

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

INTEL CORPORATION,
Petitioner,

v.

ADVANCED CLUSTER SYSTEMS, INC.,
Patent Owner.

IPR2025-00914
U.S. Patent No. 12,021,679

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

TABLE OF CONTENTS

I.	INTRODUCTION.....	3
II.	PRIOR ART ANALYSIS	5
	A. Menon/Trefethen teaches the “1→2→3→1” order of operations in the Challenged Claims	5
	1. Menon/Trefethen’s MultiMATLAB system performs the claimed order of operations	5
	2. “Master-controlled dispatch model” is not a distinction.....	8
	3. The claims do not exclude “user-authored application code”	10
	B. Combining Menon and Trefethen would have been obvious to a POSITA 18	
	1. Menon and Trefethen teach how to implement MultiMATLAB.....	18
	2. Menon does not teach away from the combination	20
	C. A POSITA would have known how to locate the cited technical materials 22	
III.	DECLARATION	24

I. INTRODUCTION

1. I, Chandrajit L. Bajaj, have been retained by counsel for Intel Corporation (“Petitioner” or “Intel”) as a technical expert in connection with the proceeding identified above. I submit this supplemental declaration in support of Intel’s Petitioner Reply in pending *Inter Partes* Review (“IPR”) IPR2025-00914 of claims 1-19 (the “Challenged Claims”) of U.S. Patent No. 12,021,679 (“the ’679 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 22, “POR”), the Board’s Institution Decision (Paper 16), and in Petitioner’s Reply being filed concurrently with this declaration.

5. In the preparation of this declaration, I have considered all materials listed my prior declaration (Ex.1003, ¶4) 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, including computer cluster systems and cluster computing techniques as described below; and relevant background additional 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.

9. For ease of reference, I use “POR” to refer to Patent Owner’s Response (Paper 22). I understand the Board’s Institution Decision was preliminary, and my opinions below are directed to the technical issues Patent Owner raises in the POR.

II. PRIOR ART ANALYSIS

A. Menon/Trefethen teaches the “1→2→3→1” order of operations in the Challenged Claims

1. Menon/Trefethen’s MultiMATLAB system performs the claimed order of operations

10. I understand that Patent Owner refers to data passing operations between nodes as the “1→2→3→1 Order of Operations.” POR, 21.

11. As described in my prior declaration, Menon teaches a “MultiMATLAB” system with interconnected nodes (“first node,” a “second node,” and a “third node” as shown in Menon’s FIG. 1). Ex.1005, 3, 6, FIG. 1; Ex.1003, ¶¶197-223:

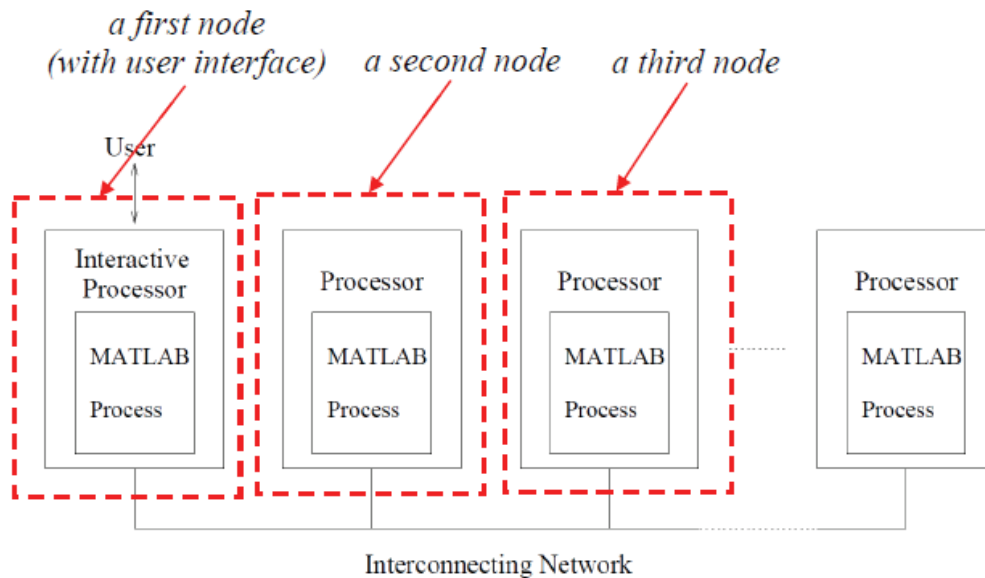


Figure 1: MultiMATLAB Architecture

Ex.1005, FIG. 1 (annotated); Ex.1003, ¶210

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

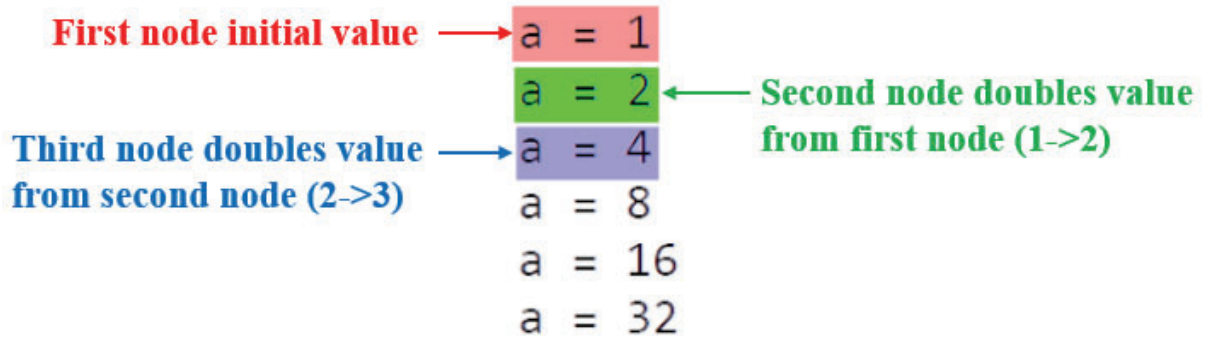
```
if ID==0                % first process: send
    a = 1
    Send(ID+1,a)
elseif ID == Nproc-1    % last process: receive and double
    a = 2*Recv
else                    % middle processes: receive, double, and send
    a = 2*Recv
    Send(ID+1,a)
end;
```

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, ¶183.

13. In Trefethen's example, (i) Process 0 (i.e., the first node) “creates the variable *a* with a value 1” and sends the variable to process 1 (i.e., the second node), then (ii) process 1 (second node) “receives the message” from the process 0 (first node) and “doubles the value of *a*” (now *a*=2), and “sends it along” to process 2 (i.e., the third node). Ex.1006, 6; Ex.1003, ¶111. Trefethen further describes that each MATLAB process transmits its output “to the master process” (i.e., the first node) “as soon as it is ready.” Ex.1006, 4; Ex.1003, ¶215. When this sequence is executed in an example MULTIMATLAB system including six nodes, the user interface of the first node produces the following output:



Ex.1006, 6 (annotated); Ex.1003, ¶214

14. As shown, the value of *a* at the first node is 1 (“*a*=1” in the first line).

Ex.1006, 6; Ex.1003, ¶¶211-215. The value of *a* from the first node (1) is doubled

by the second node to 2 (“*a*=2” in the second line). Ex.1006, 6; Ex.1003, ¶¶211-

215. The value of *a* from the second node (2) is doubled again by the third node to 4

(“*a*=4” in the third line). Ex.1006, 6; Ex.1003, ¶¶211-215. And that value of *a* from

the third node (“*a*=4”) is sent to the first node and displayed to the user. Ex.1006, 6-

7; Ex.1003, ¶¶220-222.

15. This example is not merely a theoretical capability divorced from the disclosed system. Trefethen provides example MultiMATLAB code and then shows the output produced when that code is executed in the MultiMATLAB environment. Ex.1006, 6. A POSITA would understand the example as a disclosure of how MultiMATLAB nodes are configured, through MATLAB/MultiMATLAB commands, to pass values from one process to the next, perform mathematical expression evaluations, and return the resulting output to the interactive process.

16. Patent Owner's shorthand "1→2→3→1" describes the same sequence shown in the Trefethen example: process 0 sends a value to process 1; process 1 evaluates an expression using that value and sends the result to process 2; process 2 evaluates an expression using that result; and the resulting output from process 2 is transmitted to the master/interactive process and displayed. Ex.1006, 4, 6; Ex.1003, ¶¶211-228.

2. **"Master-controlled dispatch model" is not a distinction**

17. I understand that Patent Owner characterizes Menon/Trefethen's architecture as a "master-controlled dispatch model" in which "a user interacts with an 'interactive process'" (i.e., the first node) "and **other processes**" (i.e., the second and third nodes) **wait for commands from that interactive process.**" POR, 22. In my opinion this characterization does not distinguish the teachings of Menon/Trefethen from the claims.

18. First, Patent Owner's characterization of the MultiMATLAB interactive process does not identify a technical difference from the Challenged Claims. Claim 1 recites a node that receives user instructions while the other nodes wait: "second program code configured to interpret user instructions, identify an array of data elements, and distribute calls to at least one of a plurality of other nodes for execution." Ex.1001, 29:65-30:2. The claim language does not exclude worker nodes that wait for instructions or calls before executing assigned work.

19. In my opinion, Patent Owner’s use of the phrase “master-controlled” does not identify any technical distinction from the claim language. The claims require the first node to interpret user instructions and distribute calls to other nodes for execution, and they later require results to be communicated back to the first node/user interface. Ex.1001, 29:47-30:3; 34:33-49. That is consistent with an interactive process that dispatches work to other processes while also receiving and presenting returned output.

20. Second, I understand that Patent Owner argues that Trefethen’s script “passes a value from process to process using hardcoded send/receive instructions.” POR, 22. In my opinion, this acknowledges that Menon/Trefethen teaches passing a value from process to process in accordance with instructions. As discussed above, Trefethen’s script describes:

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; Ex.1003, ¶¶124-125.

21. Correspondingly, the claims recite that the second node (Trefethen’s “process 1”) receives a call from the first node (“*Process 0* creates the variable *a* with value 1 and *sends it to process 1*”), performs a mathematical operation

(process 1 “doubles the value of a”) and communicate the result to the third node (process 1 “sends it along to process 2”), which is configured to receive the call from the second node and then performs a mathematical operation on the result received from the second node (“and so on”), resulting a value of “a = 4.” Ex.1006, 6.

22. Nothing in the claim language requires that the second and third nodes originate the entire computation themselves. The claim language is satisfied when the participating nodes are configured by program code and accepted user instructions to perform the claimed receive/evaluate/communicate sequence.

3. The claims do not exclude “user-authored application code”

23. I understand that Patent Owner describes the “1→2→3→1” order of operations demonstrated by Menon/Trefethen as “simply an example of user-authored application code that passes a value from process to process using hardcoded send/receive instructions.” POR, 22. In my opinion, that characterization does not create a technical distinction. The claims recite user instructions accepted “via the user interface or via a script,” and the specification describes kernels interpreting “code” and “instructions” supplied by a “user or a **script**.” Ex.1001, 23:1-5, 30:21-24 (claim 1).

24. Menon/Trefethen describe MATLAB interpreting code and instructions supplied by user commands and scripts. Ex.1005, 13 (“this MATLAB

code”); Ex.1006, 4-5 (examples of user commands), 5-6 (example “m-file”); *see also* Ex.1016, 5 (m-files are “scripts”). For example, Menon describes that “[u]sers are given the tools to write parallel code purely in the MATLAB language.”

Ex.1005, 2. And Trefethen describes that “MATLAB code is translated to C code with parallel library routines.” Ex.1006, 11. Trefethen further describes the MATLAB languages “exceptionally user-friendly, intuitive syntax” and “easy extensibility via packages of m-files,” which are programs in the MATLAB language “conventionally have the extension ‘.m.’” Ex.1006, 2. A POSITA understand that the “m-file” examples in Trefethen are scripts that contain user instructions. Ex.1016, 5 (m-files are “scripts”).

25. Patent Owner’s argument also contradicts the specification. The specification describes that the kernel interprets “code” and “instructions” supplied by a “user or a script.” Ex.1001, 23: 1-5. Each of Menon and Trefethen describe MATLAB interpreting code and instructions supplied by user commands and scripts. Ex.1005, 13 (“this MATLAB code”); Ex.1006, 4-5 (examples of user commands), 5-6 (example “m-file”). A POSITA understand that the “m-file” examples in Trefethen are scripts that contain user instructions. Ex.1016, 5 (m-files are “scripts”).

26. Patent Owner’s “hardcoded” label also does not change the technical analysis. The claims are directed to nodes configured by program code and user

instructions, not to whether the source text was generated automatically or written by a user. Trefethen's script therefore discloses a concrete configuration of the MultiMATLAB processes.

27. A POSITA would understand that software instructions can configure hardware nodes to perform claimed operations. The claims themselves recite kernels configured to interpret user instructions, distribute calls, execute mathematical expression evaluations, and communicate results. Ex.1001, 29:51-30:27. Thus, the fact that Trefethen shows the operations in example MultiMATLAB code reinforces rather than undermines the disclosure: the code is the mechanism by which the MultiMATLAB system configures the participating processes to execute the claimed sequence.

(a) **The Challenged Claims do not require distinct “mathematical expressions” to be performed at each node**

28. I understand that Patent Owner argues that the Challenged Claims require “two **distinct** mathematical expression evaluations executed at two different nodes,” and therefore Menon/Trefethen's teaching of the “doubling operation” executed by the second node on the value received from the first node (“a first mathematical expression evaluation”) and the “doubling operation” executed by the third node on the result received from the second node (“a second mathematical expression evaluation”) do not teach the claimed features. POR, 23-24.

29. In my opinion this is incorrect. Trefethen’s example teaches distinct evaluations performed by distinct processes on distinct input values. The numerical evaluation performed by process 1 uses the value received from process 0 (1×2), while the numerical evaluation performed by process 2 uses the result received from process 1 (2×2). *See* Ex.1006, 6. A POSITA would recognize that these are separate mathematical-expression evaluations, even if the same source-code text or same general operation is used at more than one node.

30. In addition, in my opinion, Patent Owner’s arguments also conflict with the claim language. The Challenged Claims recite one node “execut[ing] the matrix operations” and another node “execut[ing] at least a mathematical expression evaluation,” i.e., each node is performing *its own* “execution” of either a matrix operation or a mathematical expression. An execution by one node is distinct from an execution by another node, particularly where the nodes use different received values. *See, e.g.,* Ex.1001, 14:37-38. Trefethen’s example code describes each node performing its own execution of a mathematical expression. *See* Ex.1006, 6.

31. Additionally, the specification confirms that different nodes may perform the “same operations.” Ex.1001, 17:1-39. The specification is describing these “same operations” in the context of each cluster node module performing “Parallel Evaluation” using “routines.” A POSITA would recognize that “Parallel

Evaluation” using routines can involve distinct nodes each performing the same operation on different data or received values.

32. The same point applies to Patent Owner’s assertion that Trefethen shows only a “single programmed routine.” POR, 23-24. The claim language is directed to mathematical expression evaluations performed at different nodes using different input values; it does not require that each node execute different source-code text. In Trefethen’s example, the expression evaluated at process 1 uses the value received from process 0, while the expression evaluated at process 2 uses the result received from process 1. Ex.1006, 6. Those are distinct evaluations performed at distinct nodes in the claimed order.

33. I also understand that the Board’s Institution Decision described Trefethen’s sample code as illustrating node 0 sending data “a = 1” to node 1, node 1 performing a mathematical calculation and sending the result to node 2, node 2 performing a mathematical calculation and sending the result to the next node, and the final result being output to the user. Paper 16, 14-15. That understanding is consistent with my technical opinion.

**(b) The Challenged Claims do not have a distinction
between “screen output” and “data”**

34. The Challenged Claims recite that “the third node is configured to ... communicate the result of the mathematical expression evaluation to the first node”

(limitation [1.5.2]) and “the first node is configured to return the result of the mathematical expression evaluation to the user interface” (limitation [1.6]).

35. In Menon/Trefethen, a result from the third node (i.e., “a=4”) is communicated to the first node and displayed to the user in a user interface of the first node. Ex.1003, ¶¶211-221.

Result from third node
displayed in the user
interface of the first node

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

Ex.1006, 6 (annotated); Ex.1003, ¶214

36. I understand that Patent Owner argues that the Challenged Claims require a “mechanism by which the value computed on a remote node is returned to the originating process as **data for further use.**” POR, 11. First, a POSITA would recognize that Menon/Trefethen teaches that the originating process (the interactive process) receives the result and then displays it. Ex.1006, 4, 6; Ex.1003, ¶220-221. In other words, the first node *does* put the result to “further use” after receiving the result: the first node displays the result on the screen for the user to review.

37. Second, in my opinion, the claims do not recite any further use requirement after the result is returned to the user interface. Other portions of the

claim recite use of a received value (e.g., the third node performs a mathematical evaluation using the value received from the second node), yet the claims contain no such language requiring any additional further use after the result is received by the first node and returned to the user interface.

38. I understand that Patent Owner also attempts to distinguish Menon/Trefethen by arguing that the display of that result by the first node in Menon/Trefethen is merely “forwarded...screen output,” and not “data.” POR, 1-18. But the Challenged Claims (e.g., limitations [1.5.2] and [1.6]) recite no requirement that the “*result of the mathematical expression evaluation*” communicated by the third node to the first node must be stored in the first node’s workspace as a reusable variable or be returned through any particular API return value mechanism. Ex.1001, claim 1. Nor does the claim language require any particular “mechanism” for the communication of the claimed result that would exclude communicating screen output that includes the computed result from the third node to the first node, as described in Menon/Trefethen. Ex.1001, claim 1.

39. There is also no technical basis for Patent Owner’s view that a numerical value ceases to be data merely because it is displayed. The displayed value “a = 4” is the numerical result of the computation performed by the third node. Trefethen also explains that MultiMATLAB uses “Eval, Send, [and] Recv”

to “pass commands and data between [] these various [MATLAB] processes.”

Ex.1006, Abstract.

40. Patent Owner’s “screen output” distinction is also inconsistent with the claim phrase “return the result ... to the user interface.” A user interface is precisely the part of the system that presents information to the user. When MultiMATLAB sends the computed output “a = 4” from process 2 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.

41. 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 node evaluates a second mathematical expression “using the received result” received from the second node. Ex.1001, 30:11-16. The absence of comparable “using” language after the result is returned to the user interface confirms that Patent Owner is importing an extra requirement not present in the claims.

42. Patent Owner’s focus on the behavior of a particular MultiMATLAB command, Eval(), also does not change my opinion. The relied-upon Trefethen example includes Send and Recv commands for point-to-point data movement, and the claim language is directed to the nodes being configured to communicate

results—not to whether a particular software command returns a value in the same manner as a function call in a programming language.

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

1. Menon and Trefethen teach how to implement MultiMATLAB

43. In my prior declaration (Ex.1003, ¶¶75-83), I describe multiple reasons why a POSITA would have been motivated to combine Menon and Trefethen, including:

- Menon suggesting the combination by **expressly citing** Trefethen and explaining that Menon builds on the teachings of Trefethen. Ex.1005, 1-2; Ex.1003, ¶81.

- 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, ¶80.

- 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, ¶80.

44. I understand that Patent Owner incorrectly reduces the reasons to combine Menon and Trefethen to “researchers had a strong desire to parallelize MATLAB” (which it labels as “Petitioner’s motivation”). POR, 28 (quoting Petition, 23). In fact, my prior testimony provides discusses this as part of why the combination would have produced predictable results and why there would have been a reasonable expectation of success, not as the only reason to combine. *See* Ex.1003, ¶83 (discussing predictable results and reasonable expectation of success), ¶¶80-82 (discussing reasons to combine).

45. 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.

46. 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.

The later architectural improvements would not cause a POSITA to discard the earlier user-level examples; instead, they would provide additional implementation context for them.

2. Menon does not teach away from the combination

47. The Menon and Trefethen references share two of the same authors—“Vijay Menon” and “Anne E. Trefethen” are the first two authors listed on both papers. Ex.1005, 1; Ex.1006, 1. Both references describe aspects of the same system—“MultiMATLAB.” Ex.1005, 1 (titled “MultiMATLAB: Integrating MATLAB with High-Performance Parallel Computing”); Ex.1006, 1 (titled “MultiMATLAB: MATLAB on Multiple Processors”). Menon explicitly cites Trefethen and explains that Menon builds on the teachings of Trefethen. Ex.1005, 1-2; Ex.1003, ¶80.

48. 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, 28-31. This is incorrect.

49. Menon does not criticize, discredit, or discourage use of the “MultiMATLAB” teachings in Trefethen. In fact, Menon describes *enhancing* the system of Trefethen. 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,” which Menon achieves by its description of adding “MEX routines” to the system of Trefethen. Ex.1005, 2-3. Far from showing an “incompatibility” between Menon and Trefethen as the POR argues, 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.

50. 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).

51. 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 user-level teachings such as Eval, Send, Recv, and the cycle.m example. 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.

52. 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 or technically wrong. 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.

53. A POSITA also would recognize that a redesign that improves performance or extensibility does not render earlier examples irrelevant. In software systems, it is common for a later implementation to preserve user-facing commands and programming examples while changing the internal implementation. That is how a POSITA would read Menon and Trefethen together.

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

54. I understand that Patent Owner also argues that Menon, Trefethen, RS6000, POEref, and MPIref were not publicly accessible. POR, 44-51. 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, ¶¶107-127.

55. 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

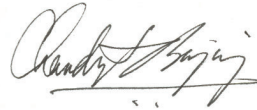
interested in high-performance parallel computing. Menon also expressly cites Trefethen's MultiMATLAB work and discusses the same MultiMATLAB system. Ex.1005, 1-2. A POSITA reading Menon would therefore have been led to Trefethen for command-level examples of Eval, Send, Recv, and cycle.m.

56. 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.