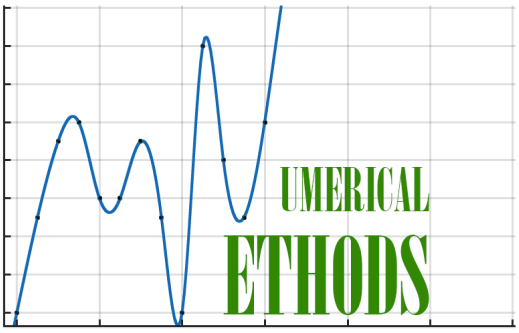


Tema de casă
Interpolare aplicată pe imagini

1 Introducere

Interpolarea este o metodă prin care se obțin valori intermediare aproximative ale unei funcții pentru care se cunosc doar o parte din valori.
O imagine poate fi interpretată ca o funcție $f : Z^2 \to R$, cu semnificația că $f(x, y)$ reprezintă intensitatea luminoasă a pixel-ului de la poziția dată de indicii x și y.
Observație: x și y sunt întregi! Un ecran fizic nu poate avea o jumătate de pixel.
Desigur, o singură funcție este suficientă doar pentru a descrie o imagine alb negru. O imagine colorată poate fi descrisă prin 3 astfel de funcții, fiecare asociată unei anumite culori, cel mai comun sistem de culori fiind roșu, verde și albastru (RGB).
Astfel, a aplica o interpolare pe o imagine înseamnă, de fapt, a aplica o interpolare pe o funcție de doi parametrii, cu mențiunea că într-o imagine finală ne vor interesa doar valorile de la coordonate întregi.

1.1 Aplicații

În practică, interpolarea aplicată pe imagini este folosită pentru:

1.1.1 Transformări geometrice

Atunci când se aplică o transformare geometrică asupra unei imagini (fie ea de scalare, de rotație, etc.), este posibil ca un pixel din imaginea finală să nu corespundă exact unui pixel din imaginea originală, adică poziția acestuia să nu fie un număr întreg. Interpolarea oferă o modalitate de a calcula valoarea luminoasă în acest punct în funcție de pixelii din jur.

1.1.2 Texture filtering

Texture filtering este un concept din grafica calculatoarelor și se ocupă de corespondența unei texturi (a unei imagini) pe un model 3D. Un pixel al modelului 3D nu va corespunde neapârat cu un pixel al texturii (numit texel). Lucrurile se complică ținând cont că modelul respectiv poate fi privit din orice poziție si din orice unghi. Astfel, se utilizează o interpolare pentru a afla culoarea unui pixel.

1.1.3 Image inpainting

O imagine poate fi ușor deteriorată, astfel încât părți din ea să lipsească. Image inpainting se ocupă cu "umplerea" părților lipsă din imagine pentru a aproxima imaginea corectă.

Figure 1: Imaginea originală si reparată.

2 Metode abordate

În cadrul temei, se vor implementa trei metode de interpolare aplicate pe imagini:

- Interpolare Proximală;
- Interpolare Bicubică;
- Interpolare Lanczos.

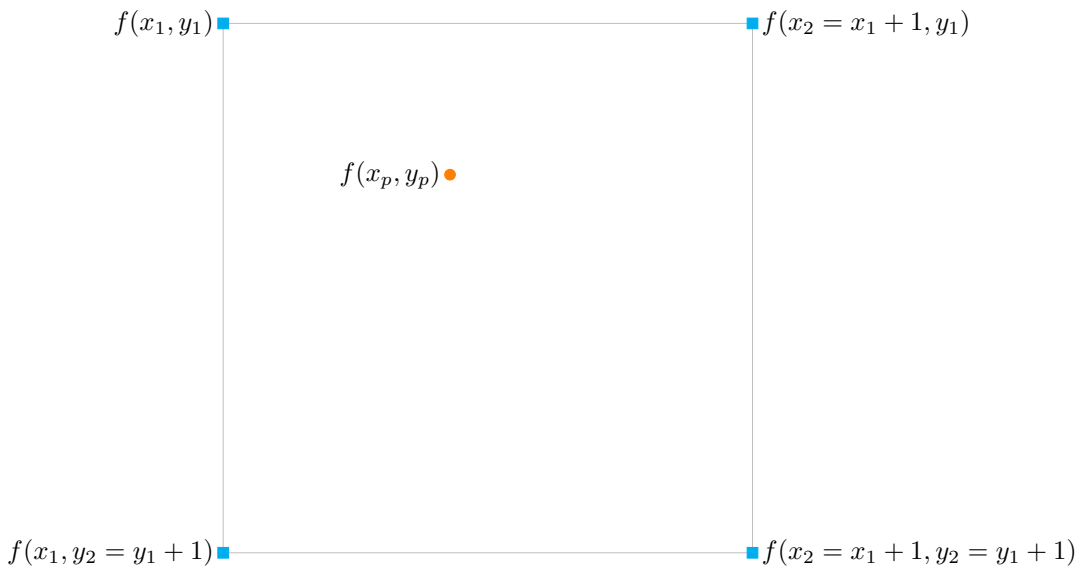
Concret, acestea vor fi folosite pentru a redimensiona și pentru a roti imagini.
Dorim să aplicăm o transformare geometrică, notată cu T, asupra unei imagini. Transformările geometrice sunt reprezentate în formă matriceală pentru că facilitează procesul de compunere.

În urma acestei transformări, unii pixeli nu o să mai aibă coordonate întregi, dar în imaginea finală adresarea pixelilor se face doar în astfel de coordonate întregi. Interpolarea oferă o modalitate de a calcula valorile pixelilor din imaginea finală pe baza celor din imaginea inițială.

Mai exact, se aplică următoarea procedură:

Pentru fiecare pixel (x_o, y_o) din imaginea finală, se aplica transformarea inversă T^{-1} pentru a trece în coordonatele imaginii inițiale și se obține punctul (x_p, y_p). În urma acestei transformări inverse, poziția rezultată nu o să mai fie neapărat întreagă, ci o sa aibă și o parte fracționară.

Se determină coordonatele celor patru pixeli din imaginea inițială ce înconjoară punctul (x_p, y_p):



Apoi, se va folosi una din metodele de interpolare menționate pentru a calcula $f(x_p, y_p)$ pe baza celor 4 puncte înconjurătoare, obținându-se astfel valoarea pixelului (x_o, y_o) din imaginea finală.

2.1 Redimensionare

Dorim să redimensionăm o imagine de dimensiune $m \times n$ astfel încât aceasta să devină de dimensiune $p \times q$ (cu p și q mai mari ca m, respectiv n). Aceasta se realizează printr-o transformare a sistemului de coordonate caracterizată de 2 constante de scalare:

$$s_x = \frac{q}{n} \qquad s_y = \frac{p}{m}$$

Astfel încât:

$$x_o = x_i * s_x \qquad y_o = y_i * s_y$$

O altă modalitate de a reprezenta transformarea de mai sus este:

$$\begin{bmatrix} x_o \\ y_o \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix}$$

Transformarea și inversa ei sunt:

$$T = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \qquad T^{-1} = \begin{bmatrix} \frac{1}{s_x} & 0 \\ 0 & \frac{1}{s_y} \end{bmatrix}$$

Observație pentru implementarea temei: în urma scalării, originea imaginii ar trebui să rămână nemișcată. În Octave, indexarea se face de la 1, iar originea imaginii se reprezintă prin coordonatele $(1, 1)$. Dacă se înmulțește acest vector cu transformarea de scalare, se va obține punctul (s_x, s_y), care nu mai reprezintă originea în imaginea finală. O soluție ar fi ca în operațiile cu transformate să se folosească coordonate indexate de la 0.

2.2 Rotație

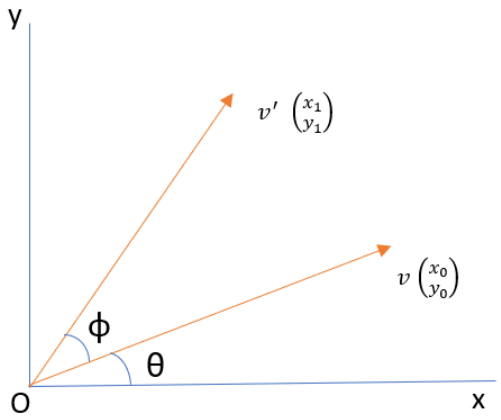


Figure 2: Rotirea unui vector 2D.

Pornind de la coordonatele polare ale varfurilor vectorilor v și v':

$$\begin{aligned} x_0 &= rcos\theta \\ y_0 &= rsin\theta \\ x_1 &= rcos(\theta+\phi) \\ y_1 &= rsin(\theta+\phi) \end{aligned}$$

Se deduc formulele pentru vectorul v':

$$\begin{aligned} x_1 &= r(cos\theta cos\phi - sin\theta sin\phi) = x_0 cos\phi - y_0 sin\phi \\ y_1 &= r(sin\theta cos\phi + cos\theta sin\phi) = y_0 cos\phi + x_0 sin\phi \end{aligned}$$

În forma matriceală:

$$\begin{bmatrix} cos\phi & -sin\phi \\ sin\phi & cos\phi \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}$$

Așadar, putem obține locația oricărui punct din plan, după o rotire la stânga cu ϕ radiani, înmulțind coordonatele punctului, la stânga, cu matricea de rotație:

$$T_{rot} = \begin{bmatrix} cos\phi & -sin\phi \\ sin\phi & cos\phi \end{bmatrix}$$

2.3 Imagine coloră

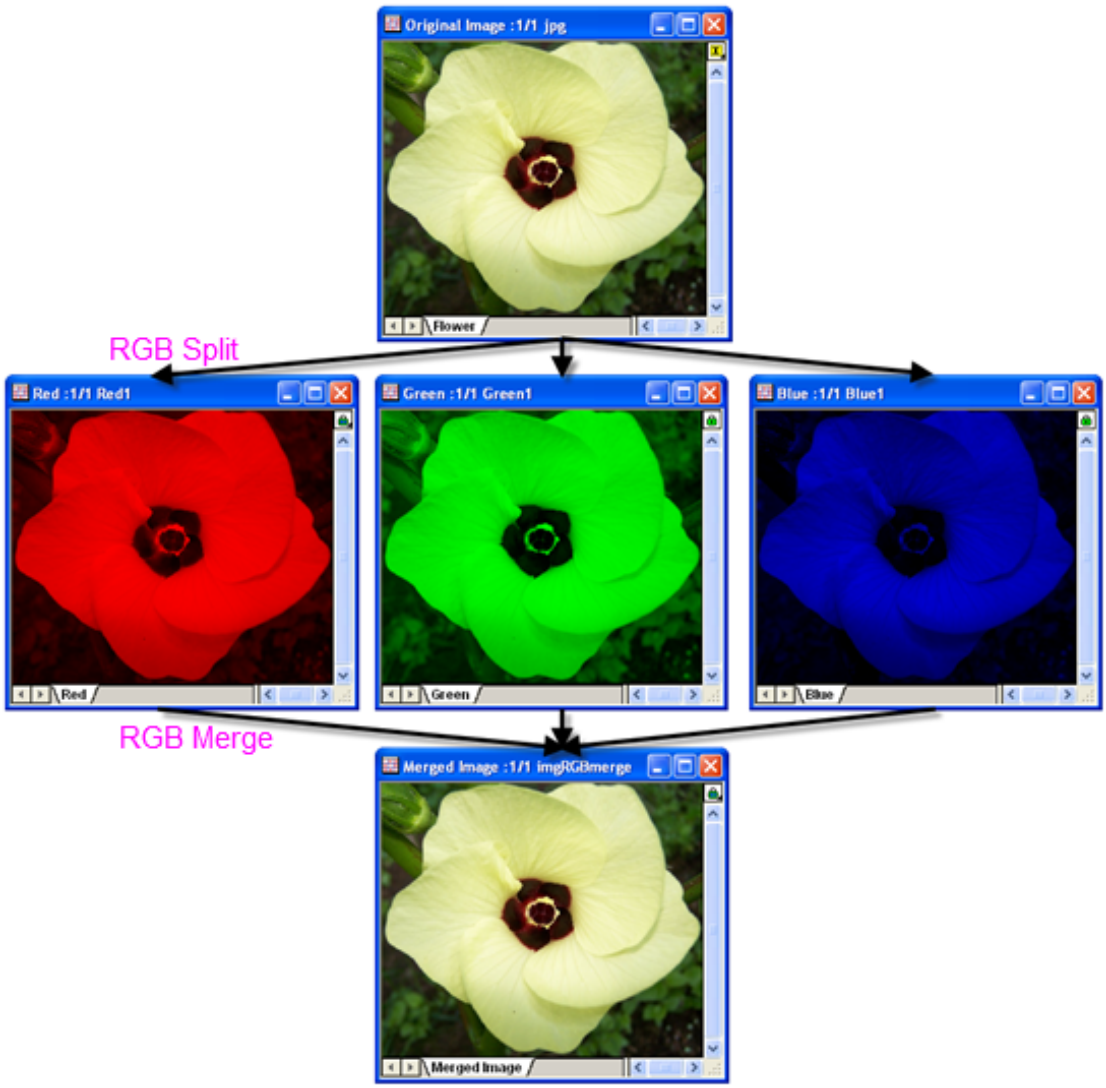


Figure 3: Separare imaginii în 3 canele

RGB (care reprezintă roșu, verde și albastru) este un model de culoare în care culorile roșu, verde și albastru sunt combinate în diferite moduri pentru a reproduce o gamă largă de culori. Modelul este folosit pentru a afișa imagini în sisteme electronice, cum ar fi televizoare și computere. Interpolarea spațiului de culoare în RGB utilizează adesea interpolările care vor fi prezentate de-a lungul temei.

3 Interpolare Proximală

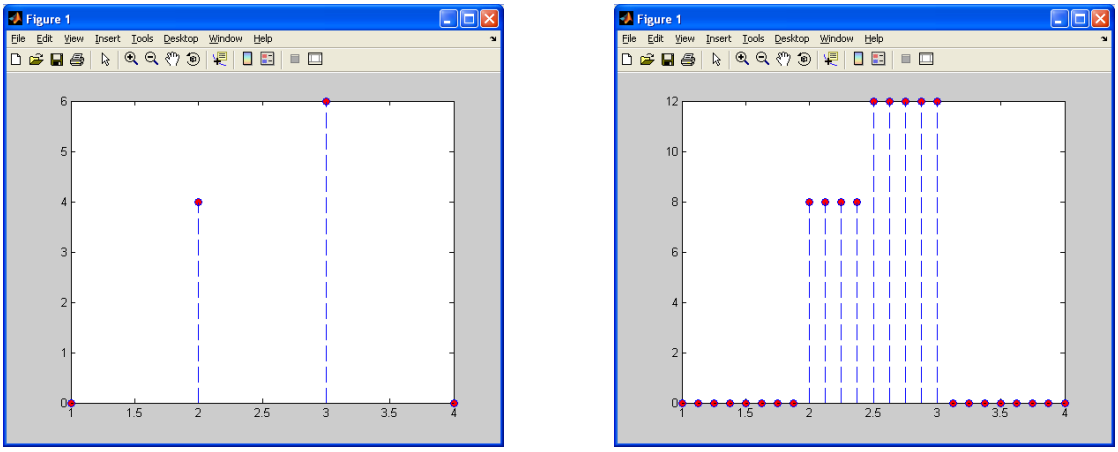


Figure 4: Exemplu interpolare 1D Proximală.

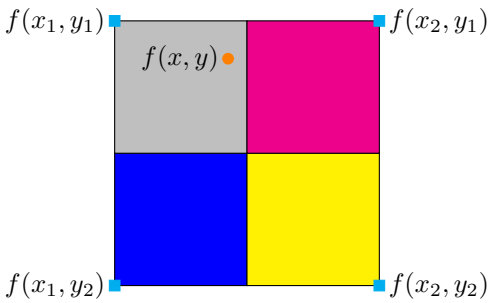
Se consideră un caz 1D, o funcție $f(x)$ cunoscută în doar câteva puncte. Metoda Proximală este cea mai simplă metodă de interpolare și funcționează astfel:

- se caută cel mai apropiat punct cunoscut;
- se aproximează valoarea funcției din acesta.

Observație
În cazul 2D, pentru fiecare pixel nou din imaginea rezultată, trebuie găsit cel mai apropiat pixel din imaginea sursă. Se fac două interpolări 1D pe fiecare dimensiune:

- pe axa Ox pentru a determina dacă cel mai apropiat punct este unul din punctele din stânga sau din dreapta;
- pe axa OY pentru a determina dacă este unul din punctele de sus sau de jos.

Rezultatul obținut va delimita 4 zone de replicare ale valorilor funcției.



Pentru rotirea unei imagini, se consideră forma funcției:

$$f(x, y) = a_0 + a_1 x + a_2 y + a_3 xy.$$

Se impune condiția ca funcția să dea valori în punctele cunoscute, obținându-se un sistem liniar de ecuații care poate fi rezolvat cu metodele studiate:

$$\begin{bmatrix} 1 & x_1 & y_1 & x_1y_1 \\ 1 & x_1 & y_2 & x_1y_2 \\ 1 & x_2 & y_1 & x_2y_1 \\ 1 & x_2 & y_2 & x_2y_2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} f(A_{11}) \\ f(A_{12}) \\ f(A_{21}) \\ f(A_{22}) \end{bmatrix}$$

3.1 Observații rezultate

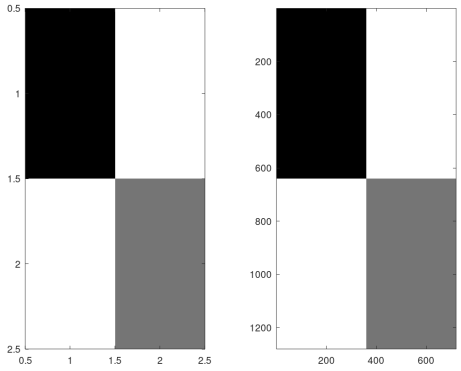
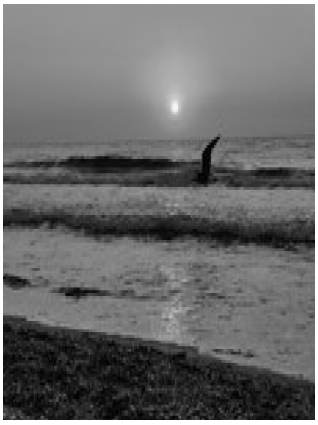


Figure 5: Rezultatul scalării folosind Interpolare Proximală.

(a) Imagine originală.



(b) Rezultatul interpolării.

Figure 6: Interpolare Proximală pe o imagine generică.

Pentru aceste exemple, rezultatul final este unul bun. Însă, din cauza faptului că funcția obținută nu este continuă, pentru o imagine obișnuită rezultatul arată destul de "pixelat". Astfel, Interpolarea Proximală nu are un rezultat dorit, însă imaginea obținută este un prim pas. Un alt rezultat vizibil, este trecerea la o scală mai mare (vezi Figura 4).

4 Interpolare Bicubică

Interpolarea Bicubică reprezintă extensia interpolării cu funcții spline cubice la două dimensiuni. Se consideră o funcție $f : [0,1] \times [0,1]$. Se cunosc valorile funcției și ale derivatelor sale ($\frac{\partial f}{\partial x} = f_x$, $\frac{\partial f}{\partial y} = f_y$, $\frac{\partial^2 f}{\partial x \partial y} = f_{xy}$) în punctele (0,0), (0,1), (1,0) și (1,1). Rezultatul Interpolării Bicubice este de forma: $f(x,y) = \sum_{i=0}^{3} \sum_{j=0}^{3} a_{ij} x^i y^j$.

Trebuie aflați cei 16 coeficienți a_{ij}. Din condiția ca funcția să aibă valorile cunoscute apar 4 ecuații:

$$f(0,0) = a_{00} \tag{1}$$
$$f(0,1) = a_{00} + a_{01} + a_{02} + a_{03} \tag{2}$$
$$f(1,0) = a_{00} + a_{10} + a_{20} + a_{30} \tag{3}$$
$$f(1,1) = \sum_{i=0}^{3} \sum_{j=0}^{3} a_{ij} \tag{4}$$

Asemănător, impunând valorile cunoscute ale derivatelor față de x și y apar 8 ecuații:

$$f_x(0,0) = a_{10} \tag{5}$$
$$f_x(0,1) = a_{10} + a_{11} + a_{12} + a_{13} \tag{6}$$
$$f_x(1,0) = a_{10} + 2a_{20} + 3a_{30} \tag{7}$$
$$f_x(1,1) = \sum_{i=0}^{3} \sum_{j=0}^{3} a_{ij} i \tag{8}$$
$$f_y(0,0) = a_{01} \tag{9}$$
$$f_y(0,1) = a_{01} + a_{11} + a_{21} + a_{31} \tag{10}$$
$$f_y(1,0) = a_{01} + 2a_{02} + 3a_{03} \tag{11}$$
$$f_y(1,1) = \sum_{i=0}^{3} \sum_{j=0}^{3} a_{ij} j \tag{12}$$

Iar pentru derivata mixtă:

$$f_{xy}(0,0) = a_{11} \tag{13}$$
$$f_{xy}(0,1) = a_{11} + 2a_{12} + 3a_{13} \tag{14}$$
$$f_{xy}(1,0) = a_{11} + 2a_{21} + 3a_{31} \tag{15}$$
$$f_{xy}(1,1) = \sum_{i=0}^{3} \sum_{j=0}^{3} a_{ij} ij \tag{16}$$

Aceste 16 ecuații formează un sistem de ecuații liniare ce poate fi rezolvat prin metodele studiate. Coeficienții a_{ij} pot fi grupați într-o matrice 4 x 4, iar sistemul de ecuații este următorul:

$$\begin{bmatrix} f(0,0) & f(0,1) & f_y(0,0) & f_y(0,1) \\ f(1,0) & f(1,1) & f_y(1,0) & f_y(1,1) \\ f_x(0,0) & f_x(0,1) & f_{xy}(0,0) & f_{xy}(0,1) \\ f_x(1,0) & f_x(1,1) & f_{xy}(1,0) & f_{xy}(1,1) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 2 \\ 0 & 1 & 0 & 3 \end{bmatrix}$$

Având soluția:

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & -1 \\ 2 & -2 & 1 & 1 \end{bmatrix} \begin{bmatrix} f(0,0) & f(0,1) & f_y(0,0) & f_y(0,1) \\ f(1,0) & f(1,1) & f_y(1,0) & f_y(1,1) \\ f_x(0,0) & f_x(0,1) & f_{xy}(0,0) & f_{xy}(0,1) \\ f_x(1,0) & f_x(1,1) & f_{xy}(1,0) & f_{xy}(1,1) \end{bmatrix} \begin{bmatrix} 1 & 0 & -3 & 2 \\ 0 & 0 & 3 & -2 \\ 0 & 1 & -2 & 1 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$

Funcția poate fi scrisă și astfel:

$$f(x,y) = \begin{bmatrix} 1 & x & x^2 & x^3 \end{bmatrix} \begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} 1 \\ y \\ y^2 \\ y^3 \end{bmatrix}$$

4.1 Aproximarea derivatelor când acestea nu sunt cunoscute

În practică, în lucrul cu imagini, valorile derivatelor în cele 4 puncte nu sunt cunoscute. Acestea sunt aproximate folosind diferențe finite.
Se consideră 16 puncte, organizate într-un pătrat 4 x 4. Față de punctele considerate anterior, se consideră și coordonatele $x, y = -1$ și $x, y = 2$ Astfel, se obțin:

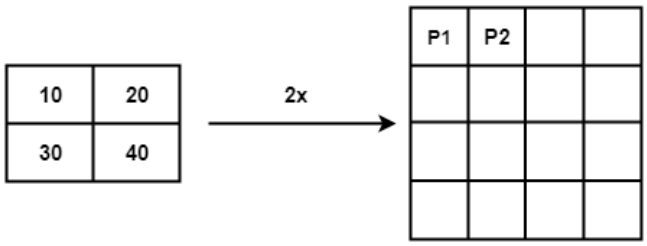
- $$f_x(x,y) = \frac{\partial f}{\partial x} = \frac{f(x+1,y) - f(x-1,y)}{2}$$
- $$f_y(x,y) = \frac{\partial f}{\partial y} = \frac{f(x,y+1) - f(x,y-1)}{2}$$
- $$f_{xy}(x,y) = \frac{\partial^2 f}{\partial x \partial y} = \frac{f_y(x+1,y) - f_y(x-1,y)}{2} =$$

$$\frac{f(x-1,y-1) + f(x+1,y+1) - f(x+1,y-1) - f(x-1,y+1)}{4}$$

La marginile imaginii se poate considera că derivatele sunt 0.

4.2 Exemplu concret

Luăm o imagine de dimensiune 2x2 și o vom scala cu un factor de 2 pentru a obține o imagine de dimensiune 4x4. Astfel, vom obține:



Centrăm matricea inițială, expandând fiecare element într-o matrcie de dimensiune 3x3. Se poate observa întregul proces în imaginile de mai jos. În prima imagine, am încadrat matricea 2x2.

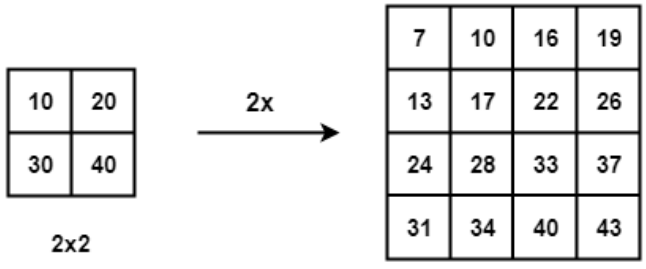
```
array([[10, 10, 10, 20, 20, 20],
       [10, 10, 10, 20, 20, 20],
       [10, 10, 10, 20, 20, 20],
       [30, 30, 30, 40, 40, 40],
       [30, 30, 30, 40, 40, 40],
       [30, 30, 30, 40, 40, 40]], dtype=uint8)
```

Mai jos, se calculează pentru primul element P = 10. Trebuie știută poziția sa pentru a realiza expansiunile.

```
array([[10, 10, 10, 20, 20, 20],
       [10, 10, 10, 20, 20, 20],
       [10, 10, 10, 20, 20, 20],
       [30, 30, 30, 40, 40, 40],
       [30, 30, 30, 40, 40, 40],
       [30, 30, 30, 40, 40, 40]], dtype=uint8)
```

P

Rezultatul final pe care îl obținem este ilustrat mai jos:



2x2

4x4

4.3 Observații rezultate

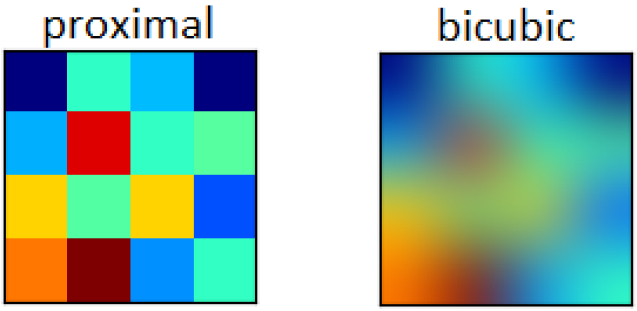


Figure 7: Comparație Interpolare Proximală și Bicubică

- Se observă că Interpolarea Bicubică pare să pastreze mai bine detaliile imaginii, în timp ce Interpolarea Proximală are tendința de a pierde muchiile bine definite.
- În Interpolarea Proximală, se observă că imaginea este foarte blurată, iar atunci când aplicăm Interpolarea Bicubică asupra unei imagini, aceasta capătă o formă mult mai atractivă.
- Dispar muchiile bine-definite, ele transformându-se în linii curbe continue.
- Acest algoritm este mai complicat și are un timp de execuție mai încet, însă este mai eficient.

Imaginea reală nu a fost obținută încă, ea urmând să fie descoperită cu ajutorul Interpolării Lanczos.

5 Interpolare Lanczos

Până în acest punct, am observat interpolarea imaginilor cu ajutorul interpolărilor:

- Proximală, în care s-au aproximat pixelii pentru a obține o valoare medie a imaginii;
- Bicubică, în care s-au aproximat pixelii pentru a obține o valoare medie a imaginii, iar pixelii centrali și-au păstrat valoarea.

Ne propunem să prezentăm o nouă interpolarea ce are o mare aplicabilitate în domeniul științific. Metodele bazate pe potrivirea momentelor multipunct sunt considerate metode puternice pentru problemele de reducere a ordinii modelului. Ele sunt legate de subspații Krylov raționale (clasice sau bloc) și se bazează pe selecția unor puncte de interpolare care reprezintă problema majoră pentru aceste metode. În această lucrare, este propus și aplicat un algoritm de tip bloc rațional adaptiv de tip Lanczos pentru reducerea ordinii de model a sistemelor dinamice independente de timp liniar multi-intrare și multi-ieșire.
Luăm în considerare următorul sistem dinamic cu mai multe intrări și mai multe ieșiri (MIMO) invariant în timp liniar (LTI):

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t), \\ y(t) = Cx(t) \end{cases} \tag{1}$$

unde $x(t)^n$ este vectorul de stare, $u(t)^n, y(t)^p$ sunt vectorii de intrare, respectiv de ieșire ai sistemului (1). Matricele sistemului $B, C^T \in R^{np}$ $și$ $A \in R^{nn}$ se presupune că au dimensiuni mari și rare. Funcția de transfer a sistemului original (1) este definită de:

$$H(s) = C(sI_n - A)^{-1}B.$$

Apoi, scopul problemei de reducere a ordinii modelului este de a produce un sistem dimensional de ordin inferior care păstrează proprietățile importante ale sistemului original și are următoarea formă:

$$\begin{cases} \dot{x}_r(t) = A_r x_r(t) + B_r u(t) \\ y_r(t) = C_r x_r(t), \end{cases} \tag{2}$$

unde $A_r \in \mathbb{R}^{r\times r}$, $B_r, C_r^T \in \mathbb{R}^{r\times p}$, iar r este mult mai mic decât n. Funcția de transfer de ordin inferior asociată este notată cu:

$$H_r(s) = C_r(sI_r - A_r)^{-1}B_r.$$

În ultimii ani, au fost explorate diverse metode de reducere a modelelor pentru sistemele MIMO. Unele dintre ele se bazează pe metodele subspațiale Krylov (potrivirea momentelor), în timp ce altele folosesc trunchierea echilibrată. În special, procesul Lanczos a fost utilizat pentru o singură intrare și o singură ieșire (SISO) (cazul p = 1) și sisteme dinamice MIMO. Versiunea standard a algoritmului Lanczos construiește modele de ordin redus care aproximează slab o anumită dinamică a frecvenței și, pentru a depăși această problemă, în ultimii ani au fost dezvoltate metode raționale subspațiale Krylov. Cu toate acestea, selectarea punctelor de interpolare este o problemă majoră a acestor metode și ele trebuie alese corespunzător pentru a obține aproximări bune. Când p > 1, putem folosi metode subspațiale Krylov bazate pe versiuni bloc ale algoritmilor Arnoldi sau Lanczos. Suntem interesați de metoda de interpolare rațională în mai multe

puncte pentru reducerea ordinii de model pentru sistemele MIMO. Prin interpolare rațională în mai multe puncte înțelegem că sistemul redus se potrivește cu momentele sistemului original la mai multe puncte de interpolare. Una dintre problemele principale este selectarea schimburilor potrivite pentru a garanta o bună convergență a procesului. În literatura de specialitate au fost propuse diferite metode pentru a construi setul de puncte de interpolare. A fost propus un algoritm iterativ rațional Krylov (IRKA) pentru a calcula un model de ordin redus care satisface condițiile de ordinul întâi pentru H2 apropiere.

5.1 Probleme de potrivire a momentelor

Fie $H(s) = C(sI_n - A)^{-1}B$ funcția de transfer a sistemului dinamic liniar descris de sistemul (1). H(s) poate fi descris ca o serie Taylor:

$$H(s) = h_0 + h_1(s - \sigma_0) + h_2(s - \sigma_0)^2 + ...$$

unde $h_j(\sigma_0) = C(\sigma_0 I_n - A)^{-(j+1)}B.$

Aceste momente sunt valorile funcției de transfer a sistemului (1) și derivatele sale evaluate la σ_0 (se mai numesc și momentele deplasate). Problema reducerii ordinului modelului folosind o metodă de potrivire a momentelor constă în găsirea unei funcții de transfer de ordin inferior $H_r(s)$ având o extindere a seriei de putere la σ_0 la fel ca:

$H_r(s) = \hat{h}_0 + \hat{h}_1(s - \sigma_0) + \hat{h}_2(s - \sigma_0)^2 + ...$ astfel încât primele k momente sunt potrivite, adică:

$h_j(\sigma_0) = \hat{h}_j(\sigma_0),\ j = 0, ..., k - 1$ pentru $k << n$.

Modelul de ordin redus rezultat este cunoscut ca o interpolare rațională [1]. Dacă $\sigma_0 = 0$, momentele satisfac $h_j = CA^{(j+1)}B$ pentru $j \geq 0$, iar problema este cunoscută ca o aproximare Padé. Seria Laurent a funcției de transfer H în jurul $\sigma_0 =$ Inf este exprimată ca:

$$H(s) = \sum_{i=0}^{\infty} h_i s^{-i}$$

unde $h_i = CA^iB$, pentru orice i ≥ 0

sunt numiți parametrii Markov și problema corespunzătoare este cunoscută ca o realizare parțială. De asemenea, putem lua în considerare mai multe puncte de interpolare, modelul de ordin redus rezultat este o aproximare Padé multipunct sau o interpolare rațională multipunct.

5.2 Algoritmul Standard Block Lanczos

Fie V și W două blocuri inițiale ale R^{nxp}, și considerăm următorul bloc de subspații Krylov:

$$\begin{aligned}\mathscr{K}_m(A, V) &= \text{Range}\left(V, AV, \ldots, A^{m-1}V\right) \text{ and } \mathscr{K}_m(A^T, W) \\ &= \text{Range}\left(W, \ldots, (A^T)^{(m-1)}W\right).\end{aligned}$$

Algoritmul Bloc Nesimetric Lanczos aplicat perechilor (A, V) si(A^T, W) generează două secvențe de matrici bi-ortonormale n × p V_i și W_j astfel încât:

$$\mathcal{K}_m(A, V) = \text{Range}(V_1, V_2, \ldots, V_m).$$

$$\mathcal{K}_m(A^T, W) = \text{Range}(W_1, W_2, \ldots, W_m).$$

Matricele V_i și W_j care sunt generate de Algoritmul bloc Lanczos satisfac condițiile de biortogonalitate, adică:

$$\begin{cases} W_j^T V_i = 0_p, & if\ i \neq j, \\ W_j^T V_i = I_p, & if\ i = j. \end{cases}$$

În continuare, oferim o versiune stabilă a Algoritmului bloc nesimetric Lanczos:

Algorithm 1. The nonsymmetric block Lanczos algorithm

- **Inputs:** $A \in \mathbb{R}^{n\times n}$, $V, W \in \mathbb{R}^{n\times p}$ and m an integer.

1. **Compute the QR decomposition of $W^T V$, i.e., $W^T V = \delta\beta$;**
   $V_1 = V\beta^{-1}$; $W_1 = W\delta$; $\widetilde{V}_2 = AV_1$; $\widetilde{W}_2 = A^T W_1$;

2. **For** $j = 1, \ldots, m$
   $\alpha_j = W_j^T \widetilde{V}_{j+1}$; $\widetilde{V}_{j+1} = \widetilde{V}_{j+1} - V_j\alpha_j$; $\widetilde{W}_{j+1} = \widetilde{W}_{j+1} - W_j\alpha_j^T$;
   Compute the QR decomposition of $\widetilde{V}_{j+1}$ and $\widetilde{W}_{j+1}$, i.e.,
   $\widetilde{V}_{j+1} = V_{j+1}\beta_{j+1}$; $\widetilde{W}_{j+1} = W_{j+1}\delta_{j+1}^T$;
   Compute the singular value decomposition of $W_{j+1}^T V_{j+1}$, i.e.,
   $W_{j+1}^T V_{j+1} = U_j \Sigma_j Z_j^T$;
   $\delta_{j+1} = \delta_{j+1} U_j \Sigma_j^{1/2}$; $\beta_{j+1} = \Sigma_j^{1/2} Z_j^T \beta_{j+1}$;
   $V_{j+1} = V_{j+1} Z_j \Sigma_j^{-1/2}$; $W_{j+1} = W_{j+1} U_j \Sigma_j^{-1/2}$;
   $\widetilde{V}_{j+2} = AV_{j+1} - V_j\delta_{j+1}$; $\widetilde{W}_{j+2} = A^T W_{j+1} - W_j\beta_{j+1}^T$;
3. **end For**.

Fie spațiile vectoriale $\mathbb{V}_m = [V_1, V_2, \ldots, V_m]$ și $\mathbb{W}_m = [W_1, W_2, \ldots, W_m]$. Atunci vom avea următoarele relații între blocurile Lanczos:

$$A\mathbb{V}_m = \mathbb{V}_m T_m + V_{m+1}\beta_{m+1}E_m^T, \quad A^T\mathbb{W}_m = \mathbb{W}_m T_m^T + W_{m+1}\delta_{m+1}^T E_m^T$$

unde E_m este ultimul bloc de dimensiune mp x p din matricea de identitate I_{mp} și T_m este blocul matrice tridiagonală de dimensiune mp x mp, definit de:

$$T_m = \begin{pmatrix} \alpha_1 & \delta_2 & & \\ \beta_2 & \alpha_2 & . & \\ & . & . & . \\ & & . & . & \delta_m \\ & & & \beta_m & \alpha_m \end{pmatrix}.$$

Vom obține următorul sistem:

$$A_m = \mathbb{W}_m^T A \mathbb{V}_m, \; B_m = \mathbb{W}_m^T B \text{ and } C_m = C\mathbb{V}_m. \tag{4}$$

5.3 Algoritmul Rational Block Lanczos

Procedura Lanczos de bloc rațional este un algoritm pentru construirea bazelor bi-ortonormale ale uniunii subspațiilor bloc Krylov. Următoarea teoremă este prezentată pentru sistemele SISO și este extinsă la cazul MIMO. Aceasta arată cum putem construi bazele biortogonale $V_1, V_2, \ldots, V_m$ și $W_1, W_2, \ldots, W_m$ a subspațiilor raționale Krylov $Range((A-\sigma_1 I_n)^{-1}B, \ldots, (A-\sigma_m I_n)^{-1}B)$ și $Range((A-\sigma_1 I_n)^{-T}C^T, \ldots, (A-\sigma_m I_n)^{-T}C^T)$, astfel încât problema interpolării raționale în mai multe puncte să fie rezolvată, adică modelul de ordin redus trebuie să interpoleze funcția de transfer inițială H(s) și prima sa derivată la punctele de interpolare.

Teorema 1

Fie H(s) funcția de transfer și m punte de interpolare, ce verifică relația $\sigma_i \neq \sigma_j$ pentru orice i $\neq j$. *Fie* $\mathbb{V}_m \in \mathbb{R}^{n\times mp}$ *și* $\mathbb{W}_m \in \mathbb{R}^{n\times mp}$, *ce pot fi obținute după cum urmează* :

$$\begin{aligned} \text{Range}(\mathbb{V}_m) &= \text{Range}\left\{(A-\sigma_1 I_n)^{-1}B, \ldots, (A-\sigma_m I_n)^{-1}B\right\} \\ \text{Range}(\mathbb{W}_m) &= \text{Range}\left\{(A-\sigma_1 I_n)^{-T}C^T, \ldots, (A-\sigma_m I_n)^{-T}C^T\right\} \end{aligned} \tag{5}$$

, unde: $\mathbb{W}_m^T \mathbb{V}_m = I_{mp}$.

Algoritmul bloc rațional de tip Lanczos este definit după cum urmează:

Algorithm 2. The rational block Lanczos-type algorithm

1. **Input:** $A \in \mathbb{R}^{n\times n}$, B, $C^T \in \mathbb{R}^{n\times p}$.
2. **Output:** two biorthogonal matrices $\mathbb{V}_{m+1}$ and $\mathbb{W}_{m+1}$ of $\mathbb{R}^{n\times(m+1)p}$.
   function $[\mathbb{V}_m, \mathbb{W}_m]$ =**Rational-Block-Lanczos**(A,B,C,$\{\sigma_1, \ldots, \sigma_m\}$)
3. **Set** $S_0 = (A-\sigma_1 I_n)^{-1}B$ and $R_0 = (A-\sigma_1 I_n)^{-T}C^T$
4. **Set** $S_0 = V_1 H_{1,0}$ and $R_0 = W_1 G_{1,0}$ such that $W_1^T V_1 = I_p$;
5. **Initialize:** $\mathbb{V}_1 = [V_1]$ and $\mathbb{W}_1 = [W_1]$.
6. **For** $k = 1, \ldots, m$
7. **if** $(k < m)$
8. **if** $\{\sigma_{k+1} = \infty\}$; $S_k = AV_k$ and $R_k = A^T W_k$; **else**
9. $S_k = (A-\sigma_{k+1}I_n)^{-1}V_k$ and $R_k = (A-\sigma_{k+1}I_n)^{-T}W_k$; **endif**
10. $H_k = \mathbb{W}_k^T S_k$ and $G_k = \mathbb{V}_k^T R_k$;
11. $S_k = S_k - \mathbb{V}_k H_k$ and $R_k = R_k - \mathbb{W}_k G_k$;
12. $S_k = V_{k+1}H_{k+1,k}$ and $R_k = W_{k+1}G_{k+1,k}$; (QR factorization)
13. $W_{k+1}^T V_{k+1} = P_k D_k Q_k^T$; (Singular Value Decomposition)
14. $V_{k+1} = V_{k+1}Q_k D_k^{-1/2}$ and $W_{k+1} = W_{k+1}P_k D_k^{-1/2}$;
15. $H_{k+1,k} = D_k^{1/2}Q_k^T H_{k+1,k}$ and $G_{k+1,k} = D_k^{1/2}P_k^T G_{k+1,k}$;
16. $\mathbb{V}_{k+1} = [\mathbb{V}_k, V_{k+1}]$; $\mathbb{W}_{k+1} = [\mathbb{W}_k, W_{k+1}]$;
17. **else**
18. **if** $\{\sigma_{m+1} = \infty\}$; $S_m = AB$ and $R_m = A^T C$; **else**
19. $S_m = A^{-1}B$ and $R_m = A^{-T}C^T$; **endif**
20. $H_m = \mathbb{W}_m^T S_m$ and $G_m = \mathbb{V}_m^T R_m$;
21. $S_m = S_m - \mathbb{V}_m H_m$ and $R_m = R_m - \mathbb{W}_m G_m$;
22. $S_m = V_{m+1}H_{m+1,m}$ and $R_m = W_{m+1}G_{m+1,m}$; (QR factorization)
23. $W_{m+1}^T V_{m+1} = P_m D_m Q_m^T$; (Singular Value Decomposition)
24. $V_{m+1} = V_{m+1}Q_m D_m^{-1/2}$ and $W_{m+1} = W_{m+1}P_m D_m^{-1/2}$;

25. $H_{m+1,m} = D_m^{1/2} Q_m^T H_{m+1,m}$ and $G_{m+1,m} = D_m^{1/2} P_m^T G_{m+1,m}$;
26. $\mathbb{V}_{m+1} = [\mathbb{V}_m, V_{m+1}]$; $\mathbb{W}_{m+1} = [\mathbb{W}_m, W_{m+1}]$;
27. **endif**
28. **endFor**.

Teorema 2
Considerăm V_{m+1} si W_{m+1} matricele generate de Algoritmul 2. Există blocurile superior Hessenberg de dimensiune (m + 1) p x mp H, G, K și L astfel încât să fie îndeplinite următoarele relații pentru subspațiile Krylov:

$$A\mathbb{V}_{m+1}\widetilde{\mathbb{H}}_m = \mathbb{V}_{m+1}\widetilde{\mathbb{K}}_m \tag{9}$$
$$A^T\mathbb{W}_{m+1}\widetilde{\mathbb{G}}_m = \mathbb{W}_{m+1}\widetilde{\mathbb{L}}_m, \tag{10}$$
$$\mathbb{H}_m = \mathbb{W}_m^T A^{-1}\mathbb{V}_m\mathbb{K}_m, \tag{11}$$
$$\mathbb{G}_m = \mathbb{V}_m^T A^{-T}\mathbb{W}_m\mathbb{L}_m, \tag{12}$$

5.4 O estimare a erorilor pentru funcția de transfer

Calculul erorii exacte a matricei de transfer între sistemul original și sistemul redus:

$$\varepsilon(s) = H(s) - H_m(s) \tag{21}$$

este important pentru măsurarea acurateței modelului de ordin redus rezultat.
Din păcate, eroarea exactă $\epsilon(s)$ nu este disponibilă, deoarece dimensiunea mai mare a sistemului original face ca calculul lui H(s) să fie foarte dificil. Pentru a remedia această situație, în literatura de specialitate au fost explorate diverse abordări pentru estimarea erorii (21).
Grimme a propus calculul erorii de modelare în termeni a doi vectori reziduali în cazul sistemelor cu o singură intrare și cu o singură ieșire. Rezultatul este extins aici la cazul cu mai multe intrări și mai multe ieșiri. Fie:

$$\begin{cases} R_B(s) = B - (sI_n - A)\mathbb{V}_m\tilde{X}_B(s), \\ R_C(s) = C^T - (sI_n - A)^T\mathbb{W}_m\tilde{X}_C(s) \end{cases}$$

unde $X_B(s)$ si $X_C(s)$ sunt solutiile sistemului:

$$\begin{cases} (sI_{mp} - A_m)\tilde{X}_B(s) = B_m, \\ (sI_{mp} - A_m)^T\tilde{X}_C(s) = C_m^T, \end{cases}$$

Teorema 3
Eroarea dintre răspunsurile în frecvență ale sistemelor originale și de ordin redus poate fi exprimată ca:

$$\varepsilon(s) = R_C^T(s)(sI_n - A)^{-1}R_B(s). \tag{22}$$

Teorema 4
Fie V_m șiW_m matricele calculate folosind algoritmul Lanczos bloc rațional. Dacă $(sI_n - A)$ și $(sI_{mp} - A_m)$ sunt nesingulare, atunci

$$H(s) - H_m(s) = C(sI_n - A)^{-1}(\mathbb{V}_m\mathbb{W}_m^T - I_n)AV_{m+1}H_{m+1,m}E_m^T\mathbb{H}_m^{-1}(sI_{mp} - A_m)^{-1}B_m.$$

5.5 Un algoritm de tip Lanczos de bloc rațional de ordin adaptiv

Reducerea ordinului modelului folosind interpolarea rațională multipunct oferă, în general, un model de ordin redus mai precis decât interpolarea în jurul unui singur punct. Din păcate, selectarea punctelor de interpolare nu este un proces automat și necesită o alegere adecvată pentru subspațiul Krylov mai precis. A fost propus algoritmul iterativ rațional Krylov (IRKA) în contextul H2-reducerea optimă a ordinii modelului prin utilizarea unui mod specific de alegere a punctelor de interpolare σ_i, $i = 1, \ldots, m$.
Pornind de la un set inițial de puncte de interpolare, se determină un sistem de ordin redus și se alege un nou set de puncte de interpolare ca valori Ritz $\lambda_i(A_m)$, i = 1 , ... , m, unde $\lambda_i(A_m)$ sunt valorile proprii ale A_m. Procesul continuă până când valorile Ritz din modelele consecutive de ordin redus stagnează.
Mai sus au fost propuse câteva tehnici de alegere a punctelor bune de interpolare. Scopul acestor metode este construirea următorului punct de interpolare la fiecare pas și se bazează pe ideea că deplasările trebuie selectate astfel încât norma de aproximare a erorilor să fie minimizată la fiecare iterație. Aici, se propune o abordare adaptativă prin utilizarea următoarei expresii de aproximare a erorilor:

$$\hat{\varepsilon}(s) = \tilde{R}_C^T(s)\tilde{R}_B(s).$$

Apoi $\sigma_k \in R$ este selectată ca:

$$\tilde{\sigma}_{k+1} = \arg\max_{s\in S} \|\tilde{R}_C^T(s)\tilde{R}_B(s)\|_2, \qquad (32)$$

dar daca este numar complex, partea sa reala este considarată ca următorul punct de interpolare.

Algorithm 3. The Adaptive Order Rational Block Lanczos-type (AORBL) algorithm for model-order reduction

1. **Input:** The original system (A,B,C), the initial values σ_1, σ_2, choose a tolerance tol and set $H_0 = I_p$.
2. **Output:** The reduced system (A_m, B_m, C_m).
3. Define $\hat{\varepsilon} = 1$ and $m = 1$.
4. **While** ($\hat{\varepsilon} > tol$) **do**
5. $[\mathbb{V}_m, \mathbb{W}_m]$ = **Rational-Block-Lanczos**(A,B,C).
6. Compute the reduced model $A_m = \mathbb{W}_m^T A\mathbb{V}_m$, $B_m = \mathbb{W}_m^T B$, $C_m = C\mathbb{V}_m$ and the corresponding transfer function H_m.
7. Compute the error estimation $\hat{\varepsilon} = \|H_m - H_{m-1}\|_\infty$.
8. Set $m = m + 1$.
9. **end while**.

6 Cerințe și punctaj

README - 0.5p

6.1 Interpolare Proximală - 3 p

Să se implementeze următoarele funcții folosind interpolarea nearest-neighbour:

- proximal_2x2(f, STEP) care aplică Interpolarea Proximală pe o imagine alb-negru f 2x2 cu puncte intermediare echidistante, având între ele distanța dată de STEP;
- proximal_2x2_RGB(f, STEP) care realizează același lucru, dar pentru o imagine RGB;
- proximal_resize(I, p, q) care redimensionează imaginea alb-negru I de dimensiune m x n astfel încât aceasta să aibă dimensiunea p x q;
- proximal_resize_RGB(I, p, q) care realizează același lucru, dar pentru o imagine RGB ;
- proximal_rotate(I, angle) care rotește o imagine alb-negru I cu unghiul dat;
- proximal_rotate_RGB(I, angle) care realizează același lucru, dar pentru o imagine RGB.

6.2 Interpolare Bicubică - 3.5 p

Să se implementeze următoarele funcții folosind interpolarea bicubică:

- precalc_d(I) care precalculează derivatele dx, dy, dxy în fiecare punct al imaginii folosind diferențe finite
- bicubic_coef(f, Ix, Iy, Ixy, x1, y1, x2, y2) care calculează matricea A de coeficienți de interpolare bicubică între cele 4 puncte, primind la intrare matricile cu derivate precalculate
- bicubic_resize(I, p, q) care redimensionează imaginea alb-negru I de dimensiune m x n a.î. aceasta să aibă dimensiunea p x q
- bicubic_resize_RGB(I, p, q) care realizează același lucru dar pentru o imagine RGB

6.3 Interpolare Lanczos - 4 p

Să se implementeze următorii algoritmi:

- Algoritmului bloc nesimetric Lanczos
- Algoritmul bloc rațional de tip Lanczos
- Algoritmul Lanczos de bloc rațional de ordin adaptiv pentru calculul sistemului de ordin redus.

Pentru temă veți avea la dispoziție un schelet de cod și un checker pentru verificare automată.

7 Referințe

1. https://graphicdesign.stackexchange.com/questions/26385/difference-between-none-linear-cubic
2. https://software.intel.com/content/www/us/en/develop/documentation/ipp-dev-reference/top/volume-2-image-processing/ipp-ref-interpolation-in-image-geometry-transform/lanczos-interpolation.html
3. https://en.wikipedia.org/wiki/Lanczos_algorithm
4. https://theailearner.com/2018/12/29/image-processing-bicubic-interpolation/
5. https://en.wikipedia.org/wiki/Lanczos_resampling
6. https://stackoverflow.com/questions/34198553/lanczos-interpolation-in-c
7. https://www.mathworks.com/help/visionhdl/ug/image-downsize.html
8. http://cmbbe2018.tecnico.ulisboa.pt/pen_cmbbe2018/pdf/WEB_PAPERS/CMBBE2018_paper_28.pdf
9. https://www.scss.tcd.ie/~koidlk/cs4062/ExampleKNN.pdf