. W takim przypadku określony musi zostać adres oraz katalog, w którym znajduje się aplikacja. Nie może zabraknąć również dokładnej procedury uruchomienia wraz z parametrami początkowymi. Zaletą wykonywania obliczeń na serwerze jest wykorzystanie pełnej jego mocy obliczeniowej jednocześnie nie obciążając maszyny użytkownika systemu.

Kolejnym krokiem do sparametryzowania procesu jest przypisanie do programu zewnętrznego wspomnianego wcześniej szablonu, który zawiera w sobie wszystkie zmienne parametry, które podczas tworzenia projektu są definiowane przez użytkownika.



*Rys.16 Część interfejsu użytkownika odpowiedzialna za dodawanie programów zewnętrznych dokonujących obliczenia numeryczne [Źródło: Opracowanie własne]*

Na potrzeby realizacji przedstawianej aplikacji przyjęta została konwencja tworzenia   
i zapisywania zmiennych parametrów pokazana na listingu 1. Dodatkowo zaimplementowane zostały schematy rozpoznające pliki wejściowe i pozwalające użytkownikowi na uzupełnienie danych szczegółowych parametru.

…

1000000 number of time steps

{Tp}; initial temperature

{ts}; time step

{augs}; austenite grain size

0 iprint

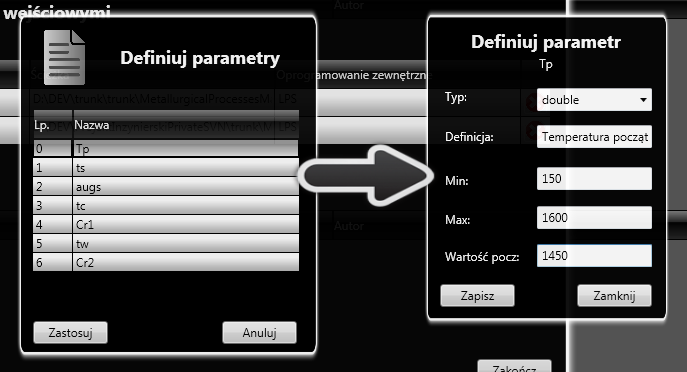
4.5700000000 c-gamma-alfa\_1 (wyraz wolny)

-0.005412000 c-gamma-alfa\_2 (nachylenie)

…

*Listing 1 Część szablonu dla programu zewnętrznego, w którym zapisane są parametry   
Tp, ts, augs.*

Zaprezentowany powyżej plik zostaje zaimportowany w miejsce programu zewnętrznego, do którego jest przypisany. Następnie interpretator przechwytuje zmienne i wyświetla je użytkownikowi, który może zdefiniować detale parametru (rysunek 17). Już na początku określania wartości następuje walidacja czy wpisane zakresy danego parametru lub wartość początkowa jest prawidłowego typu. Istnieje również możliwość uzupełnienia definicji parametru, czyli nazwy intuicyjnej dla potencjalnego użytkownika. Tak przygotowane parametry, ich zakresy i nazwy zapisywane są do bazy danych i wykorzystywane są przy tworzeniu cyklów produkcyjnych, co jest kolejnym krokiem opisywanym w pracy.



*Rys. 17 Fragment interfejsu użytkownika dotyczący określania detali dla danego parametru [Źródło: Opracowanie własne]*

Parametryzacja procesu bądź cyklu produkcyjnego nie może obejść się bez samej definicji i złożenia takiego cyklu podczas tworzenia nowego projektu. Opisany wyżej import szablonu pliku wejściowego rozpoczynał się od tworzenia procesu. To właśnie z tych zdefiniowanych procesów może zostać utworzony cykl, lecz kolejność dodawania nie jest pozostawiona przypadkowi. W tym momencie powraca wspomniany we wcześniejszych rozdziałach temat oprogramowania transformującego.

Transformatory są kluczowym elementem opisywanego w pracy oprogramowania. Cała idea została zaprojektowana i zaimplementowana przez autora. Czym są aplikacje transformujące – transformatory? To oprogramowanie wspierające każdy etap tworzenia projektu, poczynając od samej budowy cyklu produkcyjnego, przez symulacje numeryczne, na optymalizacji kończąc. Ze względu na zastosowanie można wyróżnić kilka rodzajów transformatorów:

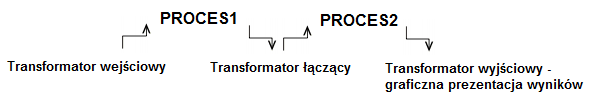
* transformatory wejściowe,
* transformatory wyjściowe,
* transformatory połączeniowe,
* transformatory optymalizacji.

Każdy z wymienionych odpowiada za inną część stworzonego cyklu produkcyjnego. Wykorzystanie oprogramowanie transformującego okazało się niezbędne podczas definiowania nowego projektu i wspomaga cały uniwersalizm aplikacji.

Prawidłowe wytłumaczenie działania aplikacji transformujących nie jest trywialne, więc najpierw warto zwrócić uwagę na same zadania, którymi się zajmują. Transformatory wejściowe przygotowują plik wejściowy, aby był akceptowalny przez przypisany do niego program symulujący. Przykładem takiej aplikacji, która jest wbudowana w zaimplementowany system, jest opisany wyżej interpretator szablonów plików wejściowych. Analizuje on zdefiniowane przez użytkownika parametry procesu, a następnie buduje plik lub system plików niezbędnych do rozpoczęcie obliczeń numerycznych.

Transformatory wyjściowe działają jak postprocesory w ogólnodostępnych aplikacjach rynkowych. Analizują one wyniki danej symulacji numerycznej i prezentują je. Aplikacje   
w intuicyjny, graficzny sposób przedstawiają otrzymane dane korzystając na przykład   
z wykresów, czy struktur. Niezbędne do prawidłowej definicji, podobnie jak w przypadku transformatorów łączących opisanych w dalszej części pracy, jest określenie plików wyjściowych oprogramowania, które jest przypisane do transformatora. Informacja ta jest istotna ze względu na mnogość generowanych plików przez aplikacje symulujące. Transformator musi posiadać wiedzę, który plik wyjściowy jest interpretowany.

Kolejnym, ważnym z punktu widzenia tworzenia cyklu produkcyjnego są transformatory łączące jeden proces z drugim. Analizują one pliki wyjściowe procesu wejściowego i tak manipulują wynikami z pierwszej symulacji, aby zbudowany przez nie plik wejściowy był interpretowalny przez kolejny proces. Dzięki temu istnieje możliwość połączenia kilku, potencjalnie nie związanych ze sobą programów symulujących w cykl, który będzie mógł zostać obliczony od pierwszego procesu, przez kolejne, aż po końcowy, po którym wykorzystany zostanie opisany wyżej transformator wyjściowy. Najłatwiej jednak zrozumieć działanie aplikacji transformujących jest zilustrowanie opisanej wyżej sytuacji, tak jak na rysunku 18.



*Rys. 18 Schemat pokazujący działanie transformatorów podczas tworzenia cyklu produkcyjnego [Źródło: Opracowanie własne]*

Jak można zauważyć aplikacje transformujące mają czynny wpływ na budowę cyklu produkcyjnego z dostępnych procesów. Jeśli użytkownik chce zbudować najprostszy cykl jednoprocesowy na liście procesów możliwych do dodania znajdują się tylko te, które posiadają transformator wejściowy, a co za tym idzie mają narzędzie pozwalające na prawidłowe przygotowanie pliku wejściowego dla tego procesu. Dodanie kolejnych uwarunkowane jest połączeniem procesów przez aplikacje łączącą. Sama symulacja numeryczna, czy optymalizacja nie może zostać wykonana bez dodanego na końcu transformatora wyjściowego, który zinterpretuje wyniki.



*Rys. 19 Fragment interfejsu graficznego, który wykorzystywany jest przy definiowaniu aplikacji transformujących [Źródło: Opracowanie własne]*

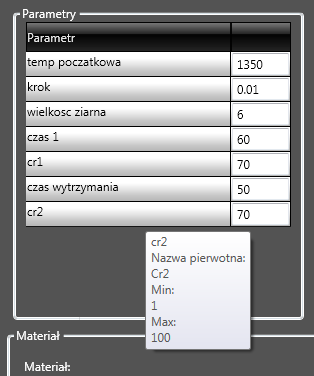
Z pozycji interfejsu użytkownika dodawanie transformatorów nie jest tak skomplikowane jak samo ich działanie. Do dyspozycji oddane są odpowiednie zakładki grupujące aplikacje transformujące według schematu przedstawionego wyżej w pracy. Istnieje również możliwość, podobnie jak w przypadku definicji aplikacji zewnętrznych dokonujących symulacji numerycznych, określenia czy dany transformator znajduje się lokalnie na dysku użytkownika, czy zamieszczony jest po stronie serwerowej. Wtedy niezbędne jest podanie adresu serwera oraz polecenia uruchamiającego aplikacje. Interfejs graficzny odpowiedzialny za dodawanie aplikacji transformujących został pokazany na rysunku 19.

Po tak zorganizowanych przygotowaniach, polegających na definicji kolejno procesu, programu zewnętrznego, pliku wejściowego wraz z parametrami, plików wyjściowych oraz transformatorów można przejść do głównej strony aplikacji – tworzenia nowego projektu. To   
w tej pozycji użytkownik definiuje cykl produkcyjny, który może zostać zasymulowany za pomocą symulacji numerycznych lub metamodelu, a później przeprowadzona może zostać   
w nim optymalizacja cyklu. Wszystko jednak rozpoczyna się od zdefiniowania podstawowych informacji na temat projektu, jakimi są nazwa, opis oraz kategoria, do której projekt jest przypisany. Cykl produkcyjny może zostać zbudowany z procesów dostępnych na warunkach opisanych wyżej, zaś lista procesów aktualizowana jest po każdej modyfikacji dokonanej   
w schemacie cyklu produkcyjnego. Sama budowa cyklu jest intuicyjna i użytkownik mimo wielu ścieżek wyboru przeprowadzany jest przez nią w prosty sposób krok po kroku jak na rysunku 20.



*Rys. 20 Etap dodawania procesów do cyklu produkcyjnego. Pokazany przykład do cykl jednoprocesowy [Źródło: Opracowanie własne]*

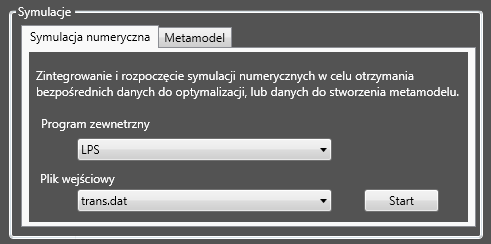
Po stworzeniu cyklu produkcyjnego następuje powrót do jego parametryzacji już na potrzeby danego projektu. Po zaznaczeniu procesu uzupełniana jest tabela zdefiniowanych dla niego, za pomocą pliku wejściowego w sposób opisany wyżej w tym podrozdziale, parametrów. Ustawiane są wartości początkowe dla danych parametrów. Użytkownik ma możliwość zdefiniowania wartości każdej zmiennej w tabeli, lecz w przedziale wcześniej zadeklarowanym. Istnieje również walidacja typu danej zmiennej. Określone podczas opisywania pliku wejściowego zakresy parametrów, czy typy zmiennych mogą zostać w każdej chwili przypomniane przez zaprojektowany interfejs graficzny. Etap definiowania parametrów pokazany został na rysunku 21.



*Rys. 21 Parametryzacja przeprowadzona dla skonstruowanego cyklu jednoprocesowego   
[Źródło: Opracowanie własne]*

## Symulacje i metamodelowanie.

Symulacje oraz moduł metamodelowania to dwa komponenty z trzech odpowiedzialne za najważniejsze funkcje w opisywanym oprogramowaniu. Udostępniają one możliwość dokonywania obliczeń numerycznych w dwojaki sposób – za pomocą opisanego w rozdziale poświęconym architekturze bazy danych oprogramowaniu zewnętrznemu oraz zaimportowanego wcześniej zdefiniowanego metamodelu. Dualizm możliwości przeprowadzenia obliczeń został zaprezentowany na rysunku 22, na którym przedstawiony został fragment interfejsu użytkownika uruchamiający symulacje.



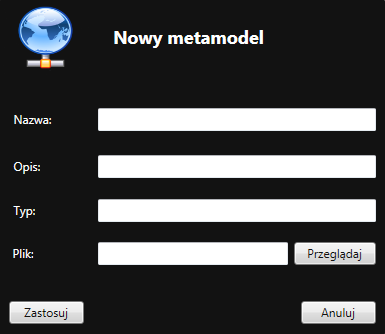
*Rys. 22 Fragment interfejsu użytkownika uruchamiający obliczenia numeryczne za pomocą symulacji numerycznej bądź metamodelu [Źródło: Opracowanie własne]*

Obliczenia numeryczne nie mogą zostać przeprowadzone bez zdefiniowania programu zewnętrznego oraz samej parametryzacji procesu opisanej w rozdziale wyżej. Po wybraniu zdefiniowanego wcześniej szablonu pliku wejściowego podmieniane są za pomocą transformatora wejściowego lub łączącego wartości zmiennych dla danego procesu. Transformator generuje docelowy plik wejściowy, z którym uruchamiany jest program odpowiedzialny za symulację numeryczną danego procesu. Po dokonaniu obliczeń, które mogą być dość długotrwałe, następuje prezentacja wyników za pomocą transformatora wyjściowego, który interpretuje pliki wyjściowe programu zewnętrznego i znajduje się w nich dane filtruje według kryterium ważności dla danego procesu.

Zarówno program zewnętrzny jak i transformatory mogą zostać uruchamiane w sposób lokalny lub zdalny. Patrząc na uruchamianie lokalne sprawa jest dość oczywista, natomiast rozpoczęcie obliczeń na zdalnym serwerze wymagało określenia logiki choćby przechowywania plików wyjściowych z transformera, które są niezbędne do rozpoczęcia symulacji. Problem został rozwiązany poprzez kopiowanie transformerów, a w szczególnych przypadkach plików wejściowych do programów zewnętrznych na serwery sftp, gdzie znajdują się programy do dokonywania obliczeń numerycznych.

Kolejnym sposobem przeprowadzenia symulacji jest rozpoczęcie obliczeń za pomocą metamodelu. Warunkiem koniecznym do dokonania tego rodzaju symulacji jest wcześniejsza definicja metamodelu. Model może zostać zaimportowany za pomocą interfejsu użytkownika przedstawionego na rysunku 23. W praktyce importowana jest architektura sztucznej sieci neuronowej, według której dany model został zbudowany.

Podczas definicji metamodelu, za pomocą interfejsu przedstawionego niezbędne jest określenie jego nazwy, opisu, typu oraz ścieżki do pliku, w którym cała architektura modelu jest zapisana. Zaimportowany metamodel można użyć do dokonania późniejszych obliczeń biorąc pod uwagę, podobnie jak w przypadku symulacji numerycznych, parametry wskazane przez użytkownika i przypisane dla danego procesu.



*Rys. 23 Interfejs odpowiedzialny za import nowego metamodelu [Źródło: Opracowanie własne]*

Biblioteka metamodeli zintegrowana z projektem nie służy wyłącznie do interpretacji zapisanych w pliku tekstowym architektur sztucznych sieci neuronowych. Rozszerzona jest również o możliwość nauczania nowych metamodeli na podstawie danych wejściowych   
i szablonu danych wyjściowych dla danego spectrum.

Główną różnicą pomiędzy dokonywaniem symulacji za pomocą obliczeń numerycznych,   
a predefiniowanych metamodeli jest czas otrzymywanych wyników. Jeśli użytkownikowi przeprowadzającemu symulacje zależy na szybkich obliczeniach powinien wybrać zaimportowany metamodel. Jednak dokładniejsze wyniki przynosi dokonywanie symulacji numerycznych, ponieważ metamodel nie polega na przeprowadzeniu fizycznej symulacji dla danych parametrów wejściowych, lecz na prognozie dokonanej przez nauczoną sztuczną sieć neuronową.

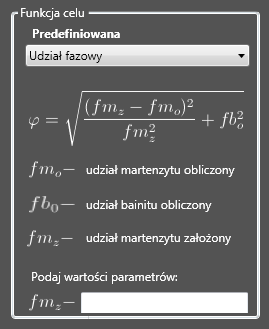
## Optymalizacja.

Przeprowadzenie symulacji numerycznych pozwala uzyskać wyniki dla konkretnego przypadku, czyli dla ściśle określonych parametrów. Co się dzieje, gdy użytkownik chce pewne parametry utrzymać bez zmian, a pozostałymi sterować w taki sposób, aby osiągnąć postawiony cel. Przykładowo może zależeć mu na najniższych kosztach produkcji jednocześnie utrzymując zadowalający poziom jakości produktu. W tym celu z pomocą przychodzi zaimplementowany   
w oprogramowaniu moduł optymalizacji. Jej zadaniem jest maksymalizacja lub minimalizacja zadanej funkcji celu przy określonych zmiennych oraz stałych procesu.

Z punktu widzenia stworzonego projektu sam proces optymalizacji wymaga podobnie jak w przypadku obliczeń numerycznych określenia programu zewnętrznego lub metamodelu, na którym dana optymalizacja będzie działać. Logiczny w takim razie wydaje się fakt, że optymalizacja przeprowadzona za pomocą metamodelu będzie wykonywać się szybciej, ale przyniesie niedokładne wyniki. Użytkownikowi pozostawiany zostaje wybór sposobu w jaki będą odbywać się obliczenia podczas procesu optymalizacji.

Początkowy etap optymalizacji sięga już początku tworzenia cyklu produkcyjnego. W tym miejscu zaznaczane są procesy, między którymi ma zostać przeprowadzona optymalizacja. Cykl może być dzielony na kilka fragmentów, które są brane pod uwagę podczas procesu optymalizacji. Funkcjonalność ta na chwilę obecną rozwoju oprogramowania jest wyłącznie udostępniona na interfejsie użytkownika, zaś nie jest podpięta do biblioteki optymalizacji.

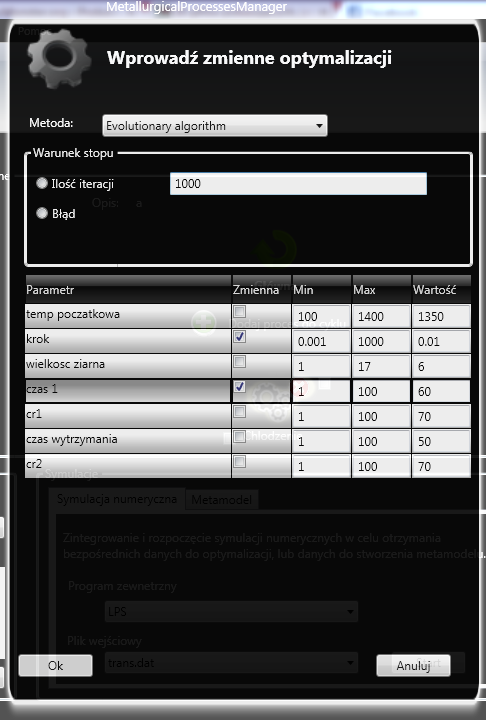
Kolejnym krokiem niezbędnym do rozpoczęcia optymalizacji jest zdefiniowanie funkcji celu, która zostanie poddana maksymalizacji, bądź minimalizacji. Początkowo funkcje te są predefiniowane w systemie, natomiast baza danych przygotowana jest do przechowywania funkcji zdefiniowanych przez użytkownika. Wybór przykładowej funkcji celu pokazany został na rysunku 24.



*Rys. 24 Interfejs odpowiedzialny za wybór predefiniowanej funkcji celu   
[Źródło: Opracowanie własne]*

W pokazanej funkcji celu tylko jedna wartość jest definiowana przez użytkownika. Biblioteka, która jest odpowiedzialna za proces optymalizacji i zintegrowana z systemem ma możliwość przeprowadzania optymalizacji wielokryterialnej, jednak taki proces nie został jeszcze przygotowany. Pozostałe wartości otrzymywane są doświadczalnie za pomocą symulacji numerycznej, bądź metamodelu w każdej iteracji optymalizacji.

Po określeniu funkcji celu pozostaje przejście do najważniejszego okna - do definiowania parametrów samej optymalizacji. Jak wyżej wspomniano niezbędnym jest określenie zmiennych optymalizacji, czyli parametrów, którymi steruje biblioteka, aby otrzymać optymalny wynik, oraz stałych procesu. Na zmienne optymalizacji istnieje również możliwość nałożenia warunków brzegowych, czyli spectrum, w którym dana wartość będzie mogła być zmieniana w celu otrzymania optymalnego wyniku. Natomiast do pozostałych parametrów przypisywana jest stała wartość, niezmienna przez cały proces optymalizacji co zostało pokazane na rysunku 25.



*Rys. 25 Interfejs aplikacji odpowiadający za rozpoczynanie procesu optymalizacji, definiowania zmiennych i stałych oraz wybór metody i warunku stopu [Źródło: Opracowanie własne*]

Zintegrowana z systemem biblioteka optymalizacji udostępnia również możliwość wyboru dostępnej metody samej optymalizacji. Obecnie w systemie proces optymalizacji może być przeprowadzony za pomocą:

* algorytmu ewolucyjnego (evolutionary algorithm),
* algorytm sztucznego układu odpornościowego (AIS – artificial immune system),
* algorytm optymalizacji rojem cząstek (particle swarm optimization),
* specjalna implementacja algorytmu ewolucyjnego dla optymalizacji wielokryterialnej (MOOPTIM),
* algorytm L-BFGS z rodziny algorytmów quasi-Newtonowskich.

Ostatnim krokiem, który użytkownik musi dokonać, aby rozpocząć optymalizacje jest określenie warunku stopu. Aktualnie w oprogramowaniu istnieje możliwość określenia wyłącznie ilości iteracji, które muszą zostać wykonane, aby zakończyć proces optymalizacji. Jednak   
w kolejnej wersji oprogramowania zostanie udostępniona funkcjonalność określenia minimalnego błędu do którego musi dążyć otrzymywana w optymalizacji wartość funkcji celu.

Po rozpoczęciu optymalizacji od strony technicznej uruchamiana jest biblioteka optymalizacji dla wybranej metody ze wskazanymi parametrami oraz narzuconymi warunkami: stopu oraz warunkami brzegowymi. W każdej iteracji uruchamiany zostaje oprócz samego oprogramowania lub modelu, z którego brane są doświadczalne dane, oprogramowanie transformujące dla optymalizacji. Jest to niezbędne, ponieważ plik wyjściowy oprogramowania musi zostać zinterpretowany i przekonwertowany na plik wejściowy dla kolejnej iteracji   
z aktualnymi, zwróconymi przez bibliotekę optymalizacji wartościami parametrów zmiennych. Sposób działania oprogramowania został pokazany na rysunku 26.

Z punktu widzenia użytkownika przebieg optymalizacji jest zauważalny na interfejsie. Parametry, które zostały zaznaczone jako zmienne podkreślone są na kolor czerwony, a ich wartość aktualizowana jest w tabeli z każdą iteracją optymalizacji. Wyświetlana jest również aktualnie obliczona funkcja celu. Finał procesu optymalizacji objawia się przez pokazanie okna dialogowego informującego o finalnej wartości funkcji celu. Zmodyfikowane parametry pozostają na tabeli co umożliwia weryfikacje poprawności optymalizacji poprzez dokonanie symulacji numerycznej bez zmian w nowopowstałych parametrach. Założony cel powinien być osiągnięty lub bliski osiągnięcia.

Rozpoczęcie procesu optymalizacji

Uruchomienie transformatora wejściowego, pobranie wartości parametrów, zmiennych, przygotowanie początkowego pliku wejściowego.

Aktualizacja interfejsu, wypisanie funkcji celu, zakończenie procesu optymalizacji.

*Rys. 26 Schemat przedstawiający sposób działania optymalizacji   
w zaimplementowanych systemie [Źródło: Opracowanie własne]*

# Analiza przypadków użycia.

Po wstępie teoretycznym dostarczającym niezbędnej wiedzy na temat mechanizmów wykorzystanych w pracy oraz zapoznaniu się z interfejsem oraz implementacją systemu, która przybliżyła sens działania i dostępne funkcjonalności, nie pozostaje nic innego jak pokazać wykorzystanie opisywanego oprogramowania na konkretnych przykładach. W poniższym rozdziale rozpatrzone zostaną trzy przypadki, aby pokazać działanie programu w przypadku wywoływania oprogramowania obliczeniowego lokalnie i zdalnie. Przeprowadzona zostanie całkowita optymalizacja cyklu produkcyjnego z procesem chłodzenia szyn, gdzie program wywoływany jest lokalnie oraz symulacja numeryczna procesu walcowania prętów (zdalny program zewnętrzny). Dodatkowo pokazana zostanie optymalizacja procesu chłodzenia prętów.

## Proces walcowania prętów – wykorzystanie zdalnego programu zewnętrznego.

Pierwszym przykładem jest integracja zdalnego programu zewnętrznego, czyli takiego który znajduje się na serwerze i jest uruchamiany z interfejsu użytkownika. Opisywany proces to walcowanie prętów, a parametrami zmiennymi w procesie są promienie opisujące kształt pręta. Transformator wejściowy został napisany w języku Python, natomiast sam szablon pliku wejściowego zapisany jest w formacie xml. Fragment szablonu pliku wejściowego użytego podczas integracji oprogramowania pokazany został na listingu 2.

…

<initialProfileArea>0.001275</initialProfileArea>

<initialTemperature>1200.0</initialTemperature>

<emissivity>0.8</emissivity>

<thermalConductivity>20.0</thermalConductivity>

<meshSeedSize>0.002</meshSeedSize>

<sampleDimensions>

<dimension><key>R1</key><value>{R1};</value></dimension>

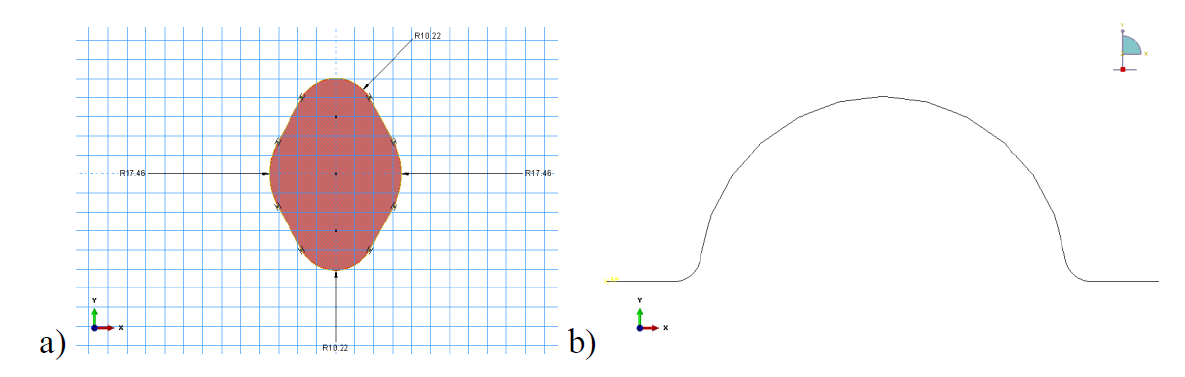
<dimension><key>R2</key><value>{R2};</value></dimension>

</sampleDimensions>

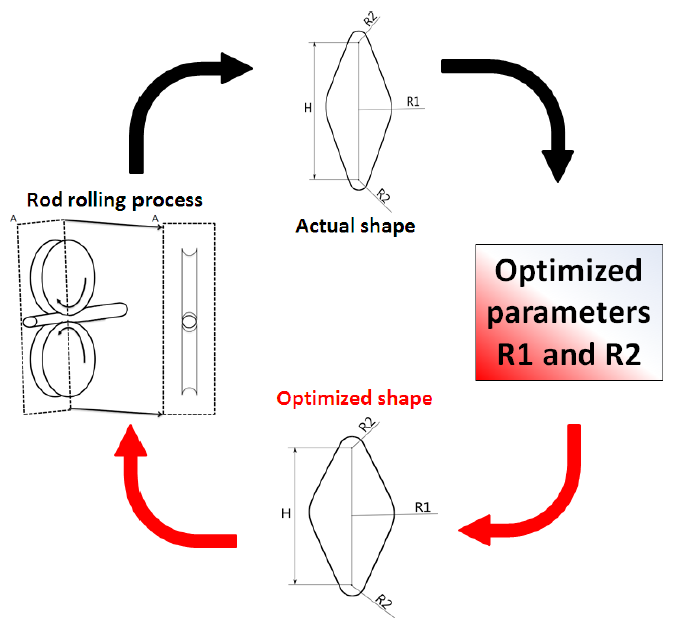
…

*Listing 2 Część pliku wejściowego dla procesu walcowania zawierający zmienne parametry procesu.*

Plik wejściowy podzielony został na 5 sekcji: ogólne ustawienia symulacji, próbka, górne narzędzie, dolne narzędzie oraz materiał próbki. Sekcje te zostały umieszczone w znacznikach xml, które są intuicyjne, przez co plik jest czytelniejszy, a użytkownik z łatwością może dokonać analizy przygotowanej symulacji. Rozwiązywany problem polegał na ustaleniu promieni opisujących kształt pręta po walcowaniu. Napisany w języku Python skrypt ustala początkowy kształt generując w zewnętrznym oprogramowaniu Abaqus narzędzie oraz materiał. Wynik działania skryptu przedstawiony został na rysunku 27, natomiast schemat przeprowadzania procesu optymalizacji kształtu podczas procesu walcowania na rysunku 28.



*Rys. 27 Wynik działania skryptu generującego geometrię w zewnętrznym oprogramowaniu   
a) materiału, b) narzędzia. [Źródło: 4]*



*Rys. 28 Schemat przeprowadzania procesu optymalizacji kształtu podczas procesu walcowania prętów. [Źródło: 4]*

Po rozpoczęciu symulacji wykonywany jest szereg czynności, w które użytkownik   
obsługujący interfejs graficzny nie może ingerować. Wygenerowany zostaje plik wejściowy, który następnie jest przekopiowywany za pomocą podpiętej do projektu biblioteki SharpSSH na serwer, na którym działa program symulacyjny, za pomocą protokołu sftp. Po prawidłowym skopiowaniu wywoływana jest komenda zdefiniowana podczas dodawania programu zewnętrznego oraz transformatora i uruchamiany jest program odpowiedzialny za dokonanie obliczeń. Ostatnim etapem jest uruchomienie transformatora wyjściowego, który ze względów wydajnościowych musi znajdować się w tym samym miejscu co program symulujący. Pliki wyjściowe z komercyjnych aplikacji służących do przeprowadzania obliczeń numerycznych często są dużych rozmiarów, dlatego kopiowanie takiego pliku pochłonęłoby zbyt dużo czasu, zwłaszcza, że często niezbędny do dalszego przeprowadzania procesu optymalizacji jest tylko fragment wyników. Sama ścieżka do plików wyjściowych podawana jest podczas definiowana plików wyjściowych i przypisywana jest do konkretnego programu obliczeniowego. Cały proces uruchamiania symulacji oprogramowania zewnętrznego został przedstawiony za pomocą kodu   
w listingu 3.

if (currentExternal.Type.Equals("remote"))

{

Sftp sftpClient = new Sftp(currentExternal.Server, "\*\*\*", "\*\*\*");

sftpClient.Connect();

sftpClient.Put(outPath = System.IO.Path.GetDirectoryName(

System.Diagnostics.Process.GetCurrentProcess().MainModule.FileName)

+ "\\simulationSetup.xml", "/home/mskiba/abaqus\_scripts/");

SshExec exec = new SshExec(currentExternal.Server, "\*\*\*");

exec.Password = "\*\*\*";

exec.Connect();

string output = exec.RunCommand("cd abaqus\_sripts;");

output = exec.RunCommand("ls;");

Console.WriteLine(output);

output = exec.RunCommand("/packages/abaqus/6.12.3/Commands/abaqus

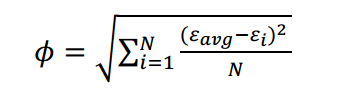
cae nogui=AbaqusController.py;");

Console.WriteLine(output);

}

*Listing 3 Fragment kodu aplikacji odpowiedzialny za uruchamianie programu Abaqus na zdalnym serwerze.*

Predefiniowana funkcja celu optymalizująca kształt podczas opisywanego procesu walcowania prętów ma postać:



*3*

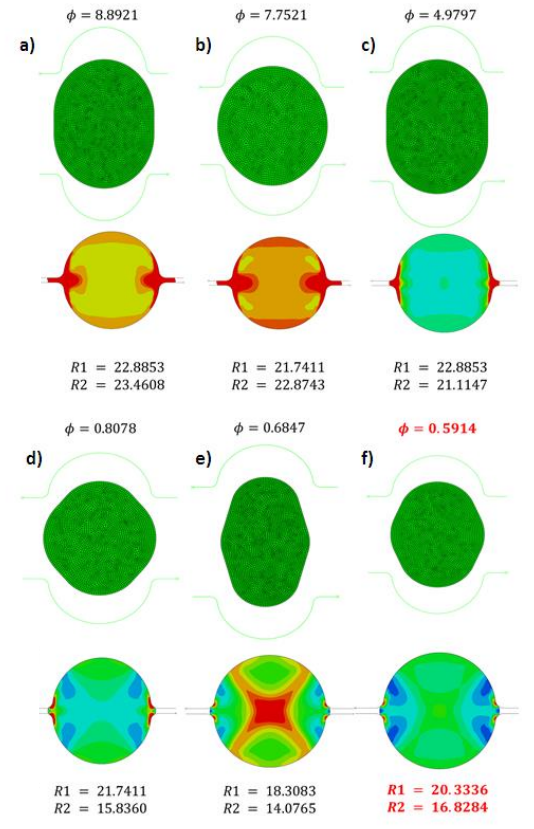
gdzie:

*N* – liczba punktów Gaussa,

*Ɛavg* – średnia wartość odkształcenia,

*Ɛi* – wartość odkształcenia w węźle i.

Wyniki otrzymane za pomocą aplikacji są zbieżne z wynikami zamieszczonymi w artykule [4],   
w którym podobny przypadek został rozwiązany metodą simpleks oraz złotego podziału. Na rysunku 29 przedstawione zostały poszczególne kroki optymalizacji kształtu, przeprowadzone dla 30 iteracji optymalizacji.



*Rys. 29 Poszczególne kroki przeprowadzania procesu optymalizacji kształtu (od początkowego kształtu do finalnej geometrii) [Źródło: 4]*

## Proces chłodzenia prętów – wykorzystanie lokalnego programu zewnętrznego.

Złożenie opisywanego w tym rozdziale procesu zostało zapoczątkowane podobnie jak   
w poprzednim przypadku od zdefiniowania lokalnego oprogramowania zewnętrznego oraz zaimportowania przygotowanego szablonu pliku wejściowego, z tą różnicą, że zarówno oprogramowanie jak i plik wejściowy dostępne są lokalnie na komputerze użytkownika. Po zinterpretowaniu pliku system określa parametry wejściowe opisywanego procesu:

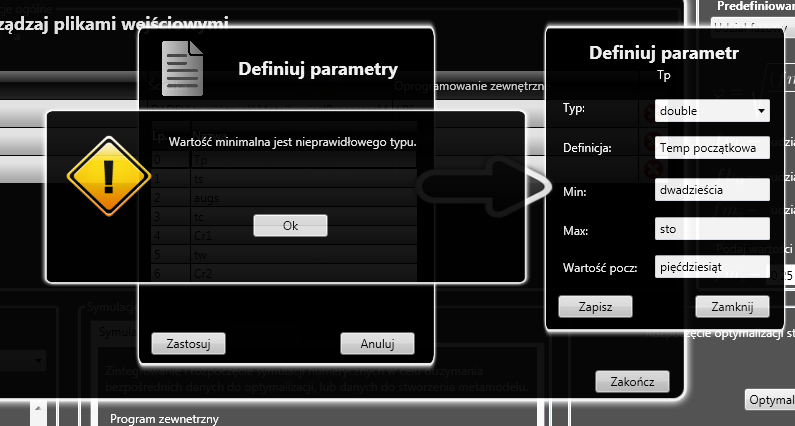
* temperatura początkowa,
* krok czasowy,
* wielkość ziarna austenitu,
* czas dla drugiego interwału chłodzenia,
* szybkość chłodzenia 1,
* czas dla trzeciego interwału chłodzenia, czas wytrzymania,
* szybkość chłodzenia 2.

Dla każdego z parametrów przypisana została definicja użytkowa, bardziej intuicyjna dla użytkownika oprogramowania, typ oraz przedziały dla których dany parametr jest prawidłowy. Konfiguracja parametrów przedstawiona została w tabeli 2. Wszystko to zostało zamieszczone   
w intuicyjnym interfejsie pokazanym na rysunku 17.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Konfiguracja parametrów procesu chłodzenia** | | | | | |
| Parametr | Nazwa użytkowa | Typ | Wartość minimalna | Wartość maksymalna | Wartość domyślna |
| Tp | Temperatura początkowa | double | 500.0 | 1500.0 | 1350.0 |
| ts | Krok czasowy | double | 0.0001 | 1 | 0.01 |
| augs | Wielkość ziarna austenitu | double | 1.0 | 10.0 | 6.0 |
| tc | Czas dla 2-giego interwału | double | 20.0 | 100.0 | 60.0 |
| Cr1 | Szybkość chłodzenia1 | double | 20.0 | 100.0 | 70.0 |
| tw | Czas wytrzymania | double | 20.0 | 100.0 | 50.0 |
| Cr2 | Szybkość chłodzenia 2 | double | 20.0 | 100.0 | 70.0 |

*Tab.2 Konfiguracja parametrów dla procesu chłodzenia [Źródło: Opracowanie własne]*

Podczas definicji parametrów nie ma możliwości wpisania nieprawidłowego parametru, który nie będzie prawidłowo zinterpretowany przez transformator wejściowy. Zaimplementowana została walidacja typów zaprezentowana na rysunku 30.



*Rys. 30 Negatywny wynik walidacji definiowanych wartości parametrów   
[Źródło: Opracowanie własne].*

Następnymi krokami do zbudowania procesu jest stworzenie i zdefiniowanie transformatorów niezbędnych do generowania nowego projektu i uruchomienia najważniejszych modułów obliczeń numerycznych i symulacji. Implementacja transformatorów nie jest skomplikowana i opiera się głównie na interpretowaniu plików wejściowych i wyjściowych programu zewnętrznego. Zaletą jest fakt, że do implementacji może zostać użyty dowolny język programowania, w tym przypadku c#.

Po tak przygotowanej bazie można przejść do definicji samego projektu, gdzie po podaniu nazwy i wyboru kategorii przypisujemy nasz proces – chłodzenie prętów. Automatycznie uzupełniana jest tabela z parametrami oraz z domyślnymi wartościami przedstawionymi   
w tabeli 2. Przypisywanie materiału w tym przypadku nie ma wpływu na wyniki symulacji, ponieważ parametry materiału nie są przekazywane do symulacji tylko są zapisane na stałe   
w pliku wykonawczym programu zewnętrznego. Podobnie jak z tabelą parametrów uzupełniany jest automatycznie domyślny (pierwszy przypisany do danego procesu) program symulujący oraz plik wejściowy.

Kolejnym krokiem jest rozpoczęcie obliczeń numerycznych. Przyjmując, że w obecnym teście parametry nie będą modyfikowane symulacja zostanie rozpoczęta z wartościami domyślnymi. Po wykonaniu obliczeń uruchamiający się transformator wyjściowy, podobnie jak postprocesory w komercyjnych aplikacjach prezentuje nam wyniki zamieszczone w tabeli 3. Jest to udział objętościowy w danej strukturze i obejmuje:

* udział objętościowy ferrytu,
* udział objętościowy perlitu,
* udział objętościowy bainitu,
* udział objętościowy martenzytu.

|  |  |
| --- | --- |
| Nazwa parametru wyjściowego | Obliczona wartość parametru |
| udział objętościowy ferrytu | 0.172 |
| udział objętościowy perlitu | 0 |
| udział objętościowy bainitu | 0.075 |
| udział objętościowy martenzytu | 0.753 |

*Tab. 3 Wyniki przeprowadzonej symulacji dla domyślnych parametrów wejściowych  
[Źródło: Opracowanie własne]*

Zakładając, że celem jest otrzymanie udziału objętościowego martenzytu utrzymującego się na poziomie 25% następuje modyfikacja parametrów i ponowne przeprowadzenie symulacji numerycznej. Zmieniając parametry wejściowe na te zamieszczone w tabeli 4 po zinterpretowaniu ich przez transformator wyjściowy otrzymaliśmy wyniki zaprezentowane   
w tabeli 5.

|  |  |
| --- | --- |
| Nazwa parametru | Ustalona wartość |
| Temperatura początkowa | 1500.0 |
| Krok czasowy | 0.01 |
| Wielkość ziarna austenitu | 3.0 |
| Czas dla 2-giego interwału | 35.0 |
| Szybkość chłodzenia1 | 70.0 |
| Czas wytrzymania | 40.0 |
| Szybkość chłodzenia 2 | 50.0 |

*Tab. 4 Wartości parametrów dla drugiego przeprowadzanego testu  
[Źródło: Opracowanie własne]*

|  |  |
| --- | --- |
| Udział objętościowy | Otrzymana wartość |
| Ferryt | 0.234 |
| Perlit | 0 |
| Bainit | 0.07 |
| Martenzyt | 0.696 |

*Tab. 5 Wyniki symulacji numerycznej przeprowadzonej dla parametrów zawartych w tabeli 4  
[Źródło: Opracowanie własne]*

Jak łatwo zauważyć wyniki otrzymane w drugim teście są bliższe uzyskania założonej wartości udziału objętościowego martenzytu. Niestety metoda prób i błędów w tym przypadku nie może się sprawdzić, ponieważ nie wiadomo, który parametr ma największy wpływ na udział fazowy martenzytu. Ponadto, może wystąpić sytuacja, że to właśnie ten parametr będzie zdefiniowany jako stały. Dlatego wykorzystywana jest zintegrowana biblioteka optymalizacji.

W miejscu odpowiedzialnym za wybór funkcji celu, ustawiona jest predefiniowana funkcja celu odpowiadająca za minimalizację udziału fazowego bainitu i osiągnięcie założonego progu martenzytu, która ma następującą postać:

*4*

gdzie:

ƒmb – obliczony udział martenzytu,

ƒbo – obliczony udział bainitu,

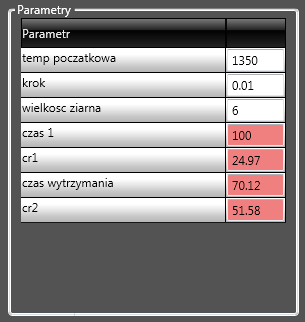
ƒmz – udział martenzytu założony przez użytkownika.

Kolejnym krokiem, po zdefiniowaniu założonego udziału martenzytu jest zdefiniowanie parametrów samej optymalizacji, gdzie istnieje możliwość wyboru metody, warunku stopu oraz określenia zmiennych i stałych procesu optymalizacji. Jako metodę wybieramy algorytm ewolucyjny, natomiast ilość iteracji wynosi 100. Dla wykonania testu jako zmienne potraktowane zostały czasy chłodzenia oraz ich szybkość. Pełna lista parametrów stałych i zmiennych oraz ich przedziały znajduje się w tabeli 6.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nazwa użytkowa | Zmienna | Wartość minimalna | Wartość maksymalna | Wartość ustalona |
| Temperatura początkowa | nie | - | - | 1350.0 |
| Krok czasowy | nie | - | - | 0.01 |
| Wielkość ziarna austenitu | nie | - | - | 6.0 |
| Czas dla 2-giego interwału | tak | 20.0 | 100.0 | - |
| Szybkość chłodzenia1 | tak | 20.0 | 100.0 | - |
| Czas wytrzymania | tak | 20.0 | 100.0 | - |
| Szybkość chłodzenia 2 | tak | 20.0 | 100.0 | - |

*Tab. 6 Wartości zmiennych oraz stałych zdefiniowanych podczas optymalizacji  
[Źródło: Opracowanie własne]*

Po tak skonfigurowanym procesie optymalizacji zmodyfikowane parametry przyjmują wartości zamieszczone na rysunku 31. W celu sprawdzenia czy optymalizacja przebiegła poprawnie dokonano ponownej symulacji numerycznej już ze zmodyfikowanymi parametrami. Wyniki tej symulacji przedstawiono w tabeli 7.



*Rys.31 Wyniki optymalizacji polegające na modyfikacji parametrów dla danego procesu w taki sposób, aby zminimalizowana była funkcja celu [Źródło: Opracowanie własne]*

|  |  |
| --- | --- |
| Udział objętościowy | Otrzymana wartość |
| Ferryt | 0.615 |
| Perlit | 0 |
| Bainit | 0.135 |
| Martenzyt | 0.25 |

*Tab. 7 Wyniki symulacji numerycznej po modyfikacji parametrów przez bibliotekę optymalizacji [Źródło: Opracowanie własne]*

Wyniki otrzymane z transformatora wyjściowego, potwierdzają poprawne przeprowadzenie procesu optymalizacji. Założona przy funkcji celu wartość jest taka sama jak po przeprowadzeniu symulacji numerycznej. Oznacza to, że zintegrowana biblioteka do obsługi procesu optymalizacji działa poprawnie.

## Cykl chłodzenia szyn.

Ostatnim przypadkiem użycia jaki zostanie przedstawiony w pracy jest optymalizacja cyklu chłodzenia szyn. Schemat przedstawiający ideę opisywanego cyklu został przedstawiony na rysunku 32. Parametrami sterującymi procesem były: czas zanurzenia (tw), temperatura otoczenia (ta), głębokość zanurzenia (imm) oraz współczynnik wymiany ciepła (htc). Testy zostały wykonane dla 5 wariantów, w których optymalizacji poddawane są różne parametry cyklu oraz zmienna jest ilość zanurzeń szyny.

a) b)

*Rys.32 Schematyczna ilustracja jednego etapu cyklu chłodzenia szyn – chłodzenie główki szyny, a) główka zanurzona w roztworze, b) główka powyżej poziomu roztworu.  
[Źródło: 5]*

Głównym zadaniem przeprowadzonej optymalizacji była minimalizacja funkcji celu przedstawionej na wzorze 5.

  *5*

gdzie:

*HVave* – przeciętna twardość główki szyny,

*ngp* – liczba punktów Gaussa w główce szyny,

*S0* – odległość międzypłytkowa w perlicie,

*Fb* – średni udział objętościowy bainitu.

Zmienność parametrów założona podczas optymalizacji została przedstawiona w tabeli 8.

|  |  |  |
| --- | --- | --- |
| Nazwa użytkowa | Wartość minimalna | Wartość maksymalna |
| Czas zanurzenia 2n  [Tw2n] | 0 | 30 |
| Czas zanurzenia 2n + 1  [Tw2n+1] | 0 | 60 |
| Czas zanurzenia n  [Twn] | 0 | 1000 |
| Głębokość zanurzenia  [Imm] | 0 | 32 |
| Temperatura otoczenia  [Ta] | 15 | 60 |
| Współczynnik wymiany ciepła  [HTC] | -500 | 500 |

gdzie:

n – ilość zanurzeń.

*Tab. 8 Zakresy zmienności parametrów podczas przeprowadzania procesu optymalizacji  
[Źródło: Opracowanie własne]*

Proces optymalizacji przeprowadzony został dla dwóch metod – algorytmu ewolucyjnego (EA) oraz optymalizacji rojem cząstek (PSO). Wyniki otrzymane tymi metodami dla pięciu różnych wariantów przedstawione zostały w tabeli 9. Silnik numeryczny obliczający poszczególne iteracje dla zdefiniowanych parametrów został szerzej opisany w pracy 6, 7.

*Wariant 1 (liczba zanurzeń n = 4).*

|  |  |  |
| --- | --- | --- |
| Nazwa parametru poddanego optymalizacji | Ustalone parametry | |
| EA | PSO |
| Tw1 | 35,21 | 59,51 |
| Tw2 | 11,06 | 6,94 |
| Tw3 | 33,81 | 49,41 |
| Tw4 | 495,15 | 473,16 |

*Wariant 2 (liczba zanurzeń n = 6).*

|  |  |  |
| --- | --- | --- |
| Nazwa parametru poddanego optymalizacji | Ustalone parametry | |
| EA | PSO |
| Tw1 | 46,32 | 47,77 |
| Tw2 | 6,31 | 0,76 |
| Tw3 | 44,49 | 5,99 |
| Tw4 | 24,94 | 5,26 |
| Tw5 | 16,2 | 39,96 |
| Tw6 | 291,7 | 645,68 |

*Wariant 3 (liczba zanurzeń n = 4, zmienna temperatura otoczenia).*

|  |  |  |
| --- | --- | --- |
| Nazwa parametru poddanego optymalizacji | Ustalone parametry | |
| EA | PSO |
| Tw1 | 51,42 | 24,86 |
| Tw2 | 15,95 | 25,37 |
| Tw3 | 5,05 | 4,12 |
| Tw4 | 35,46 | 50,32 |
| Ta2 | 227,39 | 685,34 |
| Ta4 | 49,37 | 16,53 |

*Wariant 4 (liczba zanurzeń n = 4, zmienny współczynnik htc, stały czas ostatniego zanurzenia).*

|  |  |  |
| --- | --- | --- |
| Nazwa parametru poddanego optymalizacji | Ustalone parametry | |
| EA | PSO |
| Tw1 | 53,2 | 57,98 |
| Tw2 | 8,2 | 10,59 |
| Tw3 | 22,33 | 30,9 |
| HTC | -159,05 | -191,37 |

*Wariant 5 (liczba zanurzeń n = 4, zmienny współczynnik htc, stały czas ostatniego zanurzenia, zmienna głębokość zanurzenia).*

|  |  |  |
| --- | --- | --- |
| Nazwa parametru poddanego optymalizacji | Ustalone parametry | |
| EA | PSO |
| Tw1 | 37,46 | 46,11 |
| Tw2 | 1,65 | 4,87 |
| Tw3 | 20,54 | 32,24 |
| HTC | -38,17 | 278,01 |
| Imm1 | 21,97 | 11,9 |
| Imm2 | 8,22 | 17,48 |

*Tab. 9 Wyniki otrzymane po przeprowadzeniu procesu optymalizacji  
[Źródło: Opracowanie własne]*

Jak można zauważyć wyniki otrzymane za pomocą algorytmu ewolucyjnego oraz optymalizacji rojem cząstek są stosunkowo zbieżne, a powstałe różnice wynikają z samej charakterystyki wybranej metody. Warto wspomnieć, że dla ustalonych parametrów przy różnych metodach funkcja celu została prawidłowo zminimalizowana, a potwierdzone zostało to wynikami symulacji numerycznych dla otrzymanych parametrów i przyjętej funkcji celu. Dodatkowo przeprowadzone zostało badanie czasu obliczeń zależne od ilości zmiennych parametrów oraz wybranej metody dla 10 kroków iteracyjnych. Wyniki przedstawione zostały w tabeli 10, natomiast graficznie przedstawione zostały za pomocą wykresu na rysunku 33. Analizując otrzymane czasy można wywnioskować, że algorytm ewolucyjny jest wolniejszym algorytmem, niż optymalizacja roju cząstek. Dodatkowo czas obliczeń w obu metodach rośnie proporcjonalnie do ilości parametrów poddanych optymalizacji. W przedstawionych przypadkach skuteczność wybranych metod była jednakowa, ponieważ wartość funkcji celu została prawidłowo zminimalizowana.

|  |  |  |
| --- | --- | --- |
| Ilość parametrów | Czas EA [min] | Czas PSO [min] |
| 4 (wariant 1) | 182 | 150 |
| 4 (wariant 4) | 198 | 167 |
| 6 (wariant 2) | 240 | 221 |
| 6 (wariant 3) | 234 | 210 |
| 6 (wariant 5) | 250 | 215 |

*Tab. 10 Czasy dokonywanych obliczeń z podziałem na wybraną metodę i ilość zmiennych parametrów [Źródło: Opracowanie własne]*

*Rys.33 Wykres prezentujący czas obliczeń [min] zależny od wybranego algorytmu i parametrów poddanych optymalizacji [Źródło: Opracowanie własne]*

# Podsumowanie i kierunek rozwoju.

Korzystając z przedstawionego w pracy oprogramowania użytkownik może utworzyć projekt, w którym skonfiguruje wszystkie parametry niezbędne do rozpoczęcia optymalizacji procesów metalurgicznych. W zaimplementowanym systemie istnieje możliwość wyboru parametrów procesu samodzielnie wcześniej utworzonego oraz predefiniowanej funkcji celu. Istnieje szereg funkcjonalności wspomagających przeprowadzenie procesu optymalizacji   
w szybki i skuteczny sposób.

W ręce użytkownika oddane zostaje oprogramowanie, które umożliwia przeprowadzanie obliczeń za pomocą symulacji numerycznych lub metamodeli i wykorzystywanie ich do optymalizacji procesów i cykli produkcyjnych. Jak każde oprogramowanie, to przedstawione   
w pracy posiada dużą ilość funkcjonalności, które mogą zostać rozszerzone oraz udoskonalone   
w późniejszym czasie. Jednym z głównych kierunków rozwoju jest dokonywanie obliczeń numerycznych oraz samej optymalizacji w sposób rozproszony korzystając z technologii gridowej. Umożliwi to szybsze wykonywanie symulacji numerycznych, czy optymalizacji oraz równoległe ich prowadzenie na pewnych etapach w całych procesie. Obrazuje to jak duży potencjał ma przedstawiana aplikacja, a prezentowana praca nie jest zakończeniem działań nad oprogramowaniem, tylko sprawozdaniem z pewnej ich części. Zagadnienie optymalizacji jest bardzo ważne w obecnym przemyśle, gdzie najistotniejszym założeniem jest zmniejszenie kosztów produkcji, przy jednoczesnym zachowaniu założonej wytrzymałości produktu.

Podsumowując napisana praca przedstawia nam oprogramowanie zaimplementowane do przeprowadzania optymalizacji cyklów produkcyjnych, a pokazane przypadki użycia gwarantują prawidłowe działanie biblioteki optymalizacji, co sprawia, że przedstawiana aplikacja jest zgodna z tematem i postawionymi jej założeniami.

# Bibliografia

1. Kusiak J., Danielewska-Tułecka A., Oprocha P.: Optymalizacja, Wydawnictwo Naukowe PWN, Warszawa 2009, 1-21.
2. Li Y.F., Ng S.H., Xie M., Goh T.N.: A systematic comparison of metamodeling techniques for simulation optimization in Decision Support Systems, Applied Soft Computing 10, Elsevier 2010, 1257-1273.
3. Can B., Heavey C.: A comparison of genetic programming and artificial neural networks in metamodeling of discrete-event simulation models, Computers & Operations Research 39, Elsevier 2012, 424-436.
4. Legwand A., Perzyński K., Madej Ł., Pietrzyk M., Approach for an automatic optimisation of production chain as a tool for intelligent manufacturing in metal forming, Computer Methods in Materials Science, 2013, w druku.
5. Kuziak R., Zygmunt T., A new method of rail head hardening of standard-gauge rails for improved wear and damage resistance, Steel Research International, 84, 2013, 13-19.
6. Pietrzyk M., Kuziak R., Numerical simulation of controlled cooling of rails as a tool for optimal design of this process, Computer Methods in Materials Science, 12, 2012, 233-243.
7. Pietrzyk M., Kuziak R., Zygmunt T., Analiza numeryczna możliwości produkcji szyn ze stali perlitycznych o podwyższonej odporności na zużycie ścierne, Mechanik, 86, 2013, 272-276.
8. Schwefel H.: Numerical optimization of computer models, John Wiley & Sons, Chinchester 1981, 292-324.
9. Madej Ł., Pietrzyk M., Metoda analizy wieloskalowej w zastosowaniach inżynierskich, Mechanik, 7, 2007, 652-665.
10. Hamming R. W., Numerical Methods for Scientists and Engineers, Dover 1973, 23-45.