Jerzy ZGRAJA

Politechniki Łódzkiej, Instytut Informatyki Stosowanej

doi:10.15199/48.2017.01.05

Obliczanie indukcyjnych układów grzejnych z wykorzystaniem interfejsu API komercyjnego pakietu Flux®

Streszczenie. *Obliczenia symulacyjne są współcześnie powszechnie wykorzystywane przy projektowaniu indukcyjnych układów grzejnych oraz technologii nagrzewania indukcyjnego. W przypadku złożonych programów do numerycznych obliczeń polowych dążenie do uzyskiwania dużych dokładności obliczeń, a jednocześnie łatwej obsługi czy posiadanie nietypowych funkcjonalności wskazuje na użyteczność personalizacji komercyjnych pakietów obliczeniowych. W pracy zaprezentowane takie działania dla komercyjnego pakietu Flux® do obliczeń pól elektromagnetycznych i cieplnych. Pokazano możliwość wykorzystania jego interfejsu API do zbudowania własnych programów w języku Java. Zaprezentowano różne programy, który przy prostym, łatwym do zrealizowania i wykorzystania interfejsie użytkownika, pozwalają zarówno na obliczanie podstawowych zależności elektrycznych układu wzbudnik-wsad, jak i realizację sprzężonych obliczeń elektromagnetyczno cieplnych czy nawet na rozszerzenie funkcjonalności pakietu komercyjnego o interakcyjną współpracę z falownikami zasilającymi.*

Abstract. *Computer simulations are nowadays widely used in the design of induction heating systems and induction heating technology. Personalization of complex commercial packages allows you to get, from one side, to high accuracy and speed of calculation, and on the other user-friendly interface, and the not typical functionalities. The paper presents such activities for a commercial package FLUX for electromagnetic and thermal calculations. Shown the ability to use the API to build your own Java programs which collaborating with this commercial package.* ***(Calculation of induction heating systems using API of Flux® package).***

Słowa kluczowe*:* nagrzewanie indukcyjne, API.
Keywords: induction heating, API.

Wprowadzenie

Obliczenia czy analizy indukcyjnych układów grzejnych zarówno do celów projektowania konstrukcji wzbudnika, jak i doboru technologii nagrzewania są zwykle realizowane drogą numerycznych symulacji komputerowych [1-5]. Tego typu podejście jest relatywnie szybkie i jednocześnie może być obecnie prowadzone w sposób zapewniający wysoką dokładność otrzymywanych wyników.

Dokładność i poprawność symulacji komputerowych jest w sposób oczywisty zależna od:

- jakości stosowanego oprogramowania komputerowego,
- umiejętności właściwego wykorzystania oprogramowania w tym szczególnie poprzez przyjęcie odpowiednich założeń upraszczających i zbudowania modelu symulacyjnego.

Przy podejmowaniu decyzji o użyciu programu komercyjnego, lub też budowania własnego programu, poza względami ekonomicznymi, bardzo duży wpływ na podjęcie decyzji ma z jednej strony wygoda w posługiwaniu się programem, a z drugiej strony zaufanie do poprawności jego pracy. W przypadku programów komercyjnych, posiadających szeroką rzeszę użytkowników można oczekiwać większej niezawodność ich pracy oraz mniejszego ryzyka otrzymania wyników dalece odbiegających od poprawnych. Podobnych efektów pracy, można niekiedy oczekiwać również w przypadku programów, które nie są typowymi programami komercyjnymi, ale szeroko rozpowszechnionymi np. na licencji open-source. W przypadku pisania własnych programów, należy zawsze liczyć się z ryzykiem pojawienia się poważnych błędów obliczeniowych przy analizie nowych, nie stosowanych poprzednio przypadków obliczeniowych lub też liczyć się z długą i kosztowną procedurą testowania oprogramowania. Niewątpliwą zaletą takich programów jest natomiast możliwość lepszego dopasowania oprogramowania do własnych potrzeb, szczególnie jeśli oczekiwania obliczeniowe dotyczą ściśle zdefiniowanego asortymentu indukcyjnych układów grzejnych, co ma szczególnie często miejsce z zakładach przemysłowych.

Rozwiązania przedstawionego problemu dla zagadnień nagrzewania indukcyjnego można poszukiwać w budowaniu własnego programu zawierającego komercyjny program do obliczeń polowych jako program osadzony. W takim przypadku istnieje możliwość:

- dopasowania interfejsu użytkownika do jego własnych preferencji,
- rozbudowania oprogramowania o wymagane funkcjonalności, których nie ma w programie komercyjnym,

a jednocześnie zdanie zbudowania pakietu oprogramowania jest relatywnie proste.

W programach komercyjnych istnieją zwykle pewne możliwości współpracy [6,7] z oprogramowaniem zewnętrznym. Możliwości takie istnieją również w omawianym programie komercyjnym Flux francuskiej firmy CEDRAT, który może współpracować z programami zarówno poprzez wymianę danych poprzez dysk, jak i poprzez wykorzystanie interfejsu programowania aplikacji API (***A**pplication **P**rogramming **I**nterface*).

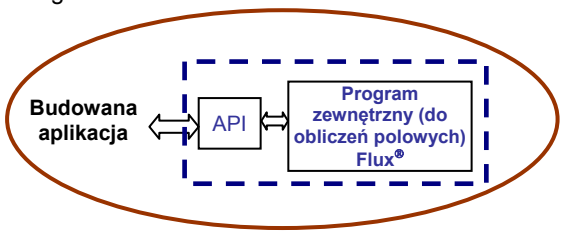
Osadzanie programu zewnętrznego w budowanej aplikacji

Przy wykorzystywaniu zamkniętego (skompilowanego), zewnętrznego programu we własnej aplikacji podstawowym zagadnieniem do rozwiązania pozostaje kwestia współpracy, wymiany informacji, między aplikacjami. Sprowadza się do zazwyczaj do wymiany informacji przez dysk, lub skorzystanie z możliwości API zewnętrznego oprogramowania. Programy komercyjne mają zwykle opracowane i udostępnione interfejsy API, podobnie jest w przypadku komercyjnego pakietu Flux®, który posiada interfejs do współpracy z programami napisanymi w różnych językach programowania m.in. w języku Matlab, Python, Java czy C [6].

Jak pokazuje Rys.1, przy wykorzystaniu interfejsu API funkcję nadrzędną w pakiecie współpracujących programów przejmuje program współpracujący z interfejsem API, czyli program własny użytkownika. Program komercyjny (Flux) staje się programem „osadzonym" w tym programie, a narzędzia programowanego interfejsu umożliwiają komunikację i sterowanie jego pracą.

W przypadku programów w języku Java, API programu Flux v.12 został sprowadzony do opracowania zestawu klas zapisanych w kilku paczkach (archiwach): fluxmp.jar, cedserver.jar, coreboot.jar, corebus.jar, jutils.jar, rsicore.jar,

CoreCommon.jar, CedUtils.jar, CssUtils.jar, CssService.jar. Wymienione wyżej paczki są dostarczane wraz z oprogramowaniem Flux i mogą być wykorzystywane przez piszącego program nadrzędny w języku Java. Do efektywnego korzystania z opracowanych klas niezbędne są informacje objaśniające sposób ich działania. W dokumentacji oprogramowania brak jest takiego opisu, producent załączyła jedynie bardzo prosty przykład w języku Java sprowadzający się zasadniczo do uruchomienia programu i wyświetleniu komunikatu zwrotnego.



Rys.1. Wykorzystująca API współpraca aplikacji użytkownika z programem zewnętrznym.

Ten prosty przykład pozwala na zorientowanie się, że podstawową klasą przeznaczoną dla użytkowników jest klasa FMP umieszczona w katalogu rsi/fluxmp paczki fluxmp.jar. Metoda FMP.FMP_init(Debag) pozwala na zainicjowanie pracy API w opcji z debagowaniem Debag=1 lub bez Debag=0, a FMP.FMP_startLocalServer() umożliwia programowi nadrzędnemu na uruchomienie pracy serwera w postaci programu Flux. Uruchomienie tego serwera wymaga określenia jego typu i argumentów, co sprowadza się np. do określenia czy należy uruchomić Flux2D czy 3D, 32 czy 64 bitową wersję oraz z jaką wielkością pamięci i językiem obsługi.
Przykładowe uruchomienie serwera sprowadza się do polecenia:

```
// Create local Flux server
String[] arguments = {NUMERICAL_MEMORY_LABEL +
        "600000000",   LANGUAGE_LABEL + "2"};
int serverUid = FMP.FMP_startLocaleServer(FLUX3D_32,
"../", arguments);
```

Po uruchomieniu serwera o przykładowej nazwie (jak wyżej) **serverUid** wykorzystując metodę klasy FMP istnieje możliwość przesyłania na uruchomiany serwer poleceń do wykonania. Przykładowo wczytanie zapisanego na dysku projektu programu Flux o nazwie Test.FLU to polecenie:

```
// Load project
  FMP.FMP_loadProject(serverUid, "Test.FLU");
```

W prosty sposób można umieszczać zmienne w przestrzeni pamięci uruchomionego serwera **serverUid** oraz pobierać wartości z tej przestrzeni pamięci. Przykładowo utworzenie tablicy **Tab_int** z wartościami typu Integer (np. liczby 123 i 456) w przestrzeni pamięci serwera i następnie ich pobranie do tablicy **tab_integer** w przestrzeni programu nadrzędnego można zapisać jako:

```
// set Array Integer
FMP.FMP_setJythonIntArrayValue(serverUid, "Tab_int",
                    new int[]{123, 456});
int[]  tab_integer=FMP.FMP_getJythonIntArrayValue
                    (serverUid, "Tab_int");
```

Podobnie utworzenie w przestrzeni pamięci serwera serverUid tablicy **Tab_dou** typu double zawierającej przykładowo określone w programie nadrzędnym zmienne double o nazwie Prad oraz Nap można zapisać jako:

```
// set Array Double
FMP.FMP_setJythonDoubleArrayValue(serverUid,
                "Tab_dou", new double[]{Prad, Nap});
```

Dla tablicy **Tab_str** typu String zawierającej przykładowo określone w programie nadrzędnym zmienne String o nazwie *Srednica* oraz *Dl* (która mogą zawierać zapisane w postaci tekstowej wartości liczbowe) można zapisać jako:

```
// set Array String
FMP.FMP_setJythonStringArrayValue (serverUid,
                "Tab_str", new String[]{Srednica, Dl});
```

Zrealizowania polecenia w programie Flux sprowadza się do wykorzystania metody
FMP.FMP_executeJythonCommand ("polecenie w jezyku Python").
Zostało to zaprezentowane poniżej na przykładzie utworzenia w programie Flux parametru typu geometrycznego o nazwie SRED_WSAD, któremy przypisano pierwszą wartość z tablicy jednowymiarowej **Tab_str** (została tam zapisana, jak wyżej, wartość zmiennej *Srednica*):

```
// create parameter
FMP.FMP_executeJythonCommand(serverUid,
"ParameterGeom['SRED_WSAD'].expression=Tab_str[0]");
```

Wykorzystywania FMP.FMP_executeJythonCommand("") jest bardzo wszechstronne, pozwala na tworzenie, modyfikowanie czy pobieranie różnego typu parametrów czy wartości zmiennych występujących w programu Flux. Rodzaj wykonywanej operacji jest zapisany w postaci wartości typu string zawierającej polecenie sterujące pracą programu Flux w języku Python.

Przy pisaniu tych poleceń można, dla ułatwienia pracy, korzystać z poleceń generowanych automatycznie przez program Flux, bowiem w trakcie interaktywnej pracy z programem, przy wykorzystaniu jego pre-procesora graficznego, wszelkie wykonywane manualnie polecenia są zapisywane za pomocą komend języka Python w tzw. pliku śledzącym. Można więc wykonać operację w sposób manualny, a następnie pobrać wygenerowane polecenia i umieścić je w pisanym programie zarządzającym.
Zakończenie pracy z serwerem programu Flux jest realizowane poleceniem:

```
// Close Flux server
FMP.FMP_stopServer(serverUid);
```

Budowa własnej aplikacji z wykorzystaniem API jest zwykle realizowana w celu opracowania oprogramowania dedykowanego do konkretnego zastosowania praktycznego czy wręcz konkretnego użytkownika poprzez:

- uproszczenia obsługi programu komercyjnego interfejsem w pełni dopasowanym do potrzeb konkretnego użytkownika,
- opracowanie złożonego pakietu, który znacząco rozszerza możliwości programu komercyjnego.

Budowa inerfejsu ułatwiającego pracę z programem komercyjnym – obliczenia elektromagnetyczne

Korzystając z opisanego wyżej interfejsu API programu Flux, zbudowano aplikację obliczającą podstawowe parametry elektryczne cylindrycznego układu wzbudnik-wsad. Ze względu na występującą symetrię osiową oraz rozpatrywanie jedynie zagadnień elektrycznych przy wymuszeniach sinusoidalnych budowę programu oparto o wykorzystanie pakietu Flux2D z modułami do obliczeń pola elektromagnetycznego AC.

W założeniach projektowych przyjęto, że program ma umożliwiać realizowanie obliczeń dla:

- narzuconej wartości skutecznej oraz częstotliwości prądu wzbudnika ,
- zarówno liniowych, jak i nieliniowych magnetycznie materiałów wsadu,
- wartości rezystancji i parametrów magnetycznych materiału wsadu określonych przez użytkownika,
- obecności lub braku zewnętrznego bocznika magnetycznego o parametrach materiałowych i geometrycznych określonych przez użytkownika,
- parametrów geometrycznych wzbudnika cylindrycznego (w tym liczby zwojów, przekroju rurki oraz wzajemnych odległości zwojów) i wsadu walcowego określonych przez użytkownika.

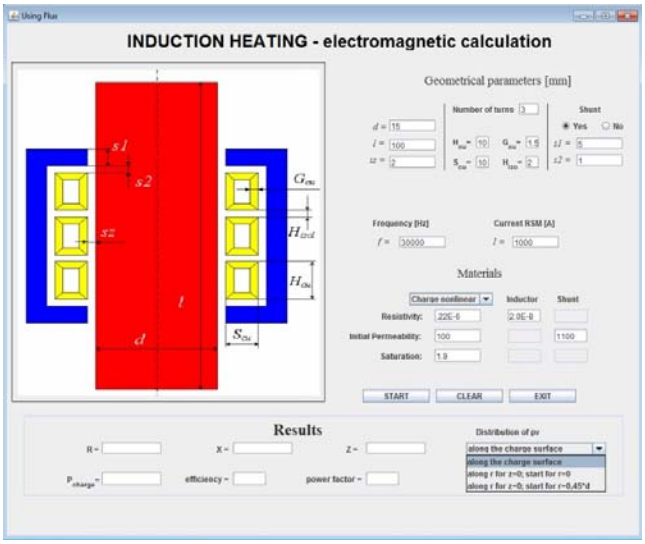
Przy budowie programu postawiono wymaganie możliwie największego uproszczenia pracy wykonywanej przez obsługę programu i sprowadzenia jej jedynie do podania określonych wyżej danych wejściowych.

Przyjęto, że danymi wyjściowymi będą:

- rezystancja, reaktancja i impedancja układy wzbudnik-wsad,
- moc czynna wydzielona we wsadzie,
- sprawność elektryczna oraz współczynnik mocy układu wzbudnik-wsad.

Dla tak określonych wymagań projektowych przyjęto dodatkowo, że w programie będzie istniała możliwość wykonywania obliczeń wielokrotnych oraz w celu ułatwienia obsługi będą występować wartości domyślne wielkości ustalanych przez użytkownika przyjmowane w chwili uruchomienia programu.

W wyniku przeprowadzonych prac zbudowana program w języku Java wykorzystujący API programu Flux. Jego interfejs użytkownika przedstawiono na Rys. 2.

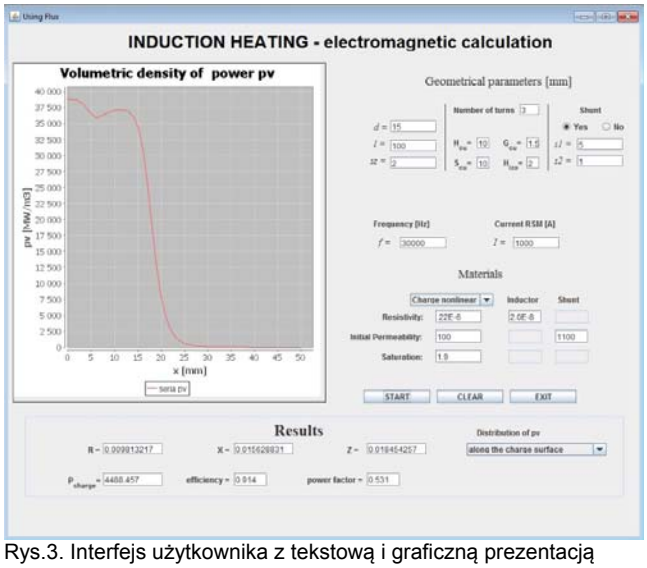


Rys.2. Interfejs użytkownika programu do obliczeń elektrycznych cylindrycznego układu wzbudnik-wsad z wykorzystaniem API programu Flux.

Pracujący program korzysta ze skryptu w języku Python umożliwiającego automatyczną budowę modelu geometrycznego o liczbie zwojów wzbudnika oraz wymiarach geometrycznych wzbudnika, bocznika i wsadu określanych przez użytkownika. Opracowany skrypt pozwala również na wykonanie automatycznego, zależnego od częstotliwości prądu wzbudnika oraz parametrów materiałowych, podziału obszaru obliczeń na elementy skończone. Meshing jest wykonywany w oparciu o zasadę występowania minimum dwóch elementów skończonych na głębokości wnikania fali elektromagnetycznej.

Należy podkreślić, że sposób i rodzaj prezentowanych wyników obliczeniowych w interfejsie wynikał z przyjętych założeń projektowych, bowiem przy wykorzystaniu API można mieć dostęp do wszystkich wyników obliczeń programu komercyjnego, które dodatkowo są zapisywane w pliku i mogą być dostępne w sposób typowy dla obsługi programu komercyjnego.

Na Rys. 3 przedstawiono, zrealizowaną w programie, prezentację wyników zarówno w postaci tekstowej, jak i graficznie jako rozkład obliczonej gęstości objętościowej mocy wydzielonej wzdłuż wybranej ścieżki (w przykładzie długość wsadu poczynając od punktu środkowego).



Rys.3. Interfejs użytkownika z tekstową i graficzną prezentacją wyników obliczeń.

Budowa inerfejsu ułatwiającego sprzężone obliczenia elektromagnetyczno-cieplne

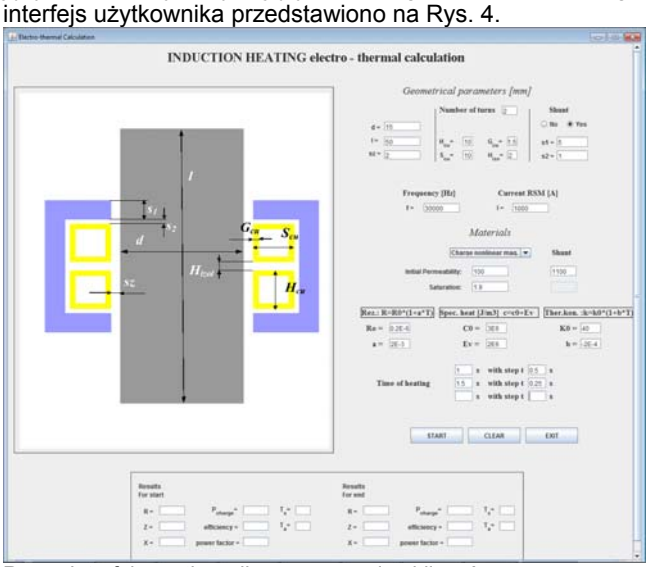
Oprogramowanie Flux posiada szereg modułów obliczeniowych nakierowanych na rozwiązywanie zagadnień polowych różnego typu. W przypadku nagrzewania indukcyjnego bardziej użyteczne od analizowania samego pola elektromagnetycznego układu wzbudnik-wsad jest zazwyczaj realizacja sprzężonych obliczeń elektromagnetyczno-cieplych. Program Flux® posiada moduły pozwalające na zrealizowanie takich obliczeń. Tak jak w przypadku innych programów MES wymaga to zbudowania obliczeniowego modelu geometrycznego, który po przypisaniu parametrów materiałowych, wymuszeń i warunków brzegowych, a następnie podziale na elementy skończone przyjmuje formę dyskretnego modelu obliczeniowego. Zrealizowanie tego zadania, może dla osób słabo zorientowanych w tematyce obliczeń polowych, stanowić pewien problem skutkujący błędami obliczeniowymi. Dotyczy to szczególnie sytuacji, gdy wymagane jest wykonywanie obliczeń w warunkach przemysłowych, przez osoby obsługujące stanowiska produkcyjne lub z obszaru marketingu.

Poniżej omówiono przykład w którym dzięki zbudowanemu w Java interfejsowi pakietu Flux® możliwe jest bardzo łatwe wykonywanie sprzężonych obliczeń nagrzewania indukcyjnego układów cylindrycznych.

Podobnie jak w przykładzie wyżej przyjęto, że budowana aplikacja, wykorzystując podane przez użytkownika parametry geometryczne i materiałowe, ma samoczynnie budować numeryczny dyskretny model obliczeniowy i podawać na ekranie podstawowe parametry elektryczne układu wzbudnik-wsad dla początkowego i końcowego etapu nagrzewania.

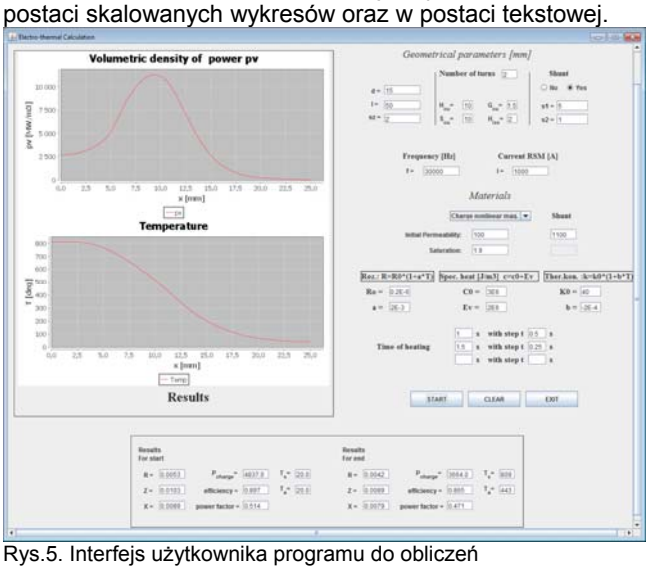
Przewidziano, również graficzne przedstawianie wyników obliczeń poprzez wykresy rozkładu gęstości objętościowej mocy i temperatury wsadu.

W wyniku przeprowadzonych prac zbudowano aplikację w języku Java wykorzystujący API programu Flux®, którego interfejs użytkownika przedstawiono na Rys. 4.



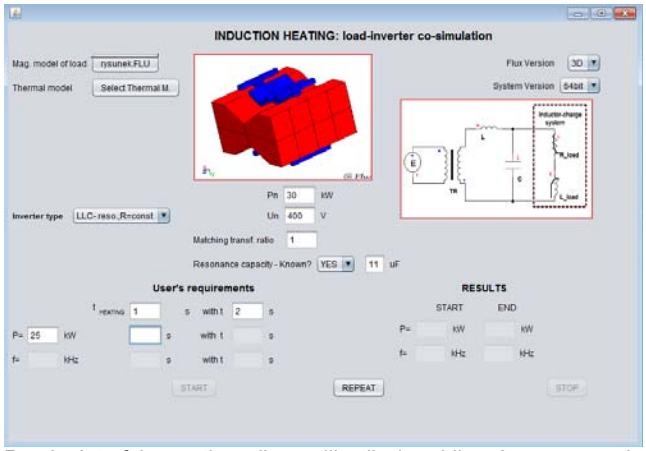
Rys.4. Interfejs użytkownika programu do obliczeń elektromagnetyczno-cieplnych z wykorzystaniem API programu Flux.

W zrealizowanej aplikacji istnieje możliwość wykorzystywania materiałów zarówno o parametrach niezależnych jak i zależnych od temperatury. Na Rys.5 przedstawiono sposób prezentacji wyników obliczeń w postaci skalowanych wykresów oraz w postaci tekstowej.



Rys.5. Interfejs użytkownika programu do obliczeń elektromagnetyczno-cieplnych- prezentacja wyników.

Wykorzystując oprogramowanie do obliczeń polowych można budować również aplikacje do nagrzewania indukcyjnego obejmujące funkcjonalności nie występujące w znanych oprogramowaniach komercyjnych, np. z uwzględnieniem wzajemnej interakcję źródło zasilania-indukcyjny układ grzejny, w tym z układami sterowania mocą czy temperaturą lub dla systemów nagrzewania dwuczęstotliwościowego. Na Rys.6 przedstawiono interfejs użytkownika tego typu aplikacji, która pozwala symulować proces indukcyjne nagrzewania w połączeniu z falownikami z rezonansem szeregowym, równoległym czy złożonymi układami LLC [8]. Aplikacja wykorzystuje modele numeryczne zbudowane programem Flux osobno dla zagadnienia elektromagnetycznego i cieplnego, co pozwala symulować proces indukcyjne nagrzewania niemal dowolnego układu wzbudnik-wsad.



Rys.6. Interfejs użytkownika aplikacji do obliczeń nagrzewania indukcyjnego z uwzględnieniem interakcji źródło-obciążenie.

Uwagi końcowe

Korzystanie z API programów komercyjnych do obliczeń elektromagnetycznych i cieplnych takich jak np. program Flux® stwarza możliwość relatywnie łatwego budowania własnych aplikacji, w których te złożone obliczenia numeryczne wykonywane będą przez sprawdzone w działaniu pakiety, co zapewnia zarówno dużą dokładność, jak i szybkość działania. Programy te mogą być łatwo dopasowywane do potrzeb użytkownika poprzez znaczące ułatwienie ich obsługi, dodanie nowych funkcjonalności lub wyposażenie pakietu w bloki realizujące obliczenia z innej dyscypliny naukowo-technicznej jak np. wyposażanie programu do obliczeń elektromagnetyczno-cieplnego w blok realizujący obliczenia metalurgiczne czy uwzględniający współpracę układu wzbudnik-wsad i źródła zasilania.

Praca finansowana ze środków NCBiR w ramach projektu Badań Stosowanych nr umowy PBS1/A4/2/2012

***Autor**: dr hab. inż. Jerzy Zgraja prof. P.Ł., Politechnika Łódzka, Instytut Informatyki Stosowanej, al. Politechniki 11, 90-924 Łódź, E-mail: jzgraja@p.lodz.pl;*

LITERATURA

[1] Blinov K., Nikanorov A., Nacke B., Klöpzig M., Numerical simulation and investigation of induction through-heaters In dynamic operation mode, *The Int. J. for Computation and Mathematics in Electrical and Electronic Engineering COMPEL*, vol. 30 (2011), no. 5, 1539-1549

[2] Zgraja J., The optimisation of induction heating system based on multiquadric function approximation, *The Int. J. for Computation and Mathematics in Electrical and Electronic Engineering COMPEL*, 24 (2005), no. 1, 305-313

[3] Fabbri M., Morandi A., and Ribani L, DC induction heating of aluminum billets using superconducting magnets, *COMPEL*, vol. 27, No. 2 (2008), pp. 480–490.

[4] Zgraja J., Simulation of Induction Hardening of Flat Surfaces of Moving Massive Elements, *Int. Journal of Materials & Product Technology*, vol. 29 (2007), pp. 103-123.

[5] Julegin A., Demidovich V., et., Coupled modelling of induction systems: heaters and power sources, *HES-13*, 21-24 May 2013, Padua, pp.237-243.

[6] Cedrat, 'Flux User's Guide' v.11.1, 2012.

[7] ANSYS, 'Maxwel 15 software documentation', ANSYS Inc, 2011.

[8] Kobos W., Zgraja J., 'Pasywne układy LLC i LCCL dopasowania impedancji obciążenia indukcyjnie nagrzewanego wsadu' *Przegląd Elektrotechniczny*, Nr 2/2014, ss. 40-43