

Ελένη Γεωργούδη – ics23103

Όλγα Σαρίδου – ics23078

Βαλεντίνος Γούδας – ics23170

# Super Mario Denoising: A Numerical Analysis Approach

Applied Numerical Analysis

Applied Informatics

UOM 2025-26

# TABLE OF CONTENTS

1. Εισαγωγή - Παρουσίαση προβλήματος .....	3
2. Μοντελοποίηση προβλήματος .....	4
2.1 Ψηφιοποίηση και Διαχωρισμός Καναλιών .....	4
2.2 Μαθηματική Σχέση Εξομάλυνσης .....	4
2.3 Συνοριακές Συνθήκες .....	5
3. Αριθμητική μέθοδος που χρησιμοποιήθηκε .....	6
3.1 Μέθοδος Jacobi .....	6
3.2 Μέθοδος Gauss-Seidel .....	7
3.3 Μέθοδος Successive Over Relaxation (SOR) .....	7
4. Λεπτομέρειες υλοποίησης .....	9
4.1 Εργαλεία και Περιβάλλον Ανάπτυξης .....	9
4.2 Διαχείριση Τύπων Δεδομένων και Ακρίβειας .....	10
4.3 Δομή Συναρτήσεων .....	11
4.4. Ροή Εκτέλεσης Κύριου Κώδικα .....	12
5. Αριθμητικά αποτελέσματα .....	12
5.1 Αξιολόγηση MSE .....	13
5.2 Οπτική Σύγκριση και Σχολιασμός .....	14
5.2.1 Οπτικοποίηση Jacobi .....	15
5.2.2 Οπτικοποίηση Gauss-Seidel .....	17
5.2.3 Οπτικοποίηση SOR .....	20
5.2.4 Υπεροχή SOR .....	22
5.3 Συμπεράσματα .....	24

# 1. Εισαγωγή - Παρουσίαση προβλήματος

Η επεξεργασία ψηφιακών εικόνων έρχεται συχνά αντιμέτωπη με την αλλοίωση της ποιότητας των δεδομένων, η οποία αποτελείται από την παρεμβολή ποικίλων μορφών θορύβου (noise) στην αρχική εικόνα. Αυτή, χάνει την καθαρότητά της και την ακρίβεια της οπτικής της πληροφορίας. Στην παρούσα εργασία, εστιάζουμε στην διαδικασία αφαίρεσης θορύβου (denoising) μιας ψηφιακής απεικόνισης του Super Mario. Στόχος είναι η ανακατασκευή της αρχικής πληροφορίας μέσω ελαχιστοποίησης των ανεπιθύμητων παρεμβολών, διατηρώντας όσο το δυνατόν περισσότερα χαρακτηριστικά της αρχικής εικόνας.

Η διαδικασία μείωσης θορύβου μοντελοποιείται ως ένα μαθηματικό πρόβλημα όπου η ψηφιακή εικόνα αντιμετωπίζεται ως μία σύνθετη δομή δεδομένων που αποτελείται από τρία επίπεδα πληροφορίας. Συγκεκριμένα, η αρχική τρισδιάστατη αναπαράσταση της εικόνας (RGB) αποσυντίθεται σε τρεις ανεξάρτητους δισδιάστατους πίνακες, οι οποίοι αντιστοιχούν στα κανάλια χρώματος Red, Green και Blue. Η επεξεργασία και αφαίρεση του θορύβου πραγματοποιείται ξεχωριστά σε κάθε κανάλι.

Σκοπός της αναφοράς είναι η αξιολόγηση της αποδοτικότητας των τριών διαφορετικών επαναληπτικών αριθμητικών αλγορίθμων μείωσης θορύβου. Συγκεκριμένα, συγκρίνονται οι μέθοδοι Jacobi, Gauss-Seidel και Successive Over Relaxation (SOR). Η αξιολόγηση βασίζεται στην οπτική ποιότητα της ανακατασκευασμένης εικόνας, της ταχύτητας σύγκλισης για ένα προκαθορισμένο αριθμό επαναλήψεων, αλλά και μέσω της μετρικής Μέσο Τετραγωνικό Σφάλμα (MSE). Επιπλέον, διερευνάται η επίδραση της παραμέτρου  $\omega$  (omega) στη μέθοδο SOR, με στόχο τον εντοπισμό της βέλτιστης τιμής η οποία θα εξασφαλίσει τη μέγιστη ταχύτητα σύγκλισης.

## 2. Μοντελοποίηση προβλήματος

### 2.1 Ψηφιοποίηση και Διαχωρισμός Καναλιών

Η έγχρωμη εικόνα εισόδου είναι αρχικά σε μορφή τρισδιάστατου πίνακα RGB. Την προσεγγίζουμε ως μια σύνθετη δομή δεδομένων που αποτελείται από τρία επίπεδα πληροφορίας. Για την αριθμητική επεξεργασία, η αρχική τρισδιάστατη αναπαράσταση αποσυντίθεται σε τρεις ανεξάρτητους δισδιάστατους πίνακες, οι οποίοι αντιστοιχούν στα κανάλια Red ( $f_R$ ), Green ( $f_G$ ) και Blue ( $f_B$ ). Αυτή η δομή μας επιτρέπει να εφαρμόσουμε τους αλγορίθμους μείωσης θορύβου σε κάθε χρώμα ξεχωριστά, σαν να επεξεργαζόμαστε τρεις διαφορετικές στρώσεις πληροφορίας.

Είναι σημαντικό να σημειωθεί, το γεγονός ότι η εικόνα του Super Mario έχει ξεκάθαρα έντονα χρώματα RGB καθιστά ευκολότερη την εφαρμογή των αλγορίθμων σε κάθε κανάλι αυτόνομα. Έτσι, βοηθάει και τον εκάστοτε αλγόριθμο να διατηρήσει την ένταση αυτών των χρωμάτων, χωρίς να μπερδεύονται μεταξύ τους κατά τη διάρκεια του denoising.

### 2.2 Μαθηματική Σχέση Εξομάλυνσης

Η διαδικασία μείωσης θορύβου βασίζεται στη θεωρία ότι η βέλτιστη τιμή ενός εικονοστοιχείου (Pixel)  $u(i,j)$  προκύπτει από έναν γραμμικό συνδυασμό της αρχικής θορυβώδους τιμής  $f(i,j)$  και της πληροφορίας που λαμβάνουμε από τον άμεσο περίγυρό του.

Στον κώδικα, η σχέση αυτή υλοποιείται από την εξίσωση:

$$u(i,j) = a*f(i,j) + b*( u(i-1,j) + u(i+1,j) + u(i,j-1) + u(i,j+1) )$$

Ανάλυση των παραμέτρων:

Ο όρος  $a*f(i,j)$  αντιπροσωπεύει την αρχική πληροφορία που θα κρατήσουμε στην ανακατασκευασμένη εικόνα. Με την σταθερά  $a=0.08$  επιλέγουμε το ποσοστό της αρχικής τιμής/ πληροφορίας που θα προέρχεται από την εικόνα με τον θόρυβο, ώστε να μη χαθούν εντελώς τα χαρακτηριστικά της.

Ο όρος  $b^*$  (...) αντιπροσωπεύει την εξομάλυνση (smoothing). Οι τιμές  $u(i-1,j)$ ,  $u(i+1,j)$ ,  $u(i,j-1)$  και  $u(i,j+1)$  είναι οι τέσσερις άμεσοι γείτονες του εικονοστοιχείου που επεξεργαζόμαστε. Είναι τα στοιχεία πάνω, κάτω, αριστερά και δεξιά αντίστοιχα. Με τη σταθερά  $b=0.23$  επιβάλουμε σε κάθε pixel να μοιάζει περισσότερο με τους γείτονές του. Έτσι, εξαλείφουμε τις απότομες διακυμάνσεις που επιφέρει ο θόρυβος.

Μία σημαντική λεπτομέρεια η οποία αξίζει να αναφερθεί είναι πως το άθροισμα των συντελεστών  $a$  και  $b$ :  $a+4*b$ , όπου  $b$  πολλαπλασιασμένο με τους 4 γείτονές του, παράγει σχεδόν την μονάδα ( $0.08 + 4*0.23 = 1$ ). Αυτό μας εξασφαλίζει πως η συνολική φωτεινότητα της εικόνας θα παραμείνει σταθερή και η εικόνα δεν θα «ασπρίσει» ούτε θα «σκοτεινιάσει» υπερβολικά κατά την διάρκεια των επαναλήψεων.

## 2.3 Συνοριακές Συνθήκες

Κατά την εφαρμογή των επαναληπτικών μεθόδων, προκύπτει το ζήτημα της επεξεργασίας των pixel που βρίσκονται στα όρια την εικόνας (border pixel). Για τα συγκεκριμένα σημεία δεν είναι δυνατός ο υπολογισμός των τεσσάρων γειτόνων και συνεπώς οι τιμές τους παραμένουν σταθερές και ίσες με τις αρχικές της θορυβώδους εικόνας.

Στην υλοποίηση του κώδικα, αυτό είναι εμφανές στο εύρος των δεικτών στους βρόχους επανάληψης των επαναληπτικών μεθόδων μείωσης θορύβου. Συγκεκριμένα, οι πίνακες στη Python ξεκινάνε την αρίθμηση τους από το 0. Το γεγονός ότι η αρχή των δύο loop σε κάθε αλγόριθμο στον κώδικα ξεκινάει από το 1 σημαίνει ότι αλγόριθμος προσπερνάει την πρώτη γραμμή και στήλη. Αντίστοιχα, τα loop τελειώνουν στο  $height-1$  και  $width-1$ , δηλαδή ακριβώς πριν την τελευταία γραμμή και στήλη.

### 3. Αριθμητική μέθοδος που χρησιμοποιήθηκε

Στο κεφάλαιο αυτό αναλύονται διεξοδικά οι τρεις επαναληπτικές μέθοδοι επίλυσης γραμμικών συστημάτων που εφαρμόστηκαν για την απαλοιφή θορύβου: Jacobi, Gauss - Seidel και Successive Over Relaxation(SOR)

Το πρόβλημα της απαλοιφής του θορύβου, μοντελοποιείται ως ένα σύστημα γραμμικών εξισώσεων, όπου κάθε pixel, αποτελεί και έναν άγνωστο. Δεδομένου ότι μία τυπική εικόνα αποτελείται από χιλιάδες ή εκατομμύρια pixels, ο πίνακας συντελεστών του συστήματος είναι τεράστιος. Λόγω αυτού του μεγέθους, οι άμεσες μέθοδοι επίλυσης (όπως είναι η απαλοιφή Gauss) είναι υπολογιστικά δύσκολες. Αντίθετα, οι επαναληπτικές μέθοδοι που ακολουθούν, ξεκινούν από μία αρχική προσέγγιση, δηλαδή την θορυβώδη εικόνα, και βελτιώνουν την λύση βήμα βήμα, μειώνοντας το σφάλμα σε κάθε επανάληψη.

Η αποτελεσματικότητα των επαναληπτικών μεθόδων στη συγκεκριμένη εφαρμογή βασίζεται στην ιδιότητα της "τοπικότητας". Σε μια εικόνα, κάθε pixel επηρεάζεται άμεσα μόνο από τους γείτονές του (πάνω, κάτω, αριστερά, δεξιά), γεγονός που σημαίνει ότι ο πίνακας του συστήματος είναι "αραιός" (sparse). Αυτό μας επιτρέπει να αποφύγουμε την αποθήκευση και τις πράξεις με εκατομμύρια μηδενικά στοιχεία που θα απαιτούσε μια άμεση μέθοδος. Έτσι, κάθε επανάληψη των Jacobi, Gauss-Seidel και SOR περιορίζεται μόνο σε τοπικούς υπολογισμούς, καθιστώντας την επεξεργασία μεγάλων εικόνων υπολογιστικά εφικτή και γρήγορη.

#### 3.1 Μέθοδος Jacobi

Η μέθοδος Jacobi είναι η απλούστερη μορφή επαναληπτικής μεθόδου και βασίζεται στην ιδέα της ταυτόχρονης ενημέρωσης όλων των σημείων του πλέγματος.

##### **Μαθηματική μέθοδος & Υλοποίηση**

Ο τύπος της μεθόδου είναι :

$$u_{i,j}^{(k+1)} = a \cdot f_{i,j} + b \cdot (u_{i-1,j}^{(k)} + u_{i+1,j}^{(k)} + u_{i,j-1}^{(k)} + u_{i,j+1}^{(k)})$$

Βασικό χαρακτηριστικό της μεθόδου είναι ότι για τον υπολογισμό της νέας τιμής  $u^{(k+1)}$  ενός pixel, χρησιμοποιούνται αποκλειστικά οι τιμές των γειτόνων από την

προηγούμενη επανάληψη  $k$ . Αυτό σημαίνει ότι η πληροφορία της εξομάλυνσης δεν μεταδίδεται άμεσα στους γείτονες εντός του ίδιου κύκλου επανάληψης.

Κατά την υπολογιστική υλοποίηση, η μέθοδος Jacobi απαιτεί τη διατήρηση δύο πινάκων στην μνήμη: έναν για την ανάγνωση των παλιών τιμών  $u$  και έναν για την εγγραφή των νέων  $u_{\text{new}}$ . Αυτό διασφαλίζει ότι οι υπολογισμοί είναι ανεξάρτητοι μεταξύ τους, αλλά γίνεται η σύγκλιση της μεθόδου αργή, καθώς η πληροφορία μειώνεται στο πλέγμα με ρυθμό μόλις ενός pixel ανά επανάληψη.

### 3.2 Μέθοδος Gauss-Seidel

Η μέθοδος Gauss-Seidel αποτελεί βελτιστοποίηση της Jacobi και επιταχύνει τη διαδικασία σύγκλισης εκμεταλλευόμενη άμεσα τα νέα δεδομένα.

#### Μαθηματική Λογική & Υλοποίηση:

Ο τύπος διαμορφώνεται ως εξής:

$$u_{i,j}^{(k+1)} = a \cdot f_{i,j} + b \cdot (u_{i-1,j}^{(k+1)} + u_{i+1,j}^{(k)} + u_{i,j-1}^{(k+1)} + u_{i,j+1}^{(k)})$$

Ουσιαστικά, η διαφορά εδώ είναι η χρήση των δεικτών  $(k+1)$  για τους γείτονες που έχουν ήδη επεξεργαστεί (πάνω και αριστερά, αν η σάρωση γίνεται κατά γραμμές). Αυτό σημαίνει ότι μόλις ένα pixel "καθαρίσει", η βελτιωμένη τιμή του επηρεάζει αμέσως τον επόμενο γείτονα στην **ίδια** επανάληψη.

Στον κώδικα, δεν απαιτείται δεύτερος βοηθητικός πίνακας, καθώς κάθε νέα τιμή αντικαθιστά την παλιά απευθείας στη μνήμη. Αυτή η άμεση ροή πληροφορίας επιτρέπει στη μέθοδο Gauss-Seidel να συγκλίνει συνήθως με σχεδόν διπλάσια ταχύτητα σε σχέση με τη Jacobi.

### 3.3 Μέθοδος Successive Over Relaxation (SOR)

Η μέθοδος SOR αποτελεί μια γενίκευση της Gauss-Seidel, σχεδιασμένη για να επιταχύνει περαιτέρω τη σύγκλιση μέσω της τεχνικής της "υπερ-χαλάρωσης" (over-relaxation).

#### Μαθηματική Λογική & Υλοποίηση:

Η σχέση που χρησιμοποιείται είναι:

$$u_{i,j}^{(k+1)} = (1-\omega)u_{i,j}^{(k)} + \omega \cdot GS_{i,j}^{(k+1)}$$

όπου  $GS_{i,j}^{(k+1)}$  είναι η τιμή που θα έδινε η μέθοδος Gauss-Seidel.

Η παράμετρος  $\omega$  (omega) λειτουργεί ως ρυθμιστής του "βήματος" που κάνει ο αλγόριθμος:

- Όταν  $\omega=1$ : Η μέθοδος ταυτίζεται πλήρως με την Gauss-Seidel.
- Όταν  $1 < \omega < 2$  (Υπερ-χαλάρωση): Αυτό σημαίνει πρακτικά, ότι ο αλγόριθμος σε αυτή την περίπτωση προχωράει με αρκετά μεγάλο βήμα. Υποθέτει ότι η τιμή του pixel θα συνεχίσει να κινείται προς την ίδια κατεύθυνση κάνοντας έτσι άλμα μεγαλύτερο από το κανονικό επιδιώκοντας να φτάσει στη λύση ταχύτερα.
- Όταν  $\omega < 1$  (Υπο-χαλάρωση): Ο αλγόριθμος εδώ, αντιδριάζει λίγο αντίθετα, από ότι στην προηγούμενη περίπτωση της υπερ-χαλάρωσης, καθώς, προχωράει με αργό βήμα. Χρησιμοποιείται σε περιπτώσεις έντονης αστάθειας των τιμών, αναγκάζοντας το σύστημα να κινείται πιο προσεκτικά.

## Πειραματικός Προσδιορισμός του $\omega$

Επειδή η βέλτιστη τιμή του omega εξαρτάται από τη δομή της συγκεκριμένης εικόνας, υλοποιήθηκε μια διαδικασία **πειραματικής βελτιστοποίησης**.

1. **Στρατηγική "Δείγματος"**: Η δοκιμή των τιμών του omega έγινε αποκλειστικά στο **κόκκινο κανάλι ( $f_R$ )**. Αυτό επιλέχθηκε επειδή στην εικόνα του Mario το κόκκινο χρώμα είναι ιδιαίτερα κυρίαρχο και αντιπροσωπευτικό. Με τον τρόπο αυτό, μειώθηκε το υπολογιστικό κόστος στο 1/3, καθώς ο θόρυβος είναι ομοιογενής και στα τρία κανάλια (RGB).
2. **Διαδικασία Δοκιμών**: Τρέξαμε τον αλγόριθμο για 10 επαναλήψεις με ένα εύρος τιμών omega [0.8, 1.8].
3. **Κριτήρια Επιλογής**: Η επιλογή βασίστηκε στην ελαχιστοποίηση του **Μέσου Τετραγωνικού Σφάλματος (MSE)** και στην αποφυγή φαινομένων αστάθειας



(όπως το "τρεμόπαιγμα" ή η εμφάνιση τεχνητών χρωμάτων που παρατηρείται σε πολύ υψηλές τιμές.

**Αποτέλεσμα:** Για την εικόνα του Mario, η βέλτιστη τιμή που προέκυψε από τον πειραματισμό ήταν  **$\omega = 0.8$** , εξασφαλίζοντας την πιο σταθερή και οπτικά βέλτιστη αποθορυβοποίηση.

Συνοψίζοντας, η επιλογή της κατάλληλης μεθόδου αποτελεί μια ισορροπία μεταξύ ταχύτητας σύγκλισης και σταθερότητας. Η μέθοδος Jacobi, αν και απλή, υστερεί σε ταχύτητα καθώς η πληροφορία "ταξιδεύει" αργά μέσα στο πλέγμα των pixels. Η Gauss-Seidel βελτιώνει αισθητά την κατάσταση ενσωματώνοντας τις νέες τιμές αμέσως μόλις υπολογιστούν. Τέλος, η μέθοδος SOR προσφέρει τον απόλυτο έλεγχο μέσω της παραμέτρου  $\omega$ . Στην περίπτωση της εικόνας του Mario, ο πειραματισμός έδειξε ότι η χρήση της υπο-χαλάρωσης ( $\omega = 0.8$ ) ήταν η πλέον ενδεδειγμένη, καθώς προσέφερε την απαραίτητη σταθερότητα για ένα καθαρό οπτικό αποτέλεσμα, αποτρέποντας τις απότομες διακυμάνσεις των τιμών που θα μπορούσαν να εισάγουν νέα σφάλματα στην εικόνα.

## 4. Λεπτομέρειες υλοποίησης

Στο κεφάλαιο αυτό περιγράφεται η αρχιτεκτονική του κώδικα και οι τεχνικές επιλογές που διασφαλίζουν την ορθή εφαρμογή των αριθμητικών μεθόδων στην ψηφιακή εικόνα.

### 4.1 Εργαλεία και Περιβάλλον Ανάπτυξης

Η υλοποίηση του κώδικα πραγματοποιήθηκε στη γλώσσα προγραμματισμού Python, αξιοποιώντας τον συνδυασμό δύο εξειδικευμένων βιβλιοθηκών:

**1. OpenCV (cv2):** Χρησιμοποιήθηκε για την ανάγνωση και αποθήκευση αρχείων εικόνας, καθώς και για την μετατροπή του χρωματικού χώρου από το πρότυπο BGR (το native της OpenCV) στο πρότυπο RGB. Αν δεν πραγματοποιηθεί η μετατροπή, ο πίνακας που είναι το Red κανάλι θα περιέχει στην πραγματικότητα τις τιμές του Μπλε. Αυτό θα οδηγήσει σε λάθος αποτελέσματα κατά την

επανένωση των χρωμάτων, καθώς τα χρώματα της Mario εικόνας θα φαίνονται μπερδεμένα (π.χ. το κόκκινο καπέλο θα βγει μπλε).

**2. NumPy:** Αποτελεί το βασικότερο εργαλείο ώστε να μπορέσουμε να υλοποιήσουμε τις μαθηματικές συναρτήσεις για τη μείωση θορύβου της εικόνας. Μέσω του NumPy πραγματοποιείται η αποσύνθεση (slicing) του τρισδιάστατου πίνακα RGB σε τρεις ανεξάρτητους πίνακες (channels). Συνεισφέρει, ακόμη, και στην εκτέλεση των αριθμητικών πράξεων των αλγορίθμων. Συγκεκριμένα, οι αλγόριθμοι Jacobi, Gauss-Seidel και SOR, βασίζονται σε επαναληπτικές πράξεις πρόσθεσης και πολλαπλασιασμού για κάθε ένα από τα εκατομμύρια pixels της εικόνας. Το NumPy επιτρέπει την εκτέλεση αυτών των πράξεων με εξαιρετική ταχύτητα μέσω της χρήσης πινάκων. Κάθε φορά που ο κώδικας εκτελεί τη πράξη  $a * f(i,j) + b * (u(i-1,j) + u(i+1,j) + u(i,j-1) + u(i,j+1))$ , το NumPy διαχειρίζεται την αποθήκευση και την ακρίβεια των δεκαδικών ψηφίων (float64). Τέλος, μέσω του NumPy πραγματοποιείται και η επανασύνδεση των καναλιών (stacking). Αφού οι αλγόριθμοι ολοκληρώσουν την επεξεργασία, δίνουν ως αποτέλεσμα τρεις δισδιάστατους πίνακες που περιέχουν τις «καθαρές» πλέον τιμές των καναλιών RGB. Χρησιμοποιεί τη συνάρτηση `np.stack`, η οποία τοποθετεί τους τρεις πίνακες τον έναν πάνω στον άλλον, δημιουργώντας ξανά έναν τρισδιάστατο πίνακα. Αυτή η δομή αποθηκεύεται σε ένα αρχείο εικόνας .png.

## 4.2 Διαχείριση Τύπων Δεδομένων και Ακρίβειας

Μία κρίσιμη λεπτομέρεια της υλοποίησης είναι η μετατροπή των δεδομένων. Οι ψηφιακές εικόνες αποτελούνται από ακέραιους αριθμούς 8 bit στο διάστημα 0-255. Ωστόσο, οι μαθηματικές μέθοδοι απαιτούν υψηλή μαθηματική ακρίβεια ώστε να συγκλίνουν.

**1. Μετατροπή σε float64:** Κατά την δημιουργία των καναλιών, οι τιμές μετατρέπονται σε αριθμούς κινητής υποδιαστολής διπλής ακρίβειας, ώστε να αποφευχθούν σφάλματα στρογγυλοποίησης κατά τους χιλιάδες υπολογισμούς. Είναι εξίσου σημαντικό να αναφερθεί πως η χρήση του float64 διασφαλίζει ότι η τελική τιμή του MSE θα αντικατοπτρίζει την πραγματική μαθηματική απόδοση του εκάστοτε αλγορίθμου, χωρίς να επιβαρύνεται από υπολογιστικές αστάθειες ή συσσώρευση σφαλμάτων στρογγυλοποίησης.

**2. Clipping και Κανονικοποίηση:** Μετά την ολοκλήρωση των επαναλήψεων, χρησιμοποιήθηκε η συνάρτηση `np.clip(x_matrix, 0, 255)`. Αυτό είναι απαραίτητο καθώς οι αριθμητικές μέθοδοι ενδέχεται να παράξουν τιμές ελαφρώς εκτός ορίων (0-255). Οι τιμές αυτές πρέπει να επαναφερθούν εντός ορίων πριν την μετατροπή σε ακέραιο 8 bit για την ορθή απεικόνιση της εικόνας.

## 4.3 Δομή Συναρτήσεων

Στον κώδικα κάθε συνάρτηση επιτελεί έναν εξειδικευμένο ρόλο στην διαδικασία μείωσης θορύβου μιας εικόνας. Συγκεκριμένα οι συναρτήσεις που υλοποιήθηκαν είναι οι εξής:

**Συνάρτηση `image_to_channels`:** Αποτελεί την πύλη εισόδου του προγράμματος. Διαβάζει την εικόνα, τη μετατρέπει σε RGB από BGR, και τη χωρίζει σε τρεις ανεξάρτητους δισδιάστατους πίνακες `float64`, προετοιμάζοντας έτσι τα δεδομένα για επεξεργασία. Η μετατροπή σε `float64` μέσω της εντολής `.astype(np.float64)` είναι απαραίτητη για τη διατήρηση της αριθμητικής ακρίβειας κατά τους υπολογισμούς.

**Συνάρτηση `channels_to_image`:** Είναι η αντίστροφη διαδικασία, η οποία αναλαμβάνει την ανασύνθεση. Πραγματοποιεί τον περιορισμό των τιμών (clipping) στο διάστημα `[0,255]` και ενώνει τα channels για την δημιουργία εικόνας.

**Συνάρτηση `calculate_mse`:** Υπολογίζει το Μέσο Τετραγωνικό Σφάλμα μεταξύ της καθαρής εικόνας (`mario_clean.png`) και της τρέχουσας επεξεργασμένης εικόνας. Λειτουργεί ως μαθηματικό μέτρο που μας επιτρέπει να κρίνουμε την ποιότητα της ανακατασκευασμένης εικόνας.

**Συναρτήσεις αλγορίθμων `jacobi`, `gauss_seidel` και `sor`:** Αποτελούν τους πυρήνες της επίλυσης. Κάθε μία εφαρμόζει τη δική της στρατηγική πάνω στα εικονοστοιχεία για μείωση θορύβου, δουλεύοντας πάνω σε πίνακες 2D. Επιστρέφουν μία βελτιωμένη έκδοση εικόνας σε μία επανάληψη.

## 4.4. Ροή Εκτέλεσης Κύριου Κώδικα

Η εκτέλεση του προγράμματος ξεκινάει από την αρχικοποίηση, όπου φορτώνονται οι εικόνες `mario_noisy.png` και `mario_clean.png`, και δημιουργούνται αντίγραφα των θορυβωδών πινάκων για κάθε μέθοδο ξεχωριστά, ώστε η σύγκριση να ξεκινάει από την ίδια αφετηρία.

Πριν την έναρξη του κύριου βρόχου, εκτελείται μία διαδικασία πειραματικού προσδιορισμού της παραμέτρου  $\omega$  (`omega`) για τη μέθοδο SOR. Μέσω μιας λίστας υποψήφιων τιμών (`candidate_omegas`) επιλέγεται εκείνη που παράγει το χαμηλότερο σφάλμα MSE στο κόκκινο κανάλι, διασφαλίζοντας τη βέλτιστη απόδοση του αλγορίθμου για τη συγκεκριμένη εικόνα.

Στη συνέχεια, το πρόγραμμα εισέρχεται στον κύριο βρόχο, εκτελώντας διαδοχικές επαναλήψεις και για τις τρεις μεθόδους παράλληλα. Σε κάθε βήμα οι τιμές των πινάκων RGB ενημερώνονται και η εικόνα εξομαλύνεται σταδιακά.

Για την παρακολούθηση της προόδου, εξάγονται τα αποτελέσματα των επαναλήψεων-στόχων (`target_iterations`) 2, 4, 6, 8, και 10. Σε αυτά τα σημεία, οι τρέχοντες πίνακες αποθηκεύονται ως αρχεία εικόνας και ταυτόχρονα καταγράφονται οι τιμές του MSE σε έναν συγκεντρωτικό πίνακα, ο οποίος εκτυπώνεται στο τέλος της εκτέλεσης του κύριου βρόχου για την τελική αξιολόγηση.

## 5. Αριθμητικά αποτελέσματα

Στο παρόν κεφάλαιο παρουσιάζονται τα αποτελέσματα της διαδικασίας αφαίρεσης θορύβου για την εικόνα του Super Mario. Η αξιολόγηση των μεθόδων Jacobi, Gauss-Seidel και SOR πραγματοποιείται μέσω της σύγκρισης της οπτικής ποιότητας των παραγόμενων εικόνων, καθώς και της ποσοτικής ανάλυσης της μείωσης σφάλματος MSE (Mean Squared Error) σε συνάρτηση με τον αριθμό των επαναλήψεων

## 5.1 Αξιολόγηση MSE

Ο παρακάτω πίνακας συγκεντρώνει τις τιμές του σφάλματος MSE για τις επαναλήψεις 2, 4, 6, 8 και 10. Το MSE μετρά τη μέση τετραγωνική απόκλιση μεταξύ της επεξεργασμένης εικόνας και της καθαρής εικόνας αναφοράς. Συνεπώς, όσο χαμηλότερη είναι η τιμή, τόσο επιτυχέστερη θεωρείται η ανακατασκευή.

Iter	Jacobi	Gauss-Seidel	SOR
2	43.14	45.52	38.40
4	49.06	58.40	49.60
6	54.98	65.03	57.44
8	59.51	68.69	62.52
10	62.93	70.88	65.93

Γίνεται αντιληπτή η υπεροχή της μεθόδου SOR, καθώς παρουσιάζει σταθερά τις χαμηλότερες τιμές σφάλματος MSE σε σύγκριση με τις άλλες δύο μεθόδους, ιδιαίτερα στις πρώτες επαναλήψεις. Αυτό επιβεβαιώνει την ορθότητα της πειραματικής διαδικασίας εύρεσης του βέλτιστου  $\omega$  (omega).

Παρατηρούμε επίσης πως η μέθοδος Gauss-Seidel, παρόλο που θεωρείται ταχύτερη από την Jacobi λόγω της χρήσης ανανεωμένων τιμών εντός της ίδιας επανάληψης, εμφανίζει υψηλότερες τιμές MSE. Αυτό ενδεχομένως οφείλεται στα έντονα χρώμα της εικόνας, όπου η πληροφορία διαχέεται ταχύτερα μεταξύ των pixels. Έτσι, η εξομάλυνση καθίσταται εντονότερη, και συνεπώς την ταχύτερη αλλοίωση των έντονων χαρακτηριστικών της εικόνας, γεγονός που διευρύνει την μαθηματική απόσταση MSE από την αρχική εικόνα.

Αξίζει να σημειωθεί πως οι τιμές του MSE αυξάνονται ελαφρώς σε κάθε επανάληψη. Αυτό ερμηνεύεται από το γεγονός ότι οι μέθοδοι επιβάλλουν διαρκή εξομάλυνση (blurring). Ενώ ο θόρυβος απομακρύνεται, η εικόνα σταδιακά χάνει την οξύτητά της, με αποτέλεσμα η απόκλιση από την αρχική εικόνα να αυξάνεται.

## 5.2 Οπτική Σύγκριση και Σχολιασμός

Πέρα από την αριθμητική ανάλυση του σφάλματος MSE, η οπτική αξιολόγηση των εικόνων που παρήχθησαν στις επαναλήψεις-στόχους (2, 4, 6, 8, 10) μας δίνει σημαντικά συμπεράσματα για τη συμπεριφορά των αλγορίθμων.

Πριν την ανάλυση των αποτελεσμάτων, παρατίθενται η αρχική καθαρή εικόνα αναφοράς και η αντίστοιχη θορυβώδης εικόνα εισόδου, ώστε να καταστεί σαφές το μέγεθος της αλλοίωσης που καλούνται να αποκαταστήσουν οι αλγόριθμοι.



*mario\_clean.png*



*mario\_noisy.png*



### 5.2.1 Οπτικοποίηση Jacobi

**Μέθοδος Jacobi:** Παρατηρείται ότι η εικόνα βελτιώνεται με τον πιο αργό ρυθμό. Ο έντονος θόρυβος της αρχικής εικόνας παραμένει ορατός για περισσότερες επαναλήψεις. Αυτό οφείλεται στο γεγονός ότι η πληροφορία της εξομάλυνσης διαδίδεται μόνο κατά ένα pixel ανά κατεύθυνση σε κάθε επανάληψη, καθώς ο αλγόριθμος χρησιμοποιεί αποκλειστικά το αντίγραφο `u.copy()` της προηγούμενης κατάστασης.



*mario\_jacobi\_iter2.png*



*mario\_jacobi\_iter4.png*



*mario\_jacobi\_iter6.png*



*mario\_jacobi\_iter8.png*





*mario\_jacobi\_iter10.png*

### 5.2.2 Οπτικοποίηση Gauss-Seidel

**Μέθοδος Gauss-Seidel:** Η απομάκρυνση του θορύβου είναι εμφανώς πιο γρήγορη από την Jacobi. Ωστόσο, οπτικά παρατηρείται μια τάση για πιο έντονο «θόλωμα» (blurring) στις ακμές του Mario. Οι αλλαγές συσσωρεύονται ταχύτερα, οδηγώντας σε μια πιο μαλακή εικόνα η οποία όμως χάνει σταδιακά τις λεπτομέρειες των χρωμάτων.



*mario\_gs\_iter2.png*



*mario\_gs\_iter4.png*



*mario\_gs\_iter6.png*



*mario\_gs\_iter8.png*



*mario\_gs\_iter10.png*



### 5.2.3 Οπτικοποίηση SOR

**Μέθοδος SOR (με  $\omega = 0.8$ ):** Η εικόνα που προκύπτει από τη μέθοδο SOR παρουσιάζει το πιο ισορροπημένο αποτέλεσμα, ειδικά στις πρώτες 4 επαναλήψεις. Η χρήση της υπο-χαλάρωσης (under-relaxation) με τιμή 0.8, φαίνεται αρκετά ευεργετική, καθώς εμποδίζει τον αλγόριθμο από το να κάνει «απώτομες» αλλαγές που θα μπορούσαν να αλλοιώσουν τη φιγούρα, διατηρώντας την καθαρότητα των χρωμάτων ενώ ταυτόχρονα μειώνει αποτελεσματικά τον θόρυβο.



*mario\_sor\_iter2.png*



*mario\_sor\_iter4.png*



*mario\_sor\_iter6.png*



*mario\_sor\_iter8.png*



*mario\_sor\_iter10.png*

Γενικά, και στις τρεις μεθόδους, καθώς ο αριθμός των επαναλήψεων αυξάνεται προς τις 10, η εικόνα γίνεται ολοένα και πιο θολή. Αυτό εξηγεί γιατί το MSE αυξάνεται μετά από ένα σημείο, καθώς ο αλγόριθμος, στην προσπάθειά του να αφαιρέσει τον θόρυβο, συνεχίζει να εφαρμόζει τον μέσο όρο των γειτόνων μέσω των σταθερών  $a$  και  $b$ , με αποτέλεσμα να απομακρύνεται από την αρχική καθαρή εικόνα αναφοράς. Η χρήση της `np.clip` στον κώδικα θεωρείται σημαντική, καθώς διατήρησε τη ζωντάνια των χρωμάτων της εικόνας στα επιτρεπτά όρια, αποτρέποντας οπτικές αλλοιώσεις στα σημεία υψηλής αντίθεσης.

#### 5.2.4 Υπεροχή SOR

Στην τελική φάση της οπτικής αξιολόγησης, παρατίθενται οι εικόνες που προέκυψαν από τη 10η επανάληψη και για τις τρεις μεθόδους. Οι εικόνες παρουσιάζονται εστιασμένες (cropped) στην περιοχή του προσώπου του χαρακτήρα, ώστε να καθίσταται δυνατή η λεπτομερής παρατήρηση του υπολειπόμενου θορύβου. Η σύγκριση αυτή καθιστά σαφές γιατί η μέθοδο SOR (Successive Over-Relaxation) επιλέγεται ως η βέλτιστη λύση για το συγκεκριμένο πρόβλημα ανακατασκευής.





*mario\_jacobi\_iter10.png*



*mario\_gs\_iter10.png*



*mario\_sor\_iter10.png*

## 5.3 Συμπεράσματα

Η παρούσα εργασία ανέδειξε την αποτελεσματικότητα των επαναληπτικών αριθμητικών μεθόδων στην επεξεργασία και αποθρομβοποίηση ψηφιακών εικόνων. Μέσα από τη μοντελοποίηση του προβλήματος και την υλοποίηση των αλγορίθμων Jacobi, Gauss-Seidel και SOR σε περιβάλλον Python, προκύπτουν τα εξής βασικά συμπεράσματα:

**Υπεροχή της μεθόδου SOR:** Η μέθοδος Successive Over Relaxation (SOR) αναδείχθηκε ως η πλέον αποδοτική, καθώς πέτυχε τη βέλτιστη ισορροπία μεταξύ μείωσης του σφάλματος MSE και διατήρησης της οπτικής ποιότητας. Η πειραματική διαδικασία προσδιορισμού του  $\omega=0.8$  απέδειξε ότι η υπο-χαλάρωση είναι καθοριστική για τη σταθερότητα του συστήματος σε εικόνες με έντονες χρωματικές αντιθέσεις, όπως αυτή του Super Mario, αποτρέποντας απότομες διακυμάνσεις των τιμών.

**Σχέση Ταχύτητας και Ποιότητας:** Επιβεβαιώθηκε η θεωρητική πρόβλεψη ότι η Gauss-Seidel συγκλίνει ταχύτερα από την Jacobi λόγω της "επί τόπου" (in-place) ενημέρωσης των τιμών, χρησιμοποιώντας άμεσα τα νέα δεδομένα των γειτόνων. Ωστόσο, η ταχύτερη σύγκλιση συνοδεύεται από εντονότερη εξομάλυνση (blurring), γεγονός που αυξάνει το MSE στις μεταγενέστερες επαναλήψεις λόγω της σταδιακής απώλειας της οξύτητας της εικόνας.

**Σημασία της Υπολογιστικής Υλοποίησης:** Η χρήση της βιβλιοθήκης NumPy ήταν καταλυτική για τη διαχείριση του μεγάλου όγκου δεδομένων και την εκτέλεση των πράξεων με υψηλή ταχύτητα μέσω πινάκων. Η μετατροπή των καναλιών σε float64 διασφάλισε την απαιτούμενη μαθηματική ακρίβεια κατά τους χιλιάδες υπολογισμούς, ενώ η χρήση της np.clip εγγυήθηκε την επαναφορά των τιμών στο έγκυρο εύρος 0-255 για την ορθή παραγωγή των αρχείων εικόνας.

**Περιορισμοί και Βελτιστοποίηση:** Διαπιστώθηκε ότι η αύξηση των επαναλήψεων πέρα από ένα βέλτιστο σημείο δεν οδηγεί απαραίτητα σε καλύτερα αποτελέσματα, καθώς η διαρκής εφαρμογή της εξομάλυνσης αλλοιώνει τα χαρακτηριστικά της αρχικής εικόνας, αυξάνοντας τη μαθηματική απόσταση MSE. Συνεπώς, η επιλογή του αριθμού των επαναλήψεων αποτελεί μια σημαντική παράμετρο βελτιστοποίησης για την τελική ποιότητα.

Συνολικά, η εργασία κατέδειξε ότι η αριθμητική ανάλυση προσφέρει ισχυρά εργαλεία για την επίλυση προβλημάτων ψηφιακής επεξεργασίας, αρκεί η επιλογή των παραμέτρων ( $\alpha$ ,  $b$ ,  $\omega$ , αριθμός επαναλήψεων) να γίνεται με γνώμονα τη φύση των δεδομένων εισόδου.