

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

INTEL CORPORATION,
Petitioner,

v.

ADVANCED CLUSTER SYSTEMS, INC.,
Patent Owner.

IPR2025-00916
U.S. Patent No. 11,811,582

SUPPLEMENTAL DECLARATION OF CHANDRAJIT L. BAJAJ, PH.D.,

TABLE OF CONTENTS

I.	INTRODUCTION.....	3
II.	PRIOR ART ANALYSIS	4
	A. Menon/Trefethen teaches the claimed “evaluation flow” in the Challenged Claims	4
	1. Menon/Trefethen’s MultiMATLAB system performs the claimed evaluation flow.....	4
	2. Patent Owner’s arguments ignore the evaluation flow taught by Menon/Trefethen.....	9
	B. The Challenged Claims do not require Patent Owner’s extra “programmable return” requirement.....	10
	C. Menon, Trefethen, and POEref teach the “cluster configuration module” recited in claim 4.....	13
	1. The ’582 specification indicates that the “cluster configuration module” can be an internal module	15
	D. Combining Menon and Trefethen would have been obvious to a POSITA 16	
	1. Menon and Trefethen teach how to implement MultiMATLAB.....	16
	2. Menon does not teach away from the combination	18
	E. A POSITA would have known how to locate the cited technical materials 19	
III.	DECLARATION	21

I. INTRODUCTION

1. I, Chandrajit L. Bajaj, have been retained by counsel for Intel Corporation (“Petitioner”) as a technical expert in connection with the proceeding identified above. I submit this declaration in support of Intel’s Petitioner Reply in pending *Inter Partes* Review (“IPR”) IPR2025-00916 of claims 1-20 (the “Challenged Claims”) of U.S. Patent No. 11,811,582 (“the ’582 patent”).

2. My qualifications are summarized in my prior declaration in this case (Ex.1003, ¶¶7-17) and in my CV (Ex.1004).

3. I am being compensated for my work in this matter at an hourly rate. I am also being reimbursed for reasonable and customary expenses associated with my work and testimony in this matter. My compensation is not contingent on the outcome of this matter or the specifics of my testimony. I have no personal or financial stake or interest in the outcome of this proceeding.

4. I have been asked to provide opinions regarding specific issues raised in Patent Owner’s Response (Paper 21) and in Petitioner’s Reply being filed concurrently with this declaration.

5. In the preparation of this declaration, I have considered the materials listed in my prior declaration (Ex.1003, ¶4), Patent Owner’s Response (Paper 21), the Institution Decision (Paper 15), and any other materials cited in this declaration.

6. In forming my opinions, I also have considered: the relevant legal standards, including the standard for obviousness, which Intel’s counsel has explained to me and which are described in my prior declaration (*see* Ex.1003, ¶¶18-23); and my own knowledge and experience based upon my work in the field of cluster computing, which includes computer cluster systems and cluster computing techniques as described below; and additional relevant background materials cited in this declaration.

7. This declaration applies the same level of ordinary skill in the art described in my prior declaration. *See* Ex.1003, ¶¶24-27.

8. Throughout my analysis below, I have used *italics* to indicate claim language, and **bolded** text in quoted material to indicate emphasis. Unless otherwise noted, all emphasis in any quoted material has been added.

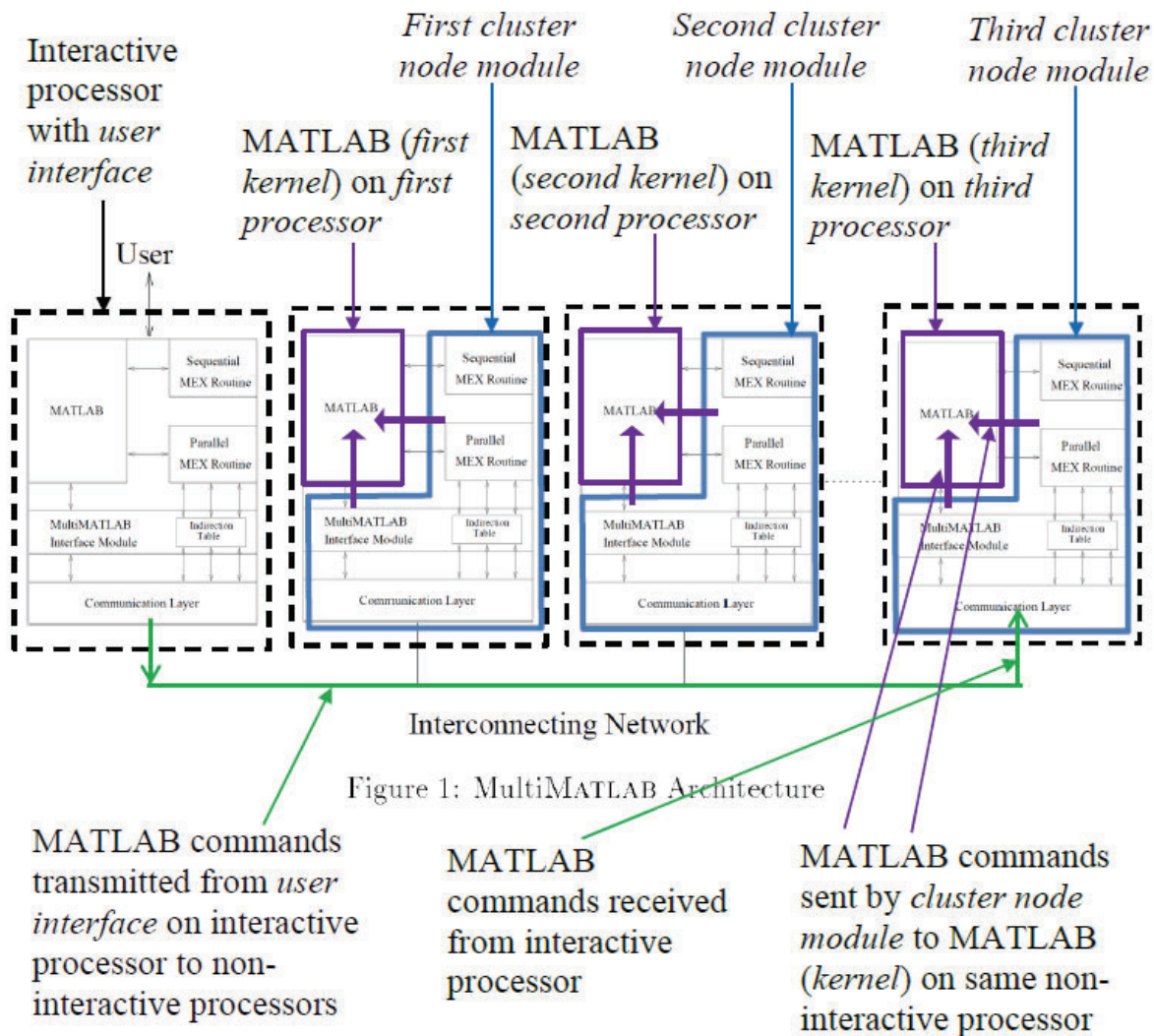
II. PRIOR ART ANALYSIS

A. Menon/Trefethen teaches the claimed “evaluation flow” in the Challenged Claims

1. Menon/Trefethen’s MultiMATLAB system performs the claimed evaluation flow

9. I understand that Patent Owner refers to the claim’s result-passing limitations as the “multi-stage, dependent evaluation flow,” and argues that Menon/Trefethen do not disclose kernels using other kernels’ results as inputs to further computation. POR, 15-19.

10. As described in my prior declaration, Menon teaches a “MultiMATLAB” system with interconnected “processors” (a “first processor,” a “second processor,” and a “third processor” as shown in Menon’s FIG. 1). Ex.1005, 3, 6, FIG. 1; Ex.1003, ¶¶178-185. Each “processor” includes a *kernel* (“MATLAB”) and a *cluster node module* as shown in the following figure from my prior declaration:



Ex.1005, FIGs. 1 and 2 (combined and annotated); Ex.1003, ¶258

11. Trefethen provides example code describing the sequence of operations that MultiMATLAB's interconnected processors perform:

```
if ID==0                % first process: send
    a = 1
    Send(ID+1,a)
elseif ID == Nproc-1    % last process: receive and double
    a = 2*Recv
else                    % middle processes: receive, double, and send
    a = 2*Recv
    Send(ID+1,a)
end;
```

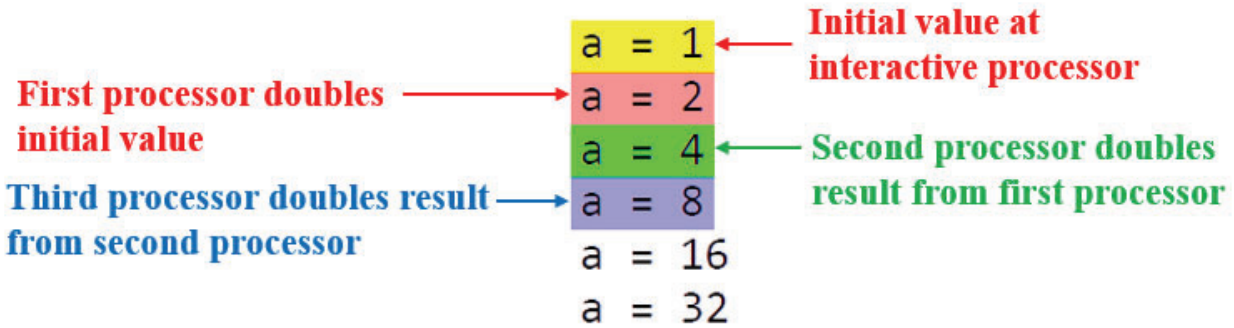
Process 0 creates the variable *a* with value 1 and sends it to process 1. Process 1 receives the message, doubles the value of *a*, and sends it along to process 2; and so on. If there are six processors the command Eval('cycle') produces the output

```
a = 1
a = 2
a = 4
a = 8
a = 16
a = 32
```

Ex.1006, 6 (annotated); Ex.1003, ¶225.

12. As described above, (i) Process 0 (i.e., the interactive processor) “creates the variable *a* with a value 1” and sends the variable to process 1 (i.e., the first non-interactive processor), then (ii) process 1 (first non-interactive processor) “receives the message” from the process 0 (interactive processor) and “doubles the value of *a*” (now *a*=2), and “sends it along” to process 2 (i.e., the second non-interactive processor) and the sequence continues. Ex.1006, 6; Ex.1003, ¶199. Trefethen further describes that each MATLAB process transmits its output “to the master process” (i.e., the first cluster node module on the interactive processor) “as soon as it is ready.” Ex.1006, 4; Ex.1003, ¶¶126, 349. Thus, Trefethen discloses both inter-process data passing by Send/Recv and return of process output to the

interactive process for presentation to the user. When this sequence is executed in an example MULTIMATLAB system including six nodes, the user interface of the interactive processor produces the following output:



Ex.1006, 6 (annotated); Ex.1003, ¶¶225, 249-250.

13. As shown, the value of *a* at the interactive processor is 1 (“*a*=1” in first line). Ex.1006, 6; Ex.1003, ¶¶127, 199. The value of *a* from the interactive processor (1) is doubled by the first non-interactive processor to 2 (“*a*=2” in second line). Ex.1006, 6; Ex.1003, ¶199. The value of *a* from the first non-interactive processor (2) is doubled again by the second non-interactive processor to 4 (“*a*=4” in third line). Ex.1006, 6; Ex.1003, ¶¶223-226. The value of *a* from the second non-interactive processor (4) is doubled again by the third non-interactive processor to 8 (“*a*=8” in fourth line). Ex.1006, 6; Ex.1003, ¶¶247-251. That value of *a* from the third non-interactive processor (“*a*=8”) is then available for further communication/output in MultiMATLAB through the same Send/Recv and output mechanisms described in Trefthen. Ex.1006, 5-7; Ex.1003, ¶¶247-251.

14. The `cycle.m` example is not merely an example of independent parallel jobs. The value `a` is a MATLAB variable transmitted from one process to the next. Each receiving process uses the received value as the input for its next computation, modifies that value, and then sends the modified value on. A POSITA would understand this as a dependent evaluation sequence: process 2's result (`a=4`) is the input received and used by process 3 to compute `a=8`. Ex.1006, 6; Ex.1003, ¶¶223-226, 247-251.

15. The example uses a simple variable for clarity, but MATLAB variables are arrays by design, and Menon and Trefethen expressly teach matrix and matrix-vector computations. Ex.1006, 4-5; Ex.1005, 10. A POSITA would therefore understand that the same Send/Recv mechanisms used to pass `a` in `cycle.m` can pass arrays/matrices, and that the MATLAB kernels are configured to perform matrix operations on received arrays/matrices.

16. Menon and Trefethen also provide examples of MATLAB performing mathematical evaluations of matrices and their elements. Ex.1006, 4-5; Ex.1005, 10; Ex.1003, ¶¶196-204. For example, Trefethen describes the computation of ‘`cond(hilb(ID))`’ at MATLAB process IDs=4 and 5 to return the results “`ans = 1.5514e+04` [and] `ans = 4.7661e+05`, the condition numbers of the Hilbert matrices of dimensions 4 and 5.” Ex.1006, 4; Ex.1003, ¶202. And Menon describes MATLAB as performing “high-level MATLAB operations” such as “matrix

multiplications [and] solve for eigenvalues,” and provides example “MultiMATLAB implementations of the conjugate gradients algorithm” that “consists of a matrix-vector multiplication at every iteration.” Ex.1005, 10; Ex.1003, ¶201.

17. Menon and Trefethen also teach that each MATLAB process (e.g., the first non-interactive process) is configured to receive a result from any other MATLAB process via its cluster node module, and to perform subsequent processing and returning the final result to the interactive processor. Ex.1005, 11-14; Ex.1011, 64; Ex.1003, ¶267. For example, Menon teaches an algorithm (the conjugate gradient algorithm) in which each kernel is processing, in parallel, and awaiting results from the other processes to iteratively calculate further results, which includes the third process using the results from the second process and the first process using the results from the third process, and ultimately displaying the final result to the user via the user interface. Ex.1005, 11-14; Ex.1003, ¶267.

2. Patent Owner’s arguments ignore the evaluation flow taught by Menon/Trefethen

18. I understand that Patent Owner argues that “Petitioner does not identify any disclosure in Menon or Trefethen in which **a kernel receives a result computed by another kernel** and performs additional matrix operations on that received result.” POR, 16. But as described above, this overlooks the dependent

data-flow taught by Trefethen's cycle.m example and the matrix-processing capability taught by both Menon and Trefethen.

19. For example, in its description of the "cycle.m" script, Trefethen discloses that "[p]rocess 0" "creates the variable *a* with value 1 and sends it to process 1" (the *first kernel* on the *first processor*). Ex.1006, 6; Ex.1003, ¶¶127, 199. "Process 1" (the *first kernel* on the *first processor*) "**receives the message, doubles the value of *a*, and sends it along to process 2**" (the second kernel on the second processor) "and so on." Ex.1006, 6; Ex.1003, ¶225. The next process therefore receives the computed value from the previous process and uses that value as the input for its next computation. And as described above, Menon and Trefethen both teach that the calculations performed at each kernel can be matrix operations. Ex.1006, 4-5; Ex.1005, 10.

B. The Challenged Claims do not require Patent Owner's extra "programmatic return" requirement

20. The Challenged Claims recite "said third kernel configured to...return a second result to one or more of the first, second, or third cluster node modules using the third processor" (limitation [1.7.2]) and "the first cluster node module is configured to return, based on the second result, a final result." (limitation [1.9]). The claims do not use the phrase "programmatic return", do not require that the returned final result be stored as a reusable workspace variable, and do not exclude returning the computed result to a user interface for presentation.

21. I understand that Patent Owner characterizes the claims as requiring “programmatic return of evaluation results.” POR, 11-15. I disagree with that interpretation from a technical perspective. A POSITA would read “return” in these claims to cover communication of computed data/output back through the cluster-node software to the relevant receiving component. The claim language itself specifies additional use where additional use is required but does not require any extra storage or reuse step after the final result is returned.

22. Trefethen describes that, for each MATLAB process, its “output is sent to the master process” (i.e., the interactive processor) “as soon as it is ready.” Ex.1006, 4; Ex.1003, ¶¶348-352.

23. I understand that Patent Owner argues that the Challenged Claims require “a cluster node module return a final result” and that “[s]creen output is not a returned result in this sense.” POR, 13. Trefethen’s cycle.m example uses Send/Recv to transfer the MATLAB variable *a* between processes. Ex.1006, 5-6. The displayed output confirms the values computed by the processes; the script itself shows the data being received, modified, and sent.

24. A POSITA would recognize that Menon/Trefethen teaches that each MATLAB process includes the same cluster-node communication functionality and is configured for point-to-point communications with other processes. Ex.1005, 3-5; Ex.1006, 5-7. Thus, the first cluster node module of the first non-interactive

processor is configured to receive a result from the third kernel/third cluster node module, perform or coordinate subsequent processing based on the received result as needed, and return a final result to the interactive processor, where *the user interface displays it*. Ex.1006, 6-7; Ex.1003, ¶¶262-270.

25. In my opinion, the claims do not recite any particular further use after the first cluster node module returns the final result. Other portions of the claim recite further use (e.g., the third kernel performs a matrix operation using the result received from the second kernel), yet the claims contain no such language requiring any particular further use after the “final result” is “returned” by the first cluster node module.

26. There is also no meaningful technical distinction in the claims between a computed value returned for display and a “final result.” Those values are data. For example, Trefethen expressly states that MultiMATLAB uses “Eval, Send, [and] Recv” to “pass commands and **data** between [] these various [MATLAB] processes.” Ex.1006, Abstract. The fact that a computed data value is displayed to the user after being returned does not change the fact that it is a computed result.

27. Patent Owner’s “screen output” distinction is also inconsistent with the claim phrase “return the final result to the user interface.” (e.g., claim 14). A user interface is precisely the part of the system that presents information to the user. When MultiMATLAB sends the computed output from a non-interactive process to

the master/interactive process for display, the computed result is being returned to the user interface in the ordinary technical sense of that phrase. Ex.1006, 4, 6.

28. Nothing in the Challenged Claims requires that the final result be stored in the first node’s MATLAB workspace as a reusable variable after it is displayed. Where the claims require later use of a value, they say so expressly—for example, the third kernel executes additional matrix operations using “the first result.” Ex.1001, 29:49-54.

29. Dependent claim 14 further confirms that a returned result may be returned to a user interface. Claim 14 recites returning the final result “to the user interface.” Ex.1001, claim 14. A user interface is the component that presents information to the user. From a technical perspective, Patent Owner’s categorical exclusion of user-interface output is inconsistent with that claim language.

C. Menon, Trefethen, and POEref teach the “cluster configuration module” recited in claim 4

30. Claim 4 recites “a cluster configuration module that initializes at least one of the first cluster node module, the second cluster node module, or the third cluster node module.” The ’582 patent describes a cluster configuration module as “includ[ing] program code for initializing a plurality of cluster node modules to add cluster computing support to computer systems.” Ex.1001, 22:4-6.

31. Menon similarly describes that a user starts a MATLAB process on each processor of the MultiMATLAB architecture by using IBM's POE, described in POEref (Ex.1008). Ex.1005, 6 ("all MATLAB processes are started via POE, IBM's Parallel Operating Environment"); Ex.1003, ¶¶393-394.

32. As described in my prior declaration, POEref describes that the initialization process is started using the "poe" command. Ex.1008, 28 ("When you invoke poe, the Partition Manager allocates processor nodes for each task and initializes the local environment"); Ex.1003, ¶¶393-394. The poe initialization "loads your program, and reproduces your local environment, on each processor node." Ex.1008, 28; Ex.1003, ¶¶393-394.

33. POE is not merely a bare hardware-allocation mechanism. POEref states that when poe is invoked, the Partition Manager allocates processor nodes for each task and "initializes the local environment," and that POE "loads your program, and reproduces your local environment, on each processor node." Ex.1008, 28. In Menon's MultiMATLAB implementation, the program being started includes the MATLAB process and its MultiMATLAB interface/module functionality. Ex.1005, 3-6. A POSITA would therefore understand POE's initialization to initialize the software environment in which the cluster node modules operate.

34. The claim language also does not require the cluster configuration module to be named “cluster configuration module.” What matters technically is whether the cited system has program code that initializes at least one of the cluster node modules. Ex.1001, 22:4-6. Menon’s use of POE and POEref’s description of the poe initialization process satisfy that functional requirement from a POSITA’s perspective. Ex.1005, 6; Ex.1008, 28; Ex.1003, ¶¶393-394.

1. **The ’582 specification indicates that the “cluster configuration module” can be an internal module**

35. I understand that Patent Owner argues that claim 4 “is not satisfied by any mechanism that merely launches processes or allocates hardware resources” and requires “**an internal module of the claimed system...**that performs initialization of cluster node modules so that they can operate together in the claimed environment.” POR, 19. But the ’582 specification describes a “**cluster configuration module stored** on one or more of the computer systems 110, 120, 130 or **on a remote computer system.**” Ex.1001, 11:36-39; POR 19. A module stored on a remote computer system is not limited to an “internal” module in the sense Patent Owner asserts.

D. Combining Menon and Trefethen would have been obvious to a POSITA

1. Menon and Trefethen teach how to implement MultiMATLAB

36. My prior declaration (Ex.1003, ¶¶77-85), describes multiple reasons why a POSITA would have been motivated to combine Menon and Trefethen, including:

- Menon suggests the combination by **expressly citing** Trefethen and explaining that Menon builds on the teachings of Trefethen. Ex.1005, 1-2; Ex.1003, ¶83.

- Menon and Trefethen both describe aspects of how the same MultiMATLAB system enables MATLAB to perform parallel mathematical calculations. Ex.1005, Abstract, 16; Ex.1006, 2; Ex.1003, ¶84.

- Menon and Trefethen together represent a combination of prior art elements according to known methods (executing MATLAB code on a multi-node system) to yield the predictable result discussed in both references: the MultiMATLAB system executing MATLAB in parallel to perform mathematical calculations. Ex.1005, Abstract; Ex.1006, 2-7; Ex.1003, ¶82.

37. A POSITA would have had an especially strong reason to read these references together because they address the same project, the same MATLAB-based parallel-computing problem, and the same MultiMATLAB command

environment. Ex.1005, 1-2; Ex.1006, 2-7. Menon expressly identifies Trefethen's MultiMATLAB work and then describes a redesigned architecture that preserves the same user-level objective—parallel MATLAB computations—while improving implementation details and extensibility. Ex.1005, 2, 14.

38. A POSITA therefore would not view the references as mutually exclusive alternatives. Rather, a POSITA would understand Trefethen as teaching specific MultiMATLAB commands and examples, including Eval, Send, Recv, and the cycle example, and Menon as teaching how the MultiMATLAB architecture was implemented and improved in a later version. Ex.1005, 2-4, 14; Ex.1006, 4-7. A POSITA would use Trefethen's command-level teachings together with Menon's improved MultiMATLAB implementation.

39. In computer systems, a later paper describing an improved implementation of the same system is a natural source to combine with an earlier paper that provides user-level commands and examples. A POSITA would not understand Menon's improved implementation to erase Trefethen's command examples; instead, a POSITA would use those examples to understand the MultiMATLAB environment that Menon continued to improve.

2. Menon does not teach away from the combination

40. I understand that Patent Owner argues that Menon “teaches away” from Trefethen because Menon describes the “MultiMATLAB architecture” of Trefethen as “different from the newer one described in” Menon. POR, 31-35. I disagree.

41. Menon does not identify differences between the first “MultiMATLAB” architecture and the newer architecture, but any such differences do not teach away from using Trefethen’s command-level teachings. Ex.1005, 2. For example, Menon describes a list of “goals” already achieved by MultiMATLAB and then states that the architecture has been “redesigned to achieve two further goals.” Ex.1005, 2. One of these two “further goals” is “extensibility.” Menon achieves this by describing the addition of “MEX routines” to the MultiMATLAB system. Ex.1005, 2-3. The “MEX routines” do not show an “incompatibility” between Menon and Trefethen. Instead, Menon’s addition of the “MEX routines” to MultiMATLAB shows that such a modification was within the capabilities of a POSITA because the authors of the Menon paper actually implemented the modification. Ex.1005, 2-3.

42. In my opinion, a POSITA reading Menon would not have been led to disregard the teachings of Trefethen, something Menon itself does not do. Ex.1005,

1-2 (citing Trefethen). Accordingly, Menon does not teach away from the combination with Trefethen.

43. Menon’s discussion of the first architecture’s “performance limitations” and lack of support for parallel MEX extensions would not discourage a POSITA from using Trefethen’s teachings. On the contrary, Menon identifies those limitations as reasons for the redesign and then explains that the newer architecture achieves further goals. Ex.1005, 2, 14. A POSITA would understand that critique as a motivation to improve MultiMATLAB while retaining relevant teachings of the earlier work, not as a warning that Trefethen’s disclosed commands and examples should be discarded.

44. I therefore disagree with Patent Owner’s assertion that Menon teaches away. Menon does not state that Trefethen’s Eval, Send, Recv, or data-passing examples are inoperable. It cites Trefethen, discusses the same MultiMATLAB system, and explains modifications that improve the architecture. Ex.1005, 1-2, 14. In my opinion, this is evidence of a reason to combine the teachings, not evidence that a POSITA would have avoided them.

E. A POSITA would have known how to locate the cited technical materials

45. I understand that Patent Owner also argues that Menon, Trefethen, RS6000, POEref, and MPIref were not publicly accessible. POR, 40-47. From a technical perspective, these are precisely the kinds of materials a POSITA working

on MATLAB-based parallel computing and IBM SP cluster implementations would have known to consult. *See* Ex.1003, ¶¶121-144.

46. Menon was an ACM/IEEE Supercomputing paper directed to “Integrating MATLAB with High-Performance Parallel Computing.” Ex.1005, 1. Supercomputing conference proceedings were a natural source for a POSITA interested in high-performance parallel computing. Menon also expressly cites Trefethen’s MultiMATLAB work and discusses the same MultiMATLAB system. Ex.1005, 1-2. A POSITA reading Menon would therefore have been led to Trefethen for command-level examples of Eval, Send, Recv, and cycle.m.

47. Similarly, RS6000 and POEref are IBM technical materials corresponding to the IBM SP/RS6000 and POE environment that Menon expressly identifies as the platform for MultiMATLAB. Ex.1005, 5-6. A POSITA implementing or evaluating MultiMATLAB on that platform would have had a technical reason to consult those IBM materials. And MPIref is the MPI standard that Menon expressly invokes because of MPI’s popularity, suitability for high-performance parallel computing, and vendor-optimized implementations. Ex.1005, 5; Ex.1017, 1. These facts reinforce my opinion that the references are not hindsight-selected, disconnected materials.

III. DECLARATION

I declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true, and that these statements were made with knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under section 1001 of Title 18 of the United States Code.

Dated: April 30, 2026



Chandrajit L. Bajaj, Ph.D.