mgr inż. Anna Butterweck
dr inż. M. H. Ghaemi
Wydział Oceanotechniki i Okrętownictwa Politechniki Gdańskiej

ZASTOSOWANIE TECHNOLOGII GPGPU DO WSPOMAGANIA INŻYNIERSKICH OBLICZEŃ NUMERYCZNYCH. NA PRZYKŁADZIE ANALIZY PRZEPŁYWU PRZEZ OŚRODEK POROWATY

W pracy zawarto podstawowe informacje na temat technologii GPGPU oraz struktury NVIDIA CUDA. Opisano równania zachowania rządzące przepływami oraz ich dyskretyzację numeryczna. Zbadano, również możliwości wykorzystania technologii GPGPU w celu zoptymalizowania czasu wykonywania obliczeń numerycznych przepływu przez ośrodek porowaty. Opisano model numeryczny ośrodka porowatego. Dla sprawdzenia poprawności działania programu wykonano test przepływu przez prostokątny kanał przepływowy bez obecności ciał stałych w kanale. Następnie przeprowadzono obliczenia pełnego modelu dla czterech rozmiarów siatek obliczeniowych 32x32, 64x64, 128x128, 256x256. Przedstawiono wyniki (rozkłady ciśnienia, prędkości wypadkowej, ciśnienia odniesionej i prędkości wypadkowej odniesionej) dla wybranego rozmiaru siatki i wybranego wariantu zagęszczenia granulek oraz ich promienia. Ponadto wyliczono odchylenie standardowe każdej z wielkości. Porównano czas wykonania obliczeń na CPU i GPU i określono granicę opłacalności każdego z nich.

APPLICATION OF GPGPU TECHNOLOGY TO OPTIMISE ENGEENERING CALCULATIONS. ON EXAMPLE OF FLUID FLOW THROUGH POROUS MEDIA.

In this thesis basic description of GPGPU technology and NVIDIA CUDA structure are presented. First, the conservation equations for fluid flow and their numerical discretisation are given. Next, the possibilities of applying GPGPU technology to optimise time of numerical calculations of fluid flow through porous media are investigated and numerical model of porous media is made. To verify the results of the solvers fluid flow through an empty channel and through the channel with single solid particles are tested. Then, the main results have been presented for four sizes of numerical grids: 32x32, 64x64, 128x128 and 256x256. For one size of the grid and solid particles density with taking into account their size calculations are done. Finally, the results chosen variant are presented (standard deviation value and distribution of pressure, velocity, reference pressure and reference velocity). The main criteria for selecting the adequate method is the time of calculation. The value of this criteria for both methods are determined for each variant and the critical grid numbers are expressed.

1. GPGPU - informacje podstawowe

Technologia GPGPU (z ang. General-Purpose Computing on Graphics Processing Units) polega na wykorzystaniu procesora graficznego GPU (z ang. Graphics Processing Unit), który zwykle zajmuje się przetwarzaniem grafiki, do wykonywania zadań zwykle przeznaczonych dla procesora sekwencyjnego CPU (ang. Central Processing Unit). Jest to możliwe dzięki wykorzystaniu potoków graficznych do przetwarzania innych danych niż tylko grafika. Do rozpowszechnienia wykorzystania technologii GPGPU przyczyniły się firmy takie jak NVIDIA czy ATI (dziś AMD). Te dwie firmy udostępniły efekty swojej pracy nad NVIDIA CUDA i ATI Stream czyli nad wielordzeniową architekturą nowoczesnych układów graficznych pozwalającą na wykorzystanie przetwarzania równoległego do zadań innych niż przetwarzanie grafiki.

Na przykład środowisko programistyczne NVIDIA CUDA jest opartym na języku programowania C środowiskiem programistycznym wysokiego poziomu. Składa się on m.in. z dedykowanego kompilatora (NVCC) oraz debuggera (umożliwia śledzenie kodu wykonywanego na CPU jak i na GPU). W językach Python, Fortran, Java, C# oraz Matlab można wykorzystywać specjalne biblioteki umożliwiające wykorzystanie CUDA. Pierwsze wydanie środowiska wspierało systemy operacyjne Windows oraz Linux. Od wersji 2.0 wspierany jest również Mac OS X.

W grach komputerowych moc obliczeniową można wykorzystać do obliczeń fizyki w grach (symulacja poruszającego się otoczenia w grach np. chmury, deszcz czy rozpryski wody), ale CUDA jest również wykorzystywana do przyspieszania obliczeń w takich dziedzinach jak biologia, fizyka, kryptografia i inne obliczenia inżynierskie. Programy inżynierskie takie jak Mathematica czy Ansys mają wsparcie CUDA. Co oznacza, że obliczenia wykonywane przez te programy mogą być prowadzone w sposób równoległy na karcie graficznej bez ingerencji użytkownika. Dla potrzeb tego segmentu NVIDIA opracowała specjalny procesor graficzny TESLA (choć CUDA współpracuje również z wcześniejszymi produktami NVIDIA np. z kartami GeForce (serii 8, 9, 100 i 200 i z minimum 256 MB pamięci), Quadro).

W prawie każdym mikroprocesorze możemy wyróżnić następujące bloki [1]:

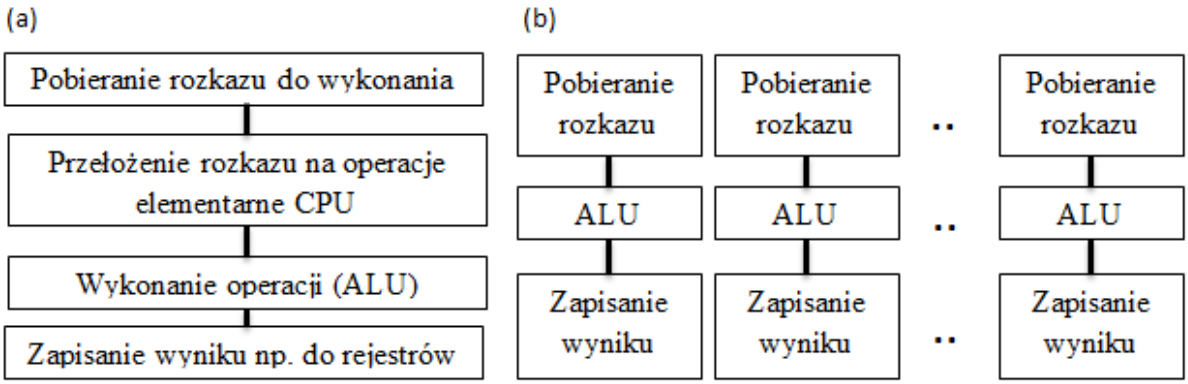
• ALU – jednostka arytmetyczno-logiczna (Arithmetic Logic Unit), wykonuje ona operacje arytmetyczno - logiczne na dostarczonych jej danych.

• Control, CU – układ sterowania (Control Unit), inaczej dekoder rozkazów. Odpowiedzialny jest on za dekodowanie dostarczonych mikroprocesorowi instrukcji i odpowiednie sterowanie pozostałymi jego blokami

• Cache, rejestry – umieszczone wewnątrz mikroprocesora komórki pamięci o niewielkich rozmiarach służące do przechowywania tymczasowych.

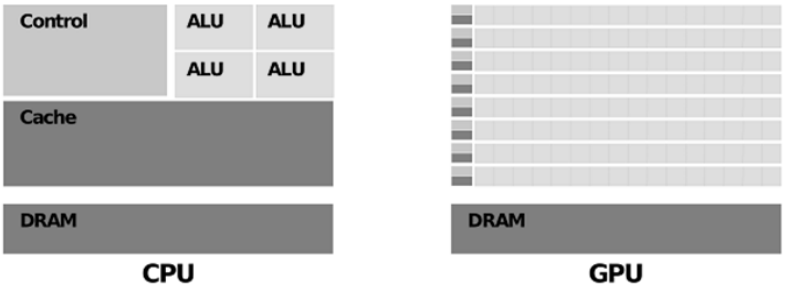
• DRAM, Pamięć dynamiczna, (ang. Dynamic Random Access Memory) – rodzaj ulotnej pamięci półprzewodnikowej o dostępie swobodnym, której bity są reprezentowane przez stan naładowania kondensatorów.

Procesor, także CPU jest to urządzenie cyfrowe sekwencyjne, które pobiera dane z pamięci, interpretuje je i wykonuje jako rozkazy. Schemat działania CPU pokazuje rys.1a. Jak można zauważyć obliczenia na CPU charakteryzują się jednotorowością czyli obliczenia wykonywane są po kolei, “jedno po drugim”.



Rys. 1 (a) Schemat działania CPU (b) Schemat działania GPU

Procesor graficzny GPU jest główną jednostką obliczeniową znajdującą się w nowych kartach graficznych. Schemat działania GPU pokazuje rys. 1b. Obliczenia na GPU charakteryzują się tym, iż są przeprowadzane równolegle.
Różnice w sposobach działania CPU i GPU wynikają bezpośrednio z budowy procesora CPU i procesora GPU (rys. 2). Jak widać na rys. 2 na GPU każdej jednostce ALU odpowiada jednostka Control i Cache w przeciwieństwie do CPU gdzie jest jedna jednostka typu Control i jedna jednostka typu Cache na wszystkie jednostki ALU. Taka właśnie budowa GPU pozwala na znaczne przyspieszenie obliczeń poprzez prowadzenie ich równolegle. Należy jednak zwrócić uwagę, iż przyspieszenie obliczeń przez zrównoleglenie możemy uzyskać wtedy i tylko wtedy gdy kolejne obliczenia nie są sekwencyjnie zależne i są powtarzane wielokrotnie.



Rys. 2 Różnice w architekturze układu CPU a GPU [1]

Symulacje której wyniki zostały przedstawione w niniejszym artykule powstały przy wykorzystaniu karty graficznej NVIDIA GeForce 9400M G co zdeterminowało użycie NVIDIA CUDA.

2. BADANIE PRZEPŁYWU PRZEZ OŚRODEK POROWATY

2.1 Równania zachowania i ich dyskretyzacja numeryczna

Głównymi równaniami zachowania, które rządzą przepływami, są: równanie zachowania masy, równanie zachowania pędu, równanie zachowania energii. Ze względu na to, iż w niniejszym opracowaniu założono brak wymiany ciepła i pracy z otoczeniem oraz założono, że przepływ jest laminarny, możliwym było pominięcie równania zachowania energii.
Równanie zachowania masy przy założeniu stałej gęstości czynnika roboczego (ρ=const):

$$div\boldsymbol{u} = 0. \tag{1}$$

Równanie zachowania pędu w postaci:

$$\rho\frac{d\boldsymbol{u}}{dt} - \rho f + \nabla p - \frac{1}{Re}\nabla^2\boldsymbol{u} = 0. \tag{2}$$

W powyższym układzie równań, możemy zauważyć, iż ciśnienie występuje tylko w równaniu pędu, w formie niejawnej. Jednak od jego wartości zależy spełnienie warunku zerowej dywergencji pola prędkości. By numerycznie wyznaczyć ciśnienie wprowadzono do równania ciągłości tzw. człon sztucznej ściśliwości. Otrzymano ostateczny układ równań:

$$\begin{cases} \frac{1}{\beta}\frac{\partial p}{\partial t} + \frac{\partial u}{\partial x} + \frac{\partial u}{\partial y} = 0 \\ \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + \frac{\partial p}{\partial x} - + \frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) = 0. \\ \frac{\partial u}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + \frac{\partial p}{\partial y} - + \frac{1}{Re}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) = 0 \end{cases} \tag{3}$$

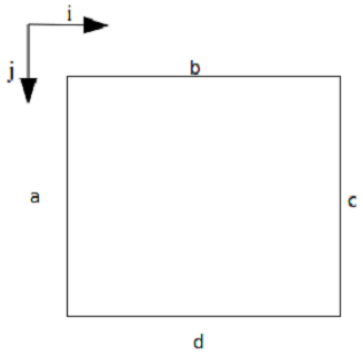
Powyższy model jest silnie nieliniowy, co determinuje konieczność rozwiązywania na drodze numerycznej. Układ równań (3) został więc zdyskretyzowany. Wprowadzono:

- dyskretyzacje czasu schematem jawnym Euler'a
- dyskretyzację członu konwekcyjnego wg. schematu upwind I rzędu
- centralną dyskretyzację pozostałych pochodnych.

Należy zaznaczyć, iż całkowanie w czasie ma na celu jedynie znalezienie rozwiązania stacjonarnego.

2.2 Założenia modelu ośrodka porowatego

Dla potrzeb tej pracy założono iż ośrodek porowaty jest to prostokątny kanał przepływowy (o wymiarach *N x M*), z losowo rozmieszczonymi ziarnami w nim. Brzegi kanału przepływowego oznaczono jak na rys. 3. Jak wlot uznaje się brzeg „a", jako wylot brzeg „c". Na ściankach „b" i „d" założono brak poślizgu i nieprzenikalność płynu dla prędkości, ponadto dla ciśnienia warunki brzegowe Neumann'a.



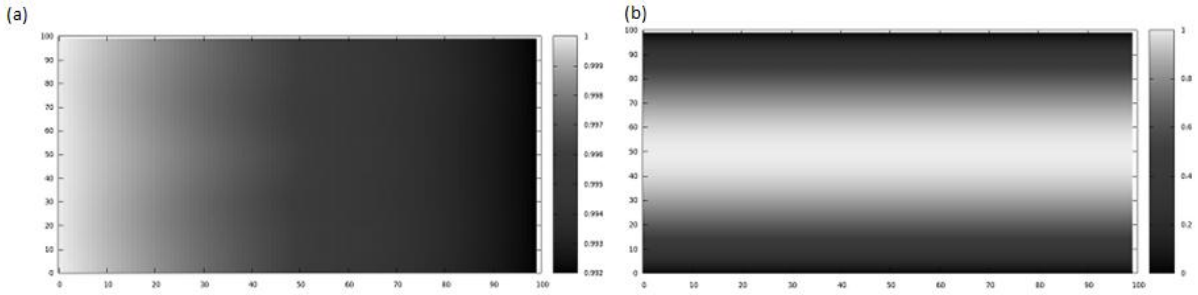
Rys. 3 Oznaczenie brzegów kanału przepływowego.

Dla sprawdzenia poprawności działania solvera Naviera-Stoke'sa przeprowadzono symulację w pustym kanale obliczeniowym (bez granulek wewnątrz). Symulację tą przeprowadzono dla warunków brzegowych jak w Tabeli 1.

Brzeg	p	u (prędkość zgodna z osią „i")	v (prędkość zgodna z osią „j")
wlot „a"	1,0	$u = f(j) = c_0 j^2 + c_1 j + c_2$	0,0
wylot „c"	0,992	$\frac{du}{dx} = 0$	$\frac{dv}{dx} = 0$

Tabela 1. Zestawienie warunków brzegowych dla symulacji sprawdzających poprawność działania programu

Wyniki symulacji (dla parabolicznego rozkładu prędkości wlotowej) przedstawiono na rys. 4.



Rys. 4 (a) Rozkład ciśnienia przy parabolicznym rozkładzie prędkości wlotowej. (b) Rozkład prędkości wypadkowej przy parabolicznym rozkładzie prędkości wlotowej

Przepływ przez ośrodek porowaty jest przepływem dwufazowym. Ośrodek porowaty [3] jest to ciało stałe lub grupa ciał stałych zawierająca wystarczającą ilość pustych przestrzeni aby umożliwić przepływ płynu. Wolne przestrzenie, nazywane również porami, tworzą złożoną i nieregularną sieć kanałów i połączeń. Porowatość ośrodka jest to stosunek objętości zajmowanej przez puste przestrzenie do objętości całości ośrodka. Porowatość nie przekazuje informacji o kształcie czy ilości porów a jedynie niesie informację o tym jaką część całości objętości stanowią przestrzenie wolne.

Aby zamodelować ośrodek porowaty do kanału przepływowego wprowadzono granulki. Granulki są zdefiniowane jako okrągłe obszary ciała stałego wewnątrz kanału

przepływowego. Ilość ziaren *D*, które są umieszczone w kanale obliczeniowym są określone przez parametr – gęstość ziaren - ρ_g – ekwiwalent porowatości.

$$D = \rho_g \frac{n \cdot m}{\pi r^2}$$

gdzie: ρ_g-gęstość ziaren, *n, m* – wymiary obszaru w kanale przepływowym w którym mogą zostać umieszczone granulki, *r* – promień granulka (stały dla wszystkich granulek w danej symulacji). Współrzędne środków granulek są generowane losowo. Numerycznie granulki są rozważane jako grupa węzłów siatki o następujących warunkach brzegowych:

- ciśnienie w punkcie jest średnią arytmetyczną ciśnień z punktów sąsiednich
- *u=0,0*
- *v=0,0.*

Przykładowy rozkład granulek pokazano na rys. 5.

2.3 Wyniki i ich interpretacja

W tej części zaprezentowane zostaną wyniki. Wyniki obliczeń na procesorze CPU i GPU zostały porównane dla kilku wariantów. Obliczenia wykonano dla czterech rozmiarów siatek (wymiary *N x M*: 32 x 32, 64 x 64, 128 x 128, 256 x 256). Dla wszystkich rozmiarów siatek obliczenia przeprowadzono dla trzech wariantów: $\rho_g = 0$; $\rho_g = 0{,}1\ r = 1$; $\rho_g = 0{,}3$ r $= 3$.

Do ostatecznych obliczeń warunki brzegowe przyjęto takie jak w Tabeli 2. Ponadto warunki na ściankach (brzeg „b" i „d") takie jak opisano powyżej.

Brzeg	p	u (prędkość zgodna z osią „i")	v (prędkość zgodna z osią „j")
wlot „a"	p(0,j)=p(1,j)	1,0	0,0
wylot „c"	1,0	$\frac{du}{dx} = 0$	$\frac{dv}{dx} = 0$

Tabela 2. Zestawienie warunków brzegowych dla symulacji.

Dla porównania wyników uzyskanych na procesorze CPU z wynikami z GPU wprowadzono pojęcie ciśnienia odniesionego które jest definiowane jako:

$$\overline{p_{i,j}} = \frac{p_{i,j}^{CPU} - p_{i,j}^{GPU}}{p_{ref}}.$$

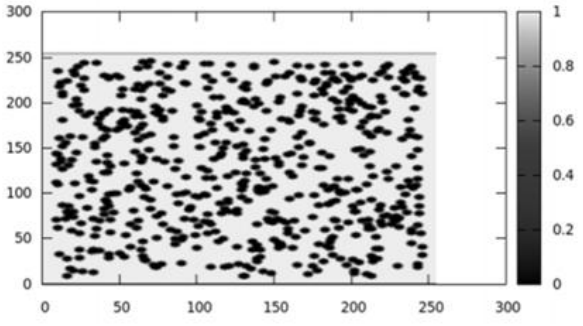
Odniesiona prędkość wypadkowa $\left(mag_{i,j} = \sqrt{u_{i,j}^2 + v_{i,j}^2}\right)$ jest zdefiniowana w podobny sposób:

$$\overline{mag_{i,j}} = \frac{mag_{i,j}^{CPU} - mag_{i,j}^{GPU}}{mag_{ref}}.$$

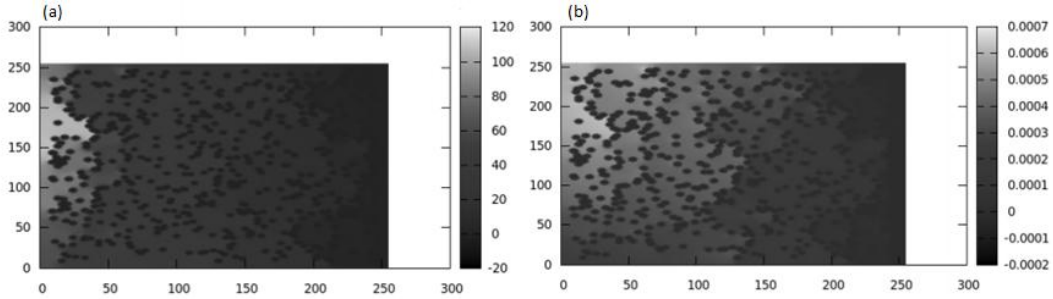
Ponadto wyliczono odchylenia standardowe (zarówno dla ciśnienia i prędkości wypadkowej):

$$\sigma_\Phi = \sqrt{\frac{\sum_{i=1}^{NM}(\Phi_i - \Phi_{śr})}{NM}}$$

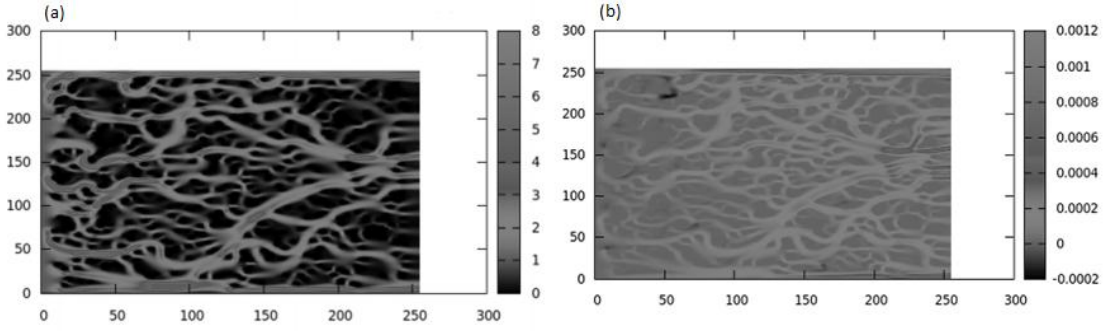
Dodatkowo porównano czasy wykonywania obliczeń zarówno na CPU i GPU.
Na rys. 5-7 przedstawiono przykładowe wyniki dla siatki 256 x 256; $\rho_g = 0{,}3;\ \mathrm{r} = 3$.



Rys. 5 Rozmieszczenie granulek w kanale przepływowym



Rys. 6 (a) rozkład ciśnienia (CPU) (b) rozkład ciśnienia odniesionego



Rys. 7 (a) rozkład prędkości wypadkowej (CPU) (b) rozkład prędkości wypadkowej odniesionej

Pozostałe wyniki przedstawiono w tabeli 3.

Odchylenie standardowe ciśnienia	$\sigma_p = 0{,}019914$
Odchylenie standardowe prędkości wypadkowej	$\sigma_{mag} = 0{,}000345$
Czas wykonywania obliczeń na CPU	15m21.115s
Czas wykonywania obliczeń na GPU	06m58.198s

Tabela 3 Pozostałe wyniki

3. Podsumowanie

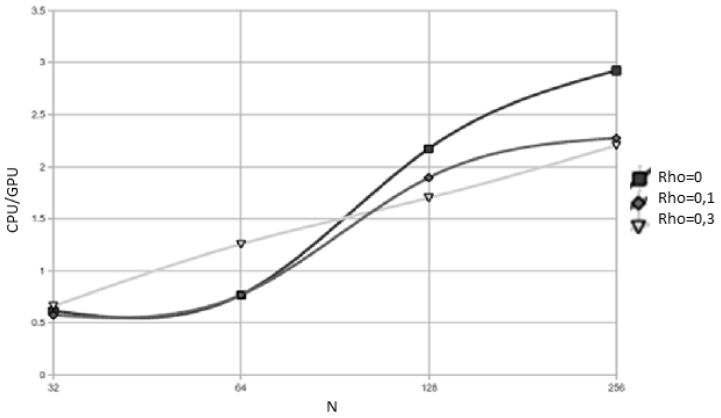
Celem tej pracy było sprawdzenie czy zastosowanie technologii GPGPU może znacząco przyspieszyć czas wykonywania obliczeń inżynierskich. Na przykładzie przepływu płynu nieściśliwego, lepkiego przez ośrodek porowaty wykazano, że jest to możliwe. Należy jednak zwrócić uwagę na kilka szczegółów.

Pamiętać trzeba o tym, iż zaimplementowany solver na GPU nie jest napisany w sposób optymalny. Na pewno istnieje możliwość znaczącego przyspieszenia używając obszaru pamięci współdzielonej karty graficznej czy przez dokładniejszą synchronizację wątków (szczegóły [1]).

Z drugiej jednak strony należy zwrócić uwagę na to, że zastosowanie technologii GPGPU wymaga zastosowania nowych algorytmów, zupełnie innych niż te które są obecnie powszechnie używane w nauce i technice. Dlatego też nie można jednoznacznie stwierdzić, że sumarycznie (uwzględniając czas tworzenia nowych algorytmów i oprogramowania) zastosowanie GPGPU zawsze będzie się wiązało z oszczędnością czasu. Przy dużych programach, które będą wielokrotnie używane do przetwarzania dużych ilości danych ma to sens (czego przykładem są programy komercyjne typu Ansys czy Mathematica które używają wsparcia NVIDIA CUDA) jednak dla mniejszych ilości danych warto przekalkulować czy tak duża zmiana, jaką jest przejście na technologię GPGPU, będzie opłacalną.

Do zalet technologii GPGPU należy niewątpliwie zaliczyć cenę kart graficznych, która jest kilkukrotnie niższa w stosunku do klasycznych procesorów (czy też klastrów) o podobnej mocy obliczeniowej.

Na rys. 10 przedstawiono zależność stosunku czasu wykonania programu na CPU (t^{CPU}) do t^{GPU} w zależności od wielkości siatki i zadanej gęstości rozmieszczenia granulek ρ_g.



Rys 10 Stosunek czasu wykonania programu na CPU do GPU w zależności od wielkości siatki i zakładanej gęstości rozmieszczenia granulek.

Z rys. 10 wnioskować można, iż istnieje wielkość siatki dla której widocznie opłacalnym staje się zastosowanie technologii GPU. Dla obliczeń zawartych w tej pracy taka wielkość boku siatki kwadratowej N (zależnie od przypadku) zawiera się w przedziale $N\epsilon(64{,}128)$.

Niniejsza praca wykazuje, że technologia GPGPU jest godną uwagi gdyż może znacząco przyspieszyć obliczenia inżynierskie. Warto inwestować w nowe technologie i nowe algorytmy.

LITERATURA

[1] *NVIDIA CUDA Compute Unified Device Architecture Programming Guide Version 2.0*

[2] R. Puzyrewski, J. Sawicki, *Podstawy mechaniki płynów i hydrauliki*, Wydawnictwo Naukowe PWN, Warszawa 2000.

[3] A. E. Scheidegger, *The physics of flow through porous media*, University of Toronto Press, Toronto 1974.