CALIFORNIA STATE UNIVERSITY, NORTHRIDGE

Performance Optimization of Numerical Solution Algorithm for Microchannel Flow Boiling

A thesis submitted in partial fulfilment of the requirements

For the degree of Master of Science in Mechanical Engineering

By

Matt Blomquist

December 2017

(Official end of the semester)

Copyright by Matt Blomquist 2017

California State University, Northridge

The thesis of Matt Blomquist is approved:

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Dr. Abhijit Mukherjee |  | Date |
|  |  |  |
| Dr. <NAME> |  | Date |
|  |  |  |
| Dr. <NAME> |  | Date |

Table of Contents

Signature Page

Acknowledgements

Dedication

List of Tables

List of Figures

Abstract

Chapter 1: Introduction

Chapter 2: Objective

Chapter 3: Mathematics

Chapter 4: Numerical Model

Chapter 5: Results

Chapter 6: Conclusion

References

Abstract

Performance Optimization of Numerical Solution Algorithm for Microchannel Flow Boiling

By

Matt Blomquist

Master of Science in Mechanical Engineering

In recent years, there has been significant advancements computation hardware that allows for massive parallelism in computational fluid dynamics algorithms. However, as academia and industry increase the size and complexity of simulations computational performance needs to be enhanced through algorithmic parallelism to avoid significant bottlenecks in solution runtime. In this study, a microchannel flow boiling algorithm is investigated for potential improvements in runtime via algorithmic parallelism. A parametric study has been performed to quantify the impact of multiple Krylov-Subspace solution methods for linear systems using a variety of preconditioning methods chosen to improve speed of convergence. These solutions methods are then compared with solutions from state of the art software packages such as ANSYS, NASTRAN, and COMSOL. The highest impact solution algorithms are then implemented into the microchannel flow boiling algorithm and compared with the serial, iterative solution methodology. The results show a significant decrease in computational duration while resulting in improved accuracy and stability. Additionally, the improved microchannel flow boiling algorithm is tested on a variety of parallel hardware architectures to identify potential scalability.

# Introduction

Since the 1960s, Computational Fluid Dynamics (CFD) has been a tool implemented by scientists and engineers to solve a variety of complex problems related to fluid flow, heat and mass transfer, chemical reactions, and related phenomena. In general, CFD is a numerical solution methodology that is primarily associated with making predictions of heat transfer and fluid-flow processes that do not have exact solutions.

The aerospace industry was an early adopter of CFD, primarily using the technology to study the effects of drag and lift on aerodynamic bodies. For aircraft manufacturers like Boeing, Lockheed Martin, and Northrup Grumman, CFD provided a tool to significantly reduce costs associated with running experimental investigations. Simple mistakes during the design of aircraft were found prior to developing prototype which reduced the lead time for new designs and reduced development program costs by approximately X%.

Typical difficulties in the early days of CFD… cost of computer time, etc.

In recent years, CFD has been a tool to study microscale phenomena that is difficult to experiment on and requires significant investment on specialized equipment. Microchannel flow boiling is one of the aforementioned phenomena that heavily use CFD. Microchannel flow boiling explores two-phase fluid flow in channels with a diameter the size of a human hair.

Microchannel flow boiling and microchannel heat transfer is of general research importance today because the electronic industry is pursuing miniaturization. As the devices get smaller, the requirements to dissipate heat increase by a square of the area reduction. Microchannel heat sinks offer a possible solution to that issue. People have seen huge improvements with both experiments and numerical work, but more needs to be understood about the dynamics of bubbles in these heat sinks to create sustainable and reliable solutions.

My research team has been hard at work exploring these phenomena, but we are significantly limited by the arcane nature of our code. For most scenarios, our microchannel boiling numerical algorithm will run for approximately 4 days before generating enough information for the group to get the data we need. Changing and upgrading our hardware has only been able to provide slight improvements to the overall run time. The overall accuracy of the algorithm is very good as seen in multiple papers that have been published by our group, but the sluggish performance (total duration of runtime) limits our ability to perform analysis and research.

Extremely long runtimes are typical for many CFD applications where accuracy is paramount. NASA is most notable for creating these extremely intensive CFD simulations that can take months to complete. However, NASA has also been very proactive in creating solutions for these runtimes. The typical ideas they come up with are changing hardware, changing algorithms, and using parallel processing.

Improving hardware or using better hardware to solve computationally expensive simulations is a very common approach. However, in most hardcore research areas this solution doesn’t typically yield significantly beneficial results. In industry, where less computationally intensive workloads can be readily found, this can be a very good option.

In early years, using external servers were the only way to solve some of these problems. Places like NASA AMES and Lawrence Livermore Labs were homes to massive supercomputers used specifically for these types of problems. However, with the capability expansion of personal computers, a lot of smaller organizations were able to achieve acceptable results for purchased machines. Only within the last few years has server offloading come back into style in the form of cloud computing. New cloud computing services offer huge benefits for reasonable prices and provide normal people with access to ridiculously powerful machines. Still, a computationally expensive algorithm ends up wasting time which is money, so we will explore improvements to the algorithms themselves.

Pantankar revolutionized the numerical computation world when he developed an algorithm for solving the Navier-Stokes equations. This method was called the SIMPLE method or Semi-Implicit Method for Pressure Linked Equations. This algorithm combined the momentum and energy equations of Navier-Stokes with the continuity equation. Therefore, it was magic and allowed for numerical solution to complex fluid dynamic problems.

Pantankar developed a method using the Thomas algorithm for solving a system of linear equations that he deemed the line-by-line TDMA (tri-diagonal matrix algorithm) method. For uni-dimensional problems, this algorithm provides direct results. However, when solving 2- and 3-dimensional problems, this algorithm becomes an iterative method that provides approximate results. To solve the 2- and 3-dimensional equations, sweeps need to be made line-by-line through the higher dimension grids. This method is the time consuming part of CFD.

In recent years, however, the world of linear algebra has come up with some very elegant methods of solving these types of systems. Krylov developed a number of really awesome sub-space methods for solving large, sparse linear systems that look almost identical to the equations that arise from partial differential equations. PDEs are, in essence, what the Navier-Stokes equations are, so these methods can be readily applied to these problems. In some year, the GMRES method came out and was quickly integrated into a number of commercial software packages for CFD. This method provided a huge improvement and people were very happy. In some other year, the Bi-Conjugate Method was created based on the success of the Conjugate Gradient method that was used for other types of PDEs. There were even fancy projects that came up with software tools that would choose the best solution method so the researchers didn’t have too. Some of these are PARDISO and Mathworks MLDIVIDE algorithm.

In the industrial community, it is often preferable to use very general tools that can be applied to multiple different problems with ease. For this reason, tools like ANSYS, COMSOL, and NASTRAN use algorithms like PARDISO and others for ease of use. In research groups, however, a lot of software will be created to solve very specific problems with very high levels of accuracy. So these groups end up choosing algorithms like GMRES or BiCGStab as they fit the particular problem. In the end, it doesn’t even matter that much. All of these algorithms can be further improved by incorporating parallelism.

Parallel solution algorithms have been around since the 60s and CFD people have been really interested in them since the start of CFD. Parallel solutions can scale with hardware which creates really exciting possibilities for improving speed. The improvements in the form of parallel processing come from two main avenues: improvements in hardware and improvements to algorithms.

The Intel Corporation has been really diligent working on HPC solutions for many years and nearly dominates the market for high-performance CPUs. One of their main developments in the last decade has been vectorization for multicore machined. Vectorization is a simplistic form of parallelization where data gets lined up on the chip and processed at the same time. For independent-type algorithms, this can work great. It’s like using an FPGA in electronics. However, for algorithms with serial dependencies it doesn’t offer much benefit.

One really cool thing about this is that a number of basic libraries for linear algebra programming (linear system solutions) offer intrinsic vectorization. The Intel MKL library is made primarily for this reason and requires little to no change in existing code to generate huge improvements. Additionally, things like BLAS (Basic Linear Algebra Subprograms) and LAPACK (Linear Algebra Package) have been modified over the years to take advantage of parallelism.

Additionally, new developments in hardware have created graphics processing units that computer entirely in parallel. Year-over-year, these GPUs increase in capability and created a huge area of research with potential benefit to numerical performance.

Nvidia is the world’s largest supplier of GPU technology and have been actively working with researches in the area of high performance computing. CFD is part of HPC and a huge amount of effort has been involved with creating CFD code that works primarily on GPUs. In the typical SIMPLE method, the linear system solvers will be off-loaded from the CPU to the GPU where parallel computations can take place. In recent studies, people have seen up to 16x speed up from highly optimize code. When compared to the line-by-line TDMA, data doesn’t exist. We will provide the comparison here though.

Finally, there have been a number of improvements in linear system solution algorithms that provide greater portability to parallel processing. GMRES and BiCGStab have been modified to be run in parallel. The BLAS and LAPACK libraries have also been updated to be run on massively parallel machines. New algorithms like SPIKE have also been created specifically for parallel processing. On top of all that, pre-conditioners for linear systems have been parallelized and new methods have been created to make super iterative solvers.

# Objective

The objective of this study is to optimize the performance of a numerical solution algorithm for microchannel flow boiling problems using parallelism. The particular microchannel flow boiling algorithm that will be investigated has previously been demonstrated in literature [ref previous publications here] and is a primary candidate as it almost entirely written as a serial algorithm. Previous publications utilizing this microchannel flow boiling algorithm demonstrate a high-accuracy in predicting experimental results [ref] and, being mostly serial in nature, provides a solid foundation for comparison with a parallel alternative. The performance indicators for this study are (1) solution accuracy, (2) computational duration, and (3) algorithm stability.

As mentioned in the previous chapter, increasing the complexity of a CFD solution can significantly increase the computational runtime. In previous uses of the microchannel flow boiling algorithm that is studied herein and in referenced literature, the primary bottleneck of CFD algorithms is the solution of massive sparse linear systems. Following the SIMPLER method, the solution of these systems of equations occurs, at minimum, twice. The first instance is to solve the XXXX block. The second instance is to solve the XXXX block. Additional solution routines are also used if the energy or species equation of Navier-Stokes is required. Since these solution methodologies appear multiple times in the algorithm and dominate computational runtime, the linear solvers are the most significant bottleneck.

In order to optimize the runtime and accuracy of the microchannel flow boiling problem, the following linear equation solution methodologies will be parametrically studied:

* Line-By-Line TDMA
* Bi-Conjugate Gradients (BiCG)
* Bi-Conjugate Gradients Stabilized (BiCGStab)
* General Minimum Residual (GMRES)
* PARDISO

These solution algorithms have all been extensively studied in linear algebra, but there has not been a significant review of their abilities in the scope of computational fluid dynamics. Furthermore, their applicability to microchannel flow boiling has not been determined in an extensive review of literature. The PARDISO algorithm is a proprietary algorithm that is typically used in commercial software packages such as ANSYS, COMSOL, and NASTRAN. The PARDISO algorithm, therefore, provides a reference point for the solution speed of commercial software and will be utilized as such.

In addition to a variety of linear equation solution algorithms, a number of preconditioning methodologies will be investigated. Pre-conditioning algorithms determine starting points for the Krylov methods referenced above. A good pre-conditioning algorithm has been seen to improve convergence for these algorithms by up to XX%. However, when pre-conditioning algorithms are ill-fitted, algorithm stability can be compromised. The particular preconditioning algorithms that will be explored are:

* Jacobi Precondition
* Successive Over-Relaxation
* Algebraic Multi-grid

Finally, each of these algorithms and preconditioners will be tested in serial, parallel, and massively parallel (GPU specific) configurations. The serial method will be completed as it provides a reference point for the serial nature of the existing microchannel solution. The parallel method will be optimized to use vectorization on a single CPU and the massively parallel methodology will be optimized for use on GPU hardware. The comparison between vectorization and GPU computation will provide a reference point as hardware configurations can significantly differ.

The algorithm with the most significant improvement will then be implemented into the microchannel flow boiling algorithm and compared with the current line-by-line TDMA in terms of accuracy, runtime, and stability. Two different hardware configurations will be used for testing, one of which has GPU compute capabilities. The comparison of these test cases will serve as the optimization of the microchannel flow boiling algorithm.

# Mathematics

# Governing Equations

# Computational fluid dynamics looks at solving the Navier-Stokes equations from a numerical perspective. The Navier-Stokes equations look like this: [do I need to go through and derive these?]

# These equations are differential equations and the numerical approach to solve them involves discretizing them. For the microchannel flow boiling problem, we have utilized a central differencing scheme. There are a lot of other ways to discretize and more information can be found here [some standard text book reference]. Central differencing has error like this [] which is really good for being accurate.

# This discretization leads to a system of equations that looks something like this. These need to be solved with some known boundary conditions and initial conditions. These boundary equations can be applied to the linear system by incorporating them into the solution vector. Once this is done, all that needs to happen is solving for the unknown. This method in a single dimension looks like this:

# For higher dimensions, there are a few potential ways to do this and it depends on how you are going to solve the problem. One way is to use a line by line technique which will create a diffusive effect between lines to reach convergence. This method will be discussed later. Another method is to incorporate all of the equations into a single linear system. A comparison of these two types of systems can be seen below and will be discussed later.

# Part of solving the Navier-Stokes equations involves solving multiple equations simultaneously. The continuity equation takes care of the pressure aspect and the momentum equations consider the fluid flow. There are multiple methods for incorporating these two sets of equations including the SIMPLE, SIMPLER, and PISO. The SIMPLER algorithm is the most common approach and is used in this paper. More information on the other methodologies will not be discussed by extensive information can be found here [reference other methods for pressure linking].

# SIMPLER Method

# Computational fluid dynamics is a numerical simulation approach to solving systems that primarily involve fluid flow and heat transfer. Simulations are generated from the discretization of partial differential equations and a number of high-level algorithms have been developed to iteratively solve the Navier-Stokes equations. For the microchannel flow boiling simulations discussed herein, the SIMPLER algorithm has been implemented.

# The SIMPLER algorithm starts by utilizing an initial guess for the pressure field, velocity field, and energy field. From these guesses, pseudo-velocities are computed and fed into the pressure equation. The pressure equation is solved and the initial guess pressure field is replaced by the pressure field. Next, the discretized momentum equations are solved and the pressure correction equation is solved. Next, the velocities are corrected with the pressure correction factor and all other discretized transport equations are solved. At this point, the solution checks for convergence. If convergence is not achieved, the process starts over with the pressure, velocity, and transport variables set as the initial guess. If convergence is achieved, the process stops. A figure of this layout can be seen below.

# < figure 6.7 versteeg >

# The SIMPLER algorithm implements a linear solver at Steps 2, 3, 4, and 6. In the case of microchannel flow boiling, the linear solver is called 6 times (for U, V, and W velocities). Therefore, making an improvement to this area of the algorithm can generate up to a 6x benefit for each time-step.

# The baseline linear solver used in the microchannel flow boiling simulation is the line-by-line TDMA. The line-by-line TDMA (3D-TDMA for shorthand) solves a three-dimensional system of equations by iteratively solving one-dimensional systems. These one-dimensional systems represent a line of nodes through the computational domain. For one-dimensional systems, the TDMA is able to directly solve for unknown values in a serial manner. A number of parallel versions of the TDMA have been implemented in recent years, but their improvement in speed is typically low for small numbers of variables.

# Algorithm Comparison

# In recent studies, parallel implementations of the TDMA show significant increases in computational performance for large systems of equations (order of E10). However, the performance cost of the parallel implementation out weights the serial algorithm for smaller systems. The computational domain of the microchannel flow boiling simulation has a maximum dimensional length of 400 nodes. This translates to a linear system with 398 unknowns, with is far less than the performance inflection point of the parallel TDMA implementations. Therefore, by analysis there would be no benefit to using a parallel implementation of the TDMA for the microchannel flow boiling simulation.

# In order to use the TDMA for a three-dimensional problem, each one-dimensional line of the computational grid is solved and the results are diffused into the equations of neighboring lines. This line-by-line method was originally utilized in CFD by Pantankar, but the method results in an extremely slow process of convergence. In order to improve the results of this method, the sweeping direction of the lines can be varied based on the physics of the simulation. For example, if fluid is moving from the x=0 position to the x=400 direction, it would be ideal to sweep along the axis. This sweeping methodology increases the diffusion of unknown terms by leveraging the TDMA in the primary gradient direction. Additionally, there are a number of parallel methods that can be implemented to further increase the rate of convergence.

# The line-by-line TDMA can be improved with a simple instance of parallelism. By solving lines in parallel, the computation time can be significantly reduced. However, the diffusive properties of the algorithm need to be examined to ensure that the number of cycles for convergence does not exponentially expand. This exponential expansion in convergence steps is due to isolation of the individual line solutions. Simply put, solving the lines in parallel can adversely affect the diffusive properties of the algorithm. It is important to conserve the diffusive property of the algorithm while distributing computational load with parallel processing. This can be achieved by properly selecting the directionality of the parallel solutions. For example, starting a single thread in the top, left, north corner and a second thread in the bottom, south, right corner would reduce the restriction on diffusion.

# This type of algorithm has been explored in the last decade and has shown various runtime improvements on massively parallel clusters of computers. The Thomas algorithm has a linear complexity therefore it does not require a significant amount of memory overhead. Additionally, each line solution has a small amount of data required for transfer between compute nodes. This tends to yield some improvements to the base code. However, the number of convergence steps does increase which reduces the impact of parallelism.

# A more significant improvement in runtime can be achieved by completely replacing the line-by-line TDMA method with a standard linear algebra method. In recent years, the Conjugate Gradient algorithm has been an intense algorithm of study and optimization for positive semi-definite matrix equations. Unfortunately, most CFD equations do not yield these types of differential equations. In XXXX date, Krylov (or some other dude) came up with a conjugate method for non-positive semi-definite matrix equations. This technique can be applied to CFD problems.

# The Bi-Conjugate Gradient method is similar to the CJ, with the exception that a few steps are added to convert the matrix into positive form. Talk some more about bi-conjugate gradient methods. Additionally, improvements have been made to this algorithm to improve the stability which has yielded the Bi-Conjugate Gradients Stabilized method.

# The Generalized Minimum Residual method has also been used in many commercial software packages for its improvements in runtime. The GMRES algorithm works like this… In many tests the GMRES algorithm performs really well and demonstrates a lot of stability.

# The BiCGStab method is a hybrid approach that mixes two Krylov methods, the BiCG algorithm with the GMRES algorithm. This stuff at the end significantly improves algorithm stability and results in even faster speed up for generalized types of problems. When considering the best replacement of the line-by-line TDMA, stability is important. Therefore, it will be one of the algorithms to consider.

# Commercial software packages like ANSYS and COMSOL tend to use a proprietary algorithm called PARDISO. This algorithm will examine the matrices that are input and come up with the best potential for solution. Many of the internal solutions have similar structures as GMRES and BiCGStab, but they are proprietary. We don’t exactly know what goes on inside the code. However, since these are widely available and utilized in many commercial packages, this algorithm provides a baseline for the state-of-the-art.

# In order to determine the best potential solution algorithm the following algorithms were developed and tested in a steady state, three-dimensional heat transfer simulation. This type of simulation was chosen for its simplicity to create and implement. It is also important to note that solving the three-dimensional heat transfer problem is identical to solving the three-dimensional convection problem. This is because the coefficient matrix for the linear system is identical in shape between these two cases. The shape is derived from the differencing scheme used and both problems use central differencing scheme.

# The test setup for this parametric study utilizes a cubic geometry for simplicity. Using different geometries could improve the convergence rate of the TDMA, but a cubic geometry will provide the most generalized case. [Note: It could be interesting to use different geometries to show the benefits of running TDMA in the best direction]. Additionally heat conduction problems were solved in one-dimension and two-dimensions to show how the Krylov methods can significantly outperform TDMA as dimensionality increases.

# Both MATLAB and FORTRAN were used for testing. MATLAB uses built in versions of these algorithms (not-TDMA). It was primarily used to get a good first approximation of the differences. FORTRAN was used to test the maximum speed that can be achieved with any serial version of the algorithm. FORTRAN is compiled language and tend to operate much faster than scripted counterparts like MALTAB. Here is some recent research from <some research group> that shows the difference in computational speed for standard problems.

# One-Dimensional Diffusion

# A one-dimensional diffusion problem was solved with a heat flux boundary condition applied to the west side and a constant temperature boundary condition applied to the east side.

# Two-Dimensional Diffusion

# A two-dimensional diffusion problem was solved with a constant temperature boundary condition and a constant heat flux boundary condition. The constant temperature was applied to the north side of a two-dimensional grid and given a temperature of 100 °C. The heat flux boundary condition was applied to the west wall and given a heat flux of 500 W/m2. The nodal mesh was generated as a square and the number of nodes was varied from 10 to 100 in increments of 10 (i.e. 10, 20, 30, …100).

# < figure of BC step up >

# The following linear system solution algorithms were each used to solve the system of equations using the MATLAB scripting environment as well as in compiled FORTRAN:

# Tri-diagonal matrix algorithm (TDMA)

# Biconjugate Gradients (BiCG)

# Biconjugate Gradients Stabilized (BiCGStab)

# General Minimum Residual (GMRES)

# Pardiso (Intel MKL)

# For small grid sizes, the TDMA performed nearly as well as the Pardiso and MLDIVIDE algorithms. As the grid sizes increased, however, the TDMA convergence time significantly increased. For all solution algorithms, the compiled FORTRAN environment achieved the lowest time of convergence, with the Intel MKL Pardiso algorithm outperforming the others be a factor of 1.4 times.

# < figure results >

# With Jacobi-preconditioning applied to each of the algorithms (aside from the TDMA), the results are slightly improved for both the MATLAB and FORTRAN environments.

# < figure results with jacobi >

# With Incomplete LU-Decomposition (ILU), the convergence rates for BiCGStab and GMRES are nearly identical to the rates of convergence for the Intel MKL Pardiso and MATALB MLDIVIDE algorithms.

# < figure results with ilu >

# From the results of the two-dimensional case, it appears that the MLDIVIDE and Intel MKL Pardiso algorithms are optimized for near-symmetric matrices. This agrees with the wide-spread uses of these algorithms in typical commercial software packages such as ANSYS, NASTRAN, and COMSOL.

# Three-Dimensional Diffusion

# A three-dimensional diffusion problem was solved in a similar configuration to the three-dimensional problem. A constant temperature condition was applied to the north wall and a heat flux boundary condition was applied to the west wall. The values for these boundary conditions were the same as in the two-dimensional problem, 100 °C and 500 W/m2. Additionally, the nodal mesh was kept in a cubic configuration with m, n, and l all having the same number of nodes. These values were varied from 10 to 400 with intervals of 10 until 100 nodes per side, then in intervals of 100 to 400 nodes per side.

# < figure of BC setup >

# The following linear system solution algorithms were each used to solve the system of equations using the MATLAB scripting environment as well as in compiled FORTRAN:

# Tri-diagonal matrix algorithm (TDMA)

# Biconjugate Gradients (BiCG)

# Biconjugate Gradients Stabilized (BiCGStab)

# General Minimum Residual (GMRES)

# Pardiso (Intel MKL)

# Each of the tested algorithms performed significantly faster than the TDMA for all cases when considering the number of iterations and computational run time. The Pardiso and MLDIVIDE algorithms were also significantly slower than the biconjugate gradient stabilized method as the number of nodes increased.

# < figure results >

# With Jacobi-preconditioning applied to each of the algorithms (aside from the TDMA), the results are slightly improved for BiCGStab and GMRES.

# < figure results with jacobi >

# With ILU preconditioning, the convergence rates for BiCGStab and GMRES perform significantly better than the Intel Pardiso and MLDIVIDE algorithm.

# < figure results with ILU >

# From the results of the three-dimensional case, it appears that the MLDIVIDE and Intel MKL Pardiso algorithms not optimized for matrices that are not symmetric.

# In most CFD type problems, data storage becomes very important. For most simulations, it is common to use double precision values which take up 64 bits or 8 bytes. As solutions become more and more complex and grid sizes increase, double precision values can eat a ton of memory and hard drive space. For example, a three-dimensional problem with a grid size of 10x10x10 requires a minimum 0.288 MB of memory to solve a single linear system. Since a three dimensional problem is comprised of 12 of these, 3.456 MB is required to move through the simpler method. As the grid increases to 100x100x100, 3,456 GB would be required to solve the problem. The line-by-line TDMA method can reduce this number to 3.5 GB since only line solutions are solved at one time. For linear system methods, vector-compression methods have been developed to reduce the size of matrices where there are zero-values. By compressing the matrix, the total memory requirement can be dropped to 55.5 MB.

# The compression methods used in this study are the CRS method with stands for Compressed Row Sparse. It works like this [ref Saad].

# Numerical Model

# The numerical model of the microchannel flow boiling simulation has a computational domain of 3.96x0.99x0.99, non-dimensional units and can be seen in Figure 1. The inlet of the channel is placed at x=0.00 and the fluid outlet is at 3.96. The nucleation cavity is placed on the bottom plane at 0.00. The simulation starts with a small nucleation point to create a repeatable location for bubble growth and optimize the computational duration. This is performed in previous studies as well.

# The channel length is divided into 64, 128, and 256 non-dimensional units for this study which yields 16,384, 131,072, and 1,048,576 total nodes in the simulation respectively. Multiple grid sizes are used to perform a grid-independency study and demonstrate the trend in computational duration between linear system algorithms. In previous studies, the 128 unit grid was used as it optimized computational time and minimized the computational error.

# The complete incompressible Navier-Stokes equations are solved using the SIMPLER method metioned above. The continuity equations is turned into an equation for pressure correction and a pressure field is derived from the know velocity field. Each iteration, the velocities are corrected using velocity-correction formulas. The computations proceed to convergence via a series of continuity satisfying velocity fields. The TDMA and BiCGStab methods are used to solve the algorithm. The TDMA is further improved by supplementing it with a block-correction procedure and the BiCGStab algorithm is improved with a SSOR pre-conditioning technique. A multi-grid approach is further added to the algorithms to solve the pressure equations.

# A level-set approach is used to smooth the two-phase boundary which was initially developed by Sussman et al. []. The level set function eliminates problems of adding and subtracting nodes to create a moving grid and simulates the merging and breaking of the interface. Furthermore, the level set formulation is easily generalized for three-dimensional problems.

# The liquid vapor interface is identified as the zero level set of a smooth distance function 𝜙. The level set function 𝜙 is positive outside the vapor bubble and negative inside the vapor bubble. The interface is located by solving the level set equation. A fifth order WENO (Weighted, Essentially Non.Oscillatory) scheme is used for left sided and right-sided discretization of 𝜙 [18]. While 𝜙 is primarily a distance function, it will not remain so after solving the level set equation. Maintaining 𝜙 as a distance function is necessary for providing the interface with a width fixed in time. This is achieved by re-initialization of 𝜙. A modification of Godunov's method is used to determine the upwind directions. The re-initialization equation is solved in fictitious time after each fully complete time step. With Δτ =d/2u0, ten τ steps are taken with a third order TVD (Total Variation Diminishing) Runge Kutta method.

# The bubble is placed at x\*=0.99, y\*=0 and z\*=0, with 0.1𝑙0 radius in the domain. All velocities at the initial grid points are set to zero. The liquid inlet temperature is set to 102 oC. The vapor inside the bubble is set to the saturation temperature of 100 oC. The initial liquid temperature inside the domain is set equal to the inlet liquid temperature of 102 oC. The contact angle at the walls is set as specified later in the text.

# The boundary conditions are as follows:

# At the inlet (x\*=0):

# 𝑢=𝑢0; 𝑣=𝑤=0; 𝑇=𝑇𝑖𝑛; 𝜙𝑥=0 (18)

# Constant inlet flow velocity has been specified in the numerical calculations. In parallel microchannel heat exchangers inlet flow velocity is necessary to maintain stable operating conditions, which can be achieved using flow restrictions at the inlet [17].

# At the outlet (x\*=3.96)

# 𝑢𝑥=𝑣𝑥=𝑤𝑥=𝑇𝑥=0; 𝜙𝑥=0 (19)

# At the plane of symmetry (z\*=0)

# 𝑢𝑧=𝑣𝑧=𝑤𝑧=𝑇𝑧=0; 𝜙𝑧=0 (20)

# At the walls (y\* = 0, y\*=0.99)

# 𝑢=𝑣=𝑤=0; 𝑇=𝑇𝑤; 𝜙𝑦=−cos𝜑 (21)

# where 𝜑 is the contact angle.

# At the wall (z\*=0.495)

# 𝑢=𝑣=𝑤=0; 𝑇=𝑇𝑤; 𝜙𝑧=−cos𝜑 (22)

# Flow Regime and Reynolds Number

# In this study, boiling of liquid-water in a square cross-section microchannel is considered. The cross-sectional dimension of the square microchannel is chosen as 226×226 μm. The flow regime of the fluid in the microchannel is laminar and Reynolds numbers of 50, 250 and 500 have been chosen which are typical of microchannels.

# Wall Superheats and Bubble Contact Angles

# Constant wall superheats of 5 °C, 10 °C, 15 °C and 20 °C have been selected in this study. This range of wall superheat is selected based on the experimental results in the literature. Also, three sets of contact angles have been selected which are 60-50, 40-30 and 20-10 degrees. The first and second numbers in each set represents the advancing and receding contact angles, respectively. These three sets accounts for different surface wettability of the boiling surface which has a significant effect on the bubble dynamics as discussed in Khalighi et al. [18].

# The algorithms used for this study consist of the line-by-line TDMA, BiCGStab in serial, BiCGStab in vector form, and BiCGStab in parallel that utilizes a GPU. These algorithms are specifically incorporated at this point in the SIMPLER algorithm.

# For this study, the following hardware configurations were used for reference.

# Results

# Conclusion

References

[Note: This is just for sizing need to change these]

Abu-Nada, E. (2008). "Application of nanofluids for heat transfer enhancement of separated flows encountered in a backward facing step." International Journal of Heat and Fluid Flow **29**(1): 242-249.

Abu-Nada, E., Z. Masoud, et al. (2008). "Natural convection heat transfer enhancement in horizontal concentric annuli using nanofluids." International Communications in Heat and Mass Transfer **35**(5): 657-665.

Abu-Nada, E., K. Ziyad, et al. (2008). "Heat Transfer Enhancement in Combined Convection Around a Horizontal Cylinder Using Nanofluids." Journal of Heat Transfer **130**(8): 084505-4.

Afshar, H., M. Shams, et al. (2009). "Microchannel heat transfer and dispersion of nanoparticles in slip flow regime with constant heat flux." International Communications in Heat and Mass Transfer **36**(10): 1060-1066.

Ahn, H. S., H. Kim, et al. (2010). "Experimental study of critical heat flux enhancement during forced convective flow boiling of nanofluid on a short heated surface." International Journal of Multiphase Flow **36**(5): 375-384.

Bhattacharya, P., A. Samanta, et al. (2009). "Numerical study of conjugate heat transfer in rectangular microchannel heat sink with Al2O3/H2O nanofluid." Heat and Mass Transfer **45**(10): 1323-1333.

Bianco, V., F. Chiacchio, et al. (2009). "Numerical investigation of nanofluids forced convection in circular tubes." Applied Thermal Engineering **29**(17-18): 3632-3642.

Brinkman, H. C. (1952). "The Viscosity of Concentrated Suspensions and Solutions." The Journal of Chemical Physics **20**(4): 571-571.

Buongiorno, J. (2006). "Convective Transport in Nanofluids." Journal of Heat Transfer **128**(3): 240-250.

Chandrasekar, M. and S. Suresh (2009). "A Review on the Mechanisms of Heat Transport in Nanofluids." Heat Transfer Engineering **30**(14): 1136 - 1150.

Choi, S. U. S. (2008). "Nanofluids: A new field of scientific research and innovative applications." Heat Transfer Engineering **29**(5): 429-431.

Choi, S. U. S. (2009). "Nanofluids: From Vision to Reality Through Research." Journal of Heat Transfer **131**(3): 033106-9.

Coursey, J. S. and J. Kim (2008). "Nanofluid boiling: The effect of surface wettability." International Journal of Heat and Fluid Flow **29**(6): 1577-1585.

Daungthongsuk, W. and S. Wongwises (2007). "A critical review of convective heat transfer of nanofluids." Renewable and Sustainable Energy Reviews **11**(5): 797-817.

Dominguez-Ontiveros, E., S. Fortenberry, et al. (2010). "Experimental observations of flow modifications in nanofluid boiling utilizing particle image velocimetry." Nuclear Engineering and Design **240**(2): 299-304.

Duangthongsuk, W. and S. Wongwises (2010). "Comparison of the effects of measured and computed thermophysical properties of nanofluids on heat transfer performance." Experimental Thermal and Fluid Science **34**(5): 616-624.

Eastman, J. A., S. U. S. Choi, et al. (2001). "Anomalously increased effective thermal conductivities of ethylene glycol-based nanofluids containing copper nanoparticles." Applied Physics Letters **78**(6): 718-720.

Elena, V. T. and et al. (2010). "Particle size and interfacial effects on thermo-physical and heat transfer characteristics of water-based Î±-SiC nanofluids." Nanotechnology **21**(21): 215703.

Godson, L., B. Raja, et al. (2010). "Enhancement of heat transfer using nanofluids--An overview." Renewable and Sustainable Energy Reviews **14**(2): 629-641.

Haghshenas Fard, M., M. N. Esfahany, et al. (2010). "Numerical study of convective heat transfer of nanofluids in a circular tube two-phase model versus single-phase model." International Communications in Heat and Mass Transfer **37**(1): 91-97.

Henderson, K., Y.-G. Park, et al. (2010). "Flow-boiling heat transfer of R-134a-based nanofluids in a horizontal tube." International Journal of Heat and Mass Transfer **53**(5-6): 944-951.

Ho, C. J., L. C. Wei, et al. (2010). "An experimental investigation of forced convective cooling performance of a microchannel heat sink with Al2O3/water nanofluid." Applied Thermal Engineering **30**(2-3): 96-103.

Hong, H., B. Wright, et al. (2007). "Enhanced thermal conductivity by the magnetic field in heat transfer nanofluids containing carbon nanotube." Synthetic Metals **157**(10-12): 437-440.

Hosseini, M. and S. Ghader (2010). "A model for temperature and particle volume fraction effect on nanofluid viscosity." Journal of Molecular Liquids **153**(2-3): 139-145.

Hosseini, M. S., A. Mohebbi, et al. (2010). "Correlation of Shear Viscosity of Nanofluids Using the Local Composition Theory." Chinese Journal of Chemical Engineering **18**(1): 102-107.

Jung, J.-Y., H.-S. Oh, et al. (2009). "Forced convective heat transfer of nanofluids in microchannels." International Journal of Heat and Mass Transfer **52**(1-2): 466-472.

Kakaç, S. and A. Pramuanjaroenkij (2009). "Review of convective heat transfer enhancement with nanofluids." International Journal of Heat and Mass Transfer **52**(13-14): 3187-3196.

Khanafer, K., K. Vafai, et al. (2003). "Buoyancy-driven heat transfer enhancement in a two-dimensional enclosure utilizing nanofluids." International Journal of Heat and Mass Transfer **46**(19): 3639-3653.

Kim, H., H. S. Ahn, et al. (2010). "On the Mechanism of Pool Boiling Critical Heat Flux Enhancement in Nanofluids." Journal of Heat Transfer **132**(6): 061501-11.

Kim, H. D. and M. H. Kim (2007). "Effect of nanoparticle deposition on capillary wicking that influences the critical heat flux in nanofluids." Applied Physics Letters **91**(1): 014104-3.

Kim, J., Y. T. Kang, et al. (2004). "Analysis of convective instability and heat transfer characteristics of nanofluids." Physics of Fluids **16**(7): 2395-2401.

Kim, S. J., I. C. Bang, et al. (2006). "Effects of nanoparticle deposition on surface wettability influencing boiling heat transfer in nanofluids." Applied Physics Letters **89**(15): 153107-3.

Kim, S. J., I. C. Bang, et al. (2007). "Surface wettability change during pool boiling of nanofluids and its effect on critical heat flux." International Journal of Heat and Mass Transfer **50**(19-20): 4105-4116.

Kim, S. J., T. McKrell, et al. (2008). "Alumina Nanoparticles Enhance the Flow Boiling Critical Heat Flux of Water at Low Pressure." Journal of Heat Transfer **130**(4): 044501-3.

Kim, S. J., T. McKrell, et al. (2009). "Experimental Study of Flow Critical Heat Flux in Alumina-Water, Zinc-Oxide-Water, and Diamond-Water Nanofluids." Journal of Heat Transfer **131**(4): 043204-7.

Kim, S. J., T. McKrell, et al. (2010). "Subcooled flow boiling heat transfer of dilute alumina, zinc oxide, and diamond nanofluids at atmospheric pressure." Nuclear Engineering and Design **240**(5): 1186-1194.

Kim, T. I., Y. H. Jeong, et al. (2010). "An experimental study on CHF enhancement in flow boiling using Al2O3 nano-fluid." International Journal of Heat and Mass Transfer **53**(5-6): 1015-1022.

Kondaraju, S., E. K. Jin, et al. (2010). "Direct numerical simulation of thermal conductivity of nanofluids: The effect of temperature two-way coupling and coagulation of particles." International Journal of Heat and Mass Transfer **53**(5-6): 862-869.

Kwark, S. M., R. Kumar, et al. (2010). "Pool boiling characteristics of low concentration nanofluids." International Journal of Heat and Mass Transfer **53**(5-6): 972-981.

Lee, J. and I. Mudawar (2007). "Assessment of the effectiveness of nanofluids for single-phase and two-phase heat transfer in micro-channels." International Journal of Heat and Mass Transfer **50**(3-4): 452-463.

Li, C. H. and G. P. Peterson (2007). "Mixing effect on the enhancement of the effective thermal conductivity of nanoparticle suspensions (nanofluids)." International Journal of Heat and Mass Transfer **50**(23-24): 4668-4677.

Li, Y., J. e. Zhou, et al. (2009). "A review on development of nanofluid preparation and characterization." Powder Technology **196**(2): 89-101.

Li, Z., W.-Q. Tao, et al. (2006). "A numerical study of laminar convective heat transfer in microchannel with non-circular cross-section." International Journal of Thermal Sciences **45**(12): 1140-1148.

Liu, M.-S., M. Ching-Cheng Lin, et al. (2005). "Enhancement of thermal conductivity with carbon nanotube for nanofluids." International Communications in Heat and Mass Transfer **32**(9): 1202-1210.

Maïga, S. E. B., C. T. Nguyen, et al. (2010). "Heat transfer behaviours of nanofluids in a uniformly heated tube." Superlattices and Microstructures **35**(3-6): 543-557.

Maïga, S. E. B., S. J. Palm, et al. (2005). "Heat transfer enhancement by using nanofluids in forced convection flows." International Journal of Heat and Fluid Flow **26**(4): 530-546.

Meibodi, M. E., M. Vafaie-Sefti, et al. (2010). "The role of different parameters on the stability and thermal conductivity of carbon nanotube/water nanofluids." International Communications in Heat and Mass Transfer **37**(3): 319-323.

Milanova, D. and R. Kumar (2005). "Role of ions in pool boiling heat transfer of pure and silica nanofluids." Applied Physics Letters **87**(23): 233107-3.

Moghadassi, A., S. Masoud Hosseini, et al. (2009). "A model of nanofluids effective thermal conductivity based on dimensionless groups." Journal of Thermal Analysis and Calorimetry **96**(1): 81-84.

Munshi, A. M., V. N. Singh, et al. (2008). "Effect of nanoparticle size on sessile droplet contact angle." Journal of Applied Physics **103**(8): 084315-5.

Murshed, S. M. S., K. C. Leong, et al. (2005). "Enhanced thermal conductivity of TiO2--water based nanofluids." International Journal of Thermal Sciences **44**(4): 367-373.

Murshed, S. M. S., K. C. Leong, et al. (2009). "A combined model for the effective thermal conductivity of nanofluids." Applied Thermal Engineering **29**(11-12): 2477-2483.

Naphon, P., P. Assadamongkol, et al. (2008). "Experimental investigation of titanium nanofluids on the heat pipe thermal efficiency." International Communications in Heat and Mass Transfer **35**(10): 1316-1319.

Narayan, G. P., K. B. Anoop, et al. (2007). "Mechanism of enhancement/deterioration of boiling heat transfer using stable nanoparticle suspensions over vertical tubes." Journal of Applied Physics **102**(7): 074317-7.

Özerinç, S., S. Kakaç, et al. (2010). "Enhanced thermal conductivity of nanofluids: a state-of-the-art review." Microfluidics and Nanofluidics **8**(2): 145-170.

Oztop, H. F. and E. Abu-Nada (2008). "Numerical study of natural convection in partially heated rectangular enclosures filled with nanofluids." International Journal of Heat and Fluid Flow **29**(5): 1326-1336.

Pantzali, M. N., A. G. Kanaris, et al. (2009). "Effect of nanofluids on the performance of a miniature plate heat exchanger with modulated surface." International Journal of Heat and Fluid Flow **30**(4): 691-699.

Park, K.-J. and D. Jung (2007). "Enhancement of nucleate boiling heat transfer using carbon nanotubes." International Journal of Heat and Mass Transfer **50**(21-22): 4499-4502.

Park, K.-J., D. Jung, et al. (2009). "Nucleate boiling heat transfer in aqueous solutions with carbon nanotubes up to critical heat fluxes." International Journal of Multiphase Flow **35**(6): 525-532.

Peng, H., G. Ding, et al. (2009). "Heat transfer characteristics of refrigerant-based nanofluid flow boiling inside a horizontal smooth tube." International Journal of Refrigeration **32**(6): 1259-1270.

Phan, H. T., N. Caney, et al. (2009). "Surface wettability control by nanocoating: The effects on pool boiling heat transfer and nucleation mechanism." International Journal of Heat and Mass Transfer **52**(23-24): 5459-5471.

Roy, G., C. T. Nguyen, et al. (2010). "Numerical investigation of laminar flow and heat transfer in a radial flow cooling system with the use of nanofluids." Superlattices and Microstructures **35**(3-6): 497-511.

Saeid, V. and et al. (2009). "The effect of nanoparticles on the liquid-gas surface tension of Bi 2 Te 3 nanofluids." Nanotechnology **20**(18): 185702.

Santillán, M., F. Membrives, et al. (2008). "Characterization of TiO2 nanoparticle suspensions for electrophoretic deposition." Journal of Nanoparticle Research **10**(5): 787-793.

Taylor, R. A. and P. E. Phelan (2009). "Pool boiling of nanofluids: Comprehensive review of existing data and limited new data." International Journal of Heat and Mass Transfer **52**(23-24): 5339-5347.

Vafaei, S. and et al. (2006). "Effect of nanoparticles on sessile droplet contact angle." Nanotechnology **17**(10): 2523.

Vafaei, S. and D. Wen (2009). Subcooled flow boiling heat transfer of nanofluids in a microchannel. ASME Heat Transfer Summer Conference. San Francisco, California.

Vafaei, S. and D. Wen (2010). "Effect of Gold Nanoparticles on the Dynamics of Gas Bubbles." Langmuir **26**(10): 6902-6907.

Vafaei, S. and D. Wen (2010). "The effect of gold nanoparticles on the spreading of triple line." Microfluidics and Nanofluidics **8**(6): 843-848.

Vajjha, R. S. and D. K. Das (2009). "Experimental determination of thermal conductivity of three nanofluids and development of new correlations." International Journal of Heat and Mass Transfer **52**(21-22): 4675-4682.

Wang, X.-Q. and A. S. Mujumdar (2007). "Heat transfer characteristics of nanofluids: a review." International Journal of Thermal Sciences **46**(1): 1-19.

Wang, X.-Q. and A. S. Mujumdar (2008). "A review on nanofluids - Part I: Theoretical and numerical investigations." Brazilian Journal of Chemical Engineering **25**(4): 613-630.

Wang, X.-Q. and A. S. Mujumdar (2008). "A review on nanofluids - Part II: Experiments and applications." Brazilian Journal of Chemical Engineering **25**(4): 631-648.

Wei, X., T. Kong, et al. (2010). "CuS/Cu2S nanofluids: Synthesis and thermal conductivity." International Journal of Heat and Mass Transfer **53**(9-10): 1841-1843.

Wen, D. (2008). "Mechanisms of thermal nanofluids on enhanced critical heat flux (CHF)." International Journal of Heat and Mass Transfer **51**(19-20): 4958-4965.

Wen, D. (2008). "On the role of structural disjoining pressure to boiling heat transfer of thermal nanofluids." Journal of Nanoparticle Research **10**(7): 1129-1140.

Wen, D. and Y. Ding (2005). "Experimental investigation into the pool boiling heat transfer of aqueous based gamma-alumina nanofluids." Journal of Nanoparticle Research **7**: 265-274.

Wen, D., G. Lin, et al. (2009). "Review of nanofluids for heat transfer applications." Particuology **7**(2): 141-150.

Wen, D., L. Zhang, et al. (2009). "Flow and migration of nanoparticle in a single channel." Heat and Mass Transfer **45**(8): 1061-1067.

Wong, K. V. and O. D. Leon (2010). "Applications of Nanofluids: Current and Future." Advances in Mechanical Engineering **2010**.

Xiao, B., G. Jiang, et al. (2010). "A fractal study for nucleate pool boiling heat transfer of nanofluids." SCIENCE CHINA Physics, Mechanics & Astronomy **53**(1): 30-37.

Xinyu, W. and et al. (2009). "Pressure drop and heat transfer of Al 2 O 3 -H 2 O nanofluids through silicon microchannels." Journal of Micromechanics and Microengineering **19**(10): 105020.

Xuan, Y. and Q. Li (2000). "Heat transfer enhancement of nanofluids." International Journal of Heat and Fluid Flow **21**(1): 58-64.

Xuan, Y. and W. Roetzel (2000). "Conceptions for heat transfer correlation of nanofluids." International Journal of Heat and Mass Transfer **43**(19): 3701-3707.

Xuan, Y. and Z. Yao (2005). "Lattice Boltzmann model for nanofluids." Heat and Mass Transfer **41**(3): 199-205.

Xue, H. S., J. R. Fan, et al. (2007). "Characteristic boiling curve of carbon nanotube nanofluid as determined by the transient calorimeter technique." Applied Physics Letters **90**(18): 184107-3.

Xue, L., P. Keblinski, et al. (2004). "Effect of liquid layering at the liquid-solid interface on thermal transport." International Journal of Heat and Mass Transfer **47**(19-20): 4277-4284.

You, S. M., J. H. Kim, et al. (2003). "Effect of nanoparticles on critical heat flux of water in pool boiling heat transfer." Applied Physics Letters **83**(16): 3374-3376.

Yu, W. and S. U. S. Choi (2003). "The Role of Interfacial Layers in the Enhanced Thermal Conductivity of Nanofluids: A Renovated Maxwell Model." Journal of Nanoparticle Research **5**(1): 167-171.

Zheng, X. and Z. Silber-Li (2009). "The influence of Saffman lift force on nanoparticle concentration distribution near a wall." Applied Physics Letters **95**(12).

Zhou, L., Y. Xuan, et al. (2010). "Multiscale simulation of flow and heat transfer of nanofluid with lattice Boltzmann method." International Journal of Multiphase Flow **36**(5): 364-374.