Journal Homepage: https://dergipark.org.tr/tr/pub/isibted

Journal of Thermal Science and Technology

Archive: https://tibtd.org.tr/dergi

Assessment of Multigrid Schemes with Fixed Patterns for a Two-Dimensional Heat Transfer Problem

Dr. Mustafa Serdar TEKÇE [1, *], Prof. Dr. Kürşad Melih GÜLEREN [2]

[1] *Turkish AeroSpace, Aircraft Vice Presidency, Air Vehicle Engineering, Aerodynamics – Jet Aircrafts, Kahramankazan, 06980, Ankara, Türkiye*
[2] *Eskişehir Osmangazi University, Faculty of Engineering and Architecture, Department of Aeronautical Engineering, Odunpazarı, 26040, Eskişehir, Türkiye*

A R T I C L E I N F O

2024, vol. 44, no.2, pp. 322-338

doi: 10.47480/isibted.1403905

Research Article
Received: 12 December 2023
Accepted: 31 July 2024

* *Corresponding Author*
e-mail:
mustafaserdar.tekce@tai.com.tr

Keywords:
Thermal Diffusion
Multigrid Scheme
Computational Cost

ORCID Numbers in author order:
0009-0002-2070-9036
0000-0003-3464-7956

A B S T R A C T

Heat transfer problems and their solutions are of critical importance in almost all areas of engineering and technology While many real-world problems are inherently three-dimensional, simplifying them to 2D models offer practical advantages with reasonable models. With being fundamental class of problem of heat transfer, 2D thermal diffusion problem was selected for the study. Multigrid methods were referred as standing out in terms of cost reduction while keeping solution accuracy. Effectivity of multigrid methods employing fixed pattern schemes were subject to investigation. In order to fairly setup a numerical experimentation, authentic code generation effort was given that implements basic finite volume method, intergrid operations and iterative solvers. A reference case with an analytical Laplace solution was selected and properly validated by results. Variety of multigrid schemes with fixed patterns were explored around parameters; iterations per sweep and maximum coarsening level. Results were compiled on the focal points of cost and performance. Comparison of the direct iterative methods with multigrid schemes proved the effectivity of multigrid schemes. Multigrid schemes generated convergent solutions with only 1.2% cost of direct iterative method for the reference case. With the assessment of the cost and performance outcomes of parameter explorations, it is concluded that any multigrid scheme should visit maximum coarsest level possible while keeping a minimum number of iterations on each grid resolution.

Sabit Şemalı Çoklu Ağ Döngülerinin İki Boyutlu Bir Isı Transfer Problemi Kapsamında İncelenmesi

M A K A L E B İ L G İ S İ

Anahtar Kelimeler:
Isıl Yayınım
Çoklu Ağ Şeması
Hesaplama Maliyeti

Ö Z E T

Isı transferi problemleri ve çözümleri, mühendislik ve teknolojinin neredeyse tüm alanlarında kritik öneme sahiptir. Birçok gerçek problem doğası gereği üç boyutludur, ancak bunları makul biçimlerde iki boyutlu modellere basitleştirmek, pratik avantajlar sunmaktadır. Dolayısıyla, iki boyutlu ısıl yayınım problemi, ısı transferinin temel bir problem sınıfından olması nedeniyle çalışma alanı olarak seçildi. Çoklu ağ yöntemleri, çözüm doğruluğunu koruduğu halde maliyeti ciddi oranda azaltmasıyla anılmaktaydı. Böylelikle, sabit şemaya sahip çoklu ağ yöntemlerinin etkinliği araştırma konusu olarak seçildi. Düzgün bir sayısal deney düzeneği kurulabilmesi için, temel seviyede sonlu hacimler yöntemi, çözünürlükler arası işlemler ve iteratif çözüm yöntemleri uygulayan özgün bir kod yazıldı. Analitik Laplace çözümü bulunan referans bir vaka seçildi ve sonuçlarla doğrulandı. Sabit şemalı çoklu ağ yöntemlerinin başarımı, bir çözünürlük seviyesindeki iterasyon sayısı ve azami seyreklik seviyesi parametreleri etrafında araştırıldı. Sonuçlar maliyet ve başarım odak noktalarına göre derlendi. Doğrudan iteratif yöntemlerin çoklu ağ şemaları ile karşılaştırılması, çoklu ağ şemalarının etkinliğini kanıtladı. Referans vaka için çoklu ağ şemaları, doğrudan iteratif yöntemin sadece %1.2 maliyeti ile yakınsayan çözümler üretti. Parametre keşiflerinden gelen maliyet ve başarım sonuçlarının değerlendirilmesiyle, herhangi bir çoklu ağ şemasının, her çözünürlükte asgari bir iterasyon yapması gerekirken mümkün olan en seyrek çözünürlük seviyesini ziyaret etmesi gerektiği sonucuna varıldı.

NOMENCLATURE

A	Coefficient Matrice of Systems of Equations
A	Denotation for Area
a	Discretized Coefficient
aij	An Element of Coefficient Matrice
b	Constants Vector of Systems of Equations
B	Denotation for Boundary
C, c	Generic Denotation for Cell
E, e	Denotation for East
F	Finalization
FVM	Finite Volume Method
h	Denotation for Grid Resolution Level
I	Initialization
i	Cell Indice in the first direction of domain
i	Denotation for Cell Number
ID	Identification number of the cell
j	Cell Indice in the second direction of domain
k	Heat Transfer Coefficient
L	Lower Part of Decomposed Coefficient Matrice
N, n	Denotation for North
P	Prolongation
P	Denotation for the Cell being processed
q, Q	Heat Flux
r	Residual Vector
R	Mean Residual
R	Restriction
RWU	Reference Work Unit
S	Source Term
S	Standard Deviation
S	Sweep of Iterations
S, s	Denotation for South
T	Temperature
U	Upper Part of Decomposed Coefficient Matrice
V	Volume
W, w	Denotation for West
x	Solution Vector
x	Location in the first direction of domain
y	Intermediate Solution Vector
y	Location in the second direction of domain
Γ	Coefficient of Diffusive Property
ζ	Spatial Location
φ	Physical Property

INTRODUCTION

Heat transfer, a fundamental aspect of engineering physics and numerous engineering problems, influencing the design and performance of a vast array of systems (Annaratone, 2010). The challenge lies not only in understanding the principles of heat transfer but also in devising effective methods to predict, control, and optimize thermal behaviour within complex structures. The diversity of engineering heat transfer problems spans from the microscopic scales of electronic devices, where efficient heat dissipation is paramount, to the macroscopic scales of power plants, where thermal efficiency and material durability are critical.

The complexities of heat transfer problems necessitate various methodologies for engineers to choose from. Analytical methods, rooted in mathematical elegance, provide solutions under simplified conditions and crude assumptions. However, applicability of analytical methods often hinders realizability of real-world complexities, such as irregular geometries or nonlinearities. Numerical methods, on the other hand, offer a versatile approach, capable of handling solution by discretizing the problem into manageable components (Axelsson, 1996). The choice between analytical elegance and numerical versatility emphasizes that the way towards solutions should balance accuracy, computational efficiency, and applicability to diverse engineering contexts. Cost efficiency is the key aspect towards establishing a high performing solution method.

Dimension of the heat transfer problem is another aspect to consider. While many real-world problems are inherently three-dimensional (Onur et al., 2011; Kürekçi and Özcan, 2012; Aydar and Ekmekçi, 2012; Uğurlubilek, 2012; Karaaslan et al., 2013; Sert et al., 2019; Yıldızeli and Çadırcı, 2023), simplified 2D models (Bali, 2006; Aykan and Dursunkaya, 2008; Mançuhan et al., 2011; Alpman, 2012; Doğan et al., 2012; Şimşek et al., 2020; Yetik and Mahir, 2020; Uzuner et al., 2023) can offer practical advantages without significant loss of accuracy, provided certain assumptions are reasonable. The transition from three-dimensional (3D) heat transfer problems to their two-dimensional (2D) counterparts is a common strategy in engineering and science (Annaratone, 2010).

There are several rationales beneath the assumptions for dimensional reduction in heat transfer problems. Systems/geometries/problems exhibiting some form of symmetry along an axis can be solved in two-dimensional domain with proper modelling. Likewise, there may be a dominant dimension with overwhelming difference in either size or gradients that can be solved via dimensional reduction. Many real-world applications, such as heat exchangers, electronic devices, and building materials, exhibit a primary heat transfer direction anyway. Modelling these systems in 2D often provides accurate results for the specific heat transfer aspects of interest. Simplified 2D model can also be used a stepping stone towards higher fidelity models. 2D models are often used as a benchmark for validating more complex 3D simulations. If the 2D model adequately represents the behaviour observed in experiments, it lends confidence to the accuracy of the simplification. Additionally, starting with 2D models helps researchers grasp fundamental heat transfer principles before tackling the increased complexity of 3D problems. Furthermore, solving 3D heat transfer problems computationally can be resource-intensive. By simplifying to 2D, simulations and analyses become more computationally efficient, enabling quicker and more practical solutions. However, it is imperative to acknowledge the limitations of 2D simplifications. The validity of assumptions must be carefully considered, and the impact of neglecting certain dimensions should be assessed to ensure the model's accuracy in a given context.

Various heat transfer problems can be modelled in two-dimensional domain. In electronic devices, heat generated by

components on a circuit board needs to be efficiently dissipated to prevent overheating. The thin layers of a circuit board often allow for the simplification of heat transfer analysis to a 2D plane, considering heat flow along the surface. Heat exchangers in various industries, such as automotive and HVAC systems, involve the transfer of thermal energy between fluids. Many heat exchanger configurations exhibit a primary flow direction, allowing for 2D modelling to capture the essential heat transfer mechanisms. Analysing the thermal behaviour of building materials, like walls or windows, is crucial for energy efficiency and climate control in structures. Heat transfer through building materials often occurs primarily in two dimensions, such as through the thickness of a wall. 2D models are suitable for these analyses. Heat transfer through the thickness of the plate can also be modelled in 2D, especially if the plate is thin compared to its length and width. The 2D model would involve analysing temperature gradients along the surface and through the thickness. These examples illustrate how 2D heat transfer models are applicable to analyse heat conduction in plates, where the heat transfer primarily occurs in two dimensions. Temperature variations across the plate's surface or through its thickness can be modelled and solved for practical engineering applications.

Numerical Methods

Numerical methods stand as a compromise between theoretical formulations and the practical demands of engineering. Finite Difference Methods, efficient in their simplicity, discretize the spatial domain into a grid and employ iterative schemes to approximate the temporal evolution of temperature. Finite Element Methods, renowned for their versatility, decompose structures into finite elements with distinct mathematical representations, accommodating complex geometries and material heterogeneities. Within various of numerical techniques, Finite Volume Methods (FVM) occupy a distinctive value (Versteeg and Malalasekera, 2007).

FVM approaches the 2D heat transfer problem by dividing the domain into finite control volumes, ensuring the conservation of energy within each volume. This conservation-centric philosophy makes FVM particularly well-suited for scenarios where the preservation of global quantities is crucial. By discretizing the problem in this manner, FVM facilitates a detailed understanding of spatial variations in temperature within the 2D plate. Its application extends beyond mere computation; it provides a comprehensive picture of how heat distributes across the plate, capturing thermal gradients and material properties.

In the context of the 2D heat transfer problem on a homogeneous plate, FVM offers distinct advantages. By dividing the plate into finite control volumes, FVM acknowledges the localized nature of thermal interactions. This granularity allows for the preservation of global quantities, ensuring that the simulation accurately reflects the overall energy balance within the system. The application of FVM to the 2D plate not only provides numerical results but also affords a deeper understanding of the thermal dynamics, shedding light on how temperature gradients evolve across the plate.

The computational framework of FVM involves discretizing the governing heat conduction equation over these control volumes and iteratively solving for temperature distribution. The method excels in scenarios where conservation laws play a pivotal role, making it particularly applicable to heat transfer problems characterized by intricate geometries and varying boundary conditions. FVM's ability to handle non-uniform material properties and adapt to irregular geometries positions it as a versatile and powerful tool in the numerical simulation of 2D heat transfer.

Main problem with any numerical solution endeavour is that available computational resource is almost always limited in any situation compared to desired amount, regardless of context and case. Thus, any effort is highly valuable that achieves solution at a desired accuracy while keeping resource (memory, processor, time, power) costs minimal (Versteeg and Malalasekera, 2007). Compromization between accuracy and cost is crucial for efficiency while using iterative methods to solve the given problem. Multigrid methods were found promising for cost efficiency towards accurate solutions.

Multigrid

Although there are several approaches to reduce computational cost, multigrid methods stand out in terms of cost reduction while keeping solution accuracy (Trottenberg et al., 2001). Multigrid methods and algorithms reduce computational cost dramatically (Brandt and Livne, 2011). Additionally, multigrid methods do not conflict with most of the other cost reduction approaches, making it a much more versatile and reliable tool while amplifying effectivity.

The multigrid approach arises from the recognition that not all features of a solution need the same level of detail. In the context of 2D heat transfer on a homogeneous plate, multigrid methods offer a hierarchical solution strategy. Coarser grids capture the overarching trends of temperature distribution, while finer grids address localized variations. This multiscale approach allows the solver to efficiently navigate the solution space, accelerating convergence and significantly reducing computational costs.

The multigrid approach complements the strengths of Finite Volume Methods, providing an additional layer of efficiency in solving 2D heat transfer problems. By operating on multiple resolutions, the multigrid method stands as a testament to the continual evolution of numerical techniques, offering a powerful solution for engineers and researchers grappling with the computational demands of intricate heat transfer simulations.

In a multigrid cycle, the grid is successively coarsened and refined in a consequential manner. This process involves moving between different levels of grid resolution. At each level, a relaxation method, often a simple iterative technique like Gauss-Seidel, is applied to smooth out errors in the solution. The process of moving between grid levels allows for the transfer of information, helping to correct errors more effectively. The W-cycle and F-cycle are variations of the V-cycle, introducing more coarsening and refinement steps in the multigrid scheme. These fixed-pattern multigrid schemes aim to address issues with convergence by taking

advantage of grid hierarchy and the interactions between different resolutions.

Multigrid schemes are a category of methods to accelerate convergence in iterative solvers. Fixed-pattern multigrid schemes, such as the V-cycle, W-cycle, and F-cycle, are widely used and have demonstrated their effectiveness in improving convergence speed (Trottenberg et al., 2001).

Multigrid cycles are procedural integration of various grid operations, solution iterations and decision checks composed in a specific algorithm. Various multigrid cycle definitions are present in literature and are used in software. Selecting, defining or setting an appropriate multigrid cycle depends on the case and requires exercise before execution. Multigrid cycles improve convergence rates of iterative solutions anyway, but reckless execution of multigrid cycles may hinder effectivity. Fixed-pattern multigrid schemes are robust and well-established, making them prior choice for many numerical simulations (Versteeg and Malalasekera, 2007).

Most of the multigrid cycle definitions have predetermined sequential order of operations and iterations. Figure 1 and Figure 2 illustrates V cycle and W cycle that are mentioned (without further elaboration) to be members of the μ cycles where μ parameter is 1 and 2 respectively (Wesseling, 1992). Figure 3 illustrates F cycle that is defined with a specified pattern that progressively increases the top level of fine grid resolutions (Briggs et al., 2000).

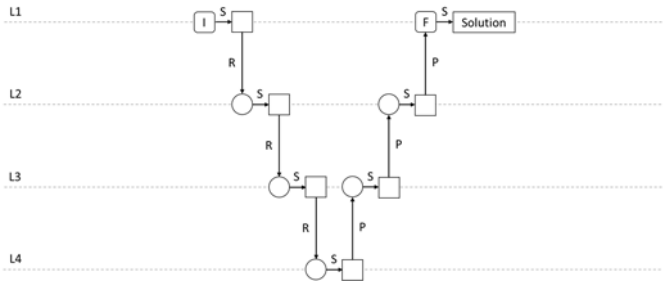

Figure 1. Pattern of a V multigrid cycle with four grid levels.

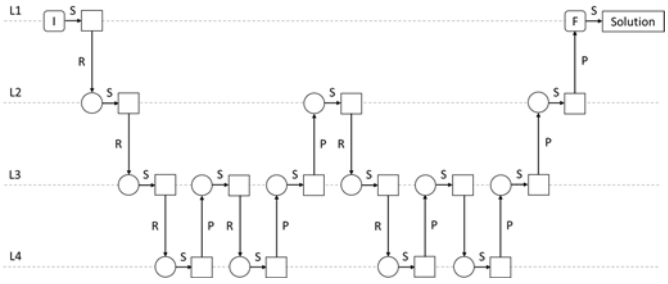

Figure 2. Pattern of a W multigrid cycle with four grid levels.

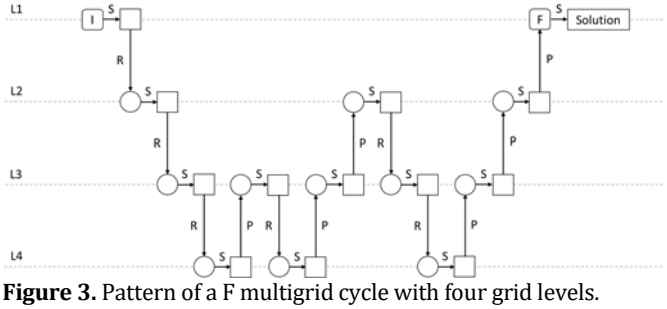

Figure 3. Pattern of a F multigrid cycle with four grid levels.

To transfer information between grids with different resolutions, multigrid employs restriction and prolongation operations (Briggs et al., 2000). Restriction operators aggregate data from finer grids to coarser grids, while prolongation operators interpolate data from coarser grids to finer grids. The entire process of transferring between grids, smoothing, and correction is repeated iteratively until a desired level of accuracy is achieved. Multigrid can significantly accelerate the convergence of iterative solvers, reducing the number of iterations required to reach a solution (Brandt and Livne, 2011).

The power of multigrid lies in its ability to address error components at different scales efficiently (Briggs et al., 2000). High-frequency error components are more effectively treated on finer grids, while low-frequency error components are addressed on coarser grids. This hierarchical approach can dramatically improve the convergence rate, making it a popular choice for solving complex problems with fine grids, where other solvers may struggle to converge in a reasonable amount of time (Brandt and Livne, 2011).

Multigrid on Heat Transfer

Multigrid methods have been used to solve heat transfer problems on numerous cases accompanied with varying numerical methods. Relevant studies were compiled below while emphasizing conclusions about multigrid methods.

In an early study, full-approximation-storage multigrid strategy had been applied to the heat conduction equation by using several iterative schemes to drive the multilevel process (Arnone and Sestini, 1991). Comparisons between single and multiple grid calculations for both explicit and implicit time-dependent schemes, in addition to stationary ones were given (Arnone and Sestini, 1991). The study concluded that multilevel technique had proven to be a very powerful and flexible strategy for speeding up convergence of both steady and unsteady formulations (Arnone and Sestini, 1991).

In another study, steady three-dimensional natural convection in a rectangular parallelepiped with a saturated porous medium was analysed numerically (Dawood and Burns, 1992). Convergence of the current three-dimensional model was enhanced using a multigrid method (Dawood and Burns, 1992). The multigrid method had been shown to work extremely well for problems of conduction and heating yielding speedup factors up to 22 (Dawood and Burns, 1992).

A multigrid algorithm was developed along with an implicit multiblock pressure-based solver for calculating flow and heat transfer problems on nonorthogonal grids (Przekwas and Lai, 1996). The full approximation scheme (FAS) was used due to nonlinearity of the Navier-Stokes equations (Przekwas and Lai, 1996). Optimum performance has been achieved for the multigrid multiblock calculations (Przekwas and Lai, 1996).

To assess the performance of a new full multigrid algorithm in which there is no restriction procedure for variables except for residuals; an algorithm was used in combination with the SIMPLE algorithm to solve fluid flows with heat transfer, using collocated grids and higher-order schemes for convective fluxes (Yan and Thiele, 1998). The modified full multigrid version of the SIMPLE algorithm using collocated grids was shown to be efficient for the calculation of fluid flow and heat transfer, with a speed-up factor of up to 25 (Yan and Thiele, 1998).

The performance of the symmetrically coupled Gauss-Seidel (SCGS) based multigrid method was investigated by applying it to three-dimensional conjugate heat transfer in two different configurations with discrete heat sources (Tang and Joshi, 1999). With multigrid SCGS (MG-SCGS), performance enhancement was even more impressive, and speed up factor can be as high as 19 (Tang and Joshi, 1999). SIMPLER failed to generate a converged solution for strong convection in a high aspect ratio channel while MG-SCGS were able to generate converged results in a timely fashion for all test cases (Tang and Joshi, 1999).

Steady-state two-dimensional solutions to the full compressible Navier–Stokes equations were computed for laminar convective motion of a gas in a square cavity with large horizontal temperature differences (Vierendeels et al., 2004). This line solver was used in a multistage stepping scheme and accelerated with the multigrid method (Vierendeels et al., 2004). The convergence of the solution method was stated as "very fast" and was shown to be independent of the Rayleigh number, the number of grid cells and the grid aspect ratio (Vierendeels et al., 2004).

A numerical tool, "Thermoflow" was developed for simulating the three-dimensional incompressible viscous flow and heat transfer on unstructured meshes (Wang and Joshi, 2006). Four iterative linear solvers have been implemented. The hybrid solver called AgMG/CG, employing agglomerated multigrid method and conjugate gradient, was found to be the best, for the symmetric equations arising from the pressure correction and heat conduction (Wang and Joshi, 2006).

A parallel multigrid-Schwarz method was used for the solution of hydrodynamics and heat transfer problems (Galante and Rizzi, 2007). The solution was obtained by a multigrid method parallelized by domain decomposition techniques, more specifically by the additive Schwarz method (Galante and Rizzi, 2007). A module called MGTool was implemented for the generation of the hierarchy of meshes and for the assemblage of the systems of equations (Galante and Rizzi, 2007). For the problem of heat transference, the methods using multigrid had converged, on average, 2 times faster (Galante and Rizzi, 2007). The study concluded that the combination of methods multigrid and domain decomposition was beneficial option in the solution of equation systems generated with partially differential equation (PDE) discretization (Galante and Rizzi, 2007).

In a fairly recent study, a parallel spatial/angular agglomeration multigrid scheme was developed to accelerate the finite-volume method (FVM) for the computation of radiative heat transfer in absorbing, emitting, and scattering gray media (Lygidakis and Nikolos, 2014). The multigrid scheme was based on the solution of the radiative transfer equation (RTE) with the full approximation scheme (FAS) on successively coarser spatial and angular resolutions (Lygidakis and Nikolos, 2014). The multigrid schemes were based on solution of the RTE on successively coarser spatial and angular resolutions, employing the FAS via a V-cycle strategy (Lygidakis and Nikolos, 2014). Despite the wide establishment of the multigrid technique in computational fluid dynamics (CFD), the angular and the combined spatial/angular methodology were stated as a new approach for the prediction of radiative heat transfer using FVM (Lygidakis and Nikolos, 2014). The acceleration obtained by the proposed multigrid scheme increased with the corresponding increase in grid size; the greater the number of DoFs, the greater the acceleration gained (Lygidakis and Nikolos, 2014).

Common key points were found in the studies including multigrid methods on heat transfer problems. First of all, even though the numerical methods and complexity of the cases were different, speed up factors were reported to be consistently around approximately 20 and 2 at worst. To put it reverse in terms of computational cost; multigrid methods were reported to decrease computational cost to 50% at worst and to 5% on average for the respective reference cases. Thus, it is safe to say that multigrid methods indeed substantially decrease computational cost. Secondly, and perhaps more importantly, multigrid methods were observed to increase robustness of the solution process towards a convergent solution, regardless of the numerical approach in all studies. Additionally, multigrid methods displayed flexibility on various cases with varying algebraic and spatial implementations, even with newly suggested ones.

Even though consistent inferences were found throughout the literature, most of the studies were conducted on specific and/or advanced cases. Cost and performance of the multigrid schemes were not particularly studied with basic numerical methods. Effectivity of multigrid methods employing fundamental schemes still needs to be investigated with a fair numerical experimentation.

METHOD

Sensible approach to explore fundamental multigrid methods was seen as the generation of a simple finite volume method solver and building multigrid algorithms over the definite infrastructure. Generation/development of a code/algorithm from scratch enabled full authority over the controlled experiments of parameters and clarity over validation. Infrastructural certainty improved the merits of the study.

Main concern of the study was to discover the fundamentals of multigrid cycles while establishing a viable and valid execution base. Thus, efforts were focused on generation/development of multigrid algorithms rather than generation/development of a solver.

Boundary condition problem was assumed as a two-dimensional diffusion. Thermal diffusion problem on a homogenous plate was selected for the reference case which may have constant temperature and constant heat flux at boundaries. Thermal coefficients were assumed to be constant within the homogenous plate.

Domain was assumed to be represented with a structured grid of uniform elements. Structured grid enabled to use geometric multigrid methods while easing the implementation of finite volume method operations.

Geometric multigrid methods were assumed in order to ease the implementation of restriction and prolongation and to focus on multigrid cycle algorithms.

Central differencing was assumed for discretization which is adequate for the reference case and is convenient to implement on structured grids and to work with geometric multigrid methods.

Gauss Seidel method without any modification was assumed for iterating the solution which is adequate for the reference case. Convergence of the solution was measured with residual which is a fundamental concept for iterative numerical methods. Residual was assumed as the absolute value of differences of a property for grid cells on successive iterations. Mean residual was defined and was calculated with Equation 2 using residual vector as Equation 1 (Versteeg and Malalasekera, 2007). Both elements of residual vector and consequently mean residual converges to zero towards exact solution. "r" denotes residual, "y" denotes intermediate solution of a property, "c" is the subscript for "cell", "i" is the subscript for "iteration", "n" denotes the number of cells within given domain.

$$r_c = |(y_c)_i - (y_c)_{i-1}| \quad (1)$$

$$\bar{r} = \frac{1}{n}\sum_{c=1}^{n}(r_c) \quad (2)$$

Study was built upon a code from scratch which is consisted of functions with scalable infrastructure. Functions were generated to work with varying inputs and to return output in a definitive format. Output of a function is input of another function and vice versa.

Environment

Selection of the medium for the tool generation is the initial step to consider. Environment should fit to the scope and purpose of the code. Capable of meeting the infrastructure requirements; an object-oriented, functional, fast, high-level, open-source language is considered for tool development. "The Julia programming language" has been preferred with the claim of being both high-level and fast. According to shared benchmark data, Julia has proven to be as fast as Fortran, even though it is as high-level as Python (Internet, 2023). Figure 4 summarizes the results of various languages for various benchmark types. The vertical axis of Figure 4 shows each benchmark time normalized against C implementation.

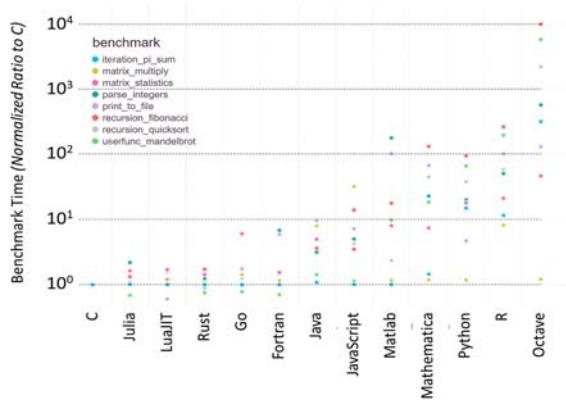

Figure 4. Benchmark results of various programming languages compared to C language (Internet, 2023).

Selection of the language led to specify which libraries to use while keeping tool development authentic as possible. Scope of the study focuses on the effectivity of multigrid methods and schemes. Thus, finite volume methods, iterative solvers and multigrid operations should be performed with authentic functions. Complimentary or basic infrastructure such as; mathematical functions, data base environment, graph plotting, file input output protocols, benchmark tools, are implemented by standard or imported libraries as seen in Table 1.

Table 1. Selected libraries for code infrastructure.

Library	***Origin***	***Explanation***
Dates	Standard	Provides functions that generates date and information in various/custom formats.
Printf	Standard	Enables macros that returns formatted output as string.
SparseArrays	Standard	Provides sparse vector/matrice infrastructure and functions.
Statistics	Standard	Offers basic and advanced statistical functions.
CSV	External	Input output interface/infrastructure for comma separated value files. Compatible with DataFrame library.
DataFrames	External	Offers database features with various filters and operations. Data can be processed in the desired format. Supports object-oriented data types.
Plots	External	Offers various plotting functions and features with various back-ends.
StatsPlots	External	Extension of Plots library. Contains many statistical recipes.

Finite Volume Method Functions

Basic operations regarding finite volume method were coded as functions. Definition of a boundary condition problem is sequential. Problem definition starts with domain description and grid generation. Once computational grid is present at desired resolution, a method for boundary condition inclusion follows. Discretization of the governing equations based on the grid and boundary conditions lead to coefficient matrices of systems of equations. Delivering the coefficient matrices of the problem completes finite volume method sequence of the solution procedure.

Generation of the domain is the very first step towards solution. Domain was assumed to be in rectangular shape on two dimensions. Size in each dimension may vary. Grid was assumed to be uniform in x and y directions. Number of divisions may vary in each direction. "initialize_domain" function works with two-dimensional size input and two number of division input. "initialize_domain" function returns the DataFrame object populated with all individual cell information regarding ID, i, j, x, y, T values. Figure 5 is an example output of the "initialize_domain" function.

6400×6 DataFrame

Row	ID	i	j	x	y	T
	Int64	Int64	Int64	Float64	Float64	Float64
1	1	1	1	0.01875	0.01875	0.0
2	2	2	1	0.05625	0.01875	0.0
3	3	3	1	0.09375	0.01875	0.0
4	4	4	1	0.13125	0.01875	0.0
5	5	5	1	0.16875	0.01875	0.0
⋮	⋮	⋮	⋮	⋮	⋮	⋮
6397	6397	77	80	2.86875	2.98125	0.0
6398	6398	78	80	2.90625	2.98125	0.0
6399	6399	79	80	2.94375	2.98125	0.0
6400	6400	80	80	2.98125	2.98125	0.0

6391 rows omitted

Figure 5. Output example of "initialize_domain" function.

Defining boundary conditions is essential for any boundary condition problem. Conventional denotations of boundaries are assumed as west, east, south and north (Versteeg and Malalasekera, 2007). Fix ("Dirichlet") and flux ("Neumann")

thermal boundary conditions are assumed. Temperature and heat flux values may vary at each boundary but does not vary within a boundary. Thus, "boundary_conditions" function works with four Dirichlet boundary condition input and four Neumann boundary condition input. Function then returns the DataFrame object contains boundary condition information to be later called as input to other functions. Figure 6 is an example output of the "boundary_conditions" function.

4×3 DataFrame

Row	boundary String	fix Float64	flux Float64
1	W	273.15	0.0
2	E	273.15	0.0
3	S	273.15	0.0
4	N	373.15	0.0

Figure 6. Output example of "boundary_conditions" function.

Validity of solutions based on finite volume method requires proper application of boundary conditions. Additionally, different grid resolutions of multigrid cycles require a compatible method for inclusion of boundary conditions. A method called link cutting using source terms of finite volume method (Versteeg and Malalasekera, 2007) while preserving discretization equations is generated in order to maintain a viable multigrid cycle execution. Boundary condition inputs are assumed as physically feasible. Since reference case was selected, constants and relations of thermal diffusion problem were assumed to be predetermined. "apply_boundary_conditions" function works with domain and boundary condition input which were assumed to be compatible.

Equation 3 and Equation 4 are used to calculate source coefficients (Versteeg and Malalasekera, 2007). Figure 7 illustrates the transformation of boundary conditions to source terms. Derivation of the source term contributions are given under the "discretised_form" function below. Source coefficient calculations assume cells to be equal in size at each direction.

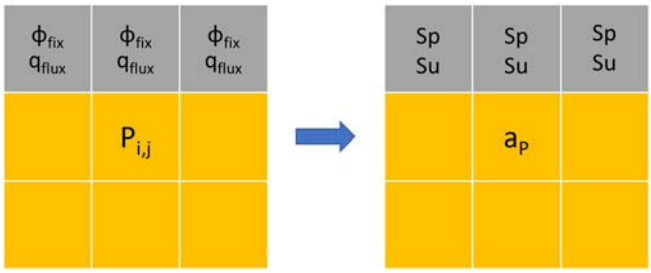

Figure 7. Transformation of boundary conditions to source coefficients.

$$S_p = -\frac{2k_B A_B}{\Delta\zeta} \quad (3)$$

$$S_u = q_B - S_p\,\phi_B \quad (4)$$

Figure 8 is an example output of the "apply_boundary_conditions" function.

6724×9 DataFrame

Row	ID Int64	i Int64	j Int64	x Float64	y Float64	T Float64	q Float64	Sp Float64	Su Float64
1	1	1	1	-0.01875	-0.01875	273.15	0.0	-20.0	5463.0
2	2	2	1	0.01875	-0.01875	273.15	0.0	-20.0	5463.0
3	3	3	1	0.05625	-0.01875	273.15	0.0	-20.0	5463.0
4	4	4	1	0.09375	-0.01875	273.15	0.0	-20.0	5463.0
5	5	5	1	0.13125	-0.01875	273.15	0.0	-20.0	5463.0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
6721	6721	79	82	2.90625	3.01875	373.15	0.0	-20.0	7463.0
6722	6722	80	82	2.94375	3.01875	373.15	0.0	-20.0	7463.0
6723	6723	81	82	2.98125	3.01875	373.15	0.0	-20.0	7463.0
6724	6724	82	82	3.01875	3.01875	373.15	0.0	-20.0	7463.0

6715 rows omitted

Figure 8. Output example of "apply_boundary_conditions" function.

Prerequisite to generate coefficient matrices of an iterative solution is to write down system of equations in a discretized form. Discretization method directly effects iterative solution progress alongside grid resolution. Gradients of physical properties are captured either inherently by grid resolution or numerically by discretization method.

Discretization method assumed as central differencing with linear interpolation. Figure 9 illustrates cell denotations used in discretization (Versteeg and Malalasekera, 2007) for a two-dimensional finite volume method. Equation 5 and Equation 6 are used for two-dimensional diffusion problem (Versteeg and Malalasekera, 2007). Equation 7 and Equation 8 are the gradually discretized form of the diffusion problem (Versteeg and Malalasekera, 2007). By distributing Equation 6 while using source terms given in Equation 9, discretized form is written as seen in Equation 10 for two-dimensional diffusion problem (Versteeg and Malalasekera, 2007). Table 2 summarizes the expressions of coefficients of discretized form.

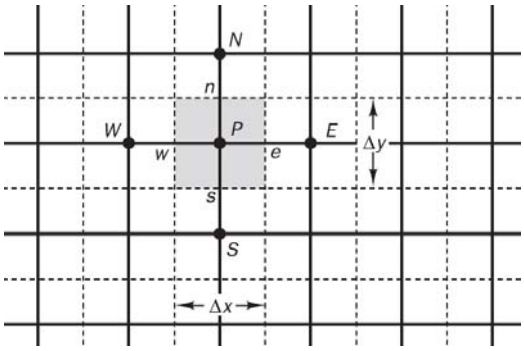

Figure 9. Denotations for discretization function (Versteeg and Malalasekera, 2007).

$$\frac{\partial}{\partial x}\left(\Gamma\frac{\partial\phi}{\partial x}\right) + \frac{\partial}{\partial y}\left(\Gamma\frac{\partial\phi}{\partial y}\right) + S_\phi = 0 \quad (5)$$

$$\int_{\Delta V}\frac{\partial}{\partial x}\left(\Gamma\frac{\partial\phi}{\partial x}\right)dx\,dy + \int_{\Delta V}\frac{\partial}{\partial y}\left(\Gamma\frac{\partial\phi}{\partial y}\right)dx\,dy + \int_{\Delta V} S_\phi\,dV = 0 \quad (6)$$

$$\left[\Gamma_e A_e\left(\frac{\partial\phi}{\partial x}\right)_e - \Gamma_w A_w\left(\frac{\partial\phi}{\partial x}\right)_w\right] + \left[\Gamma_s A_s\left(\frac{\partial\phi}{\partial x}\right)_s - \Gamma_n A_n\left(\frac{\partial\phi}{\partial x}\right)_n\right] + \bar{S}\Delta V = 0 \quad (7)$$

$$\Gamma_e A_e\frac{(\phi_E-\phi_P)}{\delta x_{PE}} - \Gamma_w A_w\frac{(\phi_P-\phi_W)}{\delta x_{WP}} + \Gamma_n A_n\frac{(\phi_N-\phi_P)}{\delta x_{PN}} - \Gamma_s A_s\frac{(\phi_P-\phi_S)}{\delta x_{SP}} + \bar{S}\Delta V = 0 \quad (8)$$

$$\bar{S}\Delta V = S_u + S_p\phi_P \quad (9)$$

Table 2. Coefficients of discretized form.

a_W	a_E	a_S	a_N	a_P
$\frac{\Gamma_w A_w}{\delta x_{WP}}$	$\frac{\Gamma_e A_e}{\delta x_{PE}}$	$\frac{\Gamma_s A_s}{\delta y_{SP}}$	$\frac{\Gamma_n A_n}{\delta y_{PN}}$	$a_W + a_E + a_S + a_N - S_p$

$$a_P\phi_P = a_W\phi_W + a_E\phi_E + a_S\phi_S + a_N\phi_N + S_u \quad (10)$$

"discretised_form" function works with DataFrame input which contains domain and boundary condition information. Equation 11 is used to calculate Sp values, Equation 12 is used to calculate aP values and Equation 13 is used to calculate Su values. Eventually, a DataFrame object is returned as output containing grid information and respective coefficients of discretization. Figure 10 is an example output of the "discretised_form" function.

$$S_{p_P} = S_{p_W} + S_{p_E} + S_{p_S} + S_{p_N} \quad (11)$$

$$a_P = a_W + a_E + a_S + a_N - S_p \quad (12)$$

$$S_{u_P} = S_{u_W} + S_{u_E} + S_{u_S} + S_{u_N} \quad (13)$$

6400×12 DataFrame

Row	ID	i	j	x	y	aW	aE	aS	aN	Sp	aP	Su
	Int64	Int64	Int64	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Float64
1	1	1	1	0.01875	0.01875	0.0	10.0	0.0	10.0	-40.0	60.0	10926.0
2	2	2	1	0.05625	0.01875	10.0	10.0	0.0	10.0	-20.0	50.0	5463.0
3	3	3	1	0.09375	0.01875	10.0	10.0	0.0	10.0	-20.0	50.0	5463.0
4	4	4	1	0.13125	0.01875	10.0	10.0	0.0	10.0	-20.0	50.0	5463.0
5	5	5	1	0.16875	0.01875	10.0	10.0	0.0	10.0	-20.0	50.0	5463.0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
6397	6397	77	80	2.86875	2.98125	10.0	10.0	10.0	0.0	-20.0	50.0	7463.0
6398	6398	78	80	2.90625	2.98125	10.0	10.0	10.0	0.0	-20.0	50.0	7463.0
6399	6399	79	80	2.94375	2.98125	10.0	10.0	10.0	0.0	-20.0	50.0	7463.0
6400	6400	80	80	2.98125	2.98125	10.0	0.0	10.0	0.0	-40.0	60.0	12926.0

6391 rows omitted

Figure 10. Output example of "discretised_form" function.

Last step before venturing towards iterative solution is to generate coefficients of matrices using discretised form of domain. Generating coefficients of matrices is basically writing down system of equations by cell number instead of neighbouring denotations. "coefficient_matrices" function merely transforms/rearranges discretised form of the domain.

"coefficient_matrices" function works with DataFrame input which contains grid information and respective coefficients of discretization. Equation 14 and Equation 15 show the output format of the matrices. Figure 11 is an example output of the "coefficient_matrices" function.

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \quad (14)$$

$$A\,x = b \quad (15)$$

A	b
6400×6400 Matrix{Float64}:	6400-element Vector{Float64}:
60.0 -10.0 0.0 0.0 0.0 0.0 … 0.0 0.0 0.0 0.0 0.0 0.0	10926.0
-10.0 50.0 -10.0 0.0 0.0 0.0 … 0.0 0.0 0.0 0.0 0.0 0.0	5463.0
0.0 -10.0 50.0 -10.0 0.0 0.0 … 0.0 0.0 0.0 0.0 0.0 0.0	5463.0
0.0 0.0 -10.0 50.0 -10.0 0.0 … 0.0 0.0 0.0 0.0 0.0 0.0	5463.0
0.0 0.0 0.0 -10.0 50.0 -10.0 … 0.0 0.0 0.0 0.0 0.0 0.0	5463.0
0.0 0.0 0.0 0.0 -10.0 50.0 … 0.0 0.0 0.0 0.0 0.0 0.0	5463.0
0.0 0.0 0.0 0.0 0.0 -10.0 … 0.0 0.0 0.0 0.0 0.0 0.0	5463.0
⋮	⋮
0.0 0.0 0.0 0.0 0.0 0.0 … -10.0 0.0 0.0 0.0 0.0 0.0	7463.0
0.0 0.0 0.0 0.0 0.0 0.0 … 50.0 -10.0 0.0 0.0 0.0 0.0	7463.0
0.0 0.0 0.0 0.0 0.0 0.0 … -10.0 50.0 -10.0 0.0 0.0 0.0	7463.0
0.0 0.0 0.0 0.0 0.0 0.0 … 0.0 -10.0 50.0 -10.0 0.0 0.0	7463.0
0.0 0.0 0.0 0.0 0.0 0.0 … 0.0 0.0 -10.0 50.0 -10.0 0.0	7463.0
0.0 0.0 0.0 0.0 0.0 0.0 … 0.0 0.0 0.0 -10.0 60.0	12926.0

Figure 11. Output example of "coefficient_matrices" function.

Iterative Solution Functions

Iterative solution functions based on iteration per sweep count were generated in order to properly engage in multigrid scheme exploration. Iterator function itself was assumed as Gauss-Seidel iterative method without modifications or relaxations. Equation 16 to Equation 20 summarizes Gauss-Seidel method used in iterative solutions (Axelsson, 1996).

$$A = L_* + U \quad (16)$$

$$L_* = \begin{bmatrix} a_{11} & 0 & \dots & 0 \\ a_{21} & a_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \quad (17a)$$

$$U = \begin{bmatrix} 0 & a_{12} & \dots & a_{1n} \\ 0 & 0 & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{bmatrix} \quad (17b)$$

$$L_* x = b - Ux \quad (18)$$

$$x^{(k+1)} = L_*^{-1}(b - Ux^{(k)}) \quad (19)$$

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)}\right)$$

$$i = 1,2,\dots,n \quad (20)$$

Iterator function works with matrices (A) and vectors (b, x) input and performs one iteration with Gauss-Seidel method. Intermediate solution vector (x) is returned as output of "gauss_seidel_iterate" function.

Iterative solution function for cycles works with matrices (A), vectors (b, x, r) and iteration count (ic) input and performs consecutive iterations. Example output of "iterate_solution_count_for_cycles" are given in Figure 12, Figure 13 and Figure 14. DataFrame object as seen in Figure 12 contains mean residual (R), standard deviation of residual (S) and time stamp (t) values of each iteration as columns. DataFrame object as seen in Figure 13 contains solution vectors of each iteration as columns. DataFrame object as seen in Figure 14 contains residual vectors of each iteration as columns.

8×3 DataFrame

Row	R	S	t
	Float64	Float64	Int64
1	274.729	12.0958	79943600
2	0.154425	0.563102	156054400
3	0.146266	0.517291	232470400
4	0.139211	0.478872	308532200
5	0.133032	0.446151	384623900
6	0.127562	0.41792	460654500
7	0.122675	0.393291	542912800
8	0.118274	0.371599	620959000

Figure 12. "iteration_history" output example of "iterate_solution_count_for_cycles" function.

6400×8 DataFrame

Row	x1	x2	x3	x4	x5	x6	x7	x8
	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Float64
1	273.15	273.15	273.15	273.15	273.15	273.15	273.15	273.15
2	273.15	273.15	273.15	273.15	273.15	273.15	273.15	273.15
3	273.15	273.15	273.15	273.15	273.15	273.15	273.15	273.15
4	273.15	273.15	273.15	273.15	273.15	273.15	273.15	273.15
5	273.15	273.15	273.15	273.15	273.15	273.15	273.15	273.15
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
6397	358.2	358.755	359.208	359.586	359.906	360.18	360.418	360.626
6398	355.701	356.112	356.446	356.723	356.957	357.157	357.33	357.482
6399	348.046	348.291	348.49	348.655	348.794	348.913	349.017	349.107
6400	322.352	322.431	322.496	322.55	322.596	322.635	322.668	322.698

6391 rows omitted

Figure 13. "field_history" output example of "iterate_solution_count_for_cycles" function.

6400×8 DataFrame

Row	x1	x2	x3	x4	x5	x6	x7	x8
	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Float64
1	272.15	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	272.15	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	272.15	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	272.15	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	272.15	0.0	0.0	0.0	0.0	0.0	0.0	0.0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
6397	357.2	0.554634	0.453622	0.37796	0.319825	0.27419	0.237708	0.208083
6398	354.701	0.410606	0.334163	0.277349	0.233961	0.200071	0.173088	0.151249
6399	347.046	0.244645	0.199065	0.165214	0.139376	0.119198	0.103136	0.0901361
6400	321.352	0.0794711	0.0648093	0.0538891	0.0455332	0.0389943	0.0337793	0.0295523

6391 rows omitted

Figure 14. "residual_history" output example of "iterate_solution_count_for_cycles" function.

Intergrid Operations Functions

Initialization, restriction, prolongation and finalization are basic operations for any multigrid cycle. Initialization and finalization operations are covered with iterative solution functions. Restriction and prolongation operations require individual and complimentary functions. Thus, functions that changes the resolution of the solution were generated for multigrid cycles to use.

Restriction is basically downgrading the resolution. Geometric multigrid operations were assumed. Domain was assumed as rectangular in two dimensions. Grid was assumed cell centred, uniform and structured. Weighting method with linear interpolation was assumed. Equation 21 was used to calculate property values of cells at downgraded resolution. Figure 15 illustrates restriction operation in two dimensions.

$$C_{i,j}^{h+1} = \frac{1}{4}\left(C_{2i-1,2j-1}^{h} + C_{2i,2j-1}^{h} + C_{2i-1,2j}^{h} + C_{2i,2j}^{h}\right) \quad (21)$$

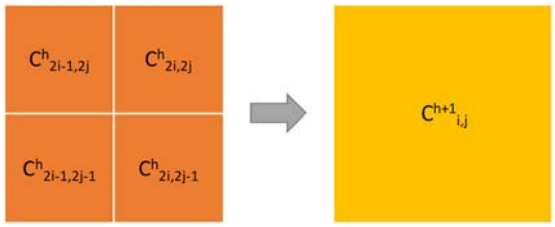

Figure 15. Restriction operation on two dimensions.

Coarsening function works with DataFrame object input that contains grid information (ID, i, j, x, y) and cell properties (T, r). Input is assumed to be compatible for a feasible restriction operation. When all cells are processed, function returns the DataFrame object populated with all individual cell information of the coarse grid. Figure 16 is an example output of the "restrict" function.

1600×7 DataFrame

Row	ID	i	j	x	y	T	r
	Int64	Int64	Int64	Float64	Float64	Float64	Float64
1	1	1	1	0.0375	0.0375	273.15	0.0
2	2	2	1	0.1125	0.0375	273.15	0.0
3	3	3	1	0.1875	0.0375	273.15	0.0
4	4	4	1	0.2625	0.0375	273.15	0.0
5	5	5	1	0.3375	0.0375	273.15	0.0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
1597	1597	37	40	2.7375	2.9625	331.586	0.0208765
1598	1598	38	40	2.8125	2.9625	331.586	0.0208765
1599	1599	39	40	2.8875	2.9625	331.191	0.0199269
1600	1600	40	40	2.9625	2.9625	315.864	0.00838497

1591 rows omitted

Figure 16. Output example of "restriction" function.

Prolongation is basically upgrading the resolution. Geometric multigrid operations were assumed. Domain was assumed as rectangular in two dimensions. Grid was assumed cell centred, uniform and structured. Linear interpolation was assumed. Equation 23 was derived by distributing Equation 22 and by arranging the expression. Equations 23 to 26 are used to calculate property values of cells at upgraded resolution. Figure 17 illustrates prolongation operation in two dimensions. Uppercase letters denote coarse grid cells while lowercase letters denote fine grid cells.

$$a = \frac{1}{4}\left(A + \frac{A+B}{2} + \frac{A+C}{2} + \frac{A+B+C+D}{4}\right) \quad (22)$$

$$a = \frac{1}{16}(9A + 3B + 3C + D) \quad (23)$$

$$b = \frac{1}{16}(3A + 9B + C + 3D) \quad (24)$$

$$c = \frac{1}{16}(3A + B + 9C + 3D) \quad (25)$$

$$d = \frac{1}{16}(A + 3B + 3C + 9D) \quad (26)$$

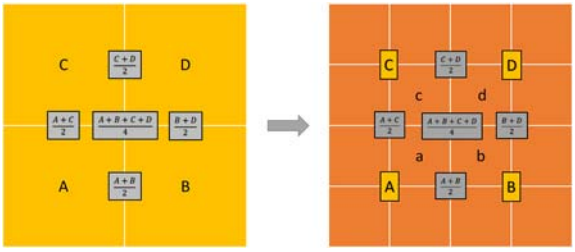

Figure 17. Prolongation operation on two dimensions.

Prolongation function works with boundary condition input in addition to DataFrame object input that contains grid information (ID, i, j, x, y) and cell properties (T, r) while calling "face_and_node_values" function to generate necessary additional information. "face_and_node_values" function calculates neighbouring cell, face and node information while respecting boundary conditions. Input is assumed to be compatible for a feasible prolongation operation. When all cells are processed, function returns the DataFrame object populated with all individual cell information of the fine grid. Figure 18 is an example output of the "prolong" function.

25600×7 DataFrame

Row	ID	i	j	x	y	T	r
	Int64	Int64	Int64	Float64	Float64	Float64	Float64
1	1	1	1	0.009375	0.009375	273.15	0.0
2	2	2	1	0.028125	0.009375	273.15	0.0
3	3	3	1	0.046875	0.009375	273.15	0.0
4	4	4	1	0.065625	0.009375	273.15	0.0
5	5	5	1	0.084375	0.009375	273.15	0.0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
25597	25597	157	160	2.93437	2.99062	360.068	0.00352723
25598	25598	158	160	2.95312	2.99062	357.477	0.00257343
25599	25599	159	160	2.97187	2.99062	351.602	0.00155082
25600	25600	160	160	2.99062	2.99062	322.693	0.000514068

25591 rows omitted

Figure 18. Output example of "prolongation" function.

Multigrid Scheme Functions

Running an iterative solution with any multigrid cycle requires working algorithms on convenient data structure. Although algorithms may vary, consistent data structure enables fair comparison between cycles/algorithms. Accessibility to recordings of solution progress is another rationale to have a practical data structure. Table 3 summarizes the columns of the DataFrame object that records run of a multigrid cycle/algorithm with brief explanations. Some of the columns may be seen as excessive in terms of computational cost, however accessibility and practicality of available information reduced the efforts towards tool development and result generation.

Excluding columns of DataFrame objects containing field and history information enables summarizing run history in a compact fashion. Figure 19 is an example to a run history that summarizes what transpires during a multigrid cycle.

Execution of any multigrid cycle is consisted of calling finite volume method and iterative solution functions while steering through resolution levels. Any multigrid cycle goes through initialization, resolution change (restriction and prolongation), iterative solution sweeps at multigrid levels and finalization activities. Figure 20 illustrates conceptual flowchart of any multigrid cycle run. Initialization and finalization activities are at initial resolution level and does not require additional functions rather than iterative solution functions. Pre-process and initial generation of data structure activities were merged under a function. Resolution changes, iteration sweeps, monitoring, conditioning, steering and other complimentary activities of multigrid cycles were merged including initialization and finalization under a function. Post-process of the run was not included within the multigrid algorithms.

10×11 DataFrame

Row	step	level	n_cell	nx	ny	initial_residual	final_residual	iteration_count	iteration_duration	operation_duration	FVM_duration
	Int64	Int64	Int64	Int64	Int64	Float64	Float64	Int64	Int64	Int64	Int64
1	1	0	6400	80	80	1.0	1.0	1	0	116400	23529600
2	2	0	6400	80	80	1.59544	0.316303	3	257544000	4800	34509400
3	3	1	1600	40	40	0.464135	0.503073	3	17744200	128600	3211300
4	4	2	400	20	20	0.845009	0.88932	3	1526200	90100	867800
5	5	3	100	10	10	1.48021	1.48271	3	269500	36100	369900
6	6	4	25	5	5	2.21309	1.88699	3	103200	37300	184100
7	7	3	100	10	10	1.39117	0.312344	3	328200	419400	281400
8	8	2	400	20	20	0.272293	0.0745363	3	1946500	771700	712800
9	9	1	1600	40	40	0.0675594	0.0198555	3	19569800	2443200	3514000
10	10	0	6400	80	80	0.0182567	0.0054156	3	258939500	11056700	32870300

Figure 19. Example to a run history of a multigrid cycle.

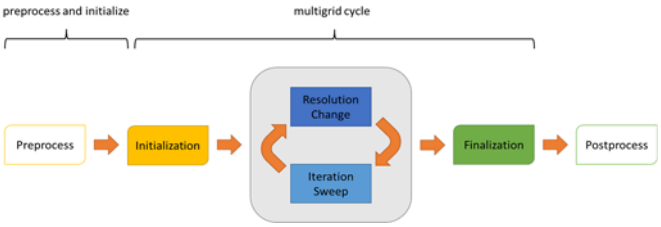

Figure 20. Conceptual flowchart of a multigrid run.

Table 3. Columns of DataFrame object that records the run.

Column	Data Type	Explanation
step	Integer	Steps of the run.
level	Integer	Resolution level of each step.
n_cell	Integer	Number of cells for the resolution on each step.
nx	Integer	Number of divisions in first direction for each step.
ny	Integer	Number of divisions in second direction for each step.
initial_residual	Float	Initial mean residual value at the beginning of each step.
final_residual	Float	Final mean residual value at the end of each step.
iteration_count	Integer	Iteration count for each step.
iteration_duration	Integer	Elapsed time while iterating the solution for each step.
operation_duration	Integer	Elapsed time while executing intergrid (restriction or prolongation) operations preceding finite volume method functions for each step.
FVM_duration	Integer	Elapsed time while executing finite volume method functions preceding iterations for each step.
initial_field	DataFrame	Initial field information (ID, i, j, x, y, T, r) at the beginning of each step.
final_field	DataFrame	Final field information (ID, i, j, x, y, T, r) at the end of each step.
iteration_history	DataFrame	Iterative solution history of mean residual (R), standard deviation of residual (S) and time stamps of each step.
field_history	DataFrame	Solution vector history of iterative solution of each step.
residual_history	DataFrame	Residual vector history of iterative solution of each step.

Every run starts with pre-process activities that generates grid upon domain input while specifying boundary conditions and solver model. Since solver model is predefined with study scope, generating grid and executing initial finite volume method operations are assumed to be the pre-process of multigrid cycles.

Creating container of run data as described in Table 3 and storing initialization phase completes preparation towards execution of multigrid cycles.

Creating algorithms to execute multigrid cycles with predefined patterns are straightforward in terms of complexity. Multigrid cycles with fixed conceptual patterns vary with the iteration count of sweeps on resolution levels in addition to desired or maximum coarsening level.

Multigrid cycle that follows fixed V pattern as illustrated in Figure 1 belongs to the μ cycle family (Briggs et al., 2000). Constant number of iterations per sweep was assumed for resolution levels. Maximum coarsening level was assumed as the maximum feasible level for multigrid methods works. Initialization phase and finalization phase was assumed to be included in the iterations on initial resolution level.

"run_fixed_V_cycle" function works with initialized data structure, boundary conditions, desired coarsening level and number of iteration per sweep input. Initialized data is copied to be later populated with run data. Maximum coarsening level is determined using a complimentary function that checks if desired coarsening level is feasible and assigns maximum possible level instead if it is not. Once maximum coarsening level is set, an array is generated that holds the sequential resolution levels of the cycle. For loop is executed for the level values in the array. Within the for loop; resolution change operations are carried out according to level direction; values of step, number of cells, number of divisions are assigned; finite volume method functions are called for the field on process; iterative solution function for cycles is called according to input; data structure is populated with the information of each step. Finalization phase of multigrid cycle inherently included with the iterative solution of the field on initial resolution at the end of the cycle. Eventually, "run_fixed_V_cycle" function returns DataFrame that contains run data as output. Figure 21 is an example output of the "run_fixed_V_cycle" function.

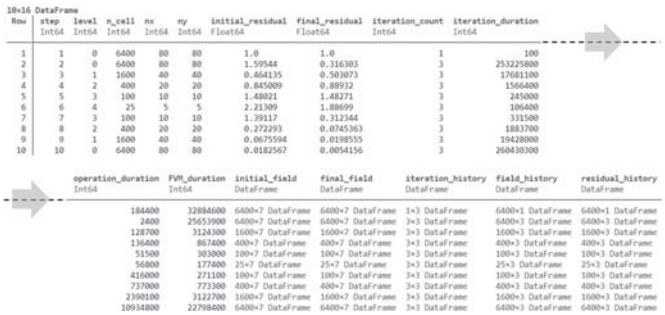

Figure 21. Output example of "run_fixed_V_cycle" function.

"run_fixed_W_cycle" and "run_fixed_F_cycle" functions are exactly same as "run_fixed_V_cycle" besides the sequence of grid resolutions which is also called pattern or scheme. They work with same type of inputs while generating same type of output.

RESULTS

Results were presented with increasing complexity starting with validation of the tools (procedures) at disposal. Validation pack serves the answer to the question "do procedures get to correct solution?". Parameter explorations were conducted to answer to questions "which parameters effect the way and/or the cost to solution?".

In order to fairly compare performances of different multigrid cycles as well as same one with differing input parameters an objective indicator should be defined. Even though there was a cost definition (Briggs et al., 2000) called "work unit" based on memory allocation for iterative methods applicable to multigrid cycles, a new and more comprehensive concept was suggested. This new concept, was also inspired by "work unit" definition was called "reference work unit" which is defined as the elapsed time of performing one iteration on the finest grid. Time cost inherently includes any varying parameter of

the environment, algorithm and system that multigrid cycle works on. Time cost also includes the energy cost to power the computer system during the process. Thus, measuring computational cost in terms of elapsed time was seen as much more convenient and comprehensive for comparing performance of multigrid cycles.

Validation

Two-dimensional thermal diffusion problem on a homogenous plate was selected for the reference case which may have constant temperature and constant heat flux at boundaries. Selecting thermal diffusion case which has analytical Laplace solution as reference inherently enabled proper validation of methods.

Thermal diffusion case has constant temperature and constant heat flux at boundaries (Patil and Prasad, 2014). Thermal coefficients and thickness are constant within the homogenous plate (Patil and Prasad, 2014). Reference case has certain dimensions on both directions (Patil and Prasad, 2014). Number of divisions of the domain were specifically selected to clearly present results before exploring higher grid resolutions.

Figure 22 was given to illustrate the reference case. Thermal conductivity was assumed constant as 1000 W/mK within the homogenous plate (Patil and Prasad, 2014). Thickness of the plate was assumed constant as 1cm (Patil and Prasad, 2014). Thermal diffusion case has constant temperature values at boundaries West, East, South and North as 0 °C, 0 °C, 0 °C and 100 °C respectively. Reference case has dimensions of 3m in both directions. Reference domain was generated with 80 division of cells in both directions.

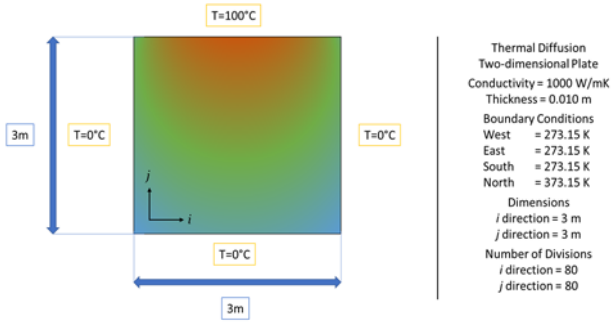

Figure 22. Illustration of the reference case definition.

Thermal diffusion case can be solved by Laplace transform using boundary conditions (Patil and Prasad, 2014). Equation 29 was used to generate temperature solutions within the domain at any location (Patil and Prasad, 2014).

$$\nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \quad x = [0, a]\ ;\ y = [0, b] \tag{27}$$

$$u(0, y) = 0°C\ ; u(a, y) = 0°C \tag{28a}$$

$$u(x, 0) = 0°C\ ; u(x, b) = 100°C \tag{28b}$$

$$u(x, y) = \frac{400\ °C}{\pi} \sum_{k=1}^{\infty} \frac{sin\left(\frac{n\pi x}{a}\right) sinh\left(\frac{n\pi y}{b}\right)}{n sinh(n\pi)} \qquad n = 2k - 1 \tag{29}$$

Field contour plot was given in Figure 23 to illustrate exact solution of the case. Since solution is exact and residual is practically zero; analytical solution of the case can be used to measure any error within the domain which are generated by other numerical methods.

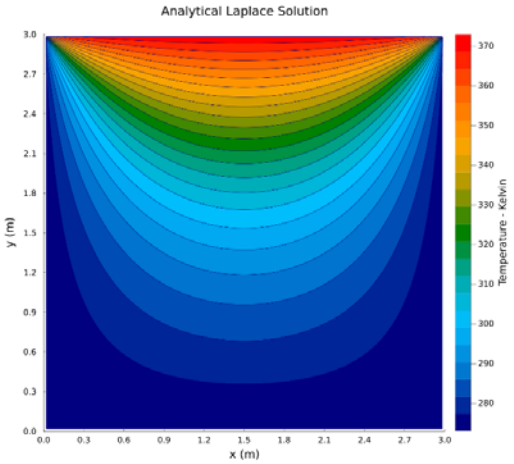

Figure 23. Analytical Laplace solution field contour plot of the reference case.

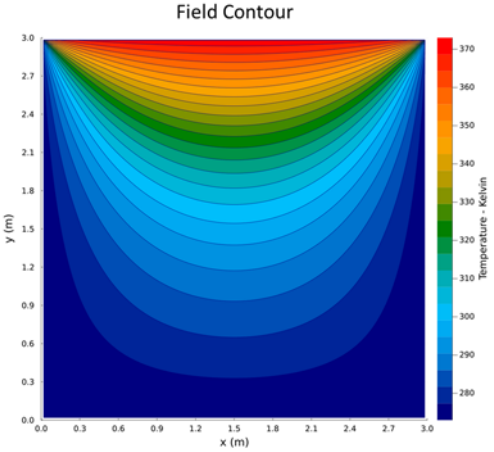

Figure 24. Direct iterative solution field contour plot (left) and residual scatter (right) plot of the reference case.

Reference case was solved via Gauss-Seidel iterative method as described after preprocessing the input by finite volume method functions. Constant initial guess value was given to the domain. Constant initial guess value was calculated as 298.15 °K by averaging temperature values of boundary conditions. Acceptable results were obtained when mean residual value was at 0.001 (°K) mean residual and was reached after 1028 iterations.

Field contour plot and residual scatter plot were given in Figure 24 to illustrate direct iterative solution and residual distribution of the reference case for a qualitative validation. Even though convergence criterion was fairly precise, uneven distribution of residual values was apparent as seen in Figure 24 for direct iterative solution. Figure 26 was given to illustrate temperature and error values over a diagonal line, a horizontal line and a vertical line as described in Figure 25 for quantitative validation of solution.

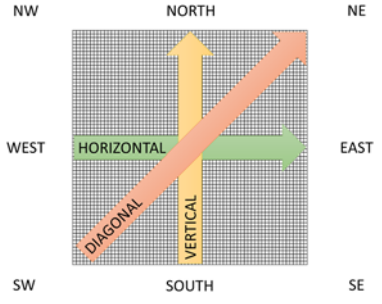

Figure 25. Data extraction lines for validation figures.

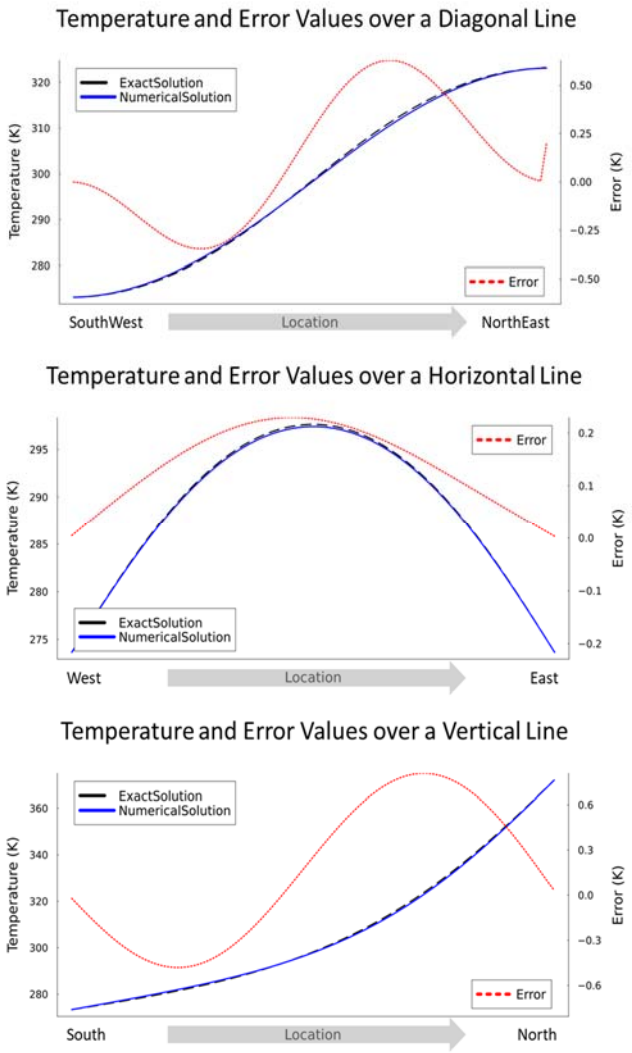

Figure 26. Validation figures of solutions regarding temperature and error values.

Verification

It is imperative to show that methods and procedures generate output changing with varying input. In order to verify that methods and code works with varying input, different sets of boundary conditions were applied to reference case.

Code accepts Dirichlet and Neumann boundary conditions for the thermal diffusion case. Dirichlet boundary conditions were given as temperature values in the unit of Kelvin. Neumann boundary conditions were given as heat flux values in the unit of Watts.

Randomized boundary condition values were selected from arrays of; 200, 300, 400, 500 °K for Dirichlet and -2000, -1500, -1000, 500, 0, 500, 1000, 1500, 2000 W for Neumann; to each West, East, South and North boundaries. Numerous solutions were generated with the runs using randomized boundary conditions in order to check if methods and/or procedures fail at some. No failures were returned for over hundred cases which verified that methods and procedures work with various boundary conditions. Examples of field contours to such solutions were given in Figure 27 and Figure 28 to emphasize that methods and procedures were capable of generating varying solutions with varying input.

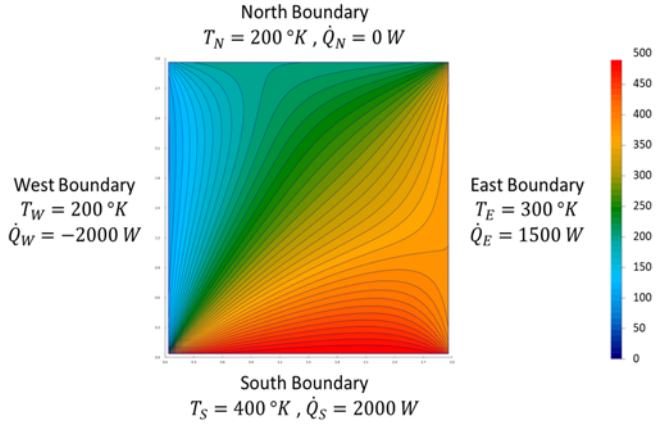

Figure 27. Example #1 of field contour plot to randomized boundary condition cases.

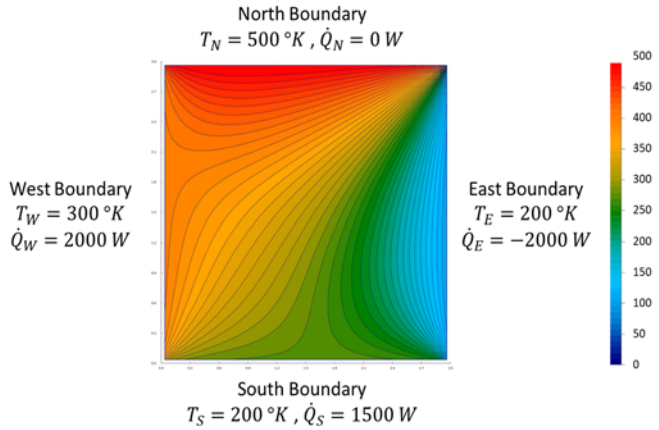

Figure 28. Example #2 of field contour plot to randomized boundary condition cases.

Parameters of Cycles

Verification of finite volume methods and iterative solution proceeded with multigrid cycle runs for varying input parameters. Multigrid cycles with fixed schemes (V, W, F) were explored around maximum coarse grid level and iterations per sweep. Restriction beyond maximum coarse grid level was infeasible and/or was limited. Iterations per sweep value was the iteration count of solution on each step of multigrid cycle. Exploration was designed to generate cost and convergence values with varying maximum coarse grid level and iteration per sweep. Each multigrid cycle was explored separately.

Maximum coarse grid levels were selected as 2, 3, 4, 5 to capture the effect on cost and convergence. Initial grid resolution was selected as 160x160, so minimum grid resolutions are 40x40, 20x20, 10x10, 5x5 respectively. Iterations per sweep values were selected as 1, 2, 3, 4, 5 to capture parameters for minimal cost at acceptable convergence.

Metrics of the exploration of V, W and F cycle that varied maximum coarse grid level and iterations per sweep were given in Table 4 to Table 9 for cost and convergence respectively. Response surface and contour plot of the exploration of V cycle were given in Figure 29 to Figure 34 to visualise exploration metrics for cost and convergence respectively.

Table 4. Exploration metrics of V cycle with varying maximum coarse level and iterations per sweep regarding cost in reference work unit.

Cost in RWU				Iterations per Sweep				
				1	2	3	4	5
Max. Coarse Level	2	Min. Grid Res.	40x40	2.996	5.119	7.241	9.512	11.530
	3		20x20	2.974	5.110	7.246	9.405	11.510
	4		10x10	2.959	5.137	7.231	9.397	11.541
	5		5x5	2.979	5.116	7.253	9.410	11.540

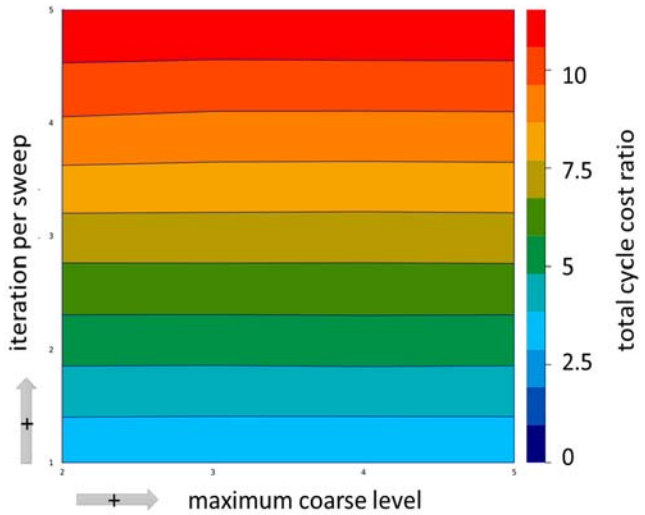

Figure 29. Contour plot of cost in reference work unit for exploration of V cycle.

Table 5. Exploration metrics of V cycle with varying maximum coarse level and iterations per sweep regarding convergence level of mean residual.

Convergence Level (Mean Residual)				Iterations per Sweep				
				1	2	3	4	5
Max. Coarse Level	2	Min. Grid Res.	40x40	~~2.45719~~	0.04608	0.03634	0.03056	0.02663
	3		20x20	~~3.55250~~	0.01975	0.01492	0.01204	0.01007
	4		10x10	~~5.52550~~	0.00703	0.00448	0.00300	0.00205
	5		5x5	~~6.83194~~	0.00137	0.00070	0.00048	0.00038

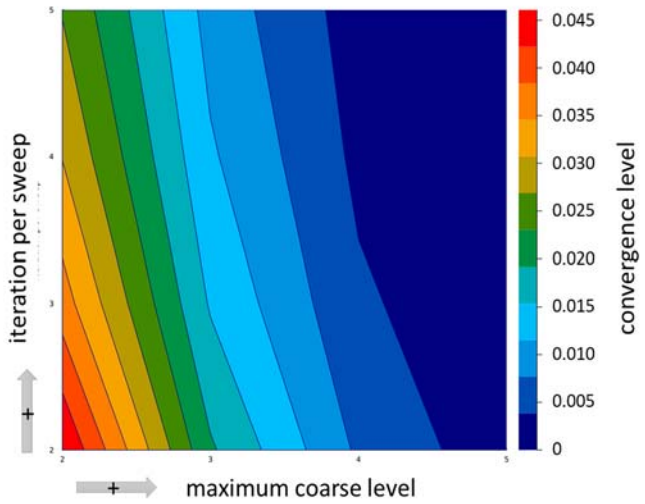

Figure 30. Contour plot of convergence of mean residual for exploration of V cycle.

Table 6. Exploration metrics of W cycle with varying maximum coarse level and iterations per sweep regarding cost in reference work unit.

Cost in RWU				Iterations per Sweep				
				1	2	3	4	5
Max. Coarse Level	2	Min. Grid Res.	40x40	3.523	5.593	7.953	10.334	12.671
	3		20x20	3.083	5.320	7.582	9.776	12.050
	4		10x10	3.083	5.300	7.523	9.770	11.972
	5		5x5	3.091	5.275	7.544	9.742	11.990

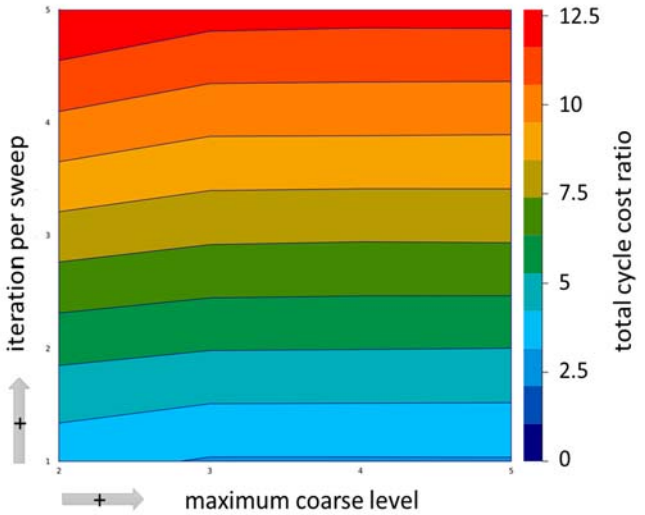

Figure 31. Contour plot of cost in reference work unit for exploration of W cycle.

Table 7. Exploration metrics of W cycle with varying maximum coarse level and iterations per sweep regarding cost in reference work unit.

Convergence Level (Mean Residual)				Iterations per Sweep				
				1	2	3	4	5
Max. Coarse Level	2	Min. Grid Res.	40x40	~~2.88776~~	0.02155	0.01660	0.01359	0.01152
	3		20x20	~~4.24468~~	0.00748	0.00494	0.00342	0.00242
	4		10x10	~~5.27066~~	0.00138	0.00068	0.00046	0.00036
	5		5x5	~~3.41220~~	0.00091	0.00064	0.00048	0.00039

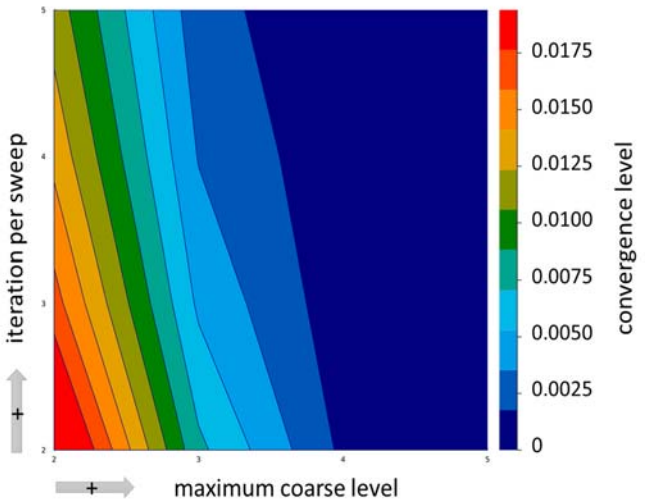

Figure 32. Contour plot of convergence of mean residual for exploration of W cycle.

Table 8. Exploration metrics of F cycle with varying maximum coarse level and iterations per sweep regarding cost in reference work unit.

Cost in RWU				Iterations per Sweep				
				1	2	3	4	5
Max. Coarse Level	2	Min. Grid Res.	40x40	3.062	5.260	7.485	9.668	11.885
	3		20x20	3.082	5.339	7.541	9.773	12.012
	4		10x10	3.075	5.325	7.562	9.789	11.999
	5		5x5	3.106	5.337	7.563	9.781	12.028

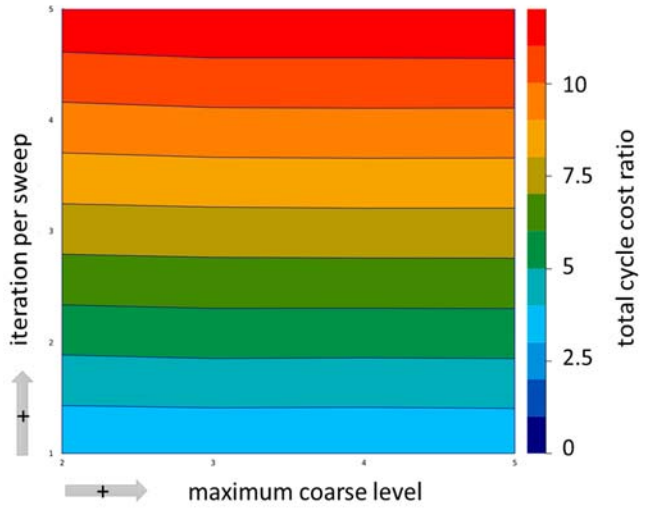

Figure 33. Contour plot of cost in reference work unit for exploration of F cycle.

Table 9. Exploration metrics of F cycle with varying maximum coarse level and iterations per sweep regarding cost in reference work unit.

Convergence Level (Mean Residual)				Iterations per Sweep				
				1	2	3	4	5
Max. Coarse Level	2	Min. Grid Res.	40x40	~~2.55164~~	0.03227	0.02530	0.02110	0.01823
	3		20x20	~~4.27735~~	0.00930	0.00644	0.00470	0.00352
	4		10x10	~~5.28459~~	0.00128	0.00065	0.00045	0.00036
	5		5x5	~~2.68124~~	0.00092	0.00064	0.00048	0.00039

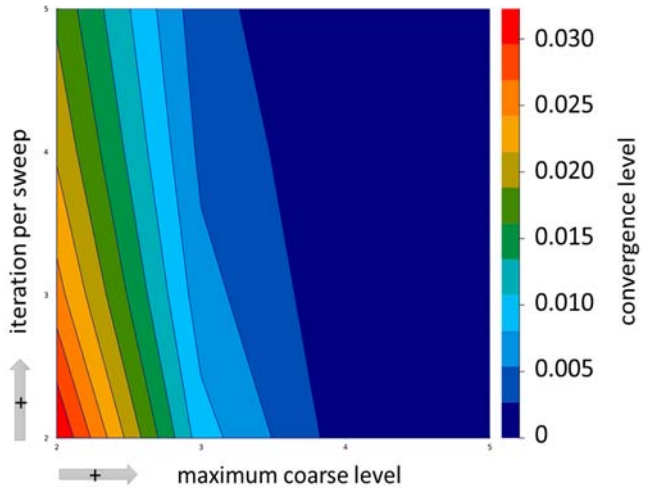

Figure 34. Contour plot of convergence of mean residual for exploration of F cycle.

Even though numbers do slightly change, explorations of V, W and F multigrid cycles resulted with same inferences. Single iterations per sweep generated unacceptable solutions as seen in Table 5, Table 7 and Table 9 regarding convergence levels of mean residual. Results of single iterations per sweep was also not comparable with other results of iterations per sweep and were excluded from illustrations.

Cost in reference work unit was directly proportional to iterations per sweep while maximum coarse level had little or no effect on cost. Convergence level of mean residual improved with increasing maximum coarse grid level. Although iterations per sweep had an effect on convergence, maximum coarse level was dominant factor relatively on convergence.

Comparison

Following the exploration of parameters (iteration per sweep and maximum coarse level) of multigrid cycles of fixed schemes; reference values for input parameters were set in order to present comparison results. Considering the exploration results of domain with 160x160 cells, it is observed that; for the grid resolution with 160x160 cells, 2 iteration per sweep and 5 coarsening was indeed satisfactory, regarding the convergence level. Since 2 was also the minimum number of iterations per sweep, differing values of input parameters were selected for comparison in order to include iteration per sweep contribution.

Comparison metrics are given in Table 10 with number of iterations per sweep as 5 and maximum coarse level as 5, while Figure 35 illustrates comparison of the residual distribution across the domain with 80x80 number of cells. Residual scatter plot given in Figure 35 was generated by marking all cells with higher than convergence target with red colour, thus enabling to filter how many cells actually converged rather than settling with monitoring mean residual.

First of all, multigrid cycles cost incredibly low compared to direct iterative solution. Multigrid cycles also performed significantly better compared to direct iterative method in terms of residual distribution while having approximately similar mean residual levels. Multigrid V, W and F cycle solutions appeared comparably same regarding residual distributions as well as mean residual levels.

Table 10. Comparison metric of Direct Iterative and Multigrid Cycle solutions.

	Direct Iterative	V Cycle	W Cycle	F Cycle
(Operation, Iteration) Count	(1, 1028)	(10, 45)	(20, 95)	(22, 105)
Mean Residual Convergence Level	0.001	0.00117	0.00122	0.00122
Converged Cell Number Percentage	54.6%	71.1%	72.4%	72.4%
Cost in Reference Work Units	1028.29	12.56	12.96	13.12
Cost with respect to Direct Iterative	100%	1.22%	1.26%	1.28%

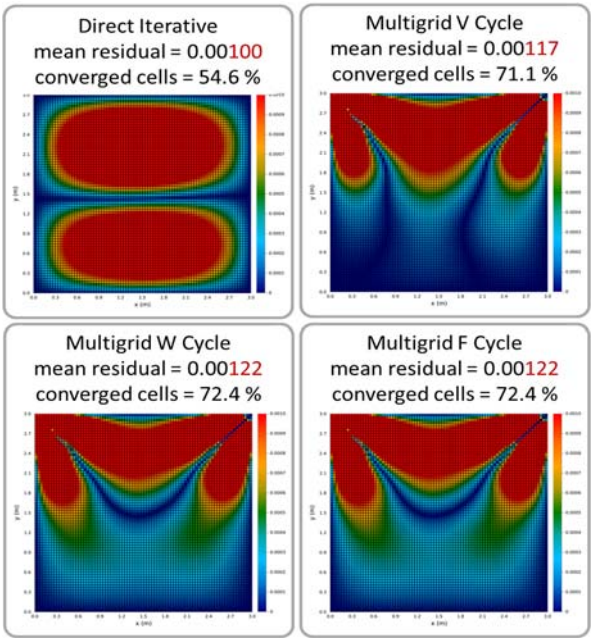

Figure 35. Comparison of converged cells of direct iterative and multigrid cycle solutions.

Figure 36 was given to illustrate residual progress with respect to computational cost while comparing direct iterative solution with multigrid cycle solutions. Figure 36 was designed with three frames with different zoom levels. At the first frame with logarithmic x axis, it is apparent that all multigrid cycles cost much less compared to direct iterative solutions regardless of the scheme. At the second frame, differences in effectivity between finest and coarser grid levels are emphasized while still comparing with the direct iterative solution progress. At the third frame, differences in solution progress between varying multigrid cycles are made visible.

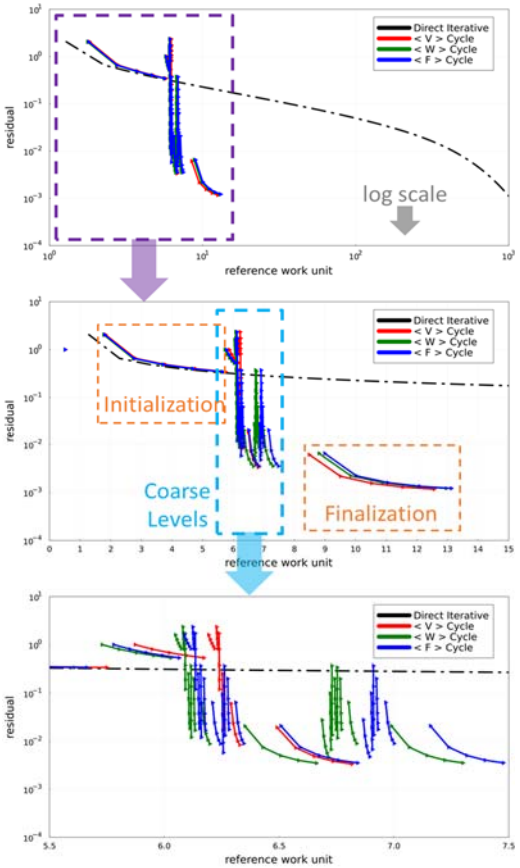

Figure 36. Effectivity of multigrid cycles compared to direct iterative method.

When compared, it was confirmed that residual progress with respect to computational cost curves found with study is indeed similar to literature (Brandt, 1973) as given in Figure 37 below.

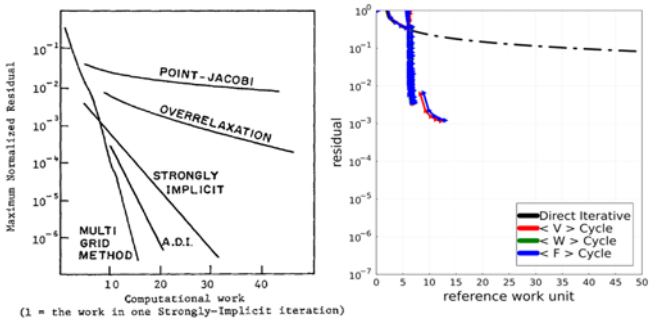

Figure 37. Effectivity of various methods (Brandt, 1973) (left) compared to study results (right).

Considering limitation of number of iterations per sweep, a unit cycle can be defined to generate reference values for minimum cost at fallout performance (convergence). Unit cycle was defined with V cycle scheme having minimum number of (2) iterations per sweep and maximum coarsest grid resolution level possible. Defining such reference enabled fair comparison of different multigrid cycles. Figure 38 was given to illustrate the metrics of the unit cycle for the reference case.

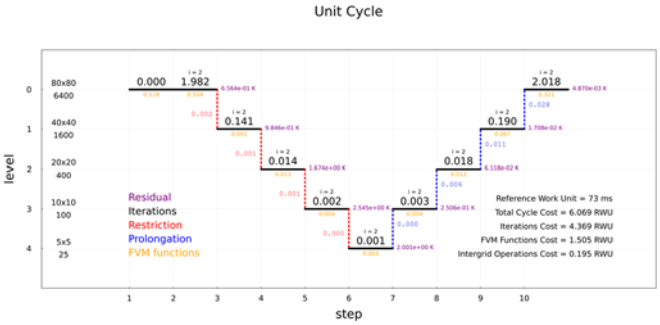

Figure 38. Detailed illustration of metrics of the unit cycle of the reference case.

Essential metrics of the results were given in Table 11 which presents clear inferences. All multigrid cycles at given input parameters performed comparably same while costing comparably same too. W and F cycle solutions may be slightly favorable considering residual distribution as seen in Figure 35. Intergrid operations and Finite Volume Methods costs consisted of approximately 12.5% of the total cost for all cycles. Iteration cost consisted almost 87.5% of the total cost for all cycles. Iterations on initialization and finalization phases cost the most of all as seen in Figure 36. Most of the effective work towards convergence was carried out on coarse grid resolutions. Iterations on initial grid resolution was also found to be the least effective in terms of convergence per computational cost. Figure 39 was given to illustrate residual progress with respect to computational cost while comparing unit cycle with multigrid cycles. Figure 40 was given to illustrate residual progress with respect to operation/iteration count in order to clearly present the scheme comparison of multigrid cycles.

Table 11. Multigrid cycle performance data compared to unit cycle.

COST	Unit Cycle	V Cycle	W Cycle	F Cycle
Intergrid Operations (RWU)	0.195	0.216	0.267	0.289
Finite Volume Method (RWU)	1.505	1.465	1.338	1.399
Iterative Solution (RWU)	4.369	10.881	11.359	11.435
Total (RWU)	6.069	12.561	12.964	13.123
Total (% of Unit Cycle)	100%	207%	214%	216%
Convergence (mean residual)	0.00487	0.00117	0.00122	0.00122

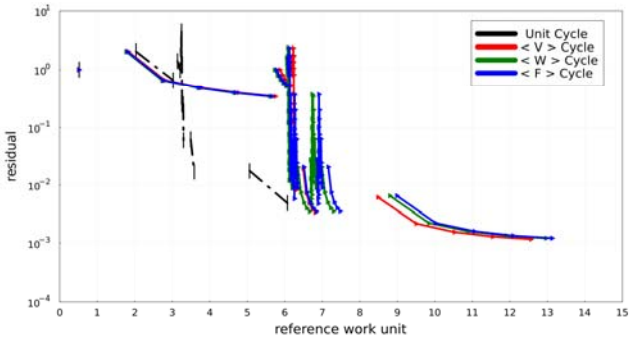

Figure 39. Multigrid cycle performance comparison plot with respect to reference work unit.

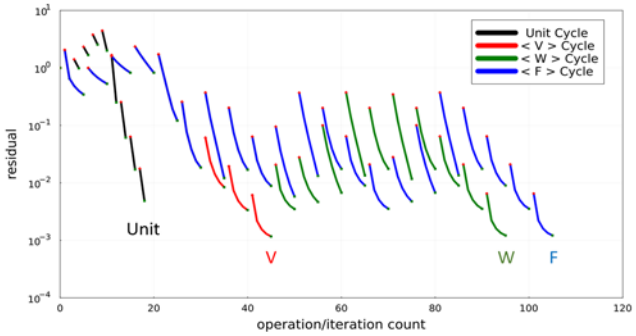

Figure 40. Multigrid cycle performance comparison plot with respect to operation/iteration count.

CONCLUSION

Simplifying a heat transfer problem with appropriate assumptions and modelling the problem over a two-dimensional domain and solving the problem with numerical methods would certainly provide advantages while also providing acceptable solution accuracy. One of the main drivers to simplify a case would be motivated by an optimization or exploration of parameters regarding a design or a process. Since the rationale of simplification is to get solutions from many runs as possible, any further convergence acceleration technique over numerical methods would be invaluable. Multigrid methods offer exceptional contribution to convergence acceleration of numerical methods while keeping solution accuracy intact if not better.

Multigrid cycles with fixed schemes proved to have incredibly reduced cost while having comparably same convergence level compared to direct iterative counterparts. Although the iterations on finest grid level seemed to have similar convergence rate compared to direct iterative counterpart; iterations on coarser grid levels had extremely high rate of convergence rate. It is imperative to state that most of the useful work towards convergence are carried out at coarse grid levels for multigrid cycle solutions. Cost reduction down to 1% of direct iterative solution infers that up to 100 more cases or parameter sets can be solved at the same time and/or computational cost.

Cost and convergence levels of multigrid cycles with fixed schemes were explored with varying iterations per sweep and maximum coarse grid level. Exploration of multigrid cycles with fixed schemes revealed that there is a minimum number of iterations per sweep for an acceptable solution accuracy. Iteration per sweep values lower than the minimum limit ended up with unacceptable solutions. Multigrid cycle cost increased with iterations per sweep while maximum coarse level did not affect cost. Little or no cost increase observed with increasing maximum coarse level parameter; because of the very nature of iterative solution which is in fact that solution cost increases with number of elements to be solved within the domain. Since coarser grid levels inherently have much smaller number of elements than finest grid level, it had almost no impact at cost. On the other hand, maximum coarse level was founded much more effective than iterations per sweep to obtain better convergence levels. It is because of the simple fact that convergence rate of solution at coarser grid levels are higher than finer grid levels. Furthermore, this effect is amplified exponentially with each consequent coarser grid level. Shortly, most of the computational work is done at minimal cost at coarse grid levels, while prolonging to finest grid level provides accurate solution.

Unit multigrid cycle was defined as a reference according to the results of the exploration of cycles with fixed schemes. Unit multigrid cycle was used to compare multigrid cycles within each other. Comparison results showed that establishing a reference baseline for cycles was indeed useful. Rationale beneath the utility of the unit cycle definition was the reference work unit definition. Reference work unit definition served as a non-dimensional cost indicator for any multigrid cycle. Reference work unit is now confidently suggested as an indicator to computational cost and proved to be simple yet comprehensive.

Effectivity of the multigrid cycles may be defined as convergence per cost at conceptual level. Uncontrollable and/or nuisance factors such as operating system, library performance, data structure, computational power; was blocked by the reference work unit definition. Thus, using the non-dimensional cost indicator enabled fair comparison of effectivity of multigrid cycles with fixed schemes. Although all of the (V, W, F) cycles performed similarly, slight nuances were observed. W and F cycles that traverse more through coarser grid levels ended up with slightly more evenly distributed residual across the domain; at a negligible computational cost compared to V cycle. In any situation, employing multigrid cycles with any scheme was proved extremely beneficial for iterative solution of heat transfer problems at two-dimensional domain, compared direct iterative methods.

Convection with a pressure-velocity coupling algorithm may be the first improvement to the code suite. Domains with varying dimensions (1D, 2D, 3D) would be a great addition to the tool. FVM methods that could operate on non-uniform grids will enhance applicability to complex geometries. Irregular or innovative multigrid schemes may be an interesting subject to study.

REFERENCES

Alpman, E. (2012). Blast Wave Simulations Using Euler Equations and Adaptive Grids, *Journal of Thermal Sciences and Technology*, 32 (2), 1-9.

Annaratone, D. (2010). *Engineering Heat Transfer*, Springer.

Arnone, A., Sestini, A. (1991). Multigrid Heat Transfer Calculations Using Different Iterative Schemes, *Numerical Heat Transfer, Part B: Fundamentals: An International Journal of Computation and Methodology*, 19 (1), 1-11.

Axelsson, O. (1996) *Iterative Solution Methods*, Cambridge University Press.

Aydar, E., Ekmekçi, İ. (2012). Thermal Efficiency Estimation of the Panel Type Radiators with CFD Analysis, *Journal of Thermal Sciences and Technology*, 32 (2), 63-71.

Aykan, F.S., Dursunkaya, Z. (2008). İki Boyutlu Yüzeylerde Isıl Aşınma Sayısal Analizi, *Isı Bilimi ve Tekniği Dergisi*, 28 (1), 43-49.

Bali, T. (2006). Numerical Analysis of Laminar and Turbulent Swirl Flows, *Journal of Thermal Sciences and Technology*, 26 (1), 1-8.

Brandt, A. (1973). Multi-level Adaptive Technique (MLAT) for Fast Numerical Solution to Boundary Value Problems, *Lecture Notes on Physics*, 18, Springer.

Brandt, A., Livne, O.E. (2011). *Multigrid Techniques: 1984 guide with applications to fluid dynamics*, Society for Industrial and Applied Mathematics.

Briggs, W.L., Henson, V.E., McCormick, S.F. (2000). *A Multigrid Tutorial*, Society for Industrial and Applied Mathematics.

Dawood, A.S., Burns, P.J (1992). Steady Three-Dimensional Convectivie Heat Transfer in A Porous Box Via Multigrid, *Numerical Heat Transfer, Part A: Applications: An International Journal of Computation and Methodology*, 22 (2), 167-198.

Doğan, A., Akkuş, S., Başkaya, Ş. (2012). Numerical Analysis of Natural Convection Heat Transfer from Annular Fins on a Horizontal Cylinder, *Journal of Thermal Sciences and Technology*, 32 (2), 31-41.

Galante, G., Rizzi, R.L. (2007). A Multigrid-Schwarz Method for the Solution of Hydrodynamics and Heat Transfer Problems in Unstructured Meshes, *19th International Symposium on Computer Architecture and High Performance Computing*, 87-94.

Internet, (2023). The Julia Programming Language, *Julia Micro-Benchmarks*, https://julialang.org/benchmarks/

Karaaslan, S., Hepkaya, E., Yücel, N. (2013). CFD Simulation of Longitudinal Ventilation Systems in a Scaled Short Tunnel, *Journal of Thermal Sciences and Technology*, 33 (1), 63-77.

Kürekçi, N.A., Özcan, O. (2012). An Experimental and Numerical Study of Laminar Natural Convection in a Differentially-Heated Cubical Enclosure, *Journal of Thermal Sciences and Technology*, 32 (1), 1-8.

Lai, Y.G., Przekwas, A.J. (1996). A Multigrid Algorithm for A Multiblock Pressure-Based Flow and Heat Transfer Solver, *Numerical Heat Transfer, Part B: Fundamentals: An International Journal of Computation and Methodology*, 30 (2), 239-254.

Lygidakis, G.N., Nikolos, I.K. (2014). Using a Parallel Spatial/Angular Agglomeration Multigrid Scheme to Accelerate the FVM Radiative Heat Transfer Computation—Part I: Methodology, *Numerical Heat Transfer, Part B: Fundamentals: An International Journal of Computation and Methodology*, 66 (6), 471-497.

Lygidakis, G.N., Nikolos, I.K. (2014). Using a Parallel Spatial/Angular Agglomeration Multigrid Scheme to Accelerate the FVM Radiative Heat Transfer Computation—Part II: Numerical Results, *Numerical Heat Transfer, Part B: Fundamentals: An International Journal of Computation and Methodology*, 66(6), 498-525.

Mançuhan, E., Küçükada, K., Alpman, E. (2011). Mathematical Modeling and Simulation of the Preheating Zone of a Tunnel Kiln, *Journal of Thermal Sciences and Technology*, 31 (2), 79-86.

Onur, N., Turgut, O., Arslan, K. (2011). Three-Dimensional Numerical Analysis of Forced Convection Flow and Heat Transfer in a Curved Square Duct, *Journal of Thermal Sciences and Technology*, 31 (2), 13-24.

Patil, P.V., Prasad, K. (2014). Numerical Solution for Two Dimensional Laplace Equation with Dirichlet Boundary Conditions, *IOSR Journal of Mathematics*, 6, 66-75.

Sert, Z., Timuralp, Ç., Tekkalmaz, M. (2019). Heat Transfer in Three-Dimensional Rectangular Cavities with Pins, *Journal of Thermal Sciences and Technology*, 39 (1), 39-49.

Şimşek, B., Uslu, S., Ak, M.A. (2020). Validation of Aerodynamic Heating Prediction Tool, *Journal of Thermal Sciences and Technology*, 40 (1), 53-63.

Tang, L., Joshi, Y.K. (1999). Application of Block-Implicit Multigrid Approach to Three-Dimensional Heat Transfer Problems Involving Discrete Heating, *Numerical Heat Transfer, Part A: Applications: An International Journal of Computation and Methodology*, 35 (7), 717-734.

Trottenberg, U., Oosterlee, C.W., Schüller, A. (2001). *Multigrid*, Academic Press.

Uğurlubilek, N. (2012). Numerical Investigation of Heat Transfer and Flow in a Twisted-Shaped Square Duct, *Journal of Thermal Sciences and Technology*, 32 (2), 121-131.

Uzuner, M.K., Başol, A.M., Mischo, B., Jenny, P. (2023). Numerical Analysis and Diffuser Vane Shape Optimization of a Radial Compressor with the Open-Source Software SU2, *Journal of Thermal Sciences and Technology*, 43 (2), 233-242.

Versteeg, H., Malalasekera, W. (2007). *An Introduction to Computational Fluid Dynamics: The Finite Volume Method*, Pearson.

Vierendeels, J., Merci, B., Dick, E. (2004). A multigrid method for natural convective heat transfer with large temperature differences, Journal of Computational and Applied Mathematics, 168, 509-517.

Wang, Q., Joshi, Y. (2006). Algebraic Multigrid Preconditioned Krylov Subspace Methods for Fluid Flow and Heat Transfer on Unstructured Meshes, *Numerical Heat Transfer, Part B: Fundamentals: An International Journal of Computation and Methodology*, 49 (3), 197-221.

Wesseling, P. (1992). *An Introduction to Multigrid Methods*, John Wiley & Sons.

Yan, J., Thiele, F. (1998). Performance and Accuracy of a Modified Full Multigrid Algorithm for Fluid Flow and Heat Transfer, *Numerical Heat Transfer, Part B: Fundamentals: An International Journal of Computation and Methodology*, 34 (3), 323-338.

Yetik, Ö., Mahir, N. (2020). Flow and Forced Heat Transfer from Tandem Square Cylinders Near a Wall, *Journal of Thermal Sciences and Technology*, 40 (1), 99-112.

Yıldızeli, A., Çadırcı, S. (2023). Numerical Investigation of Plate Cooling Using Multiple Impinging Jets in Different Alignments, *Journal of Thermal Sciences and Technology*, 43 (1), 1-10.