Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 519654, 12 pages
http://dx.doi.org/10.1155/2014/519654

Research Article

Parallel Simulation of HGMS of Weakly Magnetic Nanoparticles in Irrotational Flow of Inviscid Fluid

Kanok Hournkumnuard,[1] Banpot Dolwithayakul,[2] and Chantana Chantrapornchai[3]

[1] *Department of Physics, Faculty of Science, Silpakorn University, Nakhon Pathom 73000, Thailand*
[2] *Department of Computing, Faculty of Science, Silpakorn University, Nakhon Pathom 73000, Thailand*
[3] *Department of Computer Engineering, Faculty of Engineering, Kasetsart University, Bangkok 10900, Thailand*

Correspondence should be addressed to Chantana Chantrapornchai; chantrapornchai@gmail.com

Received 19 December 2013; Accepted 27 March 2014; Published 11 May 2014

Academic Editors: B. Sun and F. Yu

The process of high gradient magnetic separation (HGMS) using a microferromagnetic wire for capturing weakly magnetic nanoparticles in the irrotational flow of inviscid fluid is simulated by using parallel algorithm developed based on openMP. The two-dimensional problem of particle transport under the influences of magnetic force and fluid flow is considered in an annular domain surrounding the wire with inner radius equal to that of the wire and outer radius equal to various multiples of wire radius. The differential equations governing particle transport are solved numerically as an initial and boundary values problem by using the finite-difference method. Concentration distribution of the particles around the wire is investigated and compared with some previously reported results and shows the good agreement between them. The results show the feasibility of accumulating weakly magnetic nanoparticles in specific regions on the wire surface which is useful for applications in biomedical and environmental works. The speedup of parallel simulation ranges from 1.8 to 21 depending on the number of threads and the domain problem size as well as the number of iterations. With the nature of computing in the application and current multicore technology, it is observed that 4–8 threads are sufficient to obtain the optimized speedup.

1. Introduction

High gradient magnetic separation (HGMS) is a separation technique which has been proven as a powerful one for the capture of weakly magnetic particles from suspension which conventional magnetic separation techniques using only permanent magnet cannot achieve. Some examples of potential applications of HGMS technique in the fields of biomedical and environmental science are the separation of red and white blood cells from small amount of blood sample in microfluidic device [1], trapping of infected red blood cell [2], and the separation of specific inorganic compounds from inland water [3]. Recently, many research works [4–6] have been studied to investigate the feasibility of using the principle of HGMS for concentrating nanotherapeutic carriers in specific regions within living body. This technique is called magnetic drug targeting (MDT). It is seen that the technique of HGMS becomes that which plays important role in nanotechnology so studying to understand the mechanism of nanoparticles capture by HGMS technique in various situations becomes useful for various research areas.

In this work, we study the problem of HGMS of weakly magnetic particle using a microferromagnetic wire as capture center in irrotational flow of inviscid fluid. The effect of diffusion becomes significant for nanoparticle motion so dynamic of capture process will be described statistically in terms of particle volume concentration. The problem is considered in an annular domain surrounding the wire. The continuity equation describing time rate of change of volume concentration in any part of the domain is formulated and is solved numerically as an initial and boundary value problem by using the explicit finite-difference method. A parallel algorithm for updating concentration value at each time step is developed based on openMP (http://openmp.org/wp/) to speed up the simulation because time step must be sufficiently small for the explicit method to obtain accurate simulation results.

2. Problem Description and Backgrounds

Figure 1 illustrates the definition of the problem. A long cylindrical ferromagnetic wire of radius a is placed traverse a flow of fluid containing weakly magnetic particles. In Figure 1, the flow is parallel to xy plane and wire axis is parallel to z-axis normally pointing out of the page. The distance is measured in the unit of wire radius. A uniform external magnetic field is applied perpendicular to wire axis and opposite to the incoming flow direction. This configuration is called the longitudinal mode operation. In Figure 1, the half of the domain on the right hand side of y-axis is called the upstream side while another half is called downstream side. For long wire, the flow profile of the fluid varies only on the xy plane but does not depend on z coordinates [7]. Moreover, the high gradient magnetic field around the wire also varies only on the xy plane [7]. Consequently, the problem can be treated as a two-dimensional one and the simulation domain is set up as an annular region surrounding the wire with the inner radius equal to wire radius and the outer radius is multiples of wire radius where the gradient of magnetic field is almost zero.

2.1. The Magnetic Field and Magnetic Force. By solving a magnetostatic boundary value problem in the domain using the boundary condition of uniform magnetic field (H_0) in the region that is far from the wire and magnetostatic boundary condition at the surface of the wire, the high gradient magnetic field ($\mathbf{H}$) around the wire can be expressed in polar coordinates (r_a, θ) as [7]

$$\mathbf{H} = H_0 \left[\left(1 + \frac{K_W}{r_a^2} \right) \cos\theta \mathbf{r} - \left(1 - \frac{K_W}{r_a^2} \right) \sin\theta \boldsymbol{\theta} \right], \quad (1)$$

where $K_W = M/2H_0$ is the demagnetization factor of the wire, M is the magnetization of the wire, and the boldface symbols $\mathbf{r}$ and θ represent the unit vector in radial and plane angular directions, respectively. The symbol $r_a = r/a$ means the radial distance in the unit of wire radius a. In the regions of high gradient magnetic field near to wire surface, the local magnetic field strength H is higher than the externally applied uniform magnetic field H_0, hence increasing the magnetic force acting on any weakly magnetic particles in these regions.

The magnetic force acting of any weakly magnetic particle of susceptibility χ_p dispersed in a fluid medium of susceptibility χ_f satisfied the equation [7]

$$\mathbf{F}_m = \mu_0 \left(\chi_p - \chi_f \right) V_p H \boldsymbol{\nabla} H, \quad (2)$$

where $\mu_0 = 4\pi \times 10^{-7}$ henry/m is the magnetic permeability of free space, V_p is volume of a particle, and the operator $\boldsymbol{\nabla}$

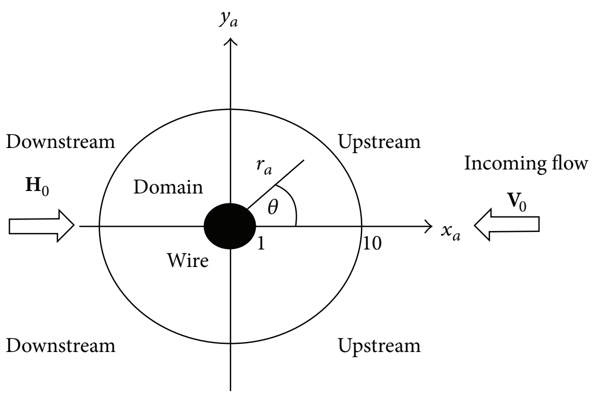


FIGURE 1: The simulation domain, direction of incoming fluid flow ($\mathbf{V}_0$), applied uniform magnetic field ($\mathbf{H}_0$), and the coordinates systems (not to scale).

expresses the gradient or nonuniformity of local magnetic field strength H.

2.2. Fluid Flow Field. In this work, we follow the work of Davies and Gerber [8] by considering an irrotational flow of inviscid fluid around the wire. The radial (V_r) and angular (V_θ) components of fluid in the domain can be expressed as [7]

$$V_r = V_0 \left(1 - \frac{1}{r_a^2} \right) \cos\theta,$$
$$V_\theta = -V_0 \left(1 + \frac{1}{r_a^2} \right) \sin\theta, \quad (3)$$

where V_0 is the incoming speed of fluid at the outer boundary of the domain.

2.3. The Continuity Equation. Dynamic of nanoparticle concentration in the domain is described by the continuity equation which states that, in any small elements of the domain, time rate of change of particle volume concentration is proportional to net volume flux passing through the elements and can be expressed as

$$\frac{\partial C}{\partial t} = D \boldsymbol{\nabla}^2 C - \boldsymbol{\nabla} \cdot (C \mathbf{V}_B) - \boldsymbol{\nabla} \cdot \left(\frac{D C \mathbf{F}_m}{k_B T} \right), \quad (4)$$

where C is particle volume concentration, t is time, D is diffusion coefficient, $\mathbf{V}_B$ is fluid velocity, k_B is Boltzmann's constant, and T is absolute temperature in Kelvin. The first, second, and third terms on the right hand side represent particle volume flux caused by diffusion process, convection by fluid flow, and the competition between effects of magnetic force and diffusion, respectively. Equation (4) is solved numerically, by using the explicit finite-difference method, as initial and boundary values problems to determine the pattern of concentration distribution within the domain.

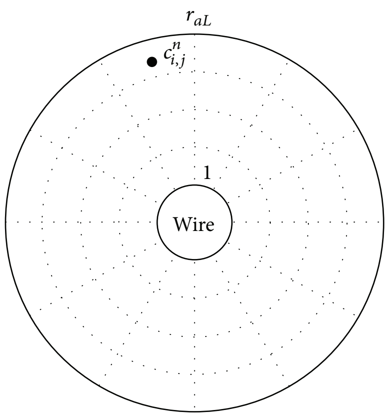


FIGURE 2: Simulation domain partitioning.

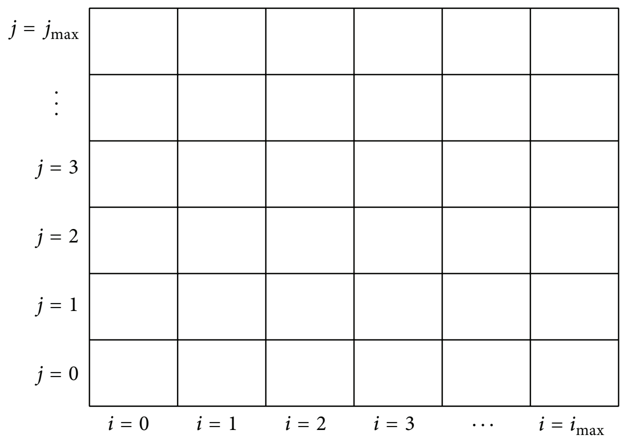


FIGURE 3: The concentration array.

3. Simulation Methodology

We partition the annular domain in Figure 1 into a structured mesh using normalized radial and angular steps Δr_a and $\Delta\theta$, respectively, as shown in Figure 2. At wire surface $r_a = 1$ and at outer boundary $r_a = r_{aL} = 10$. The symbol $C_{i,j}^n$ represents the volume concentration at the middle point $(r_{a,i}, \theta_j)$ of each element at an instantaneous time t_n.

In Figure 2, there exist two types of domain element. The first type is "ordinary" element where particle flux can pass through every surface and the second type is called "special" element which at least one of its surfaces is in contact with wire surface or the surface of static accumulation of particle. To solve (4), all derivative operations are approximated by using the finite-difference relations. The difference equation for "ordinary" element can be expressed as

$$\begin{aligned}\frac{C_{i,j}^{n+1} - C_{i,j}^n}{\Delta\tau} &= \left(\frac{C_{i+1,j}^n - 2C_{i,j}^n + C_{i-1,j}^n}{(\Delta r_a)^2}\right) + \frac{1}{(r_a)_i}\left(\frac{C_{i+1,j}^n - C_{i-1,j}^n}{2(\Delta r_a)}\right) \\ &+ \frac{1}{(r_a)_i^2}\left(\frac{C_{i,j+1}^n - 2C_{i,j}^n + C_{i,j-1}^n}{(\Delta\theta)^2}\right) - \frac{(G_r)_{i,j} C_{i,j}^n}{(r_a)_i} \\ &- (G_r)_{i,j}\left(\frac{C_{i+1,j}^n - C_{i-1,j}^n}{2(\Delta r_a)}\right) - \left(\frac{\partial G_r}{\partial r_a}\right)_{i,j} C_{i,j}^n \\ &- \frac{(G_\theta)_{i,j}}{(r_a)_i}\left(\frac{C_{i,j+1}^n - C_{i,j-1}^n}{2(\Delta\theta)}\right) - \frac{C_{i,j}^n}{(r_a)_i}\left(\frac{\partial G_\theta}{\partial\theta}\right)_{i,j}.\end{aligned} \quad (5)$$

For "special" element, the corresponding difference equation has the form as

$$\begin{aligned}\frac{C_{s,j}^{n+1} - C_{s,j}^n}{\Delta\tau} &= \frac{1}{(r_a)_s^2}\left(\frac{C_{s,j+1}^n - 2C_{s,j}^n + C_{s,j-1}^n}{(\Delta\theta)^2}\right) \\ &- \frac{(G_\theta)_{i,j}}{(r_a)_s}\left(\frac{C_{s,j+1}^n - C_{s,j-1}^n}{2(\Delta\theta)}\right) - \frac{C_{s,j}^n}{(r_a)_s}\left(\frac{\partial G_\theta}{\partial\theta}\right)_{s,j} \\ &+ \frac{(G_r)_{s,j} C_{s,j}^n}{\Delta r_a} - \left(\frac{C_{s+1,j}^n - C_{s,j}^n}{\Delta r_a}\right),\end{aligned} \quad (6)$$

where the functions $G_r = (aV_r/D) + (aF_{mr}/k_B T)$ and $G_\theta = (aV_\theta/D) + (aF_{m\theta}/k_B T)$ and $\tau = Dt/a^2$ is the normalized time.

3.1. Simulation Algorithm

3.1. Simulation Algorithm. In the simulation, two two-dimensional arrays of equal size named *old_c* and *new_c* are declared for saving the concentrations $C_{i,j}^n$ and $C_{i,j}^{n+1}$ at the old and new instantaneous times t_n and t_{n+1}, respectively. The numbers of row and column of each array are $360/\Delta\theta$ and $[(r_{aL}-1)/\Delta r_a]+1$, respectively. Figure 3 shows a concentration array of maximal column and row indices as $i_{\max}$ and $j_{\max}$, respectively, where $i_{\max}$ = total column − 1 and $j_{\max}$ = total row − 1.

The steps of simulation can be summarized as shown in Algorithm 1.

The size of normalized time step $\Delta\tau$ must be small enough to provide stable numerical results. The size of $\Delta\tau$ in this work is assigned to satisfy Courant stability condition as follows:

$$\Delta\tau \le \frac{1}{G_{\max}\sqrt{1/(\Delta r_a)^2 + 1/(\pi * \Delta\theta/180)^2}}, \quad (7)$$

where $G_{\max}$ is the maximal magnitude of G_r and G_θ which is spatial function but time independent. When stable condition (7) is satisfied, the order of error of computed concentration is $[(\Delta\tau/2) + (\Delta r_a)^2/12 + (\pi * \Delta\theta/180)^2/12]$.

```
(1) Assign initial concentration $C_0$ to every elements in array $old\_c$ and $new\_c$.
(2) while (time < end_time) do
    {
      $\tau_{n+1} = \tau_n + \Delta\tau$ /* Increase normalized time */
      Loop1: FOR ($0 \le j \le j_{\max}$) /* For all discrete angle $\theta_j$ */
      {  Begin: Find column index "$s$" of "special" element.
         Check the value $\theta_j = j * \Delta\theta$.
         IF ($\theta_j$ is on the upstream side) DO
         {  /* Assign upstream boundary condition */
            $new\_c[j][i_{\max}] = C_0$.
            Apply (5) to compute $new\_c[j][i]$ for
            $s + 1 \le i \le i_{\max} - 1$.
            Apply (6) to compute $new\_c[j][s]$ which is
            the special element }
         ELSE IF ($\theta_j$ is on the downstream side) DO
         {  Apply (6) to compute $new\_c[j][s]$ which is
            the special element.
            Apply (5) to compute $new\_c[j][i]$ for
            $s + 1 \le i \le i_{\max} - 1$.
          /* Assign downstream boundary condition */
              $new\_c[j][i_{\max}] = old\_c[j][i_{\max} - 1]$. }
          /* Assign array $new\_c$ to $old\_c$ for
              computing at next discrete time */
         Loop2:    FOR ($0 \le j \le j_{\max}$)
         Loop3:    FOR ($0 \le i \le i_{\max}$) {$old\_c[j][i] = new\_c[j][i]$ }.
      end:} /* End of simulation */
(3) Save concentration results in all elements.
```

Algorithm 1

3.2. Parallel Approach. In this work, we use openMP platform to parallelize the above algorithm. To simplify this, the challenge of the algorithm is twofold.

(1) The domain size can be large due to the domain size of Δr_a and $\Delta\theta$. If $\Delta\theta = 0.0087$ radian or 0.5 degree, we obtain that the number of rows for array C is 720. For various domain size and Δr_a, the number of columns for array C can be large. Thus, this is a big piece of shared memory considering openMP approach.

(2) The number of iterations is more due to $\Delta\tau$. The value should be small enough so that the simulation result is stable. When the number of iterations is large, it could take hours to days to finish the simulation.

Due to the first issue, there is no dependency in computing each element in array C. We use double buffer to store the current C (new_c) and the C in the previous iteration (old_c). Each element $C_{i,j}$ can be computed independently by (5)-(6). Thus, for simplicity, we divide the work sharing using row-wise method. Each thread works on rows of computing $new\_c[i][j]$.

Depending on the openMP schedule used, the work rows may be distributed to threads statically or dynamically. For example, we may try with the clause *static* with $chunk\_size = N/T$ where N is the total number of rows and T is the number of threads which assigns the work statically. Or we may try with the clause *static* with $chunk\_size = 1$ where the row number is assigned to the thread $N \bmod T$ statically. Similarly, the clause *dynamic* with $chunk\_size = 1$. One row is assigned to compute by the thread and when the thread finishes, it will compute on the next row. The row assignment is done dynamically. Each thread may not get the same number of rows to compute on.

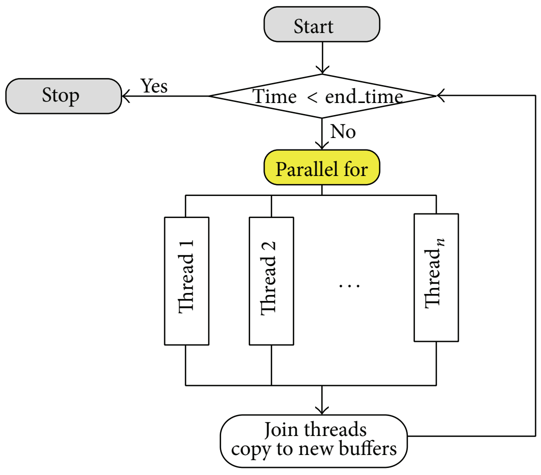


Figure 4: Threads are created based on each angle.

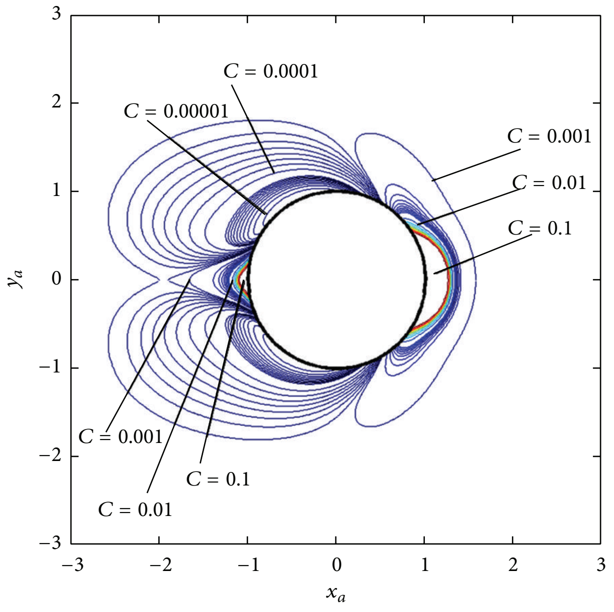


Figure 5: The concentration contours of $Mn_2P_2O_7{\cdot}3H_2O$ nanoparticles of 24 nm diameter at $t = 10.24$ s.

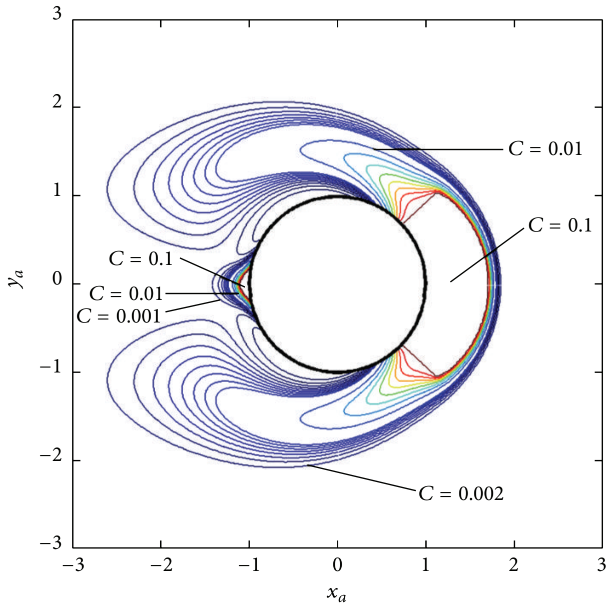


Figure 6: The concentration contours of $Mn_2P_2O_7{\cdot}3H_2O$ nanoparticles of 24 nm diameter at $t = 34.12$ s.

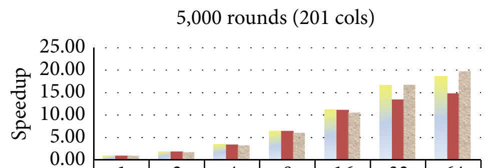
The algorithm for computing the simulation is similar to the above one except that the line marked by Loop1 is the *parallel for*. Also, all the terms are private variables except the concentration arrays (*new_c* and *old_c*). Figure 4 presents how we fork the parallel threads. The Loop1 corresponds to the *parallel for*. Each thread computes the rows it is responsible for based on the schedule approach. That is, it performs the code in line "Begin" until line "End" in the algorithm. After all the threads finish, the *new_c* is copied and they can proceed to the next time step.

4. Results

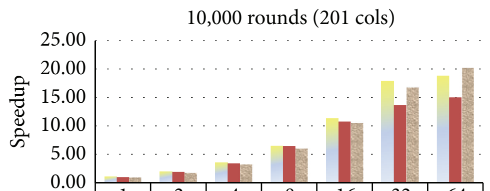
We present the results in two parts. The first part is the results of the correctness of the simulation approach and the second part is the performance of the parallel computation on the varying factors such as the domain size and the iteration numbers.

4.1. Simulation Results. In this work, we simulate concentration dynamics of weakly magnetic particle which has known magnetic susceptibility value. Simulation parameters are obtained from the previously reported work of Davies and Gerber [8]. We consider manganese pyrophosphate ($Mn_2P_2O_7{\cdot}3H_2O$) particles of 24 nm diameter dispersed in water that provides $\chi_p - \chi_f = +2.03\times10^{-3}$. The ferromagnetic wire has 50 micron radius and magnetization of 8.61×10^5 A/m. Water with volume concentration $C_0 = 0.0010$ of $Mn_2P_2O_7{\cdot}3H_2O$ particle flows traverse to the wire with incoming velocity of 1×10^{-5} m/s. The external uniform magnetic field $H_0 = 1 \times 10^7$ A/m. The saturation or maximal volume concentration in any element is limited at $C_{\mathrm{sat}} = 0.10$. The radial and angular steps for partitioning the simulation domain are $\Delta r_a = 0.010$ and $\Delta\theta = 0.0087$ radian or 0.5 degree. The normalized time step is $\Delta\tau = 1 \times 10^{-5}$ which is small enough to obtain stable results. The normalized radius at the outer boundary, r_{aL}, which indicates domain size, is varied between 3, 5, and 9. From the value of Δr_a and $\Delta\theta$, the number of rows in concentration array is 720 and the numbers of columns are 201, 401, and 801. The maximal row index is $j_{\max} = 719$ while the maximal column indices are $i_{\max} = 200, 400$, and 800 for each case.

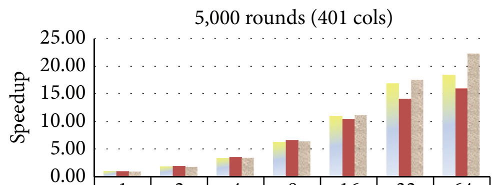
Figure 5 shows the contours of particle volume concentration at time $t = 10.24$ s which is obtained after 7500 rounds of concentration updating. The figure shows the contours only within the area of $r_a \le 3.0$ because the concentration in farther region is almost smooth at $C = C_0 = 0.001$ and no significant concentration gradient is observed. The pattern of the contours is compared with previously reported results of Davies and Gerber [8] using $r_{aL} = 3.0$ and we found similar pattern of concentration contours. In Figure 4, the saturation concentration $C_{\mathrm{sat}} = 0.1$ where particles accumulate densely exists in the horizontal direction parallel to the directions of $\mathbf{H}_0$ and $\mathbf{V}_0$ indicated in Figure 1. Consequently, particle concentration can be increased 100 times from 0.001 to 0.1. The result shows that HGMS technique can concentrate even weakly magnetic nanoparticles. Consequently, it is feasible to increase the concentration of other weakly magnetic nanoparticles with lower magnetic susceptibility value. Some examples of these particles are nanodrug carriers with gold core which are synthesized for using in cancers and tumors therapies nowadays. The magnetic susceptibility of gold particle larger than 10 nm is in the order of 10^{-5} which is about two orders of magnitude lower than that of $Mn_2P_2O_7{\cdot}3H_2O$ in this work so the concentration of gold nanodrug carriers



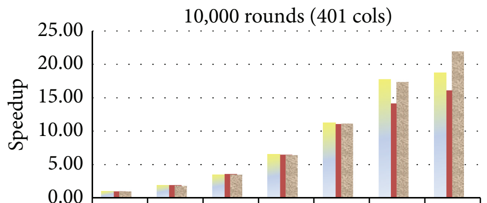
Number of threads	1	2	4	8	16	32	64
🟨 Speedup (static) (5,000 Rds) (201 cols)	1.00	1.90	3.50	6.52	11.29	16.60	18.65
🟥 Speedup (static N/T) (5,000 Rds) (201 cols)	1.00	1.91	3.47	6.50	11.19	13.50	14.85
▨ Speedup (dyn) (5,000 Rds) (201 cols)	1.00	1.78	3.25	6.04	10.55	16.68	19.67



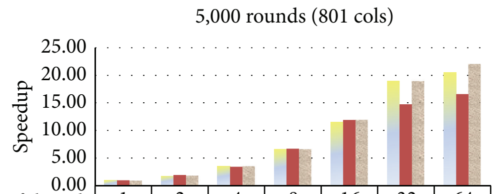
Number of threads	1	2	4	8	16	32	64
🟨 Speedup (static) (10,000 Rds) (201cols)	1.00	1.90	3.50	6.49	11.27	17.86	18.83
🟥 Speedup (static N/T)	1.00	1.90	3.40	6.45	10.76	13.65	15.01
▨ Speedup (dyn) (10,000 Rds) (201 cols)	1.00	1.79	3.29	6.05	10.52	16.72	20.21



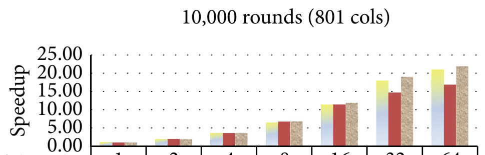
Number of threads	1	2	4	8	16	32	64
🟨 Speedup (static) (5,000 Rds) (401 cols)	1.00	1.83	3.33	6.23	10.98	16.90	18.40
🟥 Speedup (static N/T) (5,000 Rds) (401 cols)	1.00	1.92	3.56	6.64	10.45	14.10	15.96
▨ Speedup (dyn) (5,000 Rds) (401 cols)	1.00	1.85	3.47	6.45	11.19	17.53	22.32



Number of threads	1	2	4	8	16	32	64
🟨 Speedup (static) (10,000 Rds) (401 cols)	1.00	1.92	3.49	6.52	11.27	17.78	18.71
🟥 Speedup (static N/T) (10,000 Rds) (401 cols)	1.00	1.91	3.60	6.49	11.05	14.14	16.09
▨ Speedup (dyn) (10,000 Rds) (401 cols)	1.00	1.83	3.48	6.40	11.10	17.31	21.93



Number of threads	1	2	4	8	16	32	64
🟨 Speedup (static) (5,000 Rds) (801 cols)	1.00	1.77	3.50	6.66	11.55	19.01	20.54
🟥 Speedup (static N/T) (5,000 Rds) (801 cols)	1.00	1.93	3.43	6.71	11.92	14.73	16.60
▨ Speedup (dyn) (5,000 Rds) (801 cols)	1.00	1.91	3.62	6.69	12.00	18.98	22.09



Number of threads	1	2	4	8	16	32	64
🟨 Speedup (static) (10,000 Rds) (801 cols)	1.00	1.87	3.55	6.44	11.29	17.99	20.98
🟥 Speedup (static N/T) (10,000 Rds) (801 cols)	1.00	1.94	3.51	6.69	11.42	14.70	16.84
▨ Speedup (dyn) (10,000 Rds) (801 cols)	1.0	1.9	3.6	6.8	11.8	18.9	21.9

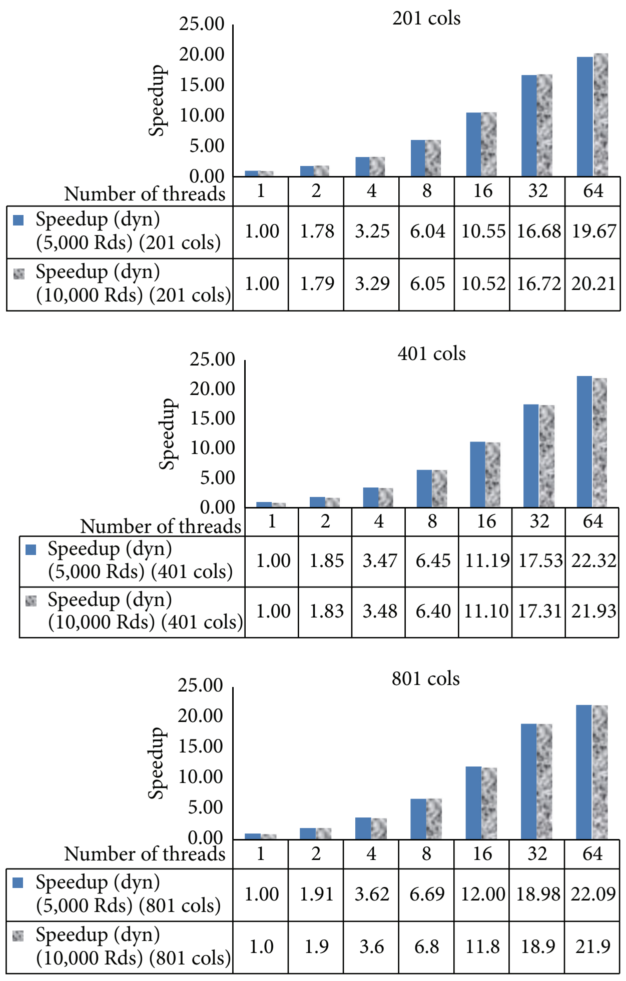
FIGURE 7: The speedup for the varying number of columns, 201, 401, and 801, and varying number of iterations, 5,000 and 10,000 iterations, varying schedule types.

may not be increased 100 times. However, increasing of drug carriers concentration from only 2 to 4 times is sufficient for therapy process so HGMS is very a interesting technique to increase the effectiveness of cancers or tumors therapies. It is also observed that the region of accumulation of the particles, $C > C_0$, occurs within the region of $r_a \leq 2.0$ which is very close to wire surface. This means that particles dispersed outside of this region are not captured on the wire. Consequently, capture efficiency can be increased by using many wires arranged in some configurations with distance between wires about 2 times of their radius.

Figure 6 shows the contours of particle volume concentration at time $t = 34.12$ s which is obtained after 25,000 rounds of concentration updating. The saturation region on the upstream side is larger while no significant change of saturation region on the downstream side is observed. The pattern of simulated particle accumulation is in good agreement with the experimental results collected in the book of Gerber and Birss [7].

4.2. Performance Results. In the experiments, we test the simulation time on Intel Xeon Phi 7110P located at Kasetsart University, Thailand. It has 61 cores at 1.1 GHz, with 64-bit addressing running Linux 2.6* and gcc 4.4.7. It contains the same two MICs. We test the simulation time based on the two issues above: the domain size and the total iterations. The domain size is varied in the maximal column indices as $i_{\max} = 200$, 400, and 800 for each case. The maximal row index is $j_{\max} = 719$. The numbers of iterations tested are 5,000 and 10,000. In the experiments, we have tried more numbers of iterations but the results are quite stable when the iteration is closed to 10,000.

Figure 7 compares varying schedule types of openMP in varying the domain sizes and the iteration numbers. We

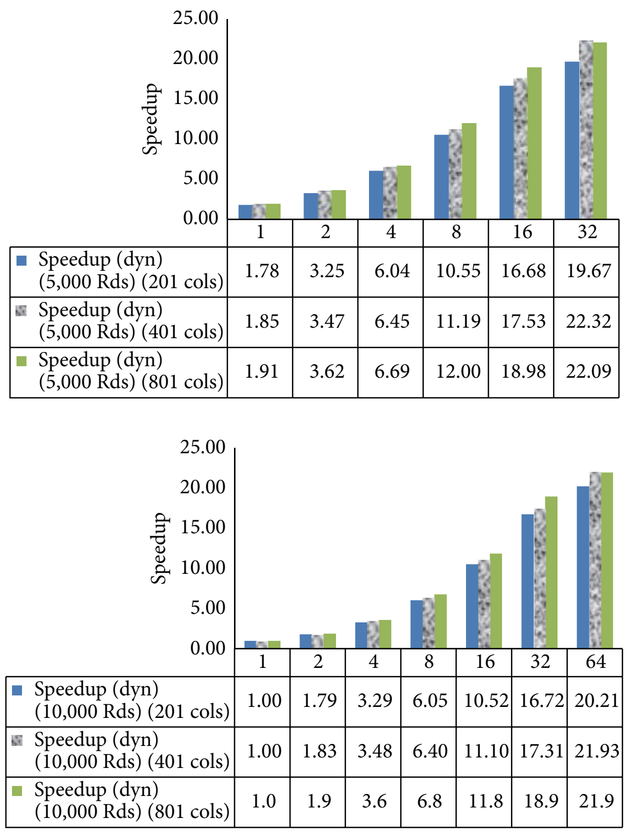


Number of threads	1	2	4	8	16	32	64
🟦 Speedup (dyn) (5,000 Rds) (201 cols)	1.00	1.78	3.25	6.04	10.55	16.68	19.67
▨ Speedup (dyn) (10,000 Rds) (201 cols)	1.00	1.79	3.29	6.05	10.52	16.72	20.21

Number of threads	1	2	4	8	16	32	64
🟦 Speedup (dyn) (5,000 Rds) (401 cols)	1.00	1.85	3.47	6.45	11.19	17.53	22.32
▨ Speedup (dyn) (10,000 Rds) (401 cols)	1.00	1.83	3.48	6.40	11.10	17.31	21.93

Number of threads	1	2	4	8	16	32	64
🟦 Speedup (dyn) (5,000 Rds) (801 cols)	1.00	1.91	3.62	6.69	12.00	18.98	22.09
▨ Speedup (dyn) (10,000 Rds) (801 cols)	1.0	1.9	3.6	6.8	11.8	18.9	21.9

FIGURE 8: The speedup for the case number of columns, 201, 401, and 801 with the dynamic schedule varying number of iterations, 5,000 and 10,000 iterations.



	1	2	4	8	16	32
🟦 Speedup (dyn) (5,000 Rds) (201 cols)	1.78	3.25	6.04	10.55	16.68	19.67
▨ Speedup (dyn) (5,000 Rds) (401 cols)	1.85	3.47	6.45	11.19	17.53	22.32
🟩 Speedup (dyn) (5,000 Rds) (801 cols)	1.91	3.62	6.69	12.00	18.98	22.09

	1	2	4	8	16	32	64
🟦 Speedup (dyn) (10,000 Rds) (201 cols)	1.00	1.79	3.29	6.05	10.52	16.72	20.21
▨ Speedup (dyn) (10,000 Rds) (401 cols)	1.00	1.83	3.48	6.40	11.10	17.31	21.93
🟩 Speedup (dyn) (10,000 Rds) (801 cols)	1.0	1.9	3.6	6.8	11.8	18.9	21.9

FIGURE 9: The speedup for 5,000 and 10,000 iterations, dynamic schedule varying number of columns.

test for static schedule with *chunk_size* = 1 (static), static schedule with *chunk_size* = N/T (static N/T), and dynamic schedule with *chunk_size* = 1 (dynamic). It is clear that the dynamic schedule outperforms the static ones in every case. This is because the work on each row is not the same. For some angle, the number of points to compute may decrease when the numbers of iterations are more. The density of the concentration is less when the location is farther away from the core. It is noted that when we double the number of threads, we obtain almost double speedup for each case until 16 threads. When the number of threads is 32, or more, the speedup is reduced since there may be too many threads for the number of rows (720). Then, the thread overhead is more. We obtain less speedup.

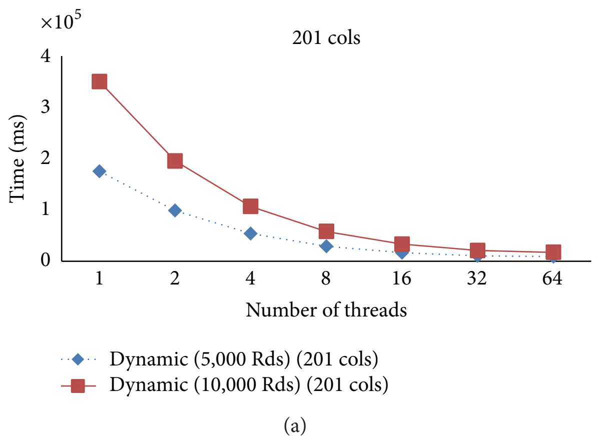
For 5,000 iterations, 1 thread, 98,585 milliseconds is used while for 10,000 iterations, 1 thread, 195,396 milliseconds is used. From Figure 8, when we double the number of iterations, the speedup goes in the same manner when doubling the number of threads with the dynamic schedule although perhaps some case like 32 and 64 threads yields a little bit more speedup.

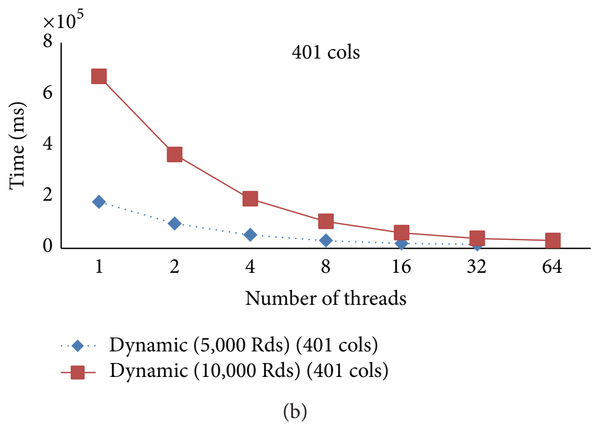
Figure 9 presents the speedup for 5,000 and 10,000 iterations when increasing the domain size by the double. Again, the approach still gives the similar speedup when the domain size is double from 201 to 401 and from 401 to 801 when the number of threads is double.

Figure 10 presents the computation time for the 5,000 and 10,000 iterations. It shows the trends of the reduction of the computation time for each case. Similarly, in Figure 11, we show the trend of the reduction in computation when the domain size is double.

For further analysis, we study the workload for each thread at each iteration. We observe that the number of points calculated for each thread is very much stable for every 1,000 iterations. For the first 1,000 iterations, all threads have about the same amount of points calculated. As the iteration goes, the concentration values become steady at some points. Thus, the thread responsible for the rows containing more of these points has less work load. Therefore the static schedule is not best performed. With the dynamic schedule, the thread that finishes fast due to this reason will be able to perform computation in other rows. Figures 12–14 show graphs related to the issues.

Figure 12 shows how the total elements needed to be calculated are reduced as the iterations pass by. We show this for 80,000 iterations. It is seen that the number of points is





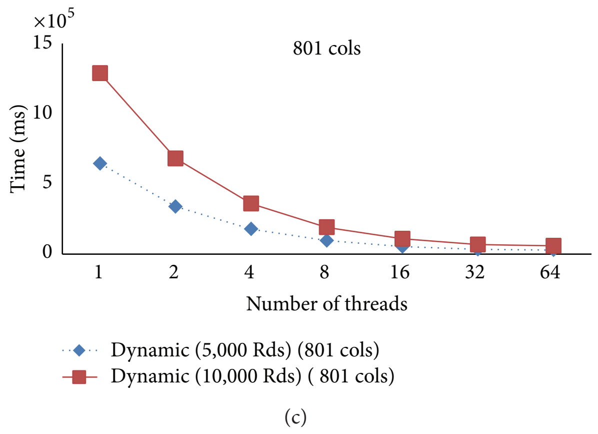
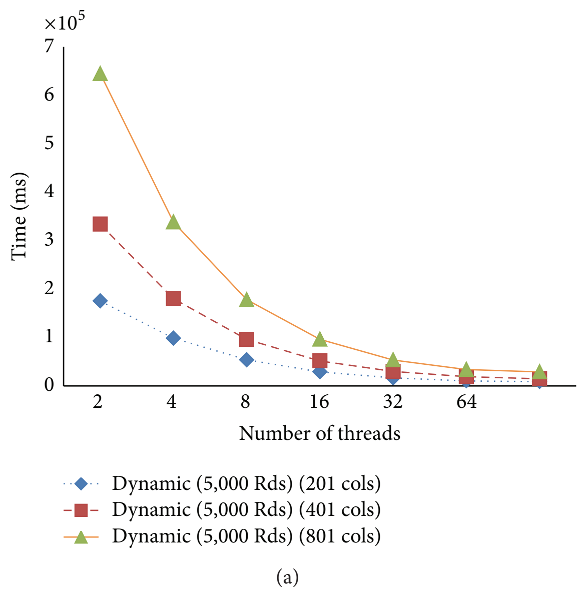


FIGURE 10: The computation time for the case numbers of columns, 201, 401, and 801, dynamic schedule varying number of iterations.



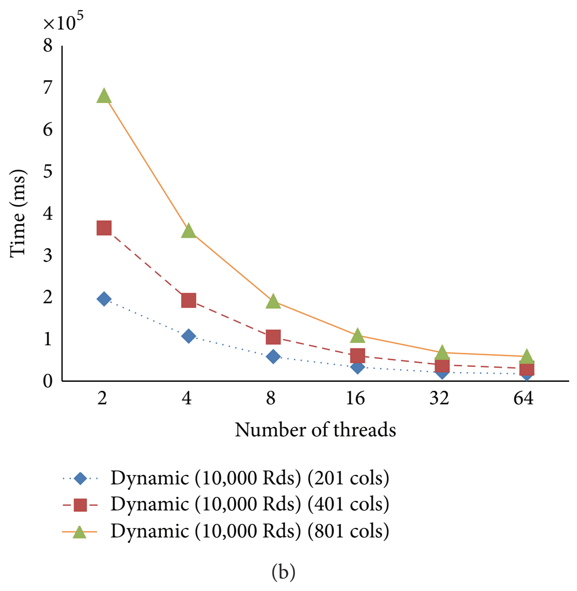
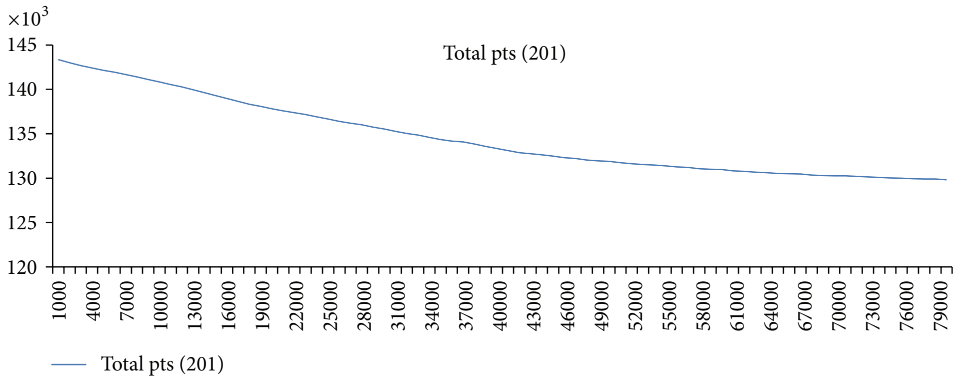
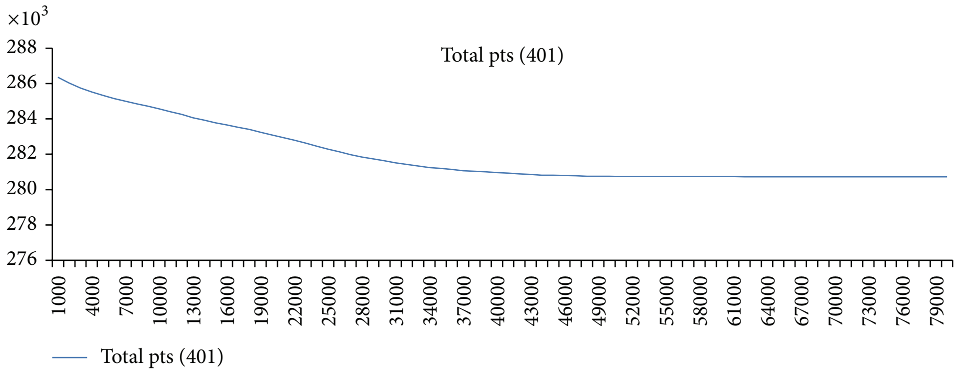


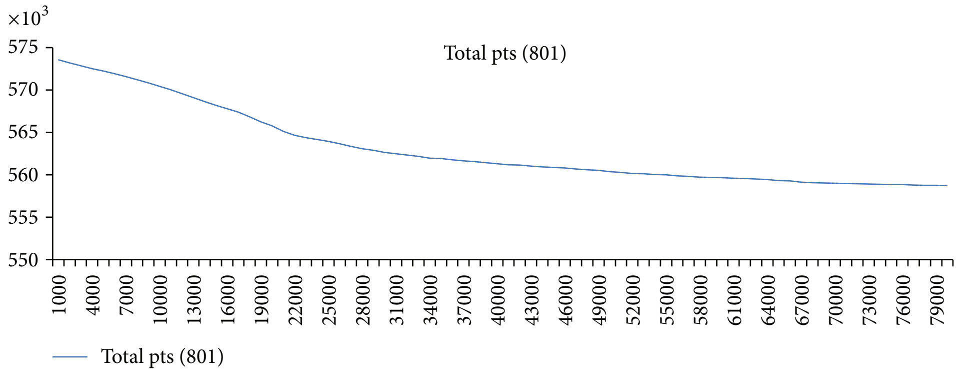
FIGURE 11: The computation time for 5,000 and 10,000 iterations, dynamic schedule varying number of columns.



(a)



(b)



(c)

Figure 12: Total points calculated for every 1,000 iterations for the case of 201, 401, 801 column size.

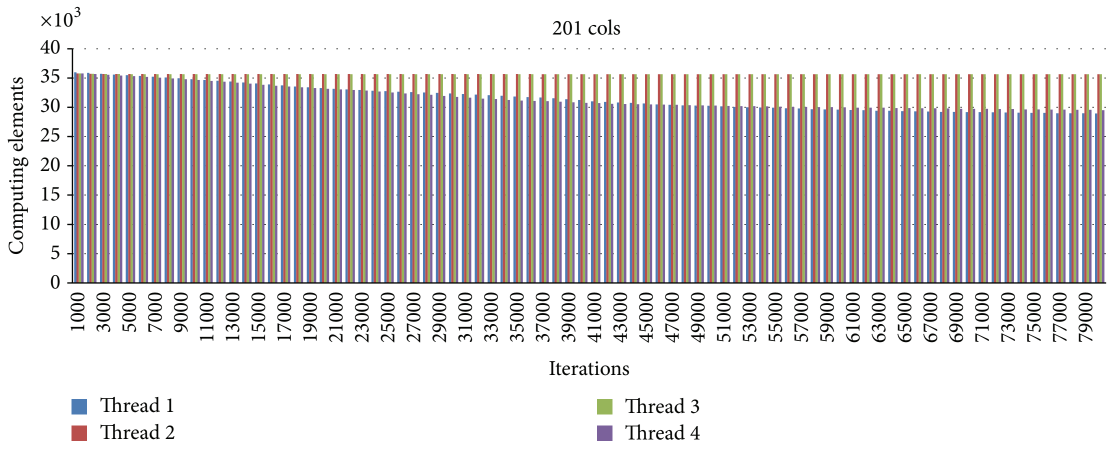
gradually reduced. In 80,000 iterations, about 10,000 points are reduced for every case.

Figure 13 presents the workload for static schedule using *chunk_size* equal to the number of rows divided by the number of threads. These examples are for 4 threads. It is seen that the 2nd and 3rd threads have more points to be calculated since the works are divided by rows. This is because the special points are more in Quadrants 1 and 4. Compare to Figure 14. With the dynamic schedule, the threads obtain the row as it finishes. The workload distribution is quite random. With this nature of steady points, the task division by other methods such as column-wise or block-wise may also be investigated in the future.

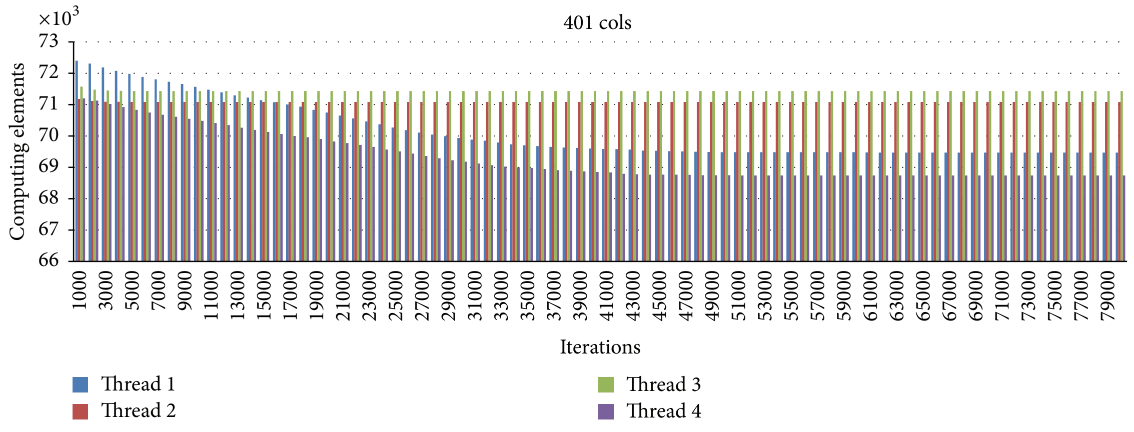
5. Conclusions

In this work, we present the process of high gradient magnetic separation (HGMS) using microferromagnetic wire for capturing the weakly magnetic nanoparticles in the irrotational flow of inviscid fluid. The parallel simulation using openMP is proposed. The domain problem is viewed in two-dimensional domains of wire radius and annular step and with the magnetic fore and fluid flow. The timing factors are the size of granularity and the iterations performed. Both also affect the correctness of the simulation process.

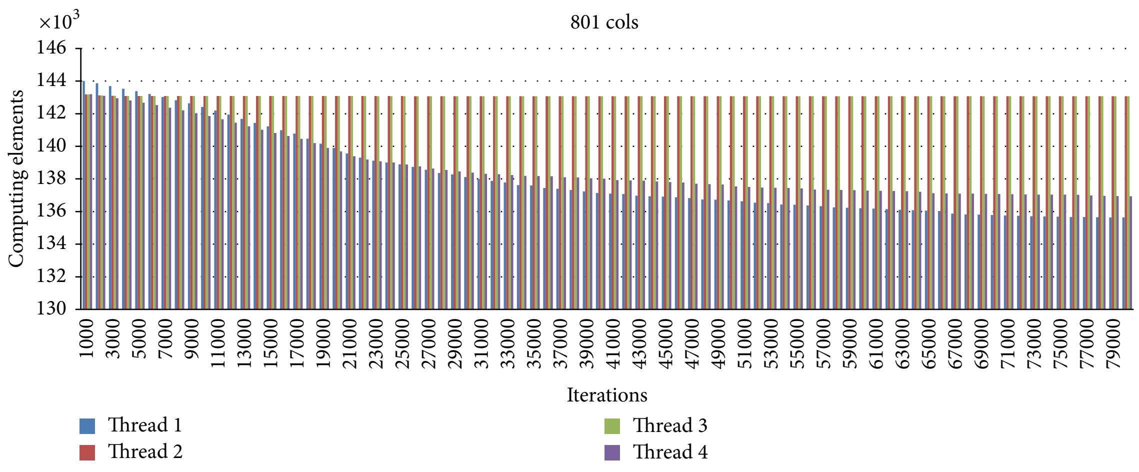
To parallelize it, each concentration element can be computed independently. Each thread computes rows of each



(a)



(b)

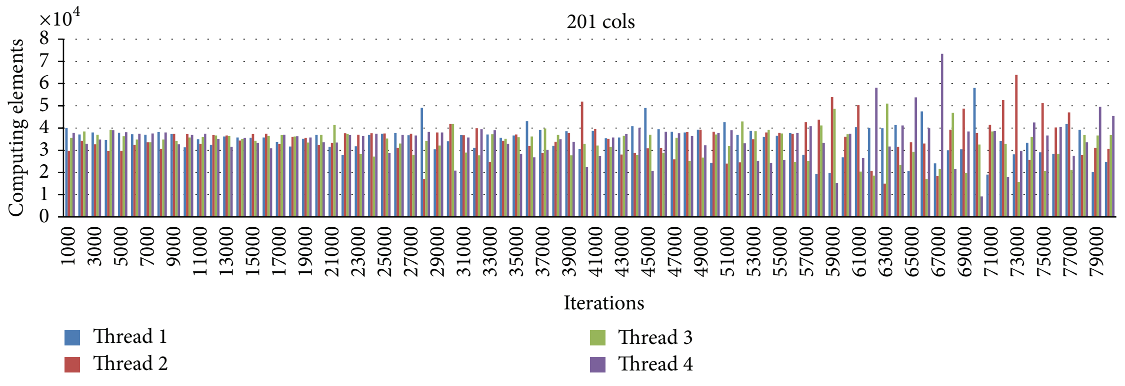


(c)

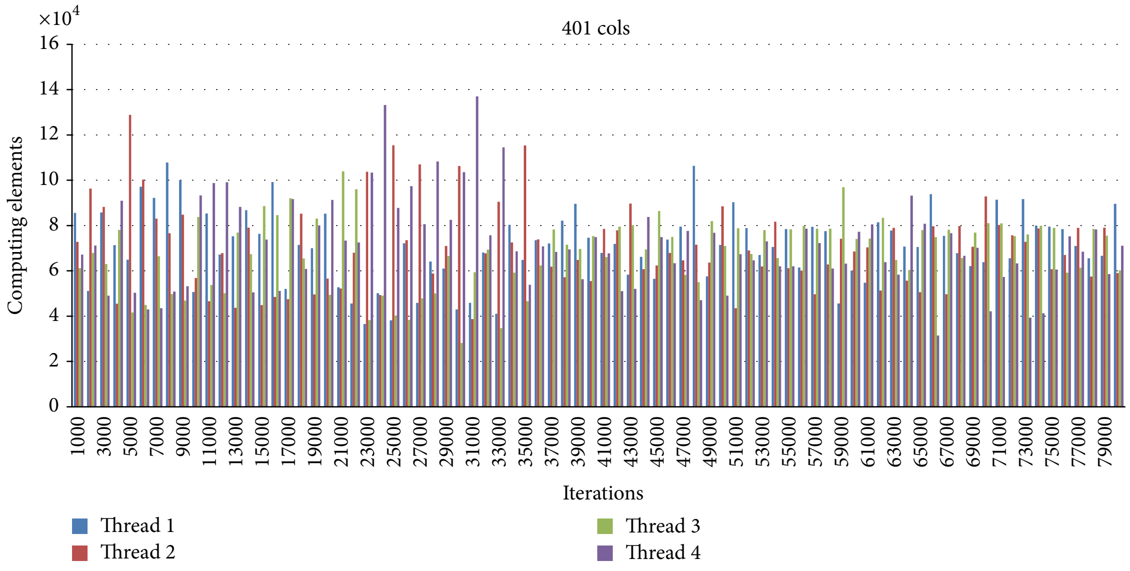
FIGURE 13: The workload of each thread for 4 threads using static schedule *chunk_size* = N/T.

concentration. The varying schedule types are considered. From the experiments, the dynamic schedule is very suitable since, along the iterations, the number of computed elements is less when the simulation moves gradually to the stable state. Also, with the parallelization by angle, we obtain double speedup when the number of threads goes double. This is consistent for the case when we double the domain size and double number of iterations. For the study of 720 rows, the number of threads should be 16 for the optimum speedup.

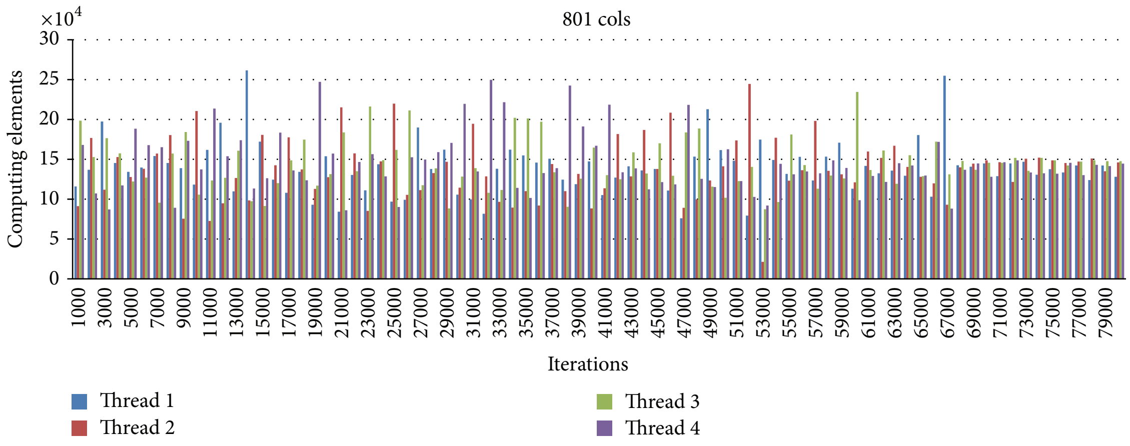
In the future, if the domain is very large, we can distribute the domain using MPI approach, and within each MPI node, the proposed openMP method can be used. With the MPI,



(a)



(b)



(c)

FIGURE 14: The workload of each thread for 4 threads using dynamic schedule *chunk_size* = 1.

the concentration array is divided with overlapped rows [9]. The synchronization is needed between iterations before going to the next iteration. The nonblocking communication may be considered to hide this latency [10].

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work is supported in part by the Thailand Research Fund through the Royal Golden Jubilee Ph.D. Program, Contract no. PHD/0275/2551, and Silpakorn University Research and Development Institute for the funding support, Contracts nos. SURDI 55/01/18 and SURDI 55/01/01. The authors are grateful to Intel Many Core Testing Lab under the collaboration of Intel Co. Ltd. and HPCNC, Kasetsart University, Thailand, for the testing and the experiments.

References

[1] K. H. Han and A. B. Fraizer, "Continuous magnetophoretic separation of blood cells in microdevice format," *Journal of Applied Physics*, vol. 96, no. 10, pp. 5797–5802, 2004.

[2] S. C. Bhakdi, A. Ottinger, S. Somsri et al., "Optimized high gradient magnetic separation for isolation of Plasmodium-infected red blood cells," *Malaria Journal*, vol. 9, no. 1, article 38, 2010.

[3] A. Merino-Martos, J. de Vicente, L. Cruz-Pizarro, and I. de Vicente, "Setting up high gradient magnetic separation for combating eutrophication of inland waters," *Journal of Hazardous Materials*, vol. 186, no. 2-3, pp. 2068–2074, 2011.

[4] J. A. Ritter, A. D. Ebner, K. D. Daniel, and K. L. Stewart, "Application of high gradient magnetic separation principles to magnetic drug targeting," *Journal of Magnetism and Magnetic Materials*, vol. 280, no. 2-3, pp. 184–201, 2004.

[5] P. J. Cregg, K. Murphy, and A. Mardinoglu, "Calculation of nanoparticle capture efficiency in magnetic drug targeting," *Journal of Magnetism and Magnetic Materials*, vol. 320, no. 23, pp. 3272–3275, 2008.

[6] A. Mardinoglu, P. J. Cregg, K. Murphy, M. Curtin, and A. Prina-Mello, "Theoretical modelling of physiologically stretched vessel in magnetisable stent assisted magnetic drug targetingapplication," *Journal of Magnetism and Magnetic Materials*, vol. 323, no. 3-4, pp. 324–329, 2011.

[7] R. Gerber and R. R. Birss, *High Gradient Magnetic Separation*, John Wiley & Sons, Chichester, UK, 1983.

[8] L. P. Davies and R. Gerber, "2-D simulation of ultra-fine particle capture by a single-wire magnetic collector," *IEEE Transactions on Magnetics*, vol. 26, no. 5, pp. 1867–1869, 1990.

[9] K. Hournkumnuard, C. Phongpensri, S. Keinprapai, O. Wongtaveesap, and S. Gorlatch, "Parallel algorithm and visualization of high gradient magnetic separation of nanoparticles," *International Journal of Computers and Applications*, vol. 33, no. 1, pp. 70–82, 2011.

[10] C. Chantrapornchai, B. Dolwithayakul, K. Huankumnerd, and S. Gorlatch, "Parallel mass transfer simulation of nanoparticles using nonblocking communication," *International Journal of Multimedia and Ubiquitous Engineering*, vol. 5, no. 1, pp. 41–52, 2010.