Συστήματα Παράλληλης Επεξεργασίας

**Άσκηση 4**

**4.1 Διάδοση θερμότητας σε δύο διαστάσεις**

|  |  |
| --- | --- |
| **Ομάδα** | parlab04 |
| Θεόδωρος Αράπης | el18028 |
| Εμμανουήλ Βλάσσης | el18086 |
| Παναγιώτης Παπαδέας | el18039 |

**Ζητούμενα**

Ως δεδομένα στην Άσκηση δόθηκαν οι σειριακές υλοποιήσεις των αλγορίθμων Jacobi, Gauss-Seidel-SOR και Red-Black-SOR. Ζητούνται:

1. Ο εντοπισμός του παραλληλισμού των αλγορίθμων και σχεδίαση της παράλληλης υλοποίησή τους σε αρχιτεκτονική κατανεμημένης μνήμης που υποστηρίζει ανταλλαγή μηνυμάτων.
2. Η παραλληλοποίησή τους με χρήση της βιβλιοθήκης MPI.
3. Μετρήσεις επίδοσης.
4. Σύγκριση αποτελεσμάτων και σχολιασμός.

**Σχεδιασμός Παραλληλοποίησης**

**Πίνακας U**

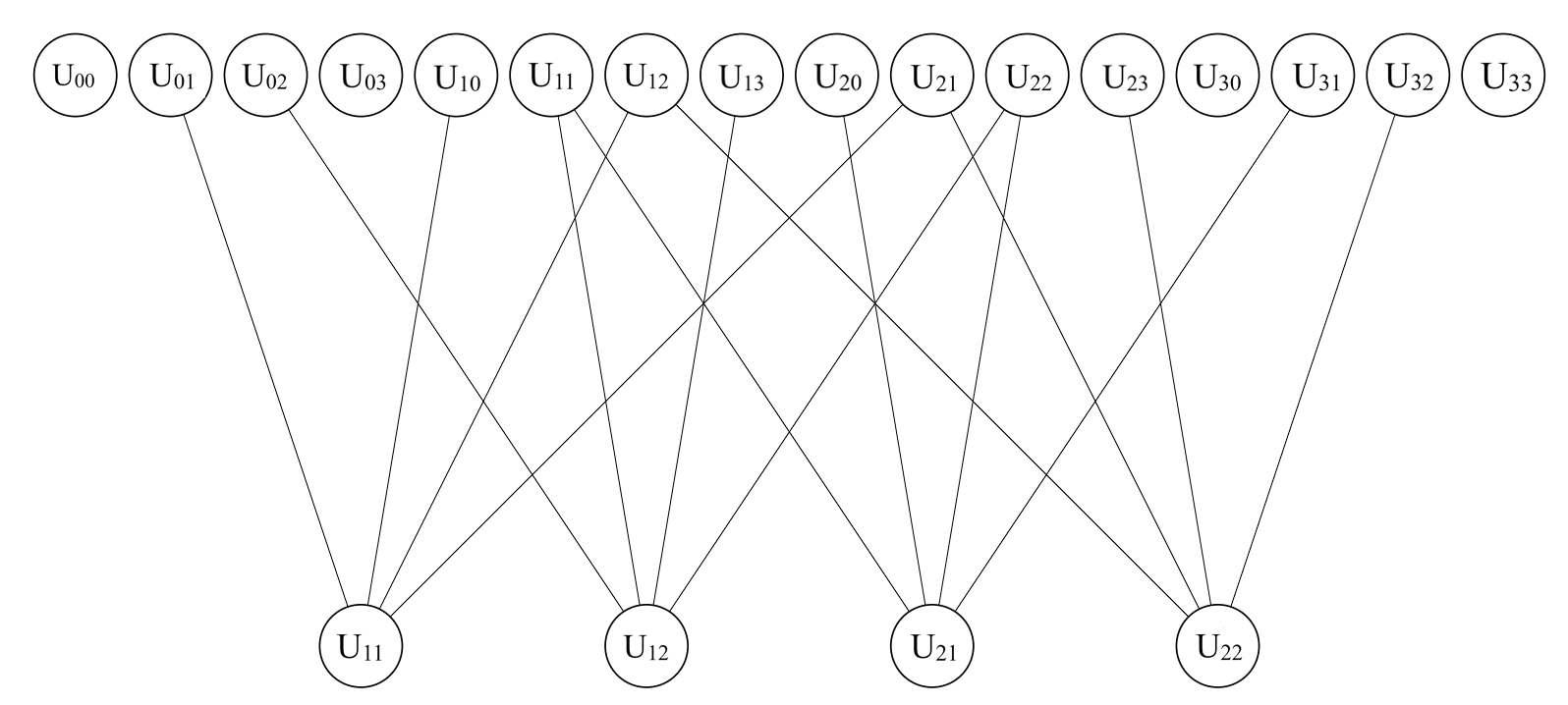
Ο πίνακας U έχει διαστάσεις . Θεωρώντας για ευκολία σχεδίασης ότι έχουμε (το οποίο μπορεί να επεκταθεί έτσι ώστε Uij = μπλοκ μεγέθους x, y):

|  |  |  |  |
| --- | --- | --- | --- |
| U00 | U01 | U02 | U03 |
| U10 | U11 | U12 | U13 |
| U20 | U21 | U22 | U23 |
| U30 | U31 | U32 | U33 |

U =

**Μέθοδος Jacobi**

Θα σχεδιάσουμε το task graph για τις δύο πρώτες χρονικές στιγμές. Χαρακτηρίζουμε ως task την θερμότητα σε κάθε θέση (ή μπλοκ θέσεων) για μία χρονική στιγμή:

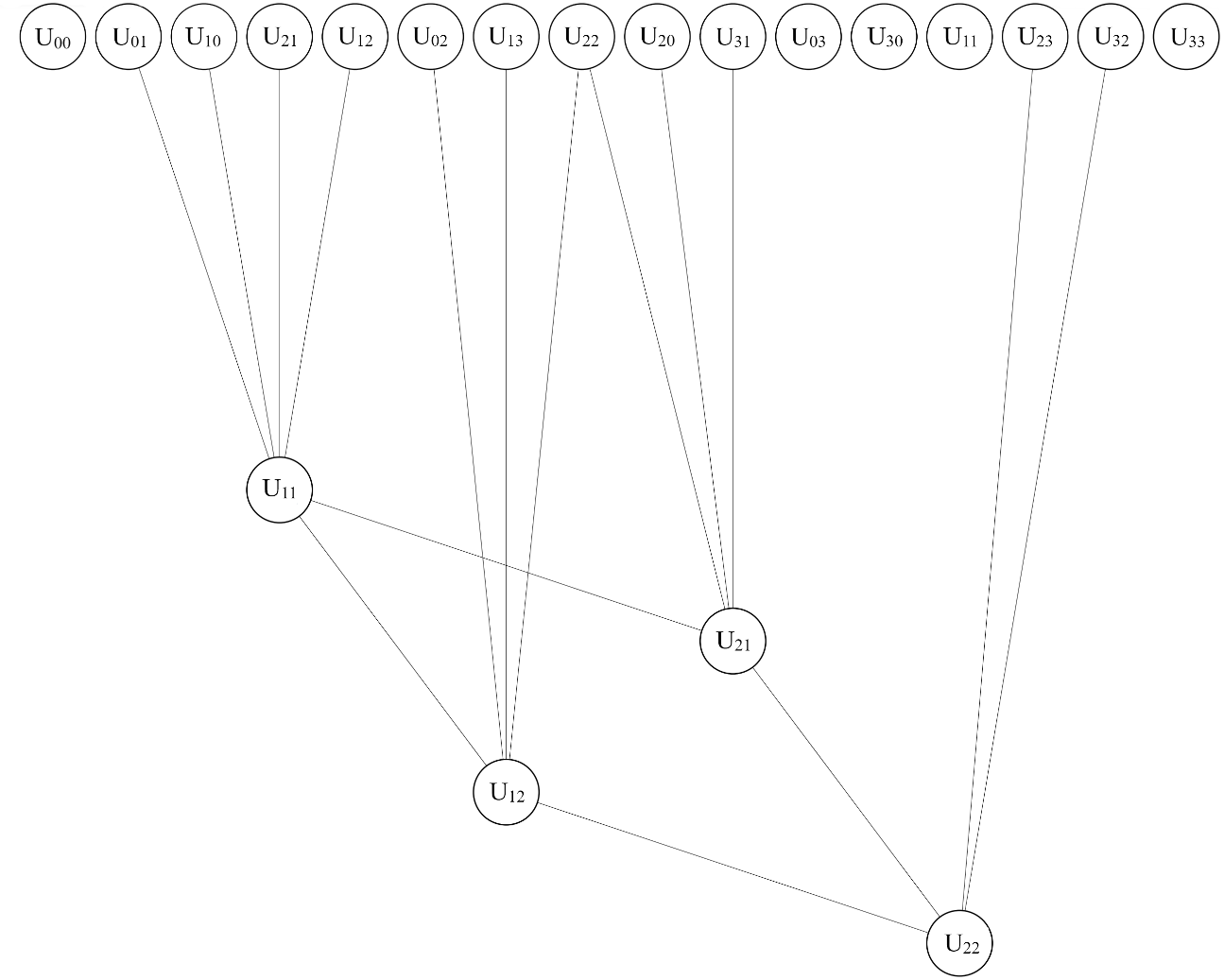
****

**t = 0**

**t = 1**

**Μέθοδος Gauss-Seidel SOR**

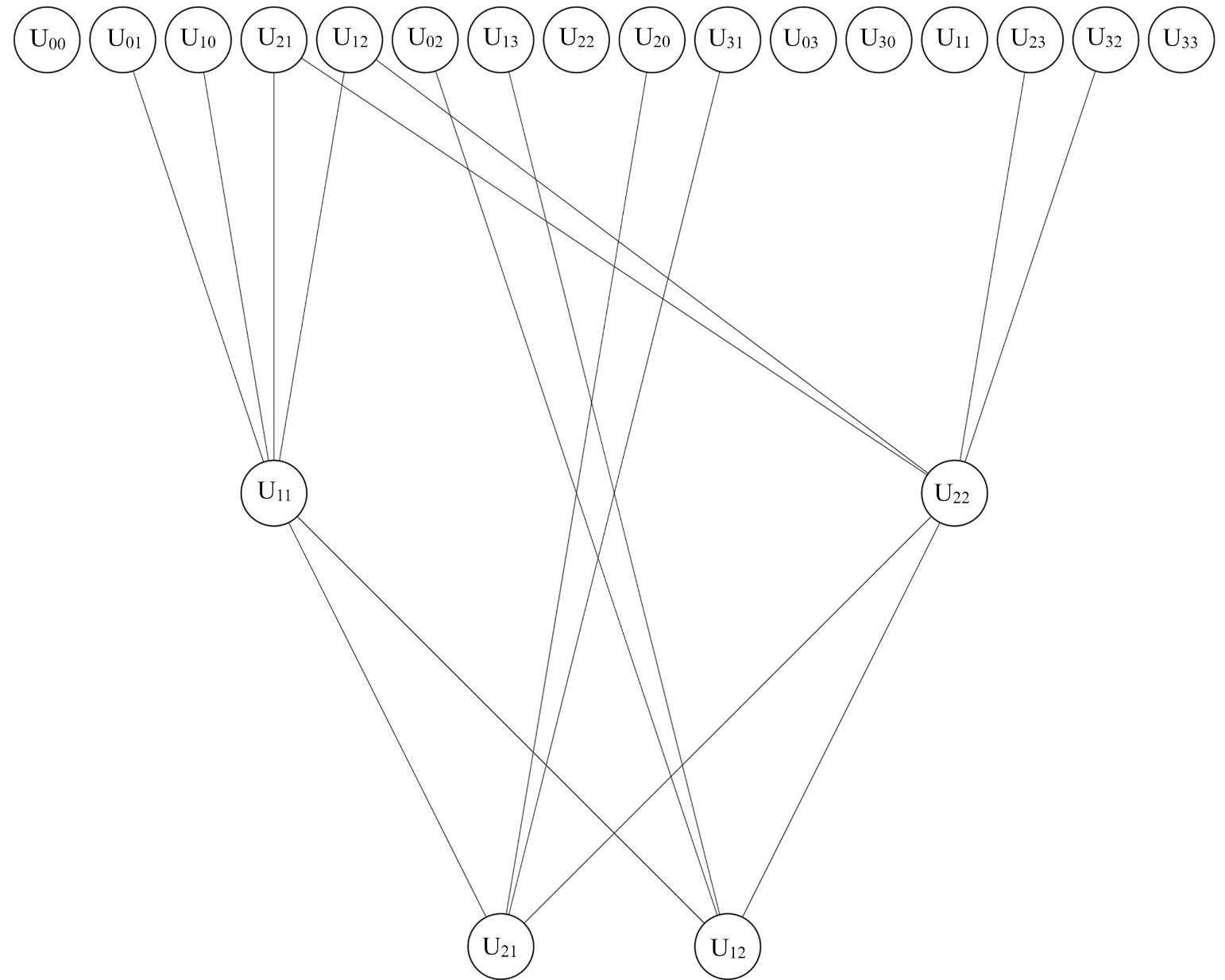
Όμοια και δω θα σχεδιάσουμε το task graph για τις δύο πρώτες χρονικές στιγμές. Θεωρούμε, όπως και πριν, ως task την θερμότητα σε κάθε θέση (ή μπλοκ θέσεων) για μία χρονική στιγμή:

****

**t = 0**

**t = 1**

**Μέθοδος Red-Black SOR**Τέλος, το task graph για τις δύο πρώτες χρονικές στιγμές για την μέθοδο Red-Black SOR, θεωρώντας πάλι ως task την θερμότητα σε κάθε θέση (ή μπλοκ θέσεων) για μία χρονική στιγμή, είναι:

****

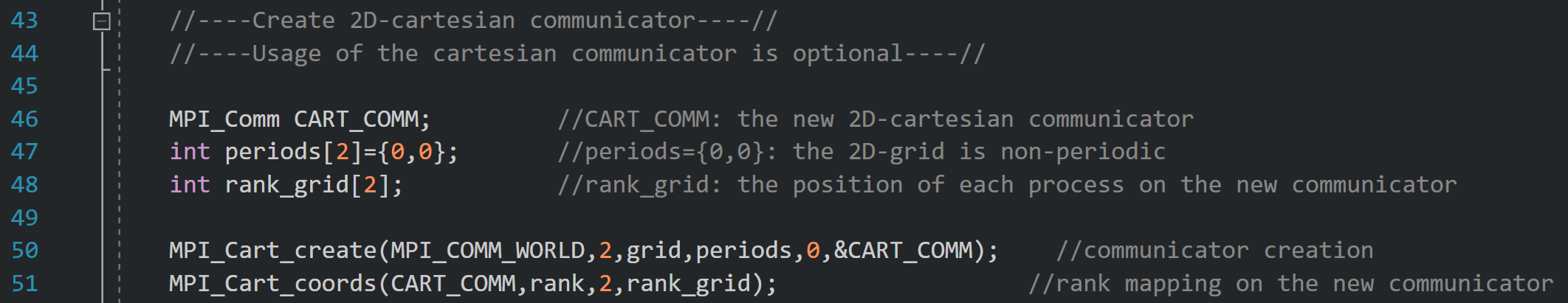
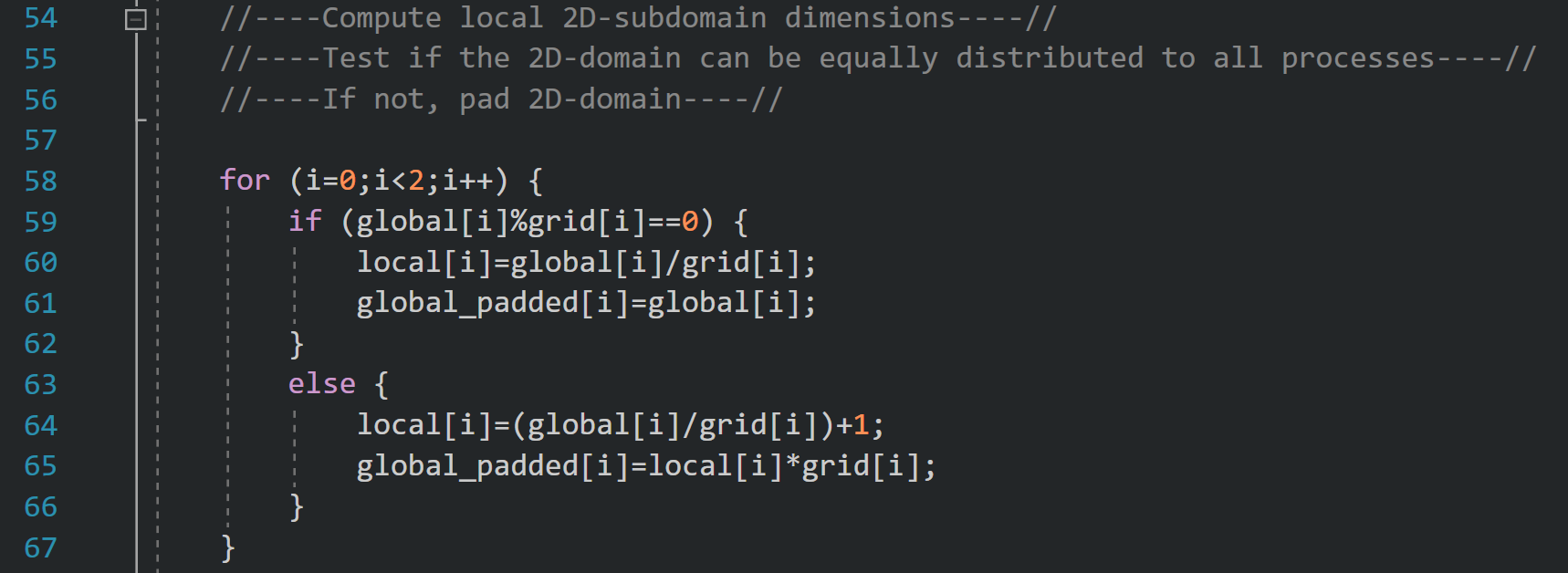
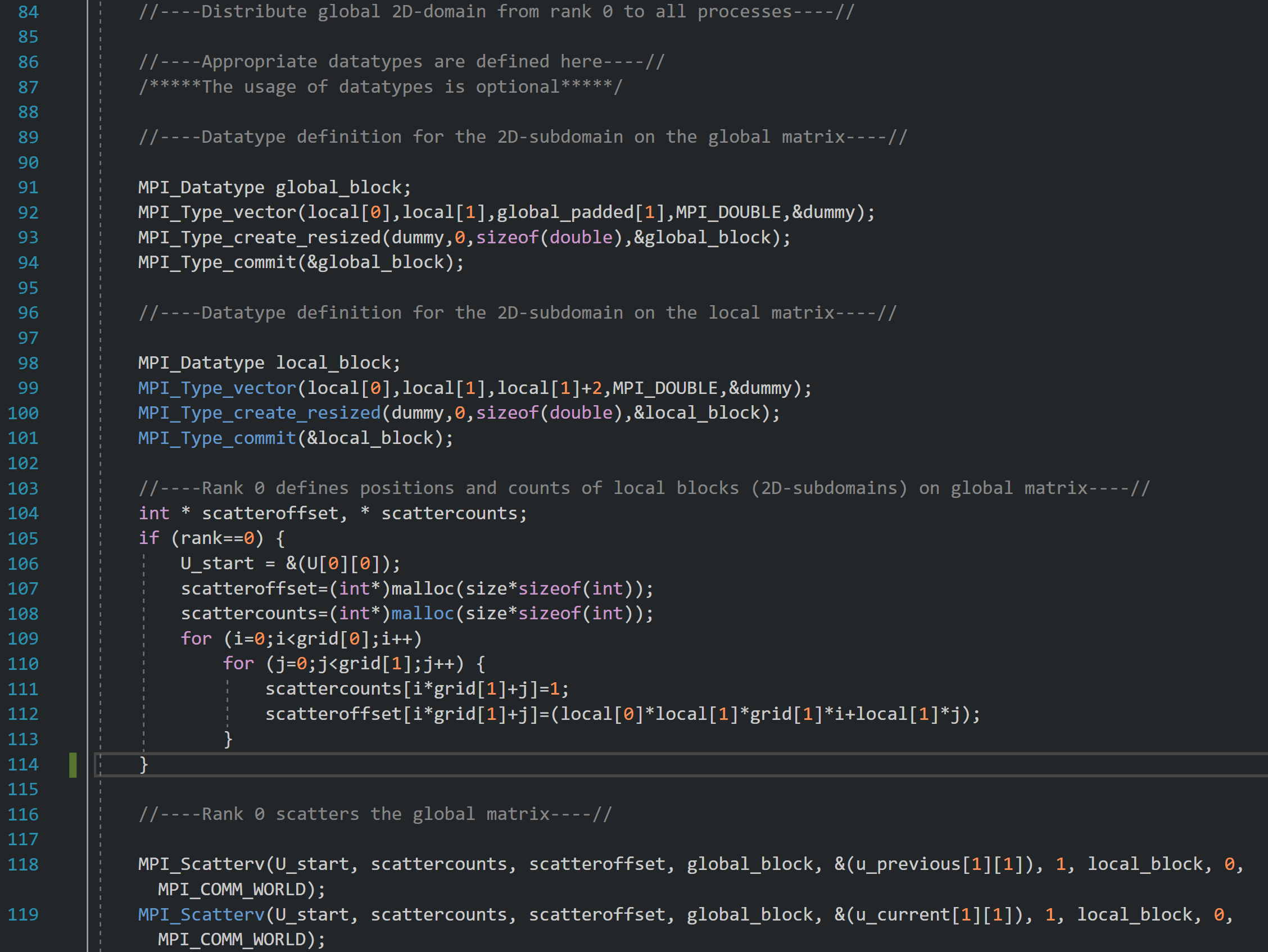
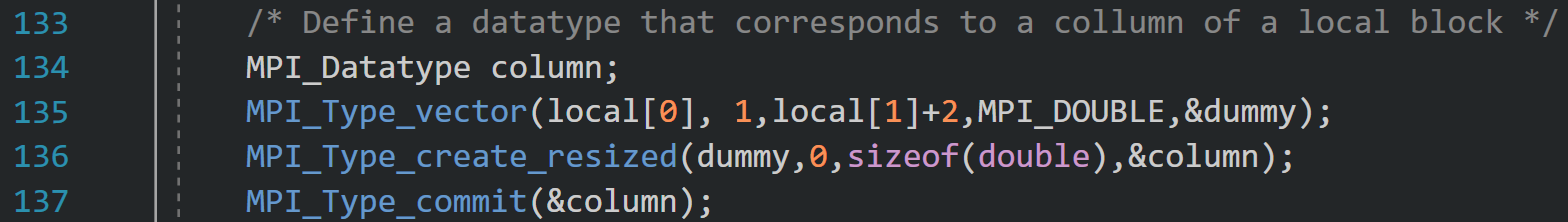
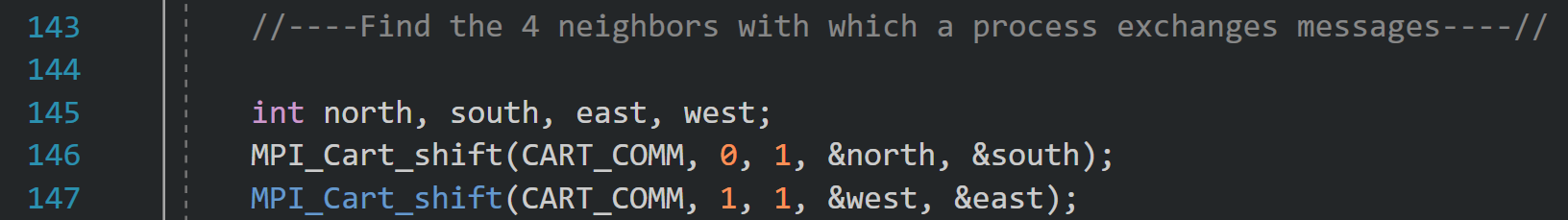
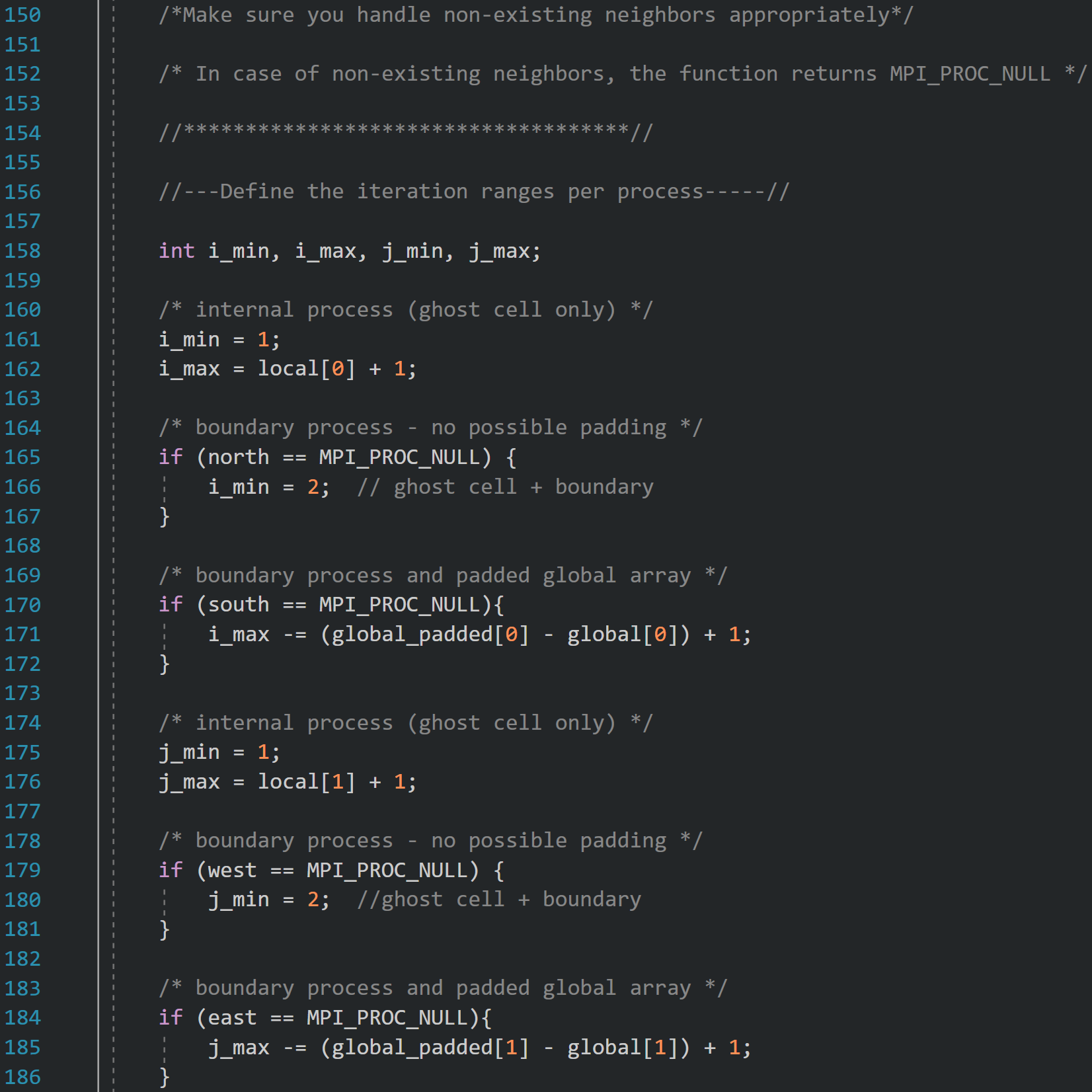
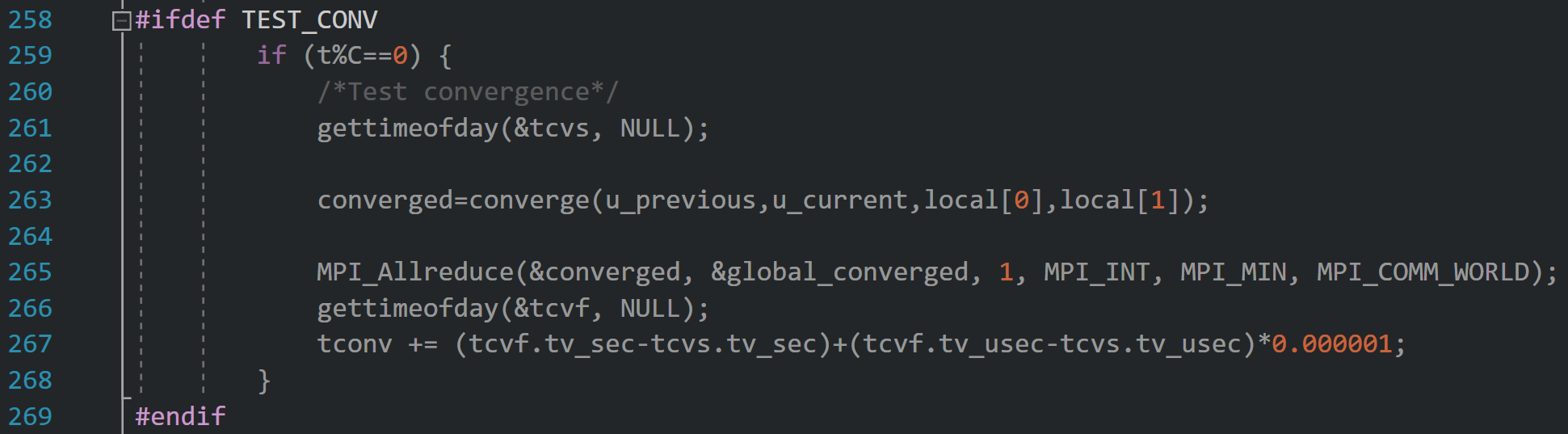
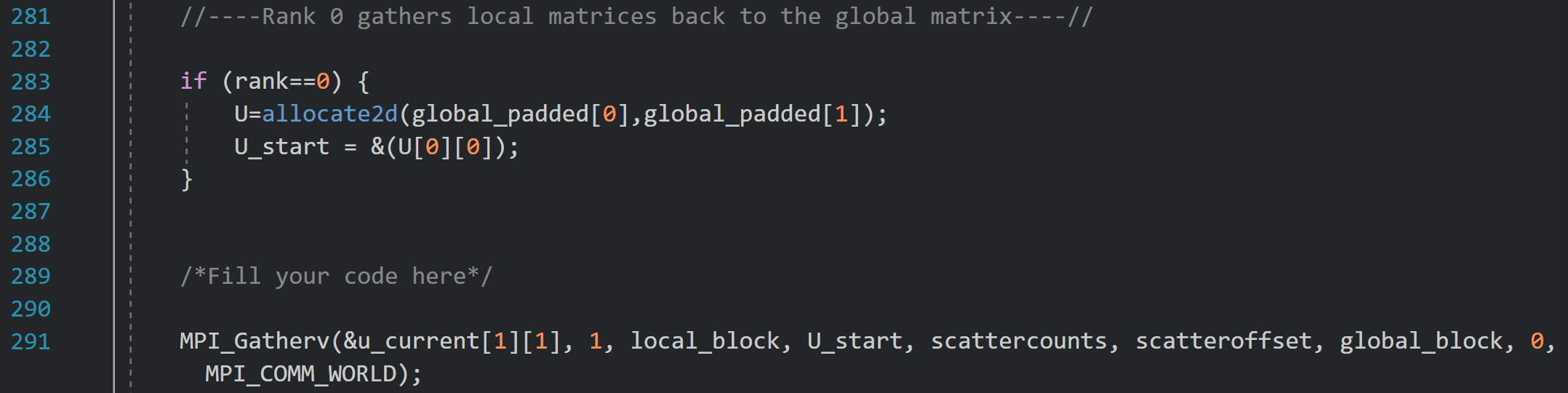
**t = 0**

**t = 1**

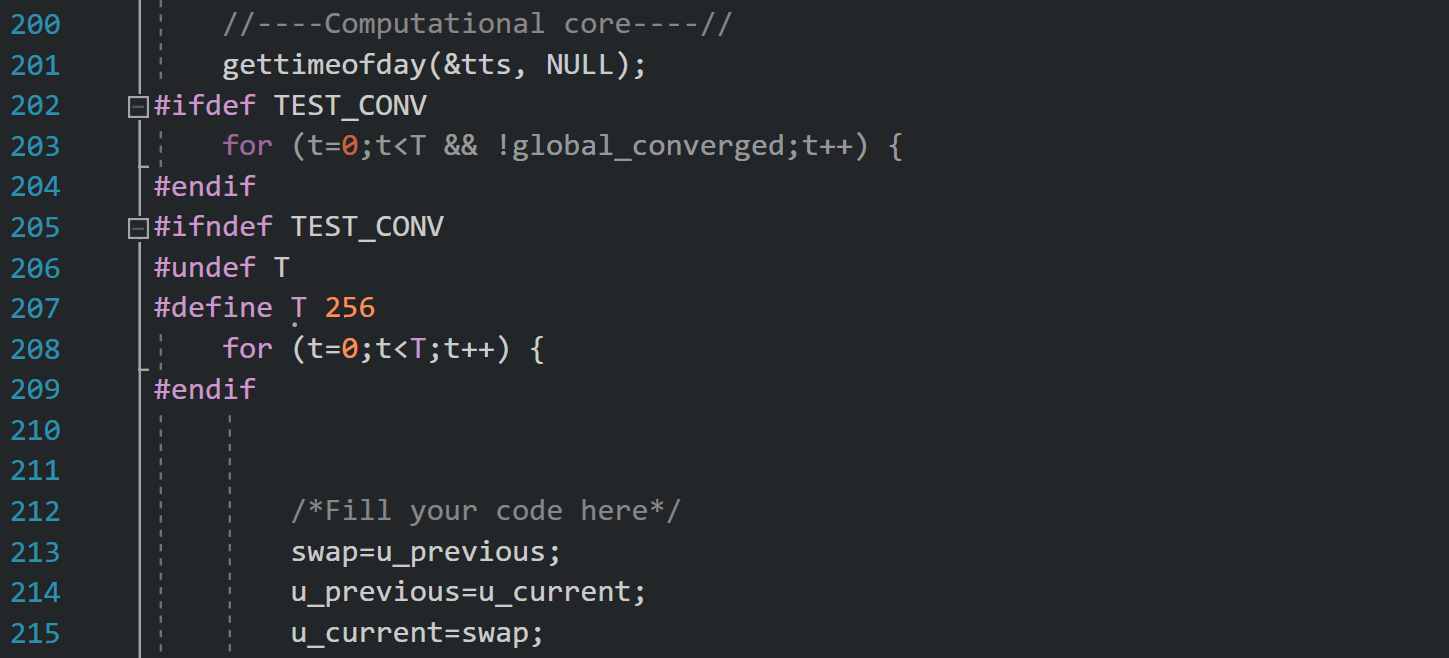
**Ανάπτυξη παράλληλων προγραμμάτων**

**Πίνακας U**

Και για τις 3 υλοποιήσεις, ο αρχικός πίνακας U χωρίζεται σε block, ένα για κάθε διεργασία. Σε κάθε επανάληψη, οι διεργασίες ανταλλάσσουν δεδομένα μεταξύ τους και ύστερα υπολογίζουν τις νέες τιμές. Όταν όλα τα block συγκλίνουν στις τελικές τιμές τους, τότε η μέθοδος συγκλίνει. Στο τέλος, οι τιμές των επιμέρους block συνενώνονται στον αρχικό πίνακα. Οι ακόλουθες διαδικασίες εφαρμόζονται και στις 3 υλοποιήσεις:

1. Δημιουργία 2D καρτεσιανού communicator.
2. Υπολογισμός διαστάσεων του επιμέρους πίνακα κάθε block (χρήση padding στον αρχικό πίνακα σε περίπτωση που ο διαμοιρασμός των block δεν χωράει ακριβώς).
3. Διαμοιρασμός του αρχικού global πίνακα U στον local πίνακα κάθε διεργασίας.
4. Ορισμός μιας δομής που αντιστοιχεί σε ένα local column, προκειμένου να διευκολύνει την μεταφορά μιας στήλης ενός πίνακα μεταξύ των διεργασιών.
5. Για κάθε διεργασία υπολογίζουμε το rank των γειτόνων της ώστε να μπορεί να επικοινωνήσει μαζί τους.
6. Ορισμός του range πάνω στο οποίο η διεργασία θα ανανεώνει τον local πίνακά της. Εδώ, πρέπει να λάβουμε υπόψιν τόσο τα ghost cell που έχουν τοποθετηθεί για την διευκόλυνση της επικοινωνίας όσο και τα πιθανά padding cells.
7. Έλεγχος σύγκλισης, (προϋπόθεση όλα τα local block να έχουν συγκλίνει).
8. Συνένωση των επιμέρους local πινάκων στον αρχικό global πίνακα U, τον οποίο χειρίζεται η διεργασία με rank = 0

***Jacobi***

Σε αυτήν την περίπτωση, κάθε διεργασία που επεξεργάζεται ένα block πρέπει να επικοινωνήσει πρώτα με της γειτονικές διεργασίες της και να μάθει τα σύνορα των γειτονικών blocks. Οι τιμές αυτές αποθηκεύονται στα ghost cells που αναφέρθηκαν προηγουμένως, με αποτέλεσμα να μην χρειάζεται να αλλάξει καθόλου ο βρόγχος ανανέωσης του σειριακού προγράμματος. Μετά το τέλος της επικοινωνίας, κάθε κελί θα έχει λάβει τις σωστές τιμές από τους γείτονές του.

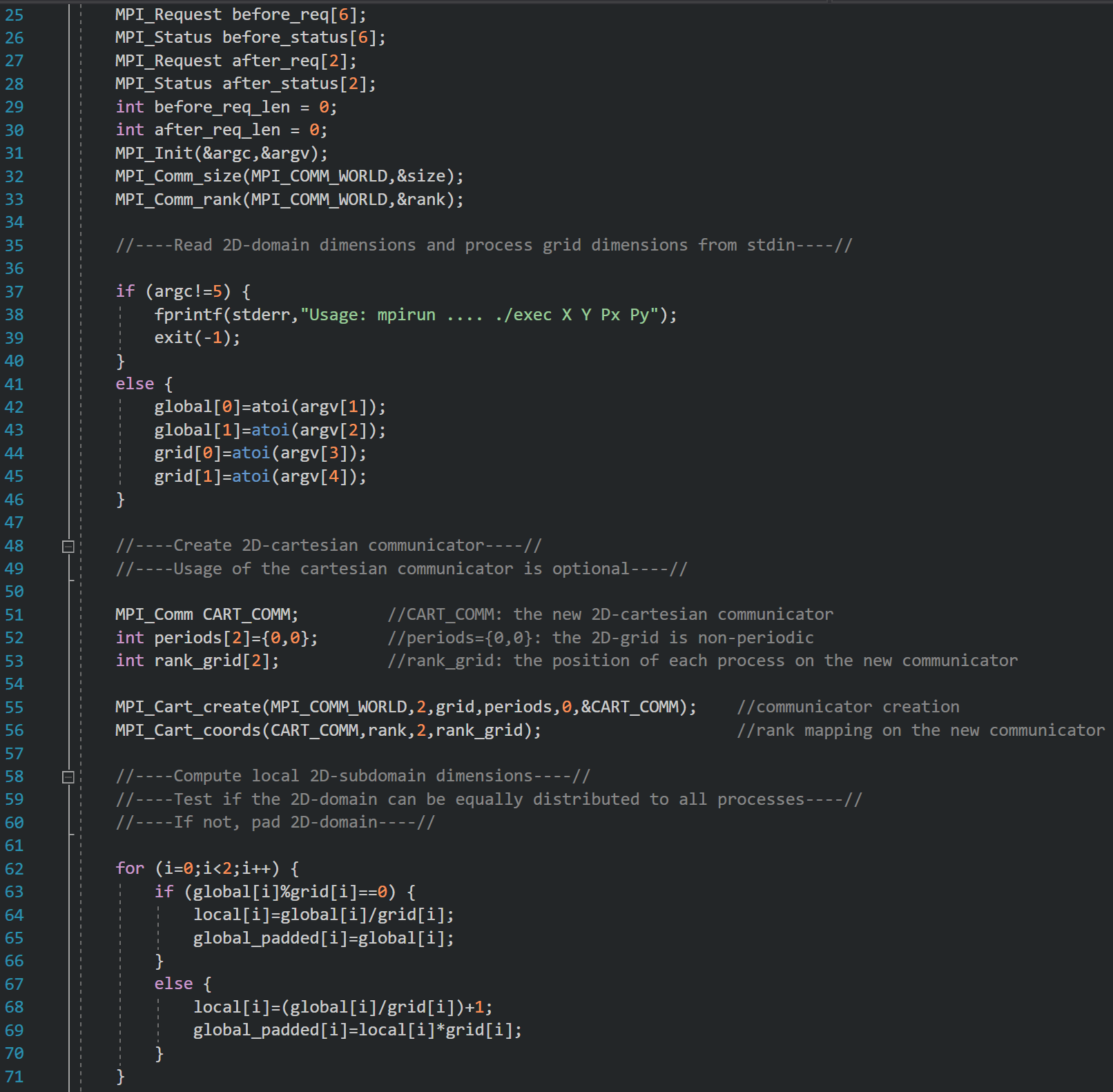


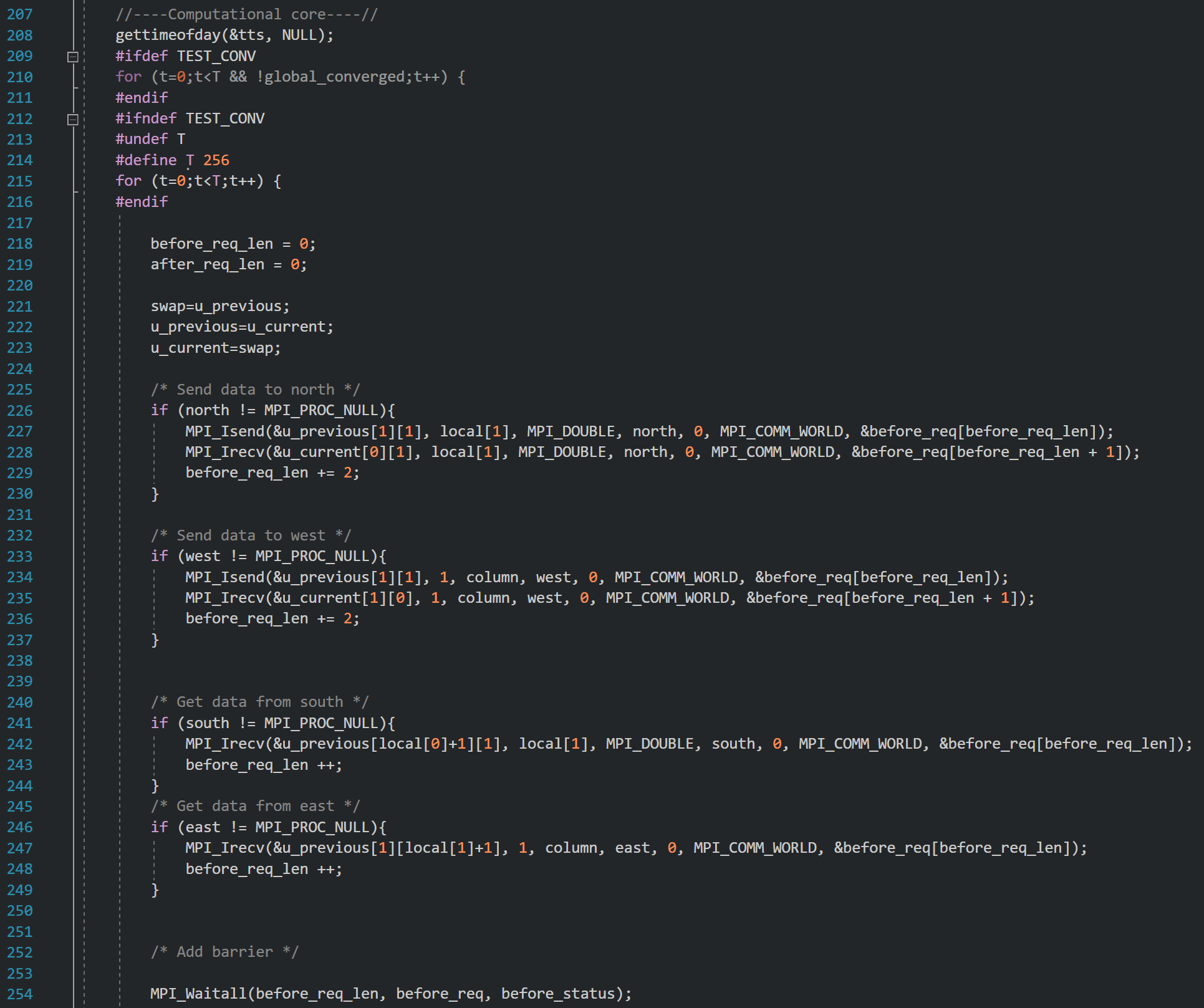
***Gauss-Siedel SOR***

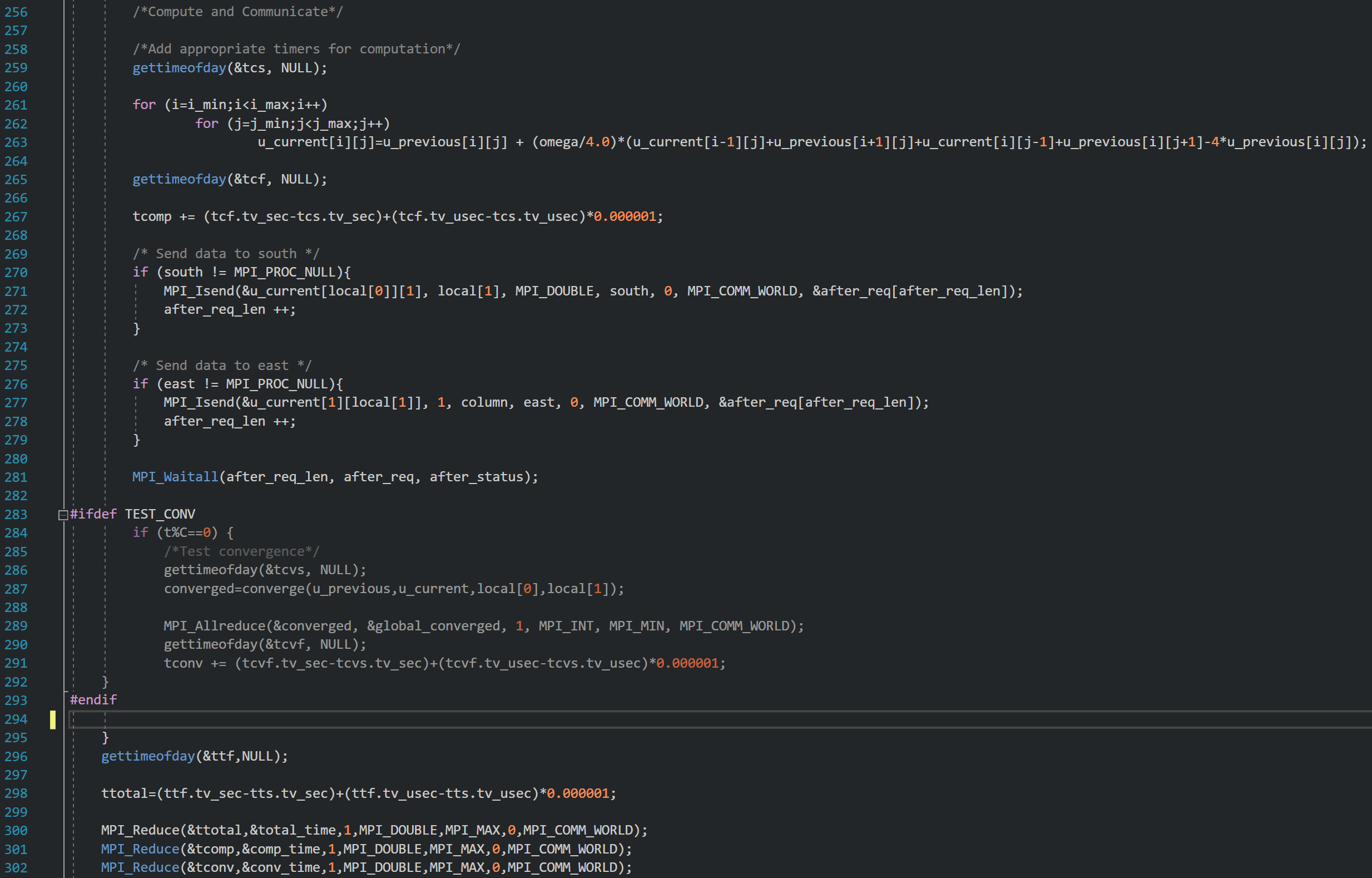
Στην μέθοδο αυτή, η υλοποίηση της επικοινωνίας (πρωτού γίνουν οι υπολογισμοί) γίνεται ως εξής για κάθε block:

Η κάθε διεργασία στέλνει στον πάνω και στον αριστερά γείτονα τα δεδομένα της προηγούμενης χρονικής στιγμής. Στη συνέχεια λαμβάνει από τον πάνω και τον αριστερά γείτονα τα δεδομένα της τωρινής χρονικής στιγμής. Επιπλέον, λαμβάνει από τον δεξιά και τον κάτω γείτονα τα δεδομένα της προηγούμενης χρονικής στιγμής. Τέλος, μετά τον υπολογισμό, στέλνει στον δεξιά και στον κάτω γείτονα τα δεδομένα της τωρινής χρονική στιγμής.

Πιο γενικά, όλες οι διεργασίες, εκτός από αυτές που βρίσκονται πάνω και αριστερά στο ταμπλό, θα μπλοκάρουν περιμένοντας τα τωρινά δεδομένα από τον πάνω και τον αριστερά γείτονά τους. Οι διεργασίες που δεν έχουν γείτονα πάνω και αριστερά θα υπολογίσουν τις νέες τιμές και θα τις στείλουν στον κάτω και δεξιά γείτονά τους. Η διαδικασία αυτή θα επαναληφθεί μέχρι όλα τα block να ανανεωθούν. Για τον λόγο αυτό, χρησιμοποιήσαμε non-blocking send και receive συναρτήσεις και θέσαμε στα σημεία όπου χρειαζόμαστε τις γειτονικές τιμές το κατάλληλο wait.

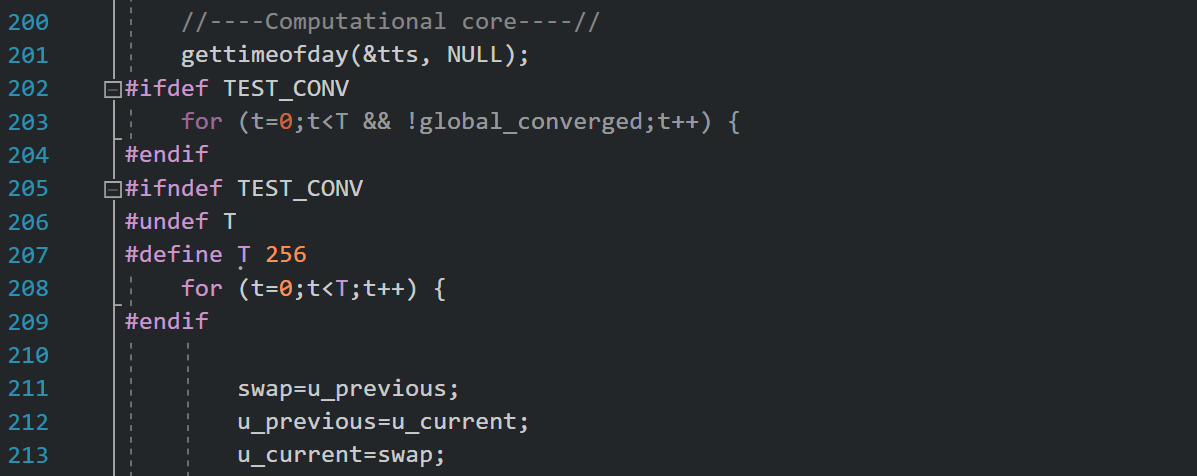


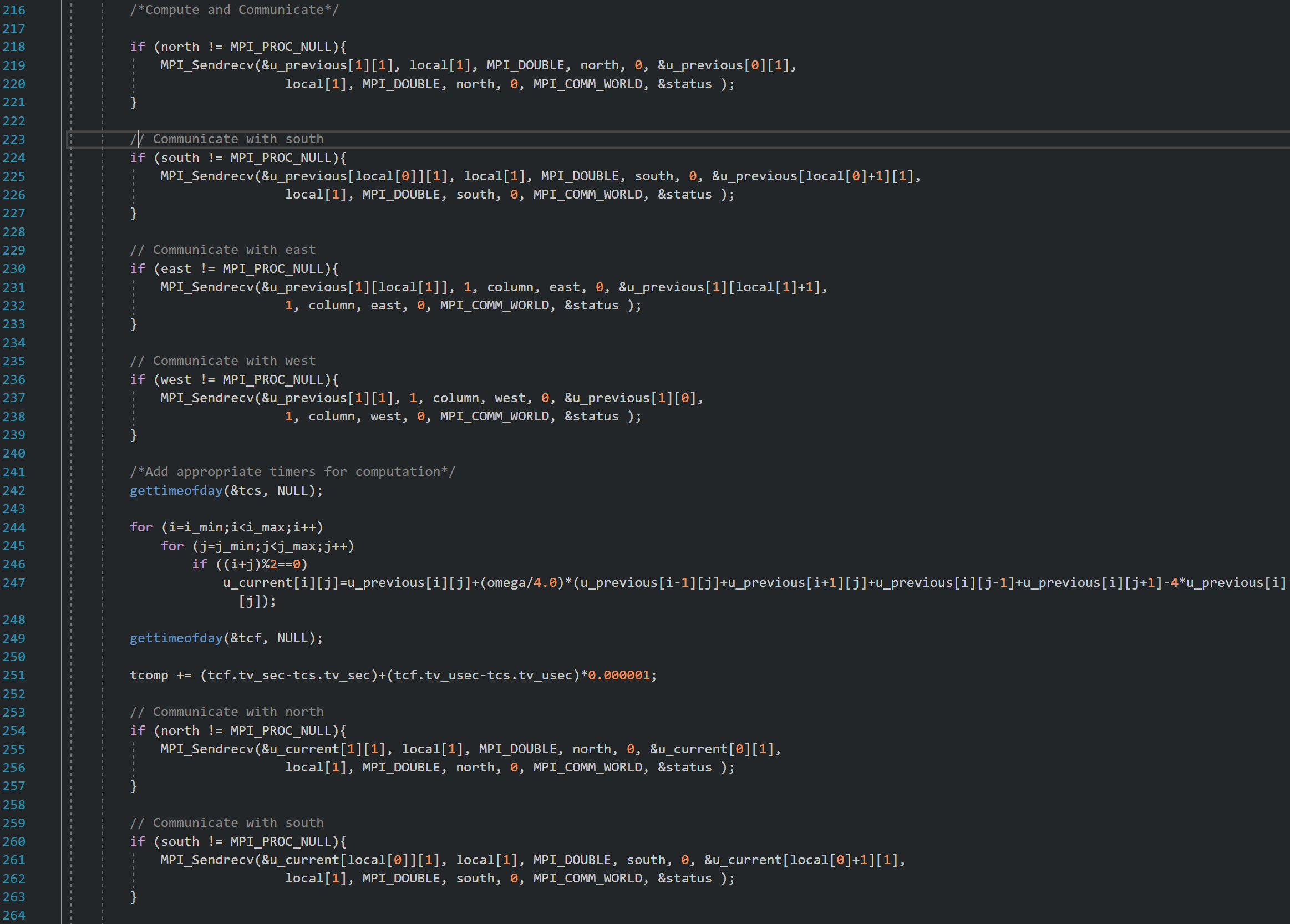
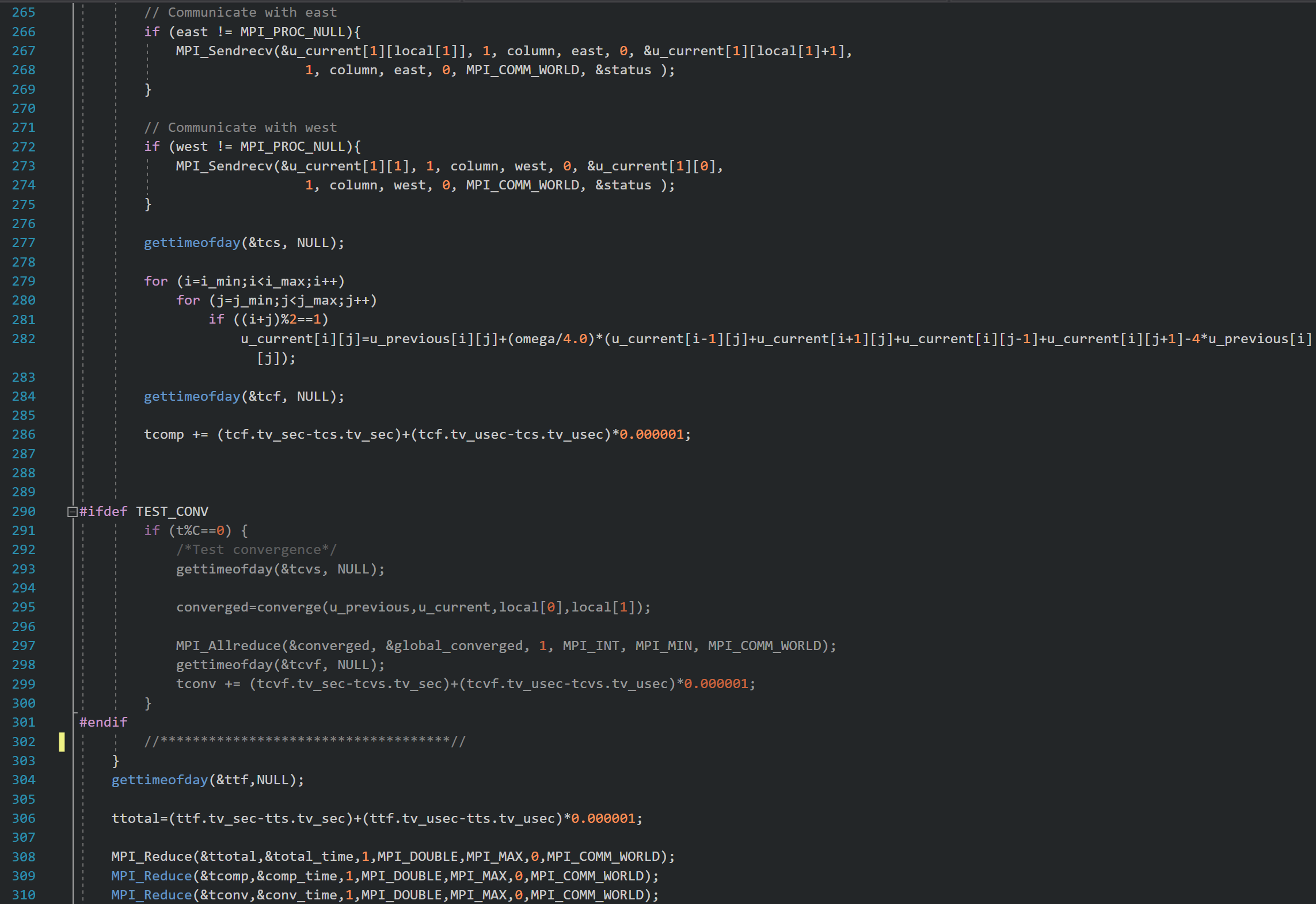
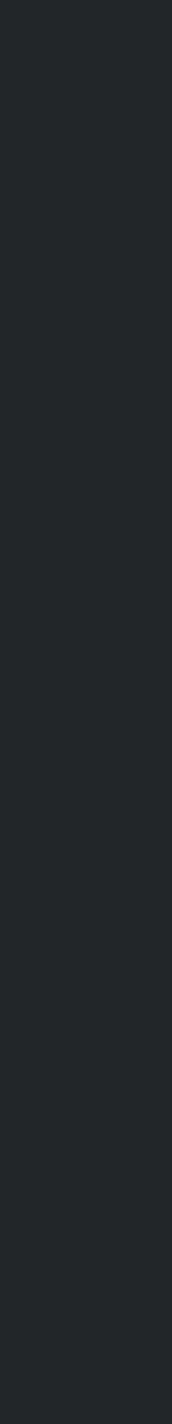
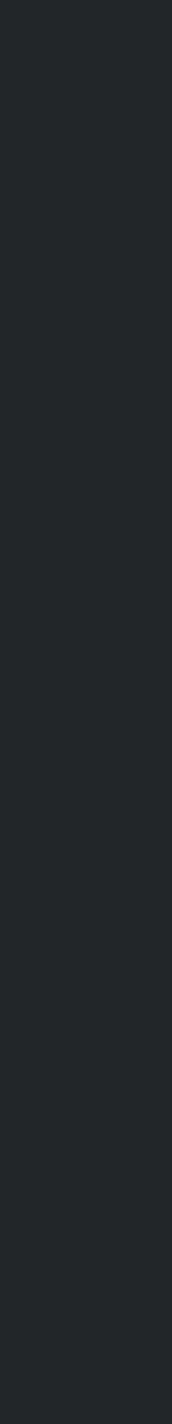




***Red-Black SOR***

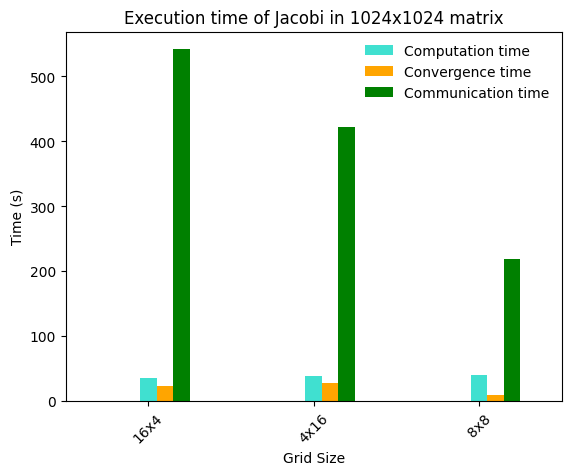
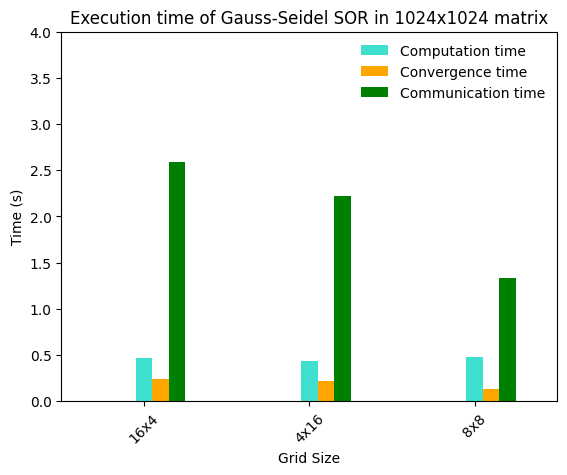
Στην περίπτωση της μεθόδου Red-Black SOR έχουμε δύο φάσεις υπολογισμού. Στην πρώτη φάση, τα κόκκινα στοιχεία πρέπει να γνωρίζουν τους γείτονές τους, ενώ στην δεύτερη φάση τα μαύρα στοιχεία πρέπει να κάνουν το ίδιο. Μιας και τα κόκκινα και τα μαύρα στοιχεία τοποθετούνται εναλλάξ στις άρτιες και περιττές θέσεις, και στις δύο φάσεις όλα τα block χρειάζεται να επικοινωνήσουν με τους γείτονές τους. Συνεπώς, η διαδικασία τώρα μοιάζει με αυτήν του Jacobi, με την διαφορά ότι τώρα θα γίνει 2 φορές η επικοινωνία, μία για να ενημερωθεί ο πίνακας u\_previous και να υπολογιστούν τα κόκκινα και μία για να ενημερωθεί ο πίνακας u\_current και να υπολογιστούν τα μαύρα στοιχεία. Ακολουθεί η υλοποίηση του κώδικα:

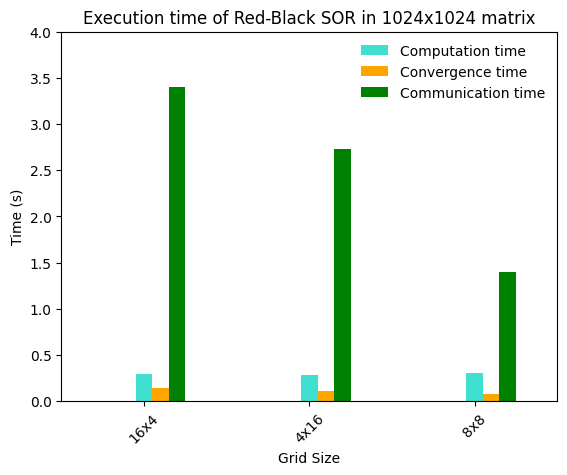




**Μετρήσεις επίδοσης**

**Μετρήσεις με έλεγχο σύγκλισης**

Στο κομμάτι αυτό, θα πραγματοποιήσαμε μετρήσεις με έλεγχο σύγκλισης για τα παράλληλα προγράμματά που υλοποιήσαμε σε MPI, για μέγεθος πίνακα 1024×1024. Γι’ αυτό το μέγεθος πίνακα, λάβαμε μετρήσεις (συνολικός χρόνος, χρόνος υπολογισμών, χρόνος ελέγχου σύγκλισης) για 64 MPI διεργασίες. Συνοψίζουμε τις μετρήσεις στα ακόλουθα διαγράμματα για κάθε μέθοδο:



Εύκολα παρατηρούμε ότι στις υλοποιήσεις που έχουμε grid size 8×8 η εκτέλεση είναι πιο γρήγορη. Αυτό συμβαίνει διότι τα μηνύματα που ανταλλάσσονται μεταξύ των διεργασιών έχουν ίδιο μέγεθος, αφού είναι συμμετρικό το grid size, ενώ στις περιπτώσεις των 16×4 και 4×16, η εκτέλεση είναι πιο αργή, καθώς η αποστολή μηνυμάτων στην πλευρά ενός μπλοκ προκαλεί σημαντικές καθυστερήσεις.

Για κάθε μέθοδο συγκεκριμένα έχουμε τις εξής παρατηρήσεις:

***Jacobi***

Η μέθοδος Jacobi έχει ως μειονέκτημα ότι απαιτεί πολλές επαναλήψεις του αλγόριθμου προκειμένου να συγκλίνει, γεγονός που την καθιστά την πιο αργή από τις τρεις μεθόδους. Ως εκ τούτου, ο ρυθμός σύγκλισής του, ο χρόνος υπολογισμού και ο χρόνος επικοινωνίας του είναι αρκετά μεγαλύτεροι σε σύγκριση με τις άλλες δύο μεθόδους.

***Gauss-Siedel SOR & Red-Black SOR***

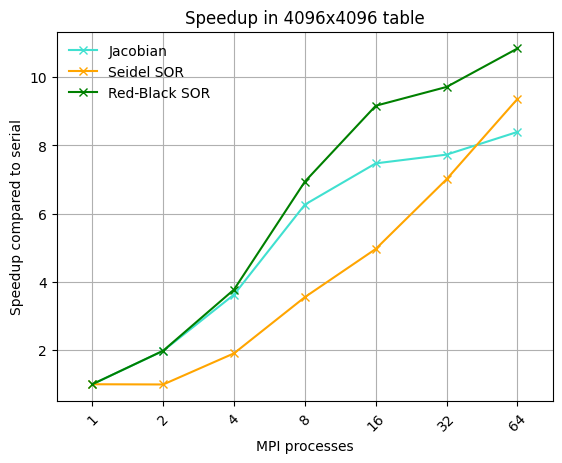
Οι δυο αυτές μέθοδοι μοιάζουν αρκετά στην υλοποίηση. Η μόνη διαφορά τους έγκειται στον χώρο ο σύγκλισής και στον χρόνο επανάληψης. Πιο ειδικά, ο Gauss-Siedel SOR πραγματοποιεί γρηγορότερος υπολογισμούς και απαιτεί μικρότερο χρόνο για επικοινωνία. Από την άλλη, η Red-Black SOR συγκλίνει αρκετά πιο γρήγορα, μιας και απαιτεί λιγότερες επαναλήψεις. Συνολικά παρατηρούμε ότι η μέθοδος Red-Black SOR φαίνεται να υπερισχύει, πράγμα που την καθιστά την γρηγορότερη από τις 3 μεθόδους.

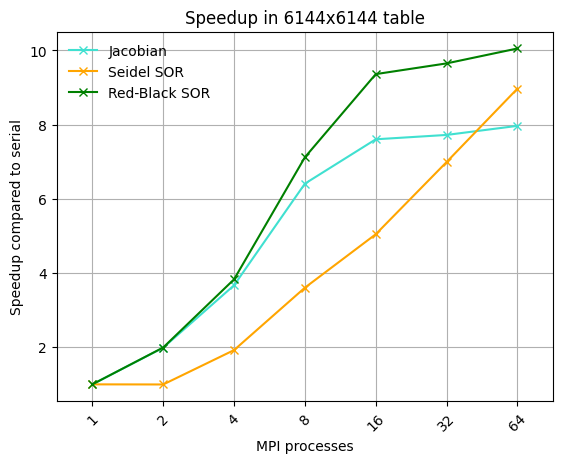
Τέλος, αξίζει να σημειωθεί ότι σημαντικός παράγοντας είναι και το δίκτυο διασύνδεσης. Ειδικότερα, βλέπουμε ότι ο χρόνος επικοινωνίας επιφέρει τις μεγαλύτερες καθυστερήσεις σε κάθε εκτέλεση. Αν και στην συγκεκριμένη αρχιτεκτονική φαίνεται ότι η μέθοδος Red-Black SOR λειτουργεί αποδοτικότερα, δεν σημαίνει πως πάντα θα ισχύει αυτό. Αν το δίκτυο διασύνδεσης είχε περισσότερη κίνηση ενδέχεται η μέθοδος Gauss-Siedel SOR να ήταν πιο γρήγορη μιας και χρειάζεται λιγότερο χρόνο για επικοινωνία.

**Μετρήσεις χωρίς έλεγχο σύγκλισης**

Θα πραγματοποιήσαμε τώρα μετρήσεις για το συνολικό χρόνο και το χρόνο υπολογισμών, απενεργοποιώντας τον έλεγχο σύγκλισης, για σταθερό αριθμό επαναλήψεων (Τ=256), για μεγέθη πίνακα 2048×2048, 4096×4096, 6144×6144, για 1, 2, 4, 8, 16, 32 και 64 MPI διεργασίες.

* Σχεδιάζουμε ένα διάγραμμα επιτάχυνσης για κάθε μέγεθος πίνακα α (x-axis: MPI διεργασίες, y-axis: speedup) που απεικονίζει τις τρεις μεθόδους.

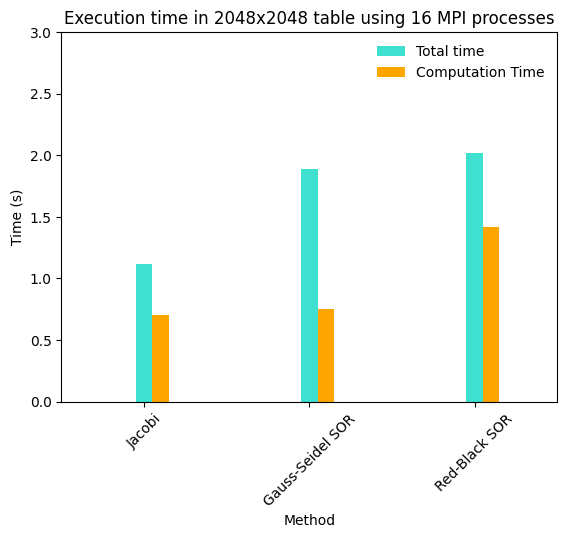
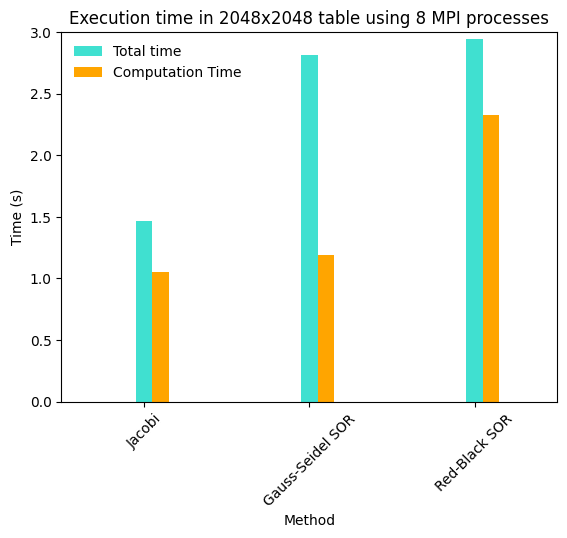


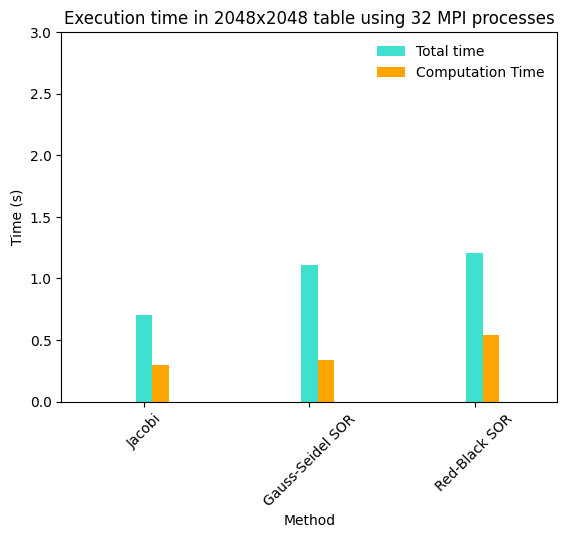


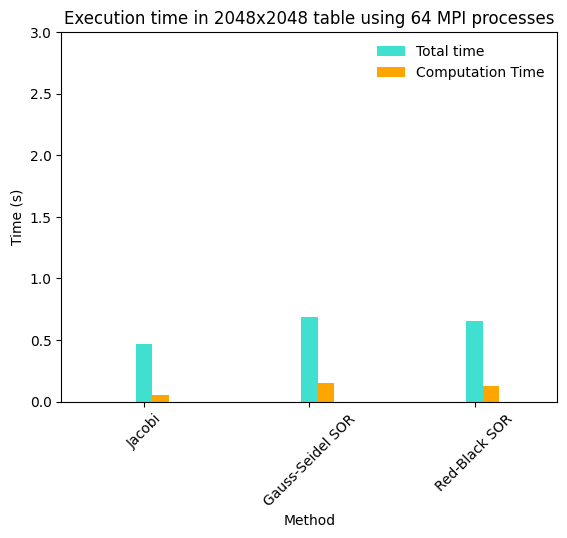
* Κατασκευάζουμε διαγράμματα με μπάρες (1 για κάθε μέγεθος πίνακα και αριθμό επεξεργαστών) που θα απεικονίζουν το συνολικό χρόνο εκτέλεσης και το χρόνο υπολογισμού για κάθε μία από τις τρεις μεθόδους (x-axis: μέθοδος, y-axis: χρόνος).

Για λόγους καλύτερης εποπτείας, κατασκευάζουμε διαγράμματα μόνο για 8, 16, 32 και 64 MPI διεργασίες και κρατάμε κοινή κλίμακα στον άξονα y ανά μέγεθος πίνακα.

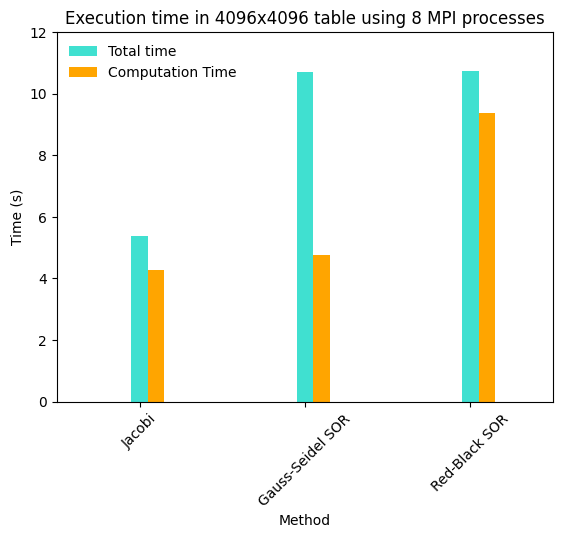
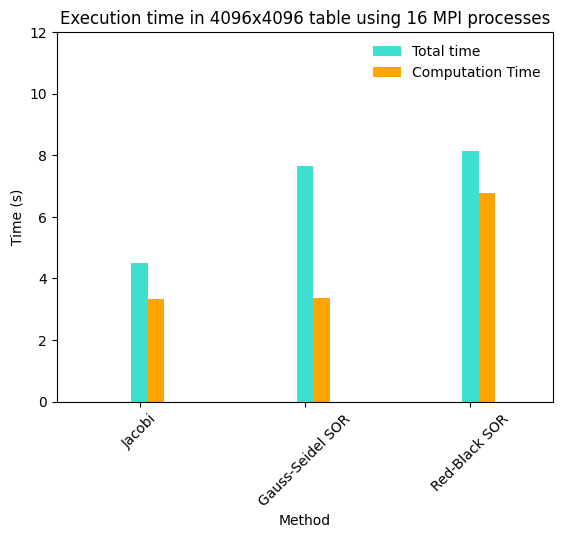
***20482048***

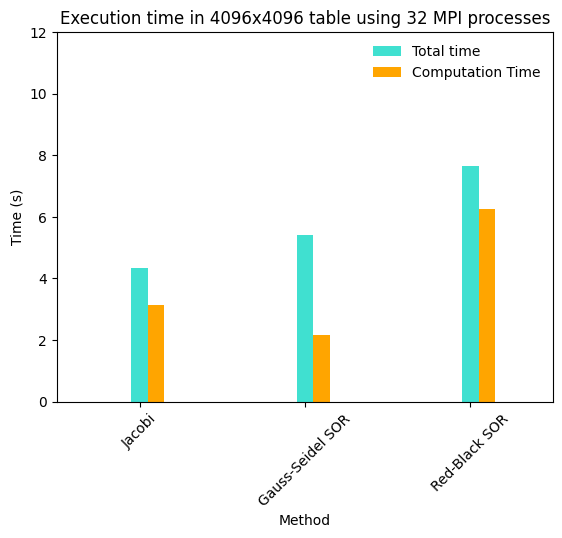
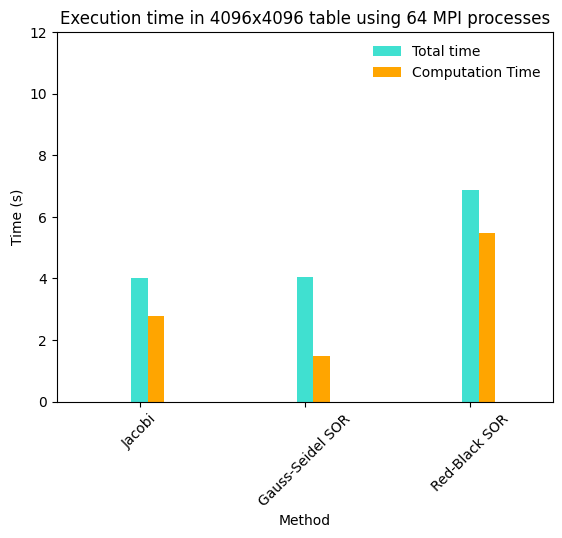
******

******

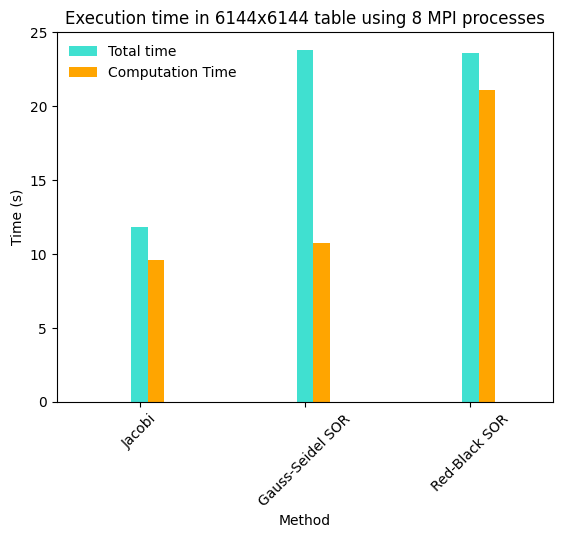
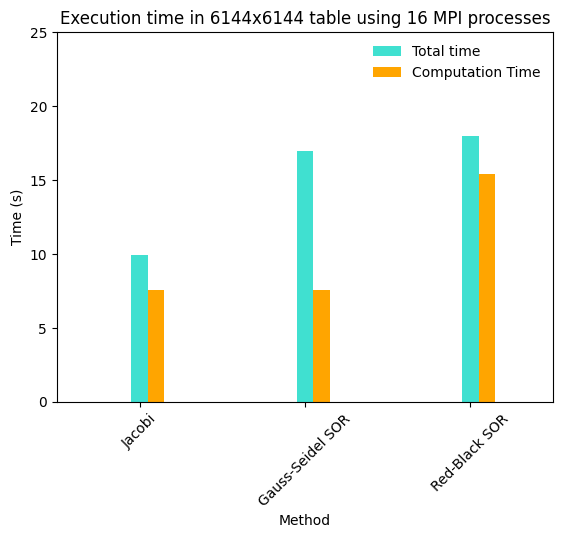
******

***40964096***

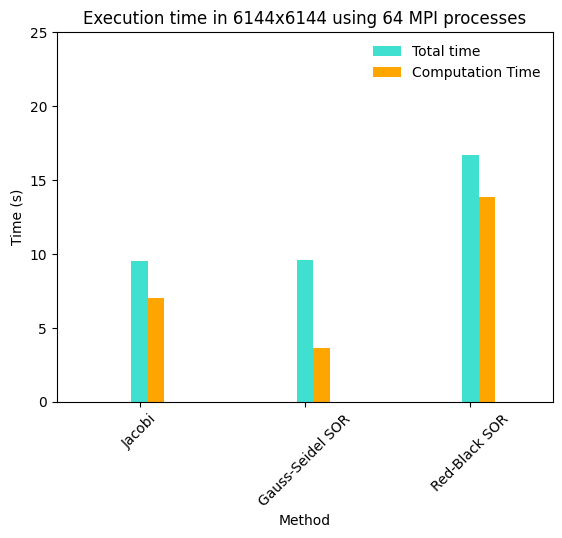
******

******

***61446144***

******

******

******

Από τα παραπάνω διαγράμματα προκύπτουν οι εξής παρατηρήσεις:

Είναι φανερό πως η μέθοδος Jacobi είναι ταχύτερη από τις άλλες δύο μεθόδους όταν έχουμε σταθερό αριθμό επαναλήψεων. Αυτό συμβαίνει διότι έχει μικρότερο υπολογιστικό κόστος και μικρότερο χρόνο επικοινωνίας συγκριτικά με τις μεθόδους Gauss-Siedel και Red-Black. Επιπρόσθετα, παρατηρούμε ότι για μέγεθος πίνακα 2048×2048, όλες οι μέθοδοι έχουν πολύ καλή κλιμάκωση, ενώ για μεγέθη 4096×4096, 61446144 φαίνεται πως οι μέθοδοι Gauss-Siedel και Red-Black κλιμακώνουν αρκετά καλύτερα από την Jacobi, ειδικά η Gauss-Siedel η οποία στα τελευταία bar plots φτάνει σχεδόν τον χρόνο της Jacobi, παρά τον μεγαλύτερο χρόνο επικοινωνίας που έχει. Πρέπει να σημειωθεί ακόμα ότι η Red-Black έχει σχεδόν διπλάσιο υπολογιστικό κόστος από τις άλλες δύο μεθόδους, γεγονός που εμποδίζει την περεταίρω κλιμάκωσή της, αν και παρουσιάζει μεγαλύτερο speedup.

Συνολικά, λοιπόν, o Gauss-Seidel SOR, αν και έχει μεγάλο κόστος επικοινωνίας, κλιμακώνει καλύτερα μειώνοντας συνεχώς το συνολικό του χρόνο, κάτι που οφείλεται σίγουρα και στην non-blocking διαχείριση της επικοινωνίας. Από την άλλη, o Red-Black SOR και o Jacobi δεν κλιμακώνουν το ίδιο καλά μιας και τα blocking SendRecv δεν διευκολύνουν τη μείωση του χρόνου επικοινωνίας.