# תקציר

דוח זה מלווה פונקציה הפותרת באופן נומרי בעיה של הולכת-חום בלוח מתכת (הולכה דו-ממדית) במצב מתמיד. לטובת פתרון הבעיה הפונקציה יוצרת סדרה של משוואות לינאריות ופותרת אותן בעזרת שיטת יעקובי, עד קירוב סביר של הטמפ' בכל נקודה בלוח.
עבור לוח של 200*200, הפונקציה החזירה פתרון מדויק מסדר גודל של $10^{-6}$ מעלות תוך 7.5 שניות בממוצע, עבור הלוח המבוקש 16*16, הוחזר פתרון באופן מיידי.

# תיאור הבעיה

הטמפרטורה בכל נקודה (מלבד הקצוות) על הלוח היא ממוצע הטמפרטורות של 4 הנקודות הסמוכות אליו, ובהתייחסות כמטריצה:

$$4 * T_{i,j} - T_{i+1,j} - T_{i-1,j} - T_{i,j+1} - T_{i,j-1} \ = \ 0$$

בהינתן שגבולות הלוח ידועים וקבועים (שניים קבועים ושניים מתקבלים כקלט עם הקריאה), ניתן להניח שלמטריצה יש פתרון יציב (נוכל להוכיח זאת מאוחר יותר, בעזרת הערכים העצמיים של מטריצת הלפלסיאן (פתרון יציב ידרוש שהווקטורים העצמיים של המטריצה יוכלו לפרוס מ"ו של כל מטריצות-הטמפרטורות האפשריות), בינתיים נוכל להניח שהיות שמטריצה של 3*3 תהיה יציבה בתנאים הללו (גבולות קבועים), גם מטריצות גדולות ממנה יכולות להגיע למצב יציב (באינדוקציה).

עלינו לכתוב פונקציה שתקבל את הטמפרטורה בגבול העליון והתחתון של הלוח ותחזיר מטריצה של 16*16 נקודות ובכל אחת מהן הטמפרטורה באותה נקודה כך שמתקיימת המשוואה הנ"ל.

# שיטת פתרון

כדי להביא את המטריצה למצבה הסטטי, נבנה מערכת משוואות לינאריות מתאימה, לכל איבר במטריצה המקורית תותאם שורה, והמשוואה הנ"ל מתורגמת עבור מאטלאב (בתור שפת-תכנות המתייחסת לכל מטריצה כווקטור עמודה המורכב מעמודות המטריצה) למשוואה הבאה:

$$4 * T_i - T_{i+1} - T_{i-1} - T_{i+n_{rows}} - T_{i-n_{rows}} \ = \ 0$$

ומכאן שווקטור השורה המתאר את המשוואה הזו יראה כך:

$$\left(0 \dots (-1)_{i-n_{row}} \dots (-1)_{i-1} \ (4)_i \ (-1)_{i+1} \dots (-1)_{i+n_{row}} \dots 0\right)$$

בגבולות מטריצת-הטמפרטורה השורה מתנוונת ל-

$$(0 \dots \ (1)_i \dots 0)$$

היות שהגבולות קבועים.

כאשר וקטור המשתנים (להלן $\vec{x}$) הוא מטריצת-הטמפרטורה פרוסה כווקטור עמודה. אם עבור כל רכיב בוקטור-המשתנים ניצור משוואה לינארית נקבל את המטריצה הבאה (הקטנתי את גודל מטריצת הטמפרטורה ל- 3*3, כדי לקבל מטריצה בגודל נסבל):

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | -1 | 0 | -1 | 4 | -1 | 0 | -1 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

נקרא לה מטריצת הלפלסיאן $Lpc$, ואטען שעבור מצב יציב, חייב להתקיים $Lpc \cdot \vec{x} = \ \vec{x}$

ומכאן שהמצב היציב של הלוח נפרס ע"י הווקטורים העצמיים של המטריצה המתאימים לערך העצמי 1 (בדיקה זריזה מגלה שהם אכן פורסים את כל מרחב הפתרונות בהינתן גבולות קבועים). על מנת לחסוך חישוב של מטריצה הופכית מלאה, אשתמש בשיטת יעקובי כדי למצוא את ה-$\vec{x}$ המתאים לתנאי-השפה. עבור השיטה, מתקיים:

$$Lpc = L + D + U \ \Rightarrow \ Lpc \cdot \vec{x} = L \cdot \vec{x} + D \cdot \vec{x} + U \cdot \vec{x} = \ \vec{x} + \vec{r}$$

כאשר $L$ מטריצה משולשת תחתונה, $D$ מטריצה אלכסונית, $U$ מטריצה משולשת עליונה ו-$\vec{r}$ וקטור שגיאות.

מכאן בשיטת יעקובי:

$$D \cdot \vec{x} = -L \cdot \vec{x} - U \cdot \vec{x} + \vec{x} + \vec{r} \Rightarrow \vec{x} = D^{-1}(\vec{r} + (I - L - U) \cdot \vec{x})$$

[המטריצה היחידה שיש להפוך היא אלכסונית ולכן ההיפוך הוא מסדר גודל $n_{row}$ של המטריצה.]

כדי לבחור את $\vec{x} + \vec{r}$ אניח שאין שום שגיאה בגבולות (ולכן $\vec{r}_{i,j} = 0$ כאשר $((i,j) \in Boundary$) והם ידועים, ואניח שהשגיאה בכל שורה שאיננה הגבול היא בגודל האיבר של $\vec{x}$ בשורה הנ"ל (כך שלכל שורה $i$ שאיננה שורת-גבול מתקיים $\vec{x}_i + \vec{r}_i = 0$), ולכן כל איבר בשורה $i$ מקיים כל איטרציה את השוויון הבא:

$$x_{new_i} = D_{i,i}^{-1}\left(\vec{r}_i + \left(I(i) - L(i) - U(i)\right) \cdot \vec{x}\right)$$

כאשר:

$$D_{i,i}^{-1} = \frac{1}{4}\ ;\ \vec{r} \quad = -\vec{x} \quad ;$$

$$\begin{aligned}\left(I(i) - L(i) - U(i)\right) \cdot \vec{x} \\ &= \vec{x}_i - \left[(-1) * \vec{x}_{i-1} + (-1) * \vec{x}_{i-n_{rows}}\right] \\ &\quad - \left[(-1) * \vec{x}_{i+1} + (-1) * \vec{x}_{i+n_{rows}}\right]\end{aligned}$$

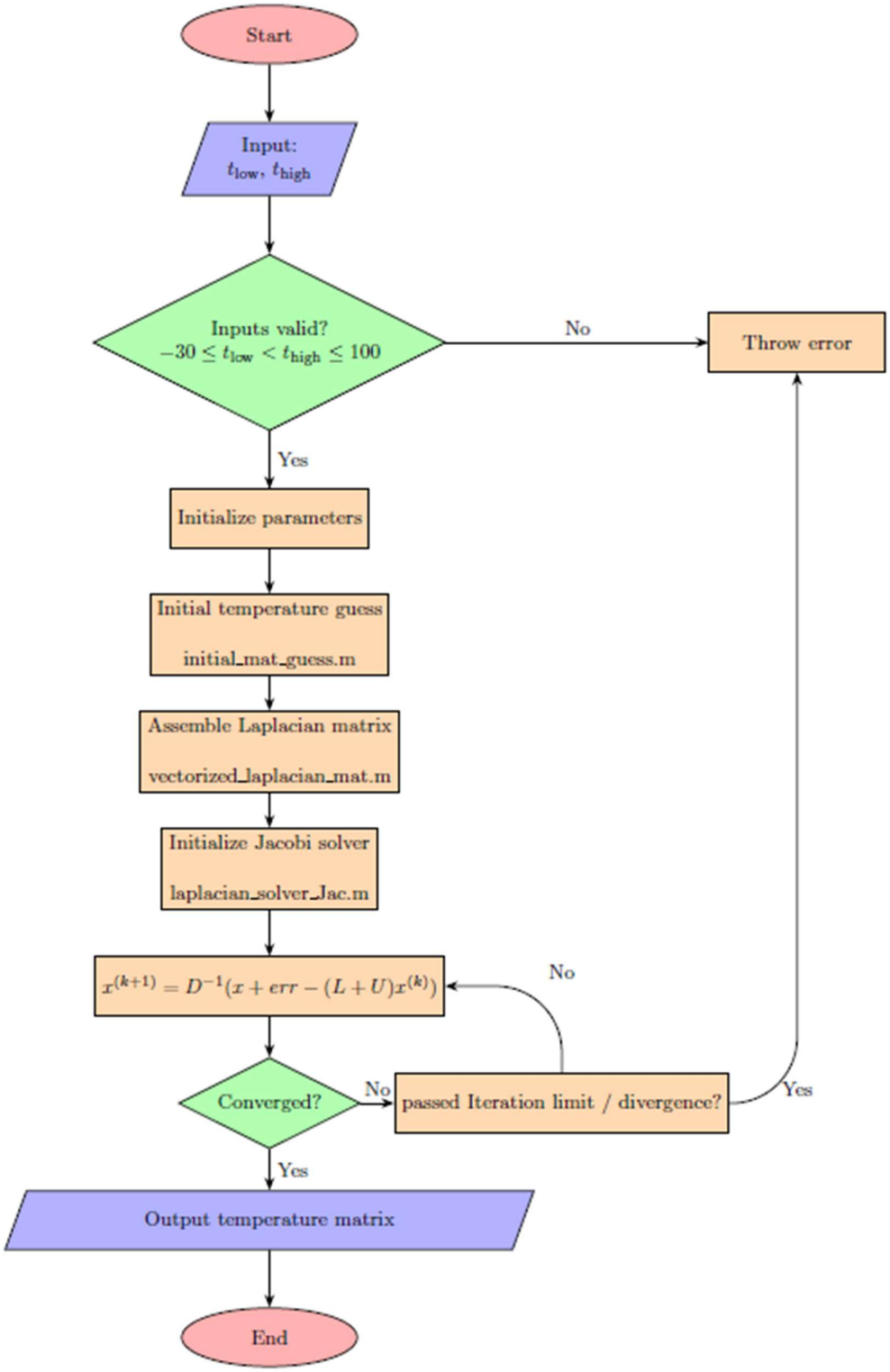
ולכן:

$$x_{new_i} = \frac{1}{4}\left[\vec{x}_{i-1} + \vec{x}_{i-n_{rows}} + \vec{x}_{i+1} + \vec{x}_{i+n_{rows}}\right]$$

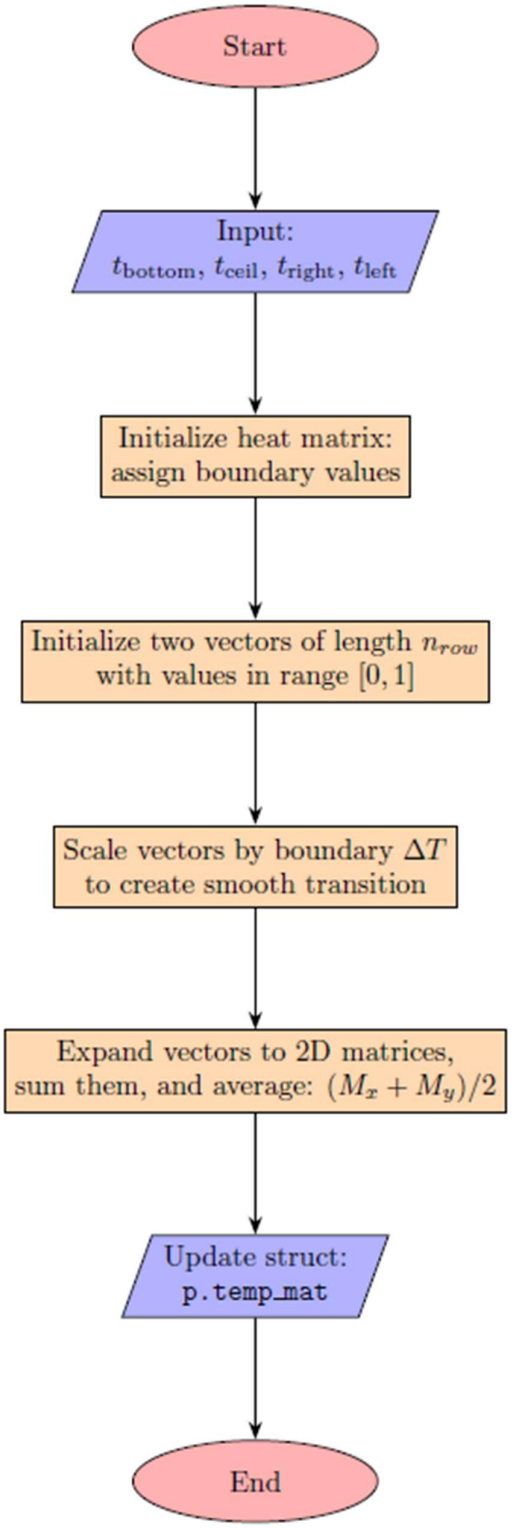
כנדרש.

תנאי העצירה הוא פשוט גבול למספר האיטרציות הנוכחי, ושינוי מקסימלי בין איטרציות (ריאלית, אין צורך בו, היות שבתוכנה כמו שהיא בנויה אין חשש להתבדרות). תנאי הסיום הוא ההפרש בין האיטרציות, במידה וההפרש המקסימלי בין איברי מטריצת הטמפרטורה באיטרציות עוקבות נמוך מאפסילון (המוגדר כ-$10^{-6}$ בפונקציה) ניתן להניח שהפונקציה התייצבה והתוצאה קרובה דיו לתוצאה הנכונה.

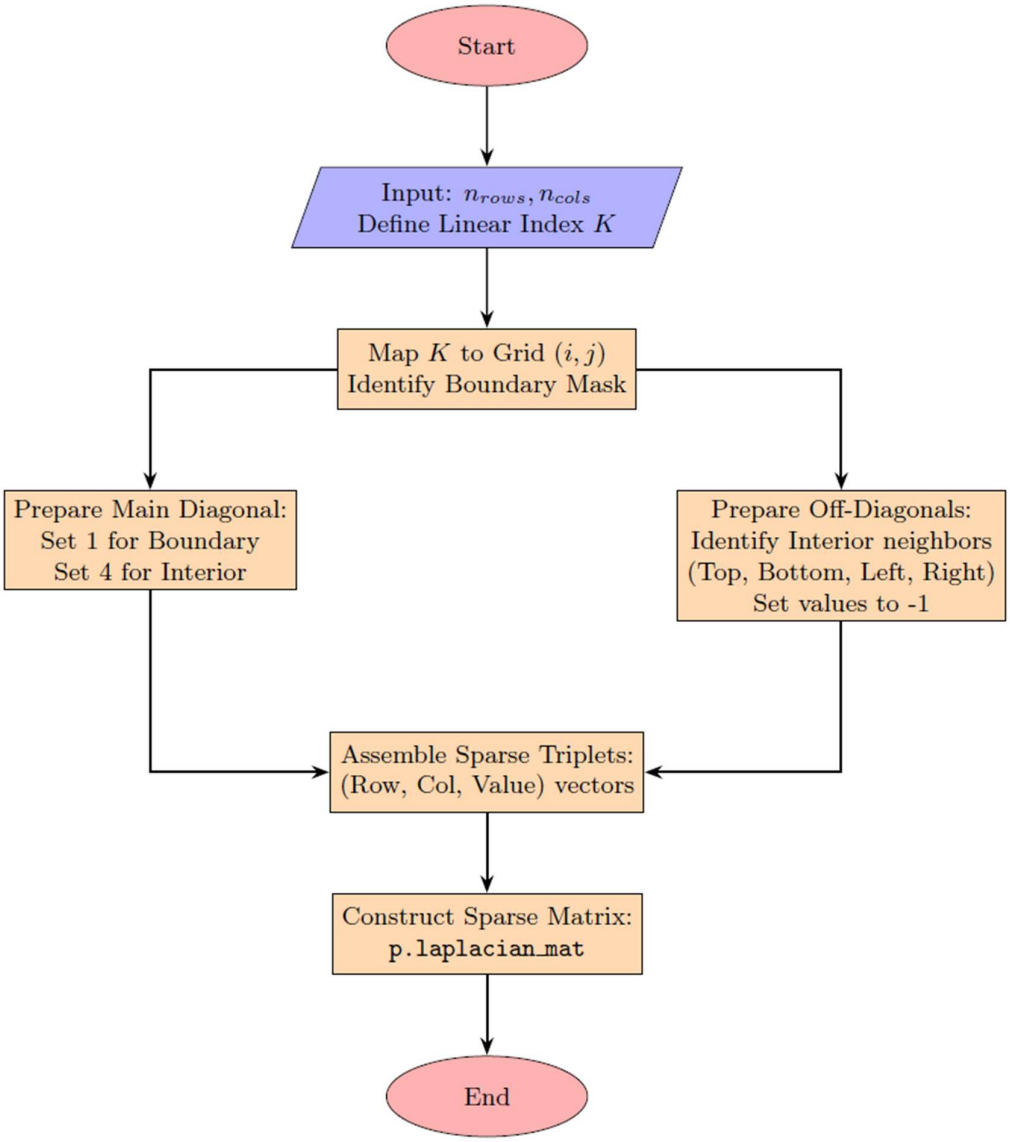
תרשים הזרימה עבור הפונקציה הראשית:



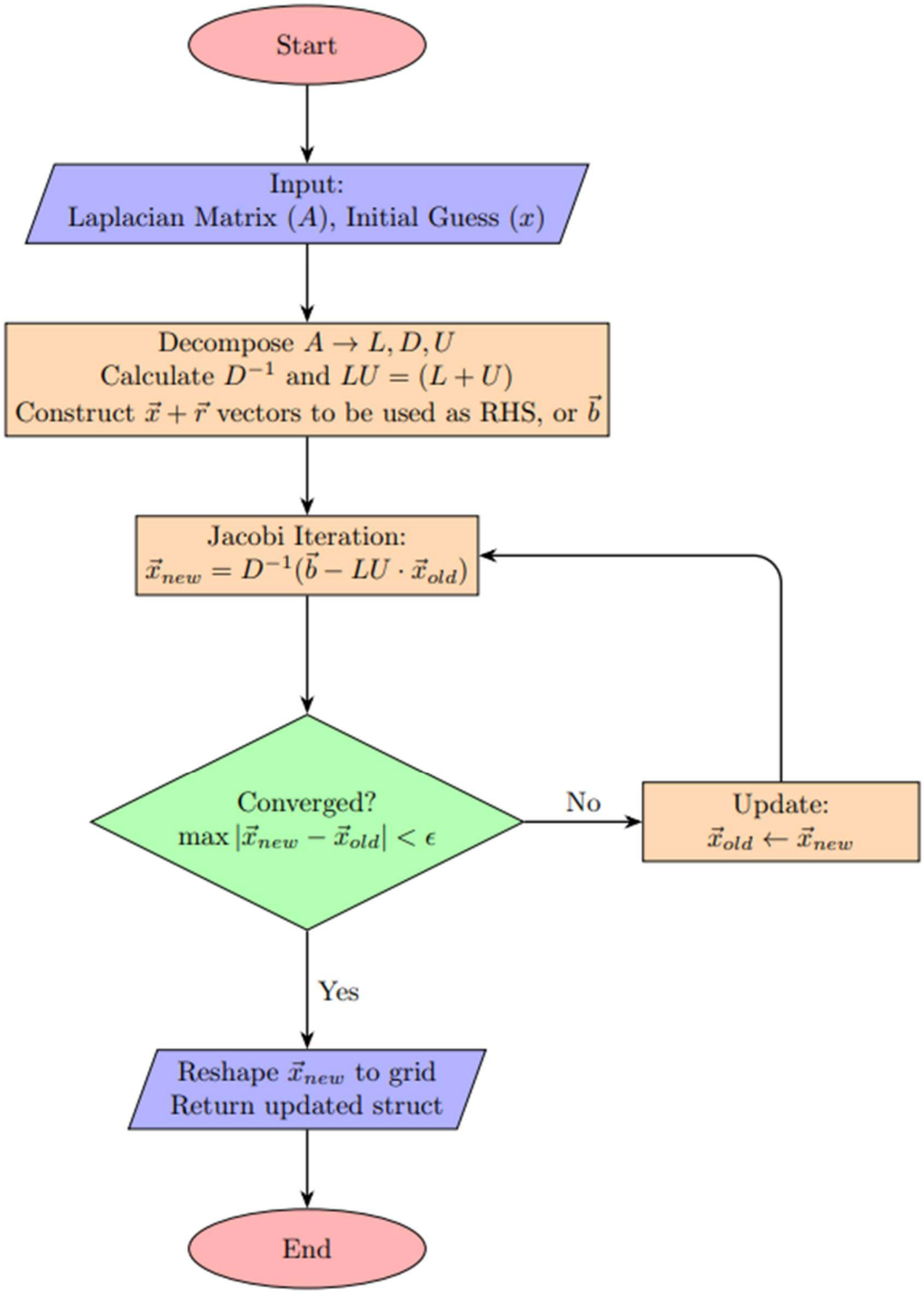
תרשים הזרימה עבור הפונקציה initial_mat_guess.m



תרשים הזרימה עבור הפונקציה vectorized_laplacian_mat.m:



תרשים הזרימה עבור הפונקציה Laplacian_solver_Jac.m




- תרשים הזרימה הנ"ל מציג את הלוגיקה הסטנדרטית לפתרון בשיטת יעקבי. עם זאת, במהלך המימוש בוצעה אופטימיזציה כך ש-LU מחושב ע"י $A - D$ ו-$D^{-1}$ מחושבת בפונקציה קודמת, כדי למקסם את היעילות של הקוד. לוגית, האלגוריתם המוצג

בתרשים הזרימה פועל; מעשית, יש הבדל זניח בינו לבין הפונקציה שמומשה בסופו של דבר.

**תוצאות:**

לתיקיית ה-7z מצורפים שני קבצי אקסל עם תוצאות של מספר הרצות של התוכנה.

כדי לבחון את ההרצות הרצתי סקריפט קצר היוצר וקטור של ההפרש בין האיבר המתאים במטריצה לממוצע השכנים שלו (מצרף אותו כקובץ נפרד), וצירפתי את השגיאה המקסימלית לכל מטריצת-חום (כולן בסדר הגודל של אפסילון, כרצוי).

**דיון ומסקנות:**

למרות התוכנית המוקדמת להשתמש בשיטת GS (ולמרות שיישמתי אותה בעזרת לולאה), ההעדפה הייתה לשימוש במטריצות ולא מצאתי דרך להשתמש במטריצות בלי שימוש בלולאות כדי ליישם את שיטת GS; במקרה כזה הפסד-הביצועים יהיה משמעותי יותר מהרווח הפוטנציאלי, בשל השימוש במאטלאב, שפה מפורשת עם "פירוש בזמן ריצה" ( JIT Compilation), ולכן מתאימה לשימוש בווקטורים הרבה יותר מלקריאות חוזרות לאיברי-מטריצה בלולאות.
זיהוי ה-RHS דרש ניתוח מתמטי של הבעיה, ועד שהושלם גרר התבדרות של הפונקציה. זיהוי המשוואה הלינארית של כל אלמנט הייתה פשוטה למדי, כשההרכבה שלה באמצעות מטריצות דלילות דרשה תיקונים חוזרים ונשנים לקוד, אבל הפכה אותו ליעיל מספיק כדי להיות מסוגל להריץ מטריצות גדולות פי 10 ויותר בקלות (הרצה של מטריצה של 200*200 דרשה בערך 8 שניות, והשגיאה הגרועה ביותר נשארה בסדר הגודל של אפסילון, אצרף הרצה של 10 מטריצות בגודל כזה עם זמני-הריצה).
ניתן לשפר משמעותית את הקוד אם נבחר ליישם גישה שונה לבעיה ולא נשתמש כלל במשוואות לינאריות, כי אם בחיבור וקטורים באופן מתאים למאטלאב, אבל בגדלי-המטריצות הנדרשים ההבדלים זניחים למדי.

# נספח:

## רשימת קבצים:

קבצי-מאטלאב:

*heat_calc.m* – הפונקציה הראשית, מקבלת את טמפ' השורה הראשונה והאחרונה של המטריצה, מאתחלת את כלל הפרמטרים של מטריצת-הטמפרטורה, קוראת לשלוש פונקציות הבאות כדי לצור את מטריצת-הטמפרטורה, את מטריצת המשוואות הלינאריות שמיישמות לפלסיאן, ואת את פונקציית הפתרון, שמשתמשת בכל אלה כדי להחזיר מטריצת-חום שעונה לתנאי הלפלסיאן.

*initial_mat_guess.m* – מקבלת את מבנה (*struct*) הפרמטרים והמשתנים (משתמשת בטמפ' הגבול השונות ובגודל מטריצת-הטמפרטורה) ומחזירה אותו אחרי שינוי (מוסיפה את מטריצת-הטמפרטורה): הפונקציה יוצרת לכל נקודה במטריצה חום ממוצע לפי מרחקה מהקירות השונים והטמפרטורה הידוע שלהם כדי לחסוך איטרציות.

*vectorized_laplacian_mat.m* – מקבלת את מבנה (*struct*) הפרמטרים והמשתנים (משתמשת בגודל מטריצת-הטמפרטורה ובמספר האלמנטים שייכלל בממוצע) ומחזירה אותו אחרי שינוי (מוסיפה את מטריצת-המקדמים של המשוואות הלינאריות שיגדירו את הלפלסיאן, את המטריצה $D^{-1}$ ואת המטריצה הבוליאנית המבחינה בין איברי-גבול לאיברי-פנים): הפונקציה מגדירה שני סוגים של איברים במטריצת-הטמפרטורה, איברי-גבול ואיברי-פנים, ויוצרת לכל איבר במטריצת-הטמפרטורה שורה במטריצת-המקדמים, לאיבר גבול זו שורת-הזהות, ולאיבר-פנים זו שורת לפלסיאן (4 באיבר *j* של השורה אם *j* הוא מספר השורה, ומינוס אחד בכל האיברים השכנים). הפונקציה יוצרת את המטריצה כמטריצה-דלילה, כדי לקצר זמני-חישוב ולחסוך מקום בזיכרון בהפעלות גדולות.

*laplacian_solver_Jac* - מקבלת את מבנה (*struct*) הפרמטרים והמשתנים (משתמשת במטריצת-הטמפרטורה ובמטריצה הבוליאנית) ומחזירה אותו אחרי שינוי (מתקנת את מטריצת-הטמפרטורה כך שתציית למשוואת הלפלסיאן): המטריצה יוצרת את $\vec{b}$ שתואר, מפצלת את המטריצה לאלכסונית וסכום המשולשות, ומשתמשת במטריצה האלכסונית אחרי היפוך לטובת לולאת *while* בה מבוצע צעד היעקובי, נבדקת הקרבה בין מטריצת-הטמפרטורה הקודמת לחדשה, ומעודכנת המטריצה לקראת האיטרציה הבאה. כשתנאי הקרבה מסופק, הפונקציה מחזירה את מטריצת-הטמפרטורה המעודכנת למבנה.

*max_error_script.m* - סקריפט בדיקה שנכתב בסיוע סוכן-*AI*. נועד לייצר הדגמות של הפונקציה עם שיעור שגיאה מקסימלי במטריצה המתקבלת ומשך זמן לביצוע.

אקסל:

*Heat_Calc_Results.xlsx* – תוצאות 10 הרצות של הפונקציה *heat_calc.m* עם משתני טמפ' שונים, ולצידם השגיאה המקסימלית בין ממוצע השכנים לערך בנקודה עבור כל הרצה וזמן הריצה.

*Heat_Calc_Results_200_matrices.m* – תוצאות 10 הרצות של הפונקציה *heat_calc.m* (אחרי שינוי כך שהפונקציה תבצע אותן פעולות על מטריצת-חום של 200*200) עם משתני טמפ' שונים, ולצידם השגיאה המקסימלית בין ממוצע השכנים לערך בנקודה עבור כל הרצה וזמן הריצה.

קבצי-*PDF*:

*Project2_solution.pdf* – חוברת מלווה לפרויקט.